كلية العلوم و التقنيات فاس
ⵜⴰⵖⵉⵡⴰⵏⵜ ⵏ ⵜⵎⴰⵙⵙⴰⵏⵉⵏ ⴷ ⵜⵙⵏⵉⴳⵉⵜⵉⵏ
Faculté des Sciences et Techniques de Fès

جامعة سيدي محمد بن عبد الله
ⵜⴰⵙⴷⴰⵡⵉⵜ ⵙⵉⴷⵉ ⵎⵓⵃⵎⵎⴰⴷ ⴱⵏ ⵄⴱⴷⵓⵍⵍⴰⵀ
Université Sidi Mohamed Ben Abdellah

Centre d'Etudes Doctorales : Sciences et Techniques de l'Ingénieur

N° d'ordre 55/2021

THESE DE DOCTORAT

Présentée par

Mlle : CHAIMAE OUCHICHA

Spécialité : Mathématiques Appliquées / Informatique

Sujet de la thèse : Segmentation et classification en imagerie médicale pour le diagnostic et le traitement de maladies neurologiques et infectieuses : cas d'Alzheimer et du Covid-19

Thèse présentée et soutenue le mercredi 14 juillet 2021 à 10h au centre de conférences devant le jury composé de

Nom Prénom	Etablissement	
Pr. Mohamed Lamrini	Faculté des Sciences Dhar El Mahraz, Fès	Président
Pr. Ahmed Drissi El Maliani	Université Mohammed V, Rabat	Rapporteur
Pr. Noureddine En Nahnahi	Faculté des Sciences Dhar El Mahraz, Fès	Rapporteur
Pr. Brahim Ouhbi	Ecole Nationale Supérieure des Arts et des Métiers, Meknès	Rapporteur
Pr. Fatima Ezzaki	Faculté des Sciences et Techniques, Fès	Examinateur
Pr. Mohammed Meknassi	Faculté des Sciences Dhar El Mahraz, Fès	Invité d'honneur
Pr. Ouafae Ammor	Faculté des Sciences et Techniques, Fès	Directeur de thèse

Laboratoire d'accueil : Modélisation et Structures Mathématiques.

Etablissement : Faculté des Sciences et Techniques de Fès

Résumé de la thèse

L'imagerie médicale est un outil essentiel dans de nombreux domaines d'applications médicales, utilisé à la fois pour le diagnostic et le traitement. Cependant, la lecture d'images médicales et l'établissement de recommandations de diagnostic ou de traitement nécessitent des médecins spécialistes. La pratique actuelle de lecture d'images médicales est laborieuse, chronophage, coûteuse et sujette aux erreurs. Il serait plus souhaitable de disposer d'un système assisté par ordinateur capable de faire automatiquement des recommandations de diagnostic et de traitement. Les progrès de l'apprentissage automatique nous permettent de repenser les modes de diagnostic clinicien à partir d'images médicales.

Notre contribution porte sur plusieurs axes : Tout d'abord, nous proposons une méthode de segmentation basée sur le clustering flou appelé noyau modifié avec une pénalité de type entropie : MKPE afin de segmenter les tissus cérébraux et de surmonter le problème de l'inhomogénéité d'intensité et du bruit présenté dans les images de résonance magnétique cérébrale (IRM). Deuxièmement, nous proposons un nouvel indice de validité floue V_{ECS} pour évaluer la qualité des clusters/segments produits par des algorithmes de segmentation floue. Cet index est amélioré pour détecter les segments chevauchés présentés dans les IRM cérébrale et les images de leucémie. Finalement, deux modèles de réseaux de neurones à convolution profonde sont développés pour la classification des images médicales. Dans ce contexte, nous proposons en premier lieu un modèle de réseau de neurones à convolution (RNC) profond appelé IRDNet pour classer la maladie d'Alzheimer à l'aide d'images de résonance magnétique cérébrale. En second lieu, nous proposons CVDNet, un modèle de RNC profond pour classer les cas infectés par COVID-19, les cas normaux et autres de pneumonie à l'aide d'images radiographiques pulmonaires.

Mots clés : Image médicale, Apprentissage automatique, Apprentissage profond, Réseau de neurones à convolution, Clustering flou, Segmentation, Indice de validité de classes, Classification.

Abstract

Medical imaging is an essential tool in many fields of medical applications, used for diagnosis and treatment. However, reading medical images and making diagnosis or treatment recommendations require especially medical specialists. The current practice of reading medical images is time-consuming, labor-intensive, error-prone and costly. It would be more desirable to have a computer-aided system that can automatically make diagnosis and treatment recommendations. Advances in machine learning enable us to rethink the ways of clinician diagnosis based on medical images.

Our contribution focuses on several areas: First, we propose an efficient segmentation method based on fuzzy clustering called Modified Kernel with Entropy-type Penalty: MKPE in order *to segment brain tissue* and to overcome the problem of intensity inhomogeneity and noise presented in brain magnetic resonance images (MRI). Second, we propose a new fuzzy cluster validity index V_{ECS} to assess the quality of the clusters/segment produced by fuzzy segmentation algorithms. This index is enhanced to detect overlapped segments presented in brain MR and leukemia images. Finally, two deep convolutional neural network (CNN) models are developed for medical image classification. In this context, we first propose a deep CNN framework called IRDNet to classify Alzheimer's disease using brain MR images. Secondly, we propose CVDNet, a deep CNN model to classify COVID-19, normal and other pneumonia cases using chest X-ray images.

Keywords: Medical image, Machine learning, Deep learning, Convolutional neural network, Fuzzy clustering, Segmentation, Cluster validity index, Classification.

THESE DE DOCTORAT

Présentée par

Mlle : CHAIMAE OUCHICHA

Spécialité : Mathématiques Appliquées / Informatique

Sujet de la thèse : Segmentation et classification en imagerie médicale pour le diagnostic et le traitement de maladies neurologiques et infectieuses : cas d'Alzheimer et du Covid-19

Thèse présentée et soutenue le mercredi 14 juillet 2021 à 10h au centre de conférences devant le jury composé de

Nom Prénom	Etablissement	
Pr. Mohamed Lamrini	Faculté des Sciences Dhar El Mahraz, Fès	Président
Pr. Ahmed Drissi El Maliani	Université Mohammed V, Rabat	Rapporteur
Pr. Noureddine En Nahnahi	Faculté des Sciences Dhar El Mahraz, Fès	Rapporteur
Pr. Brahim Ouhbi	Ecole Nationale Supérieure des Arts et des Métiers, Meknès	Rapporteur
Pr. Fatima Ezzaki	Faculté des Sciences et Techniques, Fès	Examinateur
Pr. Mohammed Meknassi	Faculté des Sciences Dhar El Mahraz, Fès	Invité d'honneur
Pr. Ouafae Ammor	Faculté des Sciences et Techniques, Fès	Directeur de thèse

Laboratoire d'accueil : Modélisation et Structures Mathématiques.

Etablissement : Faculté des Sciences et Techniques de Fès

Résumé

L'imagerie médicale est un outil essentiel dans de nombreux domaines d'applications médicales, utilisé à la fois pour le diagnostic et le traitement. Cependant, la lecture d'images médicales et l'établissement de recommandations de diagnostic ou de traitement nécessitent des médecins spécialistes. La pratique actuelle de lecture d'images médicales est laborieuse, chronophage, coûteuse et sujette aux erreurs. Il serait plus souhaitable de disposer d'un système assisté par ordinateur capable de faire automatiquement des recommandations de diagnostic et de traitement. Les progrès de l'apprentissage automatique nous permettent de repenser les modes de diagnostic clinicien à partir d'images médicales. De nombreux algorithmes basés sur l'apprentissage automatique ont été appliqués à l'imagerie médicale pour résoudre des problèmes tels que la segmentation, la classification, la restauration et la détection. En particulier, avec la large application des approches d'apprentissage en profondeur, les performances de l'analyse d'images médicales ont été considérablement améliorées. Dans cette thèse, nous examinons les méthodes d'apprentissage automatique et d'apprentissage en profondeur pour deux problèmes majeurs de l'analyse d'images médicales : la segmentation et la classification. La classification des images médicales est l'un des problèmes les plus importants dans le domaine de la reconnaissance d'images, et son objectif est de classer les images médicales en différentes catégories pour aider les médecins à diagnostiquer les maladies. La segmentation des images médicales est une étape complexe et critique dans le domaine du traitement et de l'analyse d'images médicales. Son but est d'identifier la région d'intérêt, c'est-à-dire de localiser la tumeur, la lésion et d'autres anomalies, de mesurer le volume de tissu pour mesurer la croissance de la tumeur et d'aider à la planification du traitement.

Notre contribution porte sur plusieurs axes : Tout d'abord, nous proposons une méthode de segmentation basée sur le clustering flou appelé noyau modifié avec une pénalité de type entropie : MKPE afin de segmenter les tissus cérébraux et de surmonter le problème de l'inhomogénéité d'intensité et du bruit présenté dans les images de résonance magnétique cérébrale (IRM). Deuxièmement, nous proposons un nouvel indice de validité floue V_{ECS} pour évaluer la qualité des clusters/segments produits par des algorithmes de segmentation floue. Cet index est amélioré pour détecter les segments chevauchés présentés dans les IRM cérébrale et les images de leucémie. Finalement, deux modèles de réseaux de neurones à convolution profonde sont développés pour la classification des images médicales. Dans ce contexte, nous proposons en premier lieu un modèle de réseau de neurones à convolution (RNC) profond appelé IRDNet pour classer la maladie d'Alzheimer à l'aide d'images de résonance magnétique cérébrale. En second lieu, nous proposons CVDNet, un modèle de RNC profond pour classer les cas infectés par COVID-19, les cas normaux et autres de pneumonie à l'aide d'images radiographiques pulmonaires.

Pour chacun des problèmes traités, Des expériences approfondies sont menés pour évaluer les méthodes proposées et un ensemble des résultats numériques sont présentés et comparés avec ceux issus des autres approches provenant de la littérature.

Mots clés : Image médicale, Apprentissage automatique, Apprentissage profond, Réseau de neurones à convolution, Clustering flou, Segmentation, Indice de validité de classes, Classification.

Abstract

Medical imaging is an essential tool in many fields of medical applications, used for diagnosis and treatment. However, reading medical images and making diagnosis or treatment recommendations require especially medical specialists. The current practice of reading medical images is time-consuming, labor-intensive, error-prone and costly. It would be more desirable to have a computer-aided system that can automatically make diagnosis and treatment recommendations. Advances in machine learning enable us to rethink the ways of clinician diagnosis based on medical images. Many algorithms based on machine learning have been applied in medical imaging to solve problems including, segmentation, classification, restoration and detection. Particularly, with the wide application of deep learning techniques, the performance of medical image analysis has been significantly improved. In this thesis, we investigate machine learning and deep learning methods for two key issues in medical image analysis: segmentation and classification. Medical image classification is one of the most important problems in the image recognition area, and its aim is to classifiy medical images into different categories to help doctors in disease diagnosis; and medical image segmentation is a complex and critical step in the field of medical image processing and analysis. Its purpose is to measure tissue volume to measure growth of tumor, identify region of interst i.e., locate tumor, lesion and other abnormalities and help in treatment planning.

Our contribution focuses on several areas: First, we propose an efficient segmentation method based on fuzzy clustering called Modified Kernel with Entropy-type Penalty: MKPE in order to segment brain tissue and to overcome the problem of intensity inhomogeneity and noise presented in brain magnetic resonance images (MRI). Second, we propose a new fuzzy cluster validity index V_{ECS} to assess the quality of the clusters/segment produced by fuzzy segmentation algorithms. This index is enhanced to detect overlapped segments presented in brain MR and leukemia images. Finally, two deep convolutional neural network (CNN) models are developed for medical image classification. In this context, we first propose a deep CNN framework called IRDNet to classify Alzheimer's disease using brain MR images. Secondly, we propose CVDNet, a deep CNN model to classify COVID-19, normal and other pneumonia cases using chest X-ray images.

For each treated problem, extensive experiments are carried out to evaluate the proposed methods and a set of numerical results are presented and compared with approaches from the literature.

Keywords: Medical image, Machine learning, Deep learning, Convolutional neural network, Fuzzy clustering, Segmentation, Cluster validity index, Classification.

Acknowledgments

The work presented in this thesis was carried out in the Modeling and Mathematical Structures Laboratory at the Faculty of Sciences and Technologies of Fez under the supervision of Professors Ouafae Ammor and Mohammed Meknassi.

I would like to express my utmost gratitude to all the people that helped me during my PhD.

First and foremost, I would like to express the deepest appreciation to my supervisor, Professor Ouafae Ammor, for the help that brought me, for her permanent availability and the sharing interesting and novel ideas, for his motivation, immense knowledge and contagious passion, as well as for her academic and pedagogic guidance and her encouragement throughout the dissertation stages. Thanks to the quality of its rigorous supervision, its exemplary professionalism, its relevant remarks and its real contribution to my research, this thesis was able to see the light of day. It was a great privilege and honor to study under her guidance.

I would also like to offer my special thanks to my co-supervisor, Professor Mohammed Meknassi, for his great availability and for his scientific, pedagogic and human qualities, as well as for his valuable advice, ideas, and feedback through-out all stages of this work. Thank you for engaging in my research and for guiding me in the right direction. Your constructive criticisms and words of encouragement have been essential in the development of my research.

I would like to address my most sincere thanks to Mr. Drissi El Maliani Ahmed, Professor at Mohammed V University, Rabat and to Mr. En Nahnahi Noureddine, Professor at the Faculty of Sciences Dhar Mahraz Fez, and to Mr. Ouhbi Brahim, Professor at the National School of Arts and Crafts of Meknes, for having accepted to be rapporteurs of this work, and for taking the time to judge this thesis.

I associate with these thanks Ms. Fatima Ezzaki, Professor at the Faculty of Sciences and Technologies Fez for accepting to examine this work and to Mr. Lamrini Mohamed, Professor at the Faculty of Sciences Dhar Mahraz Fez for having accepted to be part of the jury members for my thesis.

I would like to thank the Modeling and Mathematical Structures Laboratory for providing me computational resources necessary for the research. To my colleagues from this laboratory with whom I shared many experiences.

I am forever grateful to my parents for their trust, their daily encouragement and their infinite love, for the sacrifices they have made for me throughout my life. I am deeply grateful for raising me to a valuable education. I couldn't find the right expression to thank them. All I have to say to you I love you more than anything in the world. May Almighty God protect you and lend you both good health and long life.

My sincere gratitude goes to my brothers and sisters for their love, for their emotional support and for continuous encouragement when I faced with difficulties. My sincere thanks also go to my family and friends. Their support has been invaluable in completing my work.

Contents

Résumé i
Abstract ii
Acknowledgments iii
Contents iv
List of Figures viii
List of Tables xi
List of Abbreviation xiii
Author's Publications xv
1. Introduction **1**
1.1. Background and Motivation 1
1.2. Problem Statement 3
1.3. Thesis Objectives 8
1.4. Thesis Contributions 8
1.5. Thesis Organization 10
2. Preliminaries **12**
2.1. Introduction 12
2.2. Artificial Intelligence 12
2.3. Medical Image Analysis 13
2.4. Machine Learning Framework 14
2.4.1. Machine Learning 14
2.4.2. Supervised Learning Framework 15
2.4.3. Unsupervised Learning Framework 16
2.5. Image Segmentation 17
2.5.1. Clustering in Image Segmentation 18
2.5.2. Cluster Validity Indices 20
2.6. Image Classification 21
2.7. Deep Learning 22
2.8. Conclusion 23
3. **Segmentation of Brain MRI Tissue** **25**
3.1. Introduction 25
3.2. Magnetic Resonance Image Processing 25
3.3. MRI Artifacts 25
3.3.1. Noise in MR Imaging 26

3.3.2. Intensity Inhomogeneity 27
3.4. Segmentation of Brain MRI Corrupted by Noise and Intensity Inhomogeneity. 28
3.4.1. Related Work 29
3.4.2. Fuzzy Clustreing Methods for Brain MRI Segmentation 30
3.4.2.1. Spatially Constrained KFCM for Image Segmentation 31
3.4.2.2. Adaptive Kernel-based Fuzzy C-Means Clustering with Spatial Constraints 32
3.4.2.3. Kernel Generalized Fuzzy C-Means Clustering with Spatial Information 33
3.4.2.4. Improved Kernel Fuzzy Local Information C-Means Algorithm and Outlier Rejection 33
3.4.2.5. Improved Kernelized Rough-Fuzzy C-Means with Spatio-Contextual Information 34
3.4.2.6. Radial Basis Kernel based Intuitionistic Fuzzy C-Means Clustering 35
3.5. The Proposed Method for Automatic Segmentation of Brain MRI Corrupted by Noise and Intensity Inhomogeneity 35
3.5.1. Formulation 36
3.5.2. Convergence Proof of the Proposed Model 41
3.5.3. The Proposed Algorithm 45
3.6. Experimental Setup and Results 48
3.6.1. Datasets 48
3.6.2. Evaluation Metrics 49
3.6.3. Results and Discussion 50
3.6.3.1. MR Brain Image Corrupted with Noise and Intensity Inhomogeneity 50
3.6.3.2. Time Complexity 54
3.7. Conclusion 54
4. Cluster Validity Indices in Medical Image Segmentation 56
4.1. Introduction 56
4.2. Fundamental Principle of Validity Indices 57
4.2.1. Problem Position 57
4.2.2. Process for Finding the Optimal Number of Clusters 58
4.3. Fuzzy C-Means Clustering Method 58
4.4. Fuzzy Cluster Validity Indices 59
4.4.1. Review of Fuzzy Validity Indices 59
4.4.2. Mathematical Background of Some Fuzzy Cluster Validity Indices 60
4.4.2.1. Wu-and-Li Validity Index ($V_{WLI}\downarrow$) 60
4.4.2.2. Separation Validity Index ($V_{CVI_2}\uparrow$) 61
4.4.2.3. Voscar Validity Index ($V_{oscar}\downarrow$) 61
4.4.2.4. Weighted Summation Type of CVI: ($V_{WSCVI}\downarrow$) 63
4.4.2.5. CSα Validity Index ($V_{CS\alpha}\downarrow$) 63
4.4.2.6. HF Validity Index ($V_{HF}\downarrow$) 64
4.5. The Proposed Validity Index 64

4.6. Experimental Results and Analysis 69
4.6.1. Results using Brain MR images 69
4.6.2. Results using Leukemia Images 70
4.7. Conclusion 72
5. Deep CNNs for Medical Image Classification 73
5.1. Introduction 74
5.2. Introduction to Artificial Neural Networks 75
5.2.1. Perceptron 77
5.2.2. Multilayer Perceptron 78
5.3. ANN Training Procedure 79
5.3.1. Gradient Descent Methods 79
5.3.1.1. Gradient Descent 79
5.3.1.2. Mini-Batch Stochastic Gradient Descent 80
5.3.1.3. Stochastic Gradient Descent with Momentum 80
5.3.1.4. Adaptive Gradient 80
5.3.1.5. Adaptive Moment Estimation 81
5.3.2. Backpropagation Algorithm 81
5.3.3. The Activation Function 84
5.3.4. Vanishing and Exploding Gradients 86
5.4. Deep Convolutional Neural Networks 86
5.4.1. Components of a DCNN 86
5.4.2. Input Layer 88
5.4.3. Convolutional Layer 88
5.4.4. Pooling Layer 90
5.4.5. Classification Layers 91
5.5. Deep CNN Training 91
5.5.1. Backpropagation in Pooling Layers 92
5.5.2. Backpropagation in Convolutional Layer 93
5.6. Overfitting and Regularization 93
5.6.1. Dropout 94
5.6.2. Weight Decay 95
5.6.3. Batch Normalization 95
5.6.4. Early Stopping 96
5.7. Conclusion 97
Part 1 Classification of Alzheimer's disease from Brain MRI using DCNN 98
5.8. Introduction 98
5.9. Related Work 100
5.9.1. LeNet-5 102
5.9.2. AlexNet 104
5.9.3. VGG-16 107

5.9.4. GoogLeNet 109
5.9.5. Residual Network (ResNet) 112
5.9.6. DenseNet 118
5.10. Network Structure of the Proposed IRDNet 112
5.11. Experimental Setup and Results 118
5.11.1. Dataset Description 118
5.11.2. Evaluation Metrics 119
5.11.3. Results and Discussion 119
5.12. Conclusion 123
Part 2 Classification of COVID-19 pandemic from chest x-ray images using DCNN 124
5.13. Introduction 125
5.14. Related work 127
5.15. Network structure of the proposed CVDNet 129
5.16. Experimental Setup 134
5.16.1. Dataset description 134
5.16.2. Results and discussion 134
5.17. Conclusion 138
6. Conclusion and Future Work 139
A. Résumé en français 142
References 164

List of Figures

FIGURE 2. 1: Subsets of a dataset used for training a machine learning model..... 16

FIGURE 3. 1: MRI sequences: (a) T1-weighted image, (b) T2-weighted image, (c) PD-weighted image 26

FIGURE 3. 2: Rician PDF plot at different A values with fixed $\sigma = 1.$ In case of high SNR ($A \geq 3$) the Rician distribution starts to approximate to Gaussian one, whereas in image points where (only noise is present) it corresponds to Rayleigh distribution. 27

FIGURE 3. 3: (a): shows the brain MRI, in which the red rectangle is a region of 4×4 pixels. (b): represents the grayscale value of pixels in this region. 47

FIGURE 3. 4: Test images. (a) – (b): two original slices of brain MRI from the simulated data 49

FIGURE 3. 5: (e): Image (a) corrupted with 9% noise and 20% intensity inhomogeneity. (f): ground truth..... 51

FIGURE 3. 6: The images from the first to 7th row represent respectively the segmentation results of GM, WM and CSF tissues and global segmentation from KIFCM, KGFCM-S, KFCM_S, AKFCMS, KRFCMSC, KFLICM and our proposed MKPE for simulated brain MRI with 9% noise and 20% inhomogeneity. 69

FIGURE 4. 1: Test images. (a), (b), (c), (d): original slices of brain MRI from the simulated data 69

FIGURE 4. 2: (i), (j), (k), (l): segmentation results of the images (a), (b), (c) and (d) using FCM algorithm with c=4. 69

FIGURE 4. 3: The values of validity indexes in the range $c = 2,...,10$ for the image (a) to image (d)... 70

FIGURE 4. 4: Test images. (e), (f), (g), (h): original images of the Acute Leukemia 70

FIGURE 4. 5: (i), (j), (k), (l): segmentation results of the images (e), (f), (g) and (h) using FCM algorithm with c=4..... 71

FIGURE 4. 6: The values of validity indexes in the range $c = 2,...,10$ for the image (a) to image (d)..... 74

FIGURE 5. 1: Traditional machine learning versus Deep Learning. 75

FIGURE 5. 2: Comparison between a biological neuron (left) and the mathematical model of an artificial neuron (right). In biological networks, the input data comes from another neuron passing through the dendrite, the cell body does the calculations, and the result exits through the axon. The functioning of an artificial neuron is modeled on that of a biological neuron. Image taken from [49].....78

FIGURE 5. 3: A multilayer Perceptron architecture [22]. 85

FIGURE 5. 4: The graphs of four common different activation functions. The names of the above curve functions are: the sigmoid function, the tanh function, the ReLu function and the softplus function.....89

FIGURE 5. 5: Illustration of the convolution operation on the input feature map of size 28×28 neurons in layer $l-1$. The convolution uses a receptive field of size 5×5 and overlapping parameter of 2 neurons, thus producing feature maps of size 12×12 neurons in layer l. Note that neighboring neurons in convolutional layers have overlapping receptive fields. 94
FIGURE 5. 6: An example of underfitting and overfitting problems.....94
FIGURE 5. 7: Dropout Neural Net Model. Left: A standard neural net with two hidden layers. Right: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.....94
FIGURE 5. 8: Error as a function of iteration steps for training and validation set. 97
FIGURE 5. 9: MRI view of a healthy brain (left) and an Alzheimer's brain (right) [16]. (Yellow - Cortex, Blue - Ventricle, Purple - Hippocampus volume reduced) 99
FIGURE 5. 10: Diagram of LeNet-5 architecture [84]. 101
FIGURE 5. 11: AlexNet Architecture [74] 102
FIGURE 5. 12: Architecture of VGG-16 [123].....104
FIGURE 5. 13: The Inception block in GoogLeNet [126].....105
FIGURE 5. 14: Architecture of GoogLeNet [126].....106
FIGURE 5. 15: Residual learning: a building block [55].....108
FIGURE 5. 16: Architecture of ResNet-152 [55] 108
FIGURE 5. 17: A 5-layer dense block with a growth rate of k = 4. Each layer takes all preceding feature maps as input [59].....109
FIGURE 5. 18: Architecture of DenseNet-121 with 4 dense blocks [59]. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling. 110
FIGURE 5. 19: MDB diagram with 6 layers (path=2, the connections to every layer are at most 3) 113
FIGURE 5. 20: The architecture of the proposed residual dense block (MDRB).....114
FIGURE 5. 21: Multi-scale residual dense architecture (MSRD).....115
FIGURE 5. 22: The structure of the proposed IRDNet.....116
FIGURE 5. 23: Four brain MRI from the OASIS dataset: (a) Normal, from (b) to (d) Alzheimer's with different stages. 117
FIGURE 5. 24: Diagram of training, validation and testing set used in the five-fold cross-validation technique. 120
FIGURE 5. 25: (a) IRDNet model loss for training and test dataset. (b) IRDNet model accuracy for training and test dataset.....121
FIGURE 5. 26: Confusion matrix results of our IRDNet for 2-class Classification. (a) overlapped CM, (b) Fold-1 CM, (c) Fold-2 CM, (d) Fold-3 CM, (e) Fold-4 CM, and (f) Fold-5 CM. 121
FIGURE 5. 27: Example of chest X-ray images: (a) COVID-19 chest X-ray image, (b) normal chest X-ray image, (c) viral pneumonia chest X-ray image. 126
FIGURE 5. 28: The structure of the proposed CVDNet model 130
FIGURE 5. 29: Illustration of the proposed block. Inputs are connected to the output of their layers, and added to the inputs of other levels. 131

FIGURE 5. 30: Effect of stride and pooling on image resolution. (a) Original image. (b) Max Pooling, size(2,2), stride:2.....133
FIGURE 5. 31: Schematic diagram of training, validation and test set used in the 5-fold cross-validation 134
FIGURE 5. 32: (a) CVDNet model loss for training and test dataset. (b) CVDNet model accuracy for training and test dataset.....135
FIGURE 5. 33: Confusion matrix results of our CVDNet for 3-class Classification. (a) overlapped CM, (b) Fold-1 CM, (c) Fold-2 CM, (d) Fold-3 CM, (e) Fold-4 CM, and (f) Fold-5 CM. 136

List of Tables

Table 3. 1: Comparative study of six methods and the proposed method in terms of tissue segmentation accuracy on simulated brain MR image (a) 52

Table 3. 2: Comparative study of six methods and the proposed method in terms of tissue segmentation accuracy on simulated brain MR image (b) 53

Table 3. 3: Average computational time (in seconds) with MATLAB on MRI images of KIFCM, KRFCMSC, KFLICM, KFCM_S, KGFCM-S, AKFCMS and MKPE algorithms 54

Table 4. 1: The optimal number of clusters obtained by six recent validity indices and the proposed index for Brain MRI and Leukemia medical imaging data sets 72

Table 5. 1: Connection scheme between the layers and of LeNet-5 101

Table 5. 2: Configuration of LeNet-5 Architecture 102

Table 5. 3: Configuration of AlexNet Architecture 103

Table 5. 4: Configuration of VGG-16 Architecture 104

Table 5. 5: Configuration of GoogLeNet Architecture 105

Table 5. 6: Configuration of ResNet-152 Architecture and its variants 106

Table 5. 7: Configuration of DenseNet-121 Architecture and its variants 110

Table 5. 8: Advantages and drawbacks of six a rchitectures of deep convolutional neural networks 111

Table 5. 9: Feature aggregation table corresponding to the multi-scale structure 116

Table 5. 10: The configuration of our IRDNet architecture for OASIS dataset 117

Table 5. 11: Details of training, validation and testing set. 120

Table 5. 12: Precision, F1-score and Recall values in fold 1 for NC and AD classes of the proposed IRDNet 122

Table 5. 13: Precision, F1-score and Recall values in fold 2 for NC and AD classes of the proposed IRDNet 122

Table 5. 14: Precision, F1-score and Recall values in fold 3 for NC and AD classes of the proposed IRDNet 122

Table 5. 15: Precision, F1-score and Recall values in f old 4 for NC and AD classes of the proposed IRDNet 122

Table 5. 16: Precision, F1-score and Recall values in fold 5 for NC and AD classes of the proposed IRDNet 122

Table 5. 17: Performance of the proposed IRDNet on each fold 122

Table 5. 18: Average accuracy and computation time results of the models 122

Table 5. 19: Representative works for Chest X-ray images based on the detection of COVID-19 infection 128

Table 5. 20: The configuration of the proposed CVDNet architecture for COVID-19 dataset 133

Table 5. 21: Details of training, validation and test set 135

Table 5. 22: Precision, F1-score and Recall values in fold 1 for Normal, COVID-19 and pneumonia classes of the proposed CVDNet. 137
Table 5. 23: Precision, F1-score and Recall values in fold 2 for Normal, COVID-19 and pneumonia classes of the proposed CVDNet 137
Table 5. 24: Precision, F1-score and Recall values in fold 3 for Normal, COVID-19 and pneumonia classes of the proposed CVDNet. 137
Table 5. 25: Precision, F1-score and Recall values in fold 4 for Normal, COVID-19 and pneumonia classes of the proposed CVDNet. 137
Table 5. 26: Precision, F1-score and Recall values in fold 5 for Normal, COVID-19 and pneumonia classes of the proposed CVDNet. 137
Table 5. 27: Precision, F1-score and Recall values in fold 5 for Normal, COVID-19 and pneumonia classes of the proposed CVDNet. 137

List of Abbreviation

2D	Two-dimensional
3D	Three-dimensional
AD	Alzheimer's disease
Adam	Adaptive Moment Estimation
AdaGrad	Adaptive Gradient
ADNI	Alzheimer's disease Neuroimaging Dataset
AI	Artificial Intelligence.
ANN	Artificial Neural Network
Avg Pool	Average Pooling
BN	Batch Normalization
CAD	Computer-Aided System
CM	Confusion Matrix
CNN	Convolutional Neural Network
Conv	Convolution
COVID-19	Corona virus Disease 2019
CSF	Cerebral Spinal Fluid
CT	Computerized Tomography
CVI	Cluster Validity Index
DCNNs	Deep Convolutional Neural Networks
DL	Deep Learning
DNN	Deep Neural Network
FC	Fully Connected
FCM	Fuzzy C-Means
IIH	Intensity Inhomogeneity
GD	Gradient Descent
GM	Gray Matter
GPU	Graphics Processing Unit
GRBF	Gaussian Radial Basis Function
GT	Ground Truth
KFCM	Kernelized Fuzzy C-Means

MKPE	Modified Kernel with Entropy type penalty
ML	Machine Learning
MLP	Multilayer Perceptron
MP	Max Pooling
MRI	Magnetic Resonance Imaging
NC	Normal Control
LRN	Local Response Normalization
OASIS	Open Access Series of Imaging Studies
PDF	Probability Density Function
ReLU	Rectified Linear Unit
RF	Receptive Field
SBD	Synthetic Brain Database
SGD	Stochastic Gradient Descent
SNR	Signal-to-Noise ratio
Tanh	Hyperbolic Tangent Function
WM	White Matter

Author's Publications

The peer-reviewed journal papers, conference proceedings as well as the papers submitted for publication that were written by the authors during the Ph.D. are listed below.

International Journal

1. Ouchicha, C., Ammor, O. & Meknassi, M. A New Validity Index in Overlapping Clusters for Medical Images. *Aut. Control Comp*. Sci. 54, 238–248 (2020).
2. Ouchicha, C., Ammor, O. & Meknassi, M. CVDNet: A Novel Deep Learning Architecture for Detection of Coronavirus (Covid-19) from Chest X-Ray Images. *Chaos, Solitons and Fractals*. 140, 110245 (2020).
3. Ouchicha, C., Ammor, O. & Meknassi, M. A Novel Deep Convolutional Neural Network Model for Alzheimer's disease Classification using Brain MRI. . *Aut. Control Comp*. Sci.

International Conferences

1. Ouchicha, C., Ammor, O. & Meknassi, M. Cluster Validity Index: Comparative Study and a New Validity Index. *In Proceedings of the International Conference on Learning and Optimization Algorithms: Theory and Applications*, 2018.
2. Ouchicha, C., Ammor, O. & Meknassi, M. An Efficient Cluster Validity Index to Detect Optimal Number of Clusters with Overlap. *In the International Conference on Networking, Information Systems & Security*, 2018.
3. Ouchicha, C., Ammor, O. & Meknassi, M. Comparaison of Some Clustering Algorithms for Segmentation of Brain Images. *In the International Conference on Intelligent Computing in Data Sciences*, 2018.
4. Ouchicha, C., Ammor, O. & Meknassi, M. A Novel Index to Detect the Optimal Number of Clusters in High Overlapping. *In the International Conference on Applied Mathematics*, 2018.
5. Ouchicha, C., Ammor, O. & Meknassi, M. Unsupervised Brain Tumor Segmentation from Magnetic Resonance Images. *In the International Conference on Wireless Networks and Mobile Communications*, 2019.

Papers submitted for publication

1. Ouchicha, C., Ammor, O. & Meknassi, M. A New Approach Based on Penalty-type Entropy with Modified Kernel Fuzzy C-Means Clustering for Brain MRI Segmentation.
2. Ouchicha, C., Ammor, O. & Meknassi, M. Classification of Alzheimer's disease Based on Deep Convolutional Neural Network using Brain MRI.

CHAPTER 1

Introduction

Contents

1.1. Background and Motivation 1
1.2. Problem Statement 3
1.3. Thesis Objectives 8
1.4. Thesis Contributions 8
1.5. Thesis Organization 10

This chapter gives a general introduction to the thesis. It introduces the background and motivation behind the objectives to be explored, and explains the issues of research, as well as provides the contributions and the structure of this thesis.

1.1. Background and Motivation

The development and the field of application of digital technology have continued to grow in recent years, this technology which was initially dedicated to scientific calculations, today it continues to invade our daily lives, it no longer needs to be demonstrated that it occupies an important place in our life, among the components of digital systems, a large part is given to imagery. An image records a scene of the real world in a numeric representation that can be stored, transmitted and studied afterwards. The well-known proverb "a picture is worth a thousand words" indicates that an image has a powerful ability of describing the rich information it carries. Image processing saw its beginnings in the 1920s but did not really take off until the 1960s with the development of computers [87]. Initially, image processing techniques are essentially methods of restoring and compressing images. With the progress of computer science, techniques for detecting primitives and many other treatments are developing in fields as varied as medicine, remote sensing, etc. It was in the 2000s that digital images and consequently image processing became omnipresent.

In recent decades, with the enrichment of medical image databases, medical imaging [29] has become essential in the diagnosis and therapy of diseases. However, reading medical images and making diagnosis or treatment recommendations require specially trained medical specialists. The current practice of reading huge volumes of medical image data is labor-intensive, time-consuming, costly, and error-prone. It would be more desirable to have a computer-aided system (CAD) that can automatically make diagnosis and treatment recommendations. Current development in machine learning (ML) [118] and deep learning (DL) [10] methods motivate us to improve current medical image analysis systems. Machine learning has become one of the main tools for analyzing medical images to help physicians diagnose, predict the risk of disease and prevent them before it becomes too late. In general ML comprises a collection of algorithms that enables a computer to determine significant patterns from data without human intervention and DL is a new part of the machine learning field that encompasses a wide range of network

architectures designed to perform a task. ML and DL has greatly matured over the past few years and has had success in various fields such as image segmentation [146], classification, detection, recording and image reconstruction.

Despite technological advances in the medical field and in particular in medical imaging, the study of the human brain, whether anatomically or functionally, is a difficult problem and remains a subject of highly topical research, because developments in imaging equipment and techniques, the large computing capacity of computers and the implementation of complex data processing systems. There are many clinical tools for in vivo observation of the cerebral area. These include but are not limited to the X-ray scanner, magnetic resonance imaging (MRI) [91], tomography, scintigraphy. . . Among all these techniques, MRI is increasingly used in clinical routine. This non-invasive technique, which exploits the magnetic properties of tissues, indeed allows observation with fine resolution of biological tissues, and therefore in particular of cerebral tissues. Its sensitivity to slight changes in tissues is in fact a valuable and very effective ally in the context of the diagnosis of brain tumors.

For a reliable diagnosis, in the medical field, accuracy is important. In terms of image analysis, it is necessary that the segmentation be precise. Segmentation is one of the essential operations for any automatic image processing. It is an important computer vision task that transforms input data into disjoint subsets. It partitions an image into multiple separate regions that are further analyzed to retrieve meaningful high-level information. A basic view of segmentation is that we intend to find in data structures that share similar characteristics and are different from other structures. This problem is called also clustering [15]. The applicability of clustering algorithms for image segmentation has been widely studied. In the context of brain MRIs, segmentation consists of delineating the brain structures (gray matter, white matter, cerebrospinal fluid) that can be used in several applications such as morphometry (measurements of shapes) to help identify, understand and follow progression of pathologies (Alzheimer's, schizophrenia), functional mapping and visualization. However, MRI suffer from a large number of artifacts which make their automatic segmentation difficult. These artifacts are mainly due to noise, the inhomogeneity of the radio frequency, and to the patient himself from his movements during image acquisition, etc. Thus, manual segmentation on each slice is very costly in human time (manual segmentation of a few structures on a single MRI volume can take half a day). As the contrast is low, some borders between regions are difficult to define. Manual segmentation then introduces a significant intra- and inter-expert variation: (1) it is not very reproducible, even by the same expert and (2) different experts propose different versions according to their experience. Today, automatic tools are widely used (especially for segmentation of structures or lesions), but still require human intervention. Given the ambiguous, complementary and redundant nature of the information provided by different MRI sequences, fuzzy clustering techniques [116] adapt well to the segmentation of brain tissue. One of the most critical problems in clustering is choosing the right number of clusters. Several clustering algorithms actually require the number of clusters to be given as a parameter i.e., must be known in advance, something which is not always true especially if we have data with unknown characteristics or in the case of clustering problems with new data that have never been mined yet. For this purpose, researchers have proposed what is called Cluster Validity Index (CVI) [119]. CVIs are tools that allow predicting the real number of clusters of a given data set with unknown number of clusters.

Another essential operation for any automatic image processing is classification. Image classification aims to assign an image to categories or classes of interest by analyzing its contents.

In computer-aided diagnosis, image classification plays a significant role. Classification in medical imaging [30] is an important process in diagnosis, refers to the task of analyzing an input image data and assigns it an output class indicating the presence of a disease or not. For example, classifying patients as having Alzheimer's disease [66] based on 3D or 2D brain MRI data. This helps doctors and radiologists in their decision making. Recently, artificial intelligence (AI) [117] using deep learning technology has demonstrated great success in the medical imaging domain and it has performed remarkable or yet superior human-level performance on image classification due to its enables creation of end-to-end models that achieve promised results using input data, without the need for manual feature extraction. AI can offer several advantages over traditional analytics, and clinical decision-

making approaches. The health-care industry is a high priority sector where people expect the highest level of service. Proper diagnosis of disease is crucial to treatment and saving a patient's life. For example, Alzheimer's disease is not curable. But if we can diagnosis it at an early stage, physicians can at least try to delay the harmful impact of the disease. On the other hand, diagnosis is expensive and requires specialized expertise. In many countries of the world, primary medical care is limited, and expert physicians are not available in time of need. If we can build CAD systems that can automatically analyze medical images and diagnosis disease with human-level expertise, it will reduce the cost of treatment dramatically.

Recently, at the dawn of 2019 the World Health Organization (WHO) was notified by the Chinese authorities on novel coronavirus (COVID-19) [142] causing severe respiratory illness emerging from Hubei Providence of China. COVID-19 spread rapidly around the world and became a pandemic. It has caused a devastating effect on daily lives, public health, and the global economy. It is critical to detect the positive cases as early as possible so as to prevent the further spread of this epidemic and to quickly treat affected patients. The need for auxiliary diagnostic tools has increased as there are no accurate automated toolkits available. Recent findings obtained using radiology imaging techniques suggest that such images contain salient information about the COVID-19 virus. Application of advanced artificial intelligence (AI) techniques coupled with radiological imaging can be helpful for the accurate detection of this disease.

1.2. Problem Statement

This section introduces problems definition for various stages involved in this research.

1.2.1. Problem definition: Brain tissue segmentation from noisy and intensity inhomogeneity MRI

Over the last few decades, the rapid development of noninvasive brain imaging technologies has opened new horizons in analyzing and studying the brain anatomy and function. Enormous progress in accessing brain injury and exploring brain anatomy has been made using magnetic resonance imaging (MRI). MRI [91] is one of the most common ways to visualize brain structures. Based on this imaging technique, the study of the main cerebral tissues (namely, white matter (WM), gray matter (GM) and Cerebral Spinal Fluid (CSF)) is in particular a key point in the context of diagnosis and patient follow-up. The advances in brain MR imaging have also provided large amount of data with an increasingly high level of quality. The analysis of these large and complex MRI datasets has become a tedious and complex task for clinicians, who have to manually extract important information. This manual analysis is often time-consuming and prone to errors due to the large number of MRI slices for each patient which composes the three-dimensional information and also due to intra/inter-observer variability of manually segmented scans. These difficulties in brain MRI data analysis required inventions in computerized methods to improve disease diagnosis and testing. Nowadays, computerized methods for MR image segmentation have been extensively used to assist doctors in qualitative diagnosis. Brain MRI segmentation is an essential task in many clinical applications because it influences the outcome of the entire analysis. This is because different processing steps rely on accurate segmentation of anatomical regions. For example, MRI segmentation is commonly used for measuring and visualizing different brain structures, for delineating lesions, for analyzing brain development, and for image-guided interventions and surgical planning. This diversity of image processing applications has led to development of various segmentation techniques of different accuracy and degree of complexity. Effectively, the development of robust automated segmentation methods, which can segment large amounts of MRI data, is nowadays an active research field. In the recent literature, a plentiful of general approaches has been proposed on medical image segmentation [146]. They can be generally classified into active contour [136], level set methods [3], and clustering methods [15]. But due to overlaps between intensity distribution of interested objects caused by noise and intensity inhomogeneity (also known as bias field) image segmentation is still an open problem [36].

Clustering based approaches have received a great deal of attention in medical imaging research community. Clustering is a process for classifying objects or patterns in such a way that samples of the same group are more similar to one another than samples belonging to different groups. Many clustering strategies have been used, such as the hard clustering. The hard clustering method [15] restricts each point of the data set to exclusively just one cluster. As a consequence, with this approach the segmentation results are often very crisp, i.e., each pixel of the image belongs to exactly just one class. Hereby, the boundaries among classes are sharply in such partition. However, in practice, the class attributes of most objects are not strict but ambiguous; hence it is not suitable for hard partitioning. Fortunately, the fuzzy set theory provides a powerful tool for such soft partitioning. Thus, people began to deal with clustering with fuzzy fashion and named them fuzzy cluster analysis. Since fuzzy clustering obtains the degree of uncertainty of samples belonging to each class and expresses the intermediate property of their memberships, it can more objectively reflect the real world. Thereby, it has become the main content of studies on cluster analysis. More recently fuzzy clustering is an effective tool in segmenting the medical images for further treatment plan. Fuzzy c-means (FCM) [12] is the most important technique of fuzzy clustering for image segmentation which assigns each pixel to fuzzy clusters without labels. In FCM, each pixel is assigned to all desired clusters with memberships graded as fuzzy values between 0 and 1. This algorithm is sensitive to noise, outliers, and other imaging artifacts, such as the intensity inhomogeneity and sensitive to the choice of initial cluster centers. To overcome these drawbacks, many modified FCM algorithms [53] have been proposed in the literature for the purpose of image segmentation. Some of the modified FCMs for image segmentation are as follows: Many researchers have incorporated spatial information into the original FCM algorithm to segment the images [90], [110]. On the other hand, there is a trend in recent machine learning work to construct a nonlinear version of a linear algorithm using the kernel method [19], for medical image analysis. Dao-Qiang Zhang and Song Chen [19] proposed kernel FCM algorithm with spatial constraints (KFCM_S) for medical image segmentation by adding the spatial penalty term to the FCM based kernel. Though these methods are worked well on image segmentation, they have some inadequacies such as they not pay any attention to correct the bias field from the image and they take much time for segmentation because of spatial terms and additional terms. Moreover, in iterative clustering algorithms, the procedure adopted for choosing initial cluster centers is randomly. Thus, the updated centers lead the result of clustering might be incorrect when it selects outliers as initial centers.
The objective of this work is presented below in the form of a research question.

RQ 1.a How could machine learning techniques be used to improve the segmentation performance of brain MRIs in terms of accuracy and execution time in the presence of intensity inhomogeneities?

RQ 1.b How can details be preserved in a noisy image segmentation? How can we improve the results of the previously created models with an emphasis on preserving details?

RQ 1.c How to choose initial points of clusters in FCM clustering?

1.2.2. Problem definition: Determination of the number of clusters and evaluation of the quality of the partitions produced by fuzzy clustering algorithm

Clustering [15] is one of the most important and popular unsupervised learning problems in Machine learning. It helps to find a structure in a set of unlabeled data in a way that the data in each cluster are similar together while they are dissimilar to the data in other clusters. An alternative to traditional crisp clustering methods that generate partitions where each pattern belongs to only one cluster is fuzzy clustering where each pattern is associated with every cluster to various membership degrees. Most fuzzy methods have several advantages including yielding regions more homogeneous than other methods; removing noisy spots; reduced sensitivity to noise compared to other techniques. However, the main challenge of these methods is that they require the user to specify the

number, c, of clusters which the user does not usually know. A resulting fuzzy c-partition has to be validated because its quality highly depends on this parameter. To address this problem, it is necessary to use a validity index to evaluate what is the optimal number of clusters. A simple technique applied to solve this problem is to execute the clustering algorithm several times, varying the number of clusters and to choose the optimal clustering using a validity index to measure the quality of the final clustering. The earliest validity indices for fuzzy clustering, the partition coefficient V_{PC} and the partition entropy V_{PE}, were introduced by Bezdek [14]. These indices have two disadvantages, exhibiting a trend to decrease or increase as the cluster number increases and being sensitive to the fuzzifier. Recently, Askari et al. [4] proposed a validity index called V_{CVI_2} which is defined as the total difference between membership grades. This index uses only the membership degrees which increase their dependence on the number of clusters c. It increases for a very large number of clusters c, hence the fact that the use of this index is favorable for partitions containing a small number of well separated clusters in addition to this problem it suffers from the absence of information about geometric properties of data. However, most validation indices proposed during the last decades have focused on two properties: compactness and separation. Compactness is used as a measure of scattering of the data within a cluster, and separation is used to account for inter-cluster structural information. The basic goal of cluster validity index is to find cluster number which meets the minimum compactness and the maximum separation. However, the classical indices are limited in their ability to compute the compactness and the separation, which leads to an incorrect validation result. Although some indices consider the structure of data sets, these ignore the overlap between clusters. In general, a geometric distance based only on the centroid information is insufficient to provide a meaningful interpretation of cluster separation (Kim et al. [72]) i.e., conventional separation measures only give limited information about the differences of the geometric structures of two partitions. Namely, it ignores the overlap between clusters.
The objective of this work is presented below in the form of a research question.

RQ 2.a Can the appropriate number of clusters be determined automatically? And if the answer is yes, how?

RQ 2.b How would recognize the number of clusters in a dataset that contain overlapping clusters?

1.2.3. Problem definition: Classification of Alzheimer's disease from brain MRI

With the aging of contemporary society, cognitive impairment and dementia in the elderly, mainly Alzheimer's disease (AD) [66], have become increasingly serious. As a multifactor, multistage, and clinical syndrome with concomitant diseases, senile cognitive impairment will take progress to irreversible dementia after clinical symptoms appear, eventually lead to death. Alzheimer's disease is currently irreversible, effective treatments lack in clinical practice. The development of a patient's status will go through several stages, so early diagnosis is essential. Early detection and accurate diagnosis of Alzheimer's disease is challenging for the progress of more prevailing treatments and can effectively slow down the disease progression while reduce the burden on patients' families and our society. Detecting Alzheimer's is a difficult and time-consuming task, but requires brain imaging report and human expertise. Needless to say, this conventional approach to detect Alzheimer's is costly and often error prone. The performance of human diagnosis degrades due to fatigue, cognitive biases, systems faults, and distractions. Sometimes it is also possible that the doctor may not be able to detect the disease in the early stages. Therefore, developing an automated analysis system that is fast, costs less and more reliable is essential to help clinicians in the early diagnosis and detection of Alzheimer's disease. However, artificial intelligence-based diagnosis systems are less error prone and give safe support to clinicians in detection and decision making. Nowadays, through ML [118], it is possible to analyze data on a large scale and with different algorithms, detecting patterns and models in a very short period of time. In this way, there is a significant improvement in diagnostic methods using techniques which are even imperceptible to human experience and reasoning. On the top of that, these days the world of Machine Learning is more advanced than ever before thanks to the deep learning. Simply explained, deep neural networks enable the creation of systems which are powerful enough to represent any finite

deterministic mapping between any given set of inputs and a set of corresponding outputs. These networks allow powerful data processing, allowing processes as complex as image identification or natural language processing. However, developing a system that accurately classifies AD despite its different stages from normal controls using MRI in a very short time is a challenging task.

Research on convolutional neural networks has resulted in outstanding algorithms such as LeNet [84], AlexNet [74], VGGNet [123], GoogLeNet [126], ResNet [55] and DenseNet [59]. Recent studies have shown that deep neural networks can significantly improve the quality of image classification. Current researches tend to use deeper convolutional neural networks to enhance performance. However, blindly increasing the depth of the network cannot ameliorate the network effectively. Worse still, with the depth of the network increases, more problems occurred in the training process and more training tricks are needed. Among the problems is the fact that as the depth of the network increases, gradient disappearance influences classification accuracy and the corresponding increasing number of parameters to be learned increases the possibility of overfitting, especially when only a small amount labeled samples are acquired for training. Many researchers have employed residual networks (ResNets) [55] to solve these problems. Although ResNets have been proven to be effective, they exhibit problems during application. For example, a ResNet connects input and residual feature maps by summation in the skip connection stage, which might clog the flow of information. Further, a ResNet generates numerous feature maps in each layer and too many parameters to be trained may make training more difficult. If the features could be fully reutilized, only a small amount of feature maps would be needed, which would reduce redundancy and the number of parameters, thereby preventing overfitting. The connection between input and output layers can also be reduced to make each layer accept the inputs from all previous layers, in order to reduce the influence of gradient disappearance while maintaining the same expression capacity of the network, making deeper networks perform better classification. Huang [59] takes this idea to the extreme. He proposed DenseNet, for the lth layer of which, it takes all previous $l-1$ layers as its input. By this kind of network structure design, it not only alleviates vanishing gradient problem, but also achieves better feature reuse. DenseNet achieves superior performance. However, it has its own disadvantages. There are total $L(L-1)/2$ connections for a L layer DenseNet. The excessive connections not only decrease networks computation efficiency and parameter efficiency, but also make networks more prone to overfitting. Also, this architecture uses a smaller filter size (3x3). Hence, filters with receptive fields of the same size are unlikely to capture characteristics of pixels at different scales and smaller filter size need a higher processing time to train the network compared to larger filter size.

The objective of this work is presented below in the form of a research question.

RQ 3.a How can the DenseNet model be improved to accurately diagnose Alzheimer's disease? How can be decrease computation time and parameters?

RQ 3.b What are the key factors in adopting deep CNNs to learn effective feature representations for this image classification task?

RQ 3.c How can learn features with different scales? How can overfitting be avoided?

1.2.4. Problem definition: Classification of COVID-19 from chest X-ray images

From the end of 2019, a new coronavirus namely COVID-19 [142], was confirmed in human bodies as a new category of diseases that cause dangerous respiratory problems, heart infection, and even death. This virus is a pandemic infectious disease caused by severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) which has now become a global issue. It has created a worrying situation around the world because there are no vaccines or specific treatments for it. As it is able to infect people easily and can rapidly spread from person-to-person in an efficient and sustained way, the quick identification and isolation of the affected person is the very first step to fight against this virus. Polymerase chain reaction (PCR) [99] is the main method and gold standard for detecting

COVID-19 cases which can detect it from respiratory specimens (collected through a variety of means such as nasopharyngeal or orpharyngeal swabs). Though this technique is the most effective one, it is manual, complicated, laborious and time-consuming process. Moreover, there is a significant shortage of its supply, which leads to delay in the disease prevention efforts. There are many countries of the world are suffering with incorrect count of the number of patients with COVID-19, which is due to not all the patients being tested and also due to delay in the test results. These delays can lead to infected patients interacting with the healthy patients and infecting them in the process. Because of the limited availability of COVID-19 testing kits, we have to depend on different determination measures. Since COVID-19 assaults the epithelial cells that line our respiratory tract, we can utilize radiological imaging techniques such as chest X-rays and computed tomography (CT) to detect and diagnose COVID-19 to investigate the strength of a patient's lungs. In this work, chest X-rays is preferred over CT scan. The reason behind this is that X-rays machines are available in most of the hospitals and cheaper than the CT scan machine. Besides this, X-rays have low ionizing radiations than CT scan [81]. COVID-19 reveals some radiological signatures that can be easily detected through chest X-rays. The advantage of having X-rays with the scanner is that in some cases the disease can be detected in its preliminary stages, even if the screening is still negative and its drawbacks is that an x-ray analysis requires an expert in radiology, and it takes a long time; which is precious when people are sick around the world. Furthermore, it is difficult for radiologists to distinguish COVID-19 from other viral pneumonias. The symptoms of COVID-19 being similar to viral pneumonia can sometimes lead to wrong diagnosis in the current situation, where hospitals are overloaded and working round the clock. Therefore, incorrect diagnosis can lead to a non-COVID viral Pneumonia being falsely labeled as highly suspicious of having COVID-19 and thus delaying in treatment with consequent costs, effort and risk of exposure to positive COVID-19 patients. Hence, there is a need to automate analysis of chest X-rays to save medical professional's valuable time. The automatic analysis of chest X-rays can be done through deep learning-based approaches, which may accelerate the analysis time. These approaches can train the weights of networks on large datasets [57]. However, these approaches applied to chest X-rays are very limited [63]. Hence, the motive of this work is to develop an automated deep learning-based approach for the detection of infection in chest X-rays. Convolutional neural networks are a powerful tool which is extensively utilized for image classification. The hierarchical structure and efficient feature extraction characteristics from an image make CNN a dynamic model for image classification. In particular, an efficient non-linear mapping by increasing the depth and wide of the network can be learned between the low-resolution input images to high-resolution target image. However, this will lead to a substantial increase in network parameters, requiring the massive amount of training data to prevent overfitting. From literature review, Kumar et al. [76] used various convolutional neural network (CNN) models. Their study states that the ResNet model [55] provided the best performance despite the small number of images used in their investigation. This model ignores the full use of different receptive field (RF) size and therefore, achieve relatively low performance. However, tuning some critical hyperparameters of a CNN such as the receptive field size presents a major challenge due to the presence of features with different scales. The RF is the region of the input space that affects a particular unit of the CNN. Covering different RFs is important to recognize features with different scales and sizes at a specific layer of the CNN. Fixing the RF size on CNNs is an inefficient assumption. This is because the visual cortex can collect information with different scales at the same processing level. If the RF size is selected to be too large. Thus, most of the information captured is irrelevant. If the RF size is selected to be small, it does not capture holistic information. In both cases, the classification accuracy can be significantly reduced.

The objective of this work is presented below in the form of a research question.

RQ 4.a How can the Resnet model be improved to accurately differentiate chest X-ray images of COVID-19 and other diagnostic groups?

RQ 4.b How can learn features with different scales?

1.3. Thesis Objectives

This thesis seeks to address the above multifaceted challenges arising in classification and segmentation of medical images through a set of new methods. The goals of the thesis are thus fourfold.

The first objective addresses the problem of Brain tissue MRI segmentation that is an important issue for discovering the brain structure and diagnosis of subtle anatomical changes in different brain diseases. In particular it addresses the sensitivity of FCM based kernel method [19] to the initial clustering centers and noise data, and the intensity inhomogeneity. Besides dealing with the highest computational time when we add spatial terms and additional terms to the FCM based kernel method to reduce noise.

The second objective addresses the issue in clustering method that is the identification of the number of clusters in a given dataset and addresses the problem of finding it in a dataset having overlapping clusters. In particular, addresses the problem of V_{CVI_2}[4] validity index in only using membership grades which suffers from the absence of information about geometric properties of data. Moreover, it addresses the limitation of the classical indices in their ability to compute the compactness and the separation by ignoring the overlap between clusters, which leads to an incorrect validation result. Thus, we propose a new cluster validity index by considering geometrical properties and membership degrees of data, to discover this number from partitions generated by a fuzzy clustering algorithm. Furthermore, we examine the goodness of data clustering results using medical images that contain overlapping segments.

The third objective addresses the shortcomings of DenseNet [59] which are (1) the excessive connections that decrease networks computation efficiency and parameter efficiency, and make networks more prone to overfitting. (2) the uses of a smaller filter with receptive fields of size (3x3), which is unlikely to capture characteristics of pixels at different scales and needs a higher processing time to train the network compared to larger filter size. Thus, we develop a deep learning model based on convolutional neural network which is wider and has a suitable depth and more shorter paths for efficient gradient flow and that is able to classify automatically MR images as either normal or with the Alzheimer's disease, which improves DenseNet model in terms of classification accuracy and computation time in order to help clinicians in the early diagnosis and detection of AD.

The fourth objective is to propose a CAD scheme using deep CNN as an advanced tool to assist radiologists to automatically diagnose COVID-19 in chest X-ray images and differentiate it from non–COVID-19 and viral pneumonia disease. This model is developed to enhance the ResNet [55] model which ignore the full use of different levels of features.

1.4. Thesis Contributions

Based on the research questions and objectives raised above, a number of contributions were made.

As the First contribution, we address the problem of unsupervised and automatic segmentation of medical images corrupted with noise and intensity inhomogeneity as well as address the problem of clustering performance affected by initial centers of clusters and the higher computation time produced by some clustering methods. We introduce a new objective function called MKPE for segmentation of medical images with appropriate initialization algorithm. The new algorithm has been proposed using nonlinear transformations from the original pattern space into a higher dimensional feature space with properties of kernel functions and modeling the bias field to control its effect so that tissue properties can be retained leading to accurate classification as well as using a local filtering to considerably reduce the noise and the amount of calculations performed during the segmentation process.

As the second contribution, we address the issue of assessing the quality of the clusters or segments found by fuzzy clustering algorithms. In particular, it seeks an answer to the question "Can the appropriate number of clusters be determined automatically? And if the answer is yes, how?" We also extend the analysis to the problem of overlapping clustering, where items can simultaneously belong to more than one cluster. In this context, we propose a newly validity index for fuzzy clustering called V_{ECS} to validate the result of fuzzy clustering algorithm and to recognize the optimal number of clusters for a given data set containing overlapping clusters. Its principle is based on the evaluation of a fuzzy compactness separation ratio. A real number is assigned to this ratio by combining two functions. The first function involves the structure of the dataset itself. The second function uses only the properties of the membership degrees. In this work, two types of medical images contain overlapping regions or clusters are used to validate the proposed index including, simulated brain MRI and leukemia images.

As the third contribution, we proposed to design a deep learning-based diagnostic method for identifying Alzheimer's disease, which could assist doctors to reduce workload while screening the disease for existing brain MR images and could be used for clinical decision-making process to detect in a very short time AD from normal controls despite its different stages using brain MRI. In particular, we propose a deep convolutional neural network model called IRDNet for AD classification that learned features directly from the input of brain MRI and eliminated the need for the handcrafted feature generation. In particular we develop DenseNet model [59] to overcome its drawbacks that are (1) the excessive connections which decrease networks computation efficiency and parameter efficiency, also make networks more prone to overfitting. (2) the use of a smaller filters with receptive fields of the same size which are unlikely to capture characteristics of pixels at different scales and needs a higher processing time to train the network compared to larger filter size.

As the fourth contribution and by the motivation of the urgent need to develop solutions to aid in the fight against the COVID-19 pandemic, we propose a deep CNN model, namely CVDNet as an advanced tool to help radiologists to automatically detect COVID-19 infection and thus delay the rapid spread of this virus. The proposed model is developed to distinguish more precisely the patients affected by COVID-19 from healthy persons and viral pneumonia patients using chest X-ray images. In particular, we develop ResNet [55] model by aggregating the multiscale information gathered by the multicolumn convolutional neural network to improve performance.

The main contributions of this thesis are summarized as follows:

a. Segmentation of brain tissue from MRI corrupted by noise and intensity inhomogeneity.
    - Proposing a fuzzy clustering method called MKPE with appropriate initialization algorithm for automated segmentation of brain tissue from MRI corrupted by noise and intensity inhomogeneity.
    - Examining an important theoretical question regarding the mathematical properties behind the proposed fuzzy MKPE clustering algorithm such as convergence.
    - Comparing the proposed MKPE method to state-of-the-art segmentation approaches in terms of accuracy and execution time.

b. Determination of the number of clusters and evaluation of the quality of the partitions produced by fuzzy clustering algorithm
    - Proposing a cluster validity index called V_{ECS} that helps to find the optimal number of clusters/segments of data from partitions generated by a fuzzy clustering algorithm such as the Fuzzy C-Means (FCM), aiming to validate overlapping clusters.

- Evaluating empirically the proposed index with the state-of-the-art approaches to highlight the effectiveness of V_{ECS} in identifying the correct number of segments using brain MR and leukemia images.

c. Classification of Alzheimer's disease from brain MRI
   - Proposing a deep convolutional neural network model called IRDNet to accurately classify AD from normal controls in a very short time.
   - Evaluating empirically our model with the state-of-the-art approaches to highlight the effectiveness of IRDNet in terms of execution time and in detecting AD using brain MR images.
   - Helping clinicians in the early diagnosis and detection of Alzheimer's disease

d. Classification of COVID-19 from chest X-ray images
   - Building deep convolutional neural network model called CVDNet to accurately detect patients with COVID-19 in a very short time in order to assist in the early diagnosis.
   - Performing an empirical analysis of the proposed CVDNet model in order to classify COVID-19 disease from normal and other pneumonia cases using chest X-ray images since it is less costly than other imaging modalities.
   - Evaluating the performance of the proposed CVDNet compared to other existing models using X-ray images dataset.
   - Helping researchers continue to develop artificial intelligence techniques to fight the COVID-19 epidemic.

1.5. Thesis Organization

The remainder of this thesis is structured as follows. Chapter 2 presents a theoretical introduction in each of the topics to be addressed in this thesis, Chapters 3, 4 and 5 start with a review of the relevant literature followed by our contributions, experimental setup, results and discussions. Chapter 6 draws conclusions and future work.

- **Chapter 2 - Preliminaries**: deals relevant background knowledge for the research presented in this thesis
- **Chapter 3 - Segmentation of Brain Tissue from MRI**: provides a brief introduction to MRI. Also covers the literature review of the related previous works which tackle the problem of medical image segmentation with noise and intensity inhomogeneity. Afterwards, it introduces a new approach called MKPE with appropriate initialization algorithm for segmentation of brain tissue from MRI corrupted by different noise and intensity inhomogeneity levels. Finally, the performance of the proposed MKPE method is evaluated in terms of segmentation accuracy and computation time compared to other recently published approaches.
- **Chapter 4 - Cluster Validity Indices in Medical Image Segmentation**: provides the concepts of cluster validity indices and reviews the related literature works which address the problem of identifying the appropriate number of segments on overlapped data sets. Afterwards, it introduces a new cluster validity index V_{ECS} to recognize the optimal number of clusters/segments adapted to the fuzzy c-means algorithm. After this, experiments setup is carried out to assess the performance of the proposed V_{ECS} on medical imaging data sets. Finally, results of these experiments and discussion are shown as well as comparisons with other works are also conducted in this chapter.
- **Chapter 5 - Deep CNNs for Medical Image Classification:** starts with an explanation of the basic functionality of deep Convolutional Neural Network. This introduction is followed by two parts. The first part begins with an introduction to Alzheimer's disease diagnosis as well as presents a review of the current state-of-the-art of deep CNNs for image-based classification of AD. Next, it provides a novel deep CNN (IRDNet) model to classify Alzheimer's disease from normal controls (NC) using MRI. The second part begins with an introduction to COVID-19 diagnosis as well as presents a review of the current

state-of-the-art of deep CNNs for image-based classification of COVID-19. Afterwards, it provides a novel deep CNN (CVDNet) model to classify COVID-19 infection from normal and other pneumonia cases using chest X-ray images. In each part, after our contributions experiments setup, results, discussions and comparisons with other published works are conducted in this chapter.

- **Chapter 6 - Conclusions and future work:** sums up the works and findings of the research undertaken during this thesis work and discusses the perspectives gained from each contribution.

A summary of this thesis in French is available in Appendix A and references are presented at the end of this thesis

CHAPTER 2

Preliminaries

Contents

2.1. Introduction 12
2.2. Artificial Intelligence 12
2.3. Medical Image Analysis 13
2.4. Machine Learning Framework 14
2.4.1. Machine Learning 14
2.4.2. Supervised Learning Framework 15
2.4.3. Unsupervised Learning Framework 16
2.5. Image Segmentation 17
2.5.1. Clustering in Image Segmentation 18
2.5.2. Cluster Validity Indices 20
2.6. Image Classification 21
2.7. Deep Learning 22
2.8. Conclusion 23

2.1. Introduction

Image processing [87] is a vast field with many applications in a variety of areas from research to industry, from assisting tools for an expert to fully automatic systems; from the automatic sector to medical or biomedical at which we aim in this thesis. Overall, there are also many methods that can be successfully applied across several domains such as image segmentation, object detection and classification.

This chapter presents an overview of the concepts that are the most relevant to this dissertation. The sections of this chapter are organized as follows: Section 2.2 gives the main concepts of Artificial intelligence. Section 2.3 highlights the details about medical image analysis. Section 2.4 provides the concept of machine learning and more specifically supervised and unsupervised learning. Section 2.5 presents an overview of the image segmentation task. We review the technique that was employed to address this task, as well its limitation. Section 2.6 gives an overview of the image classification task. Section 2.7 discusses the general principles of deep learning with emphasis on CNN, we motivate its use for classification task.

2.2. Artificial Intelligence

Field of Artificial Intelligence (AI) is very general and spawns across several different disciplines (e.g., mathematics, computer science, philosophy, economics and even ethics). This fact suggests that this field is very broad and can be tackled from many different viewpoints. Therefore, this description will not be very exhaustive. For broader and more complex dive into subject refer to [117]. One of many possible definitions of AI can be summarized as a pursuit to develop an artificial intelligent agent. In other words, it is an attempt to create intelligent machines that are either thinking or can be perceived as thinking ones. One of the most important abilities of

intelligent agents is the sense of vision. Sense of vision is usually required to certain degree and not always it is necessary that it rivals the capabilities of human visual apparatus. First attempts to solve the vision problems were tackled from so called bottom-up approach in which the system was instructed with hard-coded set of rules describing the vision. It was expected that as the understanding of a mechanisms allowing humans to extract information from visual scene, the hard-coded systems can be fed this understanding and thus more capable systems can be created. Problem with this approach was that it highly underestimated the difficulty of formalization of these rules. As a consequence, it predominantly failed to devise such a system. This insight leads the researchers to postulating that in order to solve the problem of deploying vision capabilities for artificial intelligent system, it is necessary to introduce a process that would allow AI systems to extract patterns from provided data. That is to say, it is an introduction of systems that are able to learn. Process that enables systems to learn is generally called machine learning [118]. Machine learning is again quite extensive term that can be used in multiple different contexts. In this work, it is meant to be understood as technique that is used to create mathematical models used for image recognition. With the omnipresence of numerous data within a large number of science fields, Machine Learning has become a common tool for Data Mining, model building and prediction in a large number of areas such as biology and medicine, mathematics, finance and business, physics, chemistry, marketing and so many others. There are several types of machine learning models that are useful for different tasks. The task that is discussed in this work and is also arguably most commonly tackled is called segmentation. The performance of machine learning algorithms is very often dependent on the way the data is presented to the algorithm, often referred to as the representation of the data. Finding a good representation can be a time-consuming and challenging procedure, sometimes also requiring specific domain knowledge. Deep learning [10] is a subfamily of machine learning which consists of algorithms that are capable of extracting a useful representation automatically from raw data, many of which have been known in the machine learning community since the late 1960's. However, deep models have, historically speaking, been very difficult to train and are associated with a high computational burden. As a result of these limitations, deep learning saw little research compared to other areas of machine learning. However, in 2012 the Super Vision group won the ImageNet Large Scale Visual Recognition Challenge[1], an annual competition for visual recognition tasks, by a remarkably large margin using a deep learning-based approach. Their success spurred an explosion in deep learning research with deep models significantly improving the state-of-the-art in several computer vision tasks such as image classification. These developments have a huge potential for medical imaging technology, medical data analysis, medical diagnostics and healthcare.

2.3. Medical Image Analysis

Medical image analysis is the ability to examine medical issues based on various imaging modalities. It consists of obtaining images from the human body to help doctors in making an accurate diagnosis of possible illnesses. The most common types of medical images include X-rays, MRI, Computerized Tomography, Positron Emission Tomography, and ultrasounds [29], which allow obtaining a visualization of the inside of body parts without surgery. Moreover, some medical images use visible light cameras to capture images from outside of the body, like skin images (dermoscopy), eye fundus, endoscopic images and histological images for analyzing the shape of cells. Radiologists are doctors specialized in analyzing these kinds of medical images. They need a lot of practice to achieve high experience of being capable of providing accurate diagnosis. The current course of action of examining medical images is labor-intensive, time-consuming, expensive, and error-prone. It would be more useful to apply Computer Vision (CV) algorithms [21] capable of extracting high-level information from numerical images. Those algorithms will lead to automatic or semi-automatic systems that can help radiologists by providing clues for more reliable diagnosis and treatment recommendations. This is known as Computer-Aided Diagnosis (CAD) systems. Typically, three types of tasks can be performed to analysis medical images: classification [30], detection and segmentation. For this thesis, we attempt to cover the possible solutions to the task of classification and segmentation with machine learning.

Image classification is the task of recognizing the content of images and assigning class labels accordingly. In medicine, it often means to give diagnostic labels or determine the type of lesions in medical images. For many

[1] http://www.image-net.org/challenges/LSVRC/2012/results.html

medical problems, such as the classification of skin disease, nodular chest X-ray, Alzheimer's disease, and vertebrae, where the classification requires a considerable level of expertise or is error-prone, a computerized solution is demanded. Deep learning has its advantages in automating the classification of medical images. Due to the ability to learn hierarchical feature representations, deep learning methods are surprisingly better at learning intrinsic structures from images.
Segmentation [58] is a pixel-/voxel-level classification of images. It aims to delineate the boundaries of target objects or regions. Similar to the detection problem in medicine, medical image segmentation also has two types: 1) organ or substructure segmentation and 2) lesion or nodule segmentation. Unlike the detection problem, segmentation is required to provide not only the location of the object or region of interest (ROI) but also their boundaries. This more precise information allows for quantitative analysis of clinical parameters. For example, organ segmentation may be used for structural related analysis such as volume and shape. Lesion segmentation may be used for treatment planning and disease progression monitoring. Clinically, different medical images of a patient are usually acquired during the different phases of healthcare delivery. This opens opportunities to fuse, compare and share these different images of the patient to facilitate the diagnosis and/or treatment.

2.4. Machine Learning Framework

In this section, we give some foundations of the Machine Learning framework, where we focus on the supervised and unsupervised learning case.

2.4.1. Machine Learning

Machine learning, a subset of artificial intelligence, is a field of Computer Science that is focused on the construction of a mathematical model using sample data – also known as "training data" – to make decisions (predictions) without being specifically programmed. A machine learning task aims to identify (to learn) a function $f : \mathcal{X} \to \mathcal{Y}$ that maps the input domain $\mathcal{X}$ (of data) into output domain $\mathcal{Y}$ (of possible predictions). Function f is chosen from different function classes, dependent on the type of learning algorithm that is being used. The term machine learning was coined in 1959 by Arthur Samuel, a pioneer in the field of computer gaming and artificial intelligence, who defined it as [118]:

> *"The field of study that gives computers the ability to learn without being explicitly programmed".*

Tom M. Mitchell provided in his book Machine Learning a widely quoted, more formal definition of the algorithms studied in the machine learning field as follows [134],

> *"A computer program is said to learn from experience **E** with respect to some class of tasks **T** and performance measure P if its performance at tasks in **T**, as measured by **P**, improves with experience **E**."*

T, the task refers to what the computer is intended to do after having learned. ***P***, the performance measure is a quantitative measure of how well the model performs. Optimizing the performance measure is one of the main goals of machine learning. Usually, the choice of the metric depends on the type of the task that machine learning system is dealing with. ***E***, the experience describes source of the data which is used to learn.
Machine learning tasks are typically classified into three broad categories [26]. These are: a) supervised learning, in which the system infers a function from labeled training data, b) unsupervised learning, in which the learning system tries to infer the structure of unlabeled data, and c) reinforcement learning, in which the system interacts with a dynamic environment.

- Supervised learning [134]: The model learns a mapping between the given inputs and their outputs (labeled data). These outputs are given by a teacher who shows the model what to do. Two common tasks of supervised learning are classification (discrete outputs) and regression (continuous outputs). Classification is relevant for this work, and will be developed in section 2.6.

- Unsupervised learning [15]: The model learns interesting hidden structure of the data given only the inputs (unlabeled data). Therefore, there is no teacher that guides the learning process. Unsupervised learning is commonly used in tasks such as clustering, where the interest is to group similar items together, and density estimation task, which discover the statistical values that describe a data as well as dimension reduction task, which applied to reduce the number of columns (i.e., features) of data sets in high-dimensional spaces. Clustering is relevant for this work, and will be developed in section 2.5.
- Reinforcement learning [68]: is a general term given to a family of techniques, in which the system attempts to learn through direct interaction with the environment. It is important to mention that the system has no prior knowledge about the behavior of the environment and the only way to find out is through trial and failure (trial and error). Reinforcement learning is mainly applied to autonomous systems, due to its independence in relation to its environment.

In the following two sections, we expand on supervised and unsupervised Learning, as it is the framework used in this dissertation.

2.4.2. Supervised Learning Framework

In the supervised setting [134], we suppose we have a training dataset D_{train} with Ns examples. The dataset consists of data observations $x_i \in \mathcal{X}$, which represents the input that is fed to the system, and the targets $t_i \in \mathcal{T}$, which correspond to the desired output of the model. More formally, we define the dataset as follows:

$$D_{train} = \{(x_i, t_i); i = 1, ..., Ns\} \tag{2.1}$$

Supervised models can be classified as parametric and non-parametric based on the assumptions that should be made to apply them. Parametric models are based on the assumption that the unknown target function $f(.)$ lies in hypothesis space – a set of parameterized functions $\{f(x;\theta) \mid \theta \in \Theta\}$, where Θ is the parameter space. Non-parametric learning models make almost no assumptions about the unknown function of interest, and do not put constraints on the hypothesis space.

In supervised learning, we will restrict ourselves to the parametric case. In this case, the goal of supervised learning is to find a function $f_\theta : \mathcal{X} \to \mathcal{Y}$ such that predictions can be made on unseen data. Therefore, supervised learning can be thought of as function approximation where you have some data and want a mapping to a desired output. To achieve this goal, three main elements have to be defined:

- Family of candidate functions f_θ , parameterized by a vector of parameters $\theta \in \Theta$
- Loss function $E : \Theta \to \mathbb{R}$, which quantifies the mismatch between the outputs predicted by a candidate function f_θ and the ground truth.
- Training algorithm, which minimizes the loss function (with respect to the parameters θ over the training data

Let $f_\theta : \mathcal{X} \to \mathcal{Y}$ be the decision function, parameterized by a bit string θ of parameters that maps the input x to a proper $\mathcal{Y}$, i.e., $f_\theta(x) = y$. The output space $\mathcal{Y}$ can be different from the target space $\mathcal{T}$, in which case a pre-defined function $g: \mathcal{Y} \to \mathcal{T}$ is used to obtain the final prediction of the system.

We will explain in more details the role of g later in this section, when introducing the classification case.

Let $y_i = f_\theta(x_i)$ be the output of the decision function f_θ on example x_i . To measure how different the output y_i is from the true target t_i , we define a loss function $l(y_i, t_i): \mathcal{Y} \times \mathcal{T} \to \mathbb{R}^+$ which assesses the quality of the estimation.

We seek to approximate the predictions $g(y_i)$ as much as possible to the targets t_i. In order to quantify how far off the predictions from the targets for a given training dataset D_{train} are and decision function f_θ. We define the empirical risk as the average of the losses over the training set:

$$E(\theta, D_{train}) = \frac{1}{Ns}\sum_{i=1}^{Ns} l(f_\theta(x_i), t_i) \quad w.r.t \quad \theta \in \Theta \tag{2.2}$$

High values for the empirical risk mean that f_θ does not approximate well the training data, while a risk of zero indicates that the model perfectly describes the relationship between the input examples and the output targets. In order to correctly model the dependencies between the data and the targets, we look for the parameters θ such that the empirical risk over the training data D_{train} is minimized. We call the function that we want to minimize the objective function. Minimizing the objective function does not guarantee that f_θ will perform well in unseen data. For example, whenever f_θ has sufficient capacity, it is possible that the model exactly memorizes the training examples, which are often noisy, and possibly performs poorly in unseen examples, because it starts to model the underlying noise.

