جامعة سيدي محمد بن عبد الله بفاس
UNIVERSITÉ SIDI MOHAMED BEN ABDELLAH DE FES

كلية العلوم والتقنيات فاس
FACULTÉ DES SCIENCES ET TECHNIQUES DE FÈS

N° d'ordre : 123/2023

Centre des Etudes Doctorales : Sciences et Techniques et Sciences Médicales
Formation Doctorale : Sciences de l'Ingénieur, Sciences Physiques, Mathématiques et Informatique (SISPMI)
Discipline : Informatique
Spécialité* : Informatique
Laboratoire : Ingénierie, Systèmes et Applications

Thèse pour l'obtention de DOCTORAT

Adaptive models for Segmentation and classification of data using machine and deep learning algorithms

Préparée par : Mlle Hanae Moussaoui

Soutenue le 30/12/2023 à 10h00 devant le jury :

Nom et Prénom	Grade	Etablissement	Qualité
Khalid SATORI	PES	Faculté des Sciences Dhar EL Mehraz de Fès	Président
Abdellatif EZZOUHAIRI	PES	Ecole Nationale des Sciences Appliquées de Fès	Rapporteur
Mohammed OUANAN	PES	Faculté des Sciences de Meknès	Rapporteur
Hassan SILKAN	PES	Faculté des Sciences d'El Jadida	Rapporteur
Abdelhak BOULAALAM	PH	Ecole Nationale des Sciences Appliquées de Fès	Examinateur
Khalid HADDOUCH	PH	Ecole Nationale des Sciences Appliquées de Fès	Examinateur
Nabil EL AKKAD	PH	Ecole Nationale des Sciences Appliquées de Fès	Directeur de thèse
Mohamed BENSLIMANE	PH	Ecole Supérieure de Technologies de Fès	Co-directeur de thèse

Acknowledgments

Praise be to Allah for giving me the chance and ability to continue my university studies and achieve the doctoral cycle.

Just before the Covid-19 epidemic began, when I began my PhD, there were many difficulties and unknowns. Nonetheless, I made it with the help and support of the most important people in my life. Therefore, I want to convey my sincere thanks to the amazing people who supported and mentored me during this life-changing journey and helped me overcome the unexpected challenges I faced to get my Ph.D.

I want to start by expressing my gratitude to Professor ***Nabil El Akkad***, my PhD advisor. His ongoing assistance, direction, and encouragement have been helpful throughout the procedure. His constant presence and abundance of knowledge have played a crucial role in guiding my academic development from the early phases of fine-tuning my research proposal to the final submission of my thesis. Our sessions were valuable since they offered me plenty of encouragement and important milestones to keep me on track academically. I am incredibly appreciative of his immense contributions to my growth.

I am immensely appreciative of the chance to be a member of a world-class research organization that offers profound insights into seismology. I want to express my sincere gratitude to my co-thesis director, Professor ***Mohamed Benslimane***, for kindly sharing his vast academic expertise and helping me become established in the academic world. I owe him a sincere debt of gratitude for his firm yet kind direction, which not only enabled me to make substantial contributions to publications throughout my Ph.D. but also unlocked doors I never imagined were even feasible. Moreover, I seize the chance to thank Professor ***Khalid SATORI***, Professor ***Abdellatif EZZOUHAIRI***, Professor ***Mohammed OUANAN***, Professor ***Hassan SILKAN***, Professor ***Abdelhak BOULAALAM***, and Professor ***Khalid HADDOUCH*** to be present and give me the honor of attending and examining my Ph.D.

I want to sincerely thank my family for their support and belief in me. My successes were greatly influenced by your support. No words in the world can adequately convey my love and admiration for you ***Mom*** and ***Dad***. My two brothers ***Salah*** and ***Nabil***, please accept my sincere gratitude for all of your support and assistance. This Ph.D. thesis is dedicated to you.

Words cannot describe how much my Uncle ***Nacer*** and Aunt ***Fatiha***'s steadfast faith in me has meant to me, and I am so grateful for your love and encouragement.

Abstract

Tasks like image segmentation and object recognition have significantly improved due to computer vision's quick development. These tasks are essential for many applications, including autonomous driving, imaging in the medical field, and surveillance. However, current methods frequently consider object recognition and image segmentation as distinct operations, which creates inefficiencies and difficulties in real-time applications when both tasks must be completed concurrently. The main purpose of this thesis focus on image segmentation and object detection. The first task we dealt with was image segmentation, dividing the input image into different areas. In another way, it separates the objects from the background to detect some specific objects. therefore, in this thesis, we aim to present and explain the work done in the field of image segmentation, image classification, and object detection. During the last four years, we accomplished several tasks, proposed different methods for better accuracy, and tried to solve several issues in the computer vision field. The first published article was about a novel approach of medical imaging for spotting brain cancers in MRI images. We employed two methods for this. The first one is Birch, which groups and reduces data quickly using a hierarchical structure. The segmentation by Watershed is the second algorithm. Then, using Otsu's thresholding algorithm and the marker-controlled watershed technique, we developed a new method for skin cancer diagnosis in the second suggested approach. We then provided a thorough and accurate analysis of reinforcement learning methodologies and their influence on the area of image processing in the third proposed strategy. The fourth proposed method was a text detection and recognition approach employing deep learning, image segmentation methods, and an NLP technique to extract and recognize text from license plates. We employed a number of well-known assessment measures, including dice coefficient, sensitivity, specificity, the Jaccard similarity, F1 score, and recall to assess the effectiveness of the suggested methodologies. Additionally, we contrasted the suggested strategies with the SOTA techniques. The experiments produced highly positive outcomes for each strategy that was suggested.

Keywords: Medical imaging, Image segmentation, Thresholding, Reinforcement learning, Deep learning.

Résumé

Les problématiques telles que la segmentation des images et la reconnaissance d'objets se sont considérablement améliorées grâce au développement rapide de la vision informatique. Ces domaines sont essentiels pour de nombreuses applications, y compris la conduite autonome, l'imagerie médical et la surveillance. Le but principal de cette thèse se concentre sur la segmentation des images et la détection d'objets. Les problématiques que nous avons traitées étaient la segmentation et la classification des images ainsi que la détection d'objets. Au cours des quatre dernières années, nous avons accompli plusieurs travaux, proposé différentes méthodes pour une meilleure précision, et essayé de résoudre plusieurs problèmes dans le domaine de la vision par ordinateur. Le premier article publié est sur une nouvelle technique d'imagerie médicale pour détecter les tumeurs cérébrales dans les images MRI. Pour cela, nous avons utilisé deux algorithmes, le premier est Birch qui est une méthode hiérarchique rapide de regroupement et de réduction des données. Le deuxième algorithme est la segmentation par Watershed. Ensuite, dans la deuxième approche proposée, nous avons présenté une nouvelle méthode de détection du cancer de la peau basée sur l'algorithme de seuil d'Otsu et la méthode de Watershed. Par la suite, dans la troisième approche suggérée, nous avons donné un examen complet et précis des approches d'apprentissage du renforcement et de leurs impacts sur le domaine du traitement d'images. La quatrième méthode proposée était une approche de détection et de reconnaissance de texte en utilisant l'apprentissage en profondeur, des méthodes de segmentation d'images et une technique de traitement des langages naturels pour extraire et reconnaître le texte des plaques d'immatriculation. Pour évaluer les performances des approches proposées, nous avons utilisé une série de mesures d'évaluation bien connues telles que le coefficient de cube, la sensibilité, la spécificité, la précision, le coefficient de corrélation, F1-score, la similitude Jaccard… En outre, nous avons également comparé les approches proposées avec plusieurs méthodes bien reconnues dans la littérature. Les expériences ont donné des résultats très satisfaisants pour chaque méthode proposée.

Mots clés : Imagerie médicale, segmentation d'images, Seuillage, Apprentissage par renforcement, Apprentissage en profondeur.