Before supervised training, the dataset is split into training set and test set, shown in Fig. 2.1. A simple way for the partition is to split most of the data for training and some of it for testing, for example 80% into training set and 20% into test set. The split is commonly done in stratified manner so that the sets contain the same distribution of different examples in them [38]. As the names imply, the training set is used to train a model, whereas the test set is used for testing the model with previously unseen examples. The ability of a model to predict previously unseen examples correctly is called generalization. When a machine learning problem is being solved, several models may be trained and tested with the data. If the test set is used several times to estimate generalization, it stops serving its purpose. A good generalization performance may have been achieved by chance. In this case, a separate validation set can be used. Validation set is a subset of the training set as shown in Fig. 2.1. It can be used for initial generalization estimation. After training multiple models, the final generalization performance is estimated only with the model.

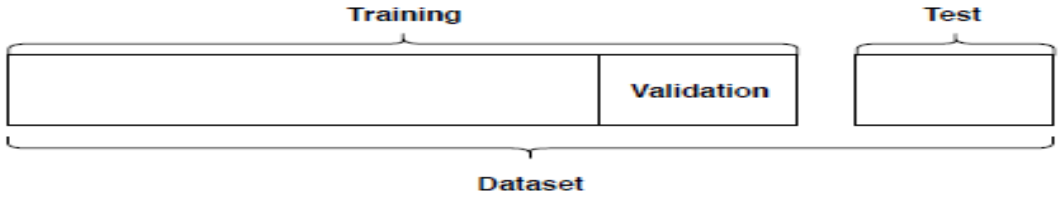


FIGURE 2. 1: Subsets of a dataset used for training a machine learning model

2.4.3. Unsupervised Learning Framework

In unsupervised learning [153], the data set is of the form $D = \{(x_i); i = 1, ..., N\}$. No labels are known and the model has to group given data samples and discover the categories. This problem is quite difficult to evaluate because there can be multiple different groupings.

Unsupervised learning model is training by observing new data and extracting patterns in the date without being instructed on what they are. Opposed to supervised learning described above, the aim of this approach is that the model is able to learn from data without supervision (as the name suggests). This means that there is no need for input data to be annotated; therefore, it takes much less time and resources to deploy these models in practice.

Unsupervised learning models can be further classified into parametric and non-parametric. In the case of parametric models, we assume a parametric distribution of data. It assumes that sample data comes from a population that follows a probability distribution based on a fixed set of parameters. Theoretically, in a normal family of distributions, all members have the same shape and are parameterized by mean and standard deviation. That means if you know the mean and standard deviation, and that the distribution is normal, you know the probability of any future observation. Parametric Unsupervised Learning involves construction of Gaussian

Mixture Models and using Expectation-Maximization algorithm to predict the class of the sample in question. This case is much harder than the standard supervised learning because there are no answer labels available and hence there is no correct measure of accuracy available to check the result. In non-parameterized version of unsupervised learning, the data is grouped into clusters, where each cluster (hopefully) says something about categories and classes present in the data. This method is commonly used to model and analyze data with small sample sizes. Unlike parametric models, nonparametric models do not require the modeler to make any assumptions about the distribution of the population, and so are sometimes referred to as a distribution-free method. In unsupervised models, we will restrict ourselves to the non-parametric case.

2.5. Image Segmentation

Image segmentation is an essential step in computer vision applications. It is considered the first step in image processing. The process of segmentation consists of partitioning a digital image into non-overlapped homogeneous regions with respect to some characteristics, such as gray value, texture, etc. [64], where each region is assumed to correspond to an object in the image. Formally, it can be defined as follows [58].

Definition 2.1. Let F be the set of all pixels and $P()$ be a uniformity (homogeneity) predicate defined on groups of connected pixels, then segmentation is a partitioning of the set F into a set of connected subsets or regions $(S_1, S_2, ..., S_n)$ such that:

$$F = \bigcup_{i=1}^{n} S_i \tag{2.3a}$$

$$S_i \cap S_j = \varnothing \quad \text{for all } i \neq j. \tag{2.3b}$$

$$\text{The uniformity predicate } P(S_i) \text{ is true for all regions } S_i \tag{2.3c}$$

$$P(S_i \cup S_j) \text{ is false when } S_i \text{ is adjacent to } S_j. \tag{2.3d}$$

The indices i and j lie in the interval $[1, c]$ and c is the number of subclasses.

This list of conditions for a partition to be segmentation does not lead to a unique algorithm for calculating segmentations, but does suggest many important aspects of such algorithms.

The first condition simply requires that the segmentation be complete: Every picture point must be in a region. The second condition implies that regions must be connected. The third condition determines what kind of properties the segmented regions should have, for example, uniform gray levels. The fourth condition expresses the maximality of each region in the segmentation.

Image segmentation is used in various applications like medical imaging, locating objects in satellite images [75], face recognition, traffic control systems, and machine vision, etc. Medical 2D or 3D image segmentation is an important image processing step in medical image analysis. Segmentation methods with high precision (including high reproducibility) and low bias are a main goal in surgical planning because they directly impact the results, e.g., the detection and monitoring of tumor progress. Warfield et al. [141] denoted the clinical importance of better characterization of white matter changes in the brain tissue and showed that particular change patterns in the white matter are associated with some brain diseases. Accurately recognizing the change patterns is of great value for early diagnosis and efficient monitoring of diseases. Therefore, assessing the accuracy and the quality of segmentation algorithms is of great importance. Medical 2D or 3D images are defined on a 2D or 3D grid that can have different sizes depending on the body parts imaged and the resolution. The grid size is given as $(w \times h \times d)$ denoting the width, height, and depth of the 2D or 3D image. Each 2D and 3D point on the grid is called a pixel and voxel, respectively. Given an anatomic feature, a binary segmentation can be seen as a partition that classifies the pixels/voxels of an image according to whether they are part or not of this feature. Examples of anatomic features are white matter, gray matter, lesions of the brain, body organs and tumors.

Several techniques for image segmentation have been proposed. They can be classified into thresholding [77], edge detection [69], region extraction [157] and clustering [15]. Clustering technique can be used for segmenting

image data as they are used for partitioning large datasets into groups according to the homogeneity of data points. The following section details the concepts of this technique.

2.5.1. Clustering in Image Segmentation

Clustering [15] is a powerful technique that has been reached in image segmentation. Clustering is a fundamental task in data mining, in which a finite set of clusters is identified so as to classify the pixels in a digital image. It can be considered as one of the most important unsupervised learning problems. There is an increasing interest in the use of clustering methods in machine learning, pattern recognition, image processing and information retrieval. It has also a rich history in other disciplines. In clustering, the number of clusters is known a priori and image pixels are grouped into appropriate clusters based on the principle of intra-cluster similarity maximization or inter-cluster similarity minimization, i.e., similar pixels are grouped together, while different pixels belong to different groups.

Clustering based segmentation algorithms can be divided into two broad categories, which are hard clustering and soft clustering. Hard clustering methods are used for datasets with sharp boundaries between clusters and a pixel belongs to only one cluster. Soft clustering algorithms are used when there are no hard boundaries amongst objects in an image and they introduce fuzziness by partly allocating pixels to all clusters with different degrees of membership.

The objective of clustering is to partition the data set of N points $X = \{x_1, ..., x_N\}$ into c hard clusters, c is an integer, $2 \le c < N$. For the time being, assume that c is known, based on prior knowledge, for instance, or it is a trial value, of which partition results must be validated. Using classical sets, a hard partition can be defined as a family of subsets $\{C_i \mid 1 \le i \le c\} \subset P(X)$, ($P(X)$ is the power set of X that is, of the set of hard subsets of X) with the following properties [15]:

$$\bigcup_{i=1}^{c} C_i = X \tag{2.4a}$$

$$C_i \cap C_j = \varnothing, 1 \le i \ne j \le c \tag{2.4b}$$

$$\varnothing \subset C \subset X, 1 \le i \le c_i \tag{2.4c}$$

These conditions mean that the subsets C_i contain all the data in X, they must be disjoint and none of them is empty nor contains all the data in X. If $c = N$, each C_i is necessarily a singleton, $C_i = \{x_i\}$, $\forall i$ since this is a trivial case, the range of c is usually $2 \le c < N$.

In terms of their function-theoretic duals, $0 \leftrightarrow \varnothing$ and $1 \leftrightarrow X$, properties (2.4) are equivalent to:

$$\bigvee_{i=1}^{c} u_{C_i} = 1 \tag{2.5a}$$

$$u_{C_i} \wedge u_{C_j} = 0, \; 1 \le i \ne j \le c \tag{2.5b}$$

$$0 < u_{C_i} < 1, \; 1 \le i \le c \tag{2.5c}$$

In equations (2.5) u_{C_i} is the characteristic function of C_i, that is,

$$u_{C_i}(x_j) = \begin{cases} 1, & x_j \in C_i \\ 0, & x_j \notin C_i \end{cases} \tag{2.6}$$

In Eq. (2.5a) and Eq. (2.5b) $\wedge$, $\vee$ are the minimum and maximum operators, respectively. The empty set $\varnothing$ is the constant function $0 \in P(F)$, $0(x) = 0 \;\; \forall x \in X$; the whole set X is the constant function $1 \in P(F)$, $1(x) = 1, \; \forall x \in X$; with $P(F)$ is the set of all characteristic functions on X.

To simplify the notations, we use u_i instead of u_{C_i}. Furthermore, it is extremely convenient to denote $u_i(x_j)$ as u_{ij}.

In terms of membership (characteristic) functions, a partition can be conveniently represented by the partition matrix $U = [u_{ij}]_{c\times N}$. The ith row of this matrix contains values of the membership function of the ith subset C_i of X. It follows from Eq. (2.5) that the elements of U must satisfy the following conditions:

$$u_{ij} \in \{0,1\},\ 1 \le i \le c,\ 1 \le j \le N \tag{2.7a}$$

$$\sum_{i=1}^{c} u_{ij} = 1,\ \ 1 \le j \le N \tag{2.7b}$$

$$0 < \sum_{j=1}^{N} u_{ij} < N, 1 \le i \le c \tag{2.7c}$$

Definition 2.2. Hard c-Partition. Let $X = \{x_1, ..., x_N\}$ be a finite set and let $2 \le c < N$ be an integer. Hard c-partition space for X is the set

$$M_{hc} = \left\{ U \in \mathbb{R}^{c\times N} \mid u_{ij} \in \{0,1\}, \forall i, j; \sum_{i=1}^{c} u_{ij} = 1, \forall j; 0 < \sum_{j=1}^{N} u_{ij} < N, \forall i \right\} \tag{2.8}$$

For convenience in referencing, we remind the reader how the entries of $U \in M_{hc}$ are interpreted, and what each of the three conditions means. Row i of U, say $U_{(i)} = (u_{i1}, u_{i2}, ..., u_{iN})$ exhibits (values of) the characteristic function of the ith partitioning subset of X: $u_{ij} = u_i(x_j)$ is one or zero, according as x_j is or is not in the ith subset; $\sum_i u_{ij} = 1$, means each x_j is in exactly one of the c subsets; and $0 < \sum_j u_{ij} < N, \forall i$ means that no subset is empty, and no subset is all of X: in other words, $2 \le c < N$.

Hard clustering methods produce exhaustive partitions of the dataset into nonempty and disjoint subsets. Such an allocation may be inadequate in the presence of points which are equidistant from the two or more clusters. These data points can represent hybrid shapes or a mixture of objects that are also similar to two or more shapes. A hard partition forces all these points to be assigned to one of the clusters, but they must also belong to all the clusters. To overcome this drawback, certain "crisp" cluster analysis techniques make it possible to predict overlapping clusters. Then, each object must be assigned to at least one cluster, but with the possibility of simultaneous assignment to other clusters. Therefore, fuzzy clustering [116] allows the progressive membership of objects to clusters in $[0,1]$. This gives the flexibility to express the belonging of objects to several clusters at the same time. In addition to assigning objects to clusters with different degrees of membership, the degree of membership can also express the level of ambiguity or certainty of belonging of an object to a cluster. The concept of these degrees of membership is proven by the definition and interpretation of fuzzy sets.

Definition 2.3. Fuzzy Set. A fuzzy set A on universe (domain) X is defined by the membership function $u_A(x)$ which is a mapping from the universe X into the unit interval: [116]

$$u_A(x): X \to [0,1] \tag{2.9}$$

Fuzzy set theory allows for a partial membership of an element in a set. If the value of the membership function, called the membership degree (grade), equals one, x belongs completely to the fuzzy set. If it equals zero, x does not belong to the set. If the membership degree is between 0 and 1, x is a partial member of the fuzzy set. In the fuzzy set literature, the term crisp is often used to denote non fuzzy quantities, e.g., a crisp number, a crisp set, etc. Fuzzy partition can be seen as a generalization of hard partition, it allows u_{ij} attaining real values in $[0,1]$. Conditions for a fuzzy partition matrix, analogical to Eq. (2.7) are given by [116]:

$$u_{ij} \in [0,1],\quad 1 \le i \le c,\quad 1 \le j \le N \tag{2.10a}$$

$$\sum_{i=1}^{c} u_{ij} = 1, \quad 1 \le j \le N \tag{2.10b}$$

$$0 < \sum_{j=1}^{N} u_{ij} < N, \quad 1 \le i \le c \tag{2.10c}$$

Definition 2.4. Fuzzy c-Partition. Let $X = \{x_1, x_2, ..., x_N\}$ be a finite set and let $2 \le c < N$ be an integer**.** Fuzzy c-partition space for X is the set.

$$M_{fc} = \left\{ U \in \mathbb{R}^{c \times N} \mid u_{ij} \in [0,1], \forall i, j; \sum_{i=1}^{c} u_{ij} = 1, \forall j; 0 < \sum_{j=1}^{N} u_{ij} < N, \forall i \right\} \tag{2.11}$$

Row i of a matrix $U \in M_{fc}$ exhibits (values of) the ith membership function (or i^{th} fuzzy subset) u_i in the fuzzy c-partition U of X . Because each column sum is 1, the total membership of each x_j in X**.** is still 1, but since $0 \le u_{ij} \le 1, \forall i, j$, it is possible for each x_j to have an otherwise arbitrary distribution of membership among the c fuzzy subsets $\{u_i\}$ partitioning X**.** There may, of course, be one or more columns of U which assign all of the membership of some x_j to a single u_i ; indeed, M_{hc} is clearly a finite subset of M_{fc} , i.e. $M_{hc} \subset M_{fc}$

2.5.2. Cluster Validity Indices

Clustering algorithms [15] with different cost functions give different solutions, and there is no single best choice of the algorithm and the cost function for all possible data sets. The task is therefore to select the best possible clustering method for a data set. For most clustering algorithms, the number of clusters is set as a parameter. However, the number of clusters is initially not available for most data sets, so determining this number is essential. After the correct clustering algorithm and the number of clusters have been selected, evaluating the clustering results on different parameter settings needs to be addressed. These problems are all related to the cluster validity analysis. There are numerous ways of evaluating cluster validity, each having a different stance on what a "good" cluster is. The approaches of evaluating cluster validity can be roughly divided into two groups: internal and external evaluation.

In internal evaluation [25], only internal data is used in the evaluation process that is, solely the input vectors and the clusters found by the algorithm. The quality of the clustering is usually determined by (in some way) calculating the compactness and separation of the clusters: the clusters are compact, if the members of one cluster are close to each other; the clusters have a high separation, if the members of different clusters are very different. There are a number of different validity indices [119], which try to calculate a score out of these properties. On the contrary, the external indices [145] validate the clustering result with a reference to the partition that is known beforehand, so in the real-world applications, one should use the internal indices to measure how well do the obtained partitions fit the input data.

In the internal evaluation process, validity indices, which represent the compactness and separation of the found clusters, are computed. Using these indices, the metaparameters like the number of clusters can be evaluated. To do this, the clustering algorithm is executed with various values of the meta-parameter, which one wants to determine. Afterwards, for every result the validity index is computed and the clustering with the best index is selected. The value of the meta-parameter of this selected clustering is therefore recommended. However, one validity index might be biased against one clustering algorithm, if both use similar definitions of compactness and separation. Thus, internal validity indices can be applied to learn, in which contexts a clustering algorithm works better than in others (i.e., tweaking the meta-parameters). In contrast, comparing two different clustering algorithms in order to evaluate which algorithm works better on the given data is not possible using only internal

validity indices. Because strict and fuzzy clusters are fundamentally different, there are different indices for both types of clustering.
Both internal and external indices are used for determining the number of clusters. We will restrict ourselves to the internal indices.

2.6. Image Classification

Image Classification is a common Machine Learning task is a subcategory of supervised learning where the goal is to learn a mapping from images to class labels. The input images could be represented by pre-extracted feature vectors or image pixels directly. In the classification setting, we suppose that each image belongs to a discrete number of classes, and the goal of the model is to be able to predict in which class the observation belongs to. There are perhaps two main types of classification tasks: binary and multi-class classification.
A binary classification task is the one that given *n*-dimensional examples and their corresponding labels $(x_1,t_1),...,(x_N,t_N)$ with $x_i \in \mathbb{R}^n, t_i \in \mathcal{T}, \forall i$ and $\mathcal{T} \subset \mathbb{R}^2$, constructs a classifier $f(x)$ that predicts the binary label t of a new unseen example x. An example of binary classification is medical diagnosis for a single medical condition (say disease vs. no disease). Multi-class classification is the problem of classifying instances into one of three or more classes. In our work, we focus on binary and multi-class image classification.
Let N_c be the number of classes that each observation can take and Ns the number of training examples in the training dataset D_{train}. We define $\mathcal{T} = \{1,...,N_c\}$ as the space of possible labels. Let the space of possible inputs $\mathcal{X} \subset \mathbb{R}^{N_D}$ be a subset of the N_D - dimensional Euclidean space, with N_D the dimensionality of the input space, and let $\mathcal{Y} \subset \mathbb{R}^{N_c}$ be the output space of f^{θ}. As before, we consider $y_i = f_\theta(x_i)$ being the output of the decision function for input x_i. One loss commonly used in for training classification models is the cross-entropy loss, which is defined as follows [5]:

$$l(y_i, t_i) = -t_i^T \log(y_i) \tag{2.12}$$

where $t_i \in \{0,1\}^{Nc}$ represents the label of the ith sample and $y_i \in [0,1]^{Nc}$ is the predicted probabilities with Softmax.
The softmax function is defined as

$$soft\max(x)_j = \frac{\exp(x_j)}{\sum_{k=1}^{Nc} \exp(x_k)} \tag{2.13}$$

The output of the softmax function can be seen as converting the input vector x such that it is interpretable as a probability distribution, as all the entries are positive (because of the exponential) and sum to 1.
The classification model assigns a score $f_\theta(x_i)_k$, with $k = 1,...,N_c,$ to each of the classes in $\mathcal{T}$ for each observation x_i.

To obtain the predicted class from the scores given by f_θ, we define the conversion function $g: \mathcal{Y} \to \mathcal{T}$ as $g: x \mapsto \arg\max_k x_k$. Thus, the predicted class $\hat{t}_i$ is the one with the highest score, and can be obtained via:

$$\hat{t}_i = \underset{k \in \{1,...,N_c\}}{\arg\max}\ f_\theta(x_i)_k \tag{2.14}$$

The network parameters can achieve $\hat{\theta} = \arg\min_\theta E(\theta, D_{train})$ by training the cross-entropy loss function, where $y = f_\theta(x),\ f_\theta$ is the network's parameter and x is the training sample.

2.7. Deep Learning

Traditional machine learning methods struggle to perform as the dimensionality of the input data increases. The dimension of a dataset corresponds to the number of attributes/features that exist in a dataset. A dataset with a large number of attributes, generally of the order of hundred or more, is referred to a high dimensional data. Some of the difficulties that come with high dimensional data manifest during analyzing or visualizing the data to identify patterns and some manifest while training machine learning models due to high dimensional data. This problem is referred to as "Curse of Dimensionality [121]. Traditional methods require manual feature engineering of raw data in order to decrease the input dimensions. However, in practice, engineering these features from a mass of data is hard without specific domain expertise. This is where deep learning steps in.

Deep learning [10] solves the curse of dimensionality by learning hierarchical representations of the data with complex models of multiple layers. This is achieved with representation learning. In representation learning, no feature engineering is used. Instead, a model learns by itself which features of the mass of data are relevant for the task. When a deep learning model is trained, the model learns abstract representations of the data by observing the data patterns. The first layers of a deep learning model learn more simple features of the data, whereas the last layers learn more specific features of the data. Why has deep learning become popular? Deep learning models require considerable amounts of data for training. Nowadays, during the era of big data, increasingly more data is being generated. This is one reason for the change. The more there is data, the better it is for model generalization. Deep learning models are also much greater in size than traditional machine learning models. The greater size increases their modeling capacity and enables them to solve very complex problems but increases their computational burden. Because of the computational burden, use of deep learning was impossible before. This has changed. Compared to computation capacity in 1990s, the computation capacity of a modern graphics processing units is over a million times faster [45]. In addition to the computation development, the technologies and software used for deep learning have made great progress.

Most of deep learning, as machine learning in general, is performed using supervised learning. Deep learning models have been studied to outperform traditional machine learning methods in many different tasks. Some of these tasks include speech recognition, object recognition and object detection. In practice, deep learning is almost always performed using deep neural networks. The main particularity of deep learning is the nature of the considered candidate functions. The term deep is related to multiple compositions of functions. The considered composed functions are differentiable and organized in layers, with the idea to progressively transform the input vector, extracting more and more complex information. The term neural network is related to the considered family of functions, represented typically by a graph. Training of the model (minimization of the loss function) is typically based on iterative optimization. Convolutional Neural Networks [86] are a commonly used type of neural networks for image classification. They exploit spatial relations between pixels (or voxels, in 3D) and are based on application of local operations such convolution and pooling (maximum, average). Units in a convolutional layer are organized in feature maps, within which each unit is connected to local patches in the feature maps of the previous layer through a set of weights called a filter bank. The result of this local weighted sum is then passed through a non-linearity. All units in a feature map share the same filter bank. Different feature maps in a layer use different filter banks. The reason for this architecture is twofold. First, in array data such as images, local groups of values are often highly correlated, forming distinctive local motifs that are easily detected. Second, the local statistics of images are invariant to location. In other words, if a motif can appear in one part of the image, it could appear anywhere, hence the idea of units at different locations sharing the same weights and detecting the same pattern in different parts of the array. Mathematically, the filtering operation performed by a feature map is a discrete convolution, hence the name. Although the role of the convolutional layer is to detect local conjunctions of features from the previous layer, the role of the pooling layer is to merge semantically similar features into one. Because the relative positions of the features forming a motif can vary somewhat, reliably detecting the motif can be done by coarse-graining the position of each feature. A typical pooling unit computes the maximum of a local patch of units in one feature map (or in a few feature maps). Neighboring pooling units take input from patches that are shifted by more than one row or column, thereby reducing the dimension of the representation and creating invariance to small shifts and distortions. Two or three stages of convolution, non-linearity and pooling are stacked,

followed by more convolutional and fully-connected layers. A Back-propagating gradient through a CNN is as simple as through a regular deep network, allowing all the weights in all the filter banks to be trained.
The objectives of such design are to limit the number of parameters of the network and to limit computational costs, as images correspond generally to very large inputs. In fact, an important property of the operations used in CNNs is that they can be parallelized and efficiently computed on a Graphical Processing Unit (GPU).
For image classification, CNN has very good performance. Its workflow includes four steps. First, input a collection of images and train the CNN. Then, use convolution to extract features from the image and create feature model. Next, train the model and evaluate the classification results. Finally, save the model for testing.

2.8. Conclusion

Currently, the advances on digital imaging systems and the storage of digital images offer very high-resolution images that can be used in many applications from several areas like medicine. Medical imaging is a fundamental and invaluable tool in modern healthcare. The use of medical imaging has greatly increased, and these massive images provide the opportunities for diagnosis. The ability to automatically analyze, quantify, and categorize the images is an important enabling technology for computer-aided diagnosis system which can be used to aid in patient care. Automated segmentation and disease classification of medical images have shown greater clinical benefits and provided second opinion to diagnosis. Machine learning and deep learning have progressed rapidly in the field of image analysis and have been fully embraced and established as necessary technologies to aid clinical procedures by most medical professionals, particularly with larger datasets and more computational power becoming readily available. At the core of these advances is the ability to exploit hierarchical feature representations learned solely from data, instead of features designed by hand according to domain-specific knowledge. Deep learning is rapidly becoming the state of the art, leading to enhanced performance in various medical applications and solve difficult problems (e.g., image detection, classification, segmentation). Segmentation of brain MRI tissues based on the use of recent machine learning methodologies will be the focus of the next chapter. However, brain MRI are affected by noise and bias field, which make the traditional FCM based kernel algorithm unable to segment tissue regions of MR images accurately. Furthermore, this clustering model is very sensitive to the set of randomly initialized centers. The objective of the following chapter is to address the above problems.

CHAPTER 3

Segmentation of Brain MRI Tissue

Contents

3.1. Introduction 25
3.2. Magnetic Resonance Image Processing 25
3.3. MRI Artifacts 26
3.3.1. Noise in MR Imaging 26
3.3.2. Intensity Inhomogeneity 27
3.4. Segmentation of Brain MRI Corrupted by Noise and Intensity Inhomogeneity. 28
3.4.1. Related Work 29
3.4.2. Fuzzy Clustreing Methods for Brain MRI Segmentation 30
3.4.2.1. Spatially Constrained KFCM for Image Segmentation 31
3.4.2.2. Adaptive Kernel-based Fuzzy C-Means Clustering with Spatial Constraints 32
3.4.2.3. Kernel Generalized Fuzzy C-Means Clustering with Spatial Information 33
3.4.2.4. Improved Kernel Fuzzy Local Information C-Means Algorithm and Outlier Rejection 33
3.4.2.5. Improved Kernelized Rough-Fuzzy C-Means with Spatio-Contextual Information 34
3.4.2.6. Radial Basis Kernel based Intuitionistic Fuzzy C-Means Clustering 35
3.5. The Proposed Method for Automatic Segmentation of Brain MRI Corrupted by Noise and Intensity Inhomogeneity 35
3.5.1. Formulation 36
3.5.2. Convergence Proof of the Proposed Model 41
3.5.3. The Proposed Algorithm 45
3.6. Experimental Setup and Results 48
3.6.1. Datasets 48
3.6.2. Evaluation Metrics 49
3.6.3. Results and Discussion 50
3.6.3.1. MR Brain Image Corrupted with Noise and Intensity Inhomogeneity 50
3.6.3.2. Time Complexity 54
3.7. Conclusion 54

3.1. Introduction

Over the last few decades, the rapid development of noninvasive brain imaging technologies has opened new horizons in analyzing and studying the brain anatomy and function. Enormous progress in accessing brain injury and exploring brain anatomy has been made using magnetic resonance imaging. MRI has been widely employed to examine the anatomical structures of the human brain in both clinical application and neuroscience research [149]. Compared to other medical imaging modalities, MRI has the advantage of the high spatial resolution and well soft-tissue contrasts [140]. This powerful technique can yield exquisite differentiation between different types of tissues, including white matter (WM), grey matter (GM) and cerebrospinal fluid (CSF). The advances in brain MR imaging have also provided large amount of data with an increasingly high level of quality. The analysis of these large and complex MRI datasets has become a tedious and complex task for clinicians, who have to manually extract important information. This manual analysis is often time-consuming due to millions of voxels in the brain MRI image and prone to errors due to various inter or intra-operator variability studies. These difficulties in brain MRI data analysis required inventions in computerized methods to improve disease diagnosis and testing. Nowadays, computerized methods for MR image segmentation, registration, and visualization have been extensively used to assist doctors in qualitative diagnosis.

Brain MRI segmentation is an essential task in many clinical applications because it influences the outcome of the entire analysis. This is because different processing steps rely on accurate segmentation of anatomical regions. For example, MRI segmentation is commonly used for measuring and visualizing different brain structures, for delineating lesions, for analyzing brain development, and for image-guided interventions and surgical planning. This diversity of image processing applications has led to development of various segmentation techniques of different accuracy and degree of complexity.

The challenge in brain MRI segmentation is due to issues such as noise, non-uniform intensity, partial volume effect, shape complexity and natural tissue intensity variations. Under such conditions, incorporation of a priori medical knowledge, commonly represented in anatomical brain atlases by state-of-the-art studies is essential for robust and accurate automatic segmentation. Several brain tissue segmentation approaches have been projected over the years. Some proposals attempt to use clustering techniques.

In this chapter, we will first give some notions about MRI, explaining the principle of acquisition of this type of images, as well as their specificities and some of its drawbacks that lead to inaccurate segmentation. Then we will present a state of the art on methods for the segmentation of brain MRI corrupted with noise and intensity inhomogeneity as well as we will provide a mathematical background of some recent ones which are compared in this work. Next, we will introduce a new approach based on kernel fuzzy c-means clustering and penalty-type entropy (MKPE) with appropriate initialization algorithm for segmentation of brain tissue from MRI corrupted by different noise and intensity inhomogeneity levels and we will examine an important theoretical question regarding the mathematical properties behind the proposed fuzzy MKPE clustering algorithm such as convergence. After our contribution, we will detail the dataset and the image modalities used. Afterwards, we will present the metrics employed to assess the performance of the proposed models. Finally, experiments setup, results, discussions and comparisons of the proposed method with other published works will be carried out.

3.2. Magnetic Resonance Image Processing

MRI is one of the most significant advances in medical imaging after the discovery of X-ray imaging. MRI is a non-invasive visualization technique of tissues and organs in the human body, allowing detection malformations, pathologies and tumors in patients without resorting to surgical exploration. Brain MRI is generally classified into two categories:

- Anatomical MRI (aMRI): Also called structural MRI, provides information about the structure, shape and content of brain tissue.
- Functional MRI (fMRI): Functional imaging is hardly interested in brain anatomy, but in the study of brain activity. fMRI is most often related to the research field of cognitive science and neuroscience.

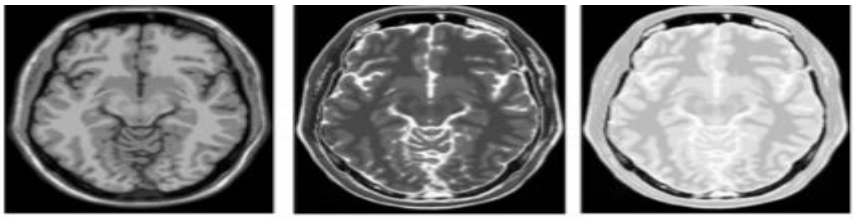
MRI technique generates 2D or 3D anatomical and functional images of the body, and particularly useful for neurological, ontological, cardiovascular, muscular and skeletal imaging. Images can be obtained in any orientation, rapidly, non-invasively, and without exposure to ionizing radiation. MRI technology is based on the application of a powerful steady magnetic field, lining up the hydrogen atoms in the tissue being imaged, and additional radio frequency fields, used to alter the alignment of the magnetization, and to produce an effect detectable by the scanner. It allows for the variation of parameters such as repetition time (TR - the time between two consecutive excitation pulses) and echo time (TE - the time between the excitation pulse and the recording of the magnetization value). By changing the values of the different sequence parameters (TE and TR), a wide variety of contrast varying volumes can be generated.

Proton density (PD) weighted image: for a long TR and a short TE, we get a density weighted sequence of PD protons. The fabrics are ordered by increasing gray levels in WM, GM and CSF.

T2-weighted image: for a long TR and TE, we obtain a T2-weighted sequence. The fabrics are ordered by increasing gray levels in WM, GM and CSF.

T1-weighted image: for a short TR and TE, we obtain a T1-weighted sequence. The fabrics are ordered by increasing gray levels in CSF, GM and WM. Each MRI sequence contains specific information, so the choice between the different sequences depends on the structure to be studied. Fig. 3.1 depicts the MRI sequences.

Many pulse sequences are available in MR imaging techniques which leads to an optimization problem. Depending on the anatomy of the structures of interest, the optimal pulse sequence has to be chosen in order to be able to optimally distinguish the tissues of interest and to undertake the segmentation procedure. For instance, MR imaging of brain tissues requires a specific setup in comparison to bone analysis. Despite the non-ionizing radiation characteristics, MR imaging experiences different imaging artifacts which generate difficulties for the segmentation techniques having considerable effects on their performances. Some of the drawbacks of this imaging technique are: noise, low contrast between certain tissues, partial volume effects, and intensity inhomogeneity. Noise and intensity inhomogeneity shortcomings shall be discussed further on.



(a) (b) (c)

FIGURE 3. 1: MRI sequences: (a) T1-weighted image, (b) T2-weighted image, (c) PD-weighted image

3.3. MRI Artifacts

MRI acquisition artifacts (errors) are numerous, varied in nature and may be related to different factors. These artifacts can mimic pathologies, leading to incorrect diagnoses. There are three types of artifacts. Physiological artifacts that are caused by the movement of the patient including breathing, heart rate, and blood flow. Artifacts that may result from the inherent physics of MRI such as the presence of metal or chemical change. Finally, those related to the hardware and software involved in the construction of MRI images. We describe in the rest of this section the random noise artifacts and inhomogeneity intensities.

3.3.1. Noise in MR Imaging

Noise is the random variations in voxel intensity levels where there should be none. Noise changes pixel configurations, weakens and can even obliterate what is supposed to be sharp edges, and can cause low spatial resolution as well as low contrast within and between anatomical structures. These negative features reduce the medical utility of an image because of a difficulty in the study of brain anatomy and the detection of disease signatures. A perfectly acquired image as viewed by the human eye contains some amount of noise. Noise becomes a nuisance when its presence attracts any level of poor visual quality. Noise is unavoidably present in images because of the imperfections of device components; and trade-offs in the operating parameters of imaging systems:

signal-to-noise ratio (SNR), resolution and length of scan time. An MRI image with very high SNR can be acquired by increasing acquisition time. In practice, acquisition time is compromised for patient comfort, particularly for very young and elderly subjects.

The full k-space acquired in MRI is assumed to be corrupted with Gaussian white noise. After Fourier transform, real and imaginary images are still corrupted by uncorrelated Gaussian noise with the same variance in both complex components [50]. The non-linear transformation leading to magnitude images modifies the noise distribution, which shows a Rician probability density function (PDF):

$$p_M(M \mid A,\sigma) = \frac{M}{\sigma^2} e^{-\frac{M^2+A^2}{2\sigma^2}} I_0\left(\frac{AM}{\sigma^2}\right) \tag{3.1}$$

where σ^2 is the noise variance in real and imaginary images; A is the noise-free image and is given by: $A^2 = A_R^2 + A_I^2$, where A_R and A_I are respectively the real and imaginary data; M is the actual magnitude image; I_0 is the 0th order modified Bessel function of first kind. As shown in Fig. 3.2, in case of high SNR, the Rician distribution tends to the Gaussian one, while, in the opposite limit $(A=0)$ it becomes the Rayleigh distribution with PDF:

$$p_M(M \mid \sigma) = \frac{M}{\sigma^2} e^{-\frac{M^2}{\sigma^2}} \tag{3.2}$$

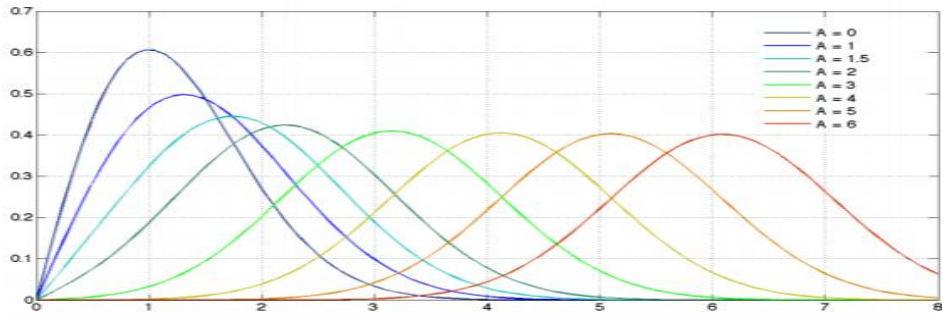


FIGURE 3. 2: Rician PDF plot at different A values with fixed $\sigma = 1.$ In case of high SNR ($A \geq 3$) the Rician distribution starts to approximate to Gaussian one, whereas in image points where (only noise is present) it corresponds to Rayleigh distribution.

3.3.2. Intensity Inhomogeneity

With the frequent application of the magnetic resonance (MR) imaging method to clinical diagnosis, automatic analysis of the acquired images using techniques from computer vision and pattern recognition has received considerable attention. In developing such computer-aided diagnosis tools, a commonly encountered problem is to correct the intensity inhomogeneity (IIH) in MR images. The IIH [88] (also termed as the bias field, or the gain field in the literature) usually refers to the slow, no anatomic intensity variations of the same tissue over the image domain.

Given an observed image I defined on a continuous domain Ω with intensity inhomogeneity, its intensities can be viewed as a product of the true image J and the bias field b, i.e,

$$I(x) = b(x)J(x) + n(x) \tag{3.3}$$

where $x \in \Omega$ and the noise $n(x)$ is assumed to be Gaussian-distributed with zero-mean and variance σ^2. Thus, the distribution of the image intensity I can be approximated by a Gaussian distribution with mean bJ and variance σ^2. The true image J characterizes an intrinsic property of objects being imaged, which ideally takes a

specific intensity c_i for the ith type of objects and is therefore assumed to be piecewise constant. That is to say, the true image J approximately takes N distinct constant values $c_1, c_2, \ldots, c_N$ in disjoint regions $\Omega_1, \Omega_2, \ldots, \Omega_N$, respectively. The bias field b is assumed to be slowly and smoothly varying as well known in the literature. The problem of image segmentation and bias correction is therefore considered as finding the specific intensities $c_1, c_2, \ldots, c_N$ and estimating the bias field b.

To simplify the computation, one often ignores the noise and takes the logarithmic transform of intensity

$$\log\left(I(x)\right) = \log\left(b(x)\right) + \log\left(J(x)\right) \tag{3.4}$$

In general, IIH is the characterization of a specific tissue class or homogeneous anatomical structure by non-uniform voxel intensities instead of uniform voxel intensities. The utility of an MRI image for manual and automated diagnosis can be significantly reduced by the presence of bias field. Visual inspection of an MRI scan degraded by the bias field is more challenging to a physician. It requires more time to discern between the presence of an abnormal anatomical structure and artifact. State-of-the-art image analysis algorithms, which are based on voxel intensities, perform poorly by misclassifying tissue classes because different intensity levels are assigned to the same tissue class. Several factors are responsible for the IIH. They include the non-uniform sensitivity of radiofrequency coils, non-uniformity of static field, improper acquisition parameter settings, gradient induced eddy currents and magnetization properties of an anatomic structure under investigation.

3.4. Segmentation of Brain MRI Corrupted by Noise and Intensity Inhomogeneity.

Image segmentation [58] is an essential image processing technique that analyzes an image by partitioning it into non-overlapped regions each region referring to a set of pixels. The pixels in a region are similar with respect to some characteristic such as color, intensity, or texture [64]. In other words, the goal of segmentation is to simplify and change the representation of an image into something that is more meaningful and easier to analyze. Image segmentation is typically used to locate objects and boundaries in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label will share certain characteristics. Image segmentation methods continue to be widely researched and can be applied in numerous applications including but not limited to medical image analysis. Processing and visualization of medical images is a rapidly growing area of research and MRI has provided a means for imaging tissue at very high resolutions providing the desired information for use in fields like radiotherapy treatment planning, stereotactic neurosurgery and others. In the context of neuro-imaging, segmentation of white matter, gray matter, CSF, etc. is extremely important for quantitative analysis such as volume measurements. It has already been established that volumetric analysis of different parts of the brain is useful in assessing progress or remission of various diseases e.g., the Alzheimer's and Epilepsy. Current image segmentation techniques include region-based segmentation, edge detection segmentation, segmentation based on clustering, segmentation based on weakly-supervised learning, and several others. Clustering techniques [15], however, are popular and established in their use for image segmentation due to their speed and effectiveness. Although numerous clustering techniques are available, all have the same purpose, that is, to divide a set of objects into clusters such that the objects in the same cluster are relatively similar, while the objects in different clusters are relatively dissimilar. Clustering techniques can be grouped into different types such as hierarchical, graph theoretic, decomposing a density function, and minimizing an objective function. This chapter focuses on the use of those clustering techniques that minimize an objective function. Clustering methods are either hard or fuzzy. In hard clustering, the clusters are disjoint and non-overlapping. However, in many real situations for images issues such as limited spatial resolution, poor contrast, intensity inhomogeneity and noise variation make this hard segmentation a difficult task. Fuzzy clustering allows an object to belong to all clusters with a specified membership degree and is uniquely effective when boundaries between clusters of data are ambiguous. To avoid this obstacle, fuzzy set theory [116] was presented; with the idea of partial membership of containing described by a membership objective function. Many researchers have presented modifications to these techniques to address these issues. This work research is to investigate such modifications made to the well-known fuzzy clustering method.

In the applicative context of this work, our purpose is to post-process ternary classification maps of MR images defined as

$$F : \Omega \to [0,1]^3$$
$$x \mapsto (p_{CSF}, p_{GM}, p_{WM})$$

where Ω corresponding to the support of an MR image, and for each point x, p_{CSF}, p_{GM} and p_{WM} are the probabilities that x belong to the cerebrospinal fluid, the grey matter (cortex) and the white matter, respectively. In particular, we have $\sum_l p_l = 1$. Such maps F are obtained by a fuzzy segmentation process described in [135]. A specificity of this segmentation strategy and many others despite its good accuracy from quantitative and geometric points of view, lies in the fact that it is not guided by topological constraints. As a consequence, its output needs to be post-processed for any further application requiring topological guarantees (for instance mesh generation or differential cortical surface analysis). Our purpose is then to build, from F, a new crisp segmentation map

$$T : \Omega \to \{0,1\}^3$$
$$x \mapsto (c_{CSF}, c_{GM}, c_{WM})$$

Such that for each point x, c_{CSF}, c_{GM} and c_{WM} are equal to either 0 or 1 depending on the (unique) class of x, i.e., with $\sum_l c_l = 1$. In addition, T should satisfy unlike F some topological priors related to the structure of the different tissues.

3.4.1. Related Work

Segmentation of brain tissues is still a very challenging task due to the presence of noise and intensity inhomogeneity. The presence of such artifacts can severely affect visual evaluation as well as segmentation based on absolute pixel intensities. There are a lot of methods available for MR image segmentation. Among them, fuzzy segmentation methods are of considerable benefits, because they could retain much more information from the original image than hard segmentation methods. In particular, the fuzzy c-means (FCM) algorithm [11], assign pixels to clusters without labels. However, these algorithms could not succeed completely to segment images corrupted by heavy noise, outliers, and other imaging artifacts, such as the intensity inhomogeneity. To overcome these drawbacks, many modified FCM algorithms have been proposed in the literature for the purpose of image segmentation. Some of the modified FCMs for image segmentation are as follows: Many researchers have incorporated spatial information into the original FCM algorithm to segment the images. Hathaway et al., [54] and (Khalilia et al., [71] proposed respectively Relational fuzzy c-means (RFCM) and Improved Relational Fuzzy c-Means (IRFCM) clustering methods; the first does not correctly detect the brain tissues and the second requires a very high time to run and fails to segment the tissues. Tolias and Panas [131] proposed a fuzzy rule-based system to impose spatial continuity on FCM, and in another paper [132], they used a small positive constant to modify the membership of the center pixel in a 3×3 window. Ahmed et al., [1] presented the spatial FCM (FCM_S) method for segmentation MRI which is formulated by modifying FCM algorithm to compensate for intensity inhomogeneity. In this model each pixel was labeled with considering its neighborhood. However, FCM-S is not efficient in the presence of noise and requires a high execution time. Pham et al. [110] modified the objective function in the FCM algorithm to include a multiplier field containing the first and second order information of the image. A rather recent approach proposed by Pham [109] is to penalize the FCM objective function to constrain the behavior of the membership functions, similar to methods used in the regularization and Markov random field (MRF) theory. Other improvements of fuzzy clustering methods for image segmentation are presented in literature [44]. On the other hand, there is a trend in recent machine learning work to construct a nonlinear version of a linear algorithm using the kernel method [19] for medical image analysis. The following section gives a mathematical background of the fuzzy clustering-based kernel methods compared in this study that are worked well on image segmentation.

3.4.2. Fuzzy Clustreing Methods for Brain MRI Segmentation

In this section, we provide a brief overview about six fuzzy clustering methods that are constructed by modifying fuzzy c-means based kernel methods.

Let $X = \{x_1, x_2, ..., x_N\}$ be a non-empty set of N data points $x_j \in \mathbb{R}^p$ and let c be the number of clusters, $2 \le c < N$. The problem of finding a (fuzzy) partition of X is denoted as a proper clustering problem if the data set contains at least $c < N$ distinct points x_j. The nondegerate fuzzy c-partitions of X are $(c \times N)$ matrices U from the set $M_{fc} = \left\{ U \in \mathbb{R}^{c \times N} \mid u_{ij} \in [0,1], \forall i, j; \sum_{i=1}^{c} u_{ij} = 1, \forall j; 0 < \sum_{j=1}^{N} u_{ij} < N, \forall i \right\}$

Let $V = (v_1, v_2, ..., v_c), v_i \in \mathbb{R}^p$ be the prototype of cluster center and the array $\{u_{ij}\} (= U)$ be a partition matrix satisfying $U \in M_{fc}$. The FCM algorithm performs clustering by solving

$$\text{Minimize } J_m(U,V;X) = \sum_{i=1}^{c} \sum_{j=1}^{N} u_{ij}^m \| x_j - v_i \|^2 \tag{3.5}$$

Subject to $V \in \mathbb{R}^{c \times p}$, $U \in M_{fc}$

where $\|.\|$ is any inner product induced norm and the parameter $m > 1$ is a weighting exponent on each fuzzy membership and determines the amount of fuzziness of the resulting classification. Optimal partitions U^* of X are taken from pairs (U^*, V^*) that are local minimizers of J_m. In image clustering, the most commonly used feature is the gray-level value. Thus, the FCM objective function is minimized when high membership values are assigned to pixels whose intensities are close to the centroid of its particular class, and low membership value are assigned when the point is far from the centroid. The membership function represents the probability that a pixel belongs to a specific cluster. In the FCM algorithm, the probability is dependent solely on the distance between the pixel and each individual cluster center in the feature domain.

We know every algorithm that only uses inner products can implicitly be executed in the feature space. This trick can also be used in clustering as shown in kernel fuzzy c-means algorithms [19]. A common philosophy behind these algorithms is based on the following kernel (substitution) trick, that is, firstly with a (implicit) nonlinear map, from the data space to the mapped feature space, $\phi : X \to F(x \to \phi(x))$, where $x \in \mathbb{R}^p$ is a sample in the original feature space and $\phi(x) \in \mathbb{R}^q$ denotes the mapped high feature space, $p << q$. A dataset $\{x_1, \ldots, x_N\} \subseteq X$ (an input data space with low dimension) is mapped into a potentially much higher dimensional feature space or inner product F, which aims at turning the original nonlinear problem in the input space into potentially a linear one in rather high dimensional feature space so as to facilitate problem solving. An interesting point about kernel function is that the inner product can be implicitly computed in F, without explicitly using or even knowing the mapping ϕ. So, kernels allow computing inner products in spaces, where one could otherwise not practically perform any computations. By referring to Eq. (3.5), the objective function of the kernel based FCM is defined as below:

$$J_m(U,V;X) = \sum_{i=1}^{c} \sum_{j=1}^{N} u_{ij}^m \left\| \phi(x_j) - \phi(v_i) \right\|^2 \tag{3.6}$$

Subject to $V \in \mathbb{R}^{c \times p}$, $U \in M_{fc}$

where ϕ is an implicit nonlinear map. $\phi(v_i)$ here is not expressed as a linearly-combined sum of all $\phi(x_j)$ anymore, a so-called dual representation, but still reviewed as a mapped point (image) of v_i in the original space, then with the kernel substitution trick, we have

$$\begin{aligned}\| \phi(x_j) - \phi(v_i) \|^2 &= \left(\phi(x_j) - \phi(v_i)\right)^T \left(\phi(x_j) - \phi(v_i)\right) \\ &= \phi(x_j)^T \phi(x_j) - \phi(v_i)^T \phi(x_j) - \phi(x_j)^T \phi(v_i) + \phi(v_i)^T \phi(v_i)\end{aligned}$$

And from the property of kernel function that $\phi(x_j)^T \phi(v_i) = K(x_j, v_i)$, the distance function (3.6) can be rewritten as

$$\| \phi(x_j) - \phi(v_i) \|^2 = K(x_j, x_j) - 2K(x_j, v_i) + K(v_i, v_i) \tag{3.7}$$

where $i = 1, 2, ..., c$ and $j = 1, 2, ...N$

Using Eq. (3.7) the objective function in Eq. (3.6) can be modified as

$$J_m(U, V; X) = \sum_{i=1}^{c} \sum_{j=1}^{N} u_{ij}^m \left(K(x_j, x_j) - 2K(x_j, v_i) + K(v_i, v_i) \right) \tag{3.8}$$

In the following subsections, we present the methods constructed by modifying the objective function of kernelized FCM.

3.4.2.1. Spatially Constrained KFCM for Image Segmentation

Dao-Qiang Zhang and Song Chen [24] proposed KFCM algorithm with spatial constraints (KFCM_S) for medical image segmentation by adding a penalty term containing spatial neighborhood information to the basic of KFCM. This penalty term acts as a regularizer and biases the solution toward piecewise-homogeneous labeling. Such regularization is helpful in segmenting images corrupted by noise. The objective function of KFCM_S is given by

$$J_m(U, V; X) = \sum_{i=1}^{c} \sum_{j=1}^{N} u_{ij}^m \left(1 - K(x_j, v_i)\right) + \frac{\alpha}{N_R} \sum_{i=1}^{c} \sum_{j=1}^{N} u_{ij}^m \sum_{r \in N_j} \left(1 - u_{ir}\right)^m \tag{3.9}$$

Subject to $V \in \mathbb{R}^{c \times p}$, $U \in M_{fc}$

where N_j stands for the set of neighbors that exist in a window around x_j (do not include x_j itself) and N_R is the cardinality of N_j. The parameter α controls the effect of the penalty term. The relative importance of the regularizing term is inversely proportional to the signal-to-noise ratio (SNR) of the MRI data. Low SNR would require a higher value of α, and vice versa. The new penalty term is minimized when membership value for a particular class of a pixel is large and the membership values for that cluster should be also large at neighboring pixels, and vice versa. In other words, a pixel's membership value is correlated to those of the other pixels at its neighborhood. In Eq. (3.9) the Gaussian RBF kernel is used. Obviously, it's satisfy $K(x, x) = 1$.

An iterative algorithm for minimizing Eq. (3.9) can be derived by evaluating the centroids and membership functions that satisfy a zero gradient condition like the KFCM. A necessary condition on u_{ij} and v_i for Eq. (3.9) to be at a local minimum or an saddle point are

$$u_{ij} = \frac{\left(\left(1 - K(x_j, v_i)\right) + \left(\alpha \sum_{r \in N_j} \left(1 - u_{ir}\right)^m / N_R \right) \right)^{1/(m-1)}}{\sum_{l=1}^{c} \left(\left(1 - K(x_j, v_l)\right) + \left(\alpha \sum_{r \in N_j} \left(1 - u_{lr}\right)^m / N_R \right) \right)^{1/(m-1)}} ; 1 \le i \le c, 1 \le j \le N$$

$$v_i = \frac{\sum_{j=1}^{N} u_{ij}^m \left(1 - K(x_j, v_i)\right) x_j}{\sum_{j=1}^{N} u_{ij}^m \left(1 - K(x_j, v_i)\right)} ; 1 \le i \le c$$

3.4.2.2. Adaptive Kernel-based Fuzzy C-Means Clustering with Spatial Constraints

Guang Hu and Zhenbin Du [48] proposed an adaptive kernel-based fuzzy C-means clustering with spatial constraints (AKFCMS) model for image segmentation approach in order to improve the efficiency of image segmentation. This model uses spatial neighborhood constraint property of image pixels into KFCM to control the influence of the neighborhood pixels to the central pixel automatically. First, the authors joined the spatial information constraint term to the objective function of KFCM as follows

$$J_m(U,V;X) = \sum_{i=1}^{c}\sum_{j=1}^{N} u_{ij}^m (1 - K(x_j, v_i) + \frac{\alpha}{S_r} \sum_{i=1}^{c}\sum_{j=1}^{N} u_{ij}^m \sum_{r \in R_j} (1 - K(x_r, v_i) \quad (3.10)$$

s.t. $V \in \mathbb{R}^{c \times p}$, $U \in M_{fc}$

where α the parameter is a weighting coefficient for the spatial information constraints, S_r is a neighborhood constant, and R_j is the neighborhood of the jth pixel. The second term in Eq. (3.10) is the objective constraint with the spatial information that is to keep the clustering continuity in x_j neighborhood.

In this model the authors set $u = K(x,v)\bar{u}$ and set the kernel function $K(x,v)$ to satisfy the equation $K(x,v) = 1$. Thus, AKFCMS is defined as Eq. (3.11).

$$J_m(U,V) = \sum_{i=1}^{c}\sum_{j=1}^{N} \left(K(x_j, v_i).\bar{u}_{ij}\right)^m (1 - K(x_j, v_i) + \frac{\alpha}{S_r} \sum_{i=1}^{c}\sum_{j=1}^{N} \left(K(x_j, v_i).\bar{u}_{ij}\right)^m \sum_{r \in R_j} (1 - K(x_r, v_i) \quad (3.11)$$

The fuzzy segmentation is carried out through iterative steps that minimize the objective function Eq. (3.11). The membership grade $\bar{u}_{ij}$ and the prototypes of center of ith cluster are update as:

$$\bar{u}_{ij} = \frac{\frac{\alpha}{S_r} \sum_{r \in R_j} \bar{u}_{ir} - \sum_{i=1}^{c} \frac{1}{\left(1 - K(x_j, v_i)\right) + \alpha}}{\left(1 - K(x_j, v_i)\right) + \alpha} ; 1 \le i \le c, 1 \le j \le N$$

$$v_i = \frac{\sum_{j=1}^{N} \bar{u}_{ij}^m K(x_j, v_i) x_j}{\sum_{j=1}^{N} \bar{u}_{ij}^m K(x_j, v_i)} ; 1 \le i \le c$$

Because $\bar{u}$ is the membership function of a denoising image, to minimize the Eq. (3.11) is equivalent to do fuzzy clustering for a denoising image, which can reduce the influence of the noise to the clustering result and improve segmentation precision and robustness.

3.4.2.3. Kernel Generalized Fuzzy C-Means Clustering with Spatial Information

In [155] a kernel generalized fuzzy c-means with spatial information (KGFCM-S) for image segmentation was proposed.

The objective function of KGFCM-S method as shown below

$$J_m(U,V;X)=2\sum_{i=1}^{c}\sum_{j=1}^{N}u_{ij}^{m}(1-K(x_j,v_i))+\sum_{j=1}^{N}a_j\sum_{i=1}^{c}u_{ij}(1-u_{ij}^{m-1})+2\beta\sum_{i=1}^{c}\sum_{j=1}^{N}u_{ij}^{m}(1-K(\bar{x}_j,v_i)) \quad (3.12)$$

s.t. $V\in\mathbb{R}^{c\times p}$, $U\in M_{fc}$

Where $a_j=\alpha.\min\left\{\|x_j-v_i\|\right\}\,|\,i\in\{1,...,c\},0\le\alpha<1$

The first term in the objective function Eq. (3.12) minimizes the compactness intra-class by using a kernel distance function. The second reward term is introduced for membership in a single data point to force a crisper assignment. The third term is the spatial constraint term, in which the parameter β controls the penalty effect of the spatial constraint.

The fuzzy segmentation is carried out through iterative steps that minimize the objective function Eq. (3.12). The membership grade u_{ij} and the prototypes of center of ith cluster are update as:

$$u_{ij}=\left(\sum_{l=1}^{c}\left(\frac{2(1-K(x_j,v_i))-a_j+2\beta(1-K(\bar{x}_j,v_i))}{2(1-K(x_j,v_l))-a_j+2\beta(1-K(\bar{x}_j,v_l))}\right)^{\frac{2}{m-1}}\right)^{-1} \quad ;1\le i\le c,1\le j\le N$$

$$v_i=\frac{\sum_{j=1}^{N}u_{ij}^{m}(K(x_j,v_i)x_j+\beta K(\bar{x}_j,v_i)\bar{x}_j)}{\sum_{j=1}^{N}u_{ij}^{m}(K(x_j,v_i)+\beta K(\bar{x}_j,v_i))} \quad ;1\le i\le c$$

3.4.2.4. Improved Kernel Fuzzy Local Information C-Means Algorithm and Outlier Rejection

A. Mekhmoukh and K. Mokrani [95] proposed a fuzzy clustering method based on Particle Swarm Optimization (PSO), Wavelets and outlier rejection caused by the membership function of the Kernel Fuzzy Local Information C-Means (KFLICM) algorithm combined with level set. Mathematically, the KFLICM is formulated as the minimization of the following cost function.

$$J_m(U,V;X)=\sum_{j=1}^{N}\sum_{i=1}^{c}u_{ij}^{m}\alpha^{(1-K(x_j-v_i))}+G_{ij} \quad (3.13)$$

Subject to $V\in\mathbb{R}^{c\times p}$, $U\in M_{fc}$

The factor G is defined as:

$$G_{ij}=\sum_{\substack{k\in N_j\\ j\neq k}}\frac{1}{d_{jk}+1}(1-u_{ik})^{m}\,\alpha^{(1-K(x_j,v_k))}$$

Where $K(x_i,v_j)=\exp\left(-\frac{\left\|x_j-v_i\right\|^2}{\sigma^2}\right)$

The role of the exponent variable α is to limit the partial distribution of the points among the two neighboring clusters rather than to all of the clusters. The exponent variable α is given as: $\alpha=\frac{(X_{\max}-X_{\min})+1}{256}+1$

Where $X_{\max}$ and $X_{\min}$ are the maximum and the minimum intensity in an image.

The membership grade u_{ij} and prototypes of center of ith cluster are update as:

$$u_{ij} = \frac{1}{\sum_{l=1}^{c}\left(\frac{\alpha^{(1-K(x_j,v_i))}+G_{ij}}{\alpha^{(1-K(x_j,v_l))}+G_{lj}}\right)^{\frac{2}{m-1}}};1\le i\le c,1\le j\le N$$

$$v_i = \frac{\sum_{j=1}^{N} u_{ij}^m K(x_j,v_i)x_j}{\sum_{j=1}^{N} u_{ij}^m K(x_j,v_i)};1\le i\le c$$

3.4.2.5. Improved Kernelized Rough-Fuzzy C-Means with Spatio-Contextual Information

A. Halder and N.A. Talukdar [51] proposed a kernelized rough fuzzy C-means clustering with spatial constraints (KRFCMSC) for brain tissue segmentation. This method is based on the judicious integration of the kernel trick in rough-fuzzy clustering with spatial constraint. Mathematically, the KRFCMSC method is formulated as the minimization of the objective function below.

$$J_m(U,V;X) = \sum_{i=1}^{c}\sum_{j=1}^{N} u_{ij}^m(1-K(x_j,v_i)) + \alpha\sum_{i=1}^{c}\sum_{j=1}^{N} u_{ij}^m(1-K(\underset{\sim}{x}_j,v_i)) \quad (3.14)$$

Subject to $V\in\mathbb{R}^{c\times p}$, $U\in M_{fc}$

Where $\underset{\sim}{x}_j$ represents the mean of the neighboring pixels around the pixel x_j and α is a parameter, that controls the effect of the neighboring term.

The criterion J_m is minimized by using the following formulas for updating the membership degree and the prototype v_i iteratively

$$u_{ij} = \left(\sum_{l=1}^{c}\left(\frac{(1-K(x_j,v_i))+\alpha(1-K(\underset{\sim}{x}_j,v_i))}{(1-K(x_j,v_l))+\alpha(1-K(\underset{\sim}{x}_j,v_l))}\right)^{\frac{1}{m-1}}\right)^{-1};1\le i\le c,1\le j\le N$$

$$v_i = \frac{\sum_{j=1}^{N} u_{ij}^m\left(K(x_j,v_i)x_j+\alpha K(\underset{\sim}{x}_j,v_i)\underset{\sim}{x}_j\right)}{\sum_{j=1}^{N} u_{ij}^m\left(K(x_j,v_i)+\alpha K(\underset{\sim}{x}_j,v_i)\right)};1\le i\le c$$

3.4.2.6. Radial Basis Kernel based Intuitionistic Fuzzy C-Means Clustering

Soni et al., [125] proposed a model called radial basis kernel based intuitionistic fuzzy c-means (KIFCM) which is an improvement of the intuitionistic fuzzy c-means [18]; it adopts a kernel induced metric in the space to replace the original Euclidian metric.

This method was applied on real and simulated MRI images. It's robust to noise and gave better segmentation results compared to other existing methods.

The KIFCM contains two terms: modified objective function of kernel fuzzy c-means using intuitionistic fuzzy set and intuitionistic fuzzy entropy. KIFCM is an iterative process that minimizes the following objective function.

$$J_m(U,V;X) = \sum_{j=1}^{N} u_{ij}^{*m}\left(1-K(x_j,v_i)\right) + \sum_{i=1}^{c}\pi_i^* e^{1-\pi_i^*} \quad (3.15)$$

Subject to $V \in \mathbb{R}^{c\times p}$, $U \in M_{fc}$

where $\pi_i^* = \frac{1}{N}\sum_{j=1}^{N}\pi_{ij},\ j \in [1, N]$ and $K(x_j, v_i) = \exp\left(-\frac{\sum |\ x_j^a - v_i^a\ |^b}{\sigma^2}\right)$

Whereas the intuitionistic fuzzy membership and hesitation degree are expressed as

$u_{ij}^* = u_{ij} + \pi_{ij}$ and $\pi_{ij} = 1 - u_{ij} - (1 - u_{ij}^{\alpha})^{1/\alpha}, \alpha > 0$

The first term in the objective function Eq. (3.15) minimizes the compactness intra-class by using a kernel distance function. The second term in the objective function represents the measures of fuzziness in fuzzy set. It gives the amount ambiguity or vagueness in a set and it's introduced to maximize the good point in the class.

The intuitionistic fuzzy membership and cluster centers are updated by the following equations

$$u_{ij}^* = \frac{\left(\frac{1}{1-K(x_j, v_i)}\right)^{\frac{1}{m-1}}}{\sum_{l=1}^{c}\left(\frac{1}{1-K(x_j, v_l)}\right)^{\frac{1}{m-1}}} \quad ;1 \le i \le c, 1 \le j \le N$$

$$v_i = \frac{\sum_{j=1}^{N} u_{ij}^{*m} K(x_j, v_i) x_j}{\sum_{j=1}^{N} u_{ij}^{*m} K(x_j, v_i)} \quad ;1 \le i \le c$$

3.5. The Proposed Method for Automatic Segmentation of Brain MRI Corrupted by Noise and Intensity Inhomogeneity.

Magnetic resonance images [136] may be affected by several sources of degeneration during image acquisition and transmission, where noise and intensity inhomogeneity are two kinds of commonly seen artifacts in MRI data. These artifacts pose a challenge in MR image segmentation. The noise in brain MRI can be Gaussian, Rician and Salt and Pepper and can affect significantly the performances of classification methods. Gaussian noise is a statistical noise having a probability density function (PDF) which is equal to that of the normal distribution. Primary sources of gaussian noise in digital images take place at the time of acquisition. Rician noise is the signal dependent noise which is initiated due to the formation of magnitude MRI which follows a Rice distribution. Salt and Pepper noise is also known as impulsive noise which is basically introduced because of bit errors in transmission or by analog to digital converter errors resulting in white pixels in dark regions and black pixels in bright region. To handle such problems, the best solutions consist of either filtering the image or embedding spatial regularization inside the classifier itself. The intensity inhomogeneity in brain MRI appears as a slowly varying nonlinear field in MR images, known as the bias field that changes the tissue statistics. Because of bias field, intensity of pixels of a particular class at other locations appears to belong to different classes. The conventional segmentation methods fail to classify such tissue classes affected by intensity inhomogeneity. However, the bias field changes from frame to frame for a particular patient and also from patient to patient in the same MR system. To handle such problems, there is a need to model the nonlinear bias field to control the effect of nonlinearity so that tissue properties can be retained leading to accurate classification. It is known from literature that FCM algorithm has proved to be an efficient tool to model the uncertainty in the data set and achieve clustering and it produces accurate results for noise free images. However, the main disadvantage of FCM is that it uses Euclidean distance measure to determine the similarity (distance) between the data points and the cluster centers. Hence, this is unable to capture the non-linear structures of the feature space adequately. However, its performance deteriorates to a great extent when the retinal images are contaminated by noise and suffer from outliers and/or other imaging

artifacts such as in-homogeneous intensity distribution, and non-uniform illumination. Recently kernel based FCM (KFCM) reported in [19] looks promising for data clustering as well as in image segmentation. The basic idea behind kernel based technique is that it transforms the low dimensional input space into a high dimensional feature space through some form of non-linear mapping. As a result, the data points not linearly separable in the original input space are now linearly separable in the high dimensional feature space and hence produce better results. KFCM and its variants have been proposed in the literature to correct the bias field, noise and to segment the brain MR images. Though these methods are worked well on image segmentation, it has some inadequacies such as, because of spatial terms and additional terms, it takes much time for segmentation result, for running the algorithm initially prototype assigned as randomly and the updated centers lead the result of clustering might be incorrect with images which are affected greatly by heavy noise intensity inhomogeneity. In order to address the above issues, we introduce a new strong objective function of kernel based fuzzy c-means to segment brain MRI tissues with both noise and intensity inhomogeneity. In the framework, intensities of a given image are viewed as a product of the true signal generated by the underlying anatomy and the bias field. Noise is considered as an additive component in the image model, which is first reduced by applying a local filtering. A Modified Kernel with Entropy-type Penalty methods with an appropriate initialization algorithm is then proposed to segment the brain MRI tissue with intensity inhomogeneity, and thereby obtain bias and noise corrected segmentation. This section is presented in two parts. The first one shows the formulation and the convergence proof of the proposed model and the second one presents the algorithm of our methodology.

3.5.1. **Formulation**

This section proposes an effective modified fuzzy c-means (MKPE) by incorporating normed kernel function and effective initialization method.

As described above the objective function of KFCM is constructed by replacing the Euclidian distance in FCM by the distance function $\| \phi(x_j) - \phi(v_i) \|^2$ where ϕ is the map on feature space and $\| \phi(x_j) - \phi(v_i) \|^2$ is the square Euclidean distance between x_j and v_i measured in the kernel space.

The basic idea of kernel trick is to solve the non-linear separability issues of the brain tissue regions. It involves applying a non-linear transformation ϕ to project the pixels into a larger dimension space which increases the chance that the classes become linear separability in the projected dimensional.

Examples of commonly used kernel functions $K : X \times X \to \mathbb{R}$ are as follows:

- Polynomial of degree d : $K(x_i, x_k) = (\gamma x_i^T x_k + \theta)^d, \gamma > 0, \theta \geq 0, d \in \mathbb{R}$
- Gaussian radial basis function (GRBF) kernel: $K(x_i, x_k) = e^{-\frac{\|x_i - x_k\|^2}{2\sigma^2}}, \sigma > 0$
- Laplacian: $K(x_i, x_k) = e^{-\gamma \|x_i - x_k\|}, \gamma > 0$
- Sigmoid: $K(x_i, x_k) = \tanh(\gamma x_i^T x_k + \theta), \gamma > 0, \theta \geq 0$

where γ , σ , θ and d are the adjustable parameters of the above kernel functions.

In this study, we used GRBF kernel. When we adopt it, $K(x, x)$ becomes 1. Then Eq. 3.8 becomes:

$$J_m(U, V; X) = 2 \sum_{i=1}^{c} \sum_{j=1}^{N} u_{ij}^m \left(1 - K(x_j, v_i)\right) \tag{3.16}$$

Based on the concept of machine learning with a learning capability to improve the performance of clustering results, an exponential-type distance is bounded and monotone increasing. Hence it can be robust to noise and

outliers when $1-K(x_j,v_i)=1-exp(-\| x_j-v_i \|^2/(2\sigma^2))$ is used to replace the Euclidean distance $\| x_j-v_i \|^2$.

Since σ is presented as a dispersion, we use the sample variance to estimate σ^2 with

$$\sigma^2=\frac{\sum_{j=1}^{N}\| x_j-\bar{x} \|^2}{N} \quad \text{with} \quad \bar{x}=\sum_{j=1}^{N} x_j \Big/ N \tag{3.17}$$

In the followings, we will introduce some modifications to the objective function in Eq. (3.16).

MR brain images are stacks of approximately 200 slices, which at their turn represent large matrices of voxels. A set of MR brain image slices contains around ten million voxels. The intensity of the voxels is generally encoded with 8 lit resolutions, that is, there are only 256 possible levels of intensity for each voxel. To considerably reduce the amount of calculations performed during the segmentation process, we first apply a local filtering to each voxel. Let us consider the neighborhood of the jth voxel. Let us denote by ξ_j the filtered intensity level of the jth voxel, and we will compute it as follows:

$$\xi_j=\frac{1}{1+\alpha}\left(x_j+\frac{\alpha}{N_R}\sum_{k\in N_j}x_k\right)$$

Where ξ_j is the linear weighted gray value of the jth voxel of the image ξ that is formed from the original image and its local spatial neighborhood, x_k is the gray value of neighborhood pixels centered at x_k and N_R denotes the pixel number of neighborhood pixels. In fact, $\left(\sum_{k\in N_j}x_k\right)/N_R$ represents the mean filtering of pixels in the neighborhood and the parameter α controls the tradeoff between the original image and its corresponding mean-filtered image. When is set to zero, the algorithm is equivalent to the original KFCM, while when it approaches infinite, the algorithm acquires the same effect as the original KFCM on the mean filtered image, its optimal value between 0.5 and 1.2 allows both to eliminate noise in the image and to preserve its content.

During the running of the algorithm, the clustering is performed on the gray level of the newly generated image ξ. As a consequence, the objective function in Eq. (3.16) can be modified as

$$J_m(U,V)=2\sum_{i=1}^{c}\sum_{j=1}^{N}u_{ij}^m\left(1-K(\xi_j,v_i)\right) \tag{3.18}$$

In MR image, bias field presents smooth variation of gray level of the pixels of the same tissue in spatial domain. Therefore, the observed MRI signal is modeled as a product of the true signal generated by the underlying anatomy, and a spatially varying factor called the gain field. Let ξ be the observed image and Y be the true image. The model is as follows:

$$\xi_k=y_k b_k+n_k \qquad \forall k\in\{1,2,...,N\} \tag{3.19}$$

where y_k and ξ_k are the true and observed intensities at the kth voxel, respectively, n_k is the noise and b_k is the gain field at the kth voxel, and is the total number of voxels in the MRI volume. By ignoring the noise term and applying the log-transform on both sides of Eq. (3.19) allows the artifact to be modeled as an additive bias field, denoting the log-transformed observed MR image data as $\tilde{\xi}_k$ and the log-transformed true intensity of underlying tissues as $\tilde{y}_k$. Thus, the MR image data can be approximated as

$$\tilde{\xi}_k=\tilde{y}_k+\beta_k \tag{3.20}$$

where $\tilde{\xi}_k = \log \xi_k$, $\tilde{y}_k = \log y_k$ and $\beta_k = \log b_k$. whereas $B = \{\beta_k\}_{k=1}^{N}$ is the log-transformed gain field. If the gain field is known, then it is relatively easy to estimate the tissue class by applying a conventional intensity-based segmented to the corrected data. Similarly, if the tissue classes are known, then we can estimate the gain field, but it may be problematic to estimate either without the knowledge of the other. We will show that by using an iterative algorithm based on fuzzy logic, we can estimate both. Before taking a log-transformation we add a constant value to each pixel to deal with zero values

Substituting (3.20) into (3.18), we have

$$J_m(U,V,B) = 2\sum_{i=1}^{c}\sum_{j=1}^{N} u_{ij}^m \left(1 - K(\tilde{\xi}_j - \beta_j, v_i)\right) \tag{3.21}$$

We next use the mixing proportion $\tau = (\tau_1, \tau_2, ..., \tau_c)$ of clusters, where τ_i presents the probability of one data point belonged to the ith cluster with the constraint $\sum_{i=1}^{c} \tau_i = 1$. Hence, $-\ln \tau_i$ is the information in the occurrence of a data point belonged to the ith cluster. Thus, we add the entropy term $\sum_{i=1}^{c}\sum_{j=1}^{N} u_{ij}^m \ln \tau_i$ to summarize the average of information for the occurrence of a data point belonged to the corresponding cluster over fuzzy memberships; with τ_i is the proportion of pixels which are from class i; $\tau_i = \dfrac{\sum_{j=1}^{N} u_{ij}^m}{\sum_{i=1}^{c}\sum_{j=1}^{N} u_{ij}^m}; i = 1, ..., c$

Thus, the objective function of MKPE is defined as

$$J_m(U,V,B) = 2\sum_{i=1}^{c}\sum_{j=1}^{N} u_{ij}^m \left(1 - K(\tilde{\xi}_j - \beta_j, v_i)\right) - w\sum_{i=1}^{c}\sum_{j=1}^{N} u_{ij}^m \ln \tau_i \tag{3.22}$$

subject to $B \in \Re^{N \times p}$, $V \in \Re^{c \times p}$, $U \in M_{fc}$

where $w \geq 0$ is a scale parameter for the penalty term $-\sum_{i=1}^{c}\sum_{j=1}^{N} u_{ij}^m \ln \tau_i$

The objective function J_m defined by Eq. (3.22) can be solved by the theorem below; in order to find the membership function and prototypes we adopt an iterative optimization algorithm that minimizes J_m.

Formally, the optimization problem comes in the form $\min_{U,V,B} J_m$, subject to $U \in M_{fc}$. The objective function can be minimized by taking the first derivatives of with respect to u_{ij}, v_i and β_j and setting them to zero results in three necessaries but not sufficient conditions for J_m to be at a local extrema. In the following subsections, we will derive these three conditions.

Theorem 3.1. Let $X = \{x_j,\ j = 1, \ldots, N / x_j \in \mathbb{R}^p\}$ an image to be segmented into c clusters. The necessary conditions for J_m to be a local extrema under the constraint of u_{ij} are

$$u_{ij}^{*}=\left(\sum_{k=1}^{c}\frac{\left(2\left(1-K(\tilde{\xi}_j-\beta_j,v_i)\right)-w\ln\tau_i\right)^{1/(m-1)}}{\left(2\left(1-K(\tilde{\xi}_j-\beta_j,v_k)\right)-w\ln\tau_k\right)^{1/(m-1)}}\right)^{-1};i=1,...,c;\ j=1,...,N \tag{3.23}$$

$$v_i^{*}=\frac{\sum_{j=1}^{N}u_{ij}^{m}\left(K(\tilde{\xi}_j-\beta_j,v_i)\right).(\tilde{\xi}_j-\beta_j)}{\sum_{j=1}^{N}u_{ij}^{m}\left(K((\tilde{\xi}_j-\beta_j,v_i)\right)};i=1,...,c \tag{3.24}$$

$$\beta_j^{*}=\tilde{\xi}_j-\frac{\sum_{i=1}^{c}u_{ij}^{m}\left(K(\tilde{\xi}_j-\beta_j,v_i)\right).v_i}{\sum_{i=1}^{c}u_{ij}^{m}\left(K((\tilde{\xi}_j-\beta_j,v_i)\right)};\ j=1,...,N \tag{3.25}$$

Proof. The problem of minimizing the objective function written in the Eq. (3.22) subject to constraint can be solved by the Lagrange multiplier method. The new objective function can be rewritten as follows

$$L_m(U,V,B,\lambda)=2\sum_{i=1}^{c}\sum_{j=1}^{N}u_{ij}^{m}\left(1-K(\tilde{\xi}_j-\beta_j,v_i)\right)-w\sum_{i=1}^{c}\sum_{j=1}^{N}u_{ij}^{m}\ln\tau_i+\sum_{j=1}^{N}\lambda_j\left(1-\sum_{i=1}^{c}u_{ij}\right)$$

With $\lambda_j, 1\le j\le N$ is the Lagrangian multiplier. Hence, we minimize the following objective function with respect to u_{ij}, v_i and β_j to obtain the optimal estimates u_{ij}^{*}, v_i^{*} and β_j^{*}.

$$(u_{ij}^{*},v_i^{*},\beta_j^{*})=\arg\min_{(u_{ij},v_i,\beta_j)}L_m$$

A: Membership Evaluation. Taking the partial derivative of L_m with respect to each λ_j and u_{ij}; and setting the results of each of them to zero: $\frac{\partial L_m}{\partial u_{ij}}=0$ and $\frac{\partial L_m}{\partial \lambda_j}=0$ we have:

$$\left[\frac{\partial L_m}{\partial u_{ij}}=2mu_{ij}^{m-1}\left(1-K(\tilde{\xi}_j-\beta_j,v_i)\right)-wmu_{ij}^{m-1}\ln\tau_i-\lambda_j\right]_{u_{ij}=u_{ij}^{*}}=0 \tag{3.26}$$

$$\frac{\partial L_m}{\partial \lambda_j}=0\ \Leftrightarrow\sum_{i=1}^{c}u_{ij}=1 \tag{3.27}$$

From Eq. (3.26) we obtain

$$u_{ij}^{*}=\left(\frac{\lambda_j}{m\left(2\left(1-K(\tilde{\xi}_j-\beta_j,v_i)\right)-mw\ln\tau_i\right)}\right)^{1/(m-1)} \tag{3.28}$$

By substituting Eq. (3.28), into Eq. (3.27), we get

$$\left(\frac{\lambda_j}{m}\right)^{1/(m-1)}=\frac{1}{\sum_{i=1}^{c}\left(2\left(1-K(\tilde{\xi}_j-\beta_j,v_i)\right)-w\ln\tau_i\right)^{1/(m-1)}} \tag{3.29}$$

Finally, by substituting Eq. (3.29), into Eq. (3.28), we have

$$u_{ij}^{*}=\left(\sum_{k=1}^{c}\frac{\left(2\left(1-K(\tilde{\xi}_j-\beta_j,v_i)\right)-w\ln\tau_i\right)^{1/(m-1)}}{\left(2\left(1-K(\tilde{\xi}_j-\beta_j,v_k)\right)-w\ln\tau_k\right)^{1/(m-1)}}\right)^{-1};i=1,...,c;\ j=1,...,N$$

B: Cluster Prototype Updating. Taking the partial derivative of L_m with respect to each v_i and setting the result of each to zero: $\frac{\partial L_m}{\partial v_i}=0$ we have:

$$\left[\frac{\partial L_m}{\partial v_i}=2\sum_{j=1}^{N}u_{ij}^{m}\left(-K(\tilde{\xi}_j-\beta_j,v_i)\right).\frac{(\tilde{\xi}_j-\beta_j-v_i)}{\sigma^2}\right]_{v_i=v_i^{*}}=0 \tag{3.30}$$

Eq. (3.30) leads to

$$v_i^{*}=\frac{\sum_{j=1}^{N}u_{ij}^{m}\left(K(\tilde{\xi}_j-\beta_j,v_i)\right).(\tilde{\xi}_j-\beta_j)}{\sum_{j=1}^{N}u_{ij}^{m}\left(K(\tilde{\xi}_j-\beta_j,v_i)\right)};i=1,...,c$$

C: Bias-Field Estimation. In a similar fashion, taking the derivative of L_m with respect to β_j and setting the result to zero: $\frac{\partial L_m}{\partial \beta_j}=0$ we have

$$\left[\sum_{i=1}^{c}\frac{\partial}{\partial\beta_j}\sum_{j=1}^{N}2.u_{ij}^{m}\left(1-K(\tilde{\xi}_j-\beta_j,v_i)\right)\right]_{\beta_j=\beta_j^{*}}=0$$

Since only the jth term in the second summation depends on β_j, we have

$$\left[\sum_{i=1}^{c}\frac{\partial}{\partial\beta_j}2.u_{ij}^{m}\left(1-K(\tilde{\xi}_j-\beta_j,v_i)\right)\right]_{\beta_j=\beta_j^{*}}=0$$

Differentiating the distance expression, we obtain

$$\left[2\sum_{i=1}^{c}u_{ij}^{m}\left(-K(\tilde{\xi}_j-\beta_j,v_i)\right).\frac{(\tilde{\xi}_j-\beta_j-v_i)}{\sigma^2}\right]_{\beta_j=\beta_j^{*}}=0$$

i.e., $$\left[\tilde{\xi}_j\sum_{i=1}^{c}u_{ij}^{m}K(\tilde{\xi}_j-\beta_j,v_i)-\beta_j\sum_{i=1}^{c}u_{ij}^{m}K(\tilde{\xi}_j-\beta_j,v_i)-\sum_{i=1}^{c}u_{ij}^{m}v_iK(\tilde{\xi}_j-\beta_j,v_i)\right]_{\beta_j=\beta_j^{*}}=0$$

Thus, the zero-gradient condition for the bias-field estimator is expressed as

$$\beta_j^{*}=\tilde{\xi}_j-\frac{\sum_{i=1}^{c}u_{ij}^{m}\left(K(\tilde{\xi}_j-\beta_j,v_i)\right).v_i}{\sum_{i=1}^{c}u_{ij}^{m}\left(K(\tilde{\xi}_j-\beta_j,v_i)\right)};\ j=1,...,N$$

3.5.2. Convergence Proof of the Proposed Model

In this section, we explore some propositions which ensure the convergence of MKPE.

Proposition 3.1. Let $\psi : \mathbb{R}^{c\times N} \to \mathbb{R}, \psi(U) = J_m(U,V,B)$, where V and B are fixed. Then $U^* \in M_{fc}$ is a strict local minimum of ψ if and only if U^* is calculated as in Eq. (3.23)

Proof.

We consider the relaxed minimization of $\psi(U)$ via Lagrange multipliers. Let $L(U,\lambda)$ be the Lagrangian and $\lambda = (\lambda_1, \lambda_2, ..., \lambda_N)$ be the multipliers

$$L_m(U,\lambda) = 2\sum_{i=1}^{c}\sum_{j=1}^{N} u_{ij}^m \left(1 - K(\tilde{\xi}_j - \beta_j, v_i)\right) - w\sum_{i=1}^{c}\sum_{j=1}^{N} u_{ij}^m \ln \tau_i + \sum_{j=1}^{N} \lambda_j \left(1 - \sum_{i=1}^{c} u_{ij}\right)$$

Then,

$$\frac{\partial L_m(U,\lambda)}{\partial u_{ij}} = 2mu_{ij}^{m-1}\left(1 - K(\tilde{\xi}_j - \beta_j, v_i)\right) - wmu_{il}^{m-1} \ln \tau_i - \lambda_j = 0 \quad \text{at } U^*$$

We calculate the second-order derivative of $L_m(U,\lambda)$

$$\frac{\partial}{\partial u_{sl}}\left(\frac{\partial L_m(U,\lambda)}{\partial u_{ij}}\right) = \begin{cases} 2(m-1)mu_{ij}^{m-2}\left(1 - K(\tilde{\xi}_j - \beta_j, v_i)\right) - w(m-1)mu_{ij}^{m-2} \ln \tau_i & if \quad s = i; l = j \\ 0 \quad otherwise \end{cases}$$

Now, we calculate the Hessian matrix $H(U^*)$ by substituting the updated formula of u_{ij} into the second-order derivation of $L_m(U,\lambda)$. The $H(U^*)$ is a nonzero entries matrix, whose diagonal elements are

$$\gamma_{ij,ij} = 2(m-1)m\left(u_{ij}^*\right)^{m-2}\left(1 - K(\tilde{\xi}_j - \beta_j, v_i)\right) - w(m-1)m\left(u_{ij}^*\right)^{m-2} \ln \tau_i$$

and $\gamma_{sl,ij} = 0$ for $s \neq i$ and $l \neq j$

We have m>1 and $0 < \tau_i < 1$. Thus $\gamma_{ij,ij} > 0$.

The Hessian of ψ at U^* has all positive eigenvalues which are $\gamma_{ij,ij}, 1 \le i \le c, 1 \le j \le N$ then U^* is a strict local minimum of ψ.

Next, we fix U, B and consider the minimization of J_m in variables $V = \{v_i\}$

Proposition 3.2. Let $\psi : \mathbb{R}^{c\times p} \to \mathbb{R}, \psi(V) = J_m(U,V,B)$, where U and B are fixed. Then, V^* is a strict local minimum of ψ if and only if $v_i^*; 1 \le i \le c$ is calculated via the updated formula in Eq. (3.24)

Although, the convergence of KFCM has been proven; and in our model, we have not modified the cluster Prototype. Thus, the Hessian matrix of $\psi(V)$ at V^* has all positive eigenvalues. Then V^* is minimum of $\psi(V)$

Next, we fix U, V and consider the minimization of J_m in variables $B = \{\beta_j\}$

Proposition 3.3. Let $\psi : \mathbb{R}^{N\times p} \to \mathbb{R}, \psi(B) = J_m(U,V,B)$, where U and V are fixed. Then B* is a strict local minimum of ψ if and only if $\beta_j; 1 \le j \le N$ is calculated via the updated formula in Eq. (3.25)

Proof.

Since there is no constrains for B, in order to minimize ψ, it is necessary to require $\nabla\psi(B^*)$ to vanish for every j.

$$\frac{\partial \psi(B)}{\partial \beta_j} = 2\sum_{i=1}^{c} u_{ij}^m \left(-K(\tilde{\xi}_j - \beta_j, v_i)\right).\frac{(\tilde{\xi}_j - \beta_j - v_i)}{\sigma^2} = 0 \quad \text{at} \quad B*$$

We calculate the second-order derivative of $\psi(B)$

$$\frac{\partial}{\partial \beta_s}\left(\frac{\partial \psi(B)}{\partial \beta_j}\right) = \begin{cases} \sum_{i=1}^{c}\left(\frac{2}{\sigma^2}u_{ij}^m\left(K(\tilde{\xi}_j - \beta_j, v_i)\right) - 2u_{ij}^m.\frac{(\tilde{\xi}_j - \beta_j - v_i)^2}{\sigma^4}K(\tilde{\xi}_j - \beta_j, v_i)\right) & if \quad s = j \\ 0 \quad otherwise \end{cases}$$

The Hessian matrix $H(U*)$ is calculated by substituting the updated formula of β_j into the second-order derivation of $\psi(B)$.

The Hessian matrix of $\psi(B)$ at $B*$ has all positive eigenvalues. Then $B*$ is minimum of $\psi(B)$

It is well known that the Zangwill's convergence theorem [152] can be adopted for the convergence proof for almost all iterative optimization algorithms. We aim to prove the convergence of MKPE by demonstrating it is a special case of Zangwill's theorem that can be summarized as follows.

Lemma 3 .1. Zangwill's convergence theorem

Let $A: Z \to P(Z)$ be a point-to-set map. Given a point $\mathbf{z}^{(0)} \in Z$, let A determine an algorithm that generates the sequence $\{\mathbf{z}^{(k)}\}$. Let $\Omega \subset Z$ be the solution set and assume that

(1) All points $\mathbf{z}^{(k)}$ are in a compact subset of Z.

(2) There is a continuous function $J: V \to \mathbb{R}$, satisfying

(a) If $\mathbf{z} \notin \Omega$, then for any $\mathbf{y} \in A(\mathbf{z})$, $J(\mathbf{y}) < J(\mathbf{z})$

(b) If $\mathbf{z}^{(k)} \in \Omega$, then either the algorithm terminates or for any $\mathbf{y} \in A(\mathbf{z})$, $J(\mathbf{y}) \leqslant J(\mathbf{z})$.

(3) The map A is closed at $\mathbf{z}$ if $\mathbf{z} \notin \Omega$.

Then, either the algorithm stops at a solution or the limit of any convergent subsequence is a solution. Here, our aim is to prove that the three conditions in Lemma 3.1 are satisfied by the MKPE algorithm when its parameters satisfy some conditions. Let us first give some definitions.

Definition 3.1.

A point to set map $A: X \to P(Y)$ is said to be closed at a point $\mathbf{x}$ in X if $\{\mathbf{x}^{(k)}\} \subset \mathbf{X}, \mathbf{x}^{(k)} \to \mathbf{x}^*$, $\mathbf{y}^{(k)} \in A(\mathbf{x}^{(k)})$, and $\mathbf{y}^{(k)} \to \mathbf{y}^*$ imply that $\mathbf{y}^* \in A(\mathbf{x}^*)$.

Definition 3.2a.

Let $T_1: \mathbb{R}^{N\times p} \times M_{fc} \to \mathbb{R}^{c\times p}$ be a function defined as:

$$T_1(B,U) = V = (v_1, v_2, \ldots, v_c)^T$$

where $\mathbf{v}_i = (v_{i1}, v_{i2}, \ldots, v_{ip})^T \in \mathbb{R}^p$, $1 \leqslant i \leqslant c$ is computed from Eq. (3.24).

Definition 3.2b.

Let $T_2: \mathbb{R}^{c\times p} \times M_{fc} \to \mathbb{R}^{N\times p}$ be a function defined as:

$$T_2(V,U) = B = (\beta_1, \beta_2, \ldots, \beta_N)^T$$

where $\beta_j = \left(\beta_{j1}, \beta_{j2}, \ldots, \beta_{jp}\right)^T$, $1 \leqslant j \leqslant N$, is computed from Eq. (3.25).

Definition 3.2c.

Let $T_3 : \mathbb{R}^{N\times p} \times \mathbb{R}^{c\times p} \to M_{fc}$ be a function defined as: $T_3(B,V) = U$, where $U = \left[u_{ij} \right]_{c\times N}$, is computed from Eq. (3.23).

Definition 3.3.

Let $T_m : \mathbb{R}^{N\times p} \times \mathbb{R}^{c\times p} \times M_{fc} \to \mathbb{R}^{N\times p} \times \mathbb{R}^{c\times p} \times M_{fc}$ be defined as:

$T_m(B,V,U) = \{(B*,V*,U*) \mid V* = T_1(B,U), B* = T_2(V*,U), U* = T_3(B*,V*)\}$

A sequence $\left\{ \left(B^{(t)}, V^{(t)}, U^{(t)} \right) \right\}$ is said to be a MKPE iteration sequence if $\left(B^{(t)}, V^{(t)}, U^{(t)} \right) \in T_m \left(B^{(t-1)}, V^{(t-1)}, U^{(t-1)} \right)$ for $t = 2,3,\ldots$

Theorem 3.2. Convergence of MKEP

Let $X = \{x_1, x_2, \ldots, x_N\}$ contain at least $c (< N)$ distinct points. Let

$\Omega = \{(B*,V*,U*) \in \mathbb{R}^{N\times p} \times \mathbb{R}^{c\times p} \times M_{fc} \mid J_m(B*,V*,U*) < J_m(B*,V*,U), \forall U* \neq U;$

$J_m(B*,V*,U*) < J_m(B,V*,U*), \forall B* \neq B;$

$J_m(B*,V*,U*) < J_m(B*,V,U*), \forall V* \neq V\}$

be the solution set of the optimization problem $min\, J_m\left(B,V,U\right)$ $s.t.\ U \in M_{fc}, B \in \mathbb{R}^{N\times p}, V \in \mathbb{R}^{c\times p}$.

Let $B^{(0)} \in \mathbb{R}^{N\times p}, V^{(0)} \in \mathbb{R}^{c\times p}, U^{(0)} = T_3(B^{(0)}, V^{(0)})$ and $(B^{(0)}, V^{(0)}, U^{(0)})$ be the starting point of iterations with T_m. Then, the iteration sequence $\{(B^{(t)}, V^{(t)}, U^{(t)})\}$, $t = 1,2,\ldots$, either terminates at a point (B^*, V^*, U^*) in Ω, or has a subsequence converging to a point in Ω.

Proof.

(1) we give detailed proof that $\left(U, V, B\right)$ belongs to a compact set.

Let $X = \left\{x_1, x_2, ..., x_N\right\}$ be a finite dataset in feature space $\mathbb{R}^d$; we define

$D(X) = \{x - y \mid x \in X, y \in X\}$

$Y = D(X) \cup X$

Let $\text{conv}(X) = \{\mathbf{x} \in \mathbb{R}^d \mid \mathbf{x} = \sum_{i=1}^n \alpha_i x_i; \sum_{i=1}^n \alpha_i = 1; \alpha_i \geq 0; x_i \in X \subset \mathbb{R}^d\}$ be the convex hull of X

Since $v_i^{(t)}$ is calculated by Eq. (3.24) so that

$$v_i^{(1)} = \frac{\sum_{j=1}^{N} \left(u_{ij}^{(1)}\right)^m \left(K(\tilde{\xi}_j - \beta_j^{(0)}, v_i^{(0)})\right).(\tilde{\xi}_j - \beta_j^{(0)})}{\sum_{j=1}^{N} \left(u_{ij}^{(1)}\right)^m \left(K(\tilde{\xi}_j - \beta_j^{(0)}, v_i^{(0)})\right)}; i = 1,...,c$$

By letting

$$\alpha_{ij} = \frac{\left(u_{ij}^{(1)}\right)^m \left(K(\tilde{\xi}_j - \beta_j^{(0)}, v_i^{(0)})\right)}{\sum_{j=1}^{N} \left(u_{ij}^{(1)}\right)^m \left(K(\tilde{\xi}_j - \beta_j^{(0)}, v_i^{(0)})\right)}; j = 1,...,N$$

we rewrite $v_i^{(1)}$ as $v_i^{(1)} = \sum_{j=1}^{N} \alpha_{ij} (\tilde{\xi}_j - \beta_j^{(0)})$

with

$$v_i^{(1)} = \sum_{j=1}^{N}\left(\frac{\left(u_{ij}^{(1)}\right)^m\left(K(\tilde{\xi}_j-\beta_j^{(0)},v_i^{(0)})\right)}{\sum_{j=1}^{N}\left(u_{ij}^{(1)}\right)^m\left(K(\tilde{\xi}_j-\beta_j^{(0)},v_i^{(0)})\right)}\right)=\left(\frac{\sum_{j=1}^{N}\left(u_{ij}^{(1)}\right)^m\left(K(\tilde{\xi}_j-\beta_j^{(0)},v_i^{(0)})\right)}{\sum_{j=1}^{N}\left(u_{ij}^{(1)}\right)^m\left(K(\tilde{\xi}_j-\beta_j^{(0)},v_i^{(0)})\right)}\right)=1$$

Furthermore, it follows from the constraints (2.10a) and (2.10c) $0\le\alpha_{ij}\le 1$ for all $1\le i\le c$ and $1\le j\le N$

Let $U^{(1)}\in M_{fc}$ be chosen, with B being initialized as $B^{(0)}=0$, then $v_i^{(1)}=\sum_{j=1}^{N}\alpha_{ij}\tilde{\xi}_j$

Thus, $\forall i,\ v_i^{(t)}\in conv(X)$ where $conv(X)$ denotes the convex hull of X

Proposition 3.4. If $\forall i,\ v_i^{(t)}\in conv(X)$, then so does $v_i^{(t+1)}$

Consider a point $\tilde{\xi}_j$ inside $conv(X)$, $\forall i,\ v_i^{(t)}$ is also inside $conv(X)$, then we use directed arrows to denote the subtraction of $\tilde{\xi}_j$ and $v_i^{(t)}$ $\left(\tilde{\xi}_j-v_i^{(t)}\right)$. Note that the endpoint of $\left(\tilde{\xi}_j-v_i^{(t)}\right)$, $\forall i$ points to $\tilde{\xi}_j$ for later convenient proof. Next, we will analyze $B^{(t)}$, which is calculated using Eq. (3.25) as follows:

$$\beta_j^{(t)}=\tilde{\xi}_j-\frac{\sum_{i=1}^{c}\left(u_{ij}^{(t)}\right)^m\left(K(\tilde{\xi}_j-\beta_j^{(t-1)},v_i^{(t)})\right).v_i^{(t)}}{\sum_{i=1}^{c}\left(u_{ij}^{(t)}\right)^m\left(K(\tilde{\xi}_j-\beta_j^{(t-1)},v_i^{(t)})\right)}=\frac{\sum_{i=1}^{c}\left(u_{ij}^{(t)}\right)^m\left(K(\tilde{\xi}_j-\beta_j^{(t-1)},v_i^{(t)})\right).\left(\tilde{\xi}_j-v_i^{(t)}\right)}{\sum_{i=1}^{c}\left(u_{ij}^{(t)}\right)^m\left(K(\tilde{\xi}_j-\beta_j^{(t-1)},v_i^{(t)})\right)}=\sum_{i=1}^{c}\alpha_{ij}\left(\tilde{\xi}_j-v_i^{(t)}\right)$$

for $j=1,...,N$

It can be easily inferred that $0\le\alpha_{ij}\le 1$, $\sum_{i=1}^{c}\alpha_{ij}=1$, in this case, with the endpoint of $\left(\tilde{\xi}_j-v_i^{(t)}\right)$ felling on $\tilde{\xi}_j$, the start points of $\beta_j^{(t)}$ would certainly satisfy

$\forall j,\tilde{\xi}_j-\beta_j^{(t)}\in con(VX)\subset con(X)$

Where $VX=\left(\tilde{\xi}_j,v_1^{(t)},...,v_c^{(t)}\right)$

Note that the objective function is independent for each dimension; thus, we can only consider the $d=1$ case. In this situation, $conv(X)$ is projected into a line segment and it is easy to deduce that $\tilde{\xi}_j-\beta_j^{(t)}\in con(X)$

Since $\forall i,\ v_i^{(1)}\in conv(X)$ has been proved, we could obtain $\forall i,t,\ v_i^{(t)}\in conv(X)$. Analogously, we could also obtain that $\forall i,t,\ \beta_i^{(t)}\in conv(D(X))$

Proposition 3.5. Let $\left[conv(Y)\right]^c$ be the c-fold Cartesian product of the convex hull of $Y=D(X)\cup X$, then $\left(U^{(t)},V^{(t)},\mathrm{B}^{(t)}\right)$ is in a compact set.

Proof. In each iteration, $U^{(t)}\in M_{fc}$, $V^{(t)}\in\left[conv(Y)\right]^c$ and $B^{(t)}\in\left[conv(Y)\right]^c$. Thus, every iteration of J_m satisfies $\left(U^{(t)},V^{(t)},\mathrm{B}^{(t)}\right)\in M_{fc}\times\left[conv(Y)\right]^c\times\left[conv(Y)\right]^c$. Y is finite and its element has finite components; thus, the diameter of Y is bounded. Since $conv(Y)$ is convex hull of finitely generators, it is closed. Thus, $conv(Y)$ is compact, $\left[conv(Y)\right]^c$ is also compact by the generalized Heine–Borel theorem. M_{fc} is bounded and closed, so it is compact. Thus, $\left(U^{(t)},V^{(t)},\mathrm{B}^{(t)}\right)\in M_{fc}\times\left[conv(Y)\right]^c\times\left[conv(Y)\right]^c$, which is compact.

(2) Next, we will show that the inequality $J_m(B^{(t+1)},V^{(t+1)},U^{(t+1)}) \le J_m(B^{(t)},V^{(t)},U^{(t)})$ is strict if $(B^{(t)},V^{(t)},U^{(t)}) \notin \Omega$. To do this, we only need to show that $J_m(B^{(t+1)},V^{(t+1)},U^{(t+1)}) \neq J_m(B^{(t)},V^{(t)},U^{(t)})$ if $(B^{(t)},V^{(t)},U^{(t)}) \notin \Omega$.

Using proof by contradiction, we assume $J_m(B^{(t+1)},V^{(t+1)},U^{(t+1)}) = J_m(B^{(t)},V^{(t)},U^{(t)})$ for some $(B^{(t)},V^{(t)},U^{(t)}) \notin \Omega$. . Then, we have

$$J_m(B^{(t)},V^{(t)},U^{(t)}) = J_m(B^{(t)},V^{(t+1)},U^{(t)}) = J_m(B^{(t+1)},V^{(t+1)},U^{(t)}) = J_m(B^{(t+1)},V^{(t+1)},U^{(t+1)}) \quad (3.31)$$

$V^{(t+1)} = T_1(B^{(t)},U^{(t)})$ is the minimizer of the function $\psi(V) = J_m(B^{(t)},V,U^{(t)})$. From Eq. (3.31), we also have $J_m(B^{(t)},V^{(t)},U^{(t)}) = J_m(B^{(t)},V^{(t+1)},U^{(t)})$. As the function J_m only has a unique global minimizer, we have $V^{(t)} = V^{(t+1)}$. In a similar way, we can have $B^{(t)} = B^{(t+1)}$ and $U^{(t)} = U^{(t+1)}$.

From $(B^{(t)},V^{(t)},U^{(t)}) \notin \Omega$ and the definition of Ω, at least one of the following three cases would happen.

① There exists $V^* \neq V^{(t)}$, satisfying

$$J_m(B^{(t)},V^{(t)},U^{(t)}) \ge J_m(B^{(t)},V^*,U^{(t)}) \quad (3.32a)$$

② There exists $B^* \neq B^{(t)}$, satisfying

$$J_m(B^{(t)},V^{(t)},U^{(t)}) \ge J_m(B^*,V^{(t)},U^{(t)}) \quad (3.32b)$$

③ There exists $U^* \neq U^{(t)}$, satisfying

$$J_m(B^{(t)},V^{(t)},U^{(t)}) \ge J_m(B^{(t)},V^{(t)},U^*) \quad (3.32c)$$

In case ①, since $V^* \neq V^{(t)}$ and $V^{(t)} = V^{(t+1)}$, we have $V^* \neq V^{(t+1)}$. $V^{(t+1)} = T_1(B^{(t)},U^{(t)})$. Then, we have

$$J_m(B^{(t)},V^{(t)},U^{(t)}) \ge J_m(B^{(t)},V^*,U^{(t)}) > J_m(B^{(t)},V^{(t+1)},U^{(t)}) = J_m(B^{(t+1)},V^{(t+1)},U^{(t+1)})$$

From Eq. (3.32a), we conclude that $J_m(B^{(t)},V^{(t)},U^{(t)}) > J_m(B^{(t+1)},V^{(t+1)},U^{(t+1)})$ which contradicts Eq. (3.31) Similarly, both case ② and case ③ can be proved by contradiction with our assumption. As a conclusion, $J_m(B^{(t+1)},V^{(t+1)},U^{(t+1)}) \leqslant J_m(B^{(t)},V^{(t)},U^{(t)})$ for every $(B^{(t+1)},V^{(t+1)},U^{(t+1)}) \in T_m(B^{(t)},V^{(t)},U^{(t)})$, $t = 1,2,\ldots$, and the inequality is strict if $(B^{(t)},V^{(t)},U^t) \notin \Omega$.

(3) Obviously, $T_1 : \mathbb{R}^{N\times p} \times M_{fc} \to \mathbb{R}^{c\times p}$ and $T_2 : \mathbb{R}^{c\times p} \times M_{fc} \to \mathbb{R}^{N\times p}$ are continuous. Next, we will show that T_3 is a closed mapping. To do so, let

$(B^{(t)},\mathrm{V}^{(t)}) \to (B^*,\mathrm{V}^*), \mathrm{t} \to \infty$

$U^{(t)} = T_3(B^{(t)},\mathrm{V}^{(t)}), \mathrm{t} = 1,2,\ldots,$

$U^{(t)} \to U^*, \mathrm{t} \to \infty.$

$u_{ij}^* = \lim_{t\to\infty} u_{ij}^{(t)}$ we have $U^* = T_3(B^*,\mathrm{V}^*)$.

According to Definition 3.1, we can conclude that the mapping $T_3 : \mathbb{R}^{N\times p} \times \mathbb{R}^{c\times p} \to M_{fc}$ is a closed mapping. Then, it is easy to have $T_m : \mathbb{R}^{N\times p} \times \mathbb{R}^{c\times p} \times M_{fc} \to \mathbb{R}^{N\times p} \times \mathbb{R}^{c\times p} \times M_{fc}$ also closed. Of course, T_m is closed over $\{D_f - \Omega\}$, where Ω is the solution set and D_f is defined as $D_f = \{(B,V,U) | B \in \mathbb{R}^{N\times p}, V \in \mathbb{R}^{c\times p}, U \in M_{fc}\}$.

3.5.3. The Proposed Algorithm

In iterative clustering algorithms, the procedure adopted for choosing initial cluster centers is extremely important as it has a direct impact on the formation of the final clusters. If we don't choose it well, the result of the clustering algorithm will be unstable. In general, the clustering algorithms choose the initial centers in random manner, which are affecting the results of clustering when it selects outliers as initial centers. To solve this problem, we propose to measure the initial centers based on the following center initialization algorithm.

Since clusters are separated groups in a feature space, it is desirable to select initial centers which are well separated. The proposed algorithm split data into c groups, based on the observation that some of the patterns are very similar to each other. The algorithm calculates the distance between the elements within each group and computes the mean value of the elements selecting as maximum measure from each group. Through computing the maximum distance between the points within each group, the outliers are screened out.

Algorithm 3.1: **Initialization of the cluster center**

Step 1: Arrange m_j's in ascending order where $m_j = (1/p)\sum_{s=1}^{p} \tilde{\xi}_{js}, j = 1,2,...,N$ for p-dimensional data $\tilde{\xi} = \{\tilde{\xi}_1, \tilde{\xi}_2, ..., \tilde{\xi}_N\}$.

Step 2: Rearrange the data matrix as $\tilde{\xi}' = \{\tilde{\xi}'_1, \tilde{\xi}'_2, ..., \tilde{\xi}'_N\}$ Partition the data into r $\left(r = \lfloor N/c \rfloor\right)$ groups, where c $(1 < c < N)$ is the number clusters, such that, the $(i-1)^{th}$ group contains the $(i-1)^{th}$ r data of $\tilde{\xi}'$, and the cth group contains the remaining all elements, where $i = 2,3,...,c$.

Step 3: Calculating the distance between the elements within each group

Step 4: Select the maximum measure from each group. Find the mean value of the elements $\tilde{\xi}'_j$ and $\tilde{\xi}'_k$ which are in the maximum distance measure in i^{th} group and assign this as i^{th} cluster center, $i = 1,2,...,c$.

The proposed MKPE algorithm described below is carried out iteratively in different steps. The process begins by setting the parameters m, ε and c also by initializing randomly the fuzzy partition matrix U such that $\sum_{i=1}^{c} u_{ij} = 1, 1 \le j \le N$ and $u_{ij} \in [0,1]$ for $i = 1,...,c$ and $j = 1,\ldots,N$ and setting the bias fields to zero values and assigning the initial cluster centers using initialization of the cluster center algorithm. Until the convergence, the fuzzy partition matrix U, the bias field and the centroids are updated. The process stops when the criterion J_m stabilize. That is, the criterion J_m from two successive iterations achieves a stationary value. When the algorithm has converged, the process of defuzzification converts the fuzzy partition U into a hard partition. A wide variety of methods have been proposed to defuzzify the U matrix, among them the maximum membership procedure that assigns the pixel j to the c class with the highest membership is the most important.

Algorithm 3.2: **MKPE Fuzzy Clustering**

Input: Data set X (image), number of clusters c, degree of fuzziness m and a convergence threshold ε

Output: fuzzy membership U , cluster centers matrix V and bias field B

1: randomly Initialize the membership function matrix $U^{(0)}$ such that $\sum_{i=1}^{c} u_{ij} = 1, 1 \le j \le N$ and $u_{ij} \in [0,1]$ for $i = 1,...,c$ and $j = 1,\ldots,N$

2: Assign the initial cluster centers $V^{(0)}$ using initialization of the cluster center algorithm and set $\{\beta_j\}_{j=1}^{N}$ to zero values.

3: $t \leftarrow 1$

4: $J_m^{(t)} \leftarrow 0$, store it as $J_{old}^{(t)}$

5: Calculate $J_m^{(t)}$ using Eq. (3.22), store it as $J_{new}^{(t)}$

6: While $\| J_{new}^{(t)} - J_{old}^{(t+1)} \| > \varepsilon$ do

7: Update matrix of prototype $V^{(t)}$ using Eq. (3.24).

8: Update $B^{(t)}$ using Eq. (3.25)

9: Update matrix of membership degree $U^{(t)}$ using Eq. (3.23)

10: $J_{old}^{(t)} \leftarrow J_{new}^{(t)}$

11: Calculate $J_m^{(t)}$ using Eq. (3.22), store it as $J_{new}^{(t)}$

$t \leftarrow t+1$

12: end while

13: $U^* \leftarrow U^{(t)}$, $V^* \leftarrow V^{(t)}$ and $B^* \leftarrow B^{(t)}$

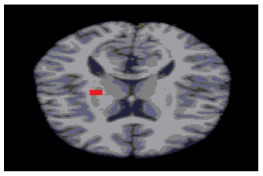
14: Return matrices U^*, V^* and B^*

15: Defuzzification based on the procedure of maximum membership value.

The steps of the proposed MKPE algorithm are illustrated in the sample below. The goal is to segment into clusters the grayscale brain MRI image that is composed of 217×181 pixels. In This case, $p = 1$, $N = 39277$ and $c = 4$ (3 tissues and background). Fig. 3.3a shows the brain MRI, in which the red rectangle is a region of 4×4 pixels. Fig. 3.3b represents the grayscale value of pixels in this region.

149	148	143	141
150	149	145	143
150	149	146	145
150	148	146	145

(a)



(b)

FIGURE 3. 3: (a): shows the brain MRI, in which the red rectangle is a region of 4×4 pixels. (b): represents the grayscale value of pixels in this region.

The fuzzy partition matrix U , whose size is 139277×4 is initialized by randomly value between 0 and 1. The cluster centroids, whose size 1×4 are initialized using algorithm (3.1); and the bias fields whose size is 139277×1 are initialized to zero.

The new fuzzy partition U and the cluster centroids are calculated using Eq. (3.23) and Eq. (3.24) and show below.

$$U^{(1)} = \begin{pmatrix} 0.2398 & 0.1302 & 0.3916 & 0.2384 \\ 0.2653 & 0.1024 & 0.3275 & 0.3048 \\ & & ..... & \\ 0.2452 & 0.1139 & 0.3161 & 0.3248 \end{pmatrix}; \mathrm{V}^{(1)} = (0,57,146,214)^T;\ B^{(1)} = (0.0152,....,0.0152)^T$$

From the result of U , B and V , the value of $\| J_m^{(t)} - J_m^{(t+1)} \|$ is calculated, if it is greater than ε, then the next iteration steps will be performed. Otherwise, the process is stopped.

By the same process, fuzzy partition matrix and cluster centroids will be updated until the stopping condition is reached. The final fuzzy partition matrix and cluster centroids are presented below.

$$U^* = \begin{pmatrix} 0.0043 & 0.9906 & 0.0026 & 0.0025 \\ 0.0043 & 0.9906 & 0.0026 & 0.0025 \\ & & ..... & \\ 0.0043 & 0.9906 & 0.0026 & 0.0025 \end{pmatrix}$$

$$\mathrm{V}^* = (144.0106, 0.0332, 54.1538, 111.5466)^T$$

$$B^* = (0.1938,....,0.1938)^T$$

The partition matrix $U*$ gives the membership degree of each pixel to a class. The largest value of membership degree of the first, second and last pixel is 0.9906, and they are located in the second column so they will be assigned to the second class.

3.6. Experimental Setup and Results

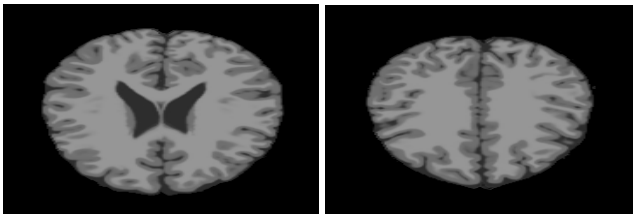
In this section, we describe the dataset used in this study and we assess by different evaluation metrics the performance of our proposed method, MKPE, to accurately segment tissue from simulated normal brain MR images corrupted by different level of noise and intensity inhomogeneity. Through our implementation, the parameters are set as follows: $\boldsymbol{m} = \mathbf{2}$, $\boldsymbol{c} = \mathbf{4}$ and $\varepsilon = 10^{-4}$. All algorithms are run on an Intel Core i5, 2.5-GHz computer with 8GB RAM memory. For performance comparison, the outputs of segmented brain tissue from our proposed method and the six recent methods presented in section 3.4.2 are compared with the ground truth.

3.6.1. Datasets

The quantitative evaluation of the performance of segmentation algorithm is a difficult task because the ground truth of real brain MR image usually does not exist in clinical practice. Brainweb, developed by the brain imaging center of the Montreal Neurology Institute at McGill University, provides a synthetic brain database (SBD) [17], including a set of data generated by an MRI simulator that is close to real brain MR images. The brain MR image data in SBD can simulate T1, T2 and PD weighted brain MR images, and the slice thickness, noise level and intensity inhomogeneity can be set by oneself. According to the requirement of the research task, 3D image can be sliced and extracted from three planes (axial, sagittal and coronal) to obtain 2D image data.

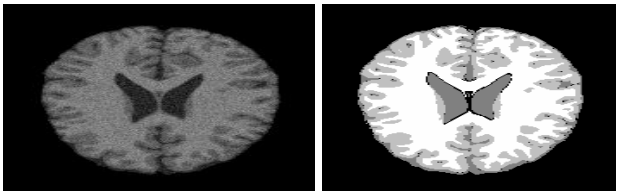
The experiments were carried out on two simulated normal brain MR images in the size of 217×181 pixels. The first one is corrupted by 5%, 9%, 13% and 20% noise each having 20% intensity inhomogeneity. The second one

corrupted by 5%, 9%, 13% and 20% noise each containing 40% intensity inhomogeneity. Fig. 3.5e shows the original input image with 9% noise and 20% intensity inhomogeneity. Furthermore, Fig. 3.5f depicts its ground truth.
The two simulated normal brain MR images are acquired T1-weighted with slice thickness of 1mm and having four classes (c=4) include background and three brain tissues (grey matter, cerebrospinal fluid and white matter). The two normal brain MR images are shown in Fig. 3.5.



(a) (b)

FIGURE 3. 4: Test images. (a) – (b): two original slices of brain MRI from the simulated data



(e) (f)

FIGURE 3. 5: (e): Image (a) corrupted with 9% noise and 20% intensity inhomogeneity. (f): ground truth

3.6.2. Evaluation Metrics

This subsection describes Accuracy, Dice and Geometric mean metrics used in this study for evaluating the performance of predicted segmentation in comparison with reference segmentation.

- SØrensen-Dice coefficient

The sØrensen-Dice coefficient (Dice) [27]; also known as overlap index, is a statistic especially used in the medical images for comparing the similarity between the result of automated segmentation and ground truth.
The Dice is defined as the ratio of overlapping segmented and truth region.

$$D(S_g, S_t) = \frac{2|S_g \cap S_t|}{|S_g| + |S_t|}$$

Where S_g and S_t are the set representing respectively the ground truth and computed segmentation whereas $|.|$ represents the cardinality of a binary set.
$D(S_g, S_t) \in [0,1]$ with $D(S_g, S_t) = 1$ when the segmented region S_t is completely overlapping with the truth region S_g and $D(S_g, S_t) = 0$ when the sets S_g and S_t are disjoint.

- Accuracy

The accuracy metric [103] is the most well-known measure for segmentation evaluation. It measures the proportion of correctly classified pixels and it is computed as

$$Accuracy = \frac{Correctly\ classified\ pixels}{Total\ number\ of\ pixels}$$

- Geometric mean measure

The geometric mean (G-mean) [6] is a common metric focused on the evaluation between ground truth and automatic segmentation. However, it is a balance measure between sensitivity and specificity. A low G-mean is an indication of a poor performance in the classification of the positive cases even if the negative cases are correctly classified as such. This measure is important in the avoidance of overfitting the negative class and under fitting the positive class.

$$G-mean=\sqrt{sensitivity.specificity}$$

The sensitivity and specificity measures are defined as [6]

$$Sensitivity=\frac{TP}{TP+FN}\ ;\ Specificity=\frac{TN}{FP+TN}$$

Where TP and TN refer to the number of pixels correctly classified as positive and negative (background) class, whereas FP and FN are the number of pixels incorrectly labeled as positive and negative examples.

3.6.3. Results and Discussion

3.6.3.1. MR Brain Image Corrupted with Noise and Intensity Inhomogeneity

To demonstrate the performance of our method on medical images, we will test our algorithm on the dataset described in section 3.6.1 and we will compare the results with other competing methods. We will focus on the segmentation of WM, GM, and CSF, i.e., the three main tissues of the human brain. Fig. 3.6 gives a qualitative comparison between segmentation algorithms. (e), (f), (g) and (h) illustrates the GM, WM, CSF soft tissues regions and global segmented image generated by the KIFCM (with $a=1,\ b=2$ and $\alpha=0.5$), AKFCMS (with $\alpha=3.8$ and $S_r=9$ (a 3×3 window centered around each pixel)), KGFCM-S (with $\alpha=0.99$, $\beta=3$ and a 3×3 window centered at a pixel is used to compute the spatial neighborhood information for each pixel in the image) , KFCM_S (with $\alpha=3.8$ and $N_r=9$ (a 3×3 window centered around each pixel)), KRFCMSC (with $\alpha=1$), KFLICM and the proposed MKPE method (with $w=0.5$ and the neighborhood is a 3×3 pixel window).

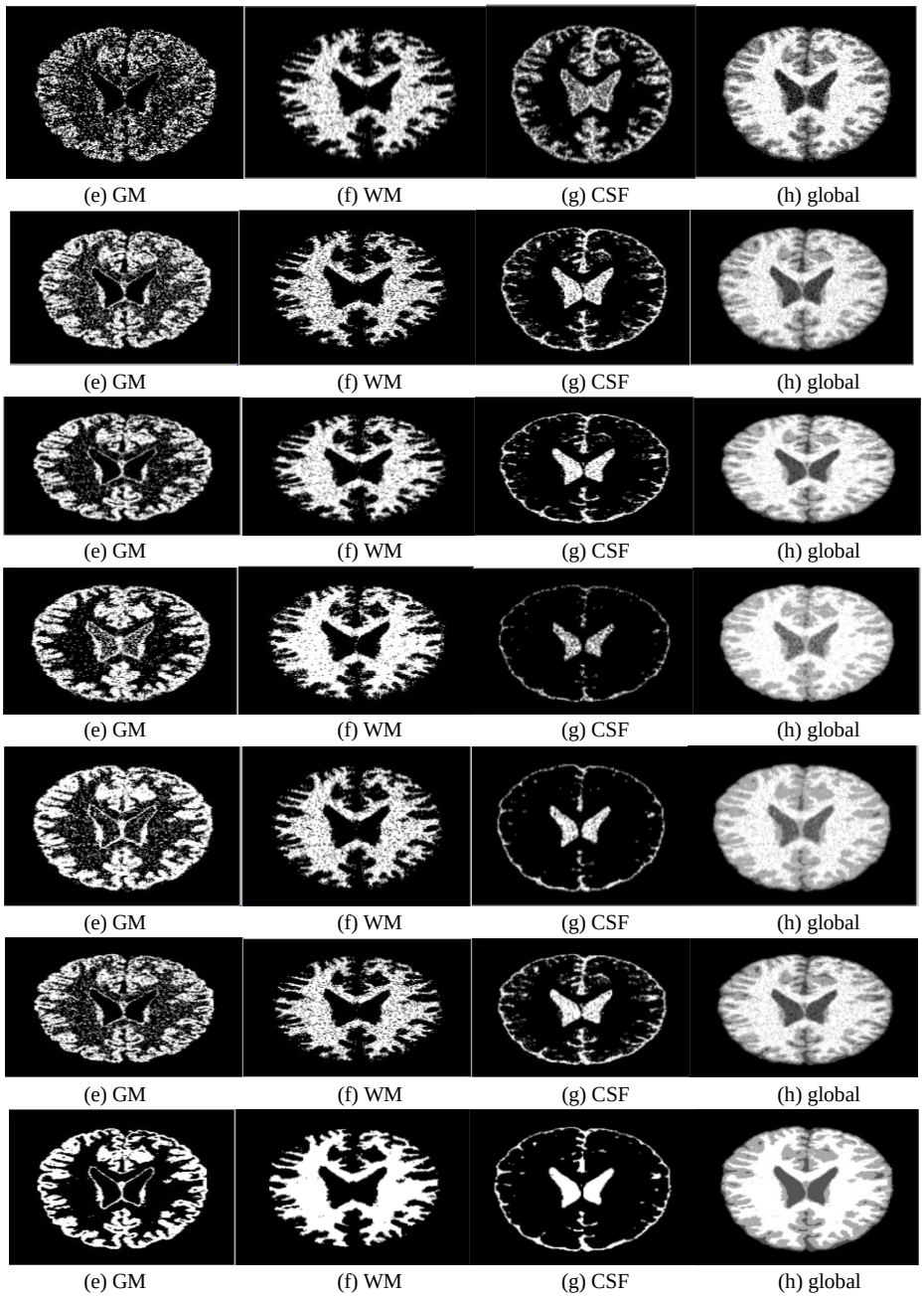


FIGURE 3. 6: The images from the first to 7th row represent respectively the segmentation results of GM, WM and CSF tissues and global segmentation from KIFCM, KGFCM-S, KFCM_S, AKFCMS, KRFCMSC, KFLICM and our proposed MKPE for simulated brain MRI with 9% noise and 20% inhomogeneity.

As observed from Fig. 3.6, our proposed model MKPE clearly justify the efficiently segmentation of the GM, CSF and WM tissue regions and the reduction of the noise presented in the image. Whereas, the six others methods are unable to classify some pixels to their regions in the case of images affected by noise.

In Fig. 3.5e, it can be seen that WM and GM were intertwined with each other due to severe noise, which meant that it was difficult to segment them accurately without an effective model. Since KIFCM does not consider local information and thus is sensitive to noisy pixels, it produces incorrect results in the case of severe noise. Therefore, it can be clearly observed that there were many outliers in the segmentation results of KIFCM. In Fig. 3.6, the 4th row shows that noisy pixels were clearly presented in the segmentation results of AKFCMS because the local spatial information was not anisotropic, so the noisy pixels in the neighborhood directly led to misclassification.

KFCM_S method gives bad performance in the presence of noise and cannot correctly classify different tissues; this is because it is affected by the parameter α.

Comparing the segmentation results of all models with the ground truth (GT), it can be seen that the segmentation results of the proposed MKPE were the closest to GT.

As mentioned earlier, the proposed method MKPE is validated by using Accuracy, Dice and G-mean evaluation measures with various noise and intensity inhomogeneity levels and applied to brain MRI image. The performance of our method is assessed by evaluating the segmentation accuracy of CSF, WM and GM brain tissues and by comparing experimental outcome with the ground truth.

Table 3.1 shows the comparison between six existing methods with our method in terms of segmentation accuracy on simulated brain MR image (a) presented in Fig. 3.4 with 5%, 9%, 13% and 20% each having 20% intensity inhomogeneity. Table 3.2 illustrates the comparison between six existing methods with our proposed method in terms of segmentation accuracy on simulated brain MR image (b) presented in Fig. 3.4 with 5%, 9% and 13% and 20% noise each having 40% intensity inhomogeneity.

Table 3. 1: Comparative study of six methods and the proposed method in terms of tissue segmentation accuracy on simulated brain MR image (a)

Volumes	**Methods**	**Tissue segmentation accuracy**								
		WM			**GM**			**CSF**		
		Accuracy	**Dice**	**G-mean**	**Accuracy**	**Dice**	**G-mean**	**Accuracy**	**Dice**	**G-mean**
5%_20%	**KIFCM**	0.9801	0.9590	0.9785	0.9511	0.8618	0.9377	0.9765	0.8061	0.8292
	KRFCMSC	0.9799	0.9578	0.9679	0.9687	0.9093	0.9474	0.9901	0.9304	0.9543
	KFLICM	0.9623	0.9147	0.9200	0.9463	0.8515	0.9393	0.9853	0.9049	0.9658
	KFCM_S	0.9813	0.9609	0.9726	0.9707	0.9137	0.9354	0.9918	0.9246	0.9681
	KGFCM-S	0.9811	0.9611	0.9751	0.9708	0.9131	0.9409	0.9781	0.9265	0.9671
	AKFCMS	0.9814	0.9608	0.9725	0.9692	0.9107	0.9398	0.9921	0.9227	0.9637
	MKPE	**0.9840**	**0.9669**	**0.9828**	**0.9715**	**0.9146**	**0.9500**	**0.9924**	**0.9309**	**0.9685**
9%_20%	**KIFCM**	0.9702	0.9390	0.9672	0.9393	0.8280	0.9149	0.9740	0.7809	0.8081
	KRFCMSC	0.9701	0.9362	0.9435	0.9387	0.8344	0.9355	0.9739	0.7804	0.8074
	KFLICM	0.9524	0.8912	0.9018	0.9312	0.8097	0.9114	0.9792	0.8694	0.9571
	KFCM_S	0.9708	0.9386	0.9583	0.9577	0.8746	0.9292	0.9883	0.9198	0.9567
	KGFCM-S	0.9712	0.9399	0.9626	0.9573	0.8704	0.9182	0.9849	0.8964	0.9562
	AKFCMS	0.9704	0.9372	0.9545	0.9569	0.8733	0.9318	0.9874	0.9142	0.9561
	MKPE	**0.9797**	**0.9585**	**0.9806**	**0.9670**	**0.8997**	**0.9358**	**0.9900**	**0.9293**	**0.9578**
13%_20%	**KIFCM**	0.9442	0.8741	0.8950	0.8707	0.5605	0.6854	0.8942	0.7374	0.7580
	KRFCMSC	0.9442	0.8741	0.8950	0.9154	0.7850	0.9216	0.9767	0.8099	0.8364
	KFLICM	0.9408	0.8643	0.8838	0.9112	0.7491	0.8638	0.9656	0.7895	0.9394
	KFCM_S	0.9505	0.8919	0.9159	0.9314	0.8035	0.8953	0.9778	0.8497	0.9325
	KGFCM-S	0.9316	0.7482	0.8379	0.8995	0.7250	0.8570	0.9590	0.7482	0.9110
	AKFCMS	0.9542	0.9064	0.9455	0.9131	0.7599	0.8782	0.9643	0.6713	0.7168
	MKPE	**0.9795**	**0.9572**	**0.9731**	**0.9660**	**0.8992**	**0.9450**	**0.9890**	**0.9223**	**0.9552**
20%_20%	**KIFCM**	0.9476	0.8826	0.9022	0.8727	0.5632	0.6848	0.8934	0.7526	0.7533
	KRFCMSC	0.9119	0.8912	0.9508	0.9339	0.8155	0.9121	0.9841	0.8920	0.9339
	KFLICM	0.9443	0.8734	0.8921	0.9164	0.7630	0.8721	0.9689	0.8085	0.9482
	KFCM_S	0.9508	0.8912	0.9119	0.9311	0.8046	0.8994	0.9782	0.8552	0.9446
	KGFCM-S	0.9502	0.9012	0.9511	0.9350	0.7938	0.8557	0.9842	0.8851	0.9223
	AKFCMS	0.9559	0.9083	0.9421	0.9162	0.7724	0.8919	0.9658	0.6898	0.7314
	MKPE	**0.9717**	**0.9396**	**0.9541**	**0.9569**	**0.8755**	**0.9387**	**0.9867**	**0.9070**	**0.9538**

Table 3. 2: Comparative study of six methods and the proposed method in terms of tissue segmentation accuracy on simulated brain MR image (b)

Volumes	Methods	Tissue segmentation accuracy								
		WM			GM			CSF		
		Accuracy	Dice	G-mean	Accuracy	Dice	G-mean	Accuracy	Dice	G-mean
5%_40%	KIFCM	0.9626	0.9487	0.9755	0.9625	0.8998	0.9394	0.9805	0.7467	0.7822
	KRFCMSC	0.9792	0.9496	0.9559	0.9590	0.8949	0.9379	0.9807	0.7467	0.7822
	KFLICM	0.9708	0.9349	0.9615	0.9635	0.9151	0.9394	0.9799	0.8802	0.9251
	KFCM_S	0.9717	0.9526	0.9658	0.9648	0.9187	0.9398	0.9806	0.8902	0.9215
	KGFCM-S	0.9715	0.9524	0.9647	0.9669	0.9152	0.9391	0.9847	0.9339	0.9441
	AKFCMS	0.9727	0.9516	0.9755	0.9598	0.8934	0.9348	0.9780	0.7012	0.7446
	MKPE	**0.9796**	**0.9531**	**0.9768**	**0.9720**	**0.9192**	**0.9413**	**0.9897**	**0.8906**	**0.9455**
9%_40%	KIFCM	0.9706	0.9409	0.7815	0.9542	0.8763	0.9380	0.9807	0.7463	0.7815
	KRFCMSC	0.9744	0.9382	0.9503	0.9541	0.8817	0.9376	0.9805	0.7461	0.7813
	KFLICM	0.9742	0.9379	0.9501	0.9622	0.8945	0.9375	0.9850	0.8562	0.8731
	KFCM_S	0.9752	0.9407	0.9552	0.9661	0.9055	0.9343	0.9775	0.8648	0.9380
	KGFCM-S	0.9664	0.9254	0.9584	0.9584	0.8771	0.9072	0.9819	0.8537	0.9337
	AKFCMS	0.9745	0.9410	0.9643	0.9501	0.8668	0.9243	0.9769	0.6798	0.7258
	MKPE	**0.9780**	**0.9488**	**0.9691**	**0.9692**	**0.9123**	**0.9422**	**0.9882**	**0.8759**	**0.9412**
13%_40%	KIFCM	0.9667	0.9232	0.9551	0.9455	0.8530	0.9233	0.9795	0.7279	0.7674
	KRFCMSC	0.9663	0.9213	0.9419	0.9561	0.8778	0.9280	0.9758	0.7385	0.7553
	KFLICM	0.9483	0.8644	0.8766	0.8078	0.7121	0.7721	0.8131	0.7697	0.5845
	KFCM_S	0.9673	0.9217	0.9423	0.9542	0.8712	0.9207	0.9673	0.8343	0.9225
	KGFCM-S	0.9567	0.8899	0.9022	0.9455	0.8581	0.9381	0.9866	0.8463	0.9235
	AKFCMS	0.9667	0.9232	0.9551	0.9416	0.8443	0.9211	0.9758	0.6612	0.7120
	MKPE	**0.9797**	**0.9529**	**0.9723**	**0.9689**	**0.9106**	**0.9453**	**0.9871**	**0.8648**	**0.9371**
20%_40%	KIFCM	0.9574	0.9021	0.9435	0.9351	0.8237	0.9020	0.9784	0.7115	0.7561
	KRFCMSC	0.9515	0.8793	0.9024	0.8394	0.7848	0.5208	0.8526	0.7226	0.6289
	KFLICM	0.9407	0.8444	0.8642	0.8060	0.7817	0.7401	0.8053	0.7553	0.5636
	KFCM_S	0.9556	0.8924	0.9197	0.9359	0.8191	0.8860	0.9748	0.7662	0.9121
	KGFCM-S	0.9571	0.9025	0.9472	0.9402	0.8193	0.8638	0.9763	0.7678	0.9097
	AKFCMS	0.9569	0.9024	0.9470	0.9299	0.8101	0.8936	0.9737	0.6223	0.6818
	MKPE	**0.9777**	**0.9476**	**0.9648**	**0.9687**	**0.9118**	**0.9370**	**0.9839**	**0.8337**	**0.9255**

From the quantitative evaluation of the results reported in Table 3.1 and Table 3.2, we can observe that our proposed method MKPE outperform consistently than the other methods when noise and intensity inhomogeneity are present in the image. Indeed, we notice that the values found by our method MKPE exceeds in several cases those found by the other methods. For example the segmentation of CSF tissue by implementing the proposed MKPE model on simulated brain MR image (a) corrupted by 5% Gaussian noise and 20% inhomogeneity has achieved an accuracy of 99.24%, G-mean of 93.09% and Dice coefficient of 96.85%, the segmentation of GM tissue by implementing the proposed MKPE model has achieved an accuracy of 97.15%, G-mean of 91.46% and Dice coefficient of 95% and %, the segmentation of WM tissue by implementing the proposed MKPE model has reached an accuracy of 98.40%, G-mean of 96.69% and Dice coefficient of 98.28%. However, the segmentation of CSF tissue by implementing KIFCM method has achieved an accuracy of 97.40%, G-mean of 78.09% and Dice coefficient of 80.81%. The segmentation of GM tissue by implementing KIFCM method has achieved an accuracy of 93.93%, G-mean of 82.80% and Dice coefficient of 91.49%. Moreover, the segmentation of WM tissue by implementing KIFCM model has reached an accuracy of 97.02%, G-mean of 93.90% and Dice coefficient of 96.72%.

Obviously, the segmentation result of KIFCM, KFCM_S, KGFCM-S, KRFCMSC, AKFCMS and KFLICM degrade with the increase of the intensity in-homogeneity and the noise levels.

The results of our experimentation on normal brain MRI images corrupted by Gaussian noise and intensity inhomogeneity, prove that the proposed model is more robust for the reducing of the noise; and it achieved better results in terms of segmentation accuracy compared to other existing methods cited in section 3.4.2.

3.6.3.2. Time Complexity

The following table shows the comparison of average computational time in seconds of the proposed MKPE algorithm and KIFCM, KFCM_S, KGFCM-S, KFLICM, KRFCMSC, AKFCMS algorithms. As can be seen in Table 3.3, the computational time are remarkable the KFLICM and AKFCMS algorithms take 272 seconds and 142 seconds, respectively, which is very long time compared to the others. The high computational complexity due to such introduction of local and spatial constraints. Whereas, the proposed MKPE algorithm requires only 19 seconds; it is executed in a few seconds as KIFCM. This because of the use of a local filtering in each pixel before segmentation process.

Table 3. 3: Average computational time (in seconds) with MATLAB on MRI images of KIFCM, KRFCMSC, KFLICM, KFCM_S, KGFCM-S, AKFCMS and MKPE algorithms

Algorithms	**Computational time (s)**
KIFCM	21
KFCM_S	63
KGFCM-S	82
KFLICM	272
KRFCMSC	74
AKFCMS	142
MKPE	19

The proposed MKPE method gives better results in terms of accuracy and computational time compared to those studied in this work. This is because of the use of a term that models the nonlinear bias field to control the effect of nonlinearity so that tissue properties can be retained leading to accurate classification and the use of a technique which makes it possible to reduce the amount of noise and calculations performed during the segmentation process. This technique used a local filtering in each pixel before segmentation process; based on taking a small neighborhood around a pixel and performing the calculate of the mean values to replace the central pixel. In other words, the elimination of noise at a pixel was local to its neighborhood.

3.7. Conclusion

In the process of brain magnetic resonance imaging, noise and intensity inhomogeneity is an inherent defect. In medical image segmentation, it will affect the segmentation accuracy of brain tissue extraction, and may further affect the treatment effect in clinical diagnosis. A novel effective fuzzy clustering model (MKPE) for image segmentation is proposed, which has the ability to reduce the noise and intensity inhomogeneity as well as the computational time. This method is an improvement of kernel fuzzy c-means model formulated by modeling the nonlinear bias field and by integrating a penalty type entropy term. In this model a local filtering in each pixel is applied to considerably reduce the amount of calculations performed during the segmentation process and to reduce the noise presented in images. MKPE is applied to simulated normal brain MRI dataset that contain different levels of noise and intensity inhomogeneity, and compared to several state-of-the-art improvement schemes. The experimental results show that MKPE is robust to noise and intensity inhomogeneity and can accurately extract the brain tissues: WM, GM and CSF, as well as its calculation speed is fast.

A core problem in applying the existing clustering methods is that the number of clusters needs to be pre-specified before the clustering is carried out. The number of clusters c is eighter identified by users based on prior information. Clustering results may largely depend on the number of clusters which always needs to be specified in order to perform any clustering analysis. Clustering with different c values will eventually produce different results. Since this number is rarely previously known, it's necessary to provide guidance for determining the

number of clusters in order to achieve appropriate clustering results. However, for well-separated classes, the clustering algorithms generally find the same number of clusters. The problem arises in the case of overlapping clusters: rare are the algorithms which detect the real number of classes, and they become invalid for a relatively high degree of overlap. Discovering this number will be explored in the following chapter.

CHAPTER 4

Cluster Validity Indices in Medical Image Segmentation

Contents

4.1. Introduction 56
4.2. Fundamental Principle of Validity Indices 57
4.2.1. Problem Position 57
4.2.2. Process for Finding the Optimal Number of Clusters 58
4.3. Fuzzy C-Means Clustering Method 58
4.4. Fuzzy Cluster Validity Indices 59
4.4.1. Review of Fuzzy Validity Indices 59
4.4.2. Mathematical Background of Some Fuzzy Cluster Validity Indices 60
4.4.2.1. Wu-and-Li Validity Index (V_{WLI} ↓) 60
4.4.2.2. Separation Validity Index (V_{CVI_2} ↑) 61
4.4.2.3. Voscar Validity Index (V_{oscar} ↓) 61
4.4.2.4. Weighted Summation Type of CVI: (V_{WSCVI} ↓) 63
4.4.2.5. CSα Validity Index ($V_{CS\alpha}$ ↓) 63
4.4.2.6. HF Validity Index (V_{HF} ↓) 64
4.5. The Proposed Validity Index 64
4.6. Experimental Results and Analysis 69
4.6.1. Results using Brain MR images 69
4.6.2. Results using Leukemia Images 70
4.7. Conclusion 72

4.1. Introduction

With the increasing size and number of medical images, the use of computers in facilitating their processing and analyses has become necessary. In particular, as a task of delineating anatomical structures and other regions of interest, image segmentation algorithms play a vital role in numerous biomedical imaging applications. Classically, image segmentation of achieving a compact region-based description of the image scene by decomposing it into meaningful or spatially coherent regions sharing similar attributes. In recent years, a number of very inspiring and pioneering image segmentation algorithms have been developed. In this study, clustering-based image segmentation approaches will be considered. Comparing with hard clustering algorithm, fuzzy clustering can retain more information in the image because of introducing the fuzziness for the belongingness of

each image pixel. One of the most important and widely used fuzzy clustering methods is the fuzzy c-means (FCM) algorithm, which was first proposed by Dunn [31] and promoted as the general FCM clustering algorithm by Bezdek [15]. The main purpose of the FCM algorithm is to make the vector space of a sample point be divided into a number of sub-spaces in accordance with a distance measure [13]. However, this algorithm or other clustering algorithms need a user-specified number of clusters or implicit cluster number control parameters in advance. For some applications, the number of clusters can be estimated in terms of the user's expertise or domain knowledge. However, in many situations, the number of clusters for a given data set is unknown in advance. It is well known that estimation of the number of clusters will considerably affect the quality of clustering results. Therefore, identifying the number of clusters in a data set (a quantity often labeled c) is a fundamental issue in clustering analysis.

In this chapter, we propose a new cluster validity index for the FCM algorithm to identify the optimal number of clusters that can improve image segmentation. First, we describe the fundamental principle of validity indices. In this context, an implementation strategy of the different validity indices for finding right number of clusters is presented. Then, we present a mathematical formulation of FCM clustering method used in this study to generate fuzzy partitions and we draw up a state of the art of the validity indices dedicated to fuzzy clustering. Finally, experiments setup, results, discussions and comparisons of the proposed validity index with other published works will be carried out.

4.2. Fundamental Principle of Validity Indices

4.2.1. Problem Position

Finding similar structures in a set of data X, for which there is no information available on the possible groups that compose it, makes the task of a clustering algorithm more and more difficult. Indeed, in clustering algorithms by partition, the choice of the number of clusters is generally fixed by the user. However, a crucial question emerges: (1) What is the clustering tendency, i.e., how many clusters (c) are present in the data set? (2) How to partition the data set into c clusters? (3) How to validate the quality of the obtained partition?

The unsupervised operating mode represents the main characteristic of a clustering algorithm. Like any unsupervised approach, the quality of its results must be validated. Validating the result of a clustering algorithm immediately involves finding the right number of clusters

By "the right number of clusters" we mean that each of these clusters must show an underlying structure to the data and thus facilitate their interpretation. Each cluster must have a meaning for the expert in the field.

Typically, two situations arise:

- Too many clusters: this situation can lead to great confusion because some clusters are "artificial", i.e., they do not represent any reality in the domain concerned
- Not enough clusters: this other case can hide important aspects present in the data.

Faced with this problem the most used approach consists of running a partition-based clustering algorithm such as the FCM for a range of numbers of clusters, testing the results according to a cluster validity index (CVI) [119] and choosing the number that optimizes the validity index. Their performance depends on whether the data set contains well separated clusters, or in other words, whether and how the clusters overlap each other. However, a crucial problem emerges, which concerns the choice of an interval at which the iterative search process should be performed. In this context if we make the initial assumption that every item in dataset X is a cluster, we will have a huge size problem and a very complicated search process. In addition to that, in the majority of real cases, the number of clusters is clearly lower than the number of objects: $c << N$. The hypothesis of limiting the research to a well-defined interval seems more reasonable and better suited to this kind of situation. Usually, the search process takes place between $\left[c_{min},\ c_{max}\right]$ (c_{min} : the minimum number of clusters, c_{max} : the maximum number of clusters).

4.2.2. Process for Finding the Optimal Number of Clusters

Fuzzy clustering algorithm strives to find c compact clusters in X where c is one of the specified parameters. But the process of selecting and adjusting c manually to obtain desirable cluster partitions in a given data set is very subjective and somewhat arbitrary. To seek the optimal cluster structure, *c* is always allowed to be overestimated, such that the distances between some clusters are not big enough or the membership values of some objects with different clusters are adjacent and ambiguous in a given data set. The traditional method to determine the optimal number of clusters is to set the search range of the number of clusters, run fuzzy clustering algorithm to generate clustering results of different number of clusters, select an appropriate clustering validity index to evaluate clustering results, and finally obtain the optimal number of clusters according to the evaluation result. The method is composed of the following steps.

Step 1: Initialize the parameters of the fuzzy clustering algorithm except for the number of clusters, *c*.

Step 2: Run the fuzzy clustering algorithm for different values of *c*'s over the range $c = 2,3,...,c_{\max}$, where $c_{\max}$ is specified by the user.

Step 3: Compute the validity index value $V_d(c)$ for each partition (U,V) obtained from step 2.

Step 4: Choose the optimal partition and the optimal c according to the criteria.

Generally, $c_{\max}$ is unknown. Usually, $c_{\max} = ln(N)$ or $c_{\max} = \sqrt{N}$ are adopted although they have not been proved up to now. In this study, we will adopt $c_{\min} = 2$ and $c_{\max} = \ln(N)$.

In the following section, we will present the fuzzy clustering algorithm used to generate fuzzy partitions of X in M_{fc}

4.3. Fuzzy C-Means Clustering Method

The FCM [12] is a nonlinear optimization problem that has been used successfully to solve problems involving clustering, feature analysis and classifier design in fields such as astronomy, geology, medical imaging, target recognition, and image segmentation. An image can be represented in various feature spaces, and the FCM algorithm classifies the image by grouping similar data points in the feature space into c- fuzzy clusters with respect to some given criteria. In this approach, partitioning is achieved by an iterative process that minimizes the weighted sum of squared errors within-group.

Definition 4.1. Let $X = \{x_1, x_2, ..., x_N\}$ be an image with *N* pixels to be partitioned into *c* clusters, where x_i represents multispectral (features) data and let $U \in M_{fc}$ be a fuzzy partition of the set X, and $V = \{v_1, v_2, ..., v_c\}, v_i \in \mathbb{R}^p$ be the set of unknown cluster centers of fuzzy clusters. The objective function of FCM $J_m : M_{fc} \times \mathbb{R}^{c \times p} \to \mathbb{R}^+$ is defined as [12]:

$$\min_{(U,V)} \left\{ J_m(U,V:X) = \sum_{i=1}^{c} \sum_{j=1}^{N} u_{ij}^m \, \| x_j - v_i \|_A^2 \right\}$$

Where $\| x \|_A = \sqrt{x^T A x}$ an inner product is induced A-norm, $A \in Y_{pp}$ positive definite, Y_{pp} the set of real $p \times p$ matrices and *m* is a parameter controlling the degree of fuzziness $(m > 1)$.

Optimal partitions of X are taken from (U^*, V^*) that are local minimizers of J_m. Approximate optimization of J_m by the FCM algorithm is based on iteration through the following necessary conditions for its local extrema.

Theorem FCM [12]: if $\| x_j - v_i \|_A^2 > 0, \forall i, j$ then $(U,V) \in M_{fc} \times \mathbb{R}^{c \times p}$ may minimize $J_m, m > 1$ only if

$$u_{ij} = \left(\sum_{l=1}^{c} \left(\frac{\| x_j - v_i \|_A}{\| x_j - v_l \|_A} \right)^{2/(m-1)} \right)^{-1}, 1 \le i \le c; 1 \le j \le N \text{ and } v_i = \frac{\sum_{j=1}^{N} u_{ij}^m x_j}{\sum_{j=1}^{N} u_{ij}^m}, 1 \le i \le c \quad (4.1)$$

The basic steps of the FCM algorithm are summarized as follows:

Input: Data set X, convergence threshold ε, number of clusters c and degree of fuzziness m

Output: Fuzzy partition membership U and the set of cluster centroid V.

(A1) Fix m, c and ε satisfying: $m > 1$, $1 < c < N$, and $\varepsilon > 0$. Pick a norm $\|.\|_A$ for J_m, and a termination norm $\|.\|$ for FCM

(A2) Randomly initialize the membership matrix $U^{(0)}$ such that $U^{(0)} \in M_{fc}$ and pick cluster center $V^{(0)} \subset \mathbb{R}^p$

Let $t = 1$

(A3) Compute $U^{(t)}$ using (4.2a) and (4.2b)

If $\forall i, l$, $d_{il} > 0$, then $u_{ij}^{(t)} = \left[\left(\sum_{l=1}^{c} d_{ij} / d_{il} \right)^{2/(m-1)} \right]$ (4.2a)

If exist j, l such that $d_{lj} = 0$ then

$u_{lj}^{(t)} = 1$ and for $i \ne l$, $u_{ij}^{(t)} = 0$ (4.2b)

(A4) Compute $V^{(t)}$ using (4.2.c)

$$v_i^{(t)} = \sum_{i=1}^{N} u_{ij}^m x_j \Big/ \sum_{i=1}^{N} u_{ij}^m \quad (4.2c)$$

(A5) Compare $U^{(t+1)}$ to $U^{(t)}$ in any convenient matrix norm. If $\| U^{(t)} - U^{(t+1)} \| < \varepsilon$ stop, otherwise set $U^{(t)} = U^{(t+1)}$ and $t = t + 1$ and go to (A3)

After several passes through (A3) and (A4), the algorithm will stop, i.e., the error between two consecutive values of the constrained fuzzy partition matrix U will be smaller than an a priori specified level. Convergence of FCM has been proven [15]. Practically numerical convergence is usually achieved in 10-25 iterations.

4.4. Fuzzy Cluster Validity Indices

Validity indices are extremely important for automatically determining the number of clusters and evaluating the fuzzy partitions produced by a fuzzy clustering algorithm. Various validity indices have been proposed in the past. In this section, we present a number of existing indices and we, discuss the main ideas underlying them as well as we provide a mathematical background of the fuzzy cluster validity indices compared in this study.

4.4.1. Review of Fuzzy Validity Indices

A number of validity indices have been proposed and developed to address the problem of finding the number of clusters. We survey some existing cluster validity indices starting with the oldest.

Bezdek has defined two indexes: partition entropy V_{PE} [14] and partition coefficient V_{PC} [14]. Both indices use

only the membership degree. However, they are sensitive to the fuzzifier and they are only useful when the clusters are well-separated. Bensaid et al. have proposed a new validity index [9] which is constructed by summing up the ratio of separation and compactness over all clusters. However, it is sensitive to overlap. In [147] a novel index called Xie-Beni (XB) has been introduced which combines intra-class compactness and separation properties. Though, it is monotonically decreasing when the number of clusters becomes very large and close to the size of the data. Gath & Geva have proposed fuzzy hypervolume index (FHV) which is defined in terms of the covariance matrix determinants and used as a measure for the volume of clusters. Nonetheless it is sensitive to overlapping dataset. K.L. Wu and M.S. Yang [96] have introduced a novel validity index for fuzzy clustering called a partition coefficient and exponential separation (PCAES) index, which uses the factors from a normalized exponential separation and partition coefficient measure for each cluster. This index only employed to detect the data structure with a well-separated clusters and compact partition. In [113] a new index V_{CWB} was proposed focusing on the density of clusters and separation. However, it fails to detect the exact number of partitions on the cluster overlap. Babak Rezaee has proposed V_{SC} index [112] which is combines two measures: the separation and the compactness measures. This index gives good results for clusters of different sizes and densities but it fails when clusters have a high degree of overlap. Zarandi et al. have proposed an Exponential Compactness and Separation validity index (ECAS) [153], which is based on two measures quantifying compactness and separation. These measures indicate homogeneity of the data within clusters and heterogeneity between clusters. This index is able to validate the results of clustering on different types of real data sets and color image segmentation. Nevertheless, it is not applicable to overlapping datasets. S. Joopudi et al, have introduced a new Graded Distance index (GD_index) [67] which uses only the fuzzy membership. Also, it gives the correct number of clusters only on low and non-overlapping datasets. The other improvements of fuzzy cluster validity indexes are reported in [116].

4.4.2. Mathematical Background of Some Fuzzy Cluster Validity Indices

Next, we present the six recent CVIs compared in this work, we avoided the crisp indices, since our purpose was to focus on fuzzy clustering. In short, we focused on fuzzy CVIs that allow us to find the best partition based on its optimal value. For each index, we give an abbreviation which will be useful in the experimental results section. In addition, we accompanied them with an arrow down or up. The up arrow indicates that the maximum value of this index corresponds to a "better" partition. The down arrow means that the minimum value of this index corresponds to a "better" partition.

Let X be a finite unlabeled dataset of N objects in an *p*-dimensional space, and *c* is the number of patterns (*i.e.*, clusters). A partition in X is a set of clusters that partitions it int *c* fuzzy clusters: $FC = \{FC_1, FC_2, ..., FC_c\}$ The result is a partition matrix $U(X)$ denoted as: $U = [u_{ij}]; 1 \le i \le c, 1 \le j \le N$; where its elements $u_{ij} \in [0,1]$ represent the grade of membership of x_j to the cluster FC_i. Furthermore, all $x \in X$ satisfies the constraint that $\sum_{i=1}^{c} u_{ij} = 1$. We will denote $V = \{v_1, v_2, ..., v_c\} \subset \mathbb{R}^p$ as the set of centroid of fuzzy clusters, $\overline{v}$ as the mean vector of V, $\| x_j - v_i \|$ as the dissimilarity measure between the objects x_j and cluster center v_i, $\|*\|$ as any norm metric and m as the degree of fuzziness.

4.4.2.1. Wu-and-Li Validity Index ($V_{WLI} \downarrow$)

The V_{WLI} validity index [144] is proposed for evaluating fuzzy clustering results. Its main characteristic is the introduction of the median distance between a pair of centroids. Similar to the existing CVIs, V_{WLI} evaluates the information about fuzzy compactness and separation of clusters as follows.

V_{WLI} evaluates the compactness of clusters by considering the fuzzy weighted distances and the fuzzy cardinality of clusters. The fuzzy weighted distances reveal the fuzzy deviation of data object x_j from cluster centroid v_i,

which is termed as $u_{ij}^2 \left\| x_j - v_i \right\|$, whereas the fuzzy cardinality of cluster v_i is $\sum_{j=1}^{N} u_{ij}$

The total fuzzy compactness WL_n of all clusters, $FC_k, 1 \le k \le c$ can then be defined as

$$WL_n = \sum_{i=1}^{c} \left(\frac{\sum_{j=1}^{N} u_{ij}^2 \left\| x_j - v_i \right\|^2}{\sum_{j=1}^{N} u_{ij}} \right) \tag{4.3}$$

V_{WLI} evaluates the compactness of clusters by considering the distance between each pair of centroids, which reveals the separation of each pair of clusters. When there are *c* centroids in the clustering analysis, $c(c-1)/2$ distances between *c* centroids can be calculated. The proposed CVI considers the fuzzy separation of clusters by two terms, MIN = $\min_{i \neq l} \left\| v_i - v_l \right\|^2$ and MED = $\underset{i \neq l}{median} \left\| v_i - v_l \right\|^2$. The term MIN is the minimum distance of all $c(c-1)/2$ distances, whereas the term MED is the median of all $c(c-1)/2$ distances and can be obtained by picking up the $\left\lfloor \frac{c(c-1)/2}{2} \right\rfloor$ th value from the sorted $c(c-1)/2$ distances. The separation measure WL_d of V_{WLI} is defined as the average of the minimum and median distances of a pair of centroids, i.e.,

$$WL_d = \frac{1}{2} \left(\min_{i \neq l} \left\| v_i - v_l \right\|^2 + \underset{i \neq l}{median} \left\| v_i - v_l \right\|^2 \right) \tag{4.4}$$

According to the Eq. (4.3) and Eq. (4.4), V_{WLI} is defined as.

$$V_{WLI}(U, V; c) = \frac{WL_n}{2 \times WL_d} \tag{4.5}$$

The constant term 1/2 in the denominator part of Eq. (4.5) is omitted for simplicity and the numerator in Eq. (4.5) is the cardinality average of fuzzy distances of data objects to the centroids of all clusters. A smaller numerator represents the better compactness of clusters. The denominator part represents the separation of clusters. A larger denominator represents better separation of clusters. Therefore, a smaller V_{WLI} value presents a better clustering result and the optimal number of clusters is the number $c*$ whose value of V_{WLI} is minimal.

4.4.2.2. Separation Validity Index $(V_{CVI_2} \uparrow)$

The measure of separation (V_{CVI_2}) [4] is defined as the total difference between membership grades. This measure is given as:

$$CVI_2(U; c) = \frac{1}{c} \sum_{k=1}^{c} \sum_{i=1}^{c} \sum_{j=1}^{N} | u_{ij} - u_{kj} |^m \tag{4.6}$$

The optimal number of clusters is the number c* whose value of V_{CVI_2} is maximal

4.4.2.3. Voscar Validity Index ($V_{oscar} \downarrow$)

The V_{oscar} index [37] is based on two properties: an overlap measurement and an SC measurement. The lower the inter-class overlap is, the more distinct interval of a fuzzy cluster with other fuzzy clusters will get; SC measurement is a measurement for intra-class compactness and inter-class separation. The larger its value is the better balance

effect it can achieve. Thus, a good fuzzy partition is expected to have a low degree of overlap and a large SC measurement.