Table of contents

Abstract…..... i
Résumé…..... ii
Table of contents ……..... iii
List of Acronyms ……..... vi
List of Figures ……..... vii
List of Tables ……..... viii
List of Algorithms ……..... ix
General introduction ……..... 1
Contributions ……..... 4
Manuscript organization ……..... 4
Chapter 1: State of the art of image segmentation, object detection, and text recognition... 6
1.1 Introduction …… 6
1.2 Computer Vision ……..... 6
1.3 Preprocessing ……..... 9
1.4 Image segmentation ……..... 11
1.4.1 Classic segmentation methods……..... 13
1.4.1.1 Edge detection…… 13
1.4.1.2 Edge detection using derivatives …… 14
1.4.1.3 Region division ……... 16
1.4.1.4 Clustering ……..... 20
1.4.1.5 Thresholding ……... 22
1.4.2 Co-segmentation methods……..... 25
1.4.3 Semantic segmentation based on deep learning……... 27
1.4.4 Evaluation metrics ……..... 42
1.5 Object detection ……..... 44
1.6 Image classification……..... 48
1.7 Text recognition ……..... 49
1.8 Conclusion …… 50
Chapter 2: Mathematical tools and algorithms for image segmentation, object detection, and text recognition…… 52
2.1 Introduction ……..... 52

2.2 Algorithms used in the first proposed method: “A brain tumor segmentation and detection technique based on Birch and Marker watershed” ……. 52
2.2.1 Median filter……..... 52
2.2.2 Fuzzy c-means ……..... 54
2.2.3 Multi Otsu ……..... 57
2.2.4 Evaluation criteria ……..... 60
2.3 Algorithms used in the second proposed method: “A hybrid skin lesions segmentation approach based on image processing methods” ……. 63
2.3.1 Birch ……... 63
2.3.2 Marker watershed ……..... 67
2.3.3 Thresholding ……..... 70
2.3.4 Morphological operations ……..... 71
2.3.5 Evaluation metrics ……..... 78
2.4 Algorithms used in the third proposed method: “Text extraction and recognition method for license plates” ……... 78
2.5 Algorithms used in the fourth proposed method: “Reinforcement learning” …… 80
2.6 Conclusion ……..... 83
Chapter 3: New algorithms for image segmentation and object detection ……..... 83
3.1 Introduction ……..... 83
3.2 A brain tumor segmentation and detection technique based on Birch and Marker watershed……..... 84
3.2.1 Introduction ……..... 84
3.2.2 Proposed method ……..... 85
3.2.3 Results ……..... 86
3.2.4 Discussion ……..... 89
3.3 A hybrid skin lesions segmentation approach based on image processing methods...……... 90
3.3.1 Introduction ……..... 90
3.3.2 Proposed method ……..... 92
3.3.3 Results and discussion ……..... 93
3.4 Text extraction and recognition method for license plates…… 95
3.4.1 Introduction ……..... 95
3.4.2 Proposed method ……..... 97

3.4.3 Results 98
3.5 Conclusion 98
Chapter 4: Reinforcement learning 100
4.1 Introduction 100
4.2 Reinforcement learning models 102
4.2.1 Model-based in reinforcement learning 102
4.2.2 Model-free in reinforcement learning 104
2.2.3 Main two models' differences 104
4.3 Multi-armed bandits 107
4.4 Markov decision process 111
4.5 Dynamic programming 114
4.6 Monte Carlo methods 115
4.7 Temporal difference learning 117
4.8 Conclusion 121
General conclusion and perspectives 122
Publications 125
References 127

List of Acronyms

AI	Artificial Intelligence
CNN	Convolutional Neural Network
R-CNN	Region-based Convolutional Neural Network
SIFT	Scale Invariant Feature Transformation
ReLU	Rectified Linear Unit
FCN	Fully Convolutional Network
NLP	Natural Language Processing
U-Net	A Convolutional neural network that was created for biomedical image segmentation
DeepLabv3	A semantic segmentation architecture that enhances DeepLabv2
PspNet	Pyramid Scene Parsing Network
ENet	Efficient Neural Network
SegNet	A Deep Convolutional Encoder-Decoder Architecture
LinkNet	Exploiting Encoder Representations for Efficient Semantic Segmentation
HRNet	High-Resolution Network
RL	Reinforcement Learning
TDL	Temporal Difference Learning
OCR	Optical Character Recognition
POS	Part-of-Speech Tagging
NLG	Natural Language Generation
UCB	Upper-Confidence-Bound

List of Figures

Figure 1.1: Image segmentation methods 12
Figure 1.2: Different types of edges 13
Figure 1.3: Intensity function and the first derivative 14
Figure 1.4: K-means clustering approach 22
Figure 1.5: FCNs architecture 27
Figure 1.6: U-net Architecture 29
Figure 1.7: DeepLabV3 architecture 31
Figure 1.8: PSPNet architecture 32
Figure 1.9: ENet architecture 33
Figure 1.10: SegNet architecture 35
Figure 1.11: LinkNet architecture 36
Figure 1.12: HRNet architecture 38
Figure 1.13: Transformers, GPT, BERT architecture 42
Figure 1.14: Object detection flowchart 45
Figure 1.15: Image Classification steps flowchart 49
Figure 2.1: The median filter flowchart 53
Figure 2.2: Fuzzy c-means flowchart 56
Figure 2.3: Multi Otsu's flowchart 59
Figure 2.4: Birch algorithm diagram 66
Figure 2.5: Marker watershed flowchart 68
Figure 3.1: Flowchart of the first proposed method 86
Figure 3.2: Obtained results for the first proposed method..... 88
Figure 3.3: Flowchart of the second proposed method 92
Figure 3.4: Obtained results for the second proposed method 93
Figure 3.5: Third proposed method..... 97
Figure 4.1: Policy optimization process..... 101
Figure 4.2: Process of the agent-environment 112
Figure 4.3: Monte Carlo prediction 117
Figure 4.4: TD (0) 118
Figure 4.5: Trajectory of an experience 119

List of Tables

Table 3.1: Comparison of the proposed technique and the two segmentation methods for the first image in Figure 3.2……..……..... 88
Table 3.2: Comparison of the suggested technique and the two segmentation methods for the second image in Figure 3.2……... 88
Table 3.3: Comparison of the suggested approach and some other methods…… 89
Table 3.4: Obtained results for image 1……. 94
Table 3.5: Obtained results for image 2……..... 94
Table 3.6: Obtained results for image 3……. 94
Table 3.7: Obtained results for image 4……. 94
Table 3.8: Obtained results for image 5…… 95
Table 3.9: Obtained results for image 6…… 95
Table 3.10: Obtained results ……. 98
Table 4.1: Reinforcement learning models …… 107

List of Algorithms

Algorithm 2.1: Median filter algorithm ……..... 54
Algorithm 2.2: Fuzzy c-means algorithm……..……..... 57
Algorithm 2.3: Multi Otsu's algorithm ……. 60
Algorithm 2.4: Birch algorithm ……..... 67
Algorithm 2.5: Marker watershed algorithm ……..……..... 69
Algorithm 2.6: Thresholding algorithm ……...…..... 71
Algorithm 2.7: OCR algorithm ……..……..……..... 79

General introduction

Data presented visually, for as through images, films, or other visual media, is referred to as visual information. It includes all the specifics, trends, and connections in visual material. Visual information may convey complex ideas and feelings that may be difficult to describe through words or numbers alone, making it a potent tool for comprehension and communication. Visual information is a core component of human vision, and understanding it has long been a problem for artificial intelligence. A subfield of AI called computer vision [1] seeks to improve and mimic how the human visual system processes and understands visual information. The importance of visual information, the function of computer vision [2] in its analysis, applications, difficulties, and hopes for the future. Furthermore, there is a lot of visual information all around us. It consists of images, videos, and other visual data that convey crucial information about things, situations, connections, and context. Visual signals are a natural way for humans to perceive their surroundings, make decisions, and communicate. Visual information is a potent tool for information transfer because it offers critical insights that word or numerical data alone may not be able to capture. Additionally, machine learning [3], especially deep learning [4], has greatly advanced computer vision by enabling algorithms to automatically learn and recognize patterns, features, and relationships within visual data. Computer vision encompasses tasks such as image recognition, object detection, scene understanding, and more. It addresses the complexity of visual information interpretation using AI techniques. The uses of computer vision are numerous and varied, it can be found in autonomous vehicle applications as well as in the healthcare domain. Computer vision is widely used in security and surveillance systems and so one. On the other hand, in order to detect and separate unique objects or regions within an image, which makes subsequent analysis and interpretation more manageable, image segmentation is a key building block of many computer vision applications. Furthermore, the ability to identify and isolate specific regions within images, which is essential for many applications that call for understanding and interpreting visual data, makes image segmentation a fundamental technique within the broader field of computer vision. Recent Developments in image segmentation [5] assist in the advancement of more sophisticated and capable computer vision systems throughout various domains. There are several kinds of image segmentation methods [6], each with a distinct strategy and set of applications that will be covered afterward.