The overlap between clusters can be quantified by computing an inter-cluster degree of overlap. Considering such situation, data sample x_j meets $\max_{1\le i\le c} u_{ij} < \alpha$, the maximum membership degree of which not up to threshold α, so it may be in the overlapping region of multiple class boundaries. In order to find out such data point, given the threshold δ, if $\left|u_{pj} - u_{qj}\right|$ for $\forall 1 \le p \le c, 1 \le q \le c$, the data sample x_j is considered in the overlapping region of class p and class q. The overlap measurement is defined as follows:

$$overlap(U;c) = \frac{1}{N} \times \frac{2}{c(c-1)} \times \sum_{j=1}^{N} \sum_{1\le p,q\le c}^{R} t(\left|u_{pj} - u_{qj}\right|) \tag{4.7}$$

$$t(\left|u_{pj} - u_{qj}\right|) = \begin{cases} 1, \max\limits_{1\le i\le c} u_{ij} < \alpha, \left|u_{pj} - u_{qj}\right| \le \delta \\ 0.5, \max\limits_{1\le i\le c} u_{ij} < \alpha, \delta \le \left|u_{pj} - u_{qj}\right| \le 2\delta \\ 0, otherwise \end{cases}$$

where R is the number of matrix element that satisfies: $\max_{1\le i\le c} u_{ij} < \alpha, \left|u_{pj} - u_{qj}\right| \le \delta$ or $\max_{1\le i\le c} u_{ij} < \alpha, \delta \le \left|u_{pj} - u_{qj}\right| \le 2\delta$.

Let α and δ be the threshold, when the maximum membership degree is lower than α; this data sample is in the overlapping region of multiple class boundaries. Now, if $\left|u_{pj} - u_{qj}\right| \le \delta$ is satisfied, indicating that the membership level of this data sample to the two clusters is equal. Let $t(\left|u_{pj} - u_{qj}\right|)$ be 1. If $\delta \le \left|u_{pj} - u_{qj}\right| \le 2\delta$ is satisfied, indicating that the membership degree of this data sample to the two clusters has a certain equal possibility, then let $t(\left|u_{pj} - u_{qj}\right|)$ be 0.5. Otherwise, let $t(\left|u_{pj} - u_{qj}\right|)$ be 0. The overlap function $Overlap(c,U)$ is obtained by the average of summing all the above qualified $t(\left|u_{pj} - u_{qj}\right|)$. The smaller its value is, the lower overlapping degree of cluster is.

SC measurement contains two properties: intra-class compactness and inter-class separation. The definition as follows:

$$SC(U,V;c) = \frac{\frac{1}{c}\left(\sum_{i=1}^{c} \left\|v_i - \bar{v}\right\|^2\right)}{\frac{1}{N}\sum_{i=1}^{c}\sum_{j=1}^{N} u_{ij}^m \left\|x_j - v_i\right\|^2} \tag{4.8}$$

where $\frac{1}{c}\left(\sum_{i=1}^{c} \left\|v_i - \bar{v}\right\|^2\right)$ reflects the degree of separation between clusters and $\frac{1}{N}\sum_{i=1}^{c}\sum_{j=1}^{N} u_{ij}^m \left\|x_j - v_i\right\|^2$ reflects the compactness within cluster. It can be seen that the larger $SC(U,V;c)$ value is, the better the clustering partition will get. According to the Eq. (4.7) and Eq. (4.8), V_{oscar} is defined as.

$$V_{oscar}(U,V;c) = \frac{overlap(U;c)}{SC(U,V;c)} \tag{4.9}$$

This index shows that the lower the inter-class overlapping degree is, and the lower $Overlap(c,U)$ value is, the more compact the intra-class and the more separated the inter-class, the larger the $SC(U,V;c)$ value is. Obviously, the better the clustering result is, the lower the V_{oscar} value is. Therefore, the optimal fuzzy c-partition or the optimum value of c is obtained by minimizing V_{oscar} over $c = 2,3,...,c_{\max}$

4.4.2.4. Weighted Summation Type of CVI: ($V_{WSCVI} \downarrow$)

K. Zhou et al. [156] introduced a novel idea of weighted CVIs, which is a weighted sum of ten CVIs for fuzzy clustering. V_{WSCVI} is defined as:

$$V_{WSCVI}(U,V;c) = \sum_{i=1}^{n} w_i CVI_i \tag{4.10}$$

where n is the number of CVIs in V_{WSCVI}. w_i is the weight of index CVI, which represents the relative importance of ith CVI. w satisfies following constraints

$$0 \le w_i \le 1 \tag{4.11a}$$

$$\sum_{i=1}^{n} w_i = 1 \tag{4.11b}$$

CVI_i is one of the following ten fuzzy CVIs: V_{PC} [14], V_{NPC} [114], V_{PE} [14], V_{NPE} [32], V_{XB} [7], V_K [78], V_{PBMF} [108], V_{FS} [41], V_T [129] and V_{SC} [9].

V_{PC}, V_{NPE} and V_{PBMF} indices are maximum type indices, i.e., the optimal cluster number c^* is achieved at the maximum value of these CVIs, while other seven indices are minimum type. In order to obtain c^* when V_{WSCVI} is minimum, we convert the three maximum type indices into their corresponding reciprocal type, which are presented as $V_{PC_r} = 1/V_{PC}$, $V_{NPC_r} = 1/V_{NPC}$ and $V_{PBMF_r} = 1/V_{PBMF}$.

The values of different CVIs change in different range. To overcome the dominate influence of CVIs in large values, all the CVIs are normalized as

$$CVI_i^{'} = \frac{CVI_i - CVI_i^{\min}}{CVI_i^{\max} - CVI_i^{\min}}$$

$CVI_i^{'}$ is the normalized value of CVI_i, $CVI_i^{\min}$ and $CVI_i^{\max}$ are the minimum and maximum values respectively when cluster number c changes from 2 to $c_{\max}$. Then all the CVIs values range in [0, 1].

The corresponding cluster number is optimal cluster number c * when $CVI_i^{'} = 0$ for the ith CVI, and when the minimum value is achieved for V_{WSCVI}.

4.4.2.5. CSα Validity Index ($V_{CS\alpha} \downarrow$)

H.-L. Shieh proposed $V_{CS\alpha}$ validity index [122] which combines compactness, separation, and α-cut properties to indicate the fitness of the cluster number as follows

$$V_{CS\alpha}(U,V;c) = \left(\frac{1}{\tau \times D_{\min}} \right) \times com \tag{4.12}$$

Here,

$$D_{\min} = \min_{i \ne j} \left\{ \left\| v_i - v_j \right\| \right\} \tag{4.13}$$

$$comp = \frac{1}{N}\sum_{i=1}^{c}\sum_{j=1}^{N} u_{ij}^{m} \| x_j - v_i \|^2 . \sum_{i=1}^{c}\sum_{j=1}^{N} u_{ij} \| x_j - v_i \| \quad (4.14)$$

τ is defined as the partition index, for identifying the ratio of data points regarded as clustered correctly in the data set; that is: $\tau = \frac{|F_\alpha|}{|X|}$

$|.|$ represents the number of elements in a set and F_α is defined as: $F_\alpha = \left\{ x_j \in X \mid \exp\left(-\frac{\| x_j - v_i \|}{\rho^2} \right) \geq \alpha \right\}$

Where $\rho = 1/\sqrt{\alpha}, \alpha = 4/r_a$ and r_a is the pre-defined radius.

Cluster number is optimized by solving $\arg\min_{2 \leq c \leq c_{\max}} V_{CS\alpha}(U,V;c)$. The $V_{CS\alpha}(U,V;c)$ is subject to three factors: $1/\tau$, $1/D_{min}$ and com. The first factor is a partition index of the inverse proportion of well-partitioned data to the whole data. A small $1/\tau$ indicates that large amounts of data have been well partitioned. The second factor D_{min}, which represents the minimal distance between clusters, is used to measure the separation between cluster centers. When each cluster is far away from other clusters, the D_{min} is large, and $1/D_{min}$ is small. Finally, the third factor com is used to measure the proper compactness of clusters. When each cluster is compact, the value of com is small. Thus, if the number of clusters is increased from 2 to $c_{\max}$, the values of the first and third factors decrease, and the value of the second factor increases. Thus, these three competing factors are balanced to find the optimal cluster number. The validity indexes in the current literature only consider compactness and separation properties while this validity index includes a partitioned index $1/\tau$ to measure the ratio of well-partitioned data to all data.

4.4.2.6. HF Validity Index ($V_{HF} \downarrow$)

The HF [52] validity index is based on the membership partition. It can be seen as the generalization of the Wu-and-Li (V_{WL}) and Tang (V_T) indices. Its particularity is the integration of a generalized ad-hoc punishing term, on the one hand, and the involving of median between centroids multiplied by the average of data per cluster for computing the separation, on the other hand. It allows avoiding the monotony from which suffer the majority of CVIs and obtaining a precise evaluation. The V_{HF} index is defined as

$$HF(U,V;c) = \frac{\sum_{j=1}^{c}\sum_{i=1}^{N} u_{ij}^{m} \left\| x_i - v_j \right\|^2 + \frac{1}{c(c-1)} \sum_{i=1}^{c} \sum_{k=1,k\neq i}^{c} \left\| v_i - v_k \right\|^2}{\frac{N}{2c}\left(\min_{i \neq j} \left\| v_i - v_j \right\|^2 + \underset{i \neq j}{median} \left\| v_i - v_j \right\|^2 \right)} \quad (4.15)$$

The optimal number of clusters c^* corresponds to the minimum of the V_{HF} index

4.5. The Proposed Validity Index

The fuzzy c-partition provided by fuzzy clustering algorithms attempts to identify the structure that is present in the input data set. In these partitions, the objects assigned to the same clusters are more similar to each other than are objects belonging to different clusters. However, although the environment is fuzzy, the aim of the classification is to generate a well-defined fuzzy c-partition that is as close as possible to the natural structure of the data. Thus, a difficult question is how a partition fits with the unknown structure of the data set. This problem calls cluster analysis using some kind of validity criteria. In the previous section we reviewed the main validity indices dedicated to unsupervised fuzzy classification. Among the difficulties encountered by these indices we can cite the presence of a large variation in the geometric structure of the clusters and the overlapping of the clusters.

In general, we can group the validity indices into those which use the properties of the membership degrees to evaluate a partition and those which use the properties of the dataset X . Recently, Askari et al [3] proposed a validity index called V_{CVI_2} which is defined as the total difference between membership grades. This index uses only the membership degrees which suffers from the absence of information about geometric properties of data. Moreover, the only use of the proprieties of dataset in validity indices suffer from the absence of information about overlapping proprieties of data. In order to overcome these problems, we propose a new function-based validity index for fuzzy c-means algorithm which combined two functions. Each function is defined by the ratio between the total internal cohesion and the separation. The first function, denoted, SC calculates the ratio of a fuzzy compactness-separation by considering geometrical properties and membership of degrees of data. The second function, called PEC, evaluates it by using only the properties of membership degrees. This evaluation, which intended to measure the quality of fuzzy clustering, assigns a real number to the outputs of the used fuzzy clustering algorithm. Two kinds of outputs are considered: the fuzzy membership degrees and the structure of the data set. For each output, a combination of fuzzy compactness and fuzzy separation criteria is evaluated. Thus, a Reliable validation should have a low compactness value, a high separation value and a low degree of overlap.

1. *Definition of* SC

Requirement. A fuzzy c-partition U of the dataset $X = \{x_1, x_2, ..., x_N \mid x_j \in \mathbb{R}^p\}$ with c cluster centers v_i such that $V = \{v_1, v_2, ..., v_c \mid v_i \in \mathbb{R}^p\}$ and $U = [u_{ij} (i = 1, 2, ..., c; j = 1, 2, ..., N)]$.

We introduce some definitions which are used to define SC

Definition 3.2. The fuzzy deviation of objects x_j from cluster v_i is defined as

$$d_{ij} = \left(u_{ij}\right)^{m/2} \left\| x_j - v_i \right\|$$

Definition 3.3. The fuzzy variation of the cluster i is called σ_i defined as

$$\sigma_i = \frac{1}{N} \sum_{j=1}^{N} (d_{ij})^2 = \frac{1}{N} \sum_{j=1}^{N} (u_{ij})^m \left\| x_j - v_i \right\|^2$$

Then, the global compactness of the *c*-partition is

$$comp = \sum_{i=1}^{c} \sigma_i$$

Definition 3.4. Let $U \in M_{fc}$ be a fuzzy partition matrix of the pattern data set X. The fuzzy compactness intra-class $comp : M_{fc} \times \mathbb{R}^{c \times p} \to \mathbb{R}^+$ is defined as:

$$comp(U; X, c) = \frac{1}{N} \sum_{i=1}^{c} \sum_{j=1}^{N} u_{ij}^m \left\| x_j - v_i \right\|^2 \tag{4.16}$$

The fuzzy compactness intra-class $\left(comp\right)$ is defined as the weighted sum of the average distance between the elements of a dataset and its gravity centers. It combines the fuzziness in U with the geometrical compactness of the representation of X via the *c* prototypes. This function indicates the average of the scattering variation within the clusters for c number of clusters. A small value for this term indicates a compact partition. As the scattering within the clusters increases, they become less compact, and therefore $comp(U; X, c)$ is a good indication of the average compactness of clusters. In general, the scattering index $comp(U; X, c)$ does not take any geometric assumption about the prototype into account. Therefore, an optimal partition should have the maximum compactness so that the clusters are far apart.

Definition 3.5. The fuzzy separation inter-class is defined as:

$$sep(V;c,X)=\frac{1}{c}\sum_{i=1}^{c}\left\|v_i-\bar{v}\right\|^2 \tag{4.17}$$

The separation inter-class function indicates the total scattering separation between the clusters. Its measures the distance between the common center of all the clusters and every cluster center. Generally, this term is influenced by the geometry of the cluster centers. Thus, higher value of this measure indicates that the clusters are well separated.

Therefore, the ratio fuzzy compactness/fuzzy separation is defined as the ratio $comp/sep$. Explicitly, SC can be written as

$$SC(U,V;c,X)=\frac{\sum_{i=1}^{c}\sum_{j=1}^{N}u_{ij}^{m}\left\|x_j-v_i\right\|^2/N}{\sum_{i=1}^{c}\left\|v_i-\bar{v}\right\|^2/c}$$

A small value of SC indicates that the fuzzy c-partition is characterized by well-separated and compact fuzzy clusters, i.e., a small value of $comp$ and large one for sep. Therefore, the optimal number of clusters is found by minimizing the distance between data to the corresponding cluster center and maximizes the distance between different cluster center

2. *Definition of PEC*

The second function PEC uses the fuzzy partition matrix U, obtained by the output of the FCM, as the clustering result. This matrix indicates also the detected structure of the data set. It consists of c rows and N columns x_j, $U=[u_{ij}]$. The rows u_i correspond to the obtained fuzzy clusters, considered as c fuzzy sets of X.

Definition 3.6. Let $U\in M_{fc}$ be a fuzzy partition matrix of N data objects. The partition coefficient measure $PC:M_{fc}\to\mathbb{R}^+$ is given as:

$$PC(U;c)=\frac{1}{N}\sum_{i=1}^{c}\sum_{j=1}^{N}u_{ij}^{m} \tag{4.18}$$

The *PC* function is defined by the average sum of the degrees of fuzziness of the elements of a data set at each cluster: it evaluates the amount and overlapping degree between clusters. This function verified that, for all $U\in M_{fcp}$; $1/c\le PC(U;c)\le 1; PC(U;c)=1/c\Leftrightarrow U=[(1/c)]$, i.e., $u_{ij}=1/c,\forall i,j$; and $PC(U;c)=1\Leftrightarrow U\in M_{hc}$ is hard.

The *PC* takes its maximum on every crisp c-partition and it reaches the unique minimum at the centroid $U=[(1/c)]$ of M_{fc}. The more the value of the *PC* function tends to 1, the more the partition contains well separated clusters, and if it reaches the minimum value 1/c then the dataset X or the partition in question doesn't have any cluster structure. Hence, the lower the fuzziness of a partition is, the larger is the *PC* value.

Definition 3.7. Let A be the fuzzy set. The measure of fuzzy entropy is defined as follows.

$$H(A)=\frac{1}{(1-m)}\sum_{j=1}^{N}\log\left(u_A^m(x_j)+(1-u_A(x_j))^m\right) \tag{4.19}$$

where m≠1, m>0.

The average of fuzzy entropy which is defined by Eq. (4.19) is given as follows

$$E(U;c)=\frac{1}{N.(1-m)}\sum_{i=1}^{c}\sum_{j=1}^{N}\log\left(u_{ij}^{m}+(1-u_{ij})^{m}\right) \tag{4.20}$$

The fuzzy entropy is used to measure the fuzziness of the ith cluster, namely the quality of the information contained therein. A cluster with low fuzzy entropy has low fuzziness. So, it is compact. Moreover, a cluster with a high value of fuzzy entropy has high fuzziness and low compactness.

Finally, the ratio fuzzy compactness/separation is defined as the ratio E/PC and PEC is explicitly written as

$$PEC(U;c)=\frac{E(U;c)}{PC(U;c)}$$

We can see that compact and well-separated fuzzy clusters correspond to a low value of PEC, i.e., a small value for E and large one for PC.

The values of SC and PEC functions change in different range. To overcome the dominate influence of these functions in large values, we normalize them.

Finally, the formulation of our proposed V_{ECS} validity index is given as

$$V_{ECS}(U,V;c,X)=SC^{N}(U,V;c,X)+PEC^{N}(U;c) \tag{4.21}$$

Where $SC^{N}(U,V;c,X)$ and $PEC^{N}(U;c)$ respectively represent the normalize value of $SC(U,V;c,X)$ and $PEC(U;c)$. Thus, normalized of these measures are computed as

$$SC^{N}(U,V;c,X)=\frac{SC(U,V;c,X)-SC_{\min}}{SC_{\max}-SC_{\min}} \tag{4.22}$$

$$PEC^{N}(U;c)=\frac{PEC(U;c)-PEC_{\min}}{PEC_{\max}-PEC_{\min}} \tag{4.23}$$

The minimum value of SC and PEC measures are given as

$$SC_{\min}=\min_{c_{\min}\le c\le c_{\max}} SC(U,V;c,X)$$

$$PEC_{\min}=\min_{c_{\min}\le c\le c_{\max}} PEC(U;c)$$

The V_{ECS} measures the degree of correspondence between the structure of the input data set (evaluated with SC) and the fuzzy c-partition resulting from the fuzzy clustering algorithm (evaluated with PEC). The minimum of V_{ECS}, for a given number of clusters c, is expected to detect well defined fuzzy c-partition. It tends to indicate a good cohesion within clusters and a small overlap between pairs of clusters. This is the more desirable partition of X

The computation process of our proposed V_{ECS} to validate the fuzzy partition obtained by FCM clustering algorithm is given by the following algorithm. This process corresponds to the execution of the FCM clustering algorithm, by varying the number of clusters between c_{min} (the minimum number of clusters) and c_{max} (the maximum number of clusters). Therefore, the optimal fuzzy partitions correspond to the minimum values of $V_{ECS}(U,V;c,X)$.

Algorithm 4.1: **Evaluation of the proposed $V_{ECS}(c)$validity index**

Input: X: the dataset, m: the fuzzy degree, ε: convergence error, T: the maximum number of iterations, c_{min} and c_{max} the minimum and the maximum number of clusters.

Output: The values of our new validity index $V_{ECS}(c)$.

1: Initialize $c = c_{\min}; c_{\min} = 2$

2: For $c = c_{\min}$ to $c_{\max}$

3: Choose an initial fuzzy partition matrix $U^{(0)} \in M_{fc}$. Then at step $t = 0,1,2,...,T$

4: Compute cluster prototypes $v_i^{(t)}, i = 1,...,c$ with $U^{(t)}$ and updated membership matrix $U^{(t+1)} = [u_{ij}^{t+1}]$ for $i = 1,...,c$ and $j = 1,...,N$ using Eq. (4.1)

5: Compare $U^{(t+1)}$ and $U^{(t)}$. If $\| U^{(t)} - U^{(t+1)} \| < \varepsilon$, stop; otherwise, set $t = t+1$ and go to step 4

6: Compute the value of entropy measure $E(U;c)$ in Eq. (4.20), partition coefficient $PC(U;c)$ in Eq. (4.18), compactness measure $comp(U;X,c)$ in Eq. (4.16) and separation measure $sep(V;c,X)$ in Eq. (4.17)

7: $c \leftarrow c+1$

8: Compute the normalized separation and compactness measures $SC^N(U,V;c,X)$ in Eq. (4.22) and normalized entropy and partition coefficient measures $PEC^N(U;c)$ in Eq. (4.23) for all values c that varies between $c_{min} = 2$ and c_{max}.

9: Compute the proposed $V_{ECS}(U,V;c,X)$ validity index in Eq. (4.21) for all values of c.

10: Select the optimal number of clusters c* that minimizes V_{ECS}. $c^* = \arg\min_{c_{\min},...,c_{\max}}(V_{ECS}(U;c))$

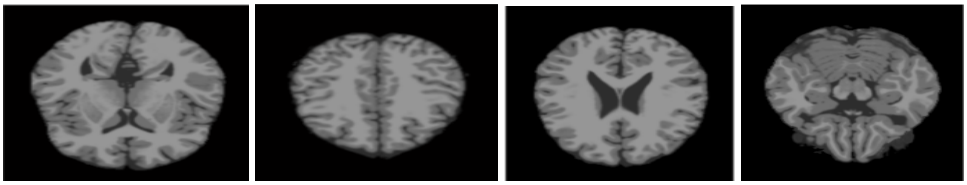
4.6. Experimental Results and Analysis

To validate the performance of our new validity index we perform a comparison with the six recent validity indices described in section 4.4.2 which are, V_{WLI}, V_{CVI_2}, V_{oscar}, V_{WSCVI}, V_{HF} and $V_{CS\alpha}$. The experiments are done on two types of overlapping data sets including brain MRI and leukemia images. In all experiments, the parameters are set as follows: $m = 2$, $\varepsilon = 10^{-4}$, $T = 500$ and $c_{\max}$ is equal to $\ln(N)$. Furthermore, the values of the validity indices V_{WLI}, V_{CVI_2}, V_{HF} and $V_{CS\alpha}$ are normalized in the range $[0,1]$

4.6.1. Results using Brain MR images

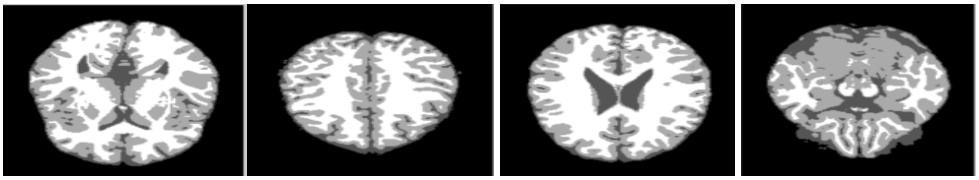
In this subsection, the existing validity indexes introduced in section 3 and the proposed V_{ECS} index have been applied to four simulated normal brain MR images of size 217×181 pixels, taken from the classical Brain Web simulated brain database (www.bic.mni.mcgill.ca/brainweb/) [13]. These images are acquired T1-weighted with slice thickness of 1mm and contain four clusters (c=4) that include three brain tissues (grey matter (GM), cerebrospinal fluid (CSF) and white matter (WM)) and background. The GM and CSF tissues are somewhat overlapped, however; the WM and GM tissues are highly overlapped. The four images are illustrated in Fig. 4.1.

Moreover, these images are segmented into four clusters using the fuzzy c-means algorithm and are shown in Fig. 4.2. Whereas, the values of our proposed V_{ECS} and the validity indexes V_{WLI}, V_{CVI_2}, V_{oscar} (with $\gamma = 0.1$ and $\alpha = 0.6$), V_{HF}, V_{WSCVI} (with equal weights; $w_i = 0.1$) and $V_{CS\alpha}$ (with $\alpha = 0.6$) are presented in Fig. 4.3.



(a) (b) (c) (d)

FIGURE 4. 1: Test images. (a), (b), (c), (d): original slices of brain MRI from the simulated data



(i) (j) (k) (l)

FIGURE 4. 2: (i), (j), (k), (l): segmentation results of the images (a), (b), (c) and (d) using FCM algorithm with c=4.

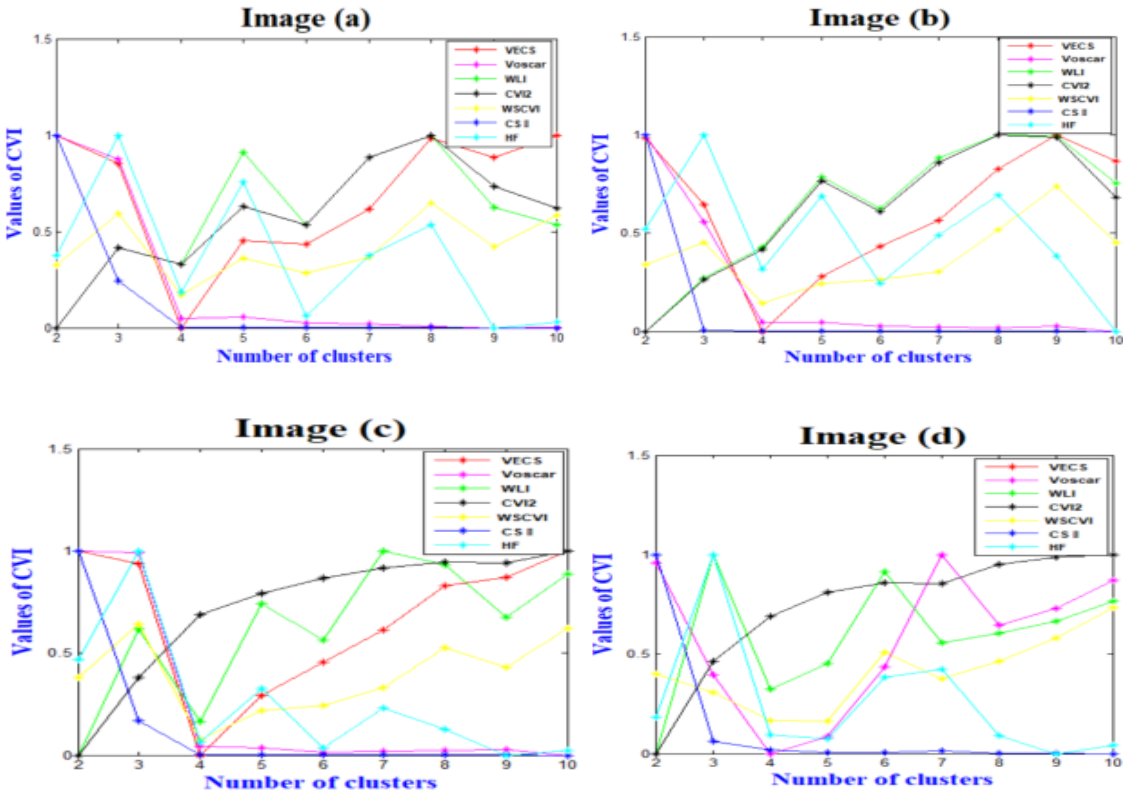
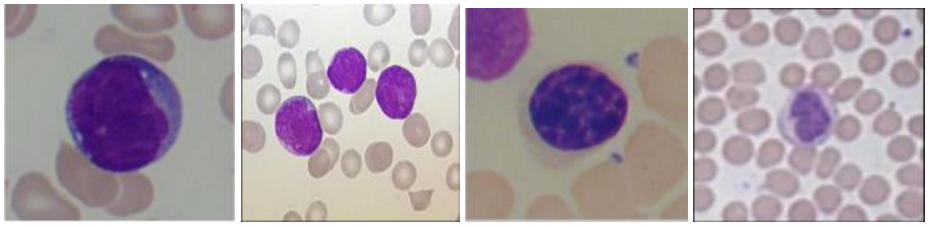


FIGURE 4. 3: The values of validity indexes in the range $c = 2,...,10$ for the image (a) to image (d)

As observed from Fig. 4.3, our proposed V_{ECS} validity index correctly recognize the real number of clusters for all simulated normal brain data sets presented in Fig. 4.1. Whereas the V_{WLI} index fails to find the appropriate number of clusters for images (a), (b), (c) and (d): it reaches the minimum at c^*=2. The V_{CVI_2} validity index achieved the maximum at c^*=10 as optimal number of clusters instead of c^*=4 for brain MR images (a), (b), (c) and (d). The V_{oscar} validity index incorrectly identifies the exact number of clusters: it found c^*=10 for images (a), (b), (c) and c^*=9 for image (d) as optimal numbers of clusters instead of c^*=4. The results of this index are unexpected since it presents a monotonic tendency of the cluster number c where it gives a largest optimal cluster number estimate. The $V_{CS\alpha}$ validity index reaches the minimum at c^*=9 for simulated brain MR image (a), as well as it reaches the minimum at c^*=10 for images (b), (c) and (d) so that are not the correct numbers of clusters. the curve of this index presents a monotonic tendency where they give a largest optimal cluster number estimate. The V_{HF} validity index cannot find the appropriate number of clusters: it reaches the minimum at c^*=9 for image (a), (c), (d) and c^*=10 for image (b). The V_{WSCVI} correctly reveals the number of clusters for images (a), (b), (c) and fails to recognize the appropriate number of clusters for image (d): it reaches the minimum at c^*=5.

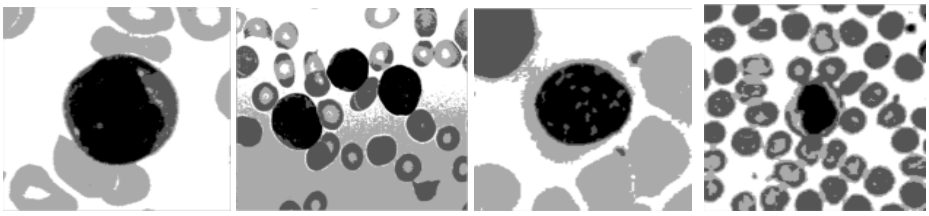
4.6.2. Results using Leukemia Images

In this subsection, four images of the acute leukemia are investigated to assess the efficiency of our validity index. These images are obtained from ALL-IDB [79] and Sigtuple WBC dataset [100], which contain 4 clusters that include Background, nucleus, cytoplasm and red blood cells. Moreover; they have overlapped cells. The four images are shown in Fig. 4.4 and they are converted to grayscale before segmenting them. In addition, these images are segmented into four clusters using the fuzzy c-means algorithm and are illustrated in Fig. 4.5. Whereas, the values of our proposed V_{ECS} and the validity indices V_{WLI}, V_{CVI_2}, V_{oscar}, V_{HF}, V_{WSCVI} and $V_{CS\alpha}$ are presented in Fig. 4.6.



(e) (f) (g) (h)

FIGURE 4. 4: Test images. (e), (f), (g), (h): original images of the Acute Leukemia



(i) (j) (k) (l)

FIGURE 4. 5: (i), (j), (k), (l): segmentation results of the images (e), (f), (g) and (h) using FCM algorithm with c=4

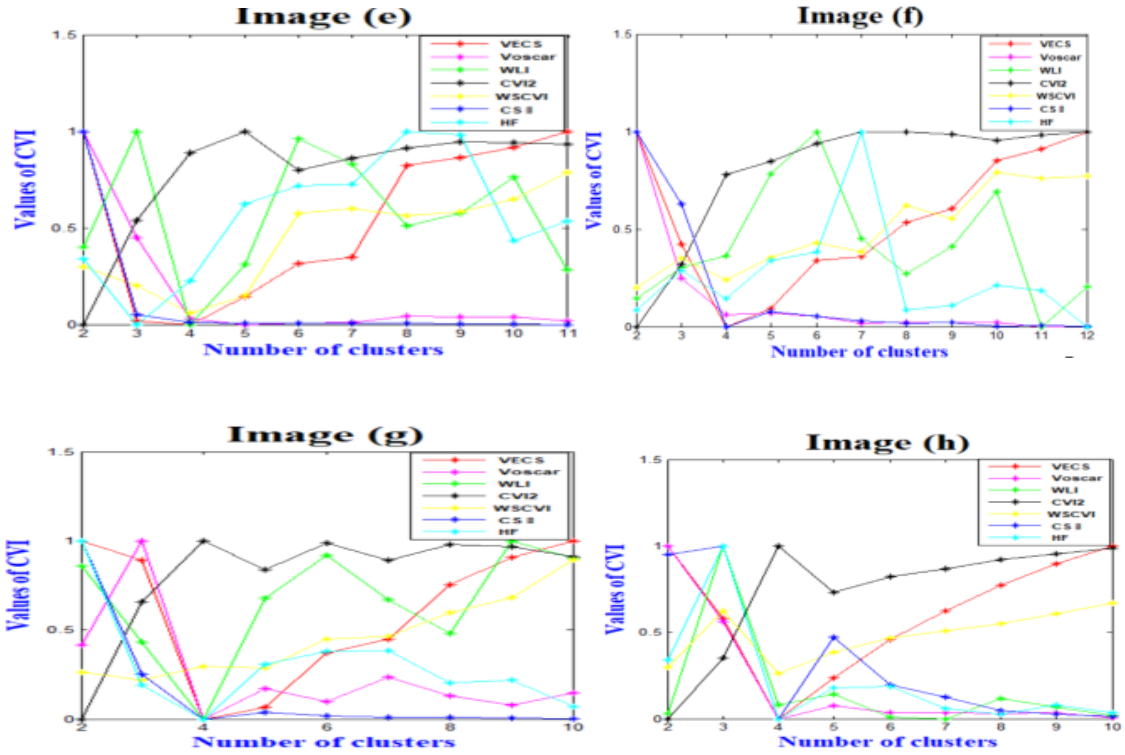


FIGURE 4. 6: The values of validity indexes in the range $c = 2,...,10$ for the image (a) to image (d)

As observed from Fig. 4.6, the proposed V_{ECS} detects the real number of clusters for all images presented in Fig. (4.4). The V_{oscar} index correctly recognizes the number of clusters for images (f) and (h); and cannot find the appropriate number of clusters for other images: it reaches the minimum at c*=5 for image (e) and c*=11 for image (g). The V_{WLI} validity index correctly selects the number of clusters only for images (e) and (f); and fails to identify the appropriate number of clusters for other images: it reaches the minimum at c*=11 for image (g) and c*=7 for image (h). The V_{CVI_2} reveals the exact number of clusters for images (f) and (h). Whereas it identifies 5 and 12 respectively as optimal numbers of clusters for images (e) and (g) which are incorrect numbers. The V_{WSCVI} validity index identifies c* = 4 as the optimal number of clusters for images (e) and (h), which is the exact number. Moreover, it identifies c* = 3 and c*=2 as the optimal number of clusters for images (f) and (g), which are not the real number. The $V_{CS\alpha}$ validity index recognizes the exact number of clusters for images (f), (g) and (h); and fails to detect the real number of clusters for image (e): it found c*=11 as optimal number of clusters instead of c*=4. The V_{HF} validity index correctly identifies the number of clusters for images (e), (f) and (h); and fails to recognize the appropriate number of clusters for image (g): it reaches the minimum at c*=12.

The following table summarizes the optimal number of clusters obtained by our new validity index V_{ECS} and the six validity indexes studied in Section 3 for all data sets presented in section 4.4.2.

Table 4. 1: The optimal number of clusters obtained by six recent validity indices and the proposed index for Brain MRI and Leukemia medical imaging data sets

Data sets	**Real Number of clusters**	V_{ECS}	V_{oscar}	V_{WLI}	V_{CVI_2}	V_{WSCVI}	$V_{CS\alpha}$	V_{HF}
Image (a)	4	4	10	2	10	4	9	9
Image (b)	4	4	10	2	10	4	10	10
Image (c)	4	4	10	2	10	4	10	9
Image (d)	4	4	9	2	10	5	10	9
Image (e)	4	4	5	4	5	4	11	4
Image (f)	4	4	4	4	4	3	4	4
Image (g)	4	4	11	11	12	2	4	12
Image (h)	4	4	4	7	4	4	4	4

From the experiments results reported in table 4.1, the proposed V_{ECS} is the only index that reveals the correct number of clusters. It performs better than the other validity indexes studied in section 4.4.2 for medical images that have overlapped clusters. The V_{WSCVI} index correctly selects the number of clusters only on some images. The V_{oscar}, V_{WLI}, V_{CVI_2}, $V_{CS\alpha}$ and V_{HF} validity indices can no longer identify the exact number of clusters for brain MR images; and they detect it on some images of the acute leukemia. These indices find it difficult to correctly identify the numbers of clusters for overlapped datasets.

4.7. Conclusion

The Fuzzy c-Means Algorithm can partition a data set into any specified number of clusters. However, the difficult problem is to evaluate the quality of the fuzzy c-partitions provided by the output of this algorithm, and to search which partition explains at best the "unknown" structure of the data set analyzed. In literature, numerous techniques, called cluster-validity criterion have been studied. In this Chapter, we have proposed a new validity index V_{ECS} for the fuzzy c-means algorithm to identify the optimal number of clusters that can improve image segmentation; the proposed V_{ECS} combines information about fuzzy separation and fuzzy compactness, and takes into account the properties of the fuzzy membership degrees and of the structure of the data. Thus, the optimal number of clusters *c** corresponds to the minimum values of the proposed V_{ECS}. The reliability of our validity index was compared with recent validity indexes. Moreover, it was validated and applied to a variety of medical imaging data sets including, brain MRI and leukemia images. The experimental results show that our proposed index is more efficient and robust to recognize the appropriate number of clusters on data sets that contain overlap clusters. Overall, the proposed index yielded satisfactory results that were more consistent with medical image segmentation.

Medical image classification is one of the most important problems in the image recognition area, and its aim is to classify medical images into different categories to help doctors in disease diagnosis or further research. The use of the traditional machine learning methods in medical image classification began long ago. However, these methods have many disadvantages. The deep learning methods, especially the convolutional neural networks provide an effective way to construct an end-to-end model that can compute final classification labels with the raw pixels of medical images. Therefore, the classification of medical images based on the use of deep learning methods will be the objective of the next chapter.

CHAPTER 5

Deep CNNs for Medical Image Classification

Contents

5.1. Introduction 74
5.2. Introduction to Artificial Neural Networks 75
5.2.1. Perceptron 77
5.2.2. Multilayer Perceptron 78
5.3. ANN Training Procedure 79
5.3.1. Gradient Descent Methods 79
5.3.1.1. Gradient Descent 79
5.3.1.2. Mini-Batch Stochastic Gradient Descent 80
5.3.1.3. Stochastic Gradient Descent with Momentum 80
5.3.1.4. Adaptive Gradient 80
5.3.1.5. Adaptive Moment Estimation 81
5.3.2. Backpropagation Algorithm 81
5.3.3. The Activation Function 84
5.3.4. Vanishing and Exploding Gradients 86
5.4. Deep Convolutional Neural Networks 86
5.4.1. Components of a DCNN 86
5.4.2. Input Layer 88
5.4.3. Convolutional Layer 88
5.4.4. Pooling Layer 90
5.4.5. Classification Layers 91
5.5. Deep CNN Training 91
5.5.1. Backpropagation in Pooling Layers 92
5.5.2. Backpropagation in Convolutional Layer 93
5.6. Overfitting and Regularization 93
5.6.1. Dropout 94
5.6.2. Weight Decay 95
5.6.3. Batch Normalization 95
5.6.4. Early Stopping 96
5.7. Conclusion 97

5.1. Introduction

Nowadays, the need for an accurate diagnosis that reduces as much as possible the influence of human error has boosted the improvement of computer aided detection and diagnosis systems that provide assistance and support to medical professionals. Moving from a scenario where the medical professional diagnoses based solely on their knowledge and experience, to a scenario where the professional supports his diagnosis in systems trained in a particular problem. In the development of these systems numerous papers support the distinction of two stages.

- Pre-Deep Learning era: in early 2000s the first computer-aided detection (CAD) is created. They were systems built using traditional methods of Machine Learning in which the features form the input were selected and extracted in a totally manual way and based on the criteria that the engineer considered appropriate. Correct selection and extraction of these features had a huge impact on system performance. Early reports warned that these systems generated more false positives and thus worse results than human reading, calling into question the benefit of such systems.
- Post Deep Learning era: Deep Learning would bring a revolutionary breakthrough changing about how the model interacts and learns autonomously discovering complicated patterns and features directly from the input data. The selection and extraction of the feature set without human intervention provided numerous advances and the results improved enormously overcoming the limitations and poor results observed in the previous CAD systems.

The Fig. 5.1 shows the difference between machine learning and deep learning. In the traditional machine learning, the Machine Learning algorithms are applied according to the features selected and extracted manually, while the deep neural network learns how to extract the most representative or complex features and patterns on its own according to the input

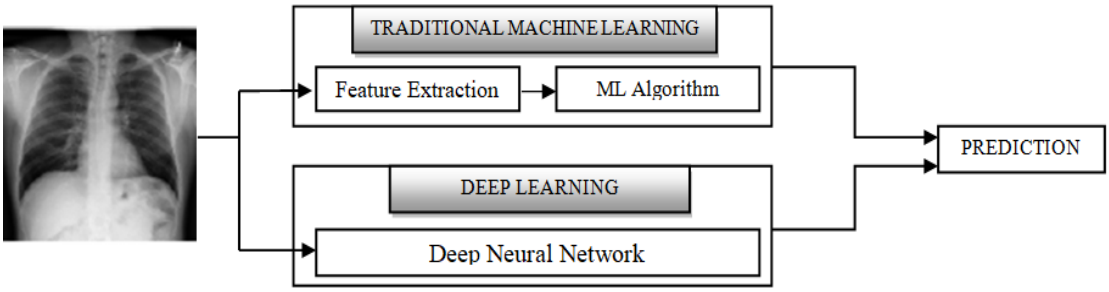


FIGURE 5. 1: Traditional machine learning versus Deep Learning.

The research and development of Deep Learning has advanced giant steps, extending its application in a multitude of areas and varied problems, in which it has been positioned as the most powerful and referential models as for example in tasks of natural language processing, voice recognition, robotic, computer vision or recommendation systems.

In computer vision, an important milestone was reached when in 2012 on the ImageNet Large-Scale Visual Recognition Challenge a Deep Learning model based on convolutional neural networks (CNNs) surpassed the performance obtained with other Machine Learning techniques. This event demonstrated the enormous potential of Deep Learning in image processing tasks. Further developments and studies placed Deep Learning techniques based on convolutional networks as the de facto standard for a wide variety of computer vision task, such as image super-resolution reconstruction, object detection, image classification, and face recognition.

The great impulse of Deep Learning has had its reflection in CAD processing systems the medical images provide a valuable diagnostic tool and assistance to the medical professional.

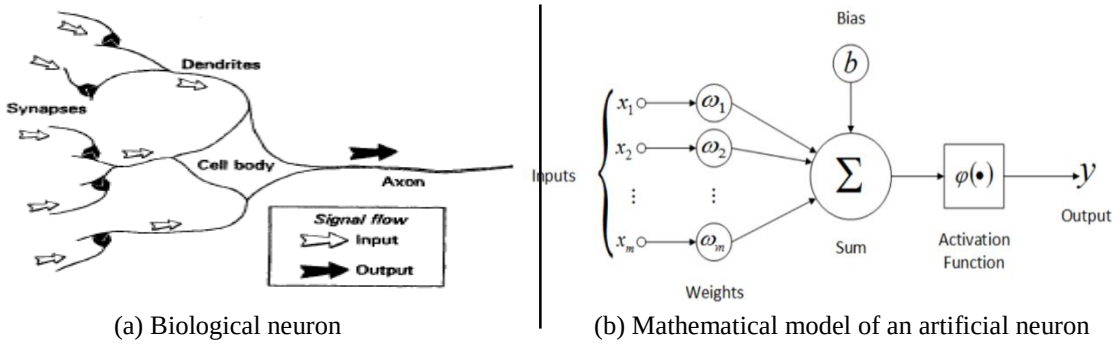
This momentum and success are underpinned by a number of factors:

- Increased availability of digitized data.
- Increased processing capacity, with increasingly powerful GPUs
- A rapid development of algorithms and techniques that have significantly improved the technology initially proposed.
- Economic and investment drive in medical imaging market based on Deep Learning

CNNs have a long history starting with the perceptron algorithm in 1958 [39]. Although CNNs can be explained intuitively as models that learn visual filters to recognize high-level image features, understanding the functionality of neural networks in a first step will facilitate understanding how CNNs work. In this chapter, we first introduce the basis of ANNs. This includes the perceptron, the feedforward neural networks and the training of them. Furthermore, we explain the architecture of CNNs and its training process as well as typical regularizations. Afterwards, we provide our contributions related to classification of medical images using deep CNNs.

5.2. Introduction to Artificial Neural Networks

Artificial Neural Networks also called connectionist networks have been used to solve a variety of problems in different scientific disciplines. For example, problems in pattern recognition, making predictions, optimization problems etc. Already existing approaches solve these kinds of problems. Nevertheless, typically these do not perform well outside their domain. ANNs on the other hand, seem to be suitable to tackle all kinds of problems and many applications could benefit from using ANNs. Indeed, modern computers outperform the human brain in multiple areas, such as numeric computation and related symbol manipulation. Yet the brain (humans) can solve complex perceptual problems without effort, which again is a challenge for computers. An example of this is face recognition. We can recognize a person in a huge crowd by just a glimpse of his/her face. Even the world's fastest computer cannot do this at such a high speed. Therefore, the architecture of the biological neural system is a highly interesting area for researchers. ANN is a machine learning prediction model that can receive inputs from the previous step, transform, and pass outputs to next step. It was inspired by biological neuron system in which neurons can receive and transmit signals to each other to maintain the operation of biological systems. As illustrated in Fig. 5.2a, a neuron contains the cell body, the dendrites, the axon, and the axon terminals. The dendrites are the connections between cells in the brain, were created when learning happens. The dendrites transform the signals to impulses and pass them to cell body. The cell body is central of the neuron that contains all the necessary biological parts to ensure the function of a neuron. Axons are the long paths for transmitting pulses from the cell body to the axon terminals. Axon terminals are responsible for connecting and passing the pulses to other cells.



(a) Biological neuron (b) Mathematical model of an artificial neuron

FIGURE 5. 2: Comparison between a biological neuron (left) and the mathematical model of an artificial neuron (right). In biological networks, the input data comes from another neuron passing through the dendrite, the cell body does the calculations, and the result exits through the axon. The functioning of an artificial neuron is modeled on that of a biological neuron. Image taken from [49].

Generally, ANNs consists of a set of neurons connected to each other and are grouped into layers to replicate the neural function of our brain. Formally, an artificial neuron has *n* inputs represented as a vector $\vec{x} \in \mathbb{R}^n$. Inputs in an artificial neuron correspond to the dendrites in a biological neuron, while a single output of an artificial neuron corresponds to the axon in a biological neuron, which is depicted in Fig. 5.2a. Each input $i,\ 1 \le i \le n$, has an assigned weight. Weighted input values are combined and run through an activation function producing some output *y*, as shown in Fig. 5.2b. The network is formed by connecting the neuron output with the input of a different neuron. ANN is therefore effectively described as an oriented graph, where vertices represent the neurons, and oriented edges represent the output-input connections between them.

A set of input neurons consists of the neurons which are the first ones in any complete path in the graph. All input neurons have exactly one input, and all inputs together represent an instance of the problem to be solved by the ANN. A set of output neurons consists of the neurons which are the last ones in any complete path in the graph. All output neurons have exactly one output, and all outputs together represent a possible solution to the problem to be solved by the ANN. A set of hidden neurons consists of the neurons which are neither input, nor output neurons. Their number and organization into layers may vary even for the same problem, but is a key feature of the network vastly influencing its performance. An ANN works by feeding the data into the input neurons. The data flows in the direction of oriented edges and ends when the output neurons are hit. The result is interpreted from the values obtained in the output neurons.

History of ANNs can be arguably dated from 1943, when Warren Mcculloch and Walter Pitts devised mathematical model inspired by Biology of central nervous systems of mammals [92]. This inspired the invention of Perceptron, created in 1958 by Frank Rosenblatt. Perceptron used very simple model mimicking biological neuron that was based on mathematical model of Pitts and Mcculloch. Definition of the Perceptron model also described an algorithm for direct learning from data.

In the beginning Perceptron seemed very promising, but it was soon discovered that it had severe limitations. Most prominent voice of criticism was Marvin Minsky. Minsky published book in which he laid out a case that Perceptron model was unable to solve complex problems [97]. Among others the book contained mathematical proof that Perceptron is unable to solve simple XOR problem. More generally the Perceptron is only capable of solving linearly separable problems. Even though according to Minsky this criticism wasn't malicious, it in effect stifled the interest in NNs for over a decade.

Interest in NNs was rejuvenated in the early 80's, when it was shown that any previously raised deficiencies could have been solved by usage of multiple units.

This was later exacerbated by invention of back-propagation learning algorithm, which enabled the possibility to gather neurons into groups called layers, which can be stacked into hierarchical structures to form a network. NN of this type were commonly called Multilayer Perceptron (MLP).

In 80s and 90s the interest in NNs plateaued again and general research of AI was more focused on other machine learning techniques. In the realm of classification problems, it was notably SVM and ensemble model. AI research community also developed several other paradigms of NNs that were similarly inspired by Biology of certain aspect of central nervous system but took different approaches. Most important examples were SOM and Recurrent Neural Network (RNN) (e.g., Hopfield networks).

By the year 2000, there were very few research groups that were devoting enough attention to the NNs. There was also certain disdain for NNs in academia and AI research community. Success of NNs that was promised almost half a century ago was finally encountered around 2009, when the first networks with large number of hidden layers were successfully trained. This led to mainstream adaption of umbrella term deep learning which by and large refers to Deep Neural Network (DNN). The word deep indicates that networks have large number of hidden layers.

The key theoretical insight was to learn complicated functions that could represent high-level abstractions (e.g., vision recognition, language understanding etc.).

ANNs in the times before DNNs had only 1 or 2 hidden layers. These are today often called shallow networks. Typical Deep Networks can have number of hidden layers in order of tens, but in some cases even hundreds.

Even though that progress of Neural Network into direction of structures with high number of hidden layers was obvious, its training was unsolved technical problem for very long time. There were basically 3 reasons why this breakthrough didn't come sooner.

- There were no techniques allowing the number of hidden layers to scale.
- There wasn't enough of labeled data necessary to train the ANN.
- The computational hardware wasn't powerful enough to train sufficiently large and complex networks effectively.

First problem was tackled by invention of CNNs [85]. Second problem was solved simply when there was more data available. This was mainly achieved thanks to effort of large companies (Google, Facebook, YouTube, etc.) but also with help of large community of professionals and hobbyists in data sciences.

Both innovation in computational hardware and improvement of training methods were needed to solve the third problem. One of the technical breakthroughs was utilization of Graphics Processing Units (GPUs) for the demanding computation involved in training of a complex network. Thanks to the fact that training process of NNs is typically large number of simple consequent computations, there is a great potential for parallelization.
In the following subsections, we introduce the basis of ANNs. This includes the Perceptron, the feedforward neural networks and the training of them.

5.2.1. Perceptron

How in the introduction mentioned, the ANNs are inspired by biological neural networks. The Perceptron is an abstract model of a single neuron and was introduced by Rosenblatt. In Fig. 5.2b is depicted a Perceptron. The Perceptron has multiple inputs $(x_1, x_2, ..., x_n)$, and only one output y. Furthermore, the Perceptron has a bias input. For every input is derived a linear combination with the weights $(w_1, \ldots, w_n)$. After the linear combination is computed, its value is inserted into the activation function ϕ. This activation function activates the output, if the threshold is fulfilled.
Formally, the Perceptron is defined as follows

Definition 5.1. (**Perceptron**). Let $f_\theta : \mathbb{R}^n \to \{0,1\}$ be a perceptron with an activation function $\phi : \mathbb{R} \to \{0,1\}$ then for all $x \in \mathbb{R}^n$ [39]:

$$f_\theta(x) = \phi\left(b + \sum_{j=1}^{n} w_j x_j \right) \tag{5.1}$$

and the number of parameters $|\theta| = n+1.$

b is called bias, $\theta \in \mathbb{R}^{n+1}$ is a vector of real-valued weights and bias. Moreover, the activation function ϕ can be linear. We consider the Heaviside function:

$$\forall s \in \mathbb{R}, \phi(s) = \begin{cases} 0 & if \quad s < 0 \\ 1 & if \quad s \geq 0 \end{cases}$$

The output of the network thus is either 0 or 1 depending on the input. The network can now be used for a classification task it can decide whether an input pattern belongs to one of two classes. If the total input is positive, the pattern will be assigned to class 1, if the total input is negative, the sample will be assigned to class 0.
With this activation function, the Perceptron has the ability to linear split the hyperspace. This means, that the Perceptron can only learn problems, which are linear separable. For many tasks, this is not sufficient. But with multilayer Perceptrons, it is possible to solve nonlinear problems, which is as next presented.
A crucial question is how to select the parameter θ so that the Perceptron defined by $f_\theta(x)$ can perform a specific task. We consider the supervised setting where we have a dataset $D_{train} = \{(x_i, t_i); i = 1, ..., N_s\}$, with, for all $i, x_i \in \mathbb{R}^n$ and $t_i \in \{0,1\}$. In the original Perceptron algorithm, the parameters are updated iteratively by re-evaluating the predictions at each parameter update, and modifying the parameters that yield incorrect predictions. More formally, let $y_i = \phi(b + w.x_i) \in \{0,1\}$ be the output of the Perceptron model. In order to find the set of parameters θ that best explains the dataset D_{train}, we perform stochastic parameter updates for every training pair (x_i, t_i) in following:

$$\theta \leftarrow \theta - \eta \frac{\partial E(\theta, D)}{\partial \theta} \tag{5.2}$$

The constant η is called the learning rate.
The updates from (5.2) are performed either for a predetermined number of iterations, or when the iteration error is less than a predefined threshold.

5.2.2. Multilayer Perceptron

The Multilayer perceptron or known under the name feedforward neural network [22] is a multilayered network of perceptrons. This means, that several perceptrons are connected in series. Fig 5.3 depicts an example of feedforward neural network. A feedforward neural network consists of multiple layers, which have specific names. The first layer is called the input layer. The last layer is called the output layer. The other layers are the hidden layers. The hidden and output layer consists of several perceptrons, which are called units. The input layer contains only the values for the input. The depth of a network describes the number of layers, and the width of a layer describes the number of units. The depicted example network in Fig. 5.3 has the depth of two. The widths of the first and the second hidden layers are n_1 and n_2, respectively; and the width of the input and output layers are n and n', respectively. The bias units are depicted with the +1, and will be not count to the width of the layer. Between every layer, the units are fully connected (FC). This means that every unit of layer $l+1$ has the output of all units from layer l as input. The number of links and consequently the number of weights, will fast increase, if the width of the layers is increased. If layer l has a units and layer $l-1$ has c units, then the number of parameters $|\theta^{(l)}|$ between this layers are $|\theta^{(l)}| = a.(b+1)$. The +1 comes from the bias input. So, it is not uncommon that modern NNs have millions of parameters, which must be learned. Through this concatenation of several layers, the feedforward neural networks get the capability to solve non-linear problems, if the activation function is not linear. Would be the activation function a linear function, then the network only has the capability to solve linear problems, because the composition of linear function produces again a linear function. So, the activation function should be a non-linear function. In Section 5.3.3 we show some useful activation functions. Furthermore, a feedforward neural network has a very powerful property. With only one hidden layer with sufficient units and the right activation function, like the sigmoid function, the network can approximate arbitrarily closely every continuous function on a closed and bounded subset of $\mathbb{R}^n$. This means that a network with a single layer has the ability to represent any function. But it is not guaranteed, that the training algorithm will find this function. It is possible, that the training algorithm is not able to find the right value for the parameters. Or the training algorithm chooses the wrong function due to overfitting. Furthermore, the single layer may be infeasible large and it exists no heuristic how large the layer should be.

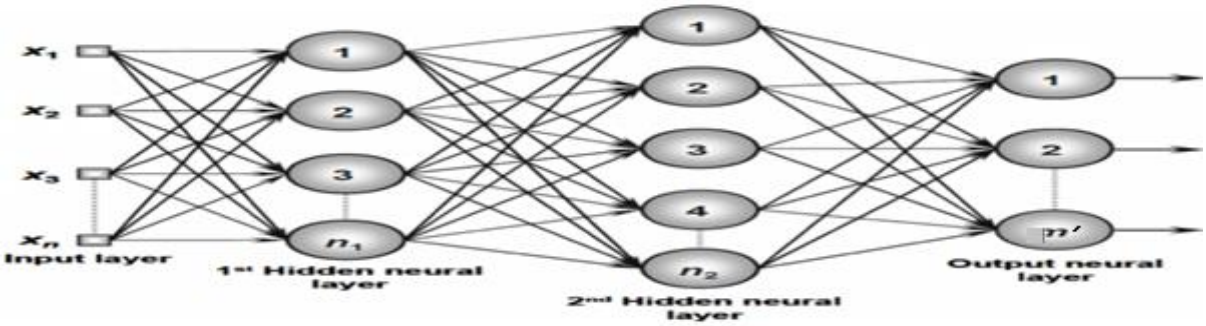


FIGURE 5. 3: A multilayer Perceptron architecture [22].

Formally, the MLP is defined as follows

Definition 5.2. (**Multilayer Perceptron**). Let $f_\theta : \mathbb{R}^n \to \mathbb{R}^{n'}$ be a MLP with m activation functions $\phi_i : \mathbb{R} \to \mathbb{R}$ fixed by layer, and m hidden layers consisting of $(h_k)_{\{1,\dots,m\}} \in \mathbb{N}^m$ neurons each, then for all $x \in \mathbb{R}^n$ [22]:

$$f_\theta(x) = f_\theta(x, 1, m+1) \tag{5.3}$$

With $f_\theta(x, j, k)$ recursively defined as:

$$f_\theta(x, j, 1) = \phi_1\left(b_j^{(1)} + \sum_{i=1}^{n} w_{ij}^{(1)} x_i\right)$$

$$f_\theta(x, j, k) = \phi_k\left(b_j^{(k)} + \sum_{i=1}^{h_{k-1}} w_{ij}^{(k)} f_\theta(x, j, k-1)\right)$$

Where $k \in \{1, ..., m+1\}$ index the layers, $j \in \{1, ..., h_k\}$ index the neurons of the layers and i index the weights of a Perceptron. The number of parameters $|\theta| = (n+1).h_1 + \sum_{i=2}^{m}(h_{i-1}+1).h_i + h_m + 1$

5.3. ANN Training Procedure

ANN is a parametric model. In order to get meaningful results for the task we should find appropriate values for its parameters – weights and biases. Just like in any other ML problem, ANN requires choosing a performance measure and an algorithm which will allow the network to improve this measure with respect to the dataset. It can be achieved by means of mathematical optimization. In supervised training, which is our concern in this study, both the inputs and the outputs are provided. The network processes the inputs and compares its resulting outputs against the desired outputs (forward pass). Errors are then propagated back through the system, causing the system to adjust the weights which control the network (backward pass). This process occurs over and over as the weights are continually tweaked. Neural networks are trained using gradient descent, where the estimate of the error used to update the weights is calculated based on the training dataset. In other words, the goal of training is to find out which weights most directly contributed to the loss (E) of the network. This is the mathematical equivalent of $\partial E / \partial \theta$, where θ are the weights and bias at a particular layer. We perform a backward pass through the network to determine which weights contributed most to the loss and finding ways to adjust them so that the loss decreases. Therefore, training can be separated into four distinct sections, the forward pass, the loss function, the backward pass, and the weight update.

We formalize the problem of training from labeled data for classification problems. We consider the problem of supervised learning with a given training image dataset D. The training process would be an iterative process consists of finding a set of parameters θ, which minimizes the average loss function presented in Eq. (2.2)

In this section we introduce two algorithms essential for ANN learning: gradient descent, which adjusts network parameters in a way that improves performance measure, and back-propagation, which computes the gradient of the cost function necessary for gradient descent.

5.3.1. Gradient Descent Methods

The main objective of training as said previously, is to minimize the cost function, or in other words, to change the network parameters so that the error is minimized. Mathematically, this is achieved by calculating the gradient $\nabla_\theta E(\theta, D)$, equaling it to 0, i.e., $\nabla_\theta E(\theta, D) = 0$ and solving for the parameters.

However, in non-linear algorithms we cannot use that approach because we cannot find a close form solution, this is the case of neural network where instead we use an iterative optimization method.

5.3.1.1. Gradient Descent

Gradient descent [115] is a first-order iterative optimization algorithm for finding a local minimum of a differentiable function. Gradient descent is simply used to find the values of a function's parameters (coefficients) that minimize (approximate) a cost function as far as possible. Gradient descent starts from a random point on the cost function and iteratively moves the points in the negative direction of the gradient of the function until the point reaches local minima.

Iteratively at each epoch, it updates the parameters following the rule

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta E(\theta, D) \tag{5.4}$$

where $\eta \in \mathbb{R}$ is the learning rate, a hyperparameter used to control the magnitude of the update, which influences convergence of the algorithm.

In the case of ANNs GD encounters problems which make the optimization particularly difficult: high number of variables (weights and biases combined) and typically high cardinality of a training set, which makes computing step of the gradient descent computationally infeasible

5.3.1.2. Mini-Batch Stochastic Gradient Descent

Mini-Batch Stochastic Gradient Descent [84] performs gradient update on the random subset of the original dataset, called batch. It addresses the GD issues. Main idea behind SGD is that computing the gradient on random batches gives sufficient approximations of the real gradient. This approach allows much faster computation in exchange for precision in calculations. Because of this the loss function usually fluctuates and does not decrease monotonically. SGD favors exploration over greedy exploitation of the loss function surface because of slightly random suboptimal gradient directions. In the context of SGD one pass through all batches is called an epoch, and processing a single batch is referred to as a step.

Step of the mini-batch SGD:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta E(\theta, D^{(I_t)}) \tag{5.5}$$

where *I* denote a vector of random dataset indexes. Length of this vector is determined by the batch size hyperparameter.

5.3.1.3. Stochastic Gradient Descent with Momentum

SGD has trouble with areas where the surface curves in one dimension much more steeply than in others. However, they are very common around local optima. In these scenarios, SGD oscillates across the slopes of the ravine while making only hesitant progress to local optimum. The momentum [104] is a method that helps accelerate SGD in a relevant direction. It does this by adding fraction of the update vector of the previous step to the current update vector:

$$u_t = \gamma u_{t-1} + \eta \nabla_\theta E(\theta, D^{(I_t)}) \tag{5.6a}$$

$$u_t = \gamma u_{t-1} + \eta \nabla_\theta E(\theta, D^{(I_t)}) \tag{5.6b}$$

where t is the current time step and $\gamma \in \mathbb{R}$ is an hyperparameter, controls the power of the momentum.

The momentum term increases for dimensions the gradients of which point in the same directions and reduces updates for dimensions the gradients of which change directions. As a result, we gain a faster convergence and a reduced oscillation.

5.3.1.4. Adaptive Gradient

The Adaptive gradient (Adagrad) [151] is a stochastic gradient descent algorithm that only changes the learning rate to parameters, thus performing larger updates for infrequent parameters and smaller updates for frequent parameters.

Up to now, we have performed updates generally for all weights, but Adagrad uses different learning rates for different weights. Let us consider we have some weight θ_i and $g_{t,i}$ is the gradient of the error function with respect to the θ_i at time step t:

$$g_{t,i} = \nabla_\theta E(\theta^{(i)}, D^{(I_t)})$$

The SGD update for every parameter θ_i at each time step t then becomes:

$$\theta_{t+1,i} = \theta_{t,i} - \eta g_{t,i}$$

In its update rule, Adagrad modifies the general learning rate η at each time step t for every parameter θ_i, based on the previous gradients:

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_t^{(i,i)}} + \varepsilon} g_{t,i} \tag{5.7}$$

where $G_t^{(i,i)}$ is a diagonal matrix, where each diagonal element (i,i) is the sum of the squares of the gradients with respect to the θ_i up to the time step t and ε is a smoothing term that avoids division by zero.

Adagrad's main benefit is that it eliminates the need to tune the learning rate manually. On the other hand, there is a problem with squared gradients accumulation in the denominator. Since all added terms are positive, the accumulated sum keeps growing during training. The result is that the learning rate is becoming very small during the training and in the end the network cannot learn any additional knowledge.

5.3.1.5. Adaptive Moment Estimation

The adaptive moment estimation (Adam) optimization [28] is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments. Adam is adapting the parameter learning rates based on the average first moment (the mean), Adam also makes use of the average of the second moments of the gradients (the uncentered variance). To estimates the moments, Adam utilizes exponentially moving averages, computed on the gradient evaluated on a current mini-batch.

$$m_t = \beta_1 . m_{t-1} + (1-\beta_1).g_t$$
$$v_t = \beta_2 . v_{t-1} + (1-\beta_2).g_t^2$$

Where m and v are moving averages, g is gradient on current mini-batch, β_1 and β_2 are exponential decay rates for the moment estimates are hyperparameters. 0.9 and 0.999 are standard values, respectively. The vectors of moving averages are initialized with zeros at the first iteration. Additionally, a bias-corrected moment estimates $\hat{m}_t$ and $\hat{v}_t$ are introduced to account for the bias in the gradient towards zero as an effect of the exponential moving average which is evident in the early stages of training.

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

The final update rule for Adam looks like this:

$$\theta_t = \theta_{t-1} - \alpha . \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} \tag{5.8}$$

5.3.2. Backpropagation Algorithm

As aforementioned, in order to optimize the parameters of the network we require calculating the gradient of the loss function $\nabla_\theta E(\theta, D)$, backpropagation (BP) [83] is an algorithm that allows computing gradients in an efficient way. When we use the feedforward network, we provide the initial information to the network through the input x and propagate the information to the hidden layers; finally, we produce a result with the output layer. We call this process the forward propagation. Once we have the prediction of the network, we define a loss function to measure the performance of the model. In order to let the network, know about the errors of the current state, we require performing a backward propagation.

We want to consider a rather general NN consisting of L layers (not counting the input layer). Let us consider an arbitrary layer, say l, which has n_l neurons $x_1^{(l)}, x_2^{(l)}, ..., x_{n_l}^{(l)}$, each with an activation function $\phi^{(l)}$. Notice that the activation function may be different from layer to layer. As in the extended Delta rule, the activation function may be given by any differentiable function, but does not need to be linear. These neurons receive signals from the neurons in the preceding layer, $l-1$. For example, neuron $x_j^{(l)}$ receives a signal from $x_i^{(l-1)}$ with a weight factor $w_{ij}^{(l)}$. Therefore, we have a n_{l-1} by n_l weight matrix, $W^{(l)}$ whose elements are given by $w_{ij}^{(l)}$, for $i = 1, 2, ..., n_{l-1}$ and $j = 1, 2, ..., n_l$ Neuron $x_j^{(l)}$ also has a bias given by $b_j^{(l)}$.

To simplify the notation, we will use $\xi_j^{(l)}$ to denote the input into neuron $x_j^{(l)}$. It is given by

$$\xi_j^{(l)} = \sum_{i=1}^{N_{l-1}} x_j^{(l-1)} w_{ij}^{(l)} + b_j^{(l)}, \quad j = 1, 2, ..., n_l \tag{5.9}$$

Thus, the activation of neuron $x_j^{(l)}$ is

$$x_j^{(l)} = \phi^{(l)}\left(\xi_j^{(l)}\right) = \phi^{(l)}\left(\sum_{i=1}^{N_{l-1}} x_j^{(l-1)} w_{ij}^{(l)} + b_j^{(l)}\right) \tag{5.10}$$

We can consider the zeroth layer as the input layer. If an input vector x has n components, then $n_0 = n$ and neurons in the input layer have activations $x_i^{(0)} = x_i, i = 1, 2, ..., n_0$. Assuming that the output vector y has Nc components, then we must have $n_L = Nc$. These components are given by $y_j = x_j^{(L)}, j = 1, 2..., Nc$. We will now consider training a rather general multilayer Perceptron for pattern association using the BP. Just like in the Delta rule, the training vectors are presented one at a time to the network during training algorithm.

To define the learning process formally and in more detail, let us consider N_s training patterns labeled $(\vec{x}^p, \vec{t}^p)$, where $\vec{x}^p$ is the input vector, $\vec{t}^p$ is the desired output vector, and $1 \le p \le N_s$. Given the current configuration of the network, the input $\vec{x}^p$ yields the output $\vec{y}^p$. Then for every pattern p we want $\vec{y}^p$ to be as close to the desired output $\vec{t}^p$ as possible. We can define the loss function for pattern p as:

$$E_p = l(y_p, t_p) \tag{5.11}$$

It may be useful to sum up the average error for all input patterns to assess the network performance on the whole dataset, which can be achieved simply by computing the mean squared error

$$E = \frac{1}{N_s} \sum_{p=1}^{N_s} E_p \tag{5.12}$$

When learning, for each interconnected pair of neurons (i, j), where i is a neuron in layer l, j is a neuron in layer $l+1$ and w_{ij} weights their connection, we want to adjust the weights w_{ij} and biases b_j to minimize the mean squared error E. Provided the activation function is differentiable everywhere on its domain, E is also differentiable. When adjusting the weight w_{ij} of the neuron j located in the output layer, we are therefore interested in the partial derivative:

$$\frac{\partial E}{\partial w_{ij}} = \frac{1}{N_s} \frac{\partial}{\partial w_{ij}} \sum_{p=1}^{N_s} E_p = \frac{1}{N_s} \sum_{p=1}^{N_s} \frac{\partial E_p}{\partial w_{ij}} \tag{5.13}$$

To be able to adjust the network after each presented input pattern, we are actually interested in the derivative for each given pattern p and its corresponding E_p. In the following equations we will therefore omit the pattern index p. For layer L, we apply the chain rule to Eq. (5.13) we get:

$$\frac{\partial E}{\partial w_{ij}^{(L)}} = \sum_{k=1}^{N_L} \frac{\partial E}{\partial \xi_k^{(L)}} \frac{\partial \xi_k^{(L)}}{\partial w_{ij}^{(L)}} \tag{5.14}$$

We now define the sensitivity vector for a general layer l to have components

$$\delta_k^{(l)} = \frac{\partial E}{\partial \xi_k^{(l)}}, \quad \text{k= 1,2,...,N}_l$$

This is called the sensitivity of neuron $x_k^{(l)}$ because it gives the change in the output error, E, per unit change in the net input it receives. For layer L it is easy to compute the sensitivity vector directly using the chain rule to obtain

$$\delta_k^{(L)} = \left(y_k - t_k\right)\phi'^{(L)}\left(\xi_k^{(L)}\right)$$

where ϕ' denotes the derivative of the activation function ϕ. We also know that

$$\frac{\partial \xi_k^{(L)}}{\partial w_{ij}^{(L)}} = \frac{\partial}{\partial w_{ij}^{(L)}}\left(\sum_{m=1}^{N_{L-1}} x_m^{(L-1)} w_{mk}^{(L)} + b_k^{(l)}\right) = s_{nj} x_i^{(L-1)}$$

Similary, $\frac{\partial E}{\partial b_j^{(L)}} = \sum_{k=1}^{N_L} \frac{\partial E}{\partial \xi_k^{(L)}} \frac{\partial \xi_k^{(L)}}{\partial b_j^{(L)}}$ and since $\frac{\partial \xi_k^{(L)}}{\partial b_j^{(L)}} = s_{nj}$

For a general layer, l, we can write

$$\frac{\partial E}{\partial w_{ij}^{(l)}} = \sum_{k=1}^{N_l} \frac{\partial E}{\partial \xi_k^{(l)}} \frac{\partial \xi_k^{(l)}}{\partial w_{ij}^{(l)}} = \sum_{k=1}^{N_l} \delta_k^{(l)} \frac{\partial \xi_k^{(l)}}{\partial w_{ij}^{(l)}}$$

$$\frac{\partial E}{\partial b_j^{(l)}} = \sum_{k=1}^{N_l} \frac{\partial E}{\partial \xi_k^{(l)}} \frac{\partial \xi_k^{(l)}}{\partial b_j^{(l)}} = \sum_{k=1}^{N_l} \delta_k^{(l)} \frac{\partial \xi_k^{(l)}}{\partial b_j^{(l)}}$$

Since

$$\xi_k^{(l)} = \sum_{m=1}^{N_{l-1}} x_m^{(l-1)} w_{mk}^{(l)} + b_k^{(l)}$$

We have

$$\frac{\partial \xi_k^{(l)}}{\partial w_{ij}^{(l)}} = s_{nj} a_i^{(l-1)} \text{ and } \frac{\partial \xi_k^{(l)}}{\partial b_j^{(l)}} = s_{nj}$$

and so $\frac{\partial E}{\partial w_{ij}^{(l)}} = x_i^{(l-1)} \delta_j^{(l)}$, $\frac{\partial E}{\partial b_j^{(l)}} = \delta_j^{(l)}$ (5.15)

Using Eq. (2.15) we can compute the gradient of the error function for each of the given patterns p.

The weight and bias adjustment are then computed by:

$$\Delta w_{ij}^{(l)} = -\eta \frac{\partial E}{\partial w_{ij}^{(l)}} = -\eta x_i^{(l-1)} \delta_j^{(l)}$$

$$\Delta b_j^{(l)} = -\eta \frac{\partial E}{\partial b_{ij}^{(l)}} = -\eta \delta_j^{(l)}$$

In order to use updating rules for the weights and biases, we need to be able to compute the sensitivity vectors $\delta^{(l)}$ for $l = 1, 2, ..., L-1$. From their definition $\delta_j^{(l)} = \frac{\partial E}{\partial \xi_j^{(l)}}$, j= 1,2,...,$N_l$, we need to know how E depends on $\xi_j^{(l)}$. The key to computing these partial derivatives is to note that $\xi_j^{(l)}$ in turn depends on $\xi_j^{(l-1)}$ for i= 1,2,...,N_{l-1}, because the net input for layer l depends on the activation of the previous layer, $l-1$, which in turn depends on the network input for layer $l-1$. Specifically

$\xi_j^{(l)} = \sum_{i=1}^{N_{l-1}} x_i^{(l-1)} w_{ij}^{(l)} + b_j^{(l)} = \sum_{i=1}^{N_{l-1}} \phi^{(l-1)}\left(\xi_i^{(l-1)}\right) w_{ij}^{(l)} + b_j^{(l)}$ for $j = 1, 2, ..., N_l$. Therefore, we have for the sensitivity of layer $l-1$

$$\begin{aligned}\delta_j^{(l-1)} &= \frac{\partial E}{\partial \xi_j^{(l-1)}} = \sum_{i=1}^{N_l} \frac{\partial E}{\partial \xi_i^{(l)}} \frac{\partial \xi_i^{(l)}}{\xi_i^{(l-1)}} \\ &= \sum_{i=1}^{N_l} \delta_i^{(l)} \frac{\partial}{\xi_i^{(l-1)}} \left(\sum_{m=1}^{N_{l-1}} \phi^{(l-1)}\left(\xi_m^{(l-1)}\right) w_{mi}^{(l)} + b_i^{(l)} \right) \\ &= \sum_{i=1}^{N_l} \delta_i^{(l)} \phi'^{(l-1)}\left(\xi_j^{(l-1)}\right) w_{ji}^{(l)} = \phi'^{(l-1)}\left(\xi_j^{(l-1)}\right) \sum_{i=1}^{N_l} w_{ji}^{(l)} \delta_i^{(l)}\end{aligned}$$

Thus, the sensitivity of a neuron in layer $l-1$ depends on the sensitivities of all the neurons in layer l. This is a recursion relation for the sensitivities of the network since the sensitivities of the last layer L is known. To find the activations or the net inputs for any given layer, we need to feed the input from the left of the network and proceed forward to the layer in question. However, to find the sensitivities for any given layer, we need to start from the last layer and use the recursion relation going backward to the given layer. This is why the training algorithm is called backpropagation.

Finally, we can summarize the weight and the bias adjustment applicable to all the layers in a given network

$$\Delta w_{ij}^{(l)} = -\eta \frac{\partial E}{\partial w_{ij}^{(l)}} = -\eta x_i^{(l-1)} \delta_j^{(l)}$$

$$\Delta b_j^{(l)} = -\eta \frac{\partial E}{\partial b_{ij}^{(l)}} = -\eta \delta_j^{(l)}$$

where

$\delta_k^{(L)} = \left(y_k - t_k\right) \phi'^{(L)}\left(\xi_k^{(L)}\right)$, for $k = 1, 2, ..., N_L$

$\delta_j^{(l)} = \phi'^{(l)}\left(\xi_j^{(l)}\right) \sum_{i=1}^{N_{l+1}} w_{ji}^{(l)} \delta_i^{(l+1)}$, for $l = L-1, ..., 2, 1$ and $j = 1, 2, ..., N_l$

5.3.3. The Activation Function

As touch upon in the previous Section, 5.2.1, $\phi(s)$ are activation functions. An activation function in ANN terminology is a function that defines how the output of a certain node will look like. However, the real purpose of these functions is to establish non-linear relationships between the input and output [7]. This is very important for neural networks if they are needed to perform non-linear mappings, is a very powerful property of ANNs. A further property for the activation function is that it must be continuously differentiable. Otherwise, the backpropagation algorithm will not work, because it cannot compute the gradient.

There are many activation functions such as sigmoid function, hyperbolic tangent function, rectified linear unit function (ReLU), etc. In theory, any activation function can be used to train an ANN. There is no mathematical or logical rule that justifies the preference of one over the others. However, empirical evidence shows that in some

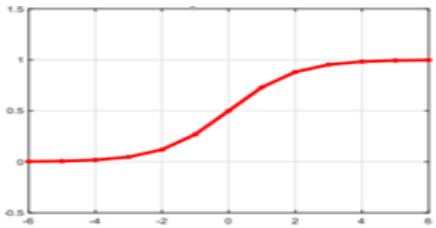
applications, specific activation functions provide better accuracy and faster "learning" than the others [7]. Fig. 5.4 depicts some common activation functions used in practice.

The curve of sigmoid function as shown in Fig. 5.4a, is a common non-linear activation function. The output of this function is bounded, and it was widely used. Although the characteristic of the sigmoid function is consistent with the synapses of neurons in neurology, and the derivative of this function is convenient to get, the function is rarely used nowadays due to its shortcomings. From the curve of sigmoid function, this function has the characteristic of soft saturability. That is, the slope of the graph tends to be zero when the input is very large or very small. When the slope of the function is close to zero, the gradient that passed to the underlying network becomes very small, which will make network parameters difficult to be trained effectively. Meanwhile, the direction of weight update only to one direction due to the output of this function is always positive, which will affect the convergence rate. The formula of sigmoid function is defined as :

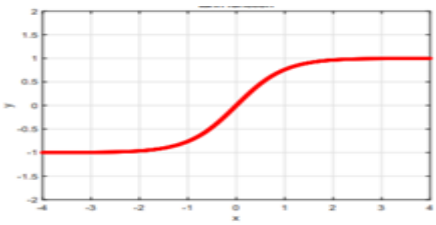
$$\phi(s) = \frac{1}{1+e^{-s}}$$

The tanh function is the updated version of the sigmoid function on the range, which is a symmetric function centered on zero. Its output is bounded, and it brings nonlinearity to the neural network. The curve of this function can be seen in Fig 5.4b. The convergence rate is higher than the sigmoid function, but the problem of gradient diffusion also exists. The formula of tanh function is defined as:

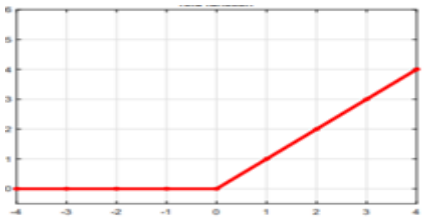
$$\phi(s) = \frac{1-e^{-2s}}{1+e^{-2s}}$$



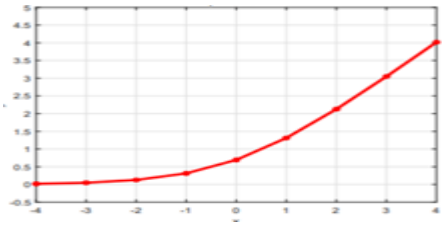
(a) The curve of sigmoid function



(b) The curve of tanh function



(c) The curve of ReLu function



(d) The curve of softplus function

FIGURE 5. 4: The graphs of four common different activation functions. The names of the above curve functions are: the sigmoid function, the tanh function, the ReLu function and the softplus function.

The trendy activation function in neural network is the ReLu function, which is a piecewise function. The curve of this function is shown in Fig. 5.4c. From the curve of this function, this function will force the output to be zero if the input value is less than or equal to zero. Otherwise, it will make the output value equal to the input value. The method of directly forcing some data to be zero can create a moderate sparse characteristic to some extent. Compared with the previous two functions, the ReLu function provides a much faster computing rate. Since ReLu is unsaturated, there is no gradient diffusion problem, unlike sigmoid and tanh functions. Although the ReLu function has great advantages, it also has some shortcomings. Since the derivative of the ReLu function is always zero when the input value is negative, it is likely to present neuronal necrosis when a neuron with a large gradient pass through the ReLu function, which will affect the final recognition result. The equation of this function is defined as:

$$\phi(s) = \begin{cases} s & if \quad s \geq 0 \\ 0 & if \quad s < 0 \end{cases}$$

The softplus function, as shown in Fig. 5.4d, is similar to ReLu function. From the curve of this function, the difference between the softplus function and the ReLu function can be seen clearly. This function has small reservations about values less than 0, which will decrease the possibility of neuronal death. But this function has much more computation than ReLu function. The formula of this function is defined as:

$$\phi(s) = \ln(1 + e^{s})$$

5.3.4. Vanishing and Exploding Gradients

A common problem occurring while training a neural network is the vanishing gradient. As the name suggests it, the gradient becomes so small that the evolution of the parameters stagnates [120]. This effect worsens as the depth of the network (i.e., the number of layers) increases. For instance, the derivative of the sigmoid function is

$$\phi'(x) = \phi(x).(1 - \phi(x))$$

and is bounded between 0 and 1/4. If the weights are assumed to have a mean of 0 and a unit variance, they are then bounded between -1 and 1. Thus, for a fully-connected network with L layers (assuming that the input and weights $w_i \in \Re$ and that there is no bias for simplicity):

$$a_1 = w_1 x$$

$$a_i = w_i z_{i-1}$$

$$z_i = \phi(a_i)$$

$$\hat{y} = z_{L-1} = \phi(a_{L-1})$$

$$\frac{d\hat{y}}{\partial w_j} = \prod_{i=j+1}^{L-1} \frac{\partial z_i}{\partial a_i} \frac{\partial a_i}{\partial z_{i-1}} \times \frac{\partial z_j}{\partial a_j} \frac{\partial a_j}{\partial w_j}$$

$$= \prod_{i=j+1}^{L-1} \frac{\partial \phi(a_i)}{\partial a_i} w_i \times \frac{\partial \phi(a_j)}{\partial a_j} \frac{\partial a_j}{\partial w_j}$$

As it can be seen in the above Equation, $\frac{\partial \hat{y}}{\partial w_j}$ can quickly vanish to 0 as the number of layers grows. One solution to the problem is to use the ReLu activation instead of the sigmoid. Indeed, the derivative of the ReLu activation is 1 when the input is positive and 0 otherwise.

The exploding gradient problem refers to the case where the network weights become increasingly large and consequently make the gradient also very large [106]. The exploding gradient problem can be solved for instance by employing a regularization function that penalizes the network training for assigning large weights. Problems related to exploding and vanishing gradients are typically encountered when dealing with very deep network architectures [111].

5.4. Deep Convolutional Neural Networks

Convolutional neural networks are used successfully in a large number of applications. The handwriting recognition task was one of the first applications of image analysis by convolutional neural networks. In addition to providing good results on object detection and image classification tasks, they also perform well when applied to facial recognition, video analysis, or even text recognition. A CNN is a type of feed-forward artificial neural network where the individual neurons are tiled in such a way that they respond to overlapping regions in the visual field. CNN is a biologically-inspired variant. From the early work of Hubel and Wiesel [60] on the cat visual cortex in 1968, it is known that the visual cortex contains a complex arrangement of neurons. These neurons are sensitive to small sub-regions of the visual field, called a receptive field. The sub-regions are tiled to cover the entire visual field. These neurons work as local filters over the input space and are well-suited to exploit spatially local correlation presented in natural images.

Fukushima's work [40] first computes models based on this local connectivity between neurons and on hierarchically organized transformations of the image. In this work, he found that when neurons with the same parameters are applied on patches of the previous layer at different locations, a form of translational invariance is achieved. According to this idea, LeCun designed and trained CNNs using the error gradient with back-propagation algorithm, obtaining state-of-the art performance on lots of pattern recognition tasks. One of the main contributions of CNN on neural network area is the implementation of weights sharing, applying same weights on same feature map, which increases the learning efficiency by significantly reducing the number of trainable parameters. Since the early 2000s, Convolutional Neural Networks or CNNs have been applied successfully to the detection and recognition of objects and regions in images. Despite this success, convolutional networks and neural networks more generally were neglected by the research community between 1997 and 2012 because of the difficulty in making it work with the computers of that time, because they were not powerful enough. In 2011 and 2012 three events suddenly changed the situation. First, Graphical Processing Units (GPUs) capable of over a trillion operations per second became available for less. These powerful specialized processors, initially designed for the graphics rendering of video games, have proven to be very efficient for calculations of neural networks. Second, experiments conducted simultaneously at Microsoft, Google and IBM with the help of Geoff Hinton's lab has shown that deep networks can halve the error rates of speech recognition systems. Third, several records in image recognition have been broken by convolutional neural networks. The highlight was the resounding victory for Team Toronto in the "ImageNet" Object Reconnaissance Competition. The decrease in error rates was such a revolution. Overnight, the majority of speech and vision research teams abandoned their preferred methods and switched to convolutional neural networks and other neural networks. The increase in computing power in recent years, and particularly on Graphics Processing Units (GPUs), has made it possible to create CNNs capable of processing images of larger dimensions, such as the AlexNet network [74]. This network uses architecture similar to that of the LeNet-5 network but with many more parameters. AlexNet has been responsible for the renewed interest in CNNs in the image processing community since winning the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) image classification competition in 2012 [74], with a significant gain in classification performance compared to competing methods of the time. Following this victory, many CNN architectures were developed with a view to solving problems associated with image analysis, such as the classification of images which are of particular interest to us in this chapter.

CNNs are primarily used for 2D image recognition, so we will illustrate their architecture on a 2D rectangular image consisting of pixels. Each pixel generally carries color information. Color can be represented by multiple channels (e.g., 3 RGB channels). For the sake of simplicity, we will consider only one single channel (shades of gray) while explaining the model. The neurons in CNNs work by considering a small portion of the image, let us call it sub image. The sub images are then inspected for features that can be recognized by the network. As a simple example, a feature may be a vertical line, an arch, or a circle. These features are then captured by the respective feature maps of the network. A combination of features is then used to classify the image. Furthermore, multiple different feature maps are used to make the network robust to varying levels of contrast, brightness, color saturation levels, noise, etc. There are two types of layers, both consisting of feature maps. The purpose of the so-called convolutional layer is to recognize the features in the input image. It usually consists of several feature maps, each map recognizing certain feature. The sub images intentionally cover overlapping regions of the original image.

Such a design is important in order to tolerate image distortions, like translation, rotation, skewing, etc. The other layer type, the so-called subsampling layer, always follows after a convolutional layer. It consists of the same number of feature maps, where each map from the convolutional layer is used as an input for the corresponding feature map in the following subsampling layer. Sub images covered by the neurons from the subsampling layer usually do not overlap. Depending on the network's depth, the convolutional and subsampling layers alternate until the last subsampling layer is reached. After the last subsampling layer, there may be any number of fully connected layers as described in Section 5.2.2, the last one of them is the output layer

5.4.1. Components of a DCNN

To explain the architecture of the convolutional network, let us consider the image data in the input layer. From a high-level point of view, a convolutional network is architecturally split into two important parts. Each part is designed to fulfill a different purpose. The first part of the network performs feature extraction, and is built of convolutional and subsampling layers. The second part performs classification based on the extracted features. We will be using fully-connected MLP for this part. Let us describe the feature extraction process in more detail. Starting with the presented input image, each pixel represents the input for the neurons grouped in feature maps of the first convolutional layer. The neurons in the feature map are organized in a 2-dimensional grid. All the neurons within the same feature map share their weights. This allows optimizing the implementation by requiring less memory and yielding a better performance. More importantly, it is an architectural feature. Each neuron in a given feature map is expected he recognize the same feature. The feature is recognized by a combination of weights, which are essentially filtering neural inputs. Sharing the same weights for all the neurons within the given feature map ensures the same filter is used for each pixel. Let F_l be the set of feature maps in the layer l. All feature maps in Fl have the same size, let us denote it $m_l \times n_l$. Further, $\forall 0 \le i < m_l, 0 \le j < n_l$ let $x_{i,j}^{\lambda,l}$ be the output of the neuron (i, j, λ, l), at the position (i, j) in the feature map λ of the layer l. We will consider the input image as the output of the layer with zeroth index, having the size of $m_0 \times n_0$ pixels. Note that the input layer can be also considered as a subsampling layer with one single feature map. The input layer is then followed by alternating convolutional and subsampling layers. The neurons of the feature map λ in the convolutional layer *l* take the input from the set of feature maps $F'_{\lambda,l}$, located in the previous subsampling layer $(\varnothing \neq F'_{\lambda,l} \subset F_{l-1})$.
The convolutional layer is then followed by a subsampling layer, which reduces the size of feature maps. The last subsampling layer ends the feature extraction process.
At this point the second part of the network begins the classification process. The classification is performed using a fully-connected MLP network, which classifies the inputs according to the extracted features.
In the following subsections, we will provide an explanation of the main layers in CNNs, such as, input layer, convolutional layer, pooling layer and classification layer

5.4.2. Input Layer

The first layer takes raw pixels of the observed image as inputs, where each neuron of this layer corresponds to one entry of pixel values. Normally, an image is described as a structure of $W \times H \times C$, where W, H are the width and height of the image, and C denotes the number of channels, for instance an RGB image has 3 channels. Neurons in the input layer perform no mathematical operation but simply provide the input value as output to be processed in the next layer.

5.4.3. Convolutional Layer

The purpose of convolutional layers is to detect the features in the presented images. It consists of multiple feature maps, each recognizing certain specific feature. The feature recognition can be thought of as running the sub image through a filter. The filtering is essentially done through weight adjustments. A filter may highlight or suppress a pixel's potential based on the surrounding pixels. The filtering to use is determined automatically by the learning

process via weight adjustments. The number of feature maps in the convolutional layer is an important architectural decision to make when designing the convolutional network. The optimal value depends on the nature of the problem being solved. The best way to determine the number of feature maps is to experiment. As a rule of thumb, more complex images are recognized better using more features, while simple tasks produce better results using just a few features. In fact, if the chosen number of features is too big, the features learned will often be duplicated. Let us consider a convolutional layer l with a set of feature maps F_l. In general, this layer is always preceded by a subsampling layer with a set of feature maps F_{l-1}. For the sake of simplicity, we will refer to a feature map in the convolutional layer as a convolutional map. A feature map in the subsampling layer will be referred to as a subsampling map.

All feature maps in the same layer have the same size. The size of convolutional maps is predetermined by the size of feature maps in the preceding layer, by the layer parameter r_c^l, and by the overlapping parameter s_c^l. Each neuron in a convolutional map takes its input from $\left(r_c^l\right)^2$ neurons in the corresponding feature map from the preceding layer. These input neurons form a square of dimension $r_c^l \times r_c^l$. The overlapping parameter $s_c^l \geq 1$ determines how many neurons apart are these squares from each other. In the following text we will refer to this square of inputs as the receptive field of the neuron.

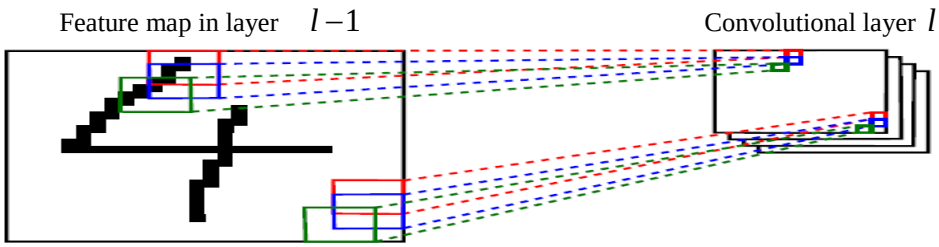


FIGURE 5. 5: Illustration of the convolution operation on the input feature map of size 28×28 neurons in layer $l-1$. The convolution uses a receptive field of size 5×5 and overlapping parameter of 2 neurons, thus producing feature maps of size 12×12 neurons in layer *l*. Note that neighboring neurons in convolutional layers have overlapping receptive fields.

As illustrated in Fig. 5.5, neighboring neurons in a convolutional map have neighboring receptive fields that overlap in exactly $\left(r_c^l - s_c^l\right)^2$ neurons. If the dimensions of the feature map are $\left(m_{l-1}, n_{l-1}\right)$, the following convolutional maps must have the dimensions of:

$$\left(m_l, n_l\right) = \left(\left\lceil \frac{m_{l-1} - r_c^l}{s_c^l} + 1 \right\rceil, \left\lceil \frac{n_{l-1} - r_c^l}{s_c^l} + 1 \right\rceil\right)$$

Each neuron $(i,\ j, \lambda,\ l)$, is thus connected to all the neurons $\left(is_c^l + \Delta i,\ js_c^l + \Delta j, \lambda', l-1\right)$ from the preceding layer, where $\lambda' \in F_{\lambda,l}$ and $0 \leq \Delta i, \Delta j < r_c^l$. Each connection within the receptive field of neuron $(i,\ j, \lambda,\ l)$, has the weight $w_{\Delta i, \Delta j}^{\lambda, \lambda', l}$. This weight is shared for all the neurons within the same feature map, i.e., for all $i,\ j$. At this point, we can compute the number of weights required by convolutional layer l. No matter what the dimensions of convolutional maps are, the total number of weights is:

$$\left|W_l\right| = \left|F_l\right| . \left(r_c^l\right)^2 \tag{5.16}$$

Each of the output of convolutional maps $x_{i,j}^{\lambda,l}$ is obtained by convoluted each $x_{i,j}^{\lambda',l-1}$ with the corresponding kernel $w_{\Delta i,\Delta j}^{\lambda,\lambda',l}$, and the results are added with the bias $b^{\lambda,l}$. Finally, a nonlinear function $\phi(.)$ is applied.

Mathematically, the output of convolutional layer can be computed as

$$x_{i,j}^{\lambda,l} = \phi\left(\sum_{\lambda' \in F_{\lambda,l-1}} \sum_{\Delta i=0}^{r_c^l-1} \sum_{\Delta j=0}^{r_c^l-1} x_{is_c^l+\Delta i,\, js_c^l+\Delta j}^{\lambda',l-1} w_{\Delta i,\Delta j}^{\lambda,\lambda',l} + b^{\lambda,l} \right) \tag{5.17}$$

The algorithm processing the convolutional layer l must sum the weights for each neuron. There are $|F_l|.m_l.n_l$ neurons in layer l. Each of them must sum up $|F_{l-1}|.\left(r_c^l\right)^2$ weights (Eq. 5.16). This yields the following time complexity for processing the convolution in layer l.

Given the above definitions, there are three important hyperparameters that need to be chosen to configure the convolutional layer.

- $|F_l|$: the number of convolutional maps
- r_c^l: the receptive field size or filter size parameter, which determines the size of the window which is convolved with the input feature map
- s_c^l: The stride of a filter which is responsible for sliding the filter over the image

As we already discussed, an adequate number of convolutional maps depends on the complexity of features in the classified input. The optimal value of the receptive field parameter also depends on the given problem. Its values are often small, so that the network can detect simple and small local features. The value of overlapping parameter is usually equal to 1. Increasing the parameter rarely makes sense, because if the receptive fields are too far apart, the network can miss important features. It might make sense to use higher overlapping parameter as an optimization technique for big images and big receptive fields. It is however disputable whether resampling the input images to a lower resolution would be a better approach.

5.4.4. Pooling Layer

The features extracted from the convolutional layers are very high for the classification layer, which increases the risk of overfitting and computation time. The pooling layer solves this problem by gradually reducing the size of the feature maps by keeping only the most important information and therefore a reduction in the number of parameters, thus stemming overfitting. In addition, this layer induces a form of invariance by translating the data, making the network more robust to the detection of particular patterns in the inputs.

The pooling map in layer l is connected only to the corresponding feature map in layer $l-1$. The 2D pooling operation is characterized by a pooling window of size $r_x^l \times r_x^l$ (hyper-parameter defined by the user) which moves horizontally and vertically with a stride of s_x^l on each feature maps produced in layer $l-1$. Formally, the new feature map sizes will be:

$$m_l = \left\lceil \frac{m_{l-1} - r_x^l}{s_x^l} + 1 \right\rceil \text{ and } n_l = \left\lceil \frac{n_{l-1} - r_x^l}{s_x^l} + 1 \right\rceil$$

Where m_{l-1} and n_{l-1} are the width and the height of the feature map to be processed.

By this definition, each neuron from the layer $l-1$ is connected to exactly one neuron from the pooling in layer l (and each feature map is connected to exactly one subsampling map). This results into a simple network design with the following time complexity when computing the output of the pooling layer in the forward pass.

$$O\left(\left|F_{l-1}\right|.m_{l-1}.n_{l-1}\right)$$

The neurons from the subsampling maps take their input from their receptive fields. Multiple inputs ($r_x^l \times r_x^l$) are then combined into a single value. The most common approaches to combine these inputs is through averaging Eq. (5.18a), or finding the maximum value Eq. (5.18b).

- Max-pooling: consists of returning the maximum local value at the level of each pooling window
- Avg-pooling: calculates the average of the local values of each pooling window. Having the feature map λ in layer l, its output is then multiplied by a trainable coefficient $a^{\lambda,l}$, added to trainable bias $b^{\lambda,l}$, and passed to an activation function ϕ (Eq. 5.18c).

$$\xi_{i,j}^{\lambda,l} = \frac{1}{r_x^l r_x^l} \sum_{\Delta i=0}^{r_x^l-1} \sum_{\Delta j=0}^{r_x^l-1} x_{ir_x^l+\Delta i,\, jr_x^l+\Delta j}^{\lambda,l-1} \tag{5.18a}$$

$$\xi_{i,j}^{\lambda,l} = \max_{\substack{\Delta i\in(0,r_x^l-1)\\ \Delta j\in(0,r_x^l-1)}} \left(x_{ir_x^l+\Delta i,\, jr_x^l+\Delta j}^{\lambda,l-1} \right) \tag{5.18b}$$

$$x_{i,j}^{\lambda,l} = \phi\left(a^{\lambda,l}.\xi_{i,j}^{\lambda,l} + b^{\lambda,l}\right) \tag{5.18c}$$

5.4.5. Classification Layers

Once the convolutional and pooling stages have operated, the patterns obtained at output are injected as input data into a multi-layer Perceptron network.

The goal of this one is to classify the data by deducing a probability on the different possible results. To inject the patterns from the convolutional network, we go through a flattening step. This operation consists in putting all the data in a single vector. This vector will allow the creation of a first layer of fully connected neurons, that is to say each of the values of this vector will be connected to the neurons of the first layer of the network, allowing the classification of the image. Classification layers generally contain one or more fully connected layers at the end of deep neural networks depending on the level of feature abstraction desired. In these layers, the neurons are connected with each of the neurons of the volume corresponding to its input, i.e., with each of the neurons of the previous layer. This layer outputs a vector of Nc dimensions, where Nc represents the number of classes that the network will be able to predict. This vector contains the classification probabilities for each class of an image. The fully-connected (*FC*) layer determines the link between the position of features in the image and a class. Indeed, the entry being the result of the previous layer, it corresponds to a feature map: the high values indicate the location of this feature in the image. The final *FC* layer is forwarding to a softmax function, which determines the class probabilities given the input images. This output layer has one neuron corresponding to each possible class. Softmax function is one of the probabilistic-based classification models which measure the correlation between an output value and a reference value by a probability score. For the given input, the probability that the input belongs to class c is computed as follows:

$$soft\max(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{Nc} e^{z_k}}; j = 1,..., Nc$$

Where $z = [z_1, z_2, ..., z_{Nc}]$ is the input vector to the softmax function, the output of $soft\max(z)_j$ in a range between 0 and 1 and $\sum_{j=1}^{Nc} soft\max(z)_j = 1$.

5.5. Deep CNN Training

Training the CNN corresponds to optimize the learnable parameters in convolutional layers and fully-connected layers so that the trained model fits the training data. Because the output of CNN produces the probabilistic

distribution of the classes via SoftMax function, the input features that belong to a particular class most likely has the highest probability by feed-forward propagating the well-trained model. A standard optimization method of the CNN is to minimize the losses of the loss function. Convolutional networks are a kind of feed-forward networks. Hence it is possible to train them using the backpropagation algorithm. We will use the ideas and some definitions introduced in Section 5.2. The general objective will be the same minimize the network error rate E through gradient descend technique. The weight adjustment will again start from the last layer. Using supervised learning, the expected neuron output is known here. We will be considering convolutional networks that have a conventional Multilayer Perceptron (MLP) as their last layer. The backpropagation process of MLP is defined in Section 5.2.2, and will be the same for this part of the convolutional network. The details how to cope with different types of the network's lower layers will be discussed in the following subsections.

5.5.1. Backpropagation in Pooling Layers

The subsampling layer does not have any trainable weight parameters. In fact, its only trainable parameters are the the coefficient $a^{\lambda,l}$ and bias $b^{\lambda,l}$. As we are interested in the influence of these parameters on the errors, we need to consider $\partial E / \partial a^{\lambda,l}$ and $\partial E / \partial b^{\lambda,l}$ in the subsampling layer l. Using Eq. (5.16c) and applying the chain rule yields the following:

$$\frac{\partial E}{\partial a^{\lambda,l}} = \sum_{i,j} \frac{\partial E}{\partial x_{i,j}^{\lambda,l}} \cdot \frac{\partial x_{i,j}^{\lambda,l}}{\partial a^{\lambda,l}} = \sum_{i,j} \frac{\partial E}{\partial x_{i,j}^{\lambda,l}} . \phi'(a^{\lambda,l} . \xi_{i,j}^{\lambda,l} + b^{\lambda,l}) . \xi_{i,j}^{\lambda,l} \tag{5.19}$$

$$\frac{\partial E}{\partial b^{\lambda,l}} = \sum_{i,j} \frac{\partial E}{\partial x_{i,j}^{\lambda,l}} \cdot \frac{\partial x_{i,j}^{\lambda,l}}{\partial b^{\lambda,l}} = \sum_{i,j} \frac{\partial E}{\partial x_{i,j}^{\lambda,l}} . \phi'(a^{\lambda,l} . \xi_{i,j}^{\lambda,l} + b^{\lambda,l}) \tag{5.20}$$

The partial derivative $\partial E \big/ \partial x_{i,j}^{\lambda,l}$ in the above equations is still unknown. It expresses how the output of the layer $l-1$ influences the error. Let λ denote the respective feature map in the layer l, and λ' the corresponding feature map producing the input for λ. Note that λ' can only represent one single feature map, because λ is a subsampling map. The chain rule can be leveraged here again:

$$\frac{\partial E}{\partial x_{i',j'}^{\lambda',l-1}} = \sum_{i,j} \frac{\partial E}{\partial x_{i,j}^{\lambda,l}} \cdot \frac{\partial x_{i,j}^{\lambda,l}}{\partial \xi_{i,j}^{\lambda,l}} \cdot \frac{\partial \xi_{i,j}^{\lambda,l}}{\partial x_{i',j'}^{\lambda',l-1}} \tag{5.21}$$

The derivative $\partial E \big/ \partial x_{i,j}^{\lambda,l}$ is known by assumption. The expression $\partial x_{i,j}^{\lambda,l} \big/ \partial \xi_{i,j}^{\lambda,l}$ can be derived from Eq. (5.18c):

$$\frac{\partial x_{i,j}^{\lambda,l}}{\partial \xi_{i,j}^{\lambda,l}} = \phi'(a^{\lambda,l} . \xi_{i,j}^{\lambda,l} + b^{\lambda,l}) . a^{\lambda,l} \tag{5.22}$$

The evaluation of the last term from Eq. (5.20) depends on the subsampling type. As mentioned before, we will be considering the average or maximum value.
Averaging yields:

$$\frac{\partial \xi_{i,j}^{\lambda,l}}{\partial x_{i',j'}^{\lambda',l-1}} = \frac{1}{r_x^l . r_x^l}$$

Maximization yields:

$$\frac{\partial \xi_{i,j}^{\lambda,l}}{\partial x_{i',j'}^{\lambda',l-1}} = \begin{cases} 1 & if \quad x_{i',j'}^{\lambda',l-1} \quad is \quad the \quad \max imum \quad value \\ 0 & otherwise \end{cases}$$

We are now able to compute the error gradients for both trainable parameters $a^{\lambda,l}$ and $b^{\lambda,l}$. However, the formal description of the Backpropagation algorithm from Section 5.2.2 needs to be adjusted, as it computes adjustment

for weights Δw_{ij} in fully-connected layer. In case of a subsampling layer, it needs to compute the adjustment of the trainable parameters $\Delta a^{\lambda,l}$ and $\Delta b^{\lambda,l}$.

At this point we are able to propagate the error down towards the input layer using Eq. (5.19), and adjust the trainable parameters using Eq. (5.23) and Eq. (5.24).

$$\Delta a^{\lambda,l} = -\eta \frac{\partial E}{\partial a^{\lambda,l}} \tag{5.23}$$

$$\Delta b^{\lambda,l} = -\eta \frac{\partial E}{\partial b^{\lambda,l}} \tag{5.24}$$

5.5.2. Backpropagation in Convolutional Layer

In the case of convolutional layer, the only trainable parameters are the weights shared for neurons in the same feature map and bias.

Let us discuss how the weights affect the error by computing the derivative $\frac{\partial E}{\partial w_{i,j}^{\lambda,\lambda',l}}$ and $\frac{\partial E}{\partial b^{\lambda,l}}$ for the preceding convolutional layer l, feature map λ, and the respective feature map λ' in the layer $l-1$. These derivatives can be computed using the chain rule. The expression $\partial E \big/ \partial x_{p,q}^{\lambda,l}$ is assumed to be known.

$$\frac{\partial E}{\partial w_{i,j}^{\lambda,\lambda',l}} = \sum_{p=0}^{m_l-1}\sum_{q=0}^{n_l-1} \frac{\partial E}{\partial x_{p,q}^{\lambda,l}} \cdot \frac{\partial x_{p,q}^{\lambda,l}}{\partial w_{i,j}^{\lambda,\lambda',l}} = \sum_{p=0}^{m_l-1}\sum_{q=0}^{n_l-1} \frac{\partial E}{\partial x_{p,q}^{\lambda,l}} . x_{p+i,q+j}^{\lambda',l-1} . \phi'\left(x_{p+i,q+j}^{\lambda',l-1} . w_{p,q}^{\lambda,\lambda',l} + b^{\lambda,l} \right)$$

$$\frac{\partial E}{\partial w_{i,j}^{\lambda,\lambda',l}} = \sum_{p=0}^{m_l-1}\sum_{q=0}^{n_l-1} \frac{\partial E}{\partial x_{p,q}^{\lambda,l}} \cdot \frac{\partial x_{p,q}^{\lambda,l}}{\partial w_{i,j}^{\lambda,\lambda',l}} = \sum_{p=0}^{m_l-1}\sum_{q=0}^{n_l-1} \frac{\partial E}{\partial x_{p,q}^{\lambda,l}} . \phi'\left(x_{p+i,q+j}^{\lambda',l-1} . w_{p,q}^{\lambda,\lambda',l} + b^{\lambda,l} \right)$$

The last derivatives we need to know to be able to backpropagate the error is $\partial E \big/ \partial x_{p,q}^{\lambda',l-1}$. To compute it, we need to consider the set of neurons in layer l connected to the output $x_{p,q}^{\lambda',l-1}$.

Let us denote this set $\Theta_{p,q}^{\lambda',l-1}$. Let us further denote the output of neuron $n \in \Theta_{p,q}^{\lambda',l-1}$ with x_n, and denote the weight which connects neuron $(p,q,\lambda',l-1)$ to neuron n with w_n. Then we can apply the chain rule to obtain:

$$\frac{\partial E}{\partial x_{p,q}^{\lambda',l-1}} = \sum_{n \in \Theta_{p,q}^{\lambda',l-1}} \frac{\partial E}{\partial x_n} \cdot \frac{\partial x_n}{\partial x_{p,q}^{\lambda',l-1}} = \sum_{n \in \Theta_{p,q}^{\lambda',l-1}} \frac{\partial E}{\partial x_n} . w_n$$

Since we know all the outputs $x_{i,j}^{\lambda,l}$ in layer l, and $x_n \in \cup_{\lambda,i,j} x_{i,j}^{\lambda,l}$, the derivative $\partial E \big/ \partial x_n$ is also assumed to be

5.6. Overfitting and Regularization

Deep Neural Networks can have millions of free parameters, enabling the possibility to model a wide range of complex phenomena. However, a sufficiently large network might memorize peculiarities of the training data, achieving high performance on the training set without discovering the actual underlying distribution of the data that result in poor performance on the test set. In such cases, we say that the network is overfitting.

Fig. 5.6 displays a typical example of overfitting, where the training error keeps decreasing while the test error starts increasing. One approach to counter overfitting is to reduce the number of free parameters, i.e., reduce the capacity of the network. But determining the number of parameters needed is not a trivial task, and too few parameters might lead to the opposite effect, namely underfitting (see Fig. 5.6), where the network has insufficient capacity to model the data. Instead, we deploy a range of different techniques to reduce overfitting, referred to as

regularization techniques, which aim to restrict the network such that strong generalization capability is encouraged. This section will give an overview of the most common techniques and how to apply them.

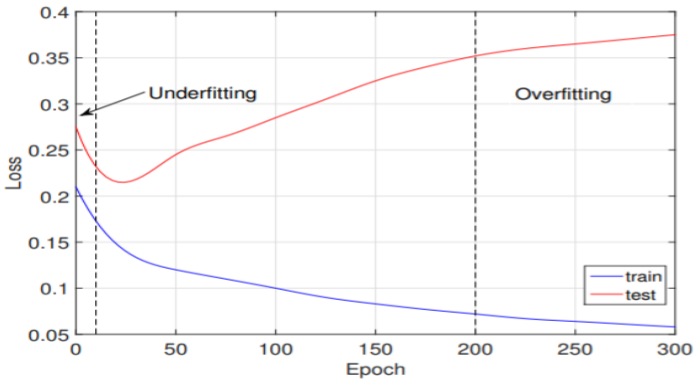


FIGURE 5. 6: An example of underfitting and overfitting problems

5.6.1. Dropout

One of the more recent regularization techniques is a technique referred to as Dropout [105]. The term dropout refers to randomly "dropping out", or omitting, units during the training process of a neural network. By dropping a unit out, we mean temporarily removing it from the network, along with all its incoming and outgoing connections, as shown in Fig. 5.7. The choice of which units to drop is random. In the simplest case, each unit is retained with a fixed probability p independent of other units, where p can be chosen using a validation set.

Dropout can be modeled as a simple layer with non-trainable weights of value one. It has one non-trainable parameter p, where $1-p$ is the probability of setting a weight to zero which effectively cut off the connection to the next layer.

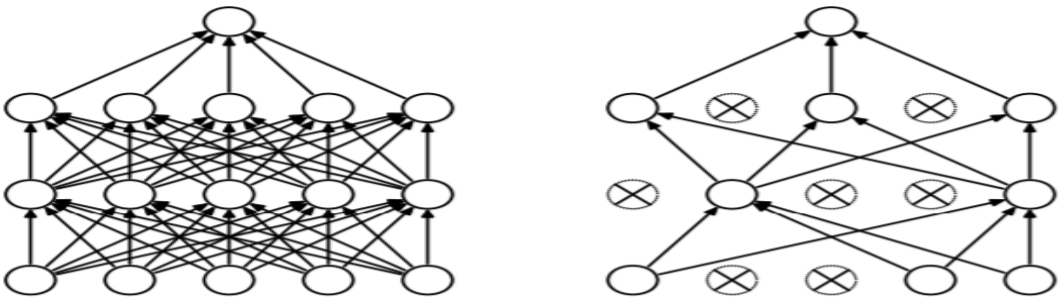
With dropout, the feed-forward operation becomes:

$$r_j^l \sim Bernoulli(p)$$

$$\tilde{x}^l = r^l * x^l$$

$$x_i^{l+1} = \phi(w_i^{l+1}\tilde{x}^l + b_i^{l+1})$$

Here $*$ denotes an element-wise product. For any layer l, r^l is a vector of independent Bernoulli random variables each of which has probability p of being 1. This vector is sampled and multiplied element-wise with the outputs of that layer, x^l, to create the thinned outputs $\tilde{x}^l$. The thinned outputs are then used as input to the next layer. This process is applied at each layer. This amounts to sampling a sub-network from a larger network. For learning, the derivatives of the loss function are backpropagated through the sub-network. At test time, the weights are scaled as $w_{test}^l = pw^l$



(a) Standard Neural Net

(b) After applying dropout.

FIGURE 5. 7: Dropout Neural Net Model. Left: A standard neural net with two hidden layers. Right: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped

5.6.2. Weight Decay

When trying to fit a high dimensional hyperplane to a number of data points the probability of a unique optimal fit is very unlikely. Any infinitesimal change to any of the hyperplane parameters has to worsen the fit which together with a finite number of data points makes this certainly improbable. It is therefore a reasonable assumption that there exist multiple sets of these parameters which optimizes the fit. Moreover, when optimizing with a subset of the data various sets of parameters might improve the fit due to utilizing local structures. A technique for selecting a specific set of parameters and avoiding overfitting on local structures in the data so that the generalization capabilities of the network improve is to introduce weight decay.
Weight decay is achieved by including a penalty term in the cost function, such that Eq. (2.2) becomes

$$L(\theta, D) = E(\theta, D) + \alpha \sum_{w} w^2$$

where the sum is over all weights in the neural network and α is the regularization term. This $L2$-regularization tells the network to prefer smaller weights. Large weights are only allowed if they make a large improvement to the loss function.

5.6.3. Batch Normalization

Batch Normalization (BN) [62] is a recent technique that aims to accelerate training of DNNs by reducing internal covariate shift. Internal covariate shift refers to the fact that the distribution of each layer's input changes during training, as the parameters of the previous layer changes. When a network grows deeper, these shifts can become amplified, and the learning rate must be kept small to avoid large adjustments. But a small learning rate slows down training considerably and might prevent the model from finding good minima. Batch Normalization aims to diminish the effect of internal covariate shift by normalizing the input of each layer, such that the mean and variance is approximately 0 and 1, respectively. In doing so, the gradient's dependence on the scale of the parameters is reduced which allows a higher learning rate without divergence issues. Additionally, Batch Normalization can act as a regularizer since it restricts the activations to a certain range.

For a layer l, the activation vector (input) x^l will be normalized by

$$\hat{x}^l = \frac{x^l - E\left[x^l\right]}{\sqrt{Var\left[x^l\right]}}$$

where the expectation and variance are computed over the training set. However, since networks are often trained using mini-batches, estimates of the mean and variance are produced from each mini-batch. But by normalizing the input, information about the absolute scale of activations is discarded, which limits the network's ability to represent data. To preserve information, γ^l and β^l, are added to each layer l that scales and shift the normalized activation vector in the following way:

$$\tilde{x}^l = \gamma^l \hat{x}^l + \beta^l$$

These normalized activations are dependent on the mini-batch used to estimate the mean and variance. However, during inference, the output should only depend on the input. There are two approaches for estimating a mean and variance that can be utilized for model testing. One approach is to use the population mean and variance during inference. Another is to keep a running average of the mean and variance during training, which enables monitoring of the accuracy during training.

The algorithm is illustrated by focusing on an activation vector x^l and considering a mini-batch B of size m,

$B = \left\{x_1^l, x_2^l, ..., x_m^l\right\}$

Normalized values are denoted $\left\{\hat{x}_1^l, \hat{x}_2^l, ..., \hat{x}_m^l\right\}$ and their linear transformations are $\left\{\tilde{x}_1^l, \tilde{x}_2^l, ..., \tilde{x}_m^l\right\}$ This transform is referred to as

$$BN_{\gamma^l,\beta^l} : \left\{x_1^l, x_2^l, ..., x_m^l\right\} \rightarrow \left\{\tilde{x}_1^l, \tilde{x}_2^l, ..., \tilde{x}_m^l\right\}$$

and the algorithm is presented in Algorithm 5.1.

Algorithm 5.1 Batch Normalizing Transform, applied to activation x^l over a mini-batch.

Input: Values of x^l over the mini-batch $\boldsymbol{B}$

Parameters to be learned: γ^l **and** β^l

Output: $\left\{\tilde{x}_i^l = BN_{\gamma^l,\beta^l}(x_i^l)\right\}$

1 : $\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i^l$ **% Mini-batch mean**

2 : $\sigma_B^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}\left(x_i^l - \mu_B\right)^2$ **% Mini-batch variance**

3 : $\hat{x}_i^l \leftarrow \frac{x_i^l - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}$ **% Normalize**

4 : $\tilde{x}_i^l \leftarrow \gamma^l \hat{x}_i^l + BN_{\gamma^l,\beta^l}(x_i^l)$ **% Scale and shift**

5.6.4. Early Stopping

For the development of DNNs it is common to split the data into the following three parts:

- Training set: A set of examples used to find the optimal parameters to perform the desired task.
- Validation set: A set of examples used for model selection and hyperparameter tuning.
- Test set: A set of examples used to evaluate the performance of the network.

Although we must always consider the available data, a rule of thumb is to split the complete data set such that the training set makes up 50% of the data while the validation and test set make up on 25% each. It is essential that the test set is kept entirely separate and treated as unknown up until evaluation. This separation is to ensure we are testing the model on unseen data to asses if the network has found a good general description of the underlying structure of the data or overfitted on the training set.

A validation set is needed for several reasons. Firstly, since the weights and biases are initialized randomly, we cannot be confident that a decrease in error on the training set was a result of an adjustment to some hyperparameter or just a particularly good initial weight configuration. By monitoring the validation set, we can see how the adjustment of hyperparameters affects data separate from the training set. Secondly, it is not guaranteed that the model corresponding to lowest error on the training set will be the model that yields the best performance on the test set, as it might be overfitted like in Fig. 5.7. Instead, we monitor the validation set, stop training when the error on the validation set starts increasing and save the model corresponding to the lowest error on the validation set. This procedure is referred to as early stopping [35], and act as a regularization technique since it limits the amount of iteration and ends the training before it overfits. However, the example shown in Fig. 5.8 is idealized, and when working with real data the output error might fluctuate, which means that the error of the validation set can increase for some steps before it decreases to a new, lower value. To solve this predicament, a new hyperparameter called patience is introduced, which determines how many iterations we are willing to wait before training is stopped. If the validation error starts increasing and does not reach a lower validation error than the previous lowest validation error before the number of iterations since the previous lowest value becomes larger than the number of iterations allowed by the patience parameter, we stop the training and save the model corresponding to the lowest validation error, as illustrated in Fig. 5.8.

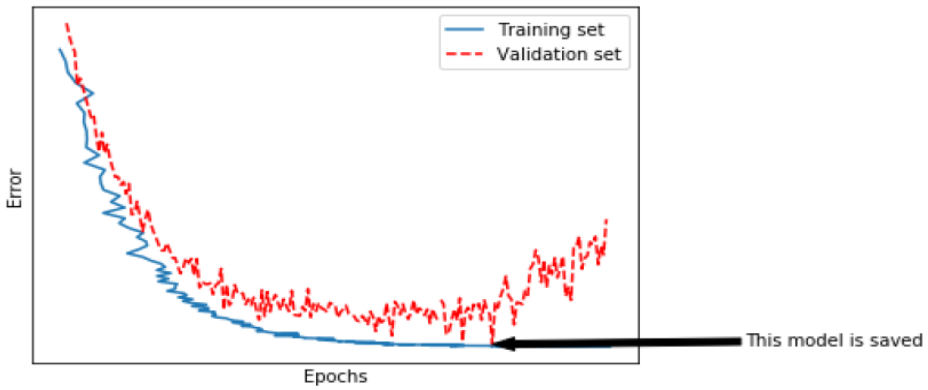


FIGURE 5. 8: Error as a function of iteration steps for training and validation set.

5.7. Conclusion

This chapter contains a theoretical overview of CNNs. We showed how CNNs are an improvement on the general idea of artificial neural networks. Introduction of parameter sharing and local connectivity made very deep CNNs not only feasible, but very efficient. Their ability to automatically learn appropriate representations of the data makes problems involving large amounts of data. As the retrospective analysis of state-of-the art models shows, the field of CNNs is rapidly changing. Most innovative CNN architectures are based on simple yet powerful ideas, and their elegant mathematical implementation makes the architecture successful. Motivated by this success, we evaluate in the following part a CNN Based framework for classifying MRI to diagnose Alzheimer's disease.

Part 1

Classification of Alzheimer's disease from Brain MRI using DCNN

Contents

5.8. Introduction 98
5.9. Related Work 100
5.9.1. LeNet-5 102
5.9.2. AlexNet 104
5.9.3. VGG-16 107
5.9.4. GoogLeNet 109
5.9.5. Residual Network (ResNet) 112
5.9.6. DenseNet 118
5.10. Network Structure of the Proposed IRDNet 112
5.11. Experimental Setup and Results 118
5.11.1. Dataset Description 118
5.11.2. Evaluation Metrics 119
5.11.3. Results and Discussion 119
5.12. Conclusion 123

5.8. Introduction

Alzheimer's disease (AD) is a chronic neurodegenerative disease which is presently incurable, the cause of which is not yet understood. As of 2015, roughly 30 million people around the world suffer from AD [73]. Some of its symptoms consist of disorientation, language difficulties, memory loss and mood swings. The average life expectancy after onset is about 3 to 9 years and the disease progressively worsen the patient's quality of life. With the aging population, efforts have been made to find better treatments for this disease. However, attempts to develop more efficient drugs have seldom been successful [66]. For this reason, more focus has shifted towards finding more efficient methods for early diagnosis [94] of the disease, and postponing the irreversible damage caused by it. Alzheimer's disease is usually diagnosed based upon the person's medical history, behavioral patterns and the medical history of relatives. The National Institute of Neurological and Communicative Disorders and Stroke (NINCDS) and the Alzheimer's Disease and Related Disorders Association (ADRDA, now known as the Alzheimer's Association) has established the most commonly used NINCDS-ADRDA Alzheimer's Criteria for diagnosis in 1984 [93]. These criteria require that the presence of cognitive impairment and a suspected dementia

syndrome, be confirmed by neuropsychological testing for a clinical diagnosis of possible or probable AD [73]. However, diagnosing Alzheimer's requires a careful medical evaluation. Moreover, neuropsychological tests such as the mini-mental state examination (MMSE) are widely used to evaluate the cognitive impairments needed for the diagnosis of the disease [34]. The main disadvantages of the MMSE are difficulty in identifying mild cognitive impairment and difficulty in recording changes in cases of severe dementia [80]. Alzheimer's disease is considered as the loss of neurons and synapses in the cerebral cortex and in certain subcortical regions [16]. This loss of neurons and synapses from the disease leads to clearly visible differences in brain tissues as illustrated in Fig. 5.9. The hippocampal atrophy, ventricle enlargement and cortex shrinkage are sensitive features of Alzheimer's disease. Therefore, doctors perform brain scans, such as Computed Tomography (CT), Magnetic Resonance Imaging (MRI), or Positron Emission Tomography (PET), to rule out the other possible causes for the symptom. Currently, MRI is the most sensitive imaging test of the brain in routine clinical practice. MRI scans are specially used in brain imaging where it provides information about the morphology of the white matter, gray matter and cerebrospinal fluid (CSF). Structural MRIs are often used to non-invasively capture the regional brain atrophy and help understand the functional and anatomical neural changes related to AD and to analyze the thickness of the gray matter of the patients, the cerebral cortex and the volume of several regions of the brain like the hippocampus, whose atrophy is one of the first signs of Alzheimer's. Thus, they are recognized as an important biomarker for AD progression and are widely studied with pattern recognition methods for AD diagnosis. These scans are currently used to assist in the diagnostic process [98] by manual inspection of a neurologist. This can give an indication regarding the correct diagnose, as the difference between a scan of a healthy person and one diagnosed with AD is clear in many cases (see Fig. 5.9). Still, this requires a medical expert to manually inspect the image, which is time consuming and less accuracy making the diagnosis of Alzheimer's disease is often difficult, particularly in the early stages. Moreover, the diagnosing accuracy mainly depends on the experience of the radiologists. In order to quickly and accurately determine the diagnosis, it is desirable to have system for CAD. For example, a system for automatic analysis of medical images could provide a medical doctor with markers giving indications of the correct diagnosis. This could potentially reduce the time and cost for the diagnostic process, as well as patient discomfort. To this end, we will in this work investigate the possibility of applying methods of artificial intelligence to develop a system that can classify automatically MR-based imaging data and, more importantly, to distinguish brain affected by AD data from healthy subjects, without the requirement for manual inspection. A multitude of machine learning techniques have been used for this task, including independent component analysis, support vector machines, and penalized regression. These methods have proven to be very effective in classifying AD and normal controls from neuroimages than human radiologists.

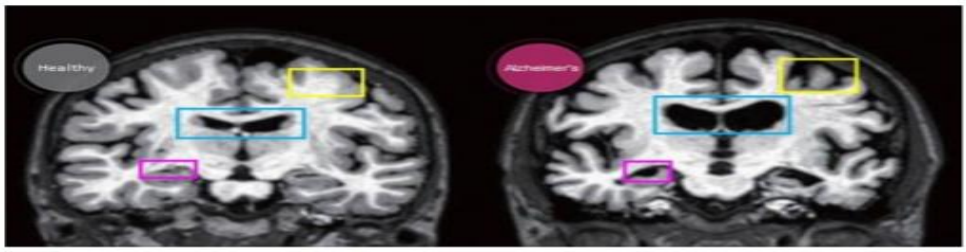


FIGURE 5. 1: MRI view of a healthy brain (left) and an Alzheimer's brain (right) [16]. (Yellow - Cortex, Blue - Ventricle, Purple - Hippocampus volume reduced)

In recent years, deep learning can perform better than the traditional machine learning method in classifying Alzheimer's disease. This technique can assist scientists and clinicians in diagnosing this brain disorder and will also aid in the accurate and timely diagnosis of Alzheimer's patients [16]. The purpose of this artificial intelligence (AI) technique is to model high-level abstractions of data so that computers can learn to distinguish AD from healthy brains. Among the deep learning models, the convolutional neural network was the most stable technique. This network is based on several layers of representation of very large datasets with successive transformations. These transformations amplify the characteristics of the input data which discriminate them and attenuate their

variation. The first successful convolutional networks (LeNet) [82] were developed in the late 1990s by Yann LeCunn. The idea of this network is to pass the image through a succession of convolutional filters providing a reduced and relevant description of the image. These features are then sent to a multilayer perceptron composed of hidden layers and a completely connected output layer allowing the classification of the objects presented in the image. This famous network successfully classified digits and was applied to hand-written check numbers. The application of this network was expanded to more complicated problems and their hyper parameters adjusted for new issues. However, more developed versions of LeNet have been successfully tested [82] and more complex CNNs architectures was developed from high volume data, including AlexNet, VGGNet, GoogLeNet, DenseNet and ResNet. In the following subsection, we describe these ten deep convolutional networks that are used to distinguish AD from normal controls.

5.9. Related work

In the recent years, a number of research studies have been published focusing on the development of a deep convolutional neural network for classification of AD from healthy controls using MRI data. Sarraf and Tofighi [151] implemented the LeNet-5 to learn the features from functional MRI data in order to classify AD and NC controls. They implemented the LeNet-5 to learn the features from functional MRI data in order to predict the output. This architecture reaches an accuracy of 96.86% in the classification of AD against normal controls on the ADNI database. Manhua Liu et al. implemented Densely Connected Convolutional Networks (DenseNet) [59] to learn features for the classification of AD. This model is evaluated on the baseline T1-weighted structural MRI data collected from 97 AD, 233 MCI, and 119 NC (healthy brain) subjects in the Alzheimer's disease Neuroimaging Initiative (ADNI) database. DenseNet method achieves an accuracy of 88.9% and an AUC (area under the ROC curve) of 92.5% for classifying AD vs. NC subjects, and an accuracy of 76.2% and an AUC of 77.5% for classifying MCI vs. NC subjects. Heung-Kook Choi et al. [139] utilize different deep learning models such as GoogLeNet, VGG-16, AlexNet and ResNet-18, and compare their efficiency to classify MR images into AD, mild cognitive impairment and NC. The performance of this method has been evaluated over Alzheimer's disease Neuroimaging (ADNI) dataset. These models achieved an accuracy of 98.25%, 93.97%, 88.66% and 96.8%, respectively for three-class classification (AD vs. Normal vs. mild cognitive impairment). In the following subsections, we describe the above famous architectures such as AlexNet, ZFNet, GooLeNet, VGGNet, ResNet and DenseNet.

5.9.1. LeNet-5

Introduced in the late 90s by LeCun et al. [84], LeNet-5 was the first commercial success of CNNs, and was deployed for Optical Character Recognition (OCR) of handwritten digits. This network possesses the basic units of convolutional neural network, such as convolutional layer, pooling layer and full connected layer. As shown in Fig. 5.10 (input image data with 32×32 pixels): LeNet-5 consists of seven layers, not counting the input, all of which contain trainable parameters (weights). In the Fig. 5.10, layers are labelled by Cx, Sx, Fx which represent convolutional layer, sub-sampling layer and fully-connected layer, respectively, and x represents layer index.

Layer $C1$ is a convolutional layer with six feature maps. Each unit in each feature map is connected to a 5×5 neighborhood in the input. The size of the feature maps is 2828 which prevents connection from the input from falling off the boundary. $C1$ contains 156 trainable parameters and 122 304 connections.

Layer $S2$ is a sub-sampling layer with six feature maps of size 14×14. Each unit in each feature map is connected to a 2×2 neighborhood in the corresponding feature map in $C1$. The four inputs to a unit in $S2$ are added, then multiplied by a trainable coefficient, and then added to a trainable bias. The result is passed through a sigmoidal function. The 2×2 receptive fields are nonoverlapping, therefore feature maps in $S2$ have half the number of rows and column as feature maps in $C1$. Layer $S2$ has 12 trainable parameters and 5880 connections.

Layer $C3$ is a convolutional layer with 16 feature maps. Each unit in each feature map is connected to several 5×5 neighborhoods at identical locations in a subset of $S2$'s feature maps. The connections between layers $S2$ and

$C3$ are not fully connected, as is shown in the Table 5.1. These combinations are arbitrary but also reduce the number of parameters of the network and forces different feature maps to extract different features as they get different inputs. The rationale behind the connection scheme in Table 5.1 is the following. The first six $C3$ feature maps take inputs from every contiguous subset of three feature maps in $S2$. The next six take input from every contiguous subset of four. The next three take input from some discontinuous subsets of four. Finally, the last one takes input from all $S2$ feature maps. Layer $C3$ has 1516 trainable parameters and 156 000 connections. Layer $S4$ is a subsampling layer with 16 feature maps of size 5×5. Each unit in each feature map is connected to a 2×2 neighborhood in the corresponding feature map in $C3$, in a similar way as $C1$ and $S2$. Layer $S4$ has 32 trainable parameters and 2000 connections.

Layer $C5$ is a convolutional layer with 120 feature maps. Each unit is connected to a 5×5 neighborhood on all 16 of $S4$'s feature maps. Here, because the size of $S4$ is also 5×5, the size of $C5$'s feature maps is 1×1. Thus, $C5$ is equivalent to a fully connected layer. Layer $C5$ has 48120 trainable connections. The fully connected layer $F6$ has 84 units connected to the 120 units of $C5$. All the units of the layers up to $F6$ have a sigmoidal activation function φ of the type: $\varphi(v_j) = A\tanh(Sv_j)$

where v_j is the activity level of the unit. A and S are two constant parameters for the sigmoid function (see Ref. [84] for the setting of A and S)

Finally, the output layer is composed of Euclidean Radial basis function units, one for each class, with 84 inputs each.

Table 5. 1: Connection scheme between the layers $S2$ and $C3$ of LeNet-5

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

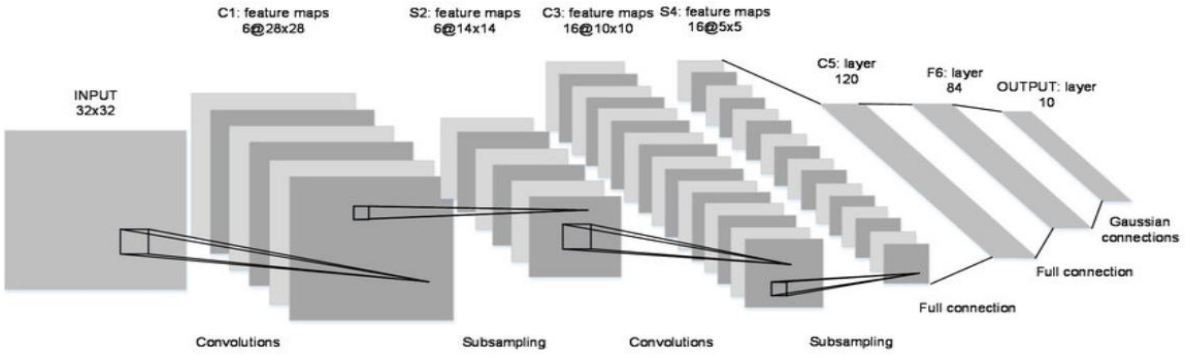


FIGURE 5. 2: Diagram of LeNet-5 architecture [84].

This famous architecture has been successfully used to classify functional MRI data of Alzheimer's subjects from normal controls. Table 5.2 summarizes the number of feature maps, output size, kernel size, stride and activation function of LeNet-5 architecture in each layer.

Table 5. 2: Configuration of LeNet-5 Architecture

Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	32×32	-	-	-
1	Convolution	6	28×28	5×5	1	Tanh
2	Average Pooling	6	14×14	2×2	2	Tanh
3	Convolution	16	10×10	5×5	1	Tanh
4	Average Pooling	16	5×5	2×2	2	Tanh
5	Convolution	120	1×1	5×5	1	Tanh
6	FC	-	84	-	-	Tanh

5.9.2. AlexNet

Alex Krizhevesky et al. [74] proposed a wider and deeper CNN model compared to LeNet and won the most difficult ImageNet challenge for visual object recognition called the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012. AlexNet achieved state-of-the-art recognition accuracy against all computer vision and the traditional machine learning approaches. It was a significant breakthrough in the field of computer vision and machine learning for classification tasks and visual recognition and is the point in history where interest in deep learning increased rapidly.

The architecture of AlexNet as shown in Fig. 5.11 contains five convolutional layers (where the convolution operation is followed by the ReLU activation function), three pooling, two local response normalizations and three fully connected layers, where the last one is a softmax layer that gives the probability of belonging to a certain class of the image. Pooling and local response normalization layers are utilized at first and second layers. The last pooling layer is applied to the output of the fifth convolutional layer. This was an innovative ordering of operations, as in the previous famous network, LeNet, a convolution was always followed by the non-linearity and pooling, not by another convolution. The network has two parallel pipelines executed in different GPUs in order to speed up the process. We notice that the first convolution layer uses a filter with receptive field 11x11, and with a stride of 4. The receptive field diminishes as we go deeper in the network to 5x5 and finally 3x3. This means that in the beginning the network is trying to capture statistics for each pixel with respect to a wider region. As filter size decreases, image is downsampled by maxpooling operations, whereas the number of filters increases from 96 to 256 and then 384. Thus, the data is compressed spatially and upsampled in depth. The model has a huge number of weights and memory needed for keeping the feature maps during the forward/backward passes.

Two novel concepts are introduced in AlexNet: the usage of the ReLU activation instead of tanh, and the local response normalization (LRN). The authors empirically show that training with a non-saturating nonlinearity trains several times faster and reaches better convergence point. ReLUs do not necessarily need input normalization, since for learning to happen it is enough that some training examples have positive input. However, they find that using local normalization helps generalization. For a unit $a^i_{x,y}$ obtained by applying filter i to position (x,y), the local response-normalized activity $b^i_{x,y}$ is defined as:

$$b^i_{x,y} = a^i_{x,y} \Big/ \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} \left(a^j_{x,y}\right)^2 \right)^{\beta}$$

Where n defines the window size used for normalization and N is the image size. Whereas, the hyper-parameters k and α are selected using a validation set. This is a way of averaging positions across feature maps, introducing intensity invariance and competition among units. Lastly, the authors make the network robust to some transformations, by exposing it to an augmented dataset (flipped, translated, reflected images where label is preserved), and address over-fitting with applying dropout in the fully-connected layers.

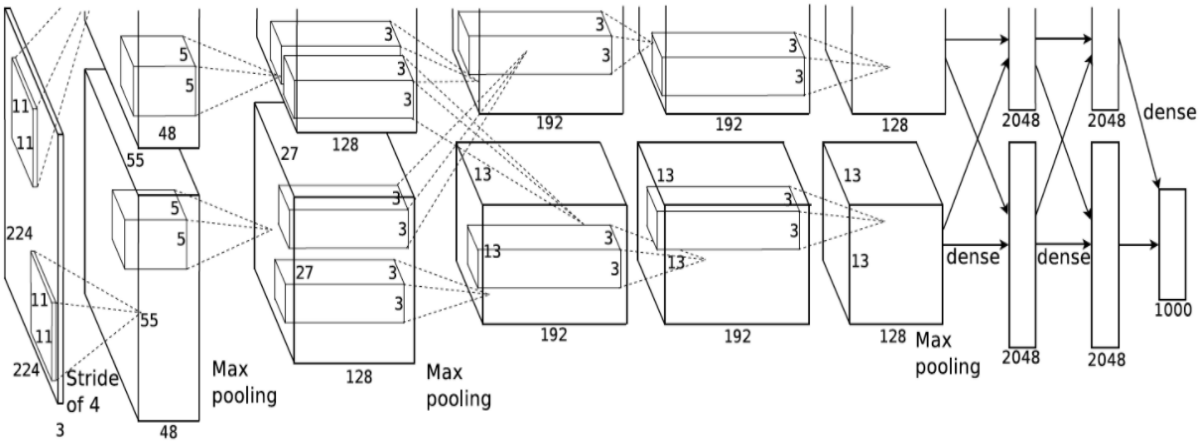


FIGURE 5. 3: AlexNet Architecture [74]

The AlexNet architecture has been successfully used to classify MRI data of Alzheimer's subjects from normal controls. Table 5.3 summarizes the number of feature maps, output size, kernel size, stride and activation function of This architecture in each layer

Table 5. 3: Configuration of AlexNet Architecture

Layer		Featute Map	Size	Kernel Size	Stride	Activation
Input	Image	1	$227\times227\times3$	-	-	-
1	Convolution	96	$55\times55\times96$	11×11	4	ReLu
	Max Pooling	96	$27\times27\times96$	3×3	2	ReLu
2	Convolution	256	$27\times27\times256$	5×5	1	ReLu
	Max Pooling	256	$13\times13\times256$	3×3	1	ReLu
3	Convolution	384	$13\times13\times384$	3×3	1	ReLu
4	Convolution	384	$13\times13\times384$	3×3	1	ReLu
5	Convolution	256	$13\times13\times256$	3×3	2	ReLu
	Max Pooling	256	$6\times6\times256$	-	-	ReLu
6	FC	-	9216	-	-	ReLu
7	FC	-	4096	-	-	ReLu
8	FC	-	4096	-	-	ReLu
Output	FC	-	Number of classes	-	-	Softmax

5.9.3. VGG-16

VGGNet [123] is a convolutional neural network designed by Karen Simonian and Andrew Zisserman, who won the runner-up in the 2014 ImageNet ILSVRC Challenge. This network uses convolution sequences instead of using a single convolution depth level as AlexNet and replaces the large convolution kernels of sizes 11×11 and 5×5 with 3×3 kernel size. The authors of this work found that the small convolution kernels can increase the depth of non-linear networks and make networks more complex and able to learn more complex features. Furthermore, they prove that the increase in network depth can get better performance. However, deeper networks also need a huge computational requirement. As shown in Fig. 5.12, VGG-16 model is composed of 16 layers including 13 convolutional and 3 fully connected layers. The first two layers consist of 64 filters of size 3×3×3 with stride of 1. After these two layers there is a max pooling layer with a window of size 2×2 and stride 2. Another two layers consist of 128 convolution filters of size 3×3×3 with stride 1 and afterwards once again max pooling layer with a window of size 2×2 and stride 2 is applied. Another 3 layers consist of 256 filters of size 3×3×3, all of them having stride 1. Once again max pooling operation is present after this triple with a window of 2×2 and stride 2. Next 6 layers consist of 512 filters of size 3×3×3 with all of them having stride 1. MaxPooling layer is applied after first 3 layers of this 6-tuple and after the last layer, both of them having window of 2×2 and stride 2. The network is

topped with three fully connected layers, where the first two are of size 4096 and use a ReLU activation function. The final classification layer is a softmax activated layer, to reflect the number of classes. During training, they also used two Dropout layers between the two final fully connected layers.



FIGURE 5. 4: Architecture of VGG-16 [123].

The VGG-16 architecture has been successfully used to classify MRI data of Alzheimer's subjects from normal controls. Table 5.4 summarizes the number of feature maps, output size, kernel size, stride and activation function of This architecture in each layer

Table 5. 4: Configuration of VGG-16 Architecture

Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	227×227×3	-	-	-
1	2×Convolution	64	55×55×96	3×3	1	ReLu
	Max Pooling	64	27×27×96	3×3	2	ReLu
3	2×Convolution	128	27×27×256	3×3	1	ReLu
	Max Pooling	128	13×13×256	3×3	2	ReLu
5	2×Convolution	256	13×13×384	3×3	1	ReLu
	Max Pooling	256	13×13×384	3×3	2	ReLu
7	3×Convolution	512	13×13×256	3×3	1	ReLu
	Max Pooling	512	-	3×3	2	ReLu
10	3×Convolution	512	-	3×3	1	ReLu
	Max Pooling	512	-	3×3	2	ReLu
13	FC	-	9216	-	-	ReLu
14	FC	-	4096	-	-	ReLu
15	FC	-	4096	-	-	ReLu
Output	FC	-	Number of classes	-	-	Softmax

5.9.4. ***Goog*LeNet**

GoogLeNet also known as Inception-V1, was a model proposed by Szegedy et al. [126] with the objective of reducing computational cost compared to the traditional CNN. The proposed method was to incorporate "Inception blocks" that had variable receptive fields, which were created by different kernel sizes (1x1, 3x3, and 5x5) executed in parallel on the characteristic map resulting from the previous layer. These receptive fields captured sparse correlation patterns and spatial information at different scales in the new feature map.

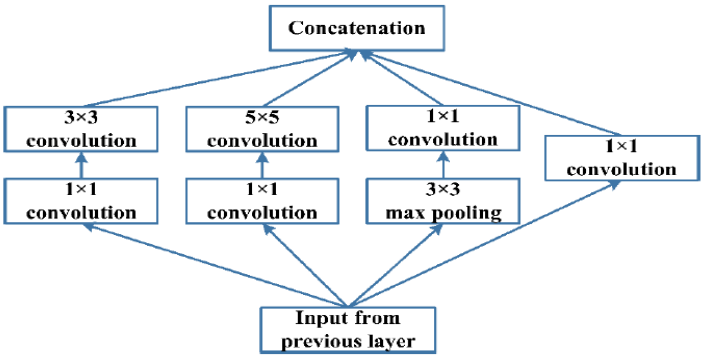


FIGURE 5. 5: The Inception block in GoogLeNet [126].

As depicted in Fig. 5.14, GoogLeNet are designed for the fixed image dimension of 224×224 in the RGB color space. Overall, it has 22 learnable layers including nine "Inception" blocks, five convolutional layers, three average pooling layers, four max-pooling layers, five fully-connected layers, and three softmax layers for the main auxiliary classifiers in the network. Moreover, it applies ReLU activation function in all of the convolutional layers and uses dropout regularization in the fully-connected layers.
As shown in Fig 5.13, the inception block contains four parallel paths. The first three paths use convolutional layers with window sizes of 1×1, 3×3, and 5×5 to extract information from different spatial sizes. The middle two paths perform a 1×1convolution on the input to reduce the number of channels, reducing the model's complexity. The fourth path uses a 3×3 maximum pooling layer, followed by a 1×1 convolutional layer to change the number of channels. The four paths all use appropriate padding to give the input and output the same height and width. Finally, the outputs along each path are concatenated along the channel dimension and comprise the block's output. However, this network is much wider and deeper than any network before it, but has a much lower number of parameters compared to VGG-16 and AlexNet. GoogLeNet had 7M network parameters when VGG-16 had 138M and AlexNet had 60M. A more detailed explanation of all the relevant parameters of GoogleNet model can be found in the original paper [126].

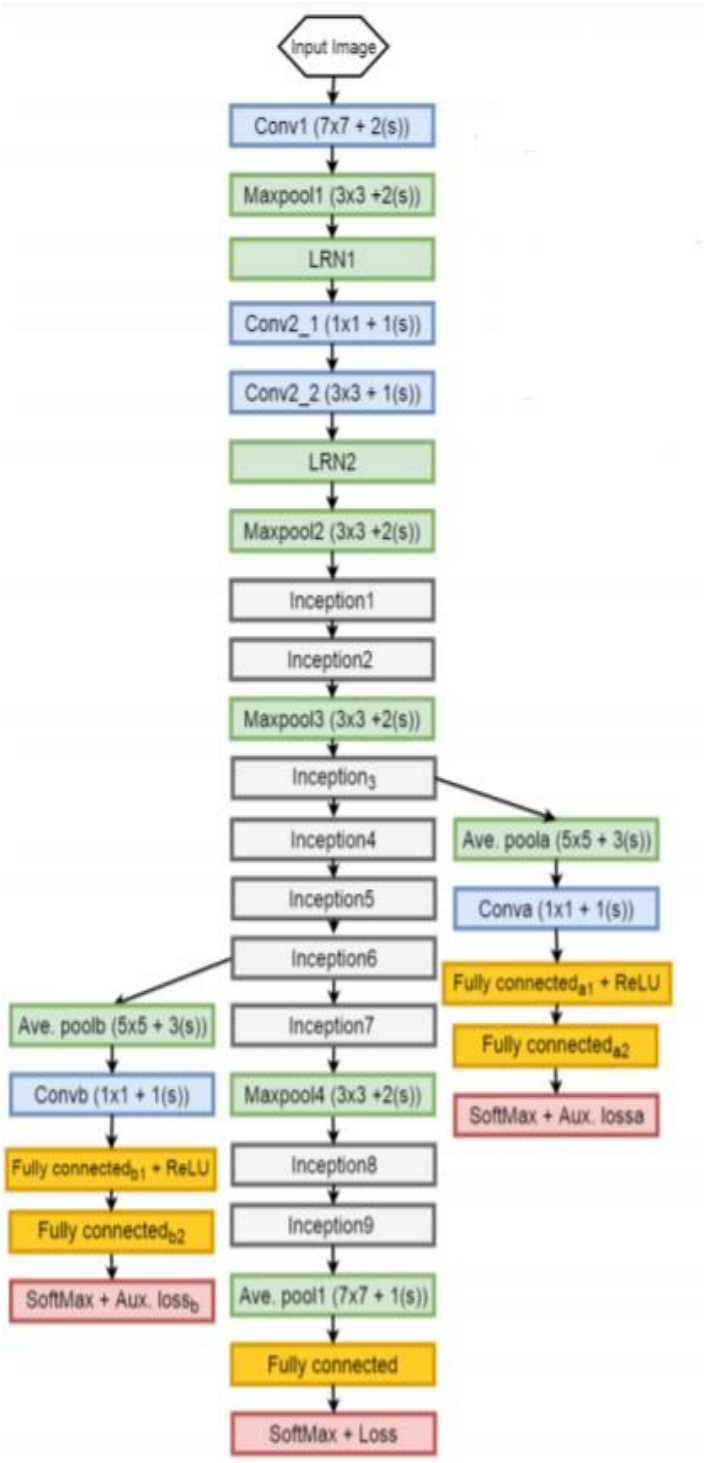


FIGURE 5. 6: Architecture of GoogLeNet [126].

The GoogLeNet architecture has been successfully used to classify MRI data of Alzheimer's subjects from normal controls. Table 5.5 summarizes the number of feature maps, outpout size, kernel size, stride and activation function of This architecture in each layer. "#3×3 reduce" and "#5×5 reduce" stands for the number of 1×1 filter in the reduction layer used before the 3×3 and 5×5 convolutions. One can see the number of 1×1 filter in the projection layer after the built-in max-pooling in the pool proj column. All these reduction/projection layers use rectified linear activation as well.

Table 5. 5: Configuration of GoogLeNet Architecture

Type	Kernel size/stride	Output size	Depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj
Convolution	7×7/2	112×112×64	1	-	-	-	-	-	-
Max Pooling	3×3/2	56×56×64	0	-	-	-	-	-	-
Convolution	3×3/1	56×56×192	2	-	64	192	-	-	-
Max Pooling	3×3/2	28×28×192	0	-	-	-	-	-	-
Inception (3a)		28×28×256	2	64	96	128	16	32	32
Inception (3b)		28×28×480	2	128	128	192	32	96	64
Max Pooling	3×3/2	14×14×480	0	-	-		-	-	-
Inception (4a)		14×14×512	2	192	96	208	16	48	64
Inception (4b)		14×14×512	2	160	112	224	24	64	64
Inception (4c)		14×14×512	2	128	128	256	24	64	64
Inception (4d)		14×14×528	2	112	144	288	32	64	64
Inception (4e)		14×14×832	2	256	160	320	32	128	128
Max Pooling	3×3/2	7×7×832	0	-	-	-	-	-	-
Inception (5a)		7×7×832	2	256	160	320	32	128	128
Inception (5b)		7×7×1024	2	384	192	384	48	128	128
Avg Pool	7×7/1	1×1×1024	0	-	-	-	-	-	-
Dropout (40%)	7×7/1	1×1×1024	0	-	-	-	-	-	-
Linear	7×7/1	1×1×number of clusters	1	-	-	-	-	-	-
Softmax	7×7/1	1×1×number of classes	0	-	-	-	-	-	-

5.9.5. Residual Network (ResNet)

The Residual Network [55] was the absolute winner of ILSVRC 2015 in all vision tasks of classification, recognition, localization, segmentation, etc. Resnet was developed by Kaiming He et al. with the intent of designing ultra-deep networks that did not suffer from the degradation problem that predecessors had. The authors formulated the degradation problem as follow: Let us consider the problem of fitting an underlying mapping $H(x)$ through a unit made of few stacked layers, where x is the input to the unit. Since we assume that such function can be approximated by the multiple non-linear layers, then we can reason that these layers can approximate the residual function $F(x) = H(x) - x$. Although, both forms can be approximated, it may be more difficult for the solver to learn one than the other, and the difficulty does not come from vanishing gradient in the forward or backward steps, as this problem is addressed by the BN-layers introduced in the Inception Networks**.** The solution is given by replacing the previous convolution network units with a new building block: The Residual Unit, shown in the Fig. 5.15. Given that x_l is the input to the layer l, y_l is the output and x_{l+1} is the input to the consecutive layer, then we can write the block as:

$$y_l = F(x_l, w_l) + h(x_l)$$
$$x_{l+1} = \phi(y_l)$$

where the first part is the residual mapping to be learned, the shortcut connection has no parameters as h is selected to be the identity mapping, they are added element-wise and then passed through a non-linearity to get the output. ϕ is selected to be the ReLU function. If the residual part of the unit changes the dimensions from the input, then in order to match we perform a linear projection through (1x1) convolutions in the identity mapping, and the formula would be: $y_l = F(x_l, w_l) + W_{s,l} x_l$. ResNet was built by several stacked residual units and developed with many different numbers of layers: 18, 34, 50, 101 and 152. As shown in Fig. 5.16, the popular ResNet-152 has a total of 152 layers and its configuration is as follows. The set consists of a convolution layer, batch normalization, and ReLU (activation function) with 151 layers, and a fully connected last layer. Only the first convolution layer is set to 7×7 kernel with stride 2 and padding 3, and the kernel size of all subsequent convolution

layers is 3×3. In addition, by setting stride 2 in 4 convolution layers, multi-scale can be considered without additional pooling layer. The total number of weights for the whole network is 60.2 M.

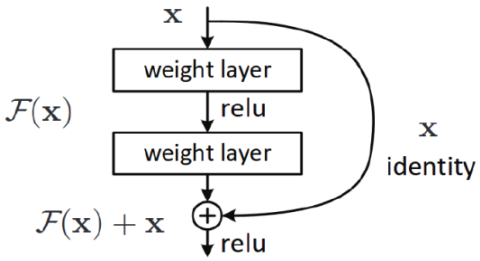


FIGURE 5. 7: Residual learning: a building block [55].

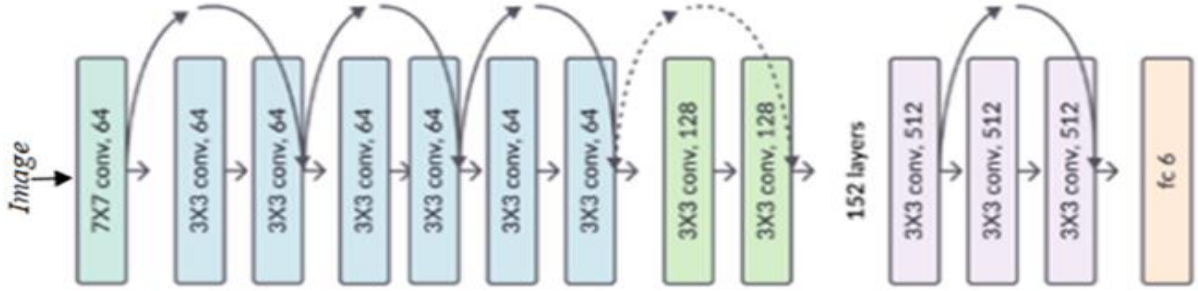


FIGURE 5. 8: Architecture of ResNet-152 [55]

The number of layers of ResNet has been modified from 152 to 18 layers so that it adapts to the classification of Alzheimer's subjects from normal controls. Table 5.6 summarizes the number of feature maps, output size, kernel size and stride of the ResNet and its variants architecture in each layer.