Despite significant advancements in segmentation techniques, a number of issues persist. These challenges are brought on by the data disparities, the images' inherent complexity, and the flaws in the present algorithms. Some of the key challenges in image segmentation are cited below:

- *Object Variability:* The size, shape, color, texture, and orientation of objects inside images can differ significantly. It is difficult to create algorithms that can precisely divide items with this level of diversity.
- *Object Occlusion:* It becomes challenging to distinguish between objects in an image when they overlap or are occluded from view. Context and connections must be understood in order to segment obscured items.
- *Fine Details:* Especially in images with significant quantities of noise or poor resolution, capturing small details and intricate patterns inside objects can be difficult.
- *Complex Backgrounds:* Images frequently include complex backgrounds with a variety of textures, hues, and structures. It can be difficult to distinguish items from crowded backgrounds, especially when the object and backdrop have similar characteristics.
- *Partial Visibility:* On sometimes, an image may only show a portion of an item. Such objects require algorithms to deduce their whole form and limits.
- *Ambiguity:* Some items may have unclear borders or resemble their surroundings visually. It might be difficult to determine where the backdrop begins and the item finishes.
- *Lack of Annotated Data:* To train reliable segmentation models, you need a lot of data labeled with the segments each pixel belongs to. Large dataset annotation may be time- and money-consuming.
- Semantic segmentation: The complexity of semantic segmentation arises from the need for models to comprehend both local and global contexts, which can be challenging in images containing a variety of objects and scenarios.
- Instance Segmentation: Individual instances of the same object class must be distinguished, which is much more difficult with instance segmentation because it calls for precise object boundary identification and overlap resolution.
- *Computational intensity:* Many cutting-edge segmentation algorithms, particularly those based on deep learning, demand a lot of computing power, which makes real-time processing difficult.

- *Generalization:* It's essential to check if segmentation models translate effectively to fresh, unexplored data. Models may perform badly on new images if they are overfitting the training set.
- *Biases and Diversity:* Diverse and underrepresented groups may have difficulty being effectively segmented by models due to prejudice in the training data.
- *Adversarial examples:* Certain segmentation models are vulnerable to adversarial assaults, in which minute, skillfully engineered perturbations result in inaccurate segmentation outcomes.
- *Interclass Variation:* Accurately segmenting all instances might be difficult when objects belonging to the same class have noticeable variances (such as various dog breeds).

Ingenious computational strategies, increased data quality and variety, and a thorough comprehension of the nuances of image content and context are all necessary to address these issues. It's conceivable that answers to these problems will help provide more reliable and accurate image segmentation techniques as computer vision research advances. On the other hand, medical imaging segmentation [7], which involves delineating and identifying certain structures or regions of interest within medical images, is an essential activity in healthcare and medical research. Although machine learning and technological developments have greatly enhanced segmentation approaches, there are still many difficulties that researchers and professionals in this field must overcome. Among these challenges, we can cite the following main tasks:

- *Complex Anatomy and Variability:* Because human anatomy varies greatly between people, it is challenging to create a segmentation algorithm that works for everyone. The effectiveness of segmentation models can be impacted by variations in the size, shape, and appearance of organs and structures.
- *Resolution and Noise:* The effectiveness of segmentation algorithms can be impacted by the resolution and noise of medical images, both of which can vary. Segmentations can be incorrectly performed for images with low resolution or images with a lot of noise.
- *Uncertainty and Ambiguity:* The uncertain borders, fuzzy edges, and overlapping structures that might be present in medical images make it difficult for segmentation algorithms to make accurate conclusions.

Contributions:

To overcome these challenges, we proposed two contributions in the image segmentation and object detection field, a third contribution in text detection and recognition, and a fourth one in reinforcement learning:

- ***A Brain Tumor Segmentation and Detection Technique Based on Birch and Marker Watershed**:* This contribution's main purpose is to detect and segment brain tumors in MRI images using a hybridization between Birch and Marker watershed methods.
- ***A hybrid skin lesions segmentation approach based on image processing methods**:* This contribution's main objective is to detect and segment tumor areas in the skin using a hybridization between Otsu's thresholding algorithm and the marker-controlled watershed method.
- ***License plate text recognition using deep learning, NLP, and image processing techniques:*** This third contribution is about the detection and recognition of license plate text. For this, we used a hybridization between the Yolo v7 deep learning method, thresholding, and Morphological operations. Afterward, we employed the function find to search for a specific character in each detected license plate. This work was subsequently developed using deep learning, especially the Yolov7, and was accepted in the "SOIC" journal.
- ***Reinforcement Learning: A review**:* In this paper, we proposed a review of reinforcement learning methods and how they can be applied to solve image segmentation problems.

Manuscript organization:

The thesis will be organized as follows:

Chapter 1: ***State of the art of image segmentation, object detection, and text recognition***

The first chapter of this thesis is constituted of four essential parts which are image segmentation, object detection, image classification as well as text detection and recognition. These three main sections are going to be presented in detail, including the pros and cons of every approach. Moreover, evaluation metrics for each method are given in detail. Moreover, we listed the evaluation criteria for each

On the other hand, a previous clarification and explanation of computer vision and image processing is given as well.

Chapter 2: mathematical tools and algorithms

This chapter is dedicated to presenting in detail the different approaches used in the several proposed contributions in this thesis, especially for segmenting and detecting objects in the images, as well as text detection and classification. Moreover, a very precise and comprehensive review of reinforcement learning is given. Furthermore, it also outlines the mathematical techniques and assessment standards that we employed throughout the experimental phase to assess and demonstrate the value of the solutions suggested in this thesis.

Chapter 3: ***Image Segmentation and object detection - Text extraction and recognition***

This third chapter outlines the key components of approaches created in the field of image segmentation. It also displays experimental results obtained by applying established methods to real images taken from reference datasets. Additionally, every experiment compared the effectiveness of our remedies to the most well-known literary solutions. Furthermore, we will be focusing on text detection and extraction in license plates using deep learning to detect the license plate, and some image segmentation methods to segment the detected license plate. Afterward, a text file will be generated in which we applied some function to look for a specific character to classify the license plate.

Chapter 4: ***Reinforcement Learning***

The fourth chapter is consecrated for the last contribution which is about a reinforcement learning review. Moreover, we gathered all the methods used in the literature. Multi-armed bandits, the Markov decision process, Monte Carlo methods, dynamic programming as well as temporal-difference learning are some of the corresponding methods used to solve reinforcement learning issues.

General conclusion and perspectives

In the conclusion and perspectives chapter, we're going to summarize the different contributions presented in this thesis, as well as contributions and future work.

Chapter 1:

State of the art of image segmentation, object detection, and text recognition

1.1 Introduction

Basic computer vision tasks like image segmentation and object recognition are essential for many different applications, including autonomous vehicles, medical imaging, surveillance, and robots, on the other hand, object detection which is defined as the process of object detection involves finding and locating various items inside an image while providing every item that is found a matching label or class. In this chapter, we will analyze these ideas and give a detailed overview of image segmentation and object recognition.

1.2 Computer vision

The definition of computer vision [8] is that it is the study of visual data, which is a very important issue. In the last several years, there has been a significant explosion in the amount of visual data in our environment, which is partly due to the widespread use of sensors [9]. Nowadays, the majority of individuals own one or more smartphones, and each smartphone contains one, two, or even three cameras. As a result, and rationally speaking, there are generally more cameras than people in the globe. These sensors are responsible for the massive daily production of visual data around the planet. Computer vision is described as the study of visual data, which is a very significant topic [8]. The amount of visual data in our surroundings has significantly increased over the past several years, which is partially attributable to the increasing usage of sensors [9]. Most people nowadays own one or more cell phones, and each one comes equipped with one, two, or even three cameras. Consequently, and logically, there are typically more cameras than people in the world. These sensors are in charge of the planet's daily creation of vast amounts of visual data. Because we must comprehend optics, image creation, and how images are physically created, we touch on fields like physics while discussing computer vision. Researchers need to comprehend how animal brains really perceive and process visual data in order to make advances in biology and psychology. The early 1960s saw the beginning of computer vision as well. Block World is a collection of papers written and published by Larry Roberts [11] and is regarded as one of the earliest Ph.D. theses in computer

vision. The visual world was reduced to basic geometric shapes with the intention of learning to identify and rebuild what these shapes are. Additionally, there was a project called "The Summer Vision Project" whose major objective was the development of a substantial component of a visual system. David Marr [12], a vision scientist at MIT, took another significant step forward when he wrote an influential book in the late 1970s about what he believed to be vision and how we should approach computer vision and the creation of algorithms that would allow computers to recognize the visual world. The idea behind his book is that we must go through a number of procedures in order to take an image and produce the ultimate comprehensive complete 3D representation of the visual world. The edges, bars, endings, virtual lines, curves, and limits are mostly portrayed in the first step, which he dubbed "Primal Sketch." The second step is what he referred to as a "2 ½ D Sketch," where he began to piece together the surfaces, depth information, layers, or discontinuities of the visual scene. Eventually, he put everything together and had a 3D model that was hierarchically organized in terms of surface and volumetric primitives, and so on.

Computer scientists started thinking about how is it possible to recognize and represent objects in the 1970s when people started asking the question, "How can we move beyond the simple block world and start recognizing or representing real-world objects?" Two ideas were proposed, one is called a Generalized Cylinder and the other is called a Pictorial Structure. Every item is made out of basic geometric primitives, according to the fundamental principle. Either representation may be used to break down the object's intricate structure into a set of smaller forms and their geometric arrangements. In the 1980s, David Lowe [13] developed a novel method for identifying razors by generating lines, edges, and mostly straight lines and combining them. Even though it has developed into one of the most significant and rapidly expanding fields of artificial intelligence, hundreds of academics are still working on some of the most fundamental challenges of vision that haven't yet been addressed. As a result, object segmentation should be carried out first if object detection is too difficult. The process of obtaining an image and classifying the pixels into useful regions is called object segmentation [14]. One of the earliest attempts, Normalized Cut from 1997, used a graph theory technique to address the issue of image segmentation. By the year 2000, face detection, another issue in computer vision, had become apparent.