Table 5. 6: Configuration of ResNet-152 Architecture and its variants

Layers	Output Size	18-layers	34-layers	50-layers	152-layers
Conv1	112×112	7×7 conv, 64, stride 2 3×3 max pool, stride 2			
Conv2	56×56	$\begin{pmatrix}3\times3,64\\3\times3,64\end{pmatrix}\times2$	$\begin{pmatrix}3\times3,64\\3\times3,64\end{pmatrix}\times3$	$\begin{pmatrix}1\times1,64\\3\times3,64\\1\times1,256\end{pmatrix}\times3$	$\begin{pmatrix}1\times1,64\\3\times3,64\\1\times1,256\end{pmatrix}\times3$
Conv3	28×28	$\begin{pmatrix}3\times3,128\\3\times3,128\end{pmatrix}\times2$	$\begin{pmatrix}3\times3,128\\3\times3,128\end{pmatrix}\times4$	$\begin{pmatrix}1\times1,128\\3\times3,128\\1\times1,512\end{pmatrix}\times4$	$\begin{pmatrix}1\times1,128\\3\times3,128\\1\times1,512\end{pmatrix}\times8$
Conv4	14×14	$\begin{pmatrix}3\times3,256\\3\times3,256\end{pmatrix}\times2$	$\begin{pmatrix}3\times3,128\\3\times3,128\end{pmatrix}\times6$	$\begin{pmatrix}1\times1,256\\3\times3,256\\1\times1,1024\end{pmatrix}\times6$	$\begin{pmatrix}1\times1,256\\3\times3,256\\1\times1,1024\end{pmatrix}\times36$
Conv5	7×7	$\begin{pmatrix}3\times3,512\\3\times3,512\end{pmatrix}\times2$	$\begin{pmatrix}3\times3,128\\3\times3,128\end{pmatrix}\times3$	$\begin{pmatrix}1\times1,512\\3\times3,512\\1\times1,2048\end{pmatrix}\times3$	$\begin{pmatrix}1\times1,512\\3\times3,512\\1\times1,2048\end{pmatrix}\times3$
	1×1	Average pooling, FC, softmax			

5.9.6. DenseNet

Similar to ResNet network, DenseNet [59] was proposed to improve the declined accuracy caused by the vanishing gradient in high-level neural networks. In simpler terms, due to the longer path between the input layer and the output layer, the information vanishes before reaching its destination. The problem with ResNet was that it explicitly preserves information through additive identity transformations due to which many layers may contribute very little or no information. To address this problem, DenseNet used cross-layer connectivity but, in a modified fashion. DenseNet connected each preceding layer to the next coming layer in a feed-forward fashion, thus the lth layer receives the feature-maps of all preceding layers, $x_0,...,x_{l-1}$, as input

$$x_l = H_l([x_0, x_1, ..., x_{l-1}])$$

Where $[x_0, x_1, ..., x_{l-1}]$ are the concatenated features for layers $0,\ldots,l-1$ and H_l is defined as a composite function of three consecutive operations: batch normalization (BN), followed by a rectified linear unit (ReLU) and a 3 × 3 convolution (Conv).

This means that instead of L connections for a traditional *L*-layers network, there are $L(L+1)/2$ connections. Furthermore, the network is divided into multiple densely connected dense blocks as the size of feature-maps may change. These blocks are then separated by a batch-normalization layer and a 1×1 convolutional layer followed by a 2×2 average pooling layer. If H produces k feature maps, the number of input feature maps in l^{th} layer is $k_l = k_0 + k \times (l-1)$ where k is the growth rate and k_0 is the number of channels in the input layer. A 5-layer dense block with growth-rate $k = 4$ is illustrated in Fig. 5.17, where $k_4 = 5 + 4 \times (4-1) = 17$.

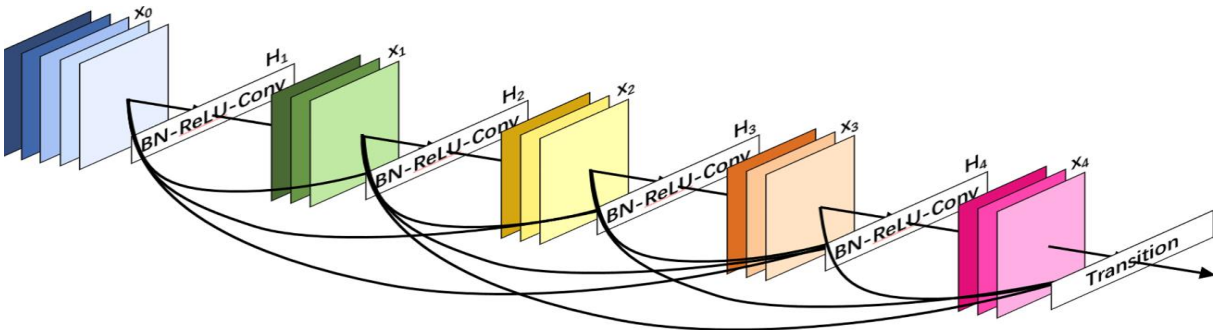


FIGURE 5. 9: A 5-layer dense block with a growth rate of *k* = 4. Each layer takes all preceding feature maps as input [59]

The DenseNet has different versions, like DenseNet-121, DenseNet-160 and DenseNet-201. The numbers denote the number of layers in the neural network. As shown in Fig. 5.18, DenseNet-121consists of one convolutional layer, four dense blocks, three transition layers, a 7×7 average pooling layer and a total of 121 layers (117-conv, 3-transition, and 1-classification). The four blocks have 6, 12, 24, 16 layers consecutively and each transition layer is between two consecutive dense blocks. The transition layer reduces the number of input feature maps by using 1×1 convolution kernel and halves the number of input feature maps by using 2 ×2 average pooling layers. DenseNet has a narrower network width, fewer parameters, and less computational cost. It not only alleviates the vanishing-gradient problem, but also achieves better performance than ResNet.

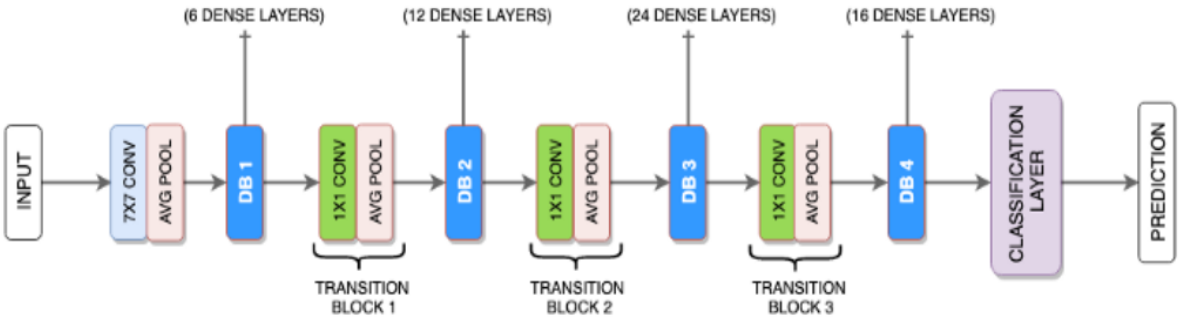


FIGURE 5. 10 Architecture of DenseNet-121 with 4 dense blocks [59]. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

The DenseNet-121 architecture has been successfully used to classify MRI data of Alzheimer's subjects from normal controls. Table 5.7 summarizes the number of feature maps, output size, kernel size and stride of the DenseNet-121 and its variants architecture in each layer.

Table 5. 7: Configuration of DenseNet-121 Architecture and its variants

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Conv	112×112	7×7 conv, stride 2			
Pooling	56×56	3×3 Max pool, stride 2, stride 2			
Dense Block (1)	56×56	$\begin{pmatrix}1\times1, conv\\3\times3, conv\end{pmatrix}\times6$	$\begin{pmatrix}1\times1, conv\\3\times3, conv\end{pmatrix}\times6$	$\begin{pmatrix}1\times1, conv\\3\times3, conv\end{pmatrix}\times6$	$\begin{pmatrix}1\times1, conv\\3\times3, conv\end{pmatrix}\times6$
Transition Layer (1)	56×56	1×1 conv			
	28×28	2×2 average pool, stride 2			
Dense Block (2)	28×28	$\begin{pmatrix}1\times1, conv\\3\times3, conv\end{pmatrix}\times12$	$\begin{pmatrix}1\times1, conv\\3\times3, conv\end{pmatrix}\times12$	$\begin{pmatrix}1\times1, conv\\3\times3, conv\end{pmatrix}\times12$	$\begin{pmatrix}1\times1, conv\\3\times3, conv\end{pmatrix}\times12$
Transition Layer (2)	28×28	1×1 conv			
	14×14	2×2 average pool, stride 2			
Dense Block (3)	14×14	$\begin{pmatrix}1\times1, conv\\3\times3, conv\end{pmatrix}\times24$	$\begin{pmatrix}1\times1, conv\\3\times3, conv\end{pmatrix}\times32$	$\begin{pmatrix}1\times1, conv\\3\times3, conv\end{pmatrix}\times48$	$\begin{pmatrix}1\times1, conv\\3\times3, conv\end{pmatrix}\times64$
Transition Layer (3)	14×14	1×1 conv			
	7×7	2×2 average pool, stride 2			
Dense Block (4)	7×7	$\begin{pmatrix}1\times1, conv\\3\times3, conv\end{pmatrix}\times16$	$\begin{pmatrix}1\times1, conv\\3\times3, conv\end{pmatrix}\times32$	$\begin{pmatrix}1\times1, conv\\3\times3, conv\end{pmatrix}\times32$	$\begin{pmatrix}1\times1, conv\\3\times3, conv\end{pmatrix}\times48$
Classific-ation Layer	1×1	7×7 global average pool FC, softmax			

Following the study of the six deep convolutional networks, we will present a table summarizing the advantages, disadvantages and the key idea for each.

Table 5. 8: Advantages and drawbacks of six architectures of deep convolutional neural networks

Architecture	Key idea	Advantage	Drawbacks
LeNet	Use convolution to extract spatial features	Spatial correlation exploited to reduce the number of parameters	Overfits in some cases
AlexNet	- Extraction of low, medium and high-level functionality using large and small filters. - Parallel use of GPUs as an accelerator to manage complex architectures. - introduction of regularization to form more complex models while maintaining lower over-adjustment levels	Reduces the overfitting problems by using the dropout layer	Requires larger memory and computational resources
VGGNet	Gave the idea of receptive field block.	Learn a lot of features	- The possibility of overfitting on given dataset. - More expensive to evaluate and uses a lot more memory and parameters
GoogLeNet	- Introduction of an inception module which considerably reduced the number of parameters in the network. - Propose a new idea of split, transform and merge.	- Obtaining more precise multi-scale spatial information. - The network less subject to overfitting and allows it to be deeper because of the reduced number of network parameters	Tedious parameter customization due to heterogeneous topology
ResNet	Adding a direct connection channel to the network to make the gradient disappear while reducing the size and speeding up training	Reduces the vanishing gradient descent problem to a minimum	Relative complex model due to the extensive use of shortcut path that skips layers, normalizations operations and pooling.
DenseNet	Introducing Dense blocks to facilitate the concatenation of previous feature maps into the input of the next layer	- Avoid relearning of redundant feature-maps - Improves the flow of information and gradients across the network, which facilitates training	Large increase in parameters due to increase in number of feature maps

5.10. Network Structure of the Proposed IRDNet

Recently, deep convolutional neural networks are ongoing towards an extended depth and width of the network architecture due to the improvement of hardware devices. From AlexNet [74] to VGGNet [123] as well as GoogleNet [126] and ResNet [55], both the depth and the width of CNNs have continued to increase, which can facilitate to achieve excellent performance in image classification. However related works demonstrate that such drawbacks as poor convergence, gradient disappearance and overfitting might be generated with comprehensive network structures. To alleviate these problems and encourage feature propagation, DenseNet allows layers access to feature maps from all of its preceding layers. DenseNet [59], one of the existing deepest CNNs, is mainly composed of Dense Blocks (extracting and propagating feature maps) and transition layers (connecting two adjacent Dense Blocks). As mentioned earlier, DenseNet obtains state-of-the-art accuracy, it exploits the residual connections in residual network (ResNet). In dense block, we represent the input image as x_0, output of the lth layer as x_l and each convolutional module as function H. Since input to the lth layer is outputs of all previous layers. The formula is presented as follows:

$$x_l = H\left(\left[x_0, x_1, ..., x_{l-1}\right]\right)$$

where $\left[x_0, x_1, ..., x_{l-1}\right]$ is the concatenation of outputs of all previous layers.

In DenseNet, number of output feature-maps for each convolution module are always the same, which is denoted by k. Thus, the output number of the lth layer is $k_0 + \left(l-1\right)\times k$, where k_0 is the number input to the first dense block. k was referenced as growth rate. As DenseNet goes deeper, number of input feature-maps would become excess very soon. To settle this problem, the author put 1×1 convolution module (as bottleneck layer) before the 3×3 convolution module. Thus, the convolutional module has changed from BN+ReLU+3×3Conv to BN+ReLU+1×1Conv+BN+ReLU+3×3Conv. (And the new convolution module is counted as two layers instead of one). The normal setup for output feature-maps of bottleneck layer (1×1Conv) is $4k$. Thus, inputs to every 3×3 Conv layer are fixed to $4k$. To further improve model compactness, number of feature-maps can also be reduced in transition layer.

Each layer in DenseNet receives input from all preceding layers and forwards its output to all subsequent layers in a block. DenseNet also concatenates feature maps to give variety in the inputs of the subsequent layers, unlike the summation in residual networks. This feature reuse (feature concatenation) operation creates many paths, resulting in an exponential increase in memory requirements and computational complexity. Although DenseNet has achieved remarkable results due to dense connectivity and deep structure, there are still some shortcomings: a) DenseNet requires quadratic memory with respect to their depth; b) DenseNet has a great amount of redundant information which makes network convergence difficult; c) There are total L(L−1)/2 connections for a L-layer DenseNet. The excessive connections not only decrease networks computation-efficiency and parameter-efficiency, but also make networks more prone to overfitting; (d) The correlation of feature maps is not explicitly modeled. It is beneficial to adaptively learn the correlation coefficients by modeling feature maps from different dimensions and network paths. These existing disadvantages are greatly related with DenseNet's performance in classification accuracy, resource occupation, training time and memory consumption. To settle the problem of the excessive connections in dense block that increase the chances of overfitting the data, we proposed a method to reduce the number of dense connections. Zuiler [158] found out that for a deep neural network, shallower layers can learn concrete features, whereas deeper layers can learn abstract features. Based on this observation, we can drop connections from middle layers and reduce connections for each layer. So, our idea is simply dropping connections from middle layers and only retaining the nearest and farthest connections. And then we can extend the network to deeper or wider, which would result in better performance.

The proposed modified dense block (MDB) consists of several layers connected in a dense pattern where each layer takes the nearest and farthest feature maps as its input (Fig. 5.19). The layers are connected using concatenation and transformed using a non-linear transformation. Let us consider an rth MDB. A non-linear

transformation performed within MDB, placed at the lth level within the network, produces feature maps x_l that are given as

$$x_l = H_l\left(\left[x_{i_r}, x_{i_r+1}, \ldots x_{i_r+s/2}, x_{l-s/2}, \ldots, x_{l-1}\right]\right)$$

$$x_l = \phi\left(W_l\left[x_{i_r}, x_{i_r+1}, \ldots x_{i_r+s/2}, x_{l-s/2}, \ldots, x_{l-1}\right] + b_l\right)$$

where $H_l(.)$ is the lth nonlinear transformation, $[x_{i_r}, x_{i_r+1}, \ldots, x_{i_r+s/2}, x_{l-s/2}, \ldots, x_{l-1}]$ refers to the concatenation of the feature maps χ produced in layers. ϕ denotes the activation function of ReLU, W_l denotes the weights of the lth convolutional layer in this r^{th} MDB, and b denotes the bias term of the lth convolutional layer.

Note that are feature maps fed into the input of the rth MDB. The input of the MDB is placed at the i_r-th level of the network, $i_r < l < o_r$ and o_r is a level at which the output of the rth MDB is placed. We call s path. we set it to 2 so that we preserve only the farthest and nearest connections. For $s = 2$, the connections to every layer are at most 3.

The lth nonlinear transformation $H_l(.)$ is a composite function of several consecutive operations. We used a basic version of the composite function. The basic version consists of three operations: Batch normalization (BN) [62], followed by a rectified linear unit (ReLu) [74], followed by a convolution (Conv) [82]. Using a short notation, this version of the function can be written as BN - ReLu- $(w \times h)$ Conv, where h and w are the height and width of the filters. If necessary, convolutions were zero-padded to keep the feature map size fixed.

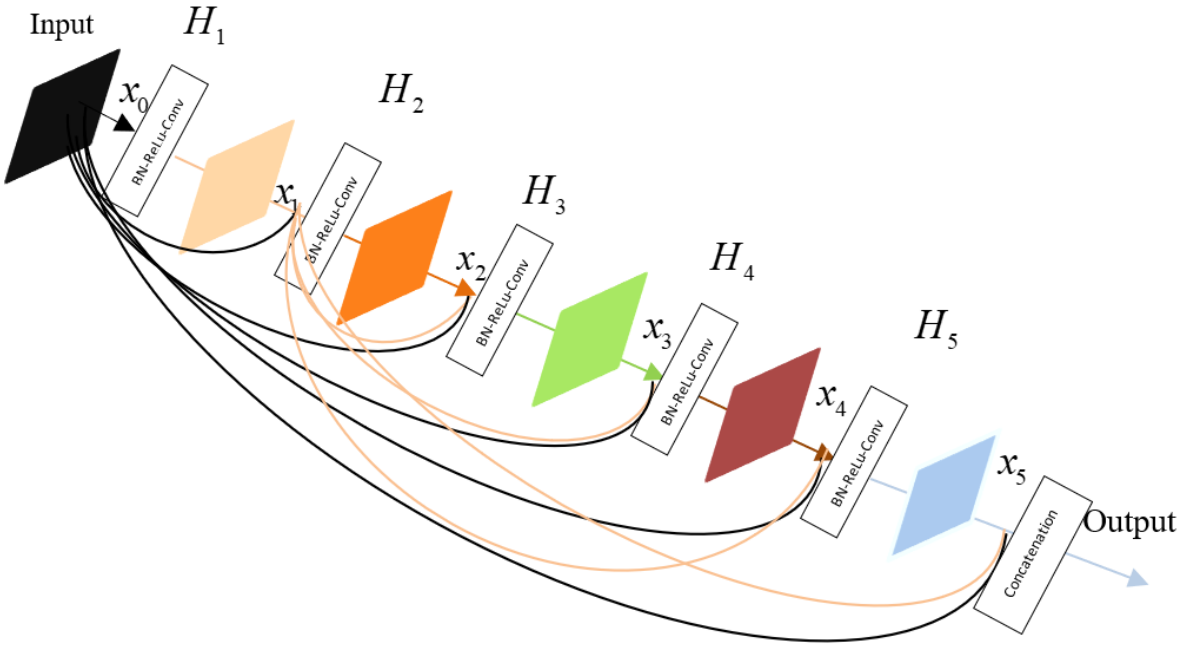


FIGURE 5. 11: MDB diagram with 6 layers (path=2, the connections to every layer are at most 3)

Fig. 5.19 shows the proposed modified dense block of six layers which is placed at the input of a network. The first layer x_0 is the input of the MDB. The second layer x_1 is applied to the input x_0 to create f feature maps, which are concatenated to the input x_0. The third layer x_2 is then applied to create another f features maps, which

are again concatenated to x_0 and x_1. The fourth layer x_3 is then applied to create another f features maps, which are again concatenated to x_0 and x_1. The fifth layer x_4 is then applied to create another f features maps, which are again concatenated to x_0 and x_1. The sixth layer x_5 is then applied to create another f features maps x_1. The output of the proposed block is the concatenation of the output of last layer x_5 with the first layer x_0 and the second layer x_1. Generally, if we have *L* layers, then the output of the block is the concatenation of the outputs of last layer with the first layer and the second layer and thus contains $3\times f$ feature maps.

In order to further ensure that the input MDB information is not lost and improve the information flow of the network, a local residual connection, as shown in Fig. 5. 20, is added between the input and the output of the MDB to better retain the characteristics of the front layer. This skip connection method can solve the problem of the gradient disappearance in deep network, and the fusion of local features of modified dense connected blocks will be achieved to ensure that the network can extract concrete and abstract features.

In order to retain the consistency of the input and the output of MDB, the local residual connection included is a 1×1 convolution. Thus, the output $x_{r,q}$ of this operation can be obtained by

$$x_{r,q} = \Psi_{conv}([x_{i_r}, x_{i_r+1}, x_l])$$

where the Ψ_{conv} denotes the operation of the 1×1 convolutional layer and $x_{r,q}$ is the output of the 1×1 convolutional layer in the rth MDB.

For simplicity, we note I_r , the input of the rth residual dense block (MDRB) and O_r , the output of the rth MDRB. Thus, the output O_r of the rth MDRB (see Fig. 5.20) can be denoted as:

$$O_r = I_r + x_{r,q}$$

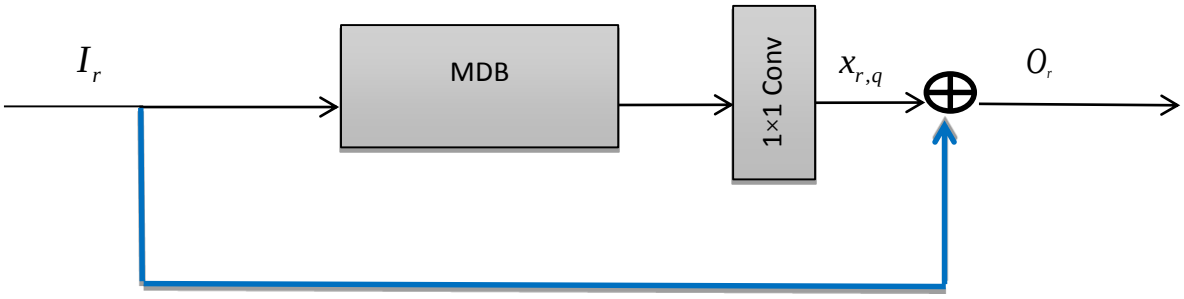


FIGURE 5. 12: The architecture of the proposed residual dense block (MDRB)

In DenseNet [59], dense block uses a small size of convolutional kernel. Fixing the kernel size on CNNs is an inefficient assumption. This is because the visual cortex can collect information with different scales at the same processing level. If the kernel size is selected to be small, the resulting of feature maps have few pixels containing small object. Therefore, single-branch CNN with fixed kernel size is unable to capture characteristics of local objects at different scales, but the fused multi-scale features may be complementary to each other. To overcome the disadvantage of using single-branch, we introduce multi-branch approach to create parallel branches with different kernel sizes and combine them to obtain highly informative feature maps. In the proposed residual dense block (MDRB), we use the multi-branch structure to provide different kernel sizes in different columns, whose overall architecture is illustrated in Fig. 5.21. There are three parallel CNNs whose structures are the same but filters of convolutional layers are in each layer in 3 different sizes (5×5, 9×9and 13×13) to capture to capture multiple size patterns over natural images. Batch normalization and rectified linear unit (ReLU) is then applied to

the convolved images and strides of filters in each column are set to one. The rationale behind using different filter sizes is to detect local features using filters of size 5×5 and rather global features by filters of size 13×13 while 9×9filter size is to detect what is missed by the other two filters.

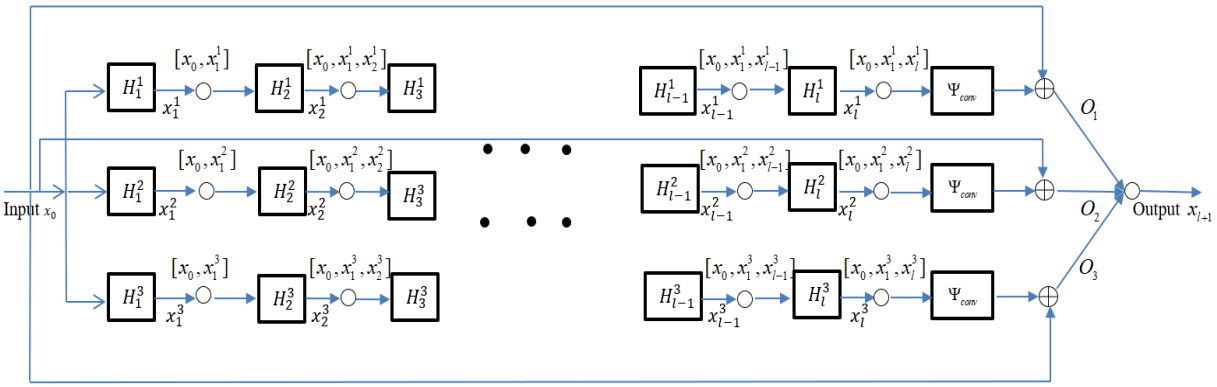


FIGURE 5. 13: Multi-scale residual dense architecture (MSRD).

Multi-scale Dense Features Fusion: we construct a three-bypass network and each bypass use different convolutional kernel. In this way, the information between those bypasses can be shared with each other, which helps to detect the image features at different scales. Moreover, to preserves the feed-forward nature and extracts local and global dense feature, the modified dense blocks presented in Fig. 5.19 is applied to adaptively fuse the states from the whole convolutional layers.

The operation in the first bypass can be defined as:

$$x_l^1 = H_l^1([x_0, x_1^1, x_{l-1}^1])$$

$$x_l^1 = \phi\left(W_l^{13\times13}.[x_0, x_1^1, x_{l-1}^1] + b_l\right)$$

The operation in the second bypass can be defined as:

$$x_l^2 = H_l^2([x_0, x_1^2, x_{l-1}^2])$$

$$x_l^2 = \phi\left(W_l^{9\times9}.[x_0, x_1^2, x_{l-1}^2] + b_l\right)$$

The operation in the third bypass can be defined as:

$$x_l^3 = H_l^3([x_0, x_1^3, x_{l-1}^3])$$

$$x_l^3 = \phi\left(W_l^{5\times5}.[x_0, x_1^3, x_{l-1}^3] + b_l\right)$$

where x_0 represent the input of the Multi-scale residual dense (MSRD). $H_l^p(.)$ denotes convolution operation, ReLU activation function and batch normalization of the pth bypass, while the subscripts 5×5, 9×9 and 13×13 represent the size of the convolutional kernel used in the layer.

Furthermore, the output of the multi-scale dense architecture can be defined as:

$$x_{l+1} = [O_1, O_2, O_3]$$

where $[O_1, O_2, O_3]$ denotes the concatenation operation of the output of the three parallel MDRB.

After stacking multiple layers, the final number of output feature map channels is the sum of the number of channels that the three parallel residual dense block generated. In order to reduce the number of channels of the final feature map, we add a layer after stacked layers (i.e., after the output of MSRD), consisting of a 1×1 convolution, which can reduce the number of feature map, thereby reducing the amount of subsequent calculations.

The following table shows the feature aggregation for the multi-scale structure in Fig. 5.21.

Table 5. 9: Feature aggregation table corresponding to the multi-scale structure

x_l^p	$p=1$	$p=2$	$p=3$
$l=1$	$H_1^1(x_0)$	$H_1^2(x_0)$	$H_1^3(x_0)$
$l=2$	$H_2^1([x_0, x_1^1])$	$H_2^2([x_0, x_1^2])$	$H_2^3([x_0, x_1^3])$
$l=3$	$H_3^1([x_0, x_1^1, x_2^1])$	$H_3^2([x_0, x_1^2, x_2^2])$	$H_3^3([x_0, x_1^3, x_2^3])$
$l=4$	$H_4^1([x_0, x_1^1, x_3^1])$	$H_4^2([x_0, x_1^2, x_3^2])$	$H_4^3([x_0, x_1^3, x_3^3])$
$l=5$	$H_5^1([x_0, x_1^1, x_4^1])$	$H_5^2([x_0, x_1^2, x_4^2])$	$H_5^3([x_0, x_1^3, x_4^3])$
…	…	…	…
$l=i$	$H_i^1([x_0, x_1^1, x_{i-1}^1])$	$H_i^2([x_0, x_1^2, x_{i-1}^2])$	$H_i^3([x_0, x_1^3, x_{i-1}^3])$

In the following, we give a detailed description of the proposed CNN (IRDNet) as well as the proposed multi-scale residual dense block.

The proposed CNN consists of four multi-scale residual dense blocks, five convolutional layer, five max pooling layers, three fully-connected layers and a softmax classifier, as illustrated in Fig. 5.

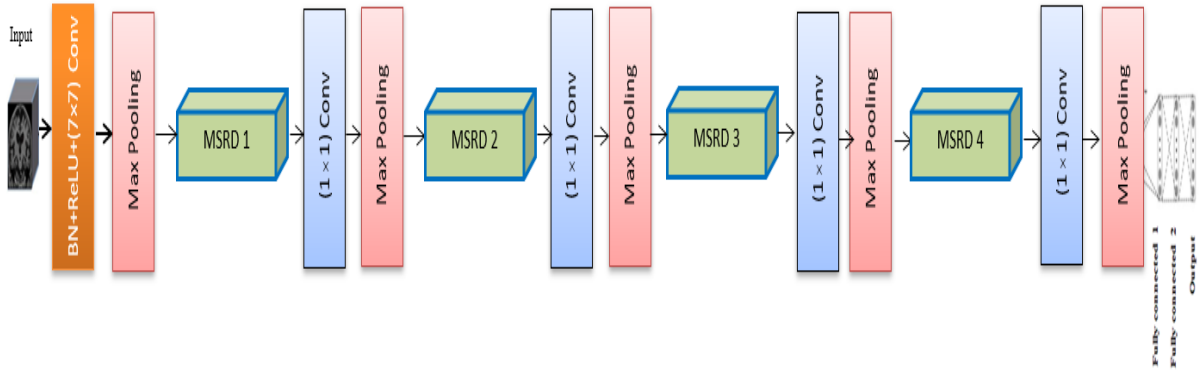


FIGURE 5. 14: The structure of the proposed IRDNet.

In our CNN, we use four blocks, each of them has six layers and three parallel paths. The concatenation of the feature maps learned by different layers increases the variation of the input of the following layers and improves the efficiency. Moreover, the use of addition layers and dense connected layers produces a much more powerful representation as it learns the complex relationships between all levels of abstraction.

Our CNN took as input grayscale images with a size of 1×215×215 pixels. Initially, the images are processed by the batch normalization and ReLU layers to speed up learning, followed by a first convolutional layer (Conv1) having a size 7×7 and a stride of one to produce 16 feature maps. The use of the convolutional layer reduced the images dimension from 1×215×215 to 16×209×209. Features generated by Conv1 are forwarded to 2×2 max pooling layer with a stride of two to sub-sample the input image by reducing its dimensionality and therefore to provide a spatial invariance and to reduce the quantity of parameters and computation in the network. The dimensions of the resulting image after max pooling layer will be reduced to 16×104×104 and the output of features maps are forwarded to MSRD 1 block (see Fig. 5.22), which contains six layers and three parallel convolutional layers followed by the batch normalization and ReLU in each layer. Three parallel filters of size 13×13, 9×9 and 5×5 are applied to the input of the block to generate respectively 5, 5 and 6 feature maps in each layer. Moreover, the output of each of them and the output of this block are concatenated to the output of three convolutional parallel

layers are forwarded to 1×1 convolutional layer with a stride of one to increase the number of feature maps to 16 in order to add them to the input of the block, and the output of the MSRD 1 block are concatenated to the output of three convolutional parallel layer. The concatenation layer is designed to exploit rich information. In this block, it has no pooling layer and before each convolutional layer we normalize each batch and we apply ReLU activation function. The output of MSRD 1 block passes through 1×1 convolutional layer with a stride of one to decrease the number of feature maps to 16, followed by max pooling layer with filter size 2×2 and a stride of 2.

The used of MSRD 1 change the feature maps dimensions from 16×104×104 to 48×101×101 and the used 1×1 convolutional layer change the feature maps dimensions from 48×101×101 to 24×101×101. Furthermore, the used of max pooling layers have reduced the feature maps dimensions from 24×101×101 to 24×52×52. After this, MSRD 2 Block is applied.

The block MSRD 2 is the same as the MSRD 1 block except it produced 8 feature maps for each convolutional layer and the output of this block passes through 1×1 convolutional layer with a stride of one to decrease the number of feature maps to 24, followed by max pooling layer with filter size 2×2 and a stride of 2. The used of MSRD 2 change the feature maps dimensions from 24×52×52 to 72×52×52 and the used 1×1 convolutional layer change the feature maps dimensions from 72×52×52 to 36×52×52. Furthermore, the used of max pooling layers have reduced the feature maps dimensions from 36×52×52 to 36×26×26. After this, MSRD 3 Block is applied.

The block MSRD 3 is the same as the MSRD 1 block except it produced 12 feature maps for each convolutional layer and the output of this block passes through 1×1 convolutional layer with a stride of one to decrease the number of feature maps to 36, followed by max pooling layer with filter size 2×2 and a stride of 2. The used of MSRD 3 change the feature maps dimensions from 36×26×26 to 108×26×26 and the used 1×1 convolutional layer change the feature maps dimensions from 108×26×26 to 54×26×26. Furthermore, the used of max pooling layers have reduced the feature maps dimensions from 54×26×26 to 54×13×13. After this, MSRD 4 Block is applied.

The block MSRD 4 is the same as the MSRD 1 block except it produced 18 feature maps for each convolutional layer and the output of this block passes through 1×1 convolutional layer with a stride of one to decrease the number of feature maps to 54, followed by max pooling layer with filter size 2×2 and a stride of 2. The used of MSRD 4 change the feature maps dimensions from 54×13×13 to 162×13×13 and the used 1×1 convolutional layer change the feature maps dimensions from 162×13×13 to 81×13×13. Furthermore, the used of max pooling layers have reduced the feature maps dimensions from 81×13×13 to 81×6×6.

Finally, the network had three fully connected layers (FC); the first one has 972 feature maps each of size 1×1. Each of the 972 is connected to all the 2916 nodes (81×6×6) in the previous layer. The second one has 648 units is connected to all the 972 nodes and finally, there is a fully connected softmax output layer with 2 possible values. The highest value determined the predicted class.

The following table summarizes the configuration of our IRDNet.

Table 5. 10: The configuration of our IRDNet architecture for OASIS dataset

Layers	**Filters size**	**Output shape (depth × height × width)**
Input image	-	1×215×215
Conv1+BN+ReLU	7×7, stride =1	16×209×209
Max Pooling	2×2, stride=2	16×104×104
MSRD 1	$\begin{pmatrix}13\times13\\9\times9\\5\times5\end{pmatrix}\times6$, stride=1	48×101×101
	1×1 Conv+BN+ReLU, stride=1	24×101×101
Max Pooling	2×2, stride=2	24×52×52

MSRD 2	$\begin{pmatrix}13\times13\\9\times9\\5\times5\end{pmatrix}\times6$, stride=1	72×52×52
	1×1 Conv+BN+ReLU, stride=1	36×52×52
Max Pooling	2×2, stride=2	36×26×26
MSRD 3	$\begin{pmatrix}13\times13\\9\times9\\5\times5\end{pmatrix}\times6$, stride=1	108×26×26
	1×1 Conv+BN+ReLU, stride=1	108×26×26
Max Pooling	2×2, stride=2	54×13×13
MSRD 4	$\begin{pmatrix}13\times13\\9\times9\\5\times5\end{pmatrix}\times6$, stride=1	162×13×13
	1×1 Conv+BN+ReLU, stride=1	81×13×13
Max Pooling	2×2, stride=2	81×6×6
Flatten	-	2916
FC1+ReLu	-	(1×972)
FC2+ReLU	-	(1×648)
FC3+Softmax	-	(1×2)

5.11. Experimental Setup and Results

In this section, we assess by different evaluation metrics the performance of our proposed model, IRDNet, to classify AD from NC in terms of accuracy and computation time using brain MRI as well as we compare its efficiency to other published deep CNN models and we discuss the obtained results. Moreover, we describe the dataset used in this study and we detail the distribution of images in training, validation and testing set.

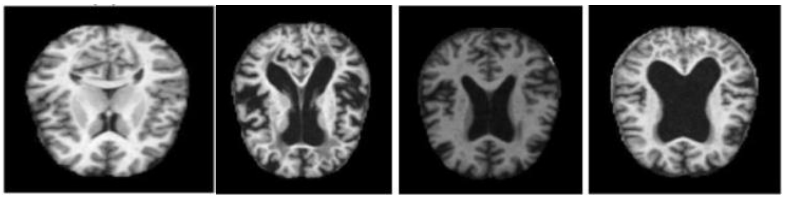
5.11.1. Dataset Description

In our experiments, we use the dataset of MRI scan which is selected as a part of the Open Access Series of Imaging Studies (OASIS) and it is available publicly at (http://www.oasisbrains.org). OASIS includes two types of data sets: cross-sectional and longitudinal MRI data. A cross-sectional set contains 416 subjects aged 18 to 96, and for each subject, 3 or 4 T1-weighted MRI are available and 100 of whom were clinically diagnosed with Alzheimer's disease. A longitudinal release includes 150 subjects aged 60 to 96. In this study, we use the cross-sectional dataset and we chose 200 subjects, 100 of whom were selected from the AD group, while the other 100 were selected from normal control group. Furthermore, 32 slices are extracted from brain MRI of each patient, which resulted in a total of 6,400 slices from the axial view of 200 patients, including 3,200 AD and the other 3,200 NC. Fig. 5.23. illustrates sample brain MRI from OASIS dataset and shows the difference between AD and NC images.

The following findings are observed in the MR images of patients affected by AD.

- The cortex curls up and damages regions associated with thinking, planning and memory.
- The narrowing is particularly marked in the hippocampus, a region of
- The cortex that plays an essential role in the formation of new memories.

- The ventricles (spaces filled with fluid inside the brain) get bigger



(a) (b) (c) (d)

FIGURE 5. 15: Four brain MRI from the OASIS dataset: (a) Normal, from (b) to (d) Alzheimer's with different stages.

5.11.2. Evaluation Metrics

To evaluate the performance of the proposed model, more performance metrics need to be investigated through this research. The most common performance measures in the field of DL are overall accuracy, precision, recall (sensitivity), and F1-score [46], which are presented from Eq. (5.25) to Eq. (5.28), respectively.

$$\Pr ecision = \frac{Sum\ of\ all\ True\ Positives\ \left(TP\right)}{Sum\ of\ all\ True\ Positives\ \left(TP\right)\ +\ all\ False\ Positives\ \left(FP\right)} \tag{5.25}$$

$$Accuracy = \frac{No.\ of\ images\ correctly\ classified}{Total\ no.of\ images} \tag{5.26}$$

$$Recall = \frac{Sum\ of\ all\ True\ Positives\ \left(TP\right)}{Sum\ of\ all\ True\ Positives\ \left(TP\right) + All\ False\ Negatives\ \left(FN\right)} \tag{5.27}$$

$$F1-score = \frac{2*\Pr ecision*\mathrm{Re}call}{\Pr ecision+\mathrm{Re}call} \tag{5.28}$$

where TP is the number of correctly labelled positive samples, FP is the number of negative samples incorrectly labelled as positive and FN is the number of positive samples incorrectly labelled as negative.

5.11.3. Results and discussion

In this section we perform experiments to assess the performance of the proposed IRDNet, which is trained to classify Brain MRI images into two categories: NC and AD. The efficiency of our model is evaluated using the 5-fold cross-validation technique for (2-class) classification problem. The dataset used was divided into five equal and independent sets. Four out of five sets were used to train and validate our model whereas the remaining set was used for the test (i.e., 70% of brain MR images are used for training, 10% for validation and 20% for testing). More details, 3200 AD brain MR images were used, where 2304 out of 3200 were randomly selected for training, 256 images for validation and 640 images for testing. Furthermore, 3200 normal brain images were used, where 2304 out of 3200 were randomly selected for training set, 256 images for validation and remaining 640 images for testing. The strategy of split dataset was repeated 5 times as illustrated in Fig. 5.24 and the distribution of brain MR images from sub-sets were summarized in table 5.11. Moreover, the proposed model was trained and were run using Tensorflow and python on a 2.5 GHz Intel Core i5 computer with 8 GB of RAM. Training is performed using Adam optimizer with following parameters: $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$ and learning rate of 0.0001. We use mini-batches of size 8, and train for about 10 epochs.

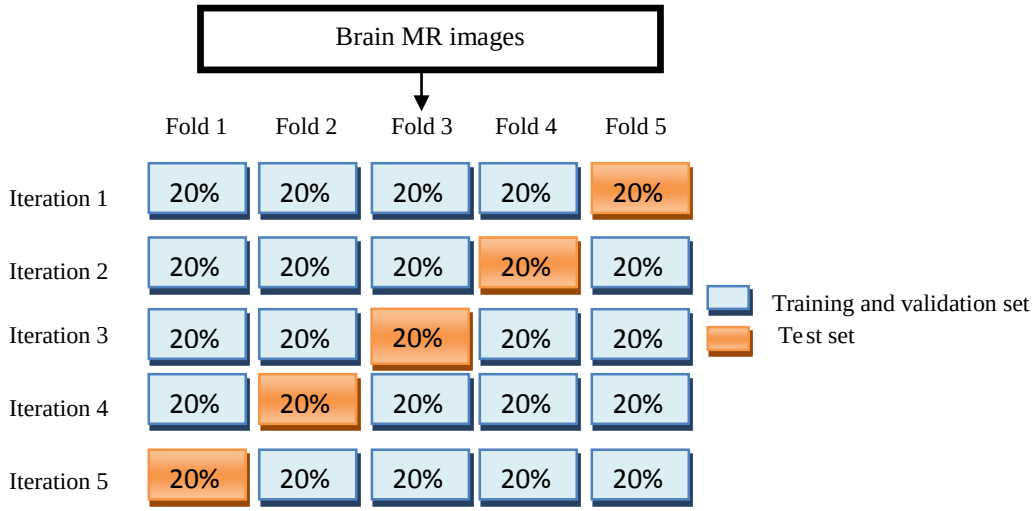


FIGURE 5. 16: Diagram of training, validation and testing set used in the five-fold cross-validation technique.

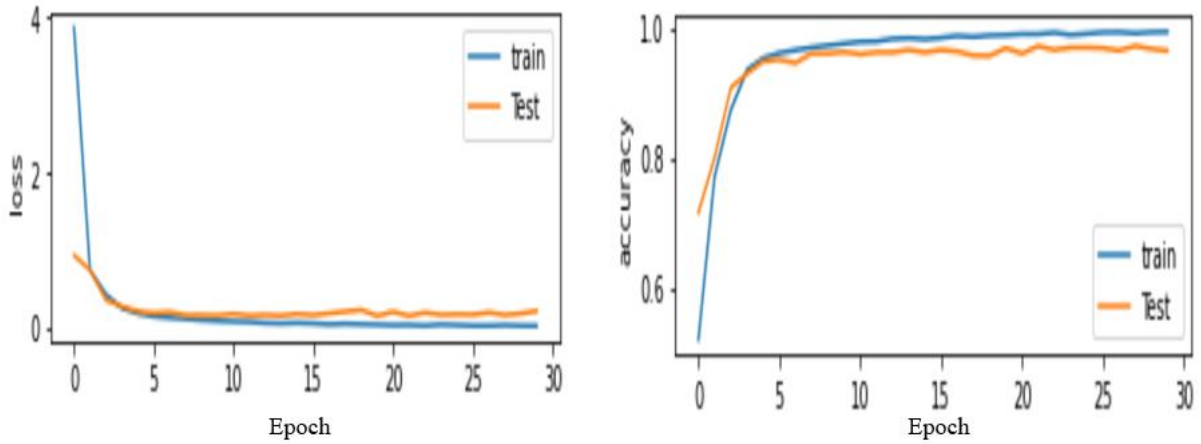
Table 5. 11: Details of training, validation and testing set.

Class	Number of images	Training	Validation	Test
AD	3200	2304	256	640
NC	3200	2304	256	640

The performance of IRDNet for 2-class classification was measured for each fold using confusion matrix (CM) and metrics given above such as precision, recall, accuracy, and F1-score. The overlapped and each separate CM are shown in Fig. 5.26. Furthermore, Overall precision, recall, accuracy, and F1-score computed for each class (AD and NC) and for each fold are presented in Tables 5.12, 5.13, 5.14, 5.15, 5.16 and 5.17 as well as the average accuracy of deep CNN models on the test set from five-fold cross-validation and the average execution time on training set to classify AD vs. NC from the OASIS database are given in Table 5.18. Moreover, the accuracy and loss curve of the proposed IRDNet model as it converges is display in Fig. 5.25

In the following, we detail the performance of our model and we discuss the obtained results.

The proposed network architecture described in the above section was implemented on the OASIS database to test the efficacy of its performance for the purpose of automated AD classification. From Fig. 5.25, it is observed that with increase in epochs, the loss value decreases, and accuracy increases. The proposed model seems to be good after visualizing the accuracy and loss of both training and testing. The condition of overfitting and under fitting did not occur significantly in the model. The training loss as well as testing loss was decreasing exponentially which conclude that the network is good for predicting the test image and perform the classification task. Fig. 5.25a shows the number of epochs versus Loss value. The total number of epochs was taken 30 and the training loss started to decrease initially. After some epoch, the loss was constant. It means that the model stops to learn from the corresponding epoch and start to saturate. The graph depicted that it was not necessary to run the model to train for many epochs more than 10. The validation accuracy seemed reasonable which showed that the model was neither overfitted nor underfitted as shown in Fig. 5.25b. However, the model trains very fast with no significant improvement in loss or accuracy after epoch 10. The training accuracy is around 99%, while the test accuracy is around 98%. This means that the model generalizes well for the AD data that it has not seen before.



(a) (b)

FIGURE 5. 17: (a) IRDNet model loss for training and test dataset. (b) IRDNet model accuracy for training and test dataset

To study the performance of the proposed model, we used confusion matrices. A Confusion matrix is an $N_c \times N_c$ matrix used for evaluating the performance of a classification model, where N_c is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. This gives us a holistic view of how well our classification model is performing and what kinds of errors it is making.

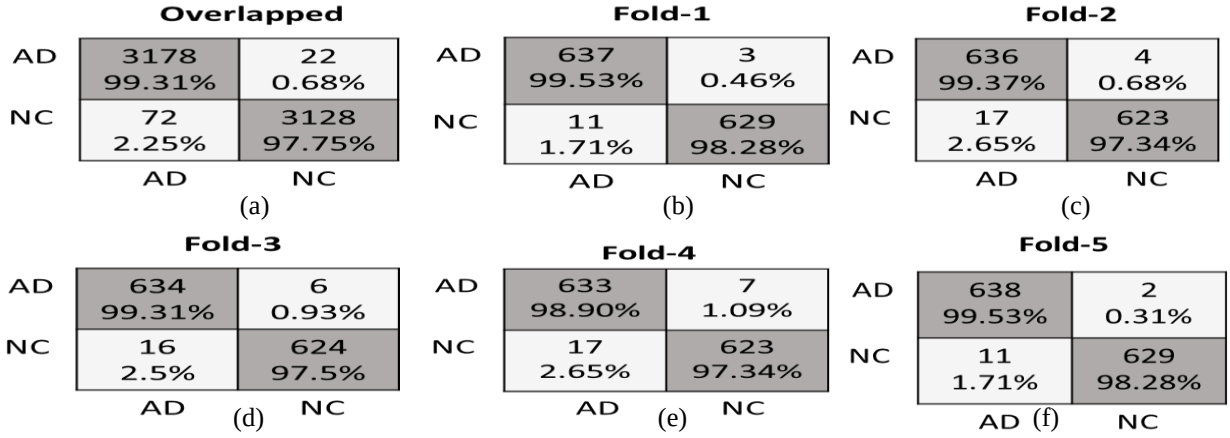


FIGURE 5. 18: Confusion matrix results of our IRDNet for 2-class Classification. (a) overlapped CM, (b) Fold-1 CM, (c) Fold-2 CM, (d) Fold-3 CM, (e) Fold-4 CM, and (f) Fold-5 CM.

It can be observed from the confusion matrix reported in Fig. 5.26 that the proposed model has detected in fold-1 637 out of 640 patients with AD as having AD and 629 out of 640 normal patients as normal patient; and it misclassified three patients with AD as normal patient and 11 normal patients as having AD. In fold-2, the proposed IRDNet has detected 636 out of 640 patients with AD as having AD and 623 out of 640 normal patients as normal patient; and it misidentified 17 normal patients as having AD and 4 patients with AD as normal patients. In fold-3, IRDNet has achieved an accuracy of 99.06%, and 97.5% to classify AD and NC, respectively. In fold-4, IRDNet has achieved an accuracy of 98.9% and 97.43% to classify AD and NC, respectively. In fold-5, our IRDNet has detected 638 out of 640 patients with AD as having AD and 629 out of 640 normal patients as normal patient; and it misidentified 11 normal patients as having AD and 2 patients with AD as NC. In overall, our model achieved an average accuracy of 99.31%and 97.75% to classify AD and NC, respectively.

Table 5. 12: Precision, F1-score and Recall values in fold 1 for NC and AD classes of the proposed IRDNet.

Class	Precision (%)	Recall (%)	F1-score (%)
AD	98.90	99.53	98.91
NC	99.53	98.28	98.90

Table 5. 13: Precision, F1-score and Recall values in fold 2 for NC and AD classes of the proposed IRDNet

Class	Precision (%)	Recall (%)	F1-score (%)
AD	97.40	99.38	98.38
NC	99.36	97.34	98.34

Table 5. 14: Precision, F1-score and Recall values in fold 3 for NC and AD classes of the proposed IRDNet.

Class	Precision (%)	Recall (%)	F1-score (%)
AD	97.54	99.06	98.29
NC	99.05	97.50	98.27

Table 5. 15: Precision, F1-score and Recall values in fold 4 for NC and AD classes of the proposed IRDNet.

Class	Precision (%)	Recall (%)	F1-score (%)
AD	97.38	98.91	98.14
NC	98.89	97.34	98.11

Table 5. 16: Precision, F1-score and Recall values in fold 5 for NC and AD classes of the proposed IRDNet.

Class	Precision (%)	Recall (%)	F1-score (%)
AD	98.31	99.69	98.99
NC	99.68	98.28	98.98

Table 5. 17: Performance of the proposed IRDNet on each fold

Folds	Precision (%)	Accuracy (%)	Recall (%)	F1-score (%)
Fold 1	99.21	98.90	98.90	98.90
Fold 2	98.29	98.35	98.36	98.36
Fold 3	98.29	98.28	98.28	98.28
Fold 4	98.18	98.12	98.12	98.12
Fold 5	98,99	98.98	98.98	98.98
Average	98,59	98.53	98.52	98.52

Table 5. 18: Average accuracy and computation time results of the models

Architectures	LeNet	AlexNet	VGGNet-16	GoogLeNet	ResNet-18	DenseNet-121	IRDNet
Average accuracy (%)	97.97	50	74.35	50	81.92	76.38	98.53
Time (s)	3,241	11,222	22,673	29,587	135,391	141,870	3,127

From table 5.12, it can be noted that our IRDNet has achieved in fold-1 a precision of 98.90% and 99.53% for AD and normal classes, respectively and a recall of 99.53% and 98.28% for AD and normal classes, respectively as well as a F1- score of 98.91% and 98.90% for AD and normal classes, respectively. From table 5.13, it can be seen that IRDNet has achieved in fold-2 a precision of 97.40% and 99.36% for AD and normal classes, respectively and a recall of 99.38% and 97.34% for AD and normal classes, respectively as well as a F1- score of 98.38% and 98.34% for AD and normal classes, respectively. From table 5.14, it can be observed that our model has achieved a precision of 97.54% and 99.05% for AD and normal classes, respectively and a recall of 99.06% and 97.50% for AD and normal classes, respectively as well as a F1- score of 98.29% and 98.27% for AD and normal classes, respectively. From table 5.15, it can be observed that our model has achieved a precision of 97.38% and 98.89%

for AD and normal classes, respectively and a recall of 98.91% and 97.34% for AD and normal classes, respectively as well as a F1- score of 98.14% and 98.11% for AD and normal classes, respectively.

From table 5.16, it can be observed that our model has achieved a precision of 98.31% and 99.68% for AD and normal classes, respectively and a recall of 99.69% and 98.28% for AD and normal classes, respectively as well as a F1-score of 98.99% and 98.98% for AD and normal classes, respectively. Therefore, the proposed IRDNet reaches an average precision, accuracy, recall and F1-score of 98,59%, 98.53, 98.52% and 98.52%, respectively for 2- class classification as shown in table 5.17.

As shown in Table 5.18, the proposed architecture works remarkably better than the others. It achieved an average accuracy of 98.53% after training it for 3,127 seconds. The LeNet model achieved the second-best performance in terms of accuracy with 97.97% after training it for 3,241 seconds. The ResNet-18 model achieved the third best performance in terms of accuracy with 81.92% after training it for 135,391 seconds. However, after training the DenseNet-121 model for 141,870 seconds, we obtained an average accuracy of 76.38%. Similarly, after 22,673 seconds of training, the VGGNet-16 model achieved an average accuracy of 74.35%. AlexNet and GoogLeNet, models achieved an average accuracy of 50% after training it for 11,222 and 29,587seconds respectively.

From these results, it can be seen that our model IRDNet is much higher and outperforms every model in terms of classification accuracy of AD vs. NC. Moreover, the accuracy of LeNet model is approximately the same as that of our IRDNet. It can be concluded that our IRDNet and LeNet detect the best features in this type of dataset. The accuracy of DenseNet-121, ResNet-18 and VGGNet-16 models is not much lower for classification of AD vs. NC. Finally, AlexNet and GoogLeNet have a worse result in terms of classification accuracy. This poor performance can be attributed to the small training size dataset that we used, which can lead overfitting and underfitting.

From this study, we have concluded that despite the different stages of Alzheimer's disease, the proposed architecture remains the most efficient and gives better results than the other models in terms of accuracy and computation time. Its efficiency lies in its structural diagram to gradually learn features and in its parallel layers that use filters of different sizes that allow extracting high-level features. Moreover, it is faster than the other models and takes 3,127 seconds to train the OASIS dataset for classification of Alzheimer's disease. Its speed resides in its architecture which uses parallel convolutional layers.

In summary, the main advantages of the proposed IRDNet are as follows:

- The model classified MR images without using manual feature extraction technique
- Our model performed remarkably well in detecting AD for two class classification problem using brain MR images.
- The model successful in classifying AD from NC with an accuracy of 98.53% after training it for 3,127.
- It is an effective model that can help clinicians in the diagnosis of AD.

In overall, the results obtained showed that our IRDNet model outperforms the studies reported in the table 5.18 in terms of computation time and accuracy for 2-class classification task (AD vs. NC). These results represent promising and encouraging performances in the detection of AD from brain MR images.

5.12. Conclusion

In this part, we proposed a new architecture of deep convolutional neural networks (IRDNet) to capture the discriminative features of brain MRI images for the diagnosis of AD in order to classify more precisely the brains affected by AD and the normal healthy brains. Our model has been trained on OASIS dataset which contain images of various AD and normal cases. Moreover, it's the classification efficiency was measured using k-fold cross-validation and was compared to other models existing in the literature. Experimental results on the OASIS dataset show that our proposed architecture succeeds in classifying AD with an average accuracy of 99.31% after training it for 3,127 seconds and an average precision, accuracy, recall and F1-score of 98.59%, 98.52, 98.52% and 98.52%, respectively for 2- class classification for two-class classification (AD vs. normal), surpassing the other CNN architectures presented in this part. These results obtained demonstrated superior and promising performance of the architecture proposed for the classification of AD and normal controls. Based on these encouraging results, the proposed IRDNet model is an interesting tool that can help clinicians to diagnose and detect AD in a very short time.

Nowadays, automatic disease detection has become a crucial issue in medical science due to rapid population growth. An automatic disease detection framework assists doctors in the diagnosis of disease and provides exact, consistent and fast results and reduces the death rate. Coronavirus has become of the most severe and acute diseases in recent times and has spread globally. As COVID-19 cases are increasing daily, many countries are facing resource shortages. Therefore, an automatic detection system, as the fastest diagnostic diagnosis, should be implemented to impede COVID-19 from spreading. This topic will be explored in the next contribution.

Part 2

Classification of COVID-19 pandemic from chest x-ray images using DCNN

Contents

5.13. Introduction 125
5.14. Related work 127
5.15. Network structure of the proposed CVDNet 129
5.16. Experimental Setup 134
5.16.1. Dataset description 134
5.16.2. Results and discussion 134
5.17. Conclusion 138

5.13. Introduction

Nowadays, we are all aware of the existence of COVID 19, known as the coronavirus. It is a contagious disease that was caused by the Severe Acute Respiratory Syndrome Coronavirus 2 (SARS-CoV-2). The disease was first discovered and became prevalent in Wuhan, Hubei Province, China, and has since spread exponentially around the world and became a serious public health problem worldwide. As we know, on March 11, 2020, the World Health Organization (WHO) proclaimed the flare-up of coronavirus pandemic [142]. As of July 12, 2020, more than 12,401,262 confirmed cases of COVID-19 and 559,047 confirmed deaths due to the disease [143]. The mortality rate is increasing alarmingly throughout the world demanding an early response to diagnose and prevent the rapid spread of this disease. Until now, Because of having no specific drugs or vaccine to combat COVID-19, the situation has become frightening to billions of individuals. The COVID-19 outbreak is considered a serious disease due to its high permeability, and contagiousness [133]. In addition, this epidemic has a great impact on the healthcare system, by virtue of the high number of patients hospitalized in intensive care units, the length of treatment, and the lack of hospital resources [47]. Currently, it is indispensable to expand effective screening strategies to distinguish COVID-19 cases and segregate the infected from others, so all infected countries are trying to enhance the capacity of the entire health care system through testing to reduce the pandemic ahead of time that is an ultimate goal in the fight against COVID-19. The most accurate screening method for new coronary pneumonia is nucleic acid detection, that is, reverse transcriptase polymerase chain reaction (RT-PCR) detection is taken from nasopharyngeal swabs, sputum and other lower respiratory tract secretions, blood, and stool samples. Although RT-PCR testing is the gold standard for COVID-19 screening, RT-PCR testing is a very complicated, time-consuming and labor-intensive process. Not all health institutions in all regions have such testing conditions. At the beginning of the outbreak, the number of RT-PCR detection reagents is very limited. In the background of

the global epidemic, some countries and regions still do not have a sufficient number of RT-PCR testing kits. Meanwhile, the limitations of RT-PCR testing have prompted researchers to find a rapid and definitive method for diagnosing COVID-19 infection. At the same time, the National Health Commissions of the People's Republic of China in the "New Coronavirus Infected Pneumonia Diagnosis and Treatment Program (Trial Fifth Edition)" [101] included suspected cases with chest imaging features into clinical diagnosis cases, that is the suspected cases of new coronary pneumonia can be diagnosed based on imaging features. The first and main imaging test for the diagnosis of new coronary pneumonia is High Resolution Computerized Tomography (HRCT) of the chest. Considering that the testing capacity and human resources of health care institutions in some regions are limited, and the cost of HRCT testing is relatively high, the radiation is relatively large, especially for patients who are suspected of being infected with new coronary pneumonia but are ultimately undiagnosed (maybe other infection like bacterial pneumonia or other virus infection). This is undoubtedly a great price for patients. Therefore, for these areas where medical resources are insufficient or the medical resources have been saturated during the epidemic, it is of great significance to use Chest X-ray (CXR) for the initial screening of the COVID-19. Chest radiography (X-ray) is one of the most important methods used for the diagnosis of pneumonia worldwide [65]. Chest X-ray is a fast, cheap [2] and common clinical method. Furthermore, the advantage of having X-rays with the scanner is that in some cases the disease can be detected in its preliminary stages, even if the screening is still negative. Fig. 5.27 shows the difference between COVID-19, viral pneumonia and normal case images.
The following findings are observed in the x-ray images of COVID-19 patients.

- Ground-glass opacities (peripheral, bilateral, subpleural, multifocal, posterior, basal and medial).
- Air space consolidation.
- Broncho vascular thickening (in the lesion).
- A crazy paving appearance. (GGOs and inter-/intra-lobular septal thickening).
- Traction bronchiectasis.

Similarly, the following findings are observed in the x-ray images of pneumonia patients

- Reticular opacity
- Ground-glass opacities (GGO) central distribution, unilateral
- Distribution more along the broncho vascular bundle
- Vascular thickening
- Bronchial wall thickening



(a) (b) (c)

FIGURE 5. 1: Example of chest X-ray images: (a) COVID-19 chest X-ray image, (b) normal chest X-ray image, (c) viral pneumonia chest X-ray image.

The chest X-ray gives the patient a lower radiation dose compared to computed tomography (CT) and magnetic resonance imaging (MRI). However, making the correct diagnosis from X-ray images requires expert knowledge and experience and it takes a long time; which is precious when people are sick around the world. By looking at the chest X-ray, COVID-19 can only be diagnosed by specialist physicians. The number of specialists who can make this diagnosis is less than the number of normal doctors. Even in normal times, the number of doctors per person is insufficient in many countries around the world. According to data from 2017, Greece ranks first with

607 doctors per 100,000 people. In other countries, this number is much lower[1]. In case of disasters such as COVID-19 pandemic, demanding health services at the same time, collapse of the health system is inevitable due to the insufficient number of hospital beds and health personnel. Also, COVID-19 is a highly contagious disease, and doctors, nurses, and caregivers are most at risk. Therefore, it is necessary to develop an automated analysis system to detect if there are anomalies in the scans in order to save time for healthcare professionals. With the advent of data science and deep learning or machine learning algorithms, we are now able to access large amounts of data, collected from different sources of information, to detect Coronavirus from computed tomography, monitoring of the epidemic, analysis of epidemiological data, and even clinical trials in record time. Technology and artificial intelligence (AI) can significantly speed up the processing of the data needed to obtain the information, responses and recommendations to manage and combat the Covid-19 pandemic, with better-than-expected accuracy. Additionally, AI approaches can be useful in eliminating disadvantages such as insufficient number of available RT-PCR test kits, test costs, and waiting time of test results [8]. A number of artificial intelligence systems based on deep learning, in particular convolutional neural networks have been proposed and the results have been very promising in terms of accuracy in the detection of patients infected with COVID-19 using chest x-ray images. The reason for this success is that deep convolutional neural networks are not based on extracting features manually, but these algorithms automatically learn features from the data itself [82]. Deep CNN has been successfully applied in many problems such as classification of skin cancer [33], arrhythmia detection [150], classification of brain diseases [127], detection of breast cancer [20], fundus image segmentation [128], detection of pulmonary pneumonia X-ray images and lung segmentation [42]. Thanks to the success of deep learning, the Canadian startup DarwinAI believes that it has developed a tool that could help doctors make crucial decisions. DarwinAI has looked into this and wondered if AI could have a role to play in the fight against the Covid-19. The answer may well prove to be positive. In this study, we review state-of-the-art models designed to identify COVID-19 infection and we propose a deep CNN model, namely CVDNet as an advanced tool to help radiologists to automatically detect COVID-19 infection using chest x-ray images and thus delay the rapid spread of this virus. The following subsection presents the literature review on recent developments in AI systems based on deep convolutional neural network for the detection of COVID-19 and describes these networks.

5.14. Related work

Motivated by the need for rapid interpretation of x-ray images, many scientists have proposed deep learning models especially convolutional neural networks to detect cases infected with COVID-19 from chest x-ray imaging. Wang et al. [137] introduced COVID-Net, a deep CNN designed for the detection of patients with Covid-19 from chest X-ray images that is an open source publicly available. The dataset consists of chest radiography images that contain four classes including COVID-19 infection, Pneumonia Viral, Pneumonia bacterial and normal (non-COVID19 infection). COVID- Net achieved an overall accuracy of 83.5% for these 4-classes and an overall accuracy of 92.4% in classifying COVID-19, normal and non-COVID pneumonia cases. Hemdan et al. [56] introduced a deep learning; called COVIDX-Net to aid radiologists to automatically detect COVID-19 in chest x-ray images. This framework is based on seven deep architectures namely MobileNetV2, VGG19, InceptionV3, DenseNet201, InceptionResNetV2, ResNetV2 and Xception. Furthermore, it is validated on 50 X-ray images comprising 25 cases with COVID-19 and 25 cases without any infections. This study reveals that The VGG19 and DenseNet models have similar performance of automated COVID-19 detection with f1-scores of 0.91 and 0.89 for COVID-19 and normal, respectively, and the InceptionV3 model produce a poor classification performance with f1-scores of 0.00 for COVID-19 cases and 0.67 for normal cases. Kumar et al. [76] proposed a system based on deep Convolutional network that is developed for the detection of COVID-19 using X-ray images. This model is trained on a dataset collected from GitHub, Kaggle and Open-I repository and achieved an accuracy of 95.38% for detecting COVID-19. Ozturk et al. [107] presented a deep CNN based on DarkNet model, namely DarkCovidNet for automatic COVID-19 identification using chest X-ray images. The DarkCovid-Net model is proposed to provide accurate diagnostics for multi class classification (COVID vs. Normal vs. Pneumonia) and binary classification (COVID vs. Normal). This model has achieved a classification accuracy of 87.02% for multi class cases and 98.08% for binary classes. Ioannis et al. [61] trained different pre- trained deep learning models on two

[1]https://tr.euronews.com.

datasets. The first one is a collection of 1427 X-ray images including 504 images of normal cases, 700 images with confirmed bacterial pneumonia and 224 images with confirmed Covid-19 cases. The second one is a dataset including 504 images of normal cases, 714 images with confirmed viral pneumonia and bacterial and 224 images with confirmed Covid-19 cases. Their model achieved an accuracy of 98.75% and 93.48% for two and three classes. Khan et al. [70] introduced CoroNet, a convolution neural network to detect COVID-19 using X-ray and CT scans. This model is based on Xception architecture and pretrained on ImageNet dataset. The experimental results show that the pretrained network provides an overall accuracy of 89.6% and 95% for 4 classes (pneumonia viral vs. COVID-19 vs. Pneumonia bacterial vs. normal) and 3 classes (normal vs. COVID-19 vs. Pneumonia). Xu et al. [148] established an early screening model to differentiate COVID-19 from Influenza-A viral pneumonia and healthy cases using 618 pulmonary CT samples (i.e., 175 healthy persons, 224 patients with Influenza-A, and 219 patients with COVID-19). This model achieves a total accuracy of 86.7%. S. Wang et al. [138] proposed a deep CNN model to classify COVID-19 from viral pneumonia using 99 Chest CT images (i.e., 55 viral pneumonia and 44 COVID-19). The results of testing dataset show an overall accuracy of 73.1%, along with a sensitivity of 74.0% and a specificity of 67.0%. L. Li et al. [89] trained A ResNet50 model (COVNet) to distinguish COVID-19 from pneumonia and non-pneumonia using 4356 chest CT images (i.e., 1735 pneumonia, 1325 non-pneumonia and 1296 COVID-19). The results show that the COVNet model provides a specificity of 96%, a sensitivity of 90%, and AUC of 0.96 in classifying COVID-19. Y. Song et al. [124] introduced a system based deep learning, called DeepPneumonia to identify patients with COVID-19 from healthy people and bacterial pneumonia patients using Chest CT images. This model achieves an overall accuracy of 86.0% for (COVID-19 vs. bacterial pneumonia) classification and an overall accuracy of 94.0% for (COVID-19 vs. healthy) classification. B. Ghoshal et al. [43] trained a Bayesian Deep Learning classifier using transfer learning method to estimate model uncertainty using COVID-19 X-Ray images. The results show that Bayesian inference improves detection ac curacy of the standard VGG16 model from 85.7% to 92.9%. Zhang et al. [154] developed a deep model to identify COVID-19 infection from X-ray images. This model is trained on a dataset comprising X-ray images from 1008 non-COVID-19 pneumonia patients and 70 COVID-19 patients and achieved sensitivity of 96.0% and specificity of 70.7% along with an AUC of 95.2%. Table 5.19 summarizes the previous works related to the detection of COVID-19 infection using Chest x-ray images

Table 5. 1: Representative works for Chest X-ray images based on the detection of COVID-19 infection

Literature	Models	Dataset	Performance
L. Wang et al.	COVID-Net pre-trained with ImageNet	5941 chest x-ray images across 2839 patient (1203 normal + 45 COVID-19 + 660 non-COVID viral pneumonia + 931 bacterial pneumonia)	Accuracy of 92.4% for 2-classes and 83.5% for 4-classes
Hemdan et al.	COVIDX-Net : based on DenseNet201, Inception v3, VGG19, MobileNet v2, Xception, InceptionResNet v2 and ResNet v2	50 X-ray images comprising 25 cases with COVID-19 and 25 cases without any infections	F1-scores of 0.89 for normal and 0.91 for COVID-19
P. Kumar et al.	Deep features from Resnet50 + SVM classifier	Dataset collected from GitHub and Kaggle comprising 25 cases with COVID-19 and 25 cases without any infections	Accuracy of 95.38%
Ozturk et al.	DarkCovidNet	X-ray images comprising 125 with COVID-19, 500 normal and 500 Pneumonia cases	Accuracy of 87.02% for 3-class cases
Ioannis et al.	VGG-19	1427 X-ray images including 504 images of normal cases, 700 images with confirmed bacterial pneumonia and 224	Accuracy of 93.48% for three classes.

		images with confirmed Covid-19 cases.	
Khan et al.	CoroNet	Images collected from Kaggle repository, comprising 290 COVID-19, 1203 normal, 931 viral Pneumonia and 660 bacterial Pneumonia chest x-ray images.	Accuracy of 89.6% and 95% for 4 and 3 classes, respectively.
X. Xu et al.	ResNet +Location Attention	618 pulmonary CT samples (i.e., 175 healthy persons, 224 patients with Influenza-A, and 219 patients with COVID-19)	Accuracy of 86.7%
S. Wang et al.	M-Inception	99 Chest CT images (i.e., 55 viral pneumonia and 44 COVID-19)	Accuracy of 73.1%, along with a sensitivity of 74.0% and a specificity of 67.0%
L. Li et al.	COVNet	4356 chest CT images (i.e., 1735 pneumonia, 1325 non-pneumonia and 1296 COVID-19).	Specificity of 96%, sensitivity of 90%, and AUC of 0.96
Y. Song et al.	DeepPneumonia	Chest CT scans of 275 patients (88 patients infected with COVID-19, 101 patients infected with bacterial pneumonia, and 86 healthy)	Accuracy of 86.0% for (COVID-19 vs. bacterial pneumonia) classification and 94.0% for (COVID-19 vs. healthy) classification
B. Ghoshal et al.	Dropweights based Bayesian Convolutional Neural Networks	5941 chest x-ray images across four classes (Bacterial Pneumonia: 2786, Normal: 1583, COVID-19: 68 and non-COVID-19 Viral Pneumonia: 1504).	Accuracy of 92,90%
J. Zhang et al.	Deep CNN based on Backbone network	X-ray images from 1008 non-COVID-19 pneumonia patients and 70 COVID-19 patients	Sensitivity of 96.0% and specificity of 70.7% along with an AUC of 95.2%.

5.15. Network structure of the proposed CVDNet

In the previous works summarized in Table 5.19, we can see that P. Kumar suggested a deep CNN that achieves the highest accuracy (95.38%). Inspired by this work which mainly focuses on the ResNet, we propose to improve this network in order to more accurately distinguish COVID-19 cases from normal and viral pneumonia cases using chest x-ray images. ResNet introduced the concept of Residual Network and used skip-connections which ignore the formation of a few layers and connects directly to the output. The idea behind this residual network is that the input x passes through a series layers which we denote by $F(x)$. This result is then added to the original input$x$: $H(x) = F(x) + x$.

The proposed designed CNN, CVDNet, as shown in Fig. 5.28 consists of two parallel columns. The two columns have the same structures except for the sizes of filters. In the following, we introduce the proposed model in detail and we explain its main components as well as the proposed block.

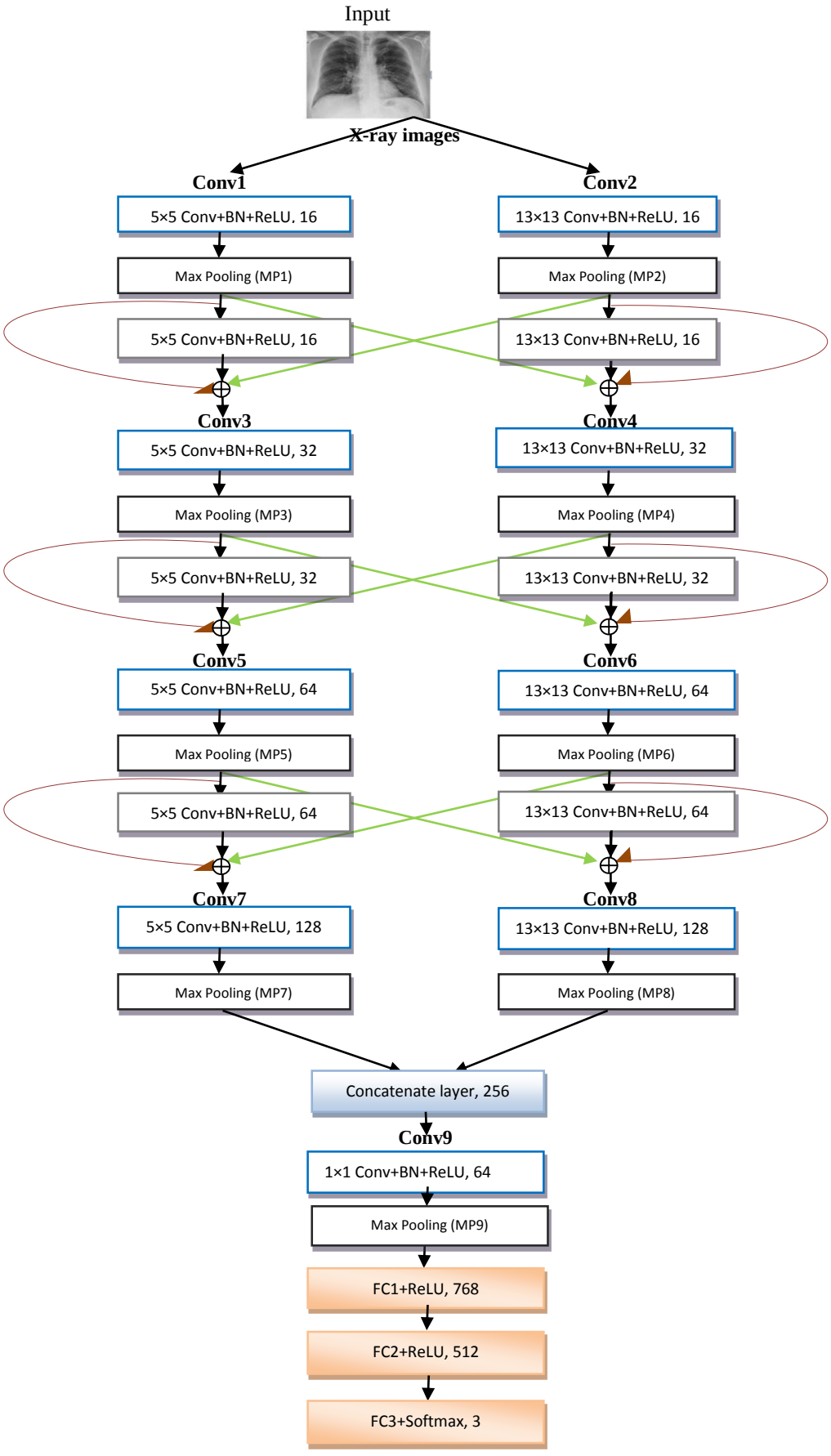


FIGURE 5. 2: The structure of the proposed CVDNet model

In chest x-ray images infected by COVID-19, Ground-glass opacities are observed, which are considered as the typical early features of COVID-19 infection. In chest x-ray images infected by COVID-19, Ground-glass opacities are observed, which is considered as the typical early features of COVID-19 infection. These features are often found in different location including in a sub-pleural and peripheral as well as they come in different sizes, shapes and quantities. For large size of Ground-glass opacities features, if using a small filter, the resulting of feature maps has few pixels containing small object. Therefore, it does not capture holistic information. This leads to poor classification accuracy. Meanwhile, for small size of Ground-glass opacities features, if using a large filter, the receptive fields are too large. Thus, most of the information captured is irrelevant. This leads to poor classification accuracy due to loss of details. Consequently, we need to use a small filter for extracting local features, and a large filter for detecting global features. However, the sizes of the features of the input image are unpredictable and random which is not practical to construct a filter size for each scale of objects. To overcome this problem, we propose to design a structure to capture multi-scale features i.e., we applied to the input two parallel convolutions with large and small filters; each level detects a scale of features. Using these two filters in parallel is more suitable for extracting different scales of features than a single scale filter. For the convolution, we use the concept of residual technique. The excellent performance of Residual network has been proven medical imaging tasks, which can solve the vanishing gradient problem. By connecting the residuals from the previous layers, this can also help to compensate the loss of details for local features or to obtain more holistic information for global features. However, to detect more details and more holistic information, we propose to merge the multi-scale features via residuals connections and we propose to link residual information not only at the same level but also at other levels. The merge of different scales of features makes it possible to identify several scales of objects. The output of the proposed block which consists of connecting a residual with all the other levels as shown in Fig. 5. 29 is defined as:

$$o_i = F_i(x_i) + \sum_{i=1}^{L} x_i, \quad i \in \{1, 2, ..., L\}$$

Where L is the total parallel levels, i is the ith level, o_i is the output of the ith level, $x_1, x_2, ..., x_L$ are the inputs of the block and $F_1(x_1), F_2(x_2), ..., F_L(x_L)$ are their corresponding outputs, respectively.

In this study, we use only two parallel levels by applying large and small filters to extract global and local features.

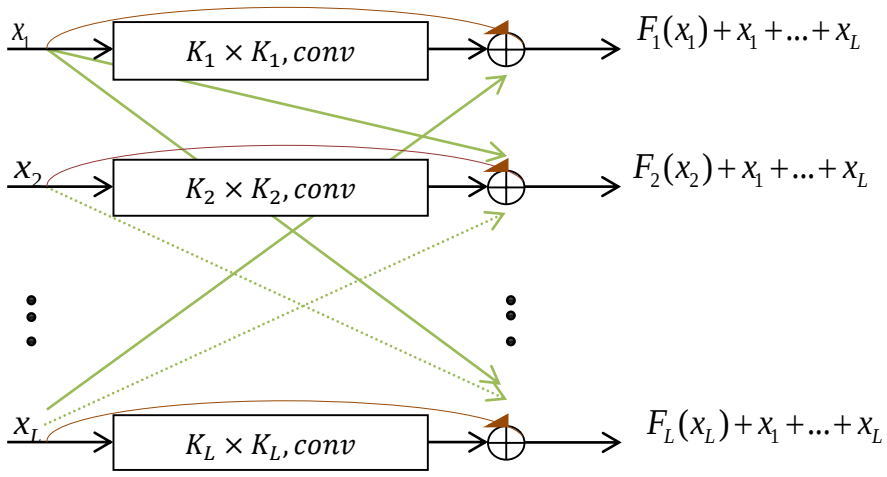


FIGURE 5. 3: Illustration of the proposed block. Inputs are connected to the output of their layers, and added to the inputs of other levels.

Based on the proposed block, we design CVDNet to classify COVID-19 infection from normal and other pneumonia cases. The proposed CVDNet model took as input chest x-ray images. Initially, the images went through a preprocessing step which consists in cropping and resizing it. Indeed, the images are resized because several letters, medical symbols and art craft are made on them as well as they are cropped because they come

from different sources and consequently their sizes vary. Therefore, the size of the input images is changed to 256-by-256-by-1. After this step, the images are processed by two parallel convolutional layers having a size of 5×5 and 13×13 with a stride of one respectively, to produce 16 feature maps for each of them, followed by the batch normalization and ReLU layers (conv1 and conv2).

The uses of conv1 and conv2 change the images dimension from 1×256×256 to16×251×251 and from 1×256×256 to 16×244×244, respectively. Features generated by conv1 and conv2 are forwarding to 2×2 max pooling layer (MP1 and MP2) with a stride of two to sub-sample the input images by reducing their dimensionality and therefore to provide a spatial invariance and to reduce the quantity of parameters and computation in the network. The dimensions of the resulting images after MP1 and MP2 layers will be reduced to 16×126×126 and 16×122×122. Moreover, the output of features maps is forwarding to block 1 (see Fig. 5.28), which contains two parallel convolutional layers with a size of 5×5 and 13×13. The output of each convolutional layer is added to the output of MP1 and MP2 in the same dimension. See Fig. 5.30 where we visually the difference between original image and MP of size 2-by-2 with stride 2 and conclude that we are not losing much information while reducing the size of the image/feature map further.

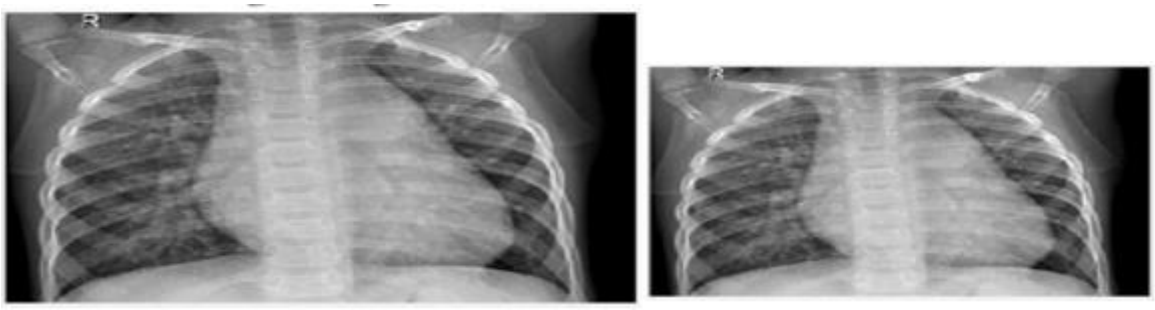
The output images dimension of this block is 16×126×126. In this block, it has no pooling layer and after each convolutional layer we normalize each batch and we apply ReLU activation function. The output of block 1 passes through 5×5 convolutional layer and 13×13 convolutional layer with a stride of 1 to generate 32 feature maps for each of them, followed by BN and ReLU function (conv3 and conv4). The uses of conv3 and conv4 change the images dimension from 16×126×126 to 32×122×122 and from 16×126×126 to 32×114×114, respectively. Features generated by conv3 and conv4 are forwarding to 2×2 max pooling layer (MP3 and MP4). The dimensions of the resulting images after MP3 and MP4 layers will be reduced to 32×61×61 and 32×57×57. The max pooling layer are forwarded to block 2 which is the same as the block 1 except it produced 64 feature maps for each convolutional layer. Next came again 5×5 and 13×13 parallel convolutional layers with a stride of 1, followed by 2×2 max pooling with a stride of 2. After this layer, block 3 is applied which is the same as the block 1 and block 2 except it produced 64 feature maps for each convolutional layer. The output of block 3 passes through 5×5 convolutional layer and 13×13 convolutional layer with a stride of 1 to generate 128 feature maps for each of them, followed by BN, ReLU function (conv7 and conv8) and max pooling (MP7 and MP8).

The output of feature maps from the two parallel layers is then concatenated and passes through 1×1 convolutional layer (conv9) with a stride of 1in order to decrease the number of filters from 256 to 64, so the output will be reduced from 256×12×12 to 64×12×12. After this, a max pooling (MP9) is applied to reduce images dimension from 64×12×12 to 64×6×6. Finally, the network had three fully connected layers; the first one has 768 feature maps each of size 1×1. Each of the 768 is connected to all the 2304 nodes (64×6×6) in the previous layer.

The second one has 512 units is connected to all the 768 nodes and finally, there is a fully connected softmax output layer with 3 possible values. The highest value determined the predicted class. In total, the network has 5,317,667 parameters.

In training phase, traditional SGD [102] is a common method used for deep neural network optimization. In recent years, it generally refers to mini-batch gradient descent. Apply this optimizer may confront two serious drawbacks: one is difficulty in the selection of suitable learning rate, and the other drawback is that it may be converged to the local optimum easily. In order to avoid drawbacks and stabilize parameters, Adaptive Moment Estimation (Adam) optimizer has been applied to optimize CVDNet objective function. This optimizer combined the advances of AdaGrad [151] and RMSprop [130], and requires less memory resource. Adam performs very well for CVDNet which process large dataset with deeper network structure. In Adam optimization process, m_t and v_t represent first Moment Estimate and Second Moment Estimate which can help to compute adaptive learning rate for different parameters.

Table 5.20 summarizes the configuration of the proposed CVDNet architecture for COVID-19 dataset.



(a) (b)

FIGURE 5. 4: Effect of stride and pooling on image resolution. (a) Original image. (b) Max Pooling, size(2,2), stride:2

Table 5. 2: The configuration of the proposed CVDNet architecture for COVID-19 dataset

Layers	Filters size	Output shape (depth × height × width)
Input image	-	1×256×256
Conv1	5×5, stride =1	16×251×251
MP 1	2×2, stride=2	16×126×126
Conv2	13×13, stride =1	16×244×244
MP 2	2×2, stride=2	16×122×122
Block 1	$\begin{bmatrix}(5\times5)\\(13\times13)\end{bmatrix}$, stride=1	16×126×126
Conv3	5×5, stride =1	32×122×122
MP 3	2×2, stride=2	32×61×61
Conv4	13×13, stride =1	32×114×114
MP 4	2×2, stride=2	32×57×57
Block 2	$\begin{bmatrix}(5\times5)\\(13\times13)\end{bmatrix}$, stride=1	32×61×61
Conv5	5×5, stride =1	64×57×122
MP 5	2×2, stride=2	64×28×28
Conv6	13×13, stride =1	64×49×49
MP 6	2×2, stride=2	64×24×24
Block 3	$\begin{bmatrix}(5\times5)\\(13\times13)\end{bmatrix}$, stride=1	64×28×28
Conv7	5×5, stride =1	128×24×24
MP 7	2×2, stride=2	128×12×12
Conv8	13×13, stride =1	128×16×16
MP 8	2×2, stride=2	128×8×8
Concatenation	-	256×12×12
Conv9	1×1, stride =1	64×12×12
MP 9	2×2, stride=2	64×6×6
Flatten	-	(1×2304)
FC1+ReLU	-	(1×768)
FC2+ReLU	-	(1×512)
FC3+Softmax	-	(1×3)

5.16. Experimental Setup

In this section, we assess by different evaluation metrics the performance of our proposed model, CVDNet, to classify patients affected with COVID-19 from viral pneumonia patients and healthy persons using chest X-ray images; as well as we describe the dataset used in this study and we detail the distribution of images to train, validation and test set.

5.16.1. Dataset Description

In this study, we used chest x-ray images obtained from Kaggle's COVID-19 Radiography Database. This dataset was created for three different types of images classified as chest x-ray images belonging to patients infected with COVID-19, chest x-ray images of cases with viral pneumonia and Chest x-ray images of healthy persons (Normal). Out of 2905 chest x-ray images, there are 219 COVID-19 images, 1345 viral pneumonia images and 1341 normal. The COVID-19 data is publicly available on Kaggle's website and it's collected from different databases: chest x-ray images with COVID-19 were taken from Italian Society of Medical and Interventional Radiology COVID1-19 Database (SIRM) and from Novel Corona Virus 2019 Dataset which is developed by Cohen et al. in Github, as well as from different recently published articles. The viral pneumonia and normal images were collected from Kaggle's Chest X-Ray pneumonia dataset [23].

5.16.2. Results and Discussion

In this section we perform experiments to assess the performance of the proposed CVDNet, which is trained to classify chest x-ray images into three categories: Normal, COVID-19 and Viral pneumonia. The efficiency of the proposed model is evaluated using the five- fold cross-validation approach for (3-class) classification problem. The dataset used was divided into 5 independent and equal sets. Four out of five sets were used to train and validate our model whereas the remaining set was employed for the test (i.e., 70% of chest X-ray images are used for training, 10% for validation and 20% for testing). More details, 219 COVID-19 chest x-ray images were used, where 158 out of 219 were randomly selected for training, 18 images for validation and 43 images for testing.
Moreover, 1341 normal chest x-ray images were used, where 965 out of 1341 were randomly selected for training set, 107 images for validation and remaining 269 images for testing as well as 1345 viral pneumonia chest x-ray images were used, where 969 out of 1345 were randomly selected for training, 107 images for validation and remaining 269 images for testing.
The strategy of split dataset was repeated five times as illustrated in Fig. 5.31 and the distribution of chest x-ray images from sub-sets were summarized in table 5.21. All experiments were implemented in TensorFlow and the proposed CVDNet was trained with mini-batches size of 8 and were run for 20 epochs. During training, the Adam optimizer is used with following parameters: $\beta_1 = 0.9, \beta_2 = 0.999,\ \varepsilon = 10^{-8}$and learning rate of 0.0001.

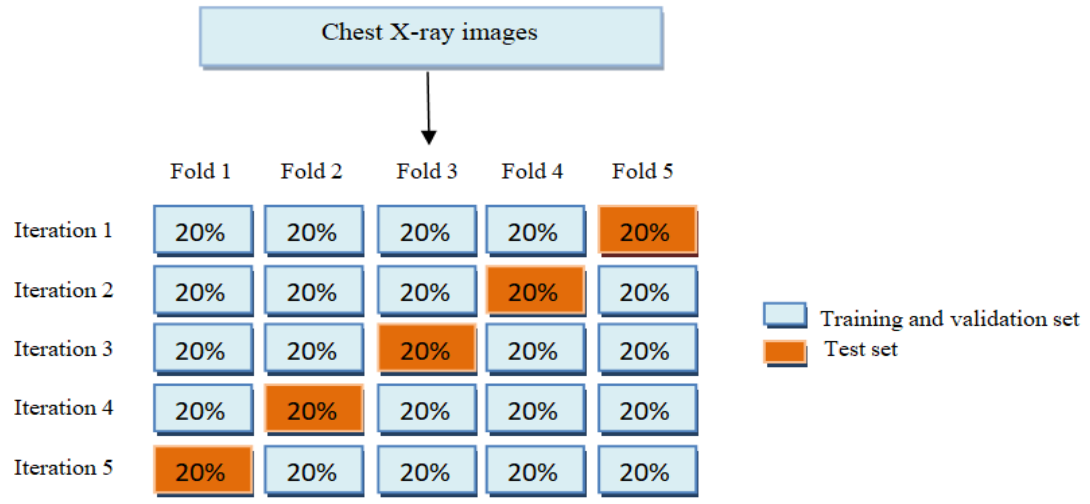


FIGURE 5. 5: Schematic diagram of training, validation and test set used in the 5-fold cross-validation

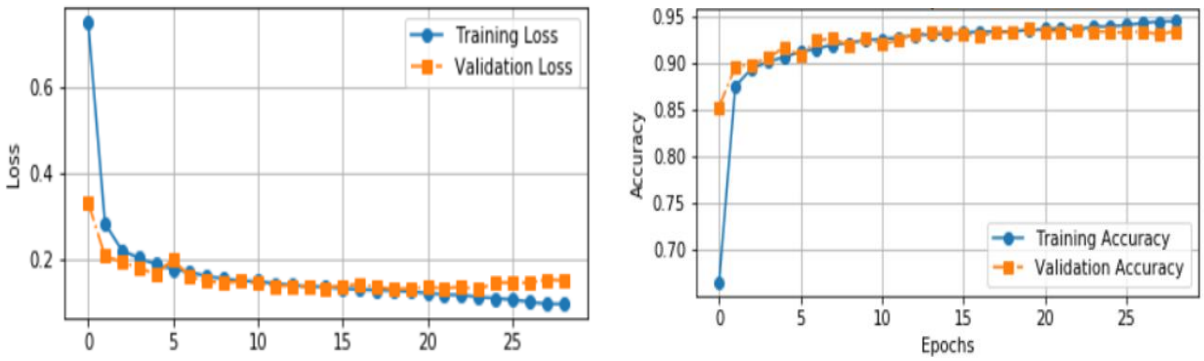
Table 5. 3: Details of training, validation and test set

Class	COVID-19	Viral pneumonia	Normal
Number of images	219	1345	1341
Training	158	969	965
Validation	18	107	107
Test	43	269	269

The performance of CVDNet for 3-class classification was measured for each fold using confusion matrix (CM) and metrics given above in part 1 such as precision, recall, accuracy, and F1-score. The overlapped and each separate CM are shown in Fig. 5.33 Furthermore, Overall precision, recall, accuracy, and F1-score computed for each class (COVID-19, Normal and viral pneumonia) as well as for each fold are presented in Tables 5.22, 5.23, 5.24, 5.25, 5.26 and 5.27. Moreover, the accuracy and loss curve of the proposed CVDNet model as it converges is display in Fig. 5.32.

In the following, we detail the performance of our model and we discuss the obtained results.

The proposed network architecture described in the above section was implemented on the COVID database to test the efficacy of its performance for the purpose of automated AD classification. From Fig. 5.32, it is observed that with increase in epochs, the loss value decreases, and accuracy increases. The proposed model seems to be good after visualizing the accuracy and loss of both training and validation. The condition of overfitting and under fitting did not occur significantly in the model. The training loss as well as validation loss was decreasing exponentially which conclude that the network is good for predicting the test image and perform the classification task. Fig. 5.32a shows the number of epochs versus Loss value. The total number of epochs was taken 30 and the training and validation loss started to decrease initially. After some epoch, the loss was constant. It means that the model stops to learn from the corresponding epoch and start to saturate. The graph depicted that it was not necessary to run the model to train for many epochs more than 20. The validation accuracy seemed reasonable which showed that the model was neither overfitted nor underfitted as shown in Fig. 5.32b. However, the model trains very fast with no significant improvement in loss or accuracy after epoch 20. The training accuracy is around 95%, while the test accuracy is around 95%. This means that the model generalizes well for the COVID data that it has not seen before.



(a) (b)

FIGURE 5. 6: (a) CVDNet model loss for training and test dataset. (b) CVDNet model accuracy for training and test dataset

To study the performance of the proposed model, we used confusion matrices. The confusion matrix is a common way to evaluate the performance of a multiclass classification algorithm, which plots the actual class label against the predicted class label. Specifically, the confusion matrix for an Nc class problem is an $Nc \times Nc$ matrix H, where the jth entry h_{ij} represents the percentage of test instances of the ith class that were predicted to belong to the jth class.

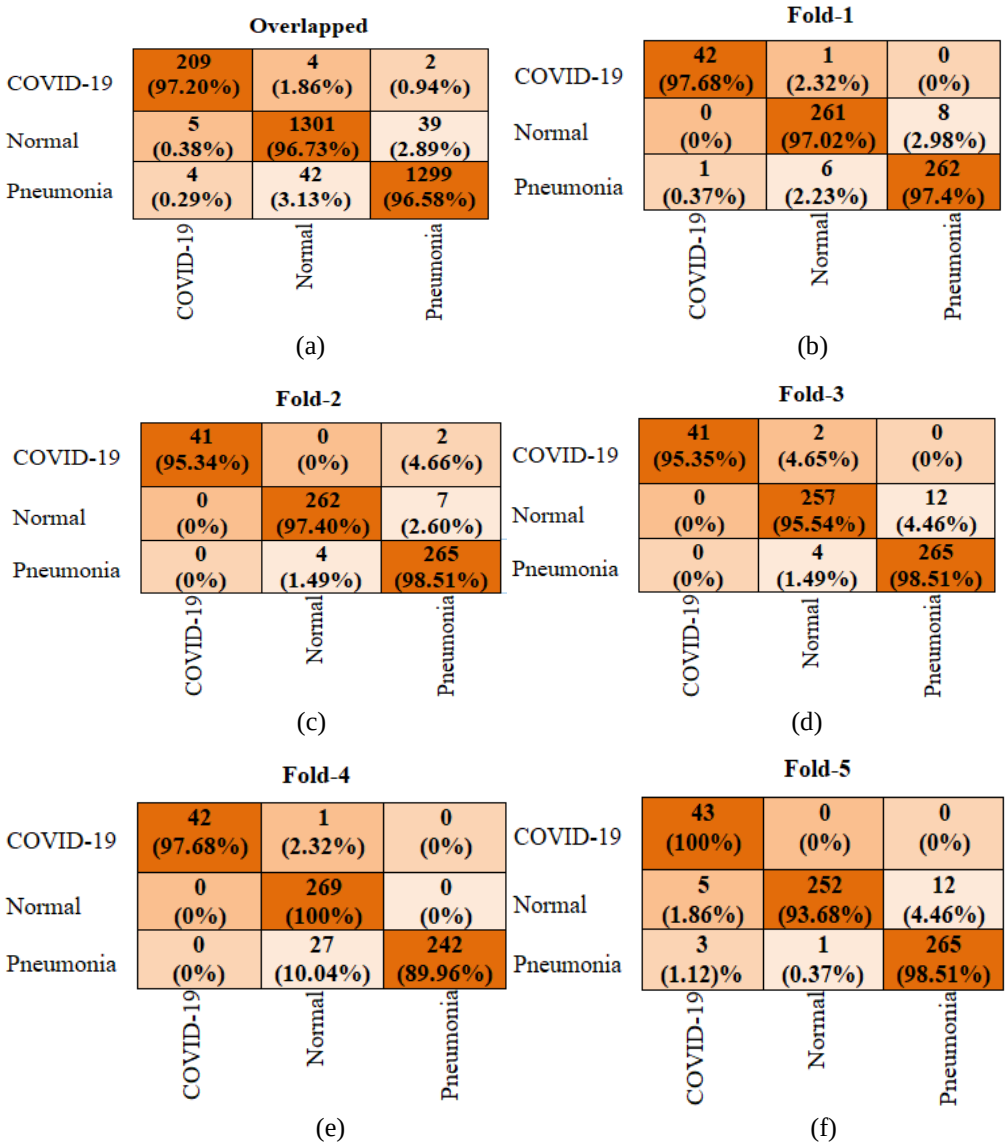


FIGURE 5. 7: Confusion matrix results of our CVDNet for 3-class Classification. (a) overlapped CM, (b) Fold-1 CM, (c) Fold-2 CM, (d) Fold-3 CM, (e) Fold-4 CM, and (f) Fold-5 CM.

It can be observed from the confusion matrix reported in Fig. 5.33 that the proposed model has detected in fold-1 42 out of 43 patients with COVID-19 as having COVID-19, 261 out of 269 normal patients as normal patient and 262 out of 269 patients with viral pneumonia as having viral pneumonia infection; and it misclassified one patient with COVID-19 as normal patient, 8 normal patients as having viral pneumonia infection, one patient with viral pneumonia infection as having COVID-19 and 6 patients with viral pneumonia infection as normal patients. In fold-2, the proposed CVDNet has detected 41 out of 43 patients with COVID-19 as having COVID-19, 262 out of 269 normal patients as normal patient and 265 out of 269 patients with viral pneumonia as having viral pneumonia infection; and it misidentified 7 normal patients as having viral pneumonia infection and 4 patients with viral pneumonia infection as normal patients. In fold-3, CVDNet has achieved an accuracy of 95.35%, 95.54% and 98.51% to classify COVID-19, normal and viral pneumonia infection, respectively. In fold-4, CVDNet has achieved an accuracy of 97.68%, 100% and 89.96% to classify COVID-19, normal and viral pneumonia infection, respectively. In fold-5, our CVDNet has detected 43 patients with COVID-19 as having COVID-19, 252 out of 269 normal patients as normal patient and 265 out of 269 patients with viral pneumonia as having viral pneumonia infection; and it misidentified 5 normal patients as having COVID-19 infection, 12 normal patients as having viral pneumonia infection, 3 patients with viral pneumonia infection as COVID-19 infection and one patient with viral

pneumonia infection as normal patients. In overall, our model achieved an average accuracy of 97.20%, 96.73% and 96.58% to classify COVID-19, normal and viral pneumonia infection, respectively.

Table 5. 4: Precision, F1-score and Recall values in fold 1 for Normal, COVID-19 and pneumonia classes of the proposed CVDNet.

Class	Precision (%)	Recall (%)	F1-score (%)
COVID-19	97.67	97.67	97.67
Normal	97.39	97.03	97.21
Pneumonia	97.04	97.40	97.22

Table 5. 5: Precision, F1-score and Recall values in fold 2 for Normal, COVID-19 and pneumonia classes of the proposed CVDNet

Class	Precision (%)	Recall (%)	F1-score (%)
COVID-19	100	95.35	97.62
Normal	98.50	97.40	97.94
Pneumonia	96.72	98.51	97.61

Table 5. 6: Precision, F1-score and Recall values in fold 3 for Normal, COVID-19 and pneumonia classes of the proposed CVDNet.

Class	Precision (%)	Recall (%)	F1-score (%)
COVID-19	100	95.35	97.62
Normal	97.72	95.54	96.62
Pneumonia	95.67	98.51	97.07

Table 5. 7: Precision, F1-score and Recall values in fold 4 for Normal, COVID-19 and pneumonia classes of the proposed CVDNet.

Class	Precision (%)	Recall (%)	F1-score (%)
COVID-19	100	97.67	98.82
Normal	90.57	100	95.05
Pneumonia	100	89.96	94.72

Table 5. 8: Precision, F1-score and Recall values in fold 5 for Normal, COVID-19 and pneumonia classes of the proposed CVDNet.

Class	Precision (%)	Recall (%)	F1-score (%)
COVID-19	84.31	100	91.49
Normal	99.60	93.68	96.55
Pneumonia	95.67	98.51	97.07

Table 5. 9: Precision, F1-score and Recall values in fold 5 for Normal, COVID-19 and pneumonia classes of the proposed CVDNet.

Folds	Precision (%)	Accuracy (%)	Recall (%)	F1-score (%)
Fold 1	97.37	97.25	97.37	97.37
Fold 2	98.40	97.76	97.09	97.72
Fold 3	97.80	96.90	96.47	97.10
Fold 4	96.86	95.18	95.88	96.20
Fold 5	93.20	96.39	97.40	95.04
Average	96.72	96.69	96.84	96.68

From table 5.22, it can be noted that our CVDNet has achieved in fold-1 a precision of 97.67%, 97.39% and 97.04% for COVID-19, Normal and pneumonia classes, respectively and a recall of 97.67%, 97.03% and 97.40% for COVID-19, Normal and pneumonia classes, respectively as well as a F1- score of 97.67%, 97.21% and 97.22% for COVID-19, Normal and pneumonia classes, respectively. From table 5.23, it can be seen that CVDNet has

achieved in fold-2 a precision of 100%, 98.50% and 96.72% for COVID-19, Normal and pneumonia classes, respectively and a recall of 95.35%, 97.40% and 98.51% for COVID-19, Normal and pneumonia classes, respectively as well as a F1- score of 97.62%, 97.94% and 97.61% for COVID-19, Normal and pneumonia classes, respectively. From table 5.24, it can be observed that our model has achieved a precision of 100%, 97.72% and 95.67% for COVID-19, Normal and pneumonia classes, respectively and a recall of 95.35%, 95.54% and 98.51% for COVID-19, Normal and pneumonia classes, respectively as well as a F1- score of 95.67%, 98.51% and 97.07% for COVID-19, Normal and pneumonia classes, respectively. From table 5.25, it can be observed that our model has achieved a precision of 84.31%, 99.60% and 95.67% for COVID-19, Normal and pneumonia classes, respectively and a recall of 100%, 93.68% and 98.51% for COVID-19, Normal and pneumonia classes, respectively as well as a F1- score of 91.49%, 96.55% and 97.07% for COVID-19, Normal and pneumonia classes, respectively. From table 5.26, it can be observed that our model has achieved a precision of 100%, 90.57% and 100% for COVID-19, Normal and pneumonia classes, respectively and a recall of 97.67%, 100% and 89.96% for COVID-19, Normal and pneumonia classes, respectively as well as a F1-score of 98.82%, 95.05% and 94.72% for COVID-19, Normal and pneumonia classes, respectively.Therefore, the proposed CVDNet reaches an average precision, accuracy, recall and F1-score of 96.72%, 96.69, 96.84% and 96.68%, respectively for 3-class classification as shown in table 5.27.

In summary, the main advantages of the proposed CVDNet are as follows:

- The model classified chest X-ray images without using manual feature extraction technique
- Our model performed remarkably well in detecting COVID-19 for three class classification problem using chest X-ray images.
- The model performed well in classifying COVID-19 class with an accuracy of 97.20%.
- It is an effective model that can assist radiologists in the diagnosis of COVID-19.
- It can be helpful in areas where the test kit is not sufficient

In overall, the results obtained showed that our CVDNet model outperforms the studies reported in the table 5.26 in terms of accuracy for 3-class classification task (COVID-19 vs. normal vs. viral pneumonia). These results represent promising and encouraging performances in the detection of COVID-19 from chest X-ray images.This will greatly assist radiologists in overcoming load on hospitals and the medical system.

5.17. Conclusion

In this part, we have proposed a novel deep convolutional neural network (CVDNet) model for the detection of COVID-19 cases, in order to distinguish more precisely the patients affected by COVID-19 from healthy persons and viral pneumonia patients using chest X-ray images. This model is based on the residual neural network and it is constructed by using multi parallel levels with different kernel sizes to detect local and global features and by connecting the residuals to other levels to share information. The CVDNet is trained on 2905 chest X-ray images that are open source and publicly available to classify COVID-19 cases from normal and viral pneumonia cases. Moreover, its classification efficiency was measured using k-fold cross-validation and the results found from the proposed model are compared to other studies reported in the literature. It was observed that CVDNet achieved an average accuracy of 97.20% for detecting COVID-19 and an average accuracy of 96.69% for three-class classification (COVID-19 vs. normal vs. viral pneumonia), which exhibit superior and promising performance in classifying COVID-19 cases. Based on these encouraging results, our CVDNet model is an interesting tool that can help clinicians to diagnose and detect COVID-19 infection in a very short time.

The proposed model has some limitations. Firstly, the sample size is relatively small that needs to be increased to test the generalizability of the developed model as is known, the success of deep learning depends largely on the number of labeled data. This would be overcome if more images are found in the coming days. Secondly, the proposed model cannot be efficiently classified COVID-19 images comprising multiple disease symptoms.

CHAPTER 6

Conclusion and Future Work

This chapter concludes the thesis. A meta-summary of the contributions and findings of Chapters 3-5 are given, and some limitations as well as potential future work following on from this thesis is summarized.

The thesis has focused on addressing key challenges in medical image analysis. The methods proposed in this thesis make use of different machine learning techniques, including deep neural network.
In medical image analysis, segmentation is a common challenge in many clinical applications. A generic segmentation method is useful as it can be used in different applications. With this in mind, we have developed an automated approach called MKPE for automatic segmentation of MRI brain tissues corrupted by noise and bias field. In chapter 3, the MKPE method has been validated on the simulated T1- weighted MR images with varying noise characteristics and intensity inhomogeneity. The experimental results show the efficacy of the proposed framework and that is consistently outperforms the models presented in this work for each tissue classification. The questions of this research are: how could machine learning techniques be used to improve the segmentation performance of brain MRIs in terms of accuracy and execution time in the presence of intensity inhomogeneities? And how can details be preserved in a noisy image segmentation? The answer to these questions is an open area of research. Many researchers are working to develop novel segmentation approaches. In this work, we have demonstrated that by modeling the bias field during the segmentation process and by applying a local filtering in each pixel before the segmentation process we can improve the segmentation performance of noisy and intensity inhomogeneity MRI brain tissue in terms of accuracy and execution time. Another question was answered regarding the initial choice of clusters in fuzzy clustering. In general, the fuzzy clustering algorithms choose the initial centers in random manner, which are affecting the results of clustering when it selects outliers as initial centers. To solve this problem, we proposed to measure the initial centers based on the idea of selecting initial centers well separated since clusters are separated groups in a feature space. However, deep learning recently has also gained great interest in multi-modal medical image segmentation. But the lack of available ground-truth data is a major issue to fully incorporate deep learning into the biomedical imaging domain. Due to this reason, we limited to the use of clustering methods. Therefore, deep learning models that are highly robust to variations in brain MRI or have unsupervised learning capability with less requirement on ground truth labels are needed. In addition, data augmentation approaches that realistically mimic variations in brain MRI data could alleviate the need of large amount of data. The proposed MKPE have been proven effective for image segmentation. However, they still have the following disadvantages: (1) although the introduction of local information to the corresponding objective function enhances their insensitiveness to noise to some extent, they still lack enough robustness to noise, especially in absence of prior knowledge of the noise; (2) in our objective function, there exists a crucial parameter used to balance between robustness to noise and effectiveness of preserving the details of the image, it is set manually based on experience, that is to say by choosing the most appropriate value by varying it; and (3) the time of segmenting an image is dependent on the image size. In the future, we will mitigate the disadvantages of MKPE and at the same time enhances the segmentation performance.
Since a segmentation based-clustering algorithm can produce as many partitions as desired, one needs to assess their quality in order to select the partition that most represents the structure in the data if there is any. This is the rationale for the cluster validity problem and indices. In chapter 4, we have proposed a new validity index V_{ECS} for the fuzzy c-means algorithm to identify the optimal number of clusters presented in dataset that contains high overlapping clusters to improve image segmentation. The proposed V_{ECS} was validated and applied to a variety of medical imaging data sets including, brain MRI and leukemia images. The experimental results show that our proposed index is more efficient and robust to recognize the appropriate number of clusters on data sets that contain overlap clusters. The results were states to answer the questions: (1) “Can the appropriate number of clusters be determined automatically? And if yes, how?” (2) How would recognize the number of clusters in a dataset that

contain overlapping clusters? In this study we seeked the answer to these questions. The optimal number of clusters is subjective and depends on the method used for measuring similarities. The proposed evaluation is intended to measure the quality of fuzzy clustering by considering geometrical properties of data and by using the properties of membership degree to measure the degree of overlap between clusters. Our method has a drawback is that it is sensitive to the value of degree of fuzziness m and it is assumed to be predefined. Developing a new strategy to select an appropriate m, based on the behavior of the index, needs further work. Also, we plan to apply the proposed method to more realistic examples. Fuzzy c-means is perhaps the most famous and widely used clustering algorithm. Several factors have been identified that may strongly affect the clustering performance of FCM, such as noise and outliers. The imperfect clustering results of FCM caused by these factors may also affect the decisions of existing CVIs, which will be investigated in future works.

In medical imaging, the advancement of computational intelligence, deep learning, and computer-aided detection has attracted much attention in image medical diagnosis. In chapter 5, the purpose of this work is to study how to use deep learning to classify chest X-ray and MRI images. In the first part, we have proposed a novel deep convolutional neural network (IRDNet) for the diagnosis of Alzheimer's disease, in order to accurately capture the main feature AD-related variations of anatomical brain structures and to classify more precisely the brains affected by AD from normal brains using MR images. Our model is evaluated and trained on brain MRI images acquired from 200 subjects, including 100 AD patients and 100 NC from the Open Access Series of Imaging Studies database (OASIS). Classification performance of the proposed IRDNet were measured using k-fold cross validation, and evaluated using accuracy, recall, F1-score, and precision metrics and compared to the state-of-the-art models. The experimental results reveal that our IRDNet represents a promising classification performance. The questions of this research are: (1) How can the DensNet model be improved to accurately diagnose Alzheimer's disease? How can be decrease computation time and parameters? (2) What are the key factors in adopting deep CNNs to learn effective feature representations for this image classification task? (3) How can learn features with different scales? (4) How can overfitting be avoided? In this study, we seeked the answer to these questions by introducing a method to reduce the excessive connections in dense block that increase the chances of overfitting the data. This method consists of dropping connections from middle layers and only retaining the nearest and farthest connections based on the observation that for a deep neural network, shallower layers can learn concrete features, whereas deeper layers can learn abstract features. Moreover, we designed a structure to capture multi-scale features i.e., we applied to the input three parallel convolutions with large and small filters; each level detects a scale of features. Using these three filters in parallel is more suitable for extracting different scales of features than a single scale filter. This study has limitations is that the machines were not powerful enough to train deep networks and the inaccessibility of a large database. Our future work focuses on the use of deep convolutional neural network in modeling the progression of Alzheimer's disease and the use of the CNN model for analyzing medical images for other neurological disorders.

Despite the promising results found using our deep models, there are still several unresolved challenges for the classification task. There remain a lot of place for improvement and several promising research trends. In Particular, we highlight a set of problems which could be improved in future work such as:

- The proposed deep model requires adjusting many hyperparameters such as the learning rate, the number and the size of filters and the number of layers in order to properly train the model. A simple modification of these parameters could lead radically different models with variable performance. A future contribution is to automatically find optimal hyperparameters.
- The deep networks had a weak point which is the need for a large amount of data to learn. Indeed deep learning, which claims to be something inspired by the brain, is still unable to learn as well as the brain. It's therefore important to improve this point in future research.

In the second part, we have proposed CVDNet, a deep learning-based model to detect and classify COVID-19 cases from X-ray images. This model is fully automated with an end-to-end structure without the need for manual feature extraction. The proposed model is developed to provide accurate diagnostics for multi-class classification (COVID vs. No-Findings vs. Pneumonia). In particular, we developed ResNet model by aggregating the multiscale information gathered by the multicolumn convolutional neural network to improve performance. The CVDNet is trained on 2905 chest x-ray images that is open source and publicly available to classify COVID-19 cases from normal and viral pneumonia cases and its classification efficiency was measured using k-fold cross-validation and evaluated using accuracy, recall, F1-score, and precision metrics as well as the results found from the proposed model are compared to other studies reported in the literature. The obtained results exhibit superior and promising

performance in classifying COVID-19 cases. The proposed system has some limitations. Firstly, the sample size is relatively small that needs to be increased to test the generalizability of the developed system. This would be overcome if more images are found in the coming days. Secondly, it only focuses on X-rays. Thirdly, COVID-19 images comprising multiple disease symptoms cannot be efficiently classified. Finally, the performance of our proposed system was not compared with radiologists. Hence, a comparison of the proposed system with radiologists would be part of a future study. Thus, in the future research direction, we will work to develop our CVDNet in order to detect COVID-19 cases from other lung diseases. Moreover, we intend to validate the proposed model by using more images coming from different hospitals and to address CT images for the detection of COVID-19 and to compare the obtained results with our CVDNet trained using X-ray images.

Appendix A

A. Résumé en français

A.1. Contexte

L'objectif de cette étude est de proposer des algorithmes et des méthodes capables de traiter et d'analyser des données issues d'images médicales à très haute résolution. Suivant cette idée, cette thèse est divisée en quatre axes principaux :

- Le premier axe de la thèse concerne le problème de la segmentation automatique non supervisée des images IRM (imagerie par résonance magnétique) cérébrales par les méthodes de 'clustering' et plus particulièrement celles basées sur la classification floue. Notre intérêt s'est penché essentiellement sur l'algorithme Fuzzy c-means (FCM) vu son efficacité remarquable parmi les méthodes de classification non supervisées. L'avantage principal de cet algorithme réside dans sa capacité à détecter et différencier les régions basées sur des données présentant des chevauchements, ou encore lorsque l'information à traiter est imprécise ou incertaine, ce qui est le cas des images IRM cérébrales. Cependant, cet algorithme a un inconvénient majeur vu le fait qu'il ne prend pas en compte le contexte spatial des pixels est donc très sensible au bruit et à l'inhomogénéité d'intensité. Plusieurs auteurs ont tenté de remédier à cet inconvénient en incorporant des informations spatiales dans les regroupements flous, ou encore en utilisant les techniques de noyau afin d'améliorer les performances de segmentation. Cependant, même si certaines méthodes fonctionnent relativement bien sur la segmentation d'image, elles restent toujours insuffisantes, vu le temps assez long nécessaire pour l'exécution cet algorithme car les centres de classes assignés initialement de manière aléatoire et les nouveaux centres mis à jour peuvent conduire les résultats à des regroupements incorrects avec des images fortement affectées par le bruit et par une forte inhomogénéité d'intensité. Afin de résoudre les problèmes cités ci-dessus, nous avons développé un algorithme approprié appelé MKPE de segmentation floue des données d'IRM. Cette approche est introduite en modifiant la fonction objective de l'algorithme FCM et en utilisant une métrique de distance induite par le noyau ce qui s'est avéré un moyen efficace d'éliminer le bruit et le biais relatif à l'intensité des données d'IRM, ce qui conduit à des résultats de segmentation beaucoup plus précis.
- Le deuxième axe de notre thèse porte sur l'évaluation de la qualité des clusters/segments produits par des algorithmes de segmentation floue et la détermination du nombre optimale de clusters. Dans ce contexte, notre objectif était de proposer un nouvel indice de validité de cluster floue pour détecter les clusters chevauchés présentés dans les IRM cérébrale et plus précisément dans les images de leucémie. Un indice de validité fournit une mesure objective d'un résultat de regroupement et sa valeur optimale est utilisée pour indiquer le meilleur choix possible du nombre de clusters dans un ensemble de données d'où la nécessité de son utilisation comme mesure des performances de clustering. La compacité et la séparation sont deux critères pour l'évaluation du regroupement. La variation des données au sein des clusters indique la compacité et l'isolement entre les clusters indique la séparation. Plusieurs méthodes de la classification floues existantes arrivent notamment à l'obtention de régions plus homogènes mais avec des faibles ou moyens degrés de chevauchements. C'est le fait qui a retenu notre attention et qui a constitué le principal défi de ce deuxième volet. L'objectif était de trouver un nouvel indice de validité de cluster floue capable de détecter des clusters présentant un fort degré de chevauchement dans les images IRM cérébrales et particulièrement les images de leucémie et défiant la majorité des autres indices existants.

- Dans le troisième axe, on s'est intéressé à la classification des images d'Alzheimer (MA). Cette maladie de plus en plus présente et avec une expansion plus rapide touche plusieurs catégories de la population. Habituellement recensé chez les personnes âgées, elle touche actuellement des personnes plus jeunes et avec des troubles cognitifs plus sévères. La maladie d'Alzheimer est actuellement irréversible, les traitements efficaces manquent dans la pratique clinique. L'évolution de l'état du patient passe par plusieurs étapes, un diagnostic précoce est donc essentiel. La détection précoce et le diagnostic précis de la maladie d'Alzheimer sont essentiels pour ralentir la progression de la maladie. Détecter la maladie d'Alzheimer est une tâche difficile et prend du temps, mais nécessite un rapport d'imagerie cérébrale et une expertise humaine. Inutile de dire que cette approche conventionnelle pour détecter la maladie d'Alzheimer est coûteuse et souvent sujette à des erreurs. La performance du diagnostic humain se dégrade en raison de la fatigue, des biais cognitifs, et parfois à l'incapacité du médecin à détecter la maladie à un stade précoce. Par conséquent, il est essentiel de développer un système d'analyse automatisé rapide, moins coûteux et plus fiable pour aider les cliniciens à diagnostiquer et détecter précocement la maladie d'Alzheimer. Les systèmes de diagnostic basés sur l'intelligence artificielle sont moins sujets aux erreurs et apportent un soutien sûr aux cliniciens dans la détection et la prise de décision. De nos jours, grâce au Deep Learning, il est possible d'analyser des données à grande échelle. Les réseaux de neurones profonds permettent la création de systèmes qui sont suffisamment puissants pour représenter n'importe quel mappage entre un ensemble donné d'entrées et un ensemble de sorties correspondantes. Notre défi dans cette troisième partie de la thèse réside dans la tache assez difficile de développer un modèle de réseaux de neurones à convolution profonde pour classifier les cerveaux affectés par la maladie d'Alzheimer et les cerveaux sains à partir des images de résonance magnétique cérébrale.
- Le quatrième axe est d'actualité et la situation épidémiologique présente nous a poussés à contribuer et essayer de trouver des solutions pour la situation sanitaire du COVID-19. Notre contribution dans ce volet porte sur la classification des images de COVID-19, ce virus considéré comme une nouvelle catégorie de maladies infectieuse pandémique causée par le coronavirus 2 du syndrome respiratoire aigu sévère (SRAS-CoV-2) entrainant des problèmes respiratoires dangereux et des troubles cardiaques pouvant causer même la mort. Cela a créé une situation préoccupante dans le monde car il n'y avait pas encore de vaccins ou de traitements spécifiques pour cela. La dangerosité d'affection et de propagation rapide de ce virus nécessitait alors comme première étape son identification rapide et l'isolement des personnes touchées pour lutter contre ce virus. La réaction en chaîne par polymérase (PCR) est la principale méthode pour détecter les cas de COVID-19 qui peuvent le détecter à partir d'échantillons respiratoires. Bien que cette technique soit la plus efficace, c'est un processus manuel, compliqué, laborieux et chronophage. De plus, il y a une pénurie importante de son approvisionnement, ce qui entraîne un retard dans les efforts de prévention des maladies. De nombreux pays du monde souffrent d'un décompte incorrect du nombre de patients atteints de COVID-19, ce qui est dû au fait que tous les patients ne sont pas testés et également a un retard dans les résultats des tests. Ces retards peuvent conduire les patients infectés à interagir avec les patients en bonne santé et à les infecter au cours du processus. En raison de la disponibilité limitée des kits de test COVID-19, nous devons dépendre de différentes mesures de détermination. Puisque le COVID-19 attaque les cellules épithéliales qui tapissent nos voies respiratoires, nous pouvons utiliser des rayons X pour étudier la force des poumons d'un patient. Le médecin utilise fréquemment des images radiographiques pour analyser la pneumonie, l'inflammation pulmonaire, les abcès et les ganglions lymphatiques hypertrophiés. Et presque dans tous les hôpitaux ont des appareils d'imagerie à rayons X; il pourrait être possible d'utiliser des rayons X pour tester le COVID-19 sans les kits de test dédiés. L'avantage d'avoir des rayons X avec le scanner est que dans certains cas la maladie peut être détectée à ses stades préliminaires, son inconvénient est qu'une analyse aux rayons X nécessite un expert en radiologie et prend beaucoup de temps. De plus, il est difficile pour les radiologues de distinguer le COVID-19 des autres pneumonies virales. Les symptômes du COVID-19 étant similaires à la pneumonie virale peuvent parfois conduire à un diagnostic erroné dans la situation actuelle, où les hôpitaux sont surchargés et travaillent le long de la

journée. Par conséquent, un diagnostic incorrect peut conduire à ce qu'une pneumonie virale non COVID soit faussement étiquetée comme hautement suspecte d'avoir un COVID-19 et retarde ainsi le traitement avec des coûts, des efforts et un risque d'exposition conséquents à des patients positifs au COVID-19. Par conséquent, le développement d'un système d'analyse automatisé est essentiel pour faire gagner un temps précieux aux professionnels de la santé. Avec l'avènement de la science des données et des algorithmes d'apprentissage en profondeur ou d'apprentissage automatique, nous sommes désormais en mesure d'accéder à de grandes quantités de données, collectées à partir de différentes sources d'informations, pour détecter COVID-19, suivi de l'épidémie, analyse de données épidémiologique, et mêmes essais cliniques en un temps record. Dans cette quatrième partie de la thèse, l'objectif était de développer un modèle de réseaux de neurones à convolution profonde pour classer les cas de COVID-19, les cas normaux et les cas de pneumonie à l'aide d'images radiographiques pulmonaires. Cette contribution a fait l'objet d'un article publié dans « *Chaos, Solitons and Fractals* » et a suscité un grand intérêt de plusieurs chercheurs dans ce domaine.

A.2. Résumé de la thèse

A.2.1. Apprentissage automatique et clustering

A.2.1.1. Apprentissage automatique

L'apprentissage automatique (ou Machine Learning en anglais) est un domaine de recherche lié aux mathématiques appliquées et à l'informatique, dont l'objectif est de permettre à une machine d'apprendre sans être explicitement programmé. C'est donc une science qui consiste à construire des algorithmes et des méthodes capables d'apprendre à partir de données avec divers objectifs tels que pouvoir classer les données, faire des prédictions ou encore identifier des structures des données. L'apprentissage automatique est utilisé dans de nombreux domaines scientifiques tels que les mathématiques, la biologie et la médecine, la physique, la chimie, la finance et le marketing et bien d'autres. On distingue généralement trois types d'applications pour l'apprentissage automatique :

- L'apprentissage non-supervisé, où les données sont fournies sans étiquettes. La machine cherche alors à détecter des structures ou à faire des groupes d'éléments similaires.
- L'apprentissage supervisé, où la machine apprend à partir d'exemples labellisés dans le but de construire un modèle permettant de faire des prédictions ou de classer des données non-labellisées.
- L'apprentissage par renforcement, où une machine placée dans un environnement dynamique va apprendre à effectuer une tâche donnée grâce à un système de récompenses et pénalités basé sur ses actions.

Cette thèse s'intéresse à l'apprentissage non-supervisé et plus particulièrement au clustering et à l'apprentissage supervisé et plus particulièrement la classification

A.2.1.2. Le clustering.

Le clustering consiste à trouver la distribution sous-jacente des exemples dans leur espace de description. Autrement dit, à partir d'une base de données non étiquetées, cette approche vise à former des groupes (ou clusters) homogènes en fonction d'une certaine notion de similarité. Les observations qui sont considérées similaires sont associées au même groupe alors celles qui sont considérées comme différentes sont associées à des groupes différents. Plus formellement, dans les problèmes de clustering, les données $D = \{(x_i); i = 1,..., N\}$ sont composées de N observations sans étiquette (ou classe), chacune décrite par plusieurs variables. On notera $x_i = \{x_i^1,..., x_i^p\}$, l'ensemble des p variables décrivant l'observation $i (i \in \{1,..., N\})$. L'objectif ici est donc de partitionner l'espace d'entré en c clusters. Chaque cluster

$C_k (k \in \{1,...,c\})$ doit être, d'une part, différent des autres clusters et d'autre part, doit contenir des observations similaires. D'une manière générale, le processus du clustering se divise en trois étapes principales : (1) La préparation des données. (2) Le choix de l'algorithme de clustering et (3) La validation et l'interprétation des résultats.

Les applications du clustering sont très nombreuses : reconnaissance d'écriture, détections de groupes d'utilisateurs ayant des caractéristiques communes, identification de structures, segmentation de données, organisation de résultats de recherches web, etc.

Traditionnellement, les différents algorithmes de clustering peuvent être classés en deux catégories :

- Clustering déterministe qui assigne les objets aux clusters selon une probabilité égale à 0 ou à 1.
- Clustering probabiliste qui assigne les objets aux clusters selon une probabilité sur un intervalle [0,1].

Les techniques de clustering déterministe supposent que chaque objet appartient à une seule classe et ne permettent pas de constatations sur la force d'appartenance d'un objet à une classe. A cause de leur nature déterministe, ces méthodes sont sensibles aux données isolées et ne sont pas performante dans le cas de classes chevauchées. Les méthodes de clustering probabilistes tentent de pallier à ce problème en introduisant une forme d'incertitude lors de la classification des objets. Ces méthodes supposent l'existence d'un chevauchement entre clusters. Chaque objet à classer appartient à tous les clusters. Une force ou un poids de son association à une classe mesure sa qualité d'adaptation à la description de ces clusters. Typiquement, ce poids correspond à une probabilité qui définit le degré d'appartenance d'un objet à une classe.

L'évaluation de la pertinence de la partition générée par le clustering est un domaine de recherche très actif. La difficulté de ce problème réside dans le fait que l'évaluation des résultats du clustering est en partie subjective. En effet, pour un même jeu de données, il existe souvent un grand nombre de partitions possibles. De plus, il est impossible de définir un critère universel permettant d'évaluer sans biais les résultats obtenus par l'ensemble des algorithmes de clustering. Pour atteindre cet objectif, de nombreuses techniques ont été développées pour identifier la « meilleure » partition générée par un algorithme de clustering. Cette identification est souvent liée à la méthode utilisée. Dans la littérature, les critères analytiques permettant de mesurer la qualité des résultats issus des algorithmes de clustering peuvent être catégorisés en trois grandes familles: interne, externe et relatif.

1. Les mesures de qualité interne se calculent uniquement à partir des informations contenues dans les données, sans avoir recours à des connaissances a priori. Ces mesures sont en général des approches non supervisées qui se basent sur des informations internes au clustering. Ces mesures se basent souvent sur la définition intuitive et la plus simple du clustering: les groupes d'instances doivent être les plus compacts possibles (i.e., la similarité intra clusters) et différents les uns des autres (i.e., la similarité inter clusters). Les mesures internes permettent donc d'évaluer la compacité et la séparabilité des clusters.
2. Les mesures de qualité externe s'appuient sur une connaissance a priori des caractéristiques d'un bon clustering. Ces mesures sont en général des approches supervisées qui consistent à mesurer le degré de correspondance entre la partition générée par l'algorithme de clustering et une partition connue des données
3. Les mesures de qualité relative permettent de comparer plusieurs partitionnements obtenus à partir d'un même jeu de données. Il s'agit tout simplement de l'utilisation des critères internes ou externes pour choisir la meilleure partition générée par le même algorithme sur le même jeu de données.

Des problèmes qui peuvent sembler assez simples comme trouver le nombre de clusters présents dans un jeu de données sont également toujours des questions ouvertes. Il n'est à vrai dire déjà pas évident de savoir si oui ou non des structures sont présentes dans un jeu de données et donc s'il y a des clusters à trouver.

Selon les besoins de l'utilisateur, le clustering peut être utilisé pour deux différentes raisons :

1. La tâche de clustering peut être inscrite comme une étape intermédiaire dans un traitement d'apprentissage: il s'agit ici de considérer le clustering comme une étape de prétraitement utilisée dans une autre tâche telle que la classification supervisée. Une description des clusters (i.e., l'interprétation des résultats générés par le clustering) n'est pas nécessaire dans cette situation. On cherche uniquement dans ce cas à obtenir l'appartenance des observations à l'un des clusters sans avoir besoin d'interpréter

les résultats issus de l'algorithme. Le point le plus important ici est de s'assurer que la qualité des résultats fournis par l'algorithme est bonne.

2. Les clusters générés par le clustering constituent un résultat final. Dans ce cas, le clustering constitue à lui seul un processus global de découverte de groupes. L'exploitation des clusters pour une application donnée passe alors par une description de ces derniers.

A.2.1.3. Le clustering appliqué à la segmentation d'images

La fouille de données appliquées à l'image est un domaine de recherche datant des années 80 et qui a de nombreuses applications. L'objectif de ce domaine est d'apprendre à des algorithmes à imiter la vision humaine de manière à pouvoir détecter et éventuellement identifier les différents éléments présents sur une image. La vision assistée par ordinateur comprend généralement plusieurs étapes: l'acquisition des données, le prétraitement des données (débruitage, encodage, etc.), l'analyse des données, et la validation des résultats.
Du point de vue de l'ordinateur, une image est généralement décrite sous la forme d'une grille de pixels, qui dans le cas le plus commun contient 3 canaux (rouge, vert et bleu) dont les valeurs peuvent varier de 0 à 255. Le processus de segmentation vise à diviser l'image en un ensemble de régions disjointes qui doivent être le plus homogènes possible. Cette homogénéité peut se mesurer selon différentes propriétés, par exemple : l'intensité, la couleur, la texture... Les tâches de segmentation et de clustering sont donc très liées et peuvent être utilisés pour définir le même problème. Dans ce cas précis de clustering, les données à classer correspondent à l'espace de caractéristiques qui représente les pixels de l'image et les clusters obtenus de la procédure de classification représentent les régions ou les segments. Cependant, la segmentation d'images peut être perçue comme un cas particulier de clustering où les données traitées possèdent une information spatiale qui leur est associées. En effet, l'allocation spatiale des pixels de l'image est une caractéristique importante. La connaissance à priori que les pixels voisins appartiennent fort probablement à la même région doit être prise en considération lors de la classification.

A.2.2. Apprentissage profond et classification

Depuis 2006, l'apprentissage profond est apparu comme une nouvelle zone de recherche de l'apprentissage automatique. Au cours des dernières années, les techniques développées dans l'apprentissage profond ont déjà eu un impact sur les travaux de traitement des signaux et de l'information, y compris les aspects de l'apprentissage automatique et l'intelligence artificielle. L'apprentissage profond est une classe de techniques d'apprentissage automatique qui modélisent avec un haut niveau d'abstraction des données grâce à des architectures multiples niveaux afin d'effectuer des tâches précises (ex. : prédiction, classification, segmentation) grâce à des exemples. Les caractéristiques et les concepts de niveau supérieur sont donc définis en termes de niveaux inférieurs, et une telle hiérarchie est appelée architecture profonde. Pour la tâche de classification, les données d'apprentissage sont étiquetées selon la classe à laquelle elles appartiennent, mais aucune autre information n'est inférée à la machine. Ainsi, l'apprentissage profond permet d'éviter que des humains ne spécifient explicitement les régions clés. Au lieu de cela, les caractéristiques les plus discriminantes et prédictives sont apprises à partir d'exemples étiquetés de façon hiérarchique, du plus abstrait au plus concret. Cette hiérarchie est d'ailleurs reflétée dans l'architecture en couche : les réseaux d'apprentissage profond sont une succession de modules simples où les représentations les plus abstraites sont calculées à la suite de termes moins abstraits. Enfin, la machine ajuste ses paramètres internes pour améliorer les prédictions en utilisant une méthode d'optimisation. Récemment, des méthodes d'apprentissage profond ont été appliquées pour la segmentation, la détection et la classification. Les avancées majeures en termes d'apprentissage machine et de vision par ordinateur, la disponibilité de plus de puissance de calcul ainsi que la disponibilité de données d'apprentissage ont motivé la transition depuis les algorithmes conventionnels vers des approches d'apprentissage profond. Les réseaux de neurones convolutifs sont inspirés du processus d'apprentissage humain et des voies de reconnaissance visuelle, à travers lesquelles l'information est traitée de manière séquentielle avec une complexité croissante: le cortex visuel primaire traite les informations de bas niveau comme les bordures, la luminosité, qui correspond au contexte. L'information est alors propagée à travers d'autres couches du cerveau, où les neurones sont plus actifs et réagissent de manière

optimale aux stimuli complexes (par exemple les visages). Les neurones déterminent ainsi le "comment" en se basant sur les caractéristiques de haut niveau et la pondération apprise par l'expérience. En termes simples, le cerveau voit d'abord des contours, puis comprend qu'il s'agit d'un nez, une bouche, des yeux, et en se basant sur l'expérience prévoit avec une haute probabilité que ce qu'il perçoit soit en fait un visage. De la même façon, les réseaux de neurones convolutifs commencent par extraire des caractéristiques brutes des images à l'aide de filtres, puis les informations sont propagées à travers différentes couches. Dans chaque couche, les neurones sont indépendants et connectés aux neurones de la couche suivante. Les neurones sont initialisés avec des pondérations et des biais affectés aléatoirement et optimisés avec l'entraînement. La sortie du réseau est un vecteur à N_c dimensions, ou le N_c correspond au nombre de classes. Les réseaux de neurones convolutifs sont le type de réseau le plus adaptés à la tâche de classification. Il n'existe cependant pas de règles quant au choix de l'architecture en termes de profondeur, de paramètres, de poids ou d'outils d'optimisation. Il est ainsi judicieux de se baser sur la littérature pour effectuer un premier triage. L'état de l'art dans la classification d'images médicale a été atteint par les réseaux LeNet, AlexNet, GoogLeNet, ResNet, DenseNet et VGG. Il était donc logique pour notre étude de commencer avec ces réseaux-là.

Actuellement, les progrès des systèmes d'imagerie numérique et le stockage d'images numériques offrent des images de très haute résolution qui peuvent être utilisées dans de nombreuses applications de plusieurs domaines comme la médecine. L'imagerie médicale est un outil fondamental et précieux dans les soins de santé modernes. L'utilisation de l'imagerie médicale a considérablement augmenté et ces images massives offrent des opportunités de diagnostic. La capacité d'analyser, de quantifier et de catégoriser automatiquement les images est une technologie habilitante importante pour un système de diagnostic assisté par ordinateur qui peut être utilisé pour aider les patients. La segmentation automatisée et la classification des maladies des images médicales ont montré de plus grands avantages cliniques et ont fourni un deuxième avis au diagnostic. L'apprentissage automatique et l'apprentissage en profondeur ont progressé rapidement dans le domaine de l'analyse d'images et ont été pleinement adoptés et établis en tant que technologies nécessaires pour faciliter les procédures cliniques par la plupart des professionnels de la santé, en particulier avec des ensembles de données plus volumineux et une plus grande puissance de calcul devenant facilement disponibles. Au cœur de ces avancées se trouve la capacité d'exploiter les représentations hiérarchiques de caractéristiques apprises uniquement à partir de données, au lieu de caractéristiques conçues à la main en fonction des connaissances spécifiques au domaine. L'apprentissage en profondeur devient rapidement l'état de l'art, conduisant à des performances améliorées dans diverses applications médicales et à résoudre des problèmes difficiles (par exemple, détection d'image, classification, segmentation). La segmentation des tissus d'IRM cérébrale basée sur l'utilisation de méthodologies d'apprentissage automatique récentes sera l'objectif de la première contribution.

A.3. Résumé des contributions scientifiques et perspectives

A.3.1. Contribution 1 : Segmentation des tissus d'IRM cérébrale

Les images par résonance magnétique peuvent être affectées par plusieurs sources de dégénérescence pendant l'acquisition et la transmission d'images, où le bruit et l'inhomogénéité d'intensité sont deux types d'artefacts couramment observés dans les données IRM. Ces artefacts posent un défi dans la segmentation d'images RM. Le bruit en IRM cérébrale peut être gaussien, ricien ou sel et poivre, et peut affecter de manière significative les performances des méthodes de classification. Le bruit gaussien est un bruit qui suit une fonction de densité de probabilité égale à celle de la distribution normale. Les sources primaires de bruit gaussien dans les images numériques ont lieu au moment de l'acquisition. Le bruit ricien est le bruit dépendant du signal qui est initié en raison de la formation d'une IRM de magnitude qui suit une distribution de Rice. Le bruit sel et poivre (Salt and Pepper) est également connu sous le nom de bruit impulsif qui est essentiellement introduit en raison d'erreurs de bits dans la transmission ou d'erreurs de conversion analogique-numérique entraînant des pixels blancs dans les régions sombres et des pixels noirs dans les régions lumineuses. Pour gérer ces problèmes, les meilleures solutions consistent soit à filtrer l'image, soit à intégrer la régularisation spatiale à l'intérieur du classificateur lui-

même. L'inhomogénéité d'intensité dans l'IRM cérébrale apparaît comme un champ non linéaire variant lentement dans les images RM, connu sous le nom de champ de biais. En raison du champ de polarisation, l'intensité des pixels d'une classe particulière à d'autres emplacements semble appartenir à différentes classes. Les méthodes de segmentation conventionnelles ne parviennent pas à classer ces classes de tissus affectées par une inhomogénéité d'intensité. Cependant, le champ de biais change d'une image à l'autre pour un patient particulier et également d'un patient à l'autre dans le même système RM. Pour gérer de tels problèmes, il est nécessaire de modéliser le champ de biais non linéaire pour contrôler l'effet de la non-linéarité afin que les propriétés du tissu puissent être conservées. Il est connu de la littérature que l'algorithme FCM s'est avéré être un outil efficace pour modéliser l'incertitude dans l'ensemble de données, réalise le regroupement et produit des résultats précis pour des images sans bruit. Cependant, le principal inconvénient de FCM est qu'il utilise la mesure de distance euclidienne pour déterminer la similitude entre les points de données et les centres de cluster. Par conséquent, cela ne permet pas de capturer correctement les structures non linéaires de l'espace des fonctionnalités. Récemment, le FCM basé sur le noyau (KFCM) semble prometteur pour le clustering de données ainsi que pour la segmentation d'image. L'idée de base de la technique basée sur le noyau est qu'elle transforme l'espace d'entrée de faible dimension en un espace de caractéristiques de grande dimension grâce à une forme de mappage non linéaire. En conséquence, les points de données non séparables linéairement dans l'espace d'entrée d'origine sont maintenant séparables linéairement dans l'espace de caractéristiques de grande dimension et produisent donc de meilleurs résultats. Le KFCM et ses variantes [145] ont été proposés dans la littérature pour corriger le champ de biais, le bruit et pour segmenter les images IRM cérébrales. De nombreux chercheurs ont incorporé des informations spatiales dans l'algorithme FCM basé sur le noyau pour segmenter les images [146], [147]. Dao-Qiang Zhang et Song Chen [15] ont proposé un algorithme FCM basé sur le noyau avec contraintes spatiales (KFCM_S) pour la segmentation d'images médicales en ajoutant le terme de pénalité spatiale au noyau basé sur FCM. Cette méthode fonctionnent bien sur la segmentation d'image, elle présente certaines insuffisances telles qu'elle ne prête aucune attention à corriger le champ de biais de l'image et elle prend beaucoup de temps pour la segmentation à cause des termes spatiaux et des termes supplémentaires. De plus, dans les algorithmes de clustering itératifs, la procédure adoptée pour choisir les centres de cluster initiaux est aléatoire. Ainsi, les centres mis à jour mènent le résultat du regroupement peut être incorrect lorsqu'il sélectionne des valeurs aberrantes comme centres initiaux.

Afin de résoudre les problèmes ci-dessus, une nouvelle fonction objective basée sur la méthode de FCM avec un algorithme d'initialisation approprié est proposé pour segmenter le tissu IRM cérébral corrompu par le bruit et par l'inhomogénéité d'intensité.

Soit $X = \{x_1, x_2, ..., x_N\}$ un ensemble de N observations à classifier en c classes où chaque observation x_j peut être représentée par un ensemble de p attributs : $x_j \in \mathbb{R}^p$. Les c classes sont représentées par un vecteur des centres des classes $V = \{v_1, v_2, ..., v_c\}$ où $v_i \in \Re^p$ est le centre de la classe i. Chaque observation x_j est définie par son degré d'appartenance u_{ij} à la classe i tel que $u_{ij} \in [0,1]$. On peut alors définir une matrice de partition $U = [u_{ij}]$ de dimension $(c \times N)$

L'algorithme FCM consiste à partitionner les N observations en c classes de manière à minimiser la similarité des observations à l'intérieur de chaque classe. Il se traduit par la minimisation de la fonction objectif suivante :

$$\text{minimize } J_m(U,V;X) = \sum_{i=1}^{c}\sum_{j=1}^{N} u_{ij}^{m} \| x_j - v_i \|^2 \quad (1)$$

Sous les contraintes

$V \in \mathbb{R}^{p \times c}, U \in M_{fc}$

Où $M_{fc} = \left\{ U \in \Re^{c \times N} \mid u_{ij} \in [0,1], \forall i, j; \sum_{i=1}^{c} u_{ij} = 1, \forall j; 0 < \sum_{j=1}^{N} u_{ij} < N, \forall i \right\}$

m : est le facteur de fuzzification ou facteur flou tel que $1 < m < \infty$

$\| x_j - v_i \|$ est la distance séparant l'observation x_j et le centre de la classe v_i.

Les partitions optimales U^* de X sont prises à partir de paires (U^*, V^*) qui sont des minimiseurs locaux de J_m. Dans le regroupement d'images, la fonction objectif de FCM est minimisée lorsque des valeurs d'appartenance élevées sont attribuées à des pixels dont les intensités sont proches du centroïde de sa classe, et une valeur d'appartenance faible est attribuée lorsque le point est éloigné du centroïde. La fonction d'appartenance représente la probabilité qu'un pixel appartienne à un cluster spécifique. Dans l'algorithme FCM, la probabilité dépend uniquement de la distance entre le pixel et chaque centre de cluster.

L'algorithme FCM a été largement étudié et a été utilisé dans nombreux domaines. Cependant cet algorithme qui nécessite la connaissance du nombre de classes, n'est pas robuste face aux bruits introduits par l'imprécision des attributs et son efficacité dépend fortement de l'étape d'initialisation des centres des classes car le processus itératif peut facilement fournir une solution localement optimale. De plus, il est basé sur la distance euclidienne lors de la mesure de similarité entre une observation et le centre d'une classe ce qui le rend utilisable que pour détecter des classes de forme sphérique. Afin d'éviter ces inconvénients donc améliorer les résultats de la classification, plusieurs modifications ont été apportées à l'algorithme standard et qui sont présentées comme des variantes du FCM. Récemment, le FCM basé sur le noyau (KFCM) semble prometteur pour le clustering de données. Cet algorithme applique la transformation non linéaire sur les données à classer afin de les projeter dans un espace de plus grande dimension dans lequel les classes deviennent linéairement séparables. L'algorithme KFCM modifie la fonction objectif du FCM via la transformation ϕ, comme suit :

$$J_m(U,V;X) = \sum_{i=1}^{c}\sum_{j=1}^{N} u_{ij}^m \left\| \phi\left(x_j\right) - \phi\left(v_i\right) \right\|^2 \tag{2}$$

La distance dans l'espace d'attributs entre les transformées d'une observation $\phi\left(x_j\right)$ et d'un centre d'une classe $\phi\left(v_i\right)$ est donnée par $\| \phi(x_j) - \phi(v_i) \|^2 = \left(\phi(x_j) - \phi(v_i)\right)^T \left(\phi(x_j) - \phi(v_i)\right)$

$$= \phi(x_j)^T \phi(x_j) - \phi(v_i)^T \phi(x_j) - \phi(x_j)^T \phi(v_i) + \phi(v_i)^T \phi(v_i)$$

Et à partir de la propriété de la fonction du noyau $\phi(x_j)^T \phi(v_i) = K(x_j, v_i)$, la fonction de distance (2) peut être réécrite comme

$$\| \phi(x_j) - \phi(v_i) \|^2 = K(x_j, x_j) - 2K(x_j, v_i) + K(v_i, v_i) \tag{3}$$

Où $i = 1, 2, ..., c$ et $j = 1, 2, ...N$

La fonction objectif de l'équation (2) devient donc

$$J_m(U,V;X) = \sum_{i=1}^{c}\sum_{j=1}^{N} u_{ij}^m \left(K(x_j, x_j) - 2K(x_j, v_i) + K(v_i, v_i) \right) \tag{4}$$

Voici des exemples de fonctions de noyau $K : X \times X \to \mathbb{R}$ couramment utilisées :

- Polynôme de degré d : $K(x_i, x_k) = (\gamma x_i^T x_k + \theta)^d, \gamma > 0, \theta \geq 0, d \in \mathbb{R}$
- Noyau de la fonction de base radiale gaussienne GRBF: $K(x_i, x_k) = e^{-\frac{\|x_i - x_k\|^2}{2\sigma^2}}, \sigma > 0$
- Laplacien: $K(x_i, x_k) = e^{-\gamma \|x_i - x_k\|}, \gamma > 0$
- Sigmoïde: $K(x_i, x_k) = \tanh(\gamma x_i^T x_k + \theta), \gamma > 0, \theta \geq 0$

où γ, σ, θ et d sont les paramètres ajustables des fonctions du noyau ci-dessus. Dans cette étude, nous avons utilisé le noyau GRBF. Lorsque nous l'adoptons, $K\left(x, x\right)$ devient 1. Alors l'équation (4) devient:

$$J_m(U,V;X)=2\sum_{i=1}^{c}\sum_{j=1}^{N}u_{ij}^{m}\left(1-K(x_j,v_i)\right) \quad (5)$$

Basé sur le concept d'apprentissage automatique avec une capacité d'apprentissage pour améliorer les performances des résultats de clustering, une distance de type exponentiel est bornée et monotone croissante. Par conséquent, il peut être robuste au bruit et aux valeurs aberrantes lorsque $1-K(x_j,v_i)=1-exp(-\| x_j-v_i \|^2/(2\sigma^2))$ est utilisé pour remplacer la distance euclidienne.

Comme σ est présentée comme une dispersion, nous utilisons la variance de l'échantillon pour estimer σ^2 avec

$$\sigma^2=\frac{\sum_{j=1}^{N}\| x_j-\bar{x}\|^2}{N} \text{ avec } \bar{x}=\sum_{j=1}^{N} x_j \Big/ N \quad (6)$$

Dans ce qui suit, nous allons introduire quelques modifications à la fonction objectif dans l'Eq. (5).

Les images cérébrales RM sont d'environ 200 coupes, qui à leur tour représentent de grandes matrices de voxels/pixels. Un ensemble de tranches d'images cérébrales RM contient environ dix millions de pixels. L'intensité des pixels est généralement codée avec 8 bits, c'est-à-dire qu'il n'y a que 256 niveaux d'intensité possibles pour chaque pixel. Pour réduire considérablement la quantité de calculs effectués lors du processus de segmentation, nous appliquons d'abord un filtrage local à chaque pixel. Désignons par ξ_j le niveau d'intensité filtré du j^{th} pixel, et nous le calculerons comme suit:

$$\xi_j=\frac{1}{1+\alpha}\left(x_j+\frac{\alpha}{N_R}\sum_{k\in N_j}x_k\right)$$

Où ξ_j est la valeur de gris du j^{th} voxel de l'image ξ qui est formée à partir de l'image d'origine et de son voisinage spatial local, x_k est la valeur de gris des pixels de voisinage centrés sur x_k et N_R désigne le nombre de pixels des pixels de voisinage. En fait, $\left(\sum_{k\in N_j}x_k\right)/N_R$ représente le filtrage moyen des pixels et le paramètre α contrôle le compromis entre l'image d'origine et son image filtrée moyenne correspondante. Quand est mis à zéro, l'algorithme est équivalent au KFCM d'origine, tandis que lorsqu'il s'approche de l'infini, l'algorithme acquiert le même effet que le KFCM d'origine sur l'image filtrée moyenne, sa valeur optimale entre 0.5 et 1.2 permet à la fois de réduire le bruit en l'image et de préserver son contenu. Lors de l'exécution de l'algorithme, la mise en cluster est effectuée sur le niveau de gris de l'image nouvellement générée. En conséquence, la fonction objectif dans l'Eq. (5) peut être modifié comme

$$J_m(U,V)=2\sum_{i=1}^{c}\sum_{j=1}^{N}u_{ij}^{m}\left(1-K(\xi_j,v_i)\right) \quad (7)$$

Dans l'image RM, le champ de biais présente une variation régulière du niveau de gris des pixels du même tissu dans le domaine spatial. Par conséquent, le signal IRM observé est modélisé comme un produit du vrai signal généré par l'anatomie sous-jacente et d'un facteur variant dans l'espace appelé champ de gain. Soit ξ l'image observée et Y la vraie image. Le modèle est le suivant:

$$\forall k\in\{1,2,...,N\} \quad (8)$$

Où y_k et ξ_k sont les intensités vraie et observée au k^{th} pixel, respectivement, n_k est le bruit et b_k est le champ de gain au k^{th} voxel, et N est le nombre total de pixels dans le volume IRM. En ignorant le terme de bruit et en appliquant la transformation logarithmique des deux côtés de l'équation (8) permet à l'artefact d'être modélisé comme un champ de biais additif. Ainsi, les données d'image MR peuvent être approximées comme

$$\tilde{\xi}_k = \tilde{y}_k + \beta_k \tag{9}$$

Où $\tilde{\xi}_k = \log \xi_k$, $\tilde{y}_k = \log y_k$ et $\beta_k = \log b_k$. De plus $B = \{\beta_k\}_{k=1}^{N}$ est le champ de gain transformé en log. Si le champ de gain est connu, il est alors relativement facile d'estimer la classe de tissu en appliquant une segmentation classique basée sur l'intensité aux données corrigées. De même, si les classes de tissus sont connues, alors nous pouvons estimer le champ de gain, mais il peut être problématique d'estimer l'un ou l'autre sans la connaissance de l'autre. Nous montrerons qu'en utilisant un algorithme itératif basé sur la logique floue, nous pouvons estimer les deux. Avant de prendre une transformation logarithmique, nous ajoutons une valeur constante à chaque pixel pour traiter les valeurs nulles En remplaçant (9) dans (7), nous avons

$$J_m(U,V,B) = 2\sum_{i=1}^{c}\sum_{j=1}^{N} u_{ij}^m \left(1 - K(\tilde{\xi}_j - \beta_j, v_i)\right) \tag{10}$$

Nous utilisons ensuite la proportion $\tau = (\tau_1, \tau_2, ..., \tau_c)$ de clusters, où τ_i présente la probabilité qu'un point de données appartienne au i^{th} cluster avec la contrainte $\sum_{i=1}^{c} \tau_i = 1$. Par conséquent, $-\ln \tau_i$ est l'information relatives à l'occurrence d'un point de données appartiennent-elles au i^{th} cluster. Ainsi, nous ajoutons le terme d'entropie $\sum_{i=1}^{c}\sum_{j=1}^{N} u_{ij}^m \ln \tau_i$ pour résumer la moyenne des informations pour l'occurrence d'un point de données appartenant au cluster correspondant sur les appartenances floues; avec τ_i est la proportion de pixels appartenant à la classe i ; $\tau_i = \dfrac{\sum_{j=1}^{N} u_{ij}^m}{\sum_{i=1}^{c}\sum_{j=1}^{N} u_{ij}^m}; i = 1, ..., c$

Ainsi, la fonction objective de MKPE est définie comme

$$J_m(U,V,B) = 2\sum_{i=1}^{c}\sum_{j=1}^{N} u_{ij}^m \left(1 - K(\tilde{\xi}_j - \beta_j, v_i)\right) - w\sum_{i=1}^{c}\sum_{j=1}^{N} u_{ij}^m \ln \tau_i \tag{11}$$

Sous les contraintes $B \in \mathbb{R}^{N\times p}$, $V \in \mathbb{R}^{c\times p}$, $U \in M_{fc}$

avec $w \geq 0$ est un paramètre pour le terme de pénalité $-\sum_{i=1}^{c}\sum_{j=1}^{N} u_{ij}^m \ln \tau_i$

La fonction objectif J_m définie par Eq. (11) peut être résolu par le théorème ci-dessous; afin de trouver la fonction d'appartenance, les bias et les prototypes, nous adoptons un algorithme d'optimisation itératif qui minimise J_m. Formellement, le problème d'optimisation se présente sous la forme $\min_{U,V,B} J_m$, Sous les contraintes $B \in \mathbb{R}^{N\times p}$ $V \in \mathbb{R}^{c\times p}$, $U \in M_{fc}$. La fonction objectif peut être minimisée en prenant les premières dérivées de J_m par rapport à u_{ij}, v_i et β_j et en les mettant à zéro, on obtient trois conditions nécessaires mais non suffisantes pour être à un extrema local.

Theorem 1. Soit $X = \{x_j,\ j = 1, \ldots, N / x_j \in \mathbb{R}^p\}$ une image à segmenter en c clusters. Les conditions nécessaires pour que J_m soit un extrema local sous la contrainte de u_{ij} sont

$$u_{ij}^{*}=\left(\sum_{k=1}^{c}\frac{\left(2\left(1-K(\tilde{\xi}_j-\beta_j,v_i)\right)-w\ln\tau_i\right)^{1/(m-1)}}{\left(2\left(1-K(\tilde{\xi}_j-\beta_j,v_k)\right)-w\ln\tau_k\right)^{1/(m-1)}}\right)^{-1};i=1,...,c;\ j=1,...,N \tag{12}$$

$$v_i^{*}=\frac{\sum_{j=1}^{N}u_{ij}^{m}\left(K(\tilde{\xi}_j-\beta_j,v_i)\right).(\tilde{\xi}_j-\beta_j)}{\sum_{j=1}^{N}u_{ij}^{m}\left(K((\tilde{\xi}_j-\beta_j,v_i)\right)};i=1,...,c \tag{13}$$

$$\beta_j^{*}=\tilde{\xi}_j-\frac{\sum_{i=1}^{c}u_{ij}^{m}\left(K(\tilde{\xi}_j-\beta_j,v_i)\right).v_i}{\sum_{i=1}^{c}u_{ij}^{m}\left(K((\tilde{\xi}_j-\beta_j,v_i)\right)};\ j=1,...,N \tag{14}$$

Dans les algorithmes de clustering itératifs, la procédure adoptée pour choisir les centres de cluster initiaux est extrêmement importante car elle a un impact direct sur la formation des clusters finaux. Si nous ne le choisissons pas correctement, le résultat de l'algorithme de clustering sera instable. En général, les algorithmes de clustering choisissent les centres initiaux de manière aléatoire, ce qui affecte les résultats du clustering lorsqu'il sélectionne des valeurs aberrantes comme centres initiaux. Pour résoudre ce problème, nous proposons de mesurer les centres initiaux à partir de l'algorithme d'initialisation des centres suivant.