One of the most significant items to people historically has been the face. Machine learning methods, particularly statistical machine learning methods, began to take off between 1999 and 2000. These methods include the initial wave of neural networks as well as support vector

machines, boosting, and graphical models. Additionally, Paul Viola and Michael Jones' use of the AdaBoost algorithm to do real-time face identification was one particular effort that made a lot of advances. This study was completed in 2001 when computer chips could still only recognize faces in images in real-time but were still quite sluggish. Fujifilm therefore released the first digital camera with a real-time face detector five years after the publication of this study, marking a quick transition from fundamental scientific research to practical implementation. After that, David Low, the author of SIFT, brought the concept of matching a whole object to light. It was motivating to notice that some properties or components of the object tended to be diagnostic and unaffected by alterations. Since this is a simpler process than pattern-matching the full item, the first step in object identification was to identify these essential properties of the object and compare them to comparable objects. The theory behind the Spatial Pyramid Matching algorithm, which was also recently revealed, is that there are elements in images that may help us identify the sort of scene we are looking at, such as a landscape, a kitchen, or even a motorway. In this specific work, the support vector machine technique is used to combine these characteristics from various regions of the image at various resolutions into a feature descriptor. Similar to this, a nearly identical study known as the Histogram of Gradients as well as the Deformable Part Model has gained popularity in human recognition. On the other hand, computer vision has identified object identification as a key challenge to tackle as a building block. Beginning in the early 2000s, benchmark datasets were accessible, allowing researchers to track the advancement of object identification. Pascal Visual Object Challenge is one of the most well-known benchmark datasets.

The collection contains a few thousand to ten thousand images for each of the 20 object classes that make up Pascal. However, the issue of insufficient training data leads to overfitting, therefore generalization may not be carried out correctly. One of the most well-known datasets, ImageNet [16], has 14 million images and 22,000 types of items and scenarios. It started to move the algorithm development for object identification into a new stage and is likely the largest dataset ever generated in the field of AI [17]. On the other hand, the adoption of convolutional neural networks, or CNNs, which excelled in many difficulties, has been one of the factors that has fueled the advancement of the discipline in recent years. When a CNN was developed in the 1990s to automatically detect handwritten cheques or post office addresses, it became one of the key works in the field of convolutional neural networks. It operates by identifying the pixels in an image according to whether they were letters or numbers. The proposed network's design resembles that of the 2012-released AlexNet architecture [18]. Many

recent breakthroughs, such as computing, have contributed to the popularity of these systems. Faster computers, more transistors on processors, and the development of graphics processing units, or GPUs, have made it possible to crunch these computationally demanding convolutional neural network models. GPUs are a fantastic tool for this. As a result, greater computing has been crucial in the development of deep learning. Data was another significant advance between now and the 1990s; these algorithms are quite data-demanding and require annotated images as input.

1.3 Preprocessing

An essential stage in computer vision and image analysis tasks is image preprocessing ([18], [19]). It entails using a variety of ways to improve, tidy up, and get images ready for additional analysis or machine learning models. By enhancing the data's quality and dependability, image preprocessing aims to produce better outcomes for subsequent tasks including object detection, image classification, segmentation, and other related ones. Here are a few typical techniques for image preprocessing:

- **Scaling and resizing**

Images may require to be scaled to a uniform resolution, which machine learning models frequently need. To improve convergence during training, scaling can also be employed to standardize pixel values between a specific range (for example, 0 to 1).

- **Normalization**

Normalizing pixel values, which is often done by removing the mean and dividing by the average deviation of pixel values, helps to normalize the input data. As a result, the model may become less sensitive to changes in lighting and color during training.

- **Grayscale conversion**

In some situations, converting images to grayscale might help to streamline the data and lower the processing's computing complexity. This is especially helpful when a task does not require color information.

- **Contrast Enhancement**

Tools like adaptive histogram equalization and histogram equalization can increase an image's contrast, bringing out details and enhancing the quality of future analysis.

- **Noise Reduction**

Image noise can be decreased using methods including bilateral filtering, median filtering, and Gaussian blurring. These techniques aid in removing slight fluctuations in pixel values that could be brought on by sensor noise or other elements.

- **Edge Detection**

For tasks like object detection and segmentation, edge detection techniques like Canny, Sobel, or Laplacian can be used to highlight the edges of objects in an image.

- **Image Rotation and Flipping**

Images may need to be rotated or flipped to properly align objects or to increase the diversity of training data.

- **Data Augmentation**

By applying transformations like rotations, translations, flips, and brightness modifications to images, data augmentation entails producing new versions of the original images. The generalization of models and overfitting can both be improved by data augmentation.

- **Thresholding**

By turning grayscale images into black and white based on a specific threshold value, thresholding is used to create binary images. To separate things from the background, this is frequently utilized.

- **Image Registration**

Image registration techniques can be used to match images obtained at various times or from various angles to a common coordinate system when working with many images.

- **Color Space Conversion**

By converting an image's color space (for example, from RGB to HSV or LAB), you can isolate certain elements of the image, such as hue, saturation, and brightness.

- **Removing Artifacts**

Unwanted artifacts like vignetting or lens distortion can occasionally be present in images. Geometric transformations and other methods of correction can be used to fix them.

The chosen preprocessing methods are determined by the particular task, the image properties, and the needs of the downstream analysis or machine learning model. It's crucial to test out several methods and assess how they affect how well a task is performed.

1.4 Image segmentation

Image segmentation [20] is defined as a subfield of artificial intelligence and computer vision. It's typically used to locate objects and boundaries in images, it's mostly used for partitioning an image into its regions based on some criteria where the regions are meaningful [21] and disjoint like finding a cat in an image and isolating it. However, before detecting the objects and even before classifying the image, it's required to understand what the image consists of, and this is the main purpose of image segmentation. We can divide or partition the image into various parts called segments [22]. However, it's not advised to process the entire image at the same time as there will be regions in the image which do not contain any information. Furthermore, by dividing the image into segments we can make use of the important segments for processing the image. Moreover, an image is a collection or a set of different pixels that we group in case of attribute similarities using image segmentation [23]. Therewith, image segmentation creates a pixel-wise mask for each object in the image, this technique gives a far more granular understanding of the objects in the image than object detection. Image segmentation is widely used in several real-world applications such as self-driving cars [24] where it's needed to know what the car sees and where it's exactly at all times. Another important use case of image segmentation is the medical imaging field. Medical imaging segmentation has played an important role in medical diagnosing once doctors need to reveal a disease before noticing any significant symptoms. Medical image segmentation [25] is defined as the process of automatic or semi-automatic detection of boundaries within a 2D or 3D, furthermore, the result of the segmentation can then be used to obtain further diagnostic insights. Possible applications for medical image segmentation are the automatic measurement of organs, cell counting, tumor segmentation, or simulations based on the extracted boundaries information. Moreover, image segmentation can be found in other important areas like self-driving cars, face recognition, as well as satellite imaging. Image segmentation is known for its different techniques that can be structured as shown in Figure 1.1 below:

Figure 1.1 Image Segmentation Methods

1.4.1 Classic segmentation methods

1.4.1.1 Edge detection [26]

Edges represent a rapid change in image intensity within small regions. Edges are caused by various physical phenomena such as depth discontinuity, surface normal discontinuity, surface reflectance discontinuity, and illumination discontinuity. Moreover, we can have four types of edges, the first one is the step edges where we can find the simple step function, a step edge with a gradient on either side, or a linearly changing gradient. The other two existing types of edges are the roof edge as well as the line edges as shown in Figure 1.2 below:

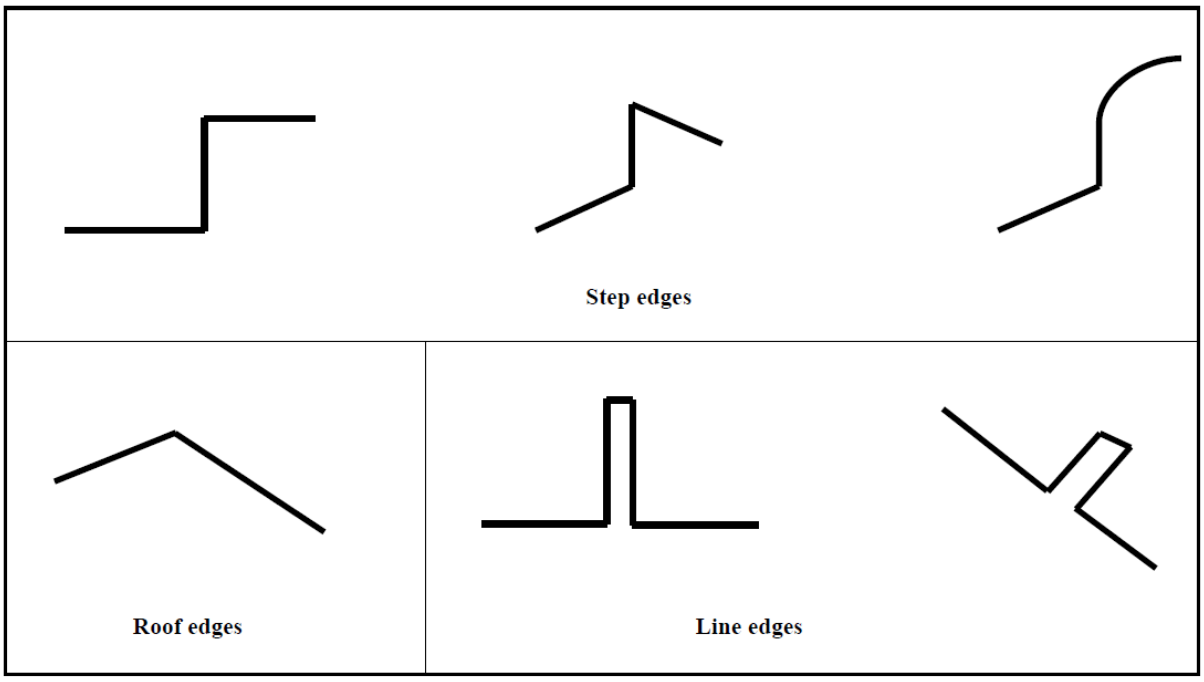


Figure 1.2 Different types of edges

The objective is to have an edge detector capable of finding the position of an edge, which doesn't necessarily mean just finding the pixel where the edge occurs, but also knowing where exactly the edge is with sub-pixel accuracy. This means where exactly inside the pixel does the edge lie. The second information is knowing what the magnitude of the edge is. Furthermore, the orientation of the edge in the image should be known as well.

As a result, an edge detector should give a high detection rate, which means figuring out if an edge exists or not. In other words, having low false positives and low false negatives. Moreover, good localization is also required in an edge detector, which means finding exactly where the edge takes place. Additionally, the resilience to noise is an asset for a good edge detector.

1.4.1.2 Edge detection using derivatives

Edges are represented by a rapid change in the image intensity function, which stands behind the steep cliff shape. Therefore, the main purpose is to find out where these steep cliffs exist in the image within a very small unit change in the pixels [27] that can be measured using a derivative or a gradient. Thereupon, by taking one particular row in the input image, the intensity function along the horizontal scaling will be as shown in the following figure. As we can see, the white color has a higher intensity and the black has a lower intensity. Consequently, a derivative of all the values in the intensity function looks similar as shown below. As we can see, there is an initial point where we may have a negative derivative because the value is falling down, and there's a later point where there's an equal magnitude of the gradient but in the opposite direction, and that's because the intensity increases at that particular point as shown in Figure 1.3:

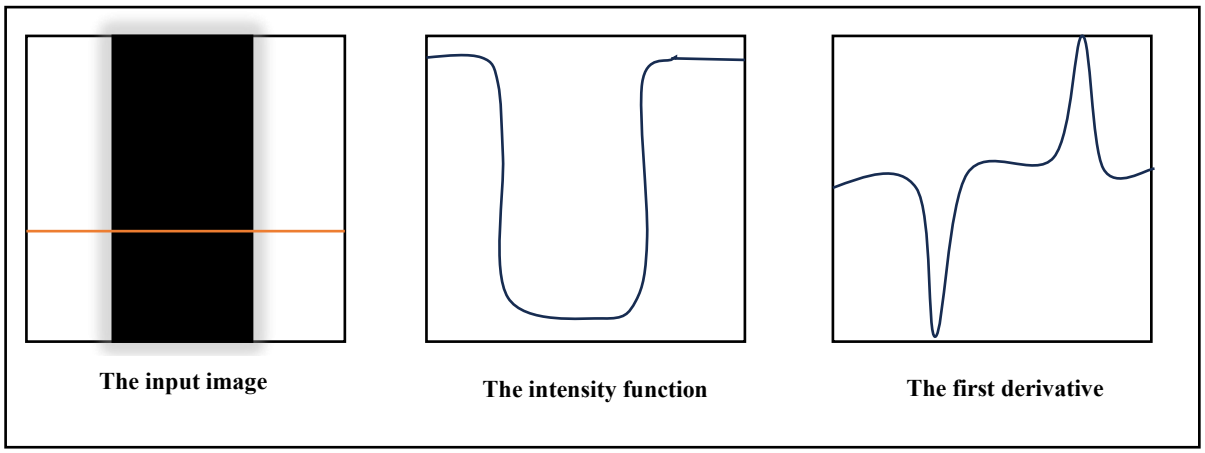


Figure 1.3 Intensity function and the first derivative

For a 2D function, f(x, y), the partial derivative can be written as follow:

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon} \tag{1.1}$$

With f (x, y) represents the intensity function of the image. For discrete data, we can approximate using finite differences:

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1,y) - f(x,y)}{1} \tag{1.2}$$

- **Robert operator**

Robert kernels [28] are derivatives concerning the diagonal elements that are known as cross-gradient operators that and based on cross-diagonal differences. Robert masks may be:

G_x= | 1 | 0 |
|---|---|
| 0 | -1 |

G_y= | 0 | 1 |
|---|---|
| -1 | 0 |

A generic gradient-based algorithm follows the main steps:

- **Step1**: Read the image and smooth it
- **Step2**: convolve the image f with G_x: $\widehat{f}$ (x) = f * G_x (1.3)
- **Step3**: convolve the image f with G_y: $\widehat{f}$ (y) = f * G_y
- Compute the edge magnitude and edge orientation
- Compare the edge magnitude with a threshold value; and if edge magnitude is higher, assign it a possible edge point.

- **Prewitt operator**

Prewitt method [29] takes the central difference of the neighboring pixels; the difference can be represented mathematically as:

$$\frac{\partial f}{\partial x} = [f(x+1) - f(x-1)]/2 \quad (1.4)$$

For 2D:

$$\frac{\partial f(x,y)}{\partial x} = [f(x+1,y) - f(x-1,y)]/2 \quad (1.5)$$

While the central difference is obtained by the mask: [-1 0 1]

G_x= | 1 | 0 | -1 |
|---|---|---|
| 1 | 0 | -1 |
| 1 | 0 | -1 |

G_y= | 1 | 1 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| -1 | -1 | -1 |

One more additional mask can be used to detect edges in diagonal as:

1	1	1
-1	0	1
-1	-1	0

-1	-1	0
-1	0	1
0	1	1

- **Sobel Operator:**

Sobel operator also relies on the central differences, it provides both a differentiating and a smoothing effect. Additionally, it can be viewed as an approximation of first Gaussian

Derivative, and the convolution is both commutative and associative that can be represented as:

$$\frac{\partial}{\partial x}(f * G) = f * \frac{\partial}{\partial x}(G) \tag{1.6}$$

G_x=

-1	0	1
-2	0	2
-1	0	1

G_y=

-1	-2	-1
0	0	0
1	2	1

Therefore, the magnitude of the gradient is given as follow:

$$G = \sqrt{(G_x^2 + G_y^2)} \tag{1.7}$$

Moreover, the orientation of the edge is given by:

$$\Theta = atan2(G_x, G_y) \tag{1.8}$$

An additional mask can be used to detect edges in diagonal as:

0	1	2
-1	0	1
-2	-1	0

-2	-1	0
-1	0	1
0	1	2

- **Canny detector**

The canny edge detector takes essentially a Sobel operator and makes it better. Therefore, the input of a Canny operator [30] is simply the output of Sobel. In the beginning, we start filtering the input image with a derivative of a Gaussian and generate a new kernel, which will give us a magnitude and orientation of the gradient. The third step is taking the magnitude and doing some local processing to enhance the edges. Afterward, the fourth step in the Canny operator is taking the gradient image and coming up with a method of linking and thresholding pixel groupings from one level to another.

1.4.1.3 Region division

The process of segmenting an image into meaningful [31] and coherent areas or segments based on specific criteria is known as region-based image segmentation. Region-based segmentation seeks to group pixels or image regions that have comparable features, such as color, texture,

intensity, or other visual attributes, as opposed to techniques that just concentrate on recognizing edges or contours. This method is particularly helpful for segmenting images that comprise objects of various complexity levels, sizes, and forms. The fundamental goal of region-based segmentation is to combine nearby pixels or regions that share some characteristics and meet certain requirements. These standards might be determined by texture patterns, color histograms, pixel intensity values, or a mix of these features. Steps like region growth, region splitting, and region merging are often included in the process.

- ***Split and merge segmentation***

In image segmentation, a procedure that includes dividing an image into useful segments or regions based on specific criteria, split and merge [32] are two often used techniques. These methods reduce complex images and make them easier to handle for additional processing or analysis. Let's examine each strategy in detail:

Splitting segmentation:

The act of splitting involves dividing a huge area or segment into smaller, easier-to-manage subareas. To manage intricate structures or objects in an image, this is frequently done. The goal is to divide a vast area into more manageable chunks that may then be separately analyzed or processed.

- ***Step 1***: Begin by dividing the entire image into segments.
- ***Step 2***: Determine whether the existing segment should be divided into more manageable subsegments by evaluating a condition (splitting criterion). This requirement could be based on similarities in hue, variations in intensity, textural traits, or other aspects of the image.
- ***Step 3***: Divide the current segment into smaller subsegments if the condition is true.
- ***Step 4***: Apply the splitting process iteratively to each of the produced subsegments up until there is no more room for splitting or a stopping condition is satisfied.

A test is required after each split to evaluate [33] whether the newly created region requires additional splitting. The homogeneity of the area serves as the test's criterion. Homogeneity can be defined in many ways; some examples include:

- If the grayscale levels in the area are constant or fall under a predetermined threshold, the region is homogenous [34].

- Local mean vs. global mean: A region is homogenous if its mean is higher than the mean of the entire image.
- The definition of the gray level variance is:

$$\sigma^2 = (1/(N-1)) \sum_{(r,c)\in R} [I(r,c) - \bar{I}]^2 \quad (1.9)$$

Where N is the number of pixels in the area, and r and c are the row and column, and $\bar{I}$ is represented as:

$$\bar{I} = (1/N) \sum_{(r,c)\in Region} I(r,c) \quad (1.10)$$

Merging segmentation:

On the other hand, merging [35] is the process of reuniting smaller sections or segments to create bigger, more cohesive regions. The merging phase follows the steps below:

- ***Step 1:*** Start combining neighboring parts after splitting them.
- ***Step 2:*** Identify whether adjacent subsegments should be merged by evaluating a condition (merging criterion). This requirement could compare features such as color similarity, physical proximity, or other pertinent features.
- ***Step 3:*** Merge the adjacent subsegments into a single segment if the condition is true.
- ***Step 4:*** Up until no further merging is possible based on the specified merger criterion, keep merging neighboring segments.

- ***Region growing segmentation***

A pixel-based segmentation technique known as "region growing" [36] begins with a seed pixel or group of seed pixels and iteratively expands the region by including nearby pixels that satisfy specific requirements.

The fundamental principle of region growing is to start with a seed pixel (or seeds) and incrementally add nearby pixels to the region if they share characteristics with those of the existing pixels. The resemblance criteria may be based on a number of elements, including the image's intensity levels, color, texture, or any other pertinent property.

Typically, the algorithm operates as follows:

- ***Step1***: Initialization; Pick one or more seed pixels that are a part of the area you want to divide.

- ***Step 2***: Growing Process: Examine the seed pixel(s)'s surrounding pixels after starting with them. A nearby pixel is included in the region if it meets the similarity requirement (for example, if its intensity is comparable to the seed pixel). Iteratively, this procedure is repeated, and newly additional pixels are incorporated into the expanding zone.
- ***Step 3***: Termination Criteria: The region grows until the similarity criteria prevents any further pixels from being added. This could occur [37] after a certain number of iterations, when a threshold is achieved, or when the growth process becomes unstable.
- ***Step 4***: Post-processing: Depending on the implementation, you might carry out post-processing operations to enhance the segmented regions, including merging.

The selection of seed locations and the resemblance criteria may be important factors in region growing. The algorithm may provide over- or under-segmentation depending on how well the seed points are selected or how well the similarity criteria are established. Achieving reliable segmentation results thus requires careful parameter tweaking and proper preprocessing procedures.

- ***Graph Cut*** **segmentation*:***

In computer vision and image processing, particularly in the context of image segmentation, graph cuts are a key concept. The graph-cut [38] approach represents the image as a graph, with pixels or superpixels acting as nodes and the connections between them acting as edges. The linkages and spatial organization between various elements of the image are captured by this graph structure. The fundamental concept behind graph cuts for image segmentation is to identify a partition of the graph (i.e., the image) into two or more distinct sets of nodes, such that the cut (the set of edges linking various sets of nodes) has the least amount of weight. The differences or costs between the connected nodes are represented by the weights on the edges.

$$cut(A, B) = \sum_{p \in A, q \in B} w_{pq} \tag{1.11}$$

Where A and B are two disconnected graphs

The objective is to reduce the total of these edge weights while ensuring that the resulting segments are coherent and significant. The "min-cut/max-flow" approach is the one that is most frequently used for graph cuts in image segmentation. By dividing the image into segments, this method seeks to determine the minimal cut in the graph. The method can determine sections that are probably different items or regions of interest by altering the weights on the edges based

on parameters like color, texture, gradient, and other image characteristics. Graph cuts have been particularly effective at segmenting images if the boundaries between items are clearly defined and stand out from their surrounds. They have been extensively applied in numerous fields, such as object recognition, scene comprehension, medical image analysis, and more.

1.4.1.4 Clustering segmentation

To organize related pixels or portions of an image into discrete clusters, the technique of clustering [39] is employed in image segmentation. Unsupervised machine learning techniques such as clustering algorithms can be used to segment images. Based on specific resemblance criteria, such as color, intensity, texture, or spatial closeness, these algorithms group pixels together. The objective is to group together pixels with similar properties, preferably denoting different objects or regions in the image. The K-means clustering technique [k-means] is a popular one for segmenting images. The algorithm's goal with this method is to divide the image pixels into a specific number of clusters. The process begins by choosing initial cluster centers, which are then iteratively updated to reduce the distance between pixels and the cluster center that is assigned to them. Each pixel is assigned to the cluster with the closest cluster center once the algorithm converges. Depending on the features of the data and the desired results, other clustering techniques, such as hierarchical clustering, DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [40], and Gaussian Mixture Models (GMM), can also be used for image segmentation. For tasks like object recognition, tracking, image editing, and applications involving computer vision, clustering in image segmentation can be especially helpful. It's crucial to remember that clustering methods may not always yield ideal results, particularly when there is a lot of overlap between items or complicated textures [41]. Therefore, in order to get accurate and reliable image segmentation findings, clustering is frequently combined with other approaches.

- ***K-means clustering*** **segmentation**

K-means clustering is a prominent unsupervised machine learning approach used for clustering or putting related data points into separate groupings [42]. It is frequently used for tasks like image segmentation in a variety of industries, including image processing and computer vision. K-means clustering algorithm works by following the main steps below:

- ***Step 1***: at this first stage, data preparation takes place where every pixel in the image is handled as a data point with its attributes, which are commonly exemplified by its color

values (for example, RGB values). For instance, each pixel in a colored image might be represented as a vector [R, G, B].

- ***Step 2***: Determining the Number of Clusters (K); Before executing the algorithm, the number of clusters, represented by "K," must be given. The goal of K-means is to divide the image into K unique segments.
- ***Step 3:*** K initial cluster centroids are picked at random from the feature space as the initialization. The clusters are represented by these centroids.

$$\mu_1, \mu_2, \ldots, \mu_k \in \mathbb{R}^n \tag{1.12}$$

- ***Step 4:*** The nearest centroid is assigned to each data point (pixel) based on a distance metric, most frequently the Euclidean distance. Around each centroid, groups of data points are formed in this step as shown in Figure 1.4.
- ***Step 5:*** Update Step; The centroids are updated by computing the mean of all data points assigned to each cluster once all data points have been assigned to clusters. The centroids are moved in this stage to more accurately depict the data points within their respective clusters.
- ***Step 6:*** Up until a stopping requirement is satisfied, steps 4 and 5 are recursively repeated. A maximum number of iterations or when the centroids don't vary noticeably between iterations could serve as the stopping criterion.

For each i:

$$c^{(i)} := arg\ \min_j ||x^{(i)} - \mu_j||^2 \tag{1.13}$$

For each j:

$$\mu_j := \frac{\sum_{i=1}^{m} 1\{c^{(i)} = j\} x^i}{\sum_{i=1}^{m} 1\{c^{(i)} = j\}} \tag{1.14}$$

- ***Step 7:*** The final centroids indicate the various segments or clusters in the image once the algorithm has reached convergence. The cluster linked to the closest centroid is then given ownership of each pixel.
- ***Step 8:*** The image can be segmented by giving a distinct color to each cluster, resulting in the division of the image into several parts depending on the similarity of pixel [43] values in each segment.