Étant donné que les clusters sont des groupes séparés dans un espace de caractéristiques, il est souhaitable de sélectionner des centres initiaux bien séparés. L'algorithme proposé divise les données en groupes, sur la base de l'observation que certains des motifs sont très similaires les uns aux autres. L'algorithme Calcule la distance entre les éléments au sein de chaque groupe et calcule la valeur moyenne des éléments sélectionnés comme mesure maximale de chaque groupe. En calculant la moyenne de la distance maximale entre le point dans chaque groupe, les valeurs aberrantes sont éliminées.

Algorithme 1: Initialisation des centres de clusters

Étape 1: Organisez les m_j dans l'ordre croissant où $m_j=(1/\ p)\sum_{s=1}^{p}\tilde{\xi}_{js},\ j=1,2,...,N$ pour les données $\tilde{\xi}=\{\tilde{\xi}_1,\tilde{\xi}_2,...,\tilde{\xi}_N\}$ à p dimensions.

Étape 2: Réorganisez la matrice de données comme $\tilde{\xi}'=\{\tilde{\xi}'_1,\tilde{\xi}'_2,...,\tilde{\xi}'_N\}$. Partitionnez les données en $r\ \left(r=\lfloor N/c\rfloor\right)$ groupes, où $c\ (1<c<N)$ est le nombre de clusters, de sorte que, le $(i-1)^{th}$ groupe contienne les $(i-1)^{th}\ r$ éléments des données de $\tilde{\xi}'$, et le groupe c^{th} contienne tous les éléments restants, où $i=2,...,c$.

Étape 3: Calcul de la distance entre les éléments dans chaque groupe

Étape 4: Sélectionnez la distance maximale de chaque groupe. Trouvez la valeur moyenne des éléments $\tilde{\xi}'_j$ et $\tilde{\xi}'_k$ qui sont de distance maximale dans le i^{th} groupe et affectez-la comme centre de cluster, $i=1,2,...,c$.

L'algorithme MKPE proposé décrit ci-dessous est réalisé de manière itérative en différentes étapes. Le processus commence par définir les paramètres m, ε et c et par initialiser aléatoirement la matrice de partition flou U de

telle que sorte que $\sum_{i=1}^{c} u_{ij} = 1, 1 \le j \le N$ et $u_{ij} \in [0,1]$ pour $i = 1,...,c$ et $j = 1,\ldots,N,$ également par initialiser les biais à des valeurs nulles et par attribuer les centres de cluster initiaux en utilisant l'initialisation de l'algorithme de centre de cluster. Jusqu'à la convergence, la matrice de partition floue U , le biais et les centres de clusters sont mis à jour. Le processus s'arrête lorsque le critère J_m se stabilise. Autrement dit, le critère J_m de deux itérations successives atteint une valeur stationnaire. Lorsque l'algorithme a convergé, le processus de défuzzification convertit la partition floue U en une partition dure. Une grande variété de méthodes ont été proposées pour défuzzifier la matrice U , parmi lesquelles la procédure d'appartenance maximale qui attribue le pixel j à la classe c avec l'appartenance la plus élevée est la plus importante.

Algorithm 2: Clustering flou MKPE

Entrée: ensemble de données X (image), nombre de clusters c , degré de flou m et un seuil de convergence ε
Sortie: la matrice d'appartenance floue U , matrice des centres de cluster V et le biais B

1: 1: Initialiser aléatoirement la matrice d'appartenance $U^{(0)}$ de telle sorte que $\sum_{i=1}^{c} u_{ij} = 1, 1 \le j \le N$ et $u_{ij} \in [0,1]$ pour $i = 1,...,c$ et $j = 1,\ldots,N$

2: Attribuez les centres de cluster initiaux $V^{(0)}$ à l'aide de l'initialisation de l'algorithme de centre de cluster et initialisez $B^{(0)}$ à zéro.

3: $t \leftarrow 1$

4: $J_m^{(t)} \leftarrow 0$, le stocker comme $J_{old}^{(t)}$

5: Calculez $J_m^{(t)}$ à l'aide de l'Eq. (11), stocker-le sous $J_{new}^{(t)}$

6: *Tant que* $\| J_{new}^{(t)} - J_{old}^{(t+1)} \| > \varepsilon$ faire

7: Mettre à jour la matrice du prototype $V^{(t)}$ à l'aide de l'équation (13).

8: Mettre à jour $B^{(t)}$ à l'aide de l'équation (14)

9: Mettre à jour la matrice du degré d'appartenance à l'aide de l'équation (12)

10: $J_{old}^{(t)} \leftarrow J_{new}^{(t)}$

11: Calculate $J_m^{(t)}$ using Eq. (11), store it as $J_{new}^{(t)}$

12: Calculez $J_m^{(t)}$ à l'aide de l'équation (11), stockez-le sous $J_{new}^{(t)}$

13: $t \leftarrow t+1$

14: *Fin Tant que*

15: $U^* \leftarrow U^{(t)}$, $V^* \leftarrow V^{(t)}$ et $B^* \leftarrow B^{(t)}$

16: Retourner les matrices U^*, V^* et B^*

17: Defuzzification basée sur la procédure de la valeur d'appartenance maximale.

Dans cette étude, nous avons évalué par différentes mesures la performance de notre méthode proposée, MKPE, pour segmenter les tissus à partir d'images IRM cérébrales normales simulées corrompues par différents niveaux de bruit et inhomogénéité d'intensité. Les résultats expérimentaux montrent que le MKPE est robuste au bruit et de l'intensité d'inhomogénéité et peut extraire avec précision les tissus cérébraux: MW, GM et CSF, ainsi que sa vitesse de calcul est rapide par rapport aux six méthodes récentes présentées dans la section 3.4.2.
Un problème majeur dans l'application des méthodes de clustering existantes est que le nombre de clusters doit être pré-spécifié avant que le clustering ne soit effectué. Le nombre de clusters c est plus identifié par les utilisateurs sur la base d'informations préalables. Les résultats du clustering peuvent largement dépendre du

nombre de clusters qui doit toujours être spécifié pour effectuer une analyse de clustering. Le regroupement avec des valeurs *c* différentes produira éventuellement des résultats différents. Étant donné que ce nombre est rarement connu auparavant, il est nécessaire de le déterminer afin d'obtenir des résultats de regroupement appropriés. Cependant, pour des classes bien séparées, les algorithmes de clustering trouvent généralement le même nombre de clusters. Le problème se pose dans le cas des clusters qui se chevauchement: rares sont les algorithmes qui détectent le nombre réel de classes, et ils deviennent invalides pour un degré de chevauchement relativement élevé. La découverte de ce nombre sera l'objectif de la deuxième contribution.

A.3.2. Contribution 2 : Nouvel indice de validité des clusters

Les partitions floues fournies par les algorithmes de clustering flou tentent d'identifier la structure présente dans l'ensemble de données d'entrée. Dans ces partitions, les objets affectés aux mêmes clusters sont plus similaires les uns aux autres que les objets appartenant à des clusters différents. Cependant, Les résultats de partitionnement obtenus dépendent fortement du nombre de classes fixé. Il est donc primordial de choisir le nombre exact de classes pour espérer avoir une bonne qualité de partition. Ce problème nécessite une analyse de validité de cluster utilisant une sorte de critères de validité. Les premiers indices de validité pour le clustering flou, le coefficient de partition et l'entropie de partition, ont été introduits par Bezdek [10]. Ces indices présentent deux inconvénients, présentant une tendance à diminuer ou à augmenter à mesure que le nombre de grappes augmente et étant sensibles au fuzzifier. Récemment, Askari et al. [3] a proposé un indice de validité appelé V_{CVI_2} qui est défini comme la différence totale entre les degrés d'appartenance. Cet indice utilise uniquement les degrés d'appartenance qui augmentent leur dépendance au nombre de clusters c. Il souffre de l'absence d'informations sur les propriétés géométriques des données. Cependant, la plupart des indices de validation proposés au cours des dernières décennies se sont concentrés sur deux propriétés: la compacité et la séparation. La compacité est utilisée comme mesure de la dispersion des données au sein d'un cluster, et la séparation est utilisée pour tenir compte des informations structurelles inter-cluster. L'objectif de base de l'indice de validité de cluster est de trouver le numéro de cluster qui répond à la compacité minimale et à la séparation maximale. Cependant, les indices classiques sont limités dans leur capacité à calculer la compacité et la séparation, ce qui conduit à un résultat de validation incorrect. Bien que certains indices prennent en compte les ensembles de données, ceux-ci ignorent le chevauchement entre les clusters. En général, une distance géométrique basée uniquement sur les informations du centroïde est insuffisante pour fournir une interprétation significative de la séparation des grappes (Kim et al. [161]), c'est-à-dire que les mesures de séparation conventionnelles qui ne prennent pas en compte la forme globale de clusters ne donne que des informations limitées sur les structures géométriques des clusters et ignore le chevauchement entre les clusters.

Afin de surmonter ces problèmes, nous proposons un nouvel indice de validité pour l'algorithme de FCM qui combine deux fonctions. La première fonction, notée, SC calcule le rapport entre la compacité et la séparation floue en considérant les propriétés géométriques et le degré d'appartenance de données. La deuxième fonction, appelée, PEC l'évalue en utilisant uniquement les propriétés des degrés d'appartenance. Cette évaluation, qui visait à mesurer la qualité du clustering flou, attribue un nombre réel aux sorties de l'algorithme de clustering flou utilisé. Deux types de sorties sont considérés: les degrés d'appartenance floue et la structure de l'ensemble de données. Pour chaque sortie, une combinaison de critères de compacité floue et de séparation floue est évaluée. Ainsi, une validation fiable doit avoir une faible valeur de compacité, une valeur de séparation élevée et un faible degré de chevauchement.

1. Définition de SC

Nous introduisons quelques définitions qui sont utilisées pour définir SC

Définition 1. L'écart flou des objets x_j par rapport au cluster v_i est défini comme

$$d_{ij} = \left(u_{ij}\right)^{m/2} \left\| x_j - v_i \right\|$$

Définition 2. La variation floue du cluster i est appelée σ_i définie comme

$$\sigma_i = \frac{1}{N}\sum_{j=1}^{N}(d_{ij})^2 = \frac{1}{N}\sum_{j=1}^{N}(u_{ij})^m \left\|x_j - v_i\right\|^2$$

Ensuite, la compacité globale de la c-partition est

$$comp = \sum_{i=1}^{c}\sigma_i$$

Définition 3. Soit $U \in M_{fc}$ la matrice de partition floue de l'ensemble de données X . La compacité floue intra-classe $comp : M_{fc} \times \mathbb{R}^{c\times p} \to \mathbb{R}^+$ est définie comme

$$comp(U; X, c) = \frac{1}{N}\sum_{i=1}^{c}\sum_{j=1}^{N}u_{ij}^m \left\|x_j - v_i\right\|^2 \tag{15}$$

La compacité floue intra-classe est définie comme la somme pondérée de la distance moyenne entre les éléments d'un jeu de données et ses centres de gravité. Cette fonction combine le flou in U avec la compacité géométrique de la représentation de X via les c-prototypes. Par conséquent, une partition optimale doit avoir la compacité maximale afin que les clusters soient éloignés les uns des autres.

Définition 4. La séparation floue inter-classe est définie comme:

$$sep(V; c, X) = \frac{1}{c}\sum_{i=1}^{c}\left\|v_i - \bar{v}\right\|^2 \tag{16}$$

La fonction de séparation interclasse mesure la distance entre le centre commun de tous les clusters et chaque centre de cluster. Ainsi, une valeur plus élevée de cette mesure indique que les grappes sont bien séparées. Ainsi, le rapport compacité floue/séparation floue est défini comme le rapport $comp / sep$. Explicitement, SC peut s'écrire

$$SC(U, V; c, X) = \frac{\sum_{i=1}^{c}\sum_{j=1}^{N}u_{ij}^m \left\|x_j - v_i\right\|^2 / N}{\sum_{i=1}^{c}\left\|v_i - \bar{v}\right\|^2 / c}$$

Une petite valeur de SC indique que la c-partition floue est caractérisée par des clusters flous bien séparés et compacts, c'est-à-dire une petite valeur de $comp$ et une grande pour sep .

2. Définition de PEC

La deuxième fonction PEC utilise la matrice de partition floue $U = [u_{ij}]$, obtenue par la sortie du FCM, comme résultat du clustering. Cette matrice indique également la structure détectée de l'ensemble de données. Il se compose de c lignes et N colonnes. Les lignes u_i correspondent aux clusters flous obtenus, considérés comme des ensembles flous de X .

Définition 5. Soit $U \in M_{fcp}$ une matrice de partition floue de N objets de données. La mesure de coefficient de partition $PC : M_{fc} \to \mathbb{R}^+$ est donné par:

$$PC(U; c) = \frac{1}{N}\sum_{i=1}^{c}\sum_{j=1}^{N}u_{ij}^m \tag{17}$$

La fonction PC est définie par la somme moyenne des degrés de flou des éléments d'un ensemble de données à chaque cluster: elle évalue la quantité et le degré de chevauchement entre les clusters. Cette fonction a vérifié

que, pour tous $U \in M_{fcp}$; $1/c \le PC(U;c) \le 1; PC(U;c) = 1/c \Leftrightarrow U = [(1/c)]$, c'est-à-dire $u_{ij} = 1/c, \forall i, j$; et $PC(U;c) = 1 \Leftrightarrow U \in M_{hcp}$. La fonction PC prend son maximum sur chaque partition nette et il atteint le minimum unique au centre de gravité $U = [(1/c)]$ de M_{fcp}. Plus la valeur de la fonction PC tend vers 1, plus la partition contient des clusters bien séparés, et si elle atteint la valeur minimale 1/c, l'ensemble de données ou la partition en question n'a pas de structure de cluster. Par conséquent, plus le flou d'une partition est faible, plus la valeur de la fonction PC est grande.

Définition 6. Soit A l'ensemble flou. La mesure de l'entropie floue est définie comme suit.

$$H(A) = \frac{1}{(1-m)} \sum_{j=1}^{N} \log\left(u_A^m(x_j) + (1-u_A(x_j))^m\right) \tag{18}$$

Où m≠1, m>0.

La moyenne de l'entropie floue qui est définie par l'équation. (18) est donné comme suit

$$E(U;c) = \frac{1}{N.(1-m)} \sum_{i=1}^{c} \sum_{j=1}^{N} \log\left(u_{ij}^m + (1-u_{ij})^m\right) \tag{19}$$

L'entropie floue est utilisée pour mesurer le flou de la classe, à savoir la qualité des informations qu'il contient. Un cluster avec une faible entropie floue a un faible flou. C'est donc compact. De plus, un cluster avec une valeur élevée d'entropie floue a un flou élevé et une faible compacité. Enfin, le rapport compacité/séparation floue est défini comme le rapport E / PC et PEC s'écrit explicitement comme

$$PEC(U;c) = \frac{E(U;c)}{PC(U;c)}$$

Nous pouvons voir que les clusters flous compacts et bien séparés correspondent à une valeur faible de PEC, c'est-à-dire une petite valeur pour E et une grande valeur pour PC. Les valeurs des fonctions SC et PEC changent dans une plage différente. Pour surmonter l'influence dominante de ces fonctions, nous les normalisons. Enfin, la formulation de notre indice de validité V_{ECS} est donnée par

$$V_{ECS}(U,V;c,X) = SC^N(U,V;c,X) + PEC^N(U;c) \tag{20}$$

où $SC^N(U,V;c,X)$ et $PEC^N(U;c)$ représentent respectivement la valeur de normalisation de $SC(U;c)$ et $PEC(U;c)$. Ainsi, les valeurs normalisées de ces mesures sont calculées comme

$$SC^N(U,V;c,X) = \frac{SC(U,V;c,X) - SC_{\min}}{SC_{\max} - SC_{\min}} \tag{21}$$

$$PEC^N(U;c) = \frac{PEC(U;c) - PEC_{\min}}{PEC_{\max} - PEC_{\min}} \tag{22}$$

La valeur minimale des mesures SC et PEC est donnée comme

$$SC_{\min} = \min_{c_{\min} \le c \le c_{\max}} SC(U;c)$$

$$PEC_{\min} = \min_{c_{\min} \le c \le c_{\max}} PEC(U;c)$$

La fonction V_{ECS} mesure le degré de correspondance entre la structure de l'ensemble de données d'entrée (évalué avec SC) et la partition c floue résultant de l'algorithme de clustering flou (évalué avec PEC). Le minimum de V_{ECS}, pour un nombre donné de clusters c, est excepté pour détecter les c-partitions floue. Cela tend à indiquer une bonne cohésion au sein des clusters et un petit chevauchement entre les paires de clusters. C'est la partition la

plus souhaitable de X . Le processus de calcul de notre V_{ECS} proposé pour valider la partition floue obtenue par l'algorithme de clustering FCM est donné par l'algorithme suivant. Ce processus correspond à l'exécution de l'algorithme de clustering FCM, en faisant varier le nombre de clusters entre c_{min} (le nombre minimum de clusters) et c_{max} (le nombre maximum de clusters). Par conséquent, les partitions floues optimales correspondent aux valeurs minimales de $V_{ECS}(U,V;c,X)$.

Algorithme 3: Évaluation de l'indice de validité $V_{ECS}(c)$ proposé

Entrée: X : l'ensemble de données, m : le degré flou, ε : erreur de convergence, T : le nombre maximum d'itérations, $c_{\min}$ et $c_{\max}$ le nombre minimum et maximum de clusters.

Sortie: Les valeurs de notre nouvel indice de validité $V_{ECS}(c)$.

1: Initialisez $c = c_{\min}; c_{\min} = 2$

2: Si $c \le c_{\max}$ alors

3: Initialisez les prototypes de clusters et **c**hoisissez une matrice de partition floue initiale $U^{(0)} \in M_{fc}$. Puis à l'étape $t = 1,2,...,T$

4: Mettre à jour les prototypes de cluster $v_i^{(t)}, i = 1,...,c$ et la matrice de partition floue $U^{(t)}$ pour $i = 1,...,c$ et $j = 1,...,N$

5: Comparez $U^{(t+1)}$ et $U^{(t)}$. Si $\| U^{(t)} - U^{(t+1)} \| < \varepsilon$, arrêtez; sinon, définissez $t = t+1$ et passez à l'étape 4.

6: Calculez la valeur de la mesure d'entropie $E(U;c)$ dans l'équation. (19), le coefficient de partition $PC(U;c)$ dans l'équation (17), la mesure de compacité $comp(U;X,c)$ à l'aide de l'équation (15) et la mesure de séparation $sep(V;c,X)$ à l'aide de l'équation. (16)

7: $c \leftarrow c+1$

8: Revenir à l'étape 2

9: Sinon passez à l'étape 11

10: Fin si

11: Calculez les mesures normalisées de séparation et de compacité $SC^N(U,V;c,X)$ à l'aide de l'équation. (21) et les mesures normalisées d'entropie et de coefficient de partition $PEC^N(U;c)$ à l'aide de l'équation. (22) pour toutes les valeurs de c variant entre et $c_{\min} = 2$ and $c_{\max}$.

12: Calculez l'indice de validité $V_{ECS}(U,V;c,X)$ proposé à l'aide de l'équation (20) pour toutes les valeurs de c

13: Sélectionnez le nombre optimal de clusters c^* qui minimise V_{ECS} . $c^* = \arg\min_{c_{\min},...,c_{\max}}(V_{ECS}(U;c))$

La fiabilité de notre indice de validité a été comparée aux indices de validité récents. De plus, il a été validé et appliqué à une variété d'ensembles de données d'imagerie médicale, y compris des images d'IRM cérébrale et de leucémie. Les résultats expérimentaux montrent que notre indice proposé est plus efficace et robuste pour reconnaître le nombre approprié de grappes sur des ensembles de données qui contiennent des grappes de chevauchement. Dans l'ensemble, l'indice proposé a donné des résultats satisfaisants qui étaient plus cohérents avec la segmentation des images médicales.

La classification des images médicales est l'un des problèmes les plus importants dans le domaine de la reconnaissance d'images, et son objectif est de classer les images médicales en différentes catégories pour aider les médecins à diagnostiquer des maladies ou à approfondir leurs recherches. L'utilisation des méthodes traditionnelles d'apprentissage automatique dans la classification des images médicales a commencé il y a longtemps. Cependant, ces méthodes présentent de nombreux inconvénients. Les méthodes d'apprentissage en profondeur, en particulier les réseaux de neurones convolutifs, offrent un moyen efficace de construire un modèle de bout en bout capable de classer les images médicales. Par conséquent, la classification des images médicales basée sur l'utilisation de méthodes d'apprentissage en profondeur sera l'objectif de la troisième contribution.

A.3.3. Contribution 3 : Classification de la maladie d'Alzheimer à partir de l'IRM cérébrale

La classification des images médicales est l'un des problèmes les plus importants dans le domaine de la reconaissance d'images, et son objectif est de classer les images médicales en différentes catégories pour aider les médecins à diagnostiquer des maladies ou à approfondir leurs recherches. L'utilisation des méthodes traditionnelles d'apprentissage automatique dans la classification des images médicales a commencé il y a longtemps. Cependant, ces méthodes présentent les inconvénients suivants: les performances sont loin d'être conformes à la norme pratique, et leur développement est assez lent ces dernières années. En outre, l'extraction et la sélection de caractéristiques prennent du temps et varient en fonction des différents objets. Les méthodes d'apprentissage en profondeur, en particulier les réseaux de neurones convolutifs, offrent un moyen efficace de construire un modèle de bout en bout capable de calculer les étiquettes de classification finales avec les pixels bruts des images médicales en raison de sa capacité à apprendre automatiquement des fonctionnalités de bas niveau importantes et à fusionner les à des entités de niveau supérieur dans les couches suivantes. La classification des images médicales basée sur l'utilisation de méthodes d'apprentissage profond sera l'objectif de cette étude. Cependant, la formation d'un réseau neuronal profond qui peut bien se généraliser à de nouvelles données est un problème difficile. Un modèle avec une capacité insuffisante ne peut pas apprendre le problème, alors qu'un modèle avec une capacité trop élevée peut l'apprendre trop bien et surajuster l'ensemble de données d'entraînement.

Au cours des dernières années, un certain nombre d'études de recherche ont été publiées, axées sur le développement d'un réseau de neurones à convolution profonde pour la classification de la maladie d'Alzheimer à l'aide de données IRM.

Récemment, les réseaux de neurones convolutionnels profonds se poursuivent vers une profondeur et une largeur étendue de l'architecture de réseau en raison de l'amélioration des périphériques matériels. D'AlexNet [63] à VGGNet [102] ainsi que GoogleNet [105] et ResNet [46], la profondeur et la largeur des CNN ont continué à augmenter, ce qui peut faciliter l'obtention d'excellentes performances dans la classification d'images. Cependant, des travaux connexes démontrent que des inconvénients tels que la faible convergence, la disparition du gradient et le surajustement peuvent être générés avec des structures de réseau complètes. Pour atténuer ces problèmes et encourager la propagation d'entités, DenseNet [49] permet aux couches d'accéder aux cartes d'entités à partir de toutes ses couches précédentes. DenseNet, l'un des CNN les plus profonds existants, est principalement composé de DenseBlocks (extraction et propagation de cartes de caractéristiques) et de couches de transition (reliant deux DenseBlocks adjacents). Comme mentionné précédemment, DenseNet obtient une précision de pointe, il exploite les connexions résiduelles dans le réseau résiduel (ResNet). Dans un bloc dense, nous représentons l'image d'entrée comme, sortie de la $l^{ème}$ couche comme x_i et chaque module convolutif comme fonction H. Étant donné que les entrées de la $l^{ème}$ couche sont les sorties de toutes les couches précédentes. La formule est définie comme suit:

$$x_l = H\left(\left[x_0, x_1, ..., x_{l-1}\right]\right)$$

Où $\left[x_0, x_1, ..., x_{l-1}\right]$ est la concaténation des sorties de toutes les couches précédentes.

Dans DenseNet, le nombre de cartes de caractéristiques de sortie pour chaque module de convolution est toujours le même, ce qui est indiqué par k. Ainsi, le nombre de cartes de caractéristiques de sortie de la $l^{ème}$ couche est $k_0 + (l-1) \times k$, où k_0 est le nombre de cartes de caractéristiques d'entré dans le premier bloc dense. k a été référencé comme taux de croissance. Au fur et à mesure que DenseNet approfondit, le nombre de cartes d'entités d'entrée deviendrait très bientôt excessif. Pour résoudre ce problème, l'auteur a placé le module de convolution 1×1 avant le module de convolution 3×3. Ainsi, le module convolutif est passé de BN+ReLU+3×3Conv à BN+ReLU+1×1Conv+BN+ReLU+3×3Conv. (Et le nouveau module de convolution est compté comme deux couches au lieu d'une). La configuration normale pour les cartes d'entités en sortie de la couche (1×1Conv) est $4k$. Ainsi, les entrées de chaque couche Conv3×3 sont fixées à 4. Pour améliorer encore la compacité du modèle, le nombre de cartes d'entités peut également être réduit dans la couche de transition. Chaque couche de DenseNet reçoit l'entrée de toutes les couches précédentes et transmet sa sortie à toutes les couches suivantes d'un bloc. DenseNet concatène également des cartes d'entités pour varier les entrées des couches suivantes, contrairement à la sommation dans les réseaux résiduels. Cette opération de réutilisation de fonctionnalités (concaténation de fonctionnalités) crée de nombreux chemins, ce qui entraîne une augmentation exponentielle des besoins en mémoire et de la complexité des calculs. Bien que DenseNet ait obtenu des résultats remarquables en raison de sa connectivité dense et de sa structure profonde, il existe encore quelques lacunes : a) DenseNet nécessite une mémoire quadratique en ce qui concerne leur profondeur; b) DenseNet a une grande quantité d'informations redondantes qui rendent la convergence du réseau difficile; c) Il existe un total de connexions *L(L−1)/2* pour un DenseNet de couche *L*. Les connexions excessives réduisent non seulement l'efficacité des calculs et l'efficacité des paramètres des réseaux, mais rendent également les réseaux plus sujets au surajustement. Ces inconvénients existants sont étroitement liés aux performances de DenseNet en termes de précision de classification, d'occupation des ressources, de temps d'entraînement et de consommation de mémoire. Pour résoudre le problème des connexions excessives en bloc dense qui augmentent les chances de surajustement des données, nous avons proposé une méthode pour réduire le nombre de connexions denses. Zuiler [166] a découvert que pour un réseau neuronal profond, les couches moins profondes peuvent apprendre des caractéristiques concrètes, tandis que les couches plus profondes peuvent apprendre des caractéristiques abstraites. Sur la base de cette observation, nous pouvons supprimer les connexions des couches intermédiaires et réduire les connexions pour chaque couche. Ainsi, notre idée consiste simplement à supprimer les connexions des couches intermédiaires et à ne conserver que les connexions les plus proches et les plus éloignées. Et puis nous pouvons étendre le réseau à plus profond ou plus large, ce qui se traduirait par de meilleures performances. Le bloc dense modifié (MDB) se compose de plusieurs couches connectées dans un motif dense où chaque couche prend les cartes d'entités les plus proches et les plus éloignées en entrée (Voir Fig. 5.19 *p.* 113). Les couches sont connectées par concaténation et transformées en utilisant une transformation non linéaire. Considérons un $r^{ème}$ MDB. Une transformation non linéaire effectuée dans MDB, placée au $l^{ème}$ niveau dans le réseau, produit des cartes d'entités x_l qui sont données comme

$$x_l = H_l\left(\left[x_{i_r}, x_{i_r+1}, \ldots x_{i_r+s/2}, x_{l-s/2}, \ldots, x_{l-1}\right]\right)$$

$$x_l = \phi\left(W_l\left[x_{i_r}, x_{i_r+1}, \ldots x_{i_r+s/2}, x_{l-s/2}, \ldots, x_{l-1}\right] + b_l\right)$$

où $H_l(.)$ est la $l^{ème}$ transformation non linéaire, $\left[x_{i_r}, x_{i_r+1}, \ldots x_{i_r+s/2}, x_{l-s/2}, \ldots, x_{l-1}\right]$ fait référence à la concaténation des cartes d'entités x produites en couches. ϕ désigne la fonction d'activation de ReLU, et W_l désigne le poids de la $l^{ème}$ couche convolutive dans ce $r^{ème}$ RDB. Notez que les cartes de caractéristiques sont introduites dans l'entrée de la $r^{ème}$ MDB. L'entrée de la MDB est placée au niveau i_r du réseau, $i_r < l < o_r$ et o_r est un niveau auquel la sortie de la $r^{ème}$ MDB est placée. Nous appelons s un chemin et nous le mettons à 2 afin de ne conserver que les connexions les plus éloignées et les plus proches. Pour s = 2, les connexions à

chaque couche sont au plus égales à 3. La transformation non linéaire est une fonction composite de plusieurs opérations consécutives. Nous avons utilisé une version de base qui se compose de trois opérations: Normalisation par lots (BN) [52], suivie d'une unité linéaire rectifiée (ReLu) [63], suivie d'une convolution (conv) [68]. En utilisant une notation courte, cette version de la fonction peut être écrite comme BN - ReLu - ($h \times w$) conv, où h et w sont la hauteur et la largeur des filtres. Si nécessaire, les convolutions ont été complétées par zéro pour maintenir la taille de la carte d'entités fixe.

Afin de garantir que les informations d'entrée MDB ne sont pas perdues et d'améliorer le flux d'informations du réseau, une connexion résiduelle locale, comme le montre la figure 5. 20, *p.* 114, est ajoutée entre l'entrée et la sortie de la MDB pour mieux conserver les caractéristiques de la couche avant. Cette méthode de connexion par sauts peut résoudre le problème de la disparition du gradient dans un réseau profond, et la fusion des caractéristiques locales de MDB sera réalisée pour garantir que le réseau peut extraire des caractéristiques concrètes et abstraites. Afin de conserver la cohérence de l'entrée et de la sortie de MDB, la connexion résiduelle locale incluse est une convolution 1×1. Ainsi, la sortie $x_{r,q}$ de cette opération peut être obtenue par

$$x_{r,q} = \Psi_{conv}([x_{i_r}, x_{i_r+1}, x_l])$$

Où Ψ_{conv} le désigne le fonctionnement de la couche convolutive 1 × 1 et $x_{r,q}$ est la sortie de la couche convolutive 1×1 dans le $r^{ème}$ MDB. Pour simplifier, notons I_r, l'entrée du $r^{ème}$ bloc dense résiduel (MDRB) et O_r la sortie du $r^{ème}$ MDRB. Ainsi, la sortie O_r du $r^{ème}$ MDRB (voir Fig. 5.20) peut être désignée comme suit:

$$O_r = I_r + x_{r,q}$$

Le bloc dense du modèle DenseNet utilise une petite taille de noyau convolutif. La fixation de la taille du noyau sur les CNN est une hypothèse inefficace. En effet, le cortex visuel peut collecter des informations à différentes échelles au même niveau de traitement. Si la taille du noyau est sélectionnée pour être petite, le résultat des cartes d'entités a peu de pixels contenant de petits objets. Par conséquent, le CNN à une seule branche avec une taille de noyau fixe est incapable de capturer les caractéristiques des objets locaux à différentes échelles, mais les caractéristiques multi-échelles fusionnées peuvent être complémentaires les unes des autres. Pour surmonter l'inconvénient d'utiliser une seule branche, nous introduisons une approche multi-branche pour créer des branches parallèles avec différentes tailles de noyau et les combinons pour obtenir des cartes de caractéristiques très informatives. Dans le bloc dense résiduel proposé (MDRB), nous utilisons la structure multi-branches pour fournir différentes tailles de noyau dans différentes colonnes, dont l'architecture globale est illustrée sur la figure 5.21, *p.* 115 Il existe trois CNN parallèles dont les structures sont les mêmes, mais des filtres de couches convolutives sont dans chaque couche dans des tailles différentes pour capturer pour capturer plusieurs features sur des images naturelles. La normalisation par lots et l'unité linéaire rectifiée (ReLU) sont ensuite appliquées aux images convoluées et les foulées des filtres dans chaque colonne sont réglées à un. La justification de l'utilisation de différentes tailles de filtre est de détecter les caractéristiques locales et globales.

Après avoir empilé plusieurs couches, le nombre final de canaux de carte d'entités en sortie est la somme du nombre de canaux générés par les trois blocs denses résiduels parallèles. Afin de réduire le nombre de canaux de la carte de caractéristiques finale, nous ajoutons une couche après les couches empilées (c'est-à-dire après la sortie de MSRD), consistant en une convolution 1×1, ce qui peut réduire le nombre de cartes de caractéristiques, réduisant ainsi le montant des calculs ultérieurs. A detailed description of the proposed CNN and its configuration is shown in Fig 5.22, p. 116 and in table 5.10, p. 117

Dans cette étude, nous avons proposé une nouvelle architecture de réseaux de neurones convolutifs profonds (IRDNet) pour capturer les caractéristiques discriminantes des images IRM cérébrales pour le diagnostic de la MA afin de classer plus précisément les cerveaux affectés par la MA et les cerveaux sains normaux. Notre modèle a été formé sur un jeu de données OASIS qui contient des images de divers cas de MA et de cas normaux. De plus, l'efficacité de la classification a été mesurée à l'aide de la validation croisée du facteur *k* et a été comparée à d'autres modèles existant dans la littérature. Les résultats expérimentaux sur l'ensemble de données OASIS montrent que notre architecture proposée réussit à classer AD avec une précision moyenne de 99,31% après l'entraînement pendant 3127 secondes, surpassant les autres architectures CNN présentées dans ce

travail. Ces résultats obtenus démontrent des performances supérieures et prometteuses de l'architecture proposée pour la classification des contrôles AD et normaux. Sur la base de ces résultats encourageants, le modèle IRDNet proposé est un outil intéressant qui peut aider les cliniciens à diagnostiquer et détecter la MA en très peu de temps.

De nos jours, la détection automatique des maladies est devenue une question cruciale dans la science médicale en raison de la croissance rapide de la population. Un cadre de détection automatique des maladies aide les médecins à diagnostiquer la maladie et fournit des résultats exacts, cohérents et rapides et réduit le taux de mortalité. Le coronavirus est devenu l'une des maladies les plus graves et les plus aiguës de ces derniers temps et s'est répandu dans le monde entier. Alors que les cas de COVID-19 augmentent chaque jour, de nombreux pays sont confrontés à des pénuries de ressources. Par conséquent, un système de détection automatique, en tant que diagnostic de diagnostic le plus rapide, doit être mis en œuvre pour empêcher la propagation du COVID-19. Ce sujet sera exploré dans la quatrième contribution.

A.3.4. Contribution 4 : Classification des cas de COVID-19 à partir d'images radiographiques pulmonaires

De nos jours, la détection automatique des maladies est devenue une question cruciale dans la science médicale en raison de la croissance rapide de la population. Un cadre de détection automatique des maladies aide les médecins à diagnostiquer la maladie et fournit des résultats exacts, cohérents et rapides et réduit le taux de mortalité. Le coronavirus est devenu l'une des maladies les plus graves et les plus aiguës de ces derniers temps et s'est répandu dans le monde entier. Comme les cas de COVID-19 augmentent chaque jour, de nombreux pays sont confrontés à des pénuries de ressources. Par conséquent, un système de détection automatique, en tant que diagnostic de diagnostic le plus rapide, doit être mis en œuvre pour empêcher la propagation du COVID-19.

Motivés par la nécessité d'une interprétation rapide des images radiographiques, de nombreux scientifiques ont proposé des modèles d'apprentissage en profondeur, en particulier des réseaux de neurones convolutifs pour détecter les cas infectés par COVID-19 à partir de l'imagerie par radiographie pulmonaire.

P. Kumar [57] a suggéré un CNN profond qui atteint une précision de 95,38%. Inspiré par ce travail qui se concentre principalement sur le ResNet, nous proposons d'améliorer ce réseau afin de mieux distinguer les cas de COVID-19 des cas normaux et de pneumonie virale à l'aide d'images radiographiques pulmonaires. ResNet introduit le concept de réseau résiduel et utilise des connexions par sauts qui ignorent la formation de quelques couches et se connectent directement à la sortie. L'idée derrière ce réseau résiduel est que l'entrée x passe par une série de couches que nous désignons par $F(x)$. Ce résultat est ensuite ajouté à l'entrée d'origine *x*:

$$H(x) = F(x) + x\,.$$

Le CNN proposé, CVDNet, comme indiqué dans la figure 5. 28 *p*. 130, se compose de deux colonnes parallèles. Les deux colonnes ont les mêmes structures à l'exception des tailles de filtres.

Dans les images radiographiques pulmonaires infectées par COVID-19, des opacités de verre dépoli sont observées, qui sont considérées comme les caractéristiques précoces typiques de l'infection par le COVID-19. Ces caractéristiques se trouvent souvent à des endroits différents, y compris dans un sous-pleural et périphérique, ainsi que dans différentes tailles, formes et quantités. Pour les fonctionnalités d'opacités de verre dépoli de grande taille, si nous utilisons un petit filtre, les cartes de caractéristiques résultantes ont peu de pixels contenant de petits objets. Par conséquent, il ne capture pas les informations holistiques. Cela conduit à une mauvaise précision de classification. Pendant ce temps, pour les fonctionnalités d'opacité de verre dépoli de petite taille, si nous utilisons un grand filtre, les champs récepteurs sont trop grands. Ainsi, la plupart des informations ne sont pas pertinentes. Cela conduit à une mauvaise précision de classification en raison de la perte de détails. Par conséquent, nous devons utiliser un petit filtre pour extraire les caractéristiques locales et un grand filtre pour détecter les caractéristiques globales. Cependant, les tailles des caractéristiques de l'image d'entrée sont imprévisibles et aléatoires, ce qui n'est pas pratique pour construire une taille de filtre pour chaque échelle d'objets. Pour surmonter ce problème, nous proposons de concevoir une structure pour capturer des caractéristiques multi-échelles, c'est-à-dire que nous avons appliqué à l'entrée deux convolutions parallèles avec

des filtres grands et petits ; chaque niveau détecte une échelle de caractéristiques. L'utilisation de ces deux filtres en parallèle est plus appropriée pour extraire différentes échelles d'entités qu'un filtre à échelle unique. Pour la convolution, nous utilisons le concept de technique résiduelle. L'excellente performance du réseau résiduel a fait ses preuves dans les tâches d'imagerie médicale, ce qui peut résoudre le problème du gradient de fuite. En connectant les résidus des couches précédentes, cela peut également aider à compenser la perte de détails pour les caractéristiques locales ou à obtenir des informations plus holistiques pour les caractéristiques globales. Cependant, pour détecter plus de détails et des informations plus holistiques, nous proposons de fusionner les caractéristiques multi-échelles via des connexions résiduelles et nous proposons de relier les informations résiduelles non seulement au même niveau mais également à d'autres niveaux. La fusion de différentes échelles de caractéristiques permet d'identifier plusieurs échelles d'objets. La sortie du bloc proposé qui consiste à connecter un résidu avec tous les autres niveaux comme indiqué sur la Fig. 5.29 *p*. 131 est définie comme:

$$o_i = F_i(x_i) + \sum_{i=1}^{L} x_i, \quad i \in \{1, 2, ..., L\}$$

où L est le total des niveaux parallèles, i est le niveau, o_i est la sortie du $i^{ème}$ niveau, $x_1, x_2, ..., x_L$ sont les entrées du bloc et $F_1(x_1), F_2(x_2), ..., F_L(x_L)$ sont leurs sorties correspondantes, respectivement. Dans cette étude, nous n'utilisons que deux niveaux parallèles en appliquant des filtres grands et petits pour extraire des caractéristiques globales et locales.

Sur la base du bloc proposé, nous concevons CVDNet pour classer l'infection au COVID-19 des cas normaux et autres de pneumonie. Le modèle CVDNet proposé a pris comme entrée des images de radiographie pulmonaire. Dans un premier temps, les images sont passées par une étape de prétraitement qui consiste à les recadrer et à les redimensionner. En effet, les images sont redimensionnées car plusieurs lettres, symboles médicaux et artisanat y sont réalisés ainsi que recadrées car elles proviennent de sources différentes et par conséquent leurs tailles varient. Par conséquent, la taille des images d'entrée est modifiée en 256×256×1. Après cette étape, les images sont traitées par deux couches convolutives parallèles ayant une taille de 5×5 et 13×13 avec une foulée=1 respectivement, pour produire 16 cartes de caractéristiques pour chacune d'elles, suivies de la normalisation par lots et des couches ReLU (conv1 et conv2). Les utilisations de conv1 et conv2 modifient la dimension des images de 1×256×256 à 16×251×251 et de 1×256×256 à 16×244×244, respectivement. Les caractéristiques générées par conv1 et conv2 sont transmises à la couche de pooling 2×2 max (MP1 et MP2) avec un pas de deux pour sous-échantillonner les images d'entrée en réduisant leur dimensionnalité et donc pour fournir une invariance spatiale et réduire la quantité de paramètres et calcul dans le réseau. Les dimensions des images résultantes après les couches MP1 et MP2 seront réduites à $16 \times 126 \times 126$ et $16 \times 122 \times 122$. De plus, la sortie des cartes de caractéristiques est transmise au bloc 1 (voir Fig. 5.28 *p*. 130), qui contient deux couches convolutives parallèles avec une taille de 5×5 et 13×13. La sortie de chaque couche convolutive est ajoutée à la sortie de MP1 et MP2 dans la même dimension. La dimension des images de sortie de ce bloc est 16×126×126. Dans ce bloc, il n'a pas de couche de pooling et après chaque couche convolutive nous normalisons chaque lot et nous appliquons la fonction d'activation ReLU. La sortie du bloc 1 passe à travers 5×5 couche convolutive et 13×13 couche convolutive avec un pas de 1 pour générer 32 cartes de caractéristiques pour chacune d'elles, suivies de la fonction BN et ReLU (conv3 et conv4). Les utilisations de conv3 et conv4 changent la dimension des images de 16×126×126 à 32×122×122 et de 16×126×126 à 32×114×114, respectivement. Les fonctionnalités générées par conv3 et conv4 sont transférées vers une couche de pooling 2×2 max (MP3 et MP4). Les dimensions des images résultantes après les couches MP3 et MP4 seront réduites à 32×61×61 et 32× 57×57. La couche de regroupement maximale est transmise au bloc 2 qui est le même que le bloc 1 sauf qu'il a produit 64 cartes de caractéristiques pour chaque couche convolutive. Viennent ensuite à nouveau 5×5 et 13×13 couches convolutives parallèles avec une foulée de 1, suivies de 2×2 max pooling avec une foulée de 2. Après cette couche, le bloc 3 est appliqué, ce qui est le même que le bloc 1 et le bloc 2 sauf qu'il produit 64 cartes d'entités pour chaque couche convolutive. La sortie du bloc 3 passe par 5×5 couche convolutive et 13×13 couche convolutive avec une foulée de 1 pour générer 128 cartes de caractéristiques pour chacune d'elles, suivie par BN, fonction ReLU (conv7 et conv8) et pooling max (MP7 et MP8). La sortie des cartes de caractéristiques des deux couches parallèles est ensuite concaténée et passe par 1×1 couche convolutive (conv9) avec une foulée

de 1 afin de diminuer le nombre de filtres de 256 à 64, donc la sortie sera réduite de $256 \times 12 \times 12$ à $64 \times 12 \times 12$. Après cela, un pooling maximum (MP9) est appliqué pour réduire la dimension des images de $64 \times 12 \times 12$ à $64 \times 6 \times 6$. Enfin, le réseau avait trois couches entièrement connectées; le premier contient 768 cartes de caractéristiques chacune de taille 1×1. Chacun des 768 est connecté à tous les 2304 nœuds ($64 \times 6 \times 6$) de la couche précédente. Le second à 512 unités est connecté à tous les 768 nœuds et enfin, il y a une couche de sortie softmax entièrement connectée avec 3 valeurs possibles. La valeur la plus élevée déterminait la classe prédite. Au total, le réseau compte 5317667 paramètres.
Dans cette étude, nous avons proposé un nouveau modèle de réseau neuronal convolutif profond (CVDNet) pour la détection des cas de COVID-19, afin de distinguer plus précisément les patients atteints de COVID-19 des personnes saines et des patients atteints de pneumonie virale. Ce modèle est basé sur le réseau neuronal résiduel et il est construit en utilisant plusieurs niveaux parallèles avec différentes tailles de noyau pour détecter les caractéristiques locales et globales et en connectant les résidus à d'autres niveaux pour partager des informations. Le CVDNet est formé sur 2905 images de radiographie pulmonaire qui sont open source et accessibles au public pour classer les cas de COVID-19 des cas de pneumonie normale et virale. De plus, son efficacité de classification a été mesurée à l'aide de la validation croisée du facteur k et les résultats du modèle proposé sont comparés à d'autres études rapportées dans la littérature. Il a été observé que CVDNet a atteint une précision moyenne de 97,20% pour la détection du COVID-19 et une précision moyenne de 96,69% pour la classification en trois classes (COVID-19 vs pneumonie normale vs virale), qui présentent des performances supérieures et prometteuses dans la classification. Sur la base de ces résultats encourageants, notre modèle CVDNet est un outil intéressant qui peut aider les cliniciens à diagnostiquer et à détecter une infection au COVID-19 en très peu de temps.

A.3.2 Perspectives

- Travaux sur la segmentation des tissus d'IRM cérébrale : La méthode MKPE proposée s'est avérée efficace pour la segmentation d'image. Cependant, elle présente certains inconvénients:
    - Bien que l'introduction d'informations locales dans les fonctions objectives correspondantes augmente dans une certaine mesure leur insensibilité au bruit, il manque encore d'une robustesse suffisante au bruit, en particulier en l'absence de connaissance préalable du bruit;
    - Dans notre fonction objective, il existe un paramètre crucial utilisé pour équilibrer la robustesse au bruit et l'efficacité de la préservation des détails de l'image, il est généralement choisi par l'utilisateur
    - Le temps de segmentation d'une image dépend de la taille de l'image.

  À l'avenir, nous essayerons d'atténuer aux inconvénients de MKPE et améliorerons en même temps les performances de segmentation.

- Travaux sur les indices de validité des clusters: Cette étude a des travaux futurs potentiels.
    - Dans cette recherche, la valeur de l'exposant de pondération m est supposée être prédéfinie. L'élaboration d'une nouvelle stratégie pour sélectionner un m approprié, en fonction du comportement de l'indice, nécessite des travaux supplémentaires.
    - Nous prévoyons d'appliquer la méthode proposée à des exemples plus réalistes.
    - Fuzzy c-means est peut-être l'algorithme de clustering le plus connu et le plus largement utilisé. Plusieurs facteurs ont été identifiés qui peuvent fortement affecter les performances de clustering de FCM, tels que le bruit et les valeurs aberrantes. Les résultats de regroupement imparfaits de la FCM causés par ces facteurs peuvent également affecter les décisions des CVI existants, qui seront étudiés dans les travaux futurs.

- Travaux sur la classification de la maladie d'Alzheimer à partir de l'IRM cérébrale: Nos travaux futurs se concentrent sur l'utilisation du réseau de neurones convolutifs profonds dans la modélisation de la progression de la maladie d'Alzheimer et sur l'utilisation du modèle CNN pour analyser des images

médicales pour d'autres troubles neurologiques. Malgré les résultats prometteurs obtenus à l'aide de notre modèle profond il reste encore beaucoup à faire et plusieurs tendances de recherche prometteuses. En particulier, nous mettons en évidence un ensemble de problèmes qui pourraient être améliorés dans les travaux futurs tels que:

- Notre modèle nécessite d'ajuster de nombreux hyperparamètres tels que le taux d'apprentissage et les fonctions du noyau afin de bien former le modèle. Une simple modification de ces paramètres pourrait conduire à des modèles radicalement différents avec des performances variables. Une contribution future est de trouver automatiquement les hyperparamètres optimaux.
- Des moyens plus efficaces de réduire le coût de calcul et la taille de poids du réseau, et des moyens plus efficaces d'explorer les caractéristiques globales sont fortement souhaités pour être étudiés.

- Classification des cas de COVID-19 à partir d'images radiographiques pulmonaires : Le model proposé présente certaines limites.
  - La taille de l'échantillon est relativement petite et doit être augmentée pour tester la généralisabilité du modèle développé. Ce serait surmonté si plus d'images étaient trouvées dans les prochains jours.
  - Il se concentre uniquement sur les rayons X.
  - Les images COVID-19 comprenant plusieurs symptômes de maladie ne peuvent pas être classés efficacement.
  - Enfin, une comparaison du système proposé avec d'autres radiologies ferait partie d'une étude future.

References

[1] Ahmed, M. N., Yamany, S. M., Mohamed, N., Farag, A., & Moriarty, T. A modified fuzzy c-means algorithm for bias field estimation and segmentation of MRI data. *IEEE transactions on medical imaging*, 21(3): 193-199 (2002)

[2] Antin, B., Kravitz, J., and Martayan, E. Detecting Pneumonia in Chest X-Rays with Supervised Learning. http://cs229.stanford.edu/proj2017/final-reports/5231221.pdf, 1-5, (2017).

[3] Apelt, D., Preim, B., Hahn, HK., Strau, G. Bildanalyse und visualisierung für die Planung von Nasennebenhöhlen-Operationen. Bildverarbeitung für die Medizin 2004. *Informatik aktuell. Springer*, S194–S198 (2004)

[4] Askari, S., Montazerin, N., Fazel Zarandi, M.H. Generalized possibilistic fuzzy c- means with novel cluster validity indices for clustering noisy data. *Applied Soft Computing Journal*. (2016)

[5] Axelrod, A., He, X., Gao, J. Domain adaptation via pseudo in-domain data selection. in Proceedings of the conference on empirical methods in natural language processing. *Association for Computational Linguistics*. 355–362 (2011).

[6] Bartol, T., Thoughtful use of diagnostic testing: Making practical sense of sensitivity, specificity, and predictive value. *The Nurse practitioner*. 40 (8): 10-12 (2015)

[7] Bekir, K. and Vehbi, O. Performance analysis of various activation functions in generalized mlp architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems*, 1(4):111–122 (2011).

[8] Bell, J. [Online]. Available: https://www.nsmedicaldevices.com/news/covid-19-testing-kits-shortages/. Last accessed date 17th Nov (2020)

[9] Bensaid, A.M., Hall, L. O., Bezdek, J. C., et al., Validity-guided (re)clustering with applications to image segmentation, *IEEE Transactions on Fuzzy Systems*. 4 (2): 112–123 (1996).

[10] Bengio, Y. Learning deep architectures for AI. Foundations and trends in Machine Learning. 2(1):1–127 (2009).

[11] Bezdek, J. C. Cluster Validity with Fuzzy Sets. *Journal of Cybernetics*, 3(3):58-72 (1974)

[12] Bezdek, J.C., Ehrlish, R., Full, W. FCM: The Fuzzy C-means Clustering Algorithm. *Computers & Geosciences*. 10 (2-3):191-203 (1984).
[13] Bezdek, J.C., Hall, L.O., Clarke, L.P. Review of MR image segmentation techniques using pattern recognition. *Medical Physics*. 20 (4): 1033–1048 (1993).
[14] Bezdek, J. C. Numerical Taxonomy with Fuzzy Sets, *J. Math.Biol*. 1: 57–71 (1977).
[15] Bezdek, J. C. Pattern recognition with fuzzy objective function algorithms. *Plenum, New York*. (1981)
[16] "Brain scans — Alzheimer's Research UK." [Online]. Available: http://www.dementialab.org/development-lab/brain-scans/.
[17] BrainWeb, Available online: Simulated Brain Database, http://brainweb.bic.mni.mcgill.ca/brainweb/ (accessed on 17 August 2004)
[18] Chaira, T. A novel intuitionistic fuzzy c-means clustering algorithm and its application to medical images. *Appl. Soft comput*. 11:1711-1717 (2011).
[19] Chen, S., Zhang, D. Robust image segmentation using fcm with spatial constraints based on new kernel-induced distance measure. *IEEE Trans. Syst. Man Cybern. Part B*. 34 (4): 1907-1916 (2004)
[20] Celik, Y., Talo, M., Yildirim, O., Karabatak, M., Acharya, U.R. Automated invasive ductal carcinoma detection based using deep transfer learning with whole-slide images**.** *Pattern Recognit. Lett.* 133 pp. 232-239(2020)
[21] Damodarasamy, S., Raman, S. Texture analysis using computer vision. *Computers in Industry* 16: 25–34 (1991)
[22] da S., N.I., Hernane, S.D., Andrade, F.R., Bartocci, L. L.H., dos Reis, A.S.F., et al. Artificial Neural Networks a Practical Course. *Springer International Publishing, Switzerland*. (2017)
[23] Das, NN., Kumar, N., Kaur, M., Kumar, V., and Singh, D. Automated Deep Transfer Learning Based Approach for Detection of COVID-19 Infection in Chest X-rays. *IRBM*. (2020).
[24] Dao-Qiang, Z., Song-Can, C. A novel kernelized fuzzy C-means algorithm with application in medical image segmentation. *Artificial Intelligence in Medicine.* 32: 37–50 (2004)
[25] Deborah, L.J., Baskaran, R., Kannan, A. A survey on internal validity measure for cluster validation. *Int. J. Comput. Sci. Eng. Surv*. 1 (2):85-102 (2010)
[26] Deo, RC. Machine learning in medicine. *Circulation*.132(20):1920– 1930 (2015).
[27] Dice LR. Measures of the amount of ecologic association between species. *Ecology*. 23(3):297-302 (1945).
[28] Diederik, P. K. and Jimmy, B. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[29] Doi, K. Computer-aided diagnosis in medical imaging: historical review, current status and future potential. *J. Computerized Medical Imaging and Graphics.* 31(4-5):198-211 (2007).
[30] Duda, R., Hart, P., Stork, D. *Pattern Classification*. John Wiley (2000)
[31] Dunn, J.C. A fuzzy relative of the ISODATA process and its use in detecting compact well separated cluster. *Journal of Cybemet.* 3: 32–57 (1974).
[32] Dunn, J. C. Indices of partition fuzziness and the detection of clusters in large data sets. *In Fuzzy Automata and Decision processes. Elsevier, NY*. (1997).
[33] Esteva, A., Kuprel, B., Novoa, R.A. *et al***.** Dermatologist-level classification of skin cancer with deep neural networks**.** *Nature*. 542 (7639): 115-118 (2017).
[34] Farina, N., Isaac, M. G. E. K. N., Clark A. R., Rusted J., and Tabet N., Vitamin E for Alzheimer's dementia and mild cognitive impairment. *The Cochrane database of systematic reviews*, vol. 11, p. CD002854, nov (2012).
[35] Federico, G., Michael, J., and Tomaso, P. Regularization theory and neural networks architectures. *Neural Computation*, 7(2):219-269 (1995).
[36] Feng, C., Li, C., Zhao, D., Davatzikos, C., Litt, H. Segmentation of the left ventricle using distance regularized two-layer level set approach, in: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2013, *Nagoya, Springer*. pp. 477–484 (2013).
[37] Feng, Z., and Fan, J.C. A novel validity index in fuzzy clustering algorithm, *Int. J. Wireless and Mobile Computing*. 10 (2) :183–190 (2016).
[38] Forman, G., Scholz, M., Apples-to-apples in cross-validation studies: pitfalls in classifer performance measurement, *ACM SIGKDD Explorations Newsletter*, 12(1) : 49–57(2010)
[39] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6) : 386, 1958.
[40] Fukushima, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*. 36 (4) : 193-202 (1980).

[41] Fukuyama, Y., Sugeno, M. A New Method of Choosing the Number of Cluster for the Fuzzy C-means Method. *Proceedings of the 5th Fuzzy Systems. Symposium*. 247-250 (1989).
[42] Gaal, G., Maga, B., Lukacs, A. Attention U-net based adversarial architectures for chest x-ray lung segmentation. *arXiv preprint.* (2020)
[43] Ghoshal, B., and Tucker, A. Estimating uncertainty and interpretability in deep learning for coronavirus (COVID-19) detection. *arXiv*: 2003.10769, (2020).
[44] Gosain, A., & Dahiya, S. Performance Analysis of Various Fuzzy Clustering Algorithms. *A Review, 7th International Conference on Communication. Computing and Virtualization.* 79: 100 – 111 (2016).
[45] Goodfellow, I., Bengio, Y., Courville, A., Deep Learning. *MIT Press*. 781 p. Available: http://www.deeplearningbook.org. (2016)
[46] Goutte, C., Gaussier, E. A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation. *In ECIR'05: proceedings of the 27th European conference on information retrieval*, 345-359 (2010)
[47] Grasselli, G., Pesenti, A., Cecconi, M. Critical cate utilization for the COVID-19 outbreak in Lombardy, *Italy JAMA*, 323 (16): 1545-1546 (2020).
[48] Guang, H., Zhenbin, D. Adaptive Kernel-Based Fuzzy C-Means Clustering with Spatial Constraints for Image Segmentation. *International Journal of Pattern Recognition and Artificial Intelligence.* 33: 1 (2019).
[49] Gurney, K. An introduction to neural networks. *UCL Press (Taylor & Francis group)*. (1997).
[50] Hakon, G. and Samuel, P. The rician distribution of noisy MRI data. *Magnetic Resonance in Medicine*. 34(6):910–914, (1995).
[51] Halder, A. & Talukdar, N.A. Brain tissue segmentation using improved kernelized rough-fuzzy C-means with spatio-contextual information from MRI. *Magnetic Resonance Imaging*. 62: 129-151 (2019).
[52] Haouas, F., Dhiaf, Z. B., Hammouda, A., Solaiman, B. A new efficient fuzzy cluster validity index: Application to images clustering. *In Proc. IEEE Int. Conf. Fuzzy Systems (FUZZ-IEEE).* 1–6 (2017).
[53] Hathaway, R.J., Bezdek J.C. Extending fuzzy and probabilistic clustering to very large data sets. *Computational Statistical and Data Analysis*. 51:215-234 (2006)
[54] Hathaway, R. J, Davenport, J. W. & Bezdek, J.C. Relational duals of the c-means clustering algorithms. *Pattern Recognit*, 22(2): 205–212 (1989).
[55] He, K., Zhang, X., Ren, S., and Sun, J. Residual Learning for Image Recognition, *Multimed. Tools Appl*. 77 (9): 10437–10453 (2015).
[56] Hemdan, E.E., Shouman, M.A., Karar, M.E. Covidx-net: a framework of deep learning classifiers to diagnose covid-19 in X-ray images. *arXiv preprint arXiv*: 2003.11055. (2020).
[57] Ho, T.K.K., Gwak, J., Prakash, O., Song, J.-I., Park C.M. Utilizing pretrained deep learning models for automated pulmonary tuberculosis detection using chest radiography. Asian conference on intelligent information and database systems, Springer. pp. 395-403 (2019).
[58] Horowitz, S. L. and Pavlidis, T., Picture segmentation by a tree traversal algorithm, *J. Ass. comput*. Mach. 368-388 (1976)
[59] Huang, G., Liu, Z., Weinberger, K. Q., and Maaten, L. Densely connected convolutional networks. *In CVPR*, 2017
[60] Hubel, D. H., Wiesel, T. N. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*. 195(1): 215-243 (1968).
[61] Ioannis, D., Apostolopoulos, T.B. COVID-19: automatic detection from x-ray im- ages utilizing transfer learning with convolutional neural networks. Physical and Engineering Sciences in medicine.1, (2020).
[62] Ioffe, S. and Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv e-print, arXiv:*1502.03167, Mar (2015)
[63] Islam, M.M., Hannan, T., Sarker, L., Ahmed, Z. COVID-DenseNet: a deep learning architecture to detect COVID-19 from chest radiology images. https://doi.org/10.20944/preprints202005.0151.v1 (2020)
[64] Jain, A. K., Murty, M.N., and Flynn, P. J. Data clustering: A review. *ACM Comput. Surv*, 31(3): 264-323, September (1999).
[65] Jaiswal, AK., Tiwari, P., Kumar, S., Gupta, D., Khanna, A., and Rodrigues, JJ. Identifying pneumonia in chest X-rays: A deep learning approach. *Measurement*, 145:511-518 (2019).
[66] Jeffrey, L. C., Travis, M., and Kate, Z. Alzheimer's disease drug-development pipeline: few candidates, frequent failures. *Alzheimer's research & therapy*. 6(4):37 (2014).
[67] Joopudi, S. Rathi, S. Suraj, Narasimhan, S., Raghunathan, R. A New Cluster Validity Index for Fuzzy Clustering. *10th IFAC International Symposium on Dynamics and Control of Process System*. (2013)
[68] Kaelbling, L.P., Littman, M.L., Moore, A.W., Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*. 4 (1996).

[69] Kass, M., Witkin, A., & Terzopoulos, D. Snakes. Active contour models. International *Journal of Computer Vision*, 1: 321-332 (1988).
[70] Khan, AI., Shah, JL., Bhat, MM, CoroNet: a deep neural network for detection and diagnosis of COVID-19 from chest X-ray images. *Comput Methods Programs Biomed* 196:105581 (2020).
[71] Khalilia, M. A. et al. Improvements to the relational fuzzy c-means clustering algorithm. *Pattern Recognition*. (2014)
[72] Kim, D.W., Lee, K.H. and Lee, D. On cluster validity index for estimation of the optimal number of fuzzy clusters, *Pattern Recognition*, Vol. 37, No. 10, pp.2009–2025 (2004).
[73] Klappel, S., Stonnington, C. M., Chu, C., Draganski, B., Scahill, R. I., Rohrer, J. D., Fox, N. C., Jack, C. R., Ashburner, J., and Frackowiak, R. S. J. Automatic classification of MR scans in Alzheimer's disease. *Brain*. 131 (3) :681–689 (2008).
[74] Krizhevsky, A., Sutskever, I and Hinton, G. Imagenet classification with deep convolutional neural networks. *In Advances in neural information processing systems*. 1097–1105 (2012).
[75] Kubat, M., Holte, R.C., Matwin, S., Machine Learning for the Detection of Oil Spills in Satellite Radar Images. *Machine Learning*. 30: 195– 215 (1998).
[76] Kumar, P. and Kumari, S. Detection of coronavirus disease (COVID-19) based on deep features. *preprints.org*, no. March, p. 9, Mar. (2020).
[77] Kumbhar, G. P, & Holambe, N. S. A review of image Thresholding techniques. *International Journal of Advanced Research in Computer Science and Software Engineering*. 5(6), June. (2015)
[78] Kwon, S. H. Cluster Validity Index for Fuzzy Clustering. *Electronics Letters*. 34(22):2176-2177 (1998)
[79] Labati, R. D., Piuri, V. and Scotti, F. ALL-IDB: The acute lymphoblastic leukemia image database for image processing. *In Proc. Int. Conf. Image Process*. pp. 2045–2048 (2011).
[80] Lancu, I. and, Olmer A. The minimental state examination–an up-todate review. *Harefuah*. 145 (9) 687–690:(2006).
[81] Latif, S., Usman, M., Manzoor, S., Iqbal, W., Qadir, J., Tyson, G., et al. Leveraging data science to combat COVID-19: a comprehensive review https://doi.org/10.36227/techrxiv.12212516.v1 (2020)
[82] LeCun, Y., Bengio, Y., Hinton, G**.** Deep learning**.** *Nature*, 521 (7553): 436-444 (2015).
[83] LeCun, Y., Bernhard, E. B., John, S. Denker, Donnie, H., Richard, E. H., Wayne, E. H., and Lawrence, D. J. Handwritten digit recognition with a back-propagation network. *In Advances in neural information processing systems*. 396–404 (1990).
[84] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324 (1998).
[85] LeCun, Y., Jackel, L. D., Boser, B., Denker, J. S., Graf, H. P., Guyon, I., Henderson, D., Howard, R. E., and Hubbard, W. Handwritten digit recognition: Applications of neural net chips and automatic learning. *IEEE Communication*. 41–46, November (1989).
[86] LeCun, Y., Yoshua, B. et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*. 3361(10): 1995 (1995).
[87] Lehmann, T., Oberschelp, W., Pelikan, E., and Repges, R., Bildverarbeitung fur die Medizin: Grundlagen, Modelle, Methoden, Anwendungen (in German). *Berlin, Germany: Springer-Verlag* (1997).
[88] Li, C. et al. A level set method for image segmentation in the presence of intensity inhomogeneities with application to MRI. *IEEE Trans. Image Process*. 20 (7): 2007–2016, Jul. (2011).
[89] Li, L., Qin, L., Xu, Z., Yin, Y., Wang, X., Kong, B., et al. Artificial intelligence distin- guishes COVID-19 from community acquired pneumonia on chest CT. *Radiology*. 2020:200905.
[90] Liew, A.W.C., Leung, S.H., Lau, W.H. Fuzzy image clustering incorporating spatial continuity. *IEEE Proceedings of the Vision, Image and Signal Processing*, 147 (2):185-192 (2000)
[91] Maas, L.C., Mukherjee, P., Carballido-Gamio, J, et al. Early laminar organization of the human cerebrum demonstrated with diffusion tensor imaging in extremely premature infants. *Neuroimage*. 22 (3):1134-1140 (2004).
[92] McCulloch, W and Pitts, W. A logical calculus of the idea's immanent nervous activity. *Bulletin of Mathematical Boiphysics*. 115–133 (1943).
[93] McKhann, G., Drachman, D., Folstein, M., Katzman, R., Price, D., and Stadlan, E. M. Clinical diagnosis of Alzheimer's disease: Report of the NINCDS-ADRDA Work Group under the auspices of Department of Health and Human Services Task Force on Alzheimer's Disease. *Neurology*, 34 (7):939–939 (1984). [Online]. Available: http://www.neurology.org/cgi/doi/10.1212/WNL.34.7.939
[94] Mei, S. C., and Suresh, S. Preclinical alzheimer's disease: diagnosis and prediction of progression. *The Lancet Neurology*, 4(9):576–579 (2005).
[95] Mekhmoukh, A. and Mokrani, K. MR Brain Image Segmentation Using an Improved Kernel Fuzzy Local Information C-Means Based Wavelet, Particle Swarm Optimization (PSO) Initialization and Outlier

Rejection with Level Set Methods. *The International Arab Journal of Information Technology*. 15 (4) 683- 692 (2018).
[96] Miin-Shen, Y. and Kuo-Lung, W. A cluster validity index for fuzzy clustering. *Pattern Recognition Letters*. 26 (9) :1275-1291 (2005).
[97] Minsky, M., and Papert, S. Perceptrons: an introduction to computational geometry. *MIT Press*, (1969).
[98] Monplaisir, K. Effect of oil palm phenolics on beta amyloid deposition in cholesterol induced rat model of Alzheimer's disease: Histological evidence. PhD thesis, Wayne State University (2016).
[99] Mullis, K.B., Faloona, F.A. Methods in Enzymology, Academic Press, New York, 155:335-350 (1987).
[100] Mundhra, D., Cheluvaraju, B., Rampure, J. and Dastidar, T.R. Analyzing microscopic images of peripheral blood smear using deep learning, *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*. 178–185 (2017).
[101] National Health Comission of the People's Republic of China. Diagnosis and treatment of pneumonia with new coronavirus infection (trial version 5)［EB/OL］.［2020-02-15]. http://www.nhc.gov.cn/yzygj/s7653p/202002/3b09b894ac9b4204a79db5b8912d4440.sht ml
[102] Nemirovski, A., Juditsky, A., Lan, G., Shapiro, A. Robust stochastic approximation approach to stochastic programming. *SIAM J Optim*. **19**:1574–609 (2009)
[103] Niessen, W. J., Bouma, C. J., Vincken, K. I. and Viergever, M. A. Error metrics for quantitative evaluation of medical image segmentation. *In R. Klette, H. S. Stiehl, M.A. Viergever, and K. L. vincken, editors, perform. Char. Comput. Vis., Computational Imaging and Vision. Dordrecht* (2000).
[104] Ning, Q. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145 – 151, (1999).
[105] Nitish, S., Geoffrey, H., Krizhevsky, A., Ilya S., and Ruslan, S. Dropout: A simple way to prevent neural networks from overffitting. *Journal of Machine Learning Research*. 15:1929-1958 (2014).
[106] Oshua, B., Patrice, S., and Paolo, F. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166 (1994).
[107] Ozturk, T., Talo, M., Yildirim, EA., Baloglu, UB., Yildirim, O., Acharya, UR. Automated detection of COVID-19 cases using deep neural networks with X-ray images. *Comput Biol Med* 2020:103792.
[108] Pakhira, M. K., Bandyopadhyay, S., Maulik, U., Validity Index for Crisp and Fuzzy Clusters, *Pattern Recognition*. 37(3):487-501 (2004).
[109] Pham, D.L. Fuzzy clustering with spatial constraints. *In: Proceedings of the IEEE International Conference on Image Processing, New York, USA, August*. (2002).
[110] Pham, D.L., Prince, J.L., An adaptive fuzzy C-means algorithm for image segmentation in the presence of intensity inhomogeneities. *Pattern Recognition Letters.* 20 (1) :57–68 (1999).
[111] Razvan, P., Tomas, M., and Yoshua, B. On the difficulty of training recurrent neural networks. *In Proceeding of International Conference on Machine Learning*, pages 1310–1318 (2013).
[112] Rezaee, B. A cluster validity index for fuzzy clustering. *Fuzzy Sets and Systems*. 161 (23):3014-3025 (2010)
[113] Rezaee, M. R., Lelieveldt, B. P.F., Reiber, J. H.C. A new cluster validity index for the fuzzy c-mean. *Pattern Recognition Letters*. 19 (3-4) : 237-246, Mar (1998).
[114] Rosenfeld, A., & Avinash, C. K. Digital picture processing. *Academic press*. (1982)
[115] Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representations by error propagation. *In: Parallel distributed processing: Explorations in the microstructure of cognition*, vol. I, pp. 318–362. Bradford Books, Cambridge, MA (1986)
[116] Ruspini, E. Numerical methods for fuzzy clustering. *Inf. Sci. 2*, 319–350 (1970).
[117] Russell, S. J. and Norvig, P. Artificial Intelligence: A Modern Approach. *Pearson Education, 2 edition*, (2003).
[118] Samuel, A.L., Some studies in machine learning using the game of checkers. *IBM J. Res. Develop*. 3 (3): 210-229 (1959).
[119] Sarma, T.H., Viswanath P., Reddy, B.E. Speeding-up the kernel K-means clustering method: a prototype-based hybrid approach. *Pattern Recognit*. Lett. 34: 564-573 (2013)
[120] Schmidhuber, J., Deep learning in neural networks: An overview, *Neural Networks*, 61: 85 – 117 (2015).
[121] Sepp H., Yoshua B., Paolo F., and Jürgen, S. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In A Field Guide to Dynamical Recurrent Neural Networks. Wiley-IEEE Press (2001).
[122] Shieh, H.-L. Robust validity index for a modified subtractive clustering algorithm. *Appl. Soft Comput. J*. (2014).
[123] Simonyan, K. and Zisserman, A., Very deep convolutional networks for large-scale image recognition. *In ICLR* (2015).

[124] Song, Y., Zheng, S., Li, L., Zhang, X., Zhang, X., Huang, et al. Deep learning enables accurate diagnosis of novel Coronavirus (COVID-19) with CT images. *MedRxiv*. (2020).
[125] Soni, A.K, Kaur, P, Gosain, A. A robust kernelized intuitionistic fuzzy c-means clustering algorithm in segmentation of noisy medical images. *Pattern Recognition Letters*. 34 (2) 163-175 (2013).
[126] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. *In CVPR (*2015).
[127] Talo, M., Yildirim, O., Baloglu, U.B., Aydin, G., Acharya, U.R. Convolutional neural networks for multi-class brain disease detection using MRI images. *Comput. Med. Imag. Graph*, 78, Article 101673(2019).
[128] Tan, J.H., Fujita, H., Sivaprasad, S., Bhandary, S., Rao, A.K., Chua, K.C., Acharya, U.R. Automated segmentation of exudates, haemorrhages, microaneurysms using single convolutional neural network. *Inf. Sci*. pp. 66-76: 420 (2017).
[129] Tang, Y.G., Sun, F.C., Sun, Z.Q., Improved Validation Index for Fuzzy Clustering, *American Control Conference*. 1120-1125 (2005).
[130] Tieleman, T., & Hinton, G. Lecture 6.5 RMSProp. COURSERA: Neural networks for machine learning, *Technical report* (2012)
[131] Tolias, Y.A., S.M. Panas. On applying spatial constraints in fuzzy image clustering using a fuzzy rule-based system. *IEEE Signal Processing Letters*. 5 (10): 245–247(1998).
[132] Tolias, Y.A., Panas, S.M. Image segmentation by a fuzzy clustering algorithm using adaptive spatially constrained membership functions. *IEEE Transactions on Systems, Man, and Cybernetics—Part 2*. 8(3): 359–369 (1998).
[133] Tolksdorf, K., Buda, S., Schuler, E., Wieler, L.H., Haas, W. Influenza-associated pneumonia as reference to assess seriousness of coronabviurs disease (COVID-19). *Euro Surveill*. 25: (11) (2020)
[134] Tom, M., Mitchell, TM. Machine Learning. *McGraw-Hill Education* (1997).
[135] Tor-Diez, C., Passat, N., Bloch, I., Faisan, S., Bednarek, N., Rousseau, F.: An iterative multi-atlas patch-based approach for cortex segmentation from neonatal MRI. *Comput. Med. Imaging Graph*. 70, 73–82 (2018)
[136] Wang, B., Gao, X., Tao, D., Li, X. A unified tensor level set for image segmentation. *IEEE Trans. Syst. Man Cybern. Part B: Cybern*, 40 (3): 857-867 (2010)
[137] Wang, L., Lin, Z.Q. and Wong, A. COVID-net: a tailored deep convolutional neural network design for detection of COVID-19 cases from chest x-ray images. *arXiv*, Mar. (2020).
[138] Wang, S., et al. A deep learning algorithm using CT images to screen for Corona Virus Disease (COVID-19). *medRxiv*, p. 2020.02.14.20023028, Apr. (2020).
[139] Wang, W., Zhang, Y. On fuzzy cluster validity indices. *Fuzzy Sets Syst*. 158 (19) 2095–2117(2007).
[140] Wang, S., Zhang, Y., Zhan, TM., Phillips, P., Zhang, YD., Liu, G., et al. Pathological brain detection by artificial intelligence in magnetic resonance imaging scanning. *Prog Electromagn Res*.156:105–133 (2016).
[141] Warfield, SK., Westin, CF., Guttmann, CRG., Albert, MS., Jolesz, FA., Kikinis, R. Fractional segmentation of white matter. *In: Proceedings of Second International Conference on Medical Imaging Computing and Computer Assisted Interventions*: 62–71 (1999).
[142] World Health Organization: Covid-2019 situation reports. https://www.who.int/emergencies/diseases/novel-coronavirus-2019/situation-reports/.
[143] World Health Organization: Covid-19 pandemic. https://www.who.int/emergencies/diseases/novel-coronavirus-2019
[144] Wu, C.H., Ouyang, C.S, Chen, L.W, Lu, L.W. A new fuzzy clustering validity index with a median factor for centroid-based clustering. *IEEE Trans. Fuzzy Syst*. 23: 701–718 (2014)
[145] Wu, J., Chen, J., Xiong, H., Xie, M. External validation measures for K-means clustering: a data distribution perspective. *Expert Syst*. Appl., 36:6050-6061 (2009)
[146] Wu, Y., He, C. A convex variational level set model for image segmentation. *Signal Process*. 106:123-133 (2015)
[147] Xie, X.L., Beni, G., A Validity Measure for Fuzzy Clustering, *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 13(8):841-847 (1991).
[148] Xu, X., et al. Deep learning system to screen coronavirus disease 2019 pneumo nia. *arXiv*, Feb. (2020).
[149] Yang, G., Nawaz, T., Barrick, TR., Howe, FA., Slabaugh, G. Discrete wavelet transform-based whole-spectral and subspectral analysis for improved brain tumor clustering using single voxel MR spectroscopy. *IEEE Trans Biomed Eng*. 62(12):2860 (2015).
[150] Yoon, S.H., Lee, K.H., *et al.* Chest radiographic and CT findings of the 2019 novel coronavirus disease (COVID-19): analysis of nine patients treated in Korea Korean J. *Radiol*. 21 (4): 494-500 (2020).
[151] Yoram, S. John, D., Elad, H. Adaptive subgradient methods for online learning and stochastic

optimization. *Journal of Machine Learning Research*. 12 (7): 2121–2159 (2011).
[152] Zangwill, W.I. Nonlinear Programming: A Unified Approach. *Englewood Cliffs*, NJ*: Prentice-Hall* (1969)
[153] Zarandi, F. M. H., Faraji, M.R. and Karbasian, M. An Exponential Cluster Validity Index for Fuzzy Clustering with Crisp and Fuzzy Data. *Transaction E: Industrial Engineering*. 17 (2): 95-110 (2010).
[154] Zhang, J., Xie, Y., Li, Y., Shen, C., and Xia, Y. COVID-19 screening on Chest X-ray images using deep learning-based anomaly detection. *arXiv*: 2003.12338 (2020).
[155] Zhao, F., Jiao, L., Liu, H. Kernel generalized fuzzy c-means clustering with spatial information for image segmentation. *Digital Signal processing*. 23: 184–199 (2013).
[156] Zhou, K.L., Ding, S., Fu, C., Yang, S.L. Comparison and Weighted Summation Type of Fuzzy Cluster Validity Indices. *Int J Comput Commun*. 9(3):370-378 (2014)
[157] Zucker, S. Region growing: childhood and adolescence. *Computer Vision Graphics and image processing*. 5:382-399 (1976).
[158] Zuiler M.D., Fergus R. Visualizing and Understanding Convolutional Networks. In: Fleet D., Pajdla T., Schiele B., Tuytelaars T. (eds) Computer Vision – ECCV 2014. ECCV 2014. *Lecture Notes in Computer Science*, vol 8689. Springer (2014)