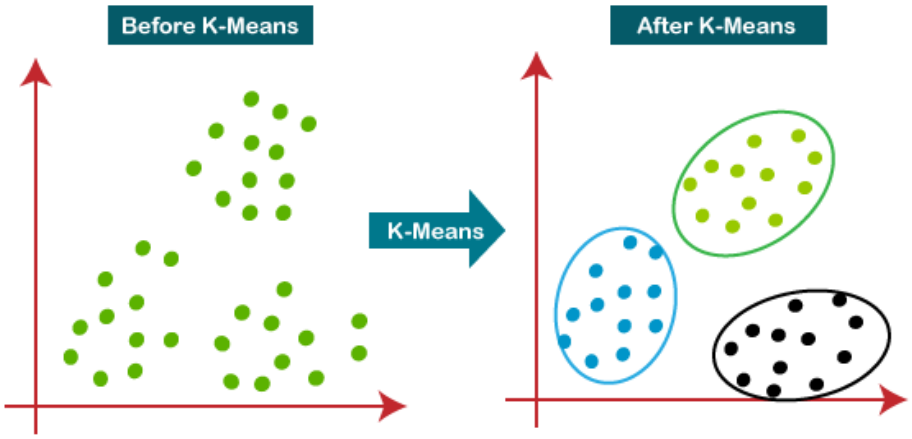


Figure 1.4 K-means clustering approach [43]

When an image comprises separate regions with comparatively uniform pixel values, K-means clustering can be a quick and efficient way for segmenting the image. K-means may, however, have trouble with complicated images, variable lighting, and situations where areas have overlapping or asymmetrical shapes. To further address these issues, other cutting-edge segmentation approaches have been created, including deep learning-based, graph-based, and watershed segmentation.

1.4.1.5 Thresholding segmentation

In the process of splitting an image into different sections or objects of interest, thresholding is a straightforward yet effective technique. Thresholding [43] is the process of establishing a threshold value for a grayscale or color image and categorizing pixels or regions according to whether their intensity or color values are above or below the threshold. As a result, pixels are categorized as either being in the background or the foreground (object of interest), creating a binary image. In other words, thresholding assists in distinguishing between the backdrop and the items you wish to recognize. The thresholding algorithm works by following the steps below:

- ***Step 1:*** Determining a Threshold Value; Depending on the properties of the image, a threshold value may be manually or automatically selected. A straightforward manual threshold value is sufficient if your image clearly distinguishes between the object and the backdrop in terms of intensity or hue. The best threshold can be determined automatically using techniques like Otsu's method in situations where the intensity distribution is more complicated.

- ***Step 2:*** The intensity or color value of each pixel is compared to the threshold once the threshold value has been established. A pixel is categorized as belonging to the foreground (an object) if its value exceeds the threshold and as belonging to the background if it doesn't.
- ***Step 3:*** A binary image is created as a result of thresholding, with the backdrop represented by black pixels with an intensity value of 0, and the object of interest represented by white pixels with an intensity value of 255.

For many different image processing applications, including object recognition, image enhancement, and feature extraction, thresholding is frequently utilized. It does, however, have some drawbacks. The segmentation may need to be more accurate because it is susceptible to changes in noise and brightness. To overcome these difficulties, cutting-edge methods like adaptive thresholding and multi-level thresholding have been developed.

There are various thresholding methods for segmenting images, each with a unique strategy and applicability for various image kinds and applications. Several common thresholding algorithms are listed below:

- **Global thresholding**

  - *Simple thresholding*: In this most basic type of thresholding [44], the entire image is subjected to a single threshold value. Foreground pixels are those with intensity values above the threshold, and background pixels are those below the threshold.

  - *Otsu's thresholding*: By minimizing the intra-class variance in the foreground and background regions, Otsu's approach determines an ideal threshold automatically. It works best when there is a distinct difference between the intensities of the object and the background.

- **Adaptive thresholding**

  - *Adaptive Mean thresholding*: This method determines a local threshold for each pixel based on the mean intensity of the pixels nearby. For images with various lighting circumstances, this is helpful.

  - Adaptive Gaussian Thresholding: this method [45] is equivalent to adaptive mean thresholding; however, it makes use of the weighted mean of the intensities of nearby pixels according to a Gaussian distribution.

- **Multi-level thresholding**

  - *Multi-Otsu thresholding:* This is a development of Otsu's multi-level segmentation approach. Multiple threshold values are automatically determined, dividing the image into different parts.

- **Hysteresis thresholding (Edge detection)**

  - *Canny edge detection:* A high threshold is used to identify strong edges, and a low threshold is used to identify weak edges, in the process of detecting canny edges. Strong edges are connected to weak edges, and these connections are kept.

- **Color-based thresholding**

  - *Color thresholding:* For thresholding in color images, color channels can be utilized in place of intensity. To segregate objects based on color, each color channel (such as red, green, or blue) can be independently thresholded.

- **Entropy-based thresholding**

  - *MaxEntropy thresholding:* With this technique, the threshold is chosen so that the entropy between the foreground and background regions is maximized. When the distribution of pixel intensities is unclear, it is helpful.

- **Mean-Shift thresholding**

  - *Mean-shift clustering thresholding:* In order to achieve convergence, this method clusters pixels according to their intensity values and shifts the cluster centers iteratively. After that, segmentation can be done using the clusters.

- **Local Binary Pattern thresholding**

  - *MaxEntropy thresholding:* Based on a comparison of each pixel's intensity with the pixels around it, this method determines a binary code for each pixel. The thresholding process then employs the binary patterns.

- **Kapur’s Entropy Maximization**

  - *Kapur’s Method:* Through the selection of the best threshold value, this technique increases the entropy between classes. It is beneficial for images with inconsistent brightness.

The properties of the image, the amount of noise, the illumination, and the intended segmentation result all influence the thresholding method selection. To determine the best thresholding technique for a given application, experimentation and tuning may frequently be required.

1.4.2 Co-Segmentation methods

When it comes to image segmentation, co-segmentation is the act of segmenting several images at once in a way that allows corresponding objects or regions to be recognized and segmented together. Co-segmentation is a technique for finding recurring items or geographic areas in a collection of linked images without the need for explicit annotations or object knowledge. By using attributes like color, texture, or shape, you can divide an image into various regions or objects in a standard image segmentation process. Co-segmentation [46], on the other hand, takes things a step further by taking into account a group of images with related subject matter. For instance, Co-Segmentation would try to segment all the cat regions in all the images as a coherent group if you had a collection of images with various cat images. Co-segmentation [47] is especially beneficial in a variety of applications, including:

- *Object Recognition*: By Co-segmenting images with identical content, you can raise the efficacy of object recognition systems since the segmentation outcomes produce reliable and persuasive information about object borders.
- *Video analysis*: By detecting the same items or regions in each frame, Co-segmentation can assist in tracking objects across numerous frames in a video.
- *Medical Imaging*: Co-segmentation can help in recognizing and following particular anatomical structures or anomalies across different slices of medical images like MRIs or CT scans.
- *Image editing*: Co-segmentation can help keep uniformity in the modified regions while working with a group of related images.
- *Content-Based Image Retrieval*: Co-segmentation is a technique that can be used to organize comparable images in image databases for quick retrieval.

Co-Segmentation algorithms [48] frequently use methods from image processing, machine learning, and computer vision. To locate similar items or regions throughout the input images, these algorithms may make use of both low-level properties (such color, texture, and shape) and high-level contextual information. These algorithms differ in their approach, level of difficulty, and efficiency. Several popular Co-Segmentation algorithms are listed below:

- *Graph Cut-Based Methods*: These algorithms define the Co-segmentation problem as an energy reduction task and model the images as a graph. Graph cut algorithms try to determine the best way to divide the graph into chunks that represent typical items. A well-known example of an algorithm that has been modified for Co-segmentation is the GrabCut method.
- *Clustering-Based Techniques*: These techniques, such as K-means clustering, spectral clustering, and normalized cuts, group similar image regions across numerous images by clustering pixel or feature representations.
- *Multi-Instance Learning Methods*: The goal of multi-instance learning algorithms is to identify which instances of an image correspond to the common objects. Each image is regarded as a collection of instances (patches or areas) in these algorithms. Global constraints are frequently used in multiple-instance learning techniques to ensure consistency between instances.
- *Submodular optimization techniques*: Submodular functions capture set coverage's declining returns. Co-segmentation can be thought of as a submodular optimization issue, with the objective being to choose a subset of areas from a set of images to maximize their combined coverage.
- *Methods Based on Deep Learning*: Convolutional Neural Networks (CNNs) have been used to do Co-Segmentation tasks since the development of deep learning. These techniques are capable of capturing high-level semantic information across numerous images and can learn complicated representations of image attributes. This category includes networks with a siamese structure and network topologies made for Co-Segmentation activities.
- *Graph Matching-Based Methods*: These algorithms seek to identify correspondences between comparable regions in various images by resolving a graph matching issue. They employ graph theory ideas to determine the best mapping between nodes representing regions in various images.
- *Saliency-Based Methods*: These techniques concentrate on locating objects or areas in a collection of images that stand out as salient. By considering the most recognizable regions across images, they leverage saliency cues to direct the Co-Segmentation process.
- *Markov Random Field (MRF) Models*: MRF models take into account the dependencies between nearby pixels or regions as well as contextual information. When co-

segmenting data using MRFs, the task is formulated as a probabilistic graphical model, and the ideal segmentation configuration is estimated.

Co-segmentation is a difficult task, particularly when dealing with a variety of illumination conditions, viewing angles, and occlusions. To increase the precision and effectiveness of Co-Segmentation procedures, researchers keep creating more complex algorithms.

1.4.3 Semantic segmentation based on Deep Learning

Each pixel in an image must be categorized into one of a set of predetermined classes in the computer vision task known as semantic segmentation [49]. The ability to create extremely accurate and effective models thanks to deep learning (DL) has greatly enhanced the field of semantic segmentation. The majority of modern approaches to semantic segmentation are built on convolutional neural networks (CNNs) [50]. To accurately classify and segment items inside images at the pixel level, semantic segmentation based on deep learning methods employs neural network architectures. Here are some well-known deep-learning techniques frequently employed in semantic segmentation:

- ***Fully Convolutional Networks (FCNs)***

One of the first structures created for semantic segmentation is the FCN [51]. To keep spatial information, convolutional layers are used in place of the completely linked layers found in conventional CNNs. Encoder-decoder structures are used by FCNs to record local and global context data. When combining features from several scales, skip connections are frequently used. The architecture of an FCN-8, which is a popular form of FCN for semantic segmentation, which is a Fully Convolutional Network (FCN) for semantic segmentation [52], designed to take an input image and produce a pixel-wise segmentation map as the output can be presented as shown in Figure 1.5 below:

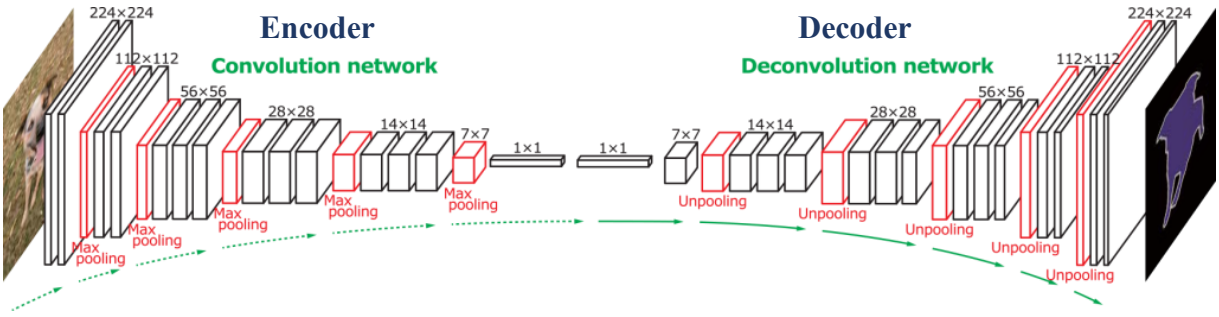


Figure 1.5 FCNs architecture [53]

- **Encoder structure**

Input layer: The network receives an image with varying dimensions as input.

Convolutional Layers: A succession of convolutional layers with increasing depth are utilized to extract hierarchical characteristics from the input image. These layers record various amounts of information from low-level textures to high-level object features.

Pooling Layers: Max-pooling layers are used to gradually reduce the spatial dimensions of the feature maps after each set of convolutional layers. As a result, the receptive field is widened and a hierarchical feature representation is produced.

- **Decoder structure**

Convolutional layer: To produce predictions with the same spatial resolution as the input image, a 1x1 convolutional layer is applied to the last feature map produced by the encoder. To prepare the network for class predictions, this layer applies a linear transformation.

Transposed Convolution (Upsampling): The feature maps are upsampled using transposed convolutional layers, also referred to as deconvolution or fractionally strided convolution. They preserve the number of channels while expanding the feature maps' spatial dimensions.

Skip Connections: Skip connections are included to link the decoder's upsampled layers to the encoder's lower-level feature maps. The accuracy of segmentation is increased and fine-grained information are retained thanks to these connections.

Element-wise summation: Combining high-resolution features with high-level contextual information, the upsampled feature maps are element-wise aggregated with the corresponding lower-level feature maps.

Upsampling steps: The original image's spatial dimensions are eventually restored through a series of iterations of the upsampling process.

Final output layer: The final pixel-wise segmentation map is produced by the decoder's last layer, the final output layer. This layer typically creates class probabilities for each pixel using a softmax activation function.

- **U-Net convolutional neural network**

Although U-Net [54] was created for biomedical image segmentation, it is also widely employed in other fields. It was developed in 2015 by University of Freiburg academics, and it

is distinguished by its distinctive U-shaped architecture [55], which combines an encoder and a decoder with skip links. Its architecture comprises of an encoder that contracts to record context and a decoder that expands symmetrically to pinpoint location precisely. The encoder and decoder's skip connections aid in maintaining fine details. The architecture draws its inspiration from the Fully Convolutional Network (FCN) idea, but it also incorporates a symmetric shape to help information move more easily across various levels of abstraction as shown in Figure 1.6:

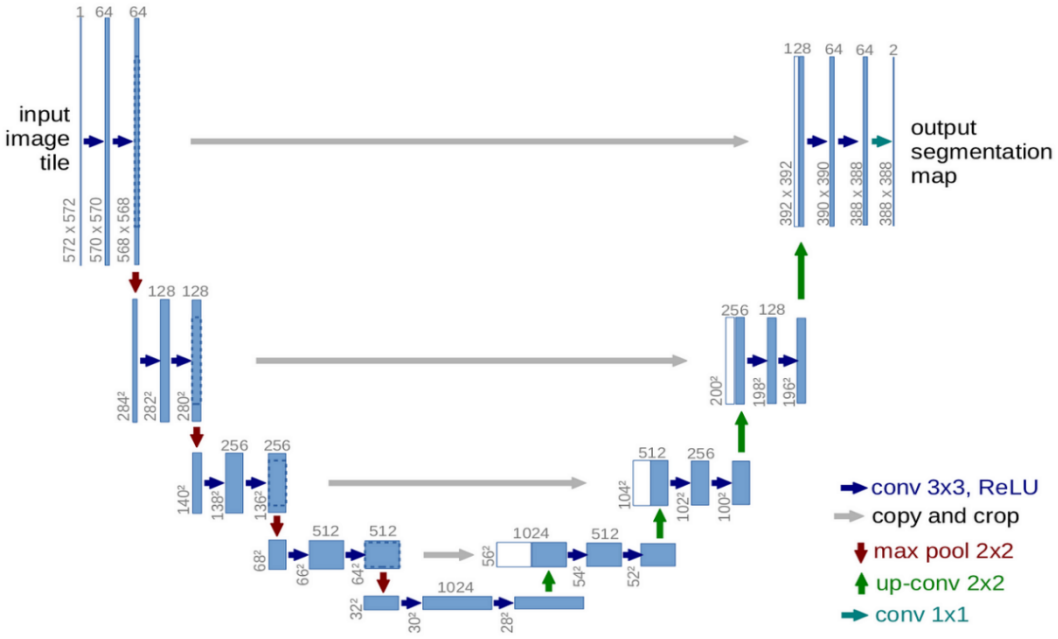


Figure 1.6 U-net Architecture [56]

The following elements make up the U-Net architecture:

- ***Contracting path (Encoder)***

- A succession of max-pooling and convolutional layers are used to process the input image.

- While the pooling layers minimize the spatial dimensions, convolutions capture features at various levels of abstraction.

- By moving from little details to more complex meanings, this approach successfully learns a hierarchy of features.

- ***Bottleneck layer***

- The center and bottom of the U shape are formed by this layer.

- Convolutional layers are usually used in stacks to retain spatial resolution while progressively lowering the number of channels.

-The encoder changes into the decoder at this stage.

- ***Expansive path (Decoder)***

- The decoder employs a number of transposed convolutional layers that upsample feature maps to improve their spatial resolution.

- A concatenation of feature maps from the relevant level in the encoder comes after each step of upsampling.

- These skip connections give the network the ability to combine high-level semantics from the decoder with fine-grained information from the encoder.

- ***Output layer***

The segmentation map is produced by the 1x1 convolutional layer that makes up the last layer of the U-Net architecture.

- A pixel-wise classification map is produced by this layer by using the softmax activation function to assign class probabilities to each pixel.

- ***DeepLab for semantic segmentation***

Google Research created the DeepLab [57] family of neural network topologies for semantic segmentation. In order to enable precise object segmentation, it seeks to precisely label each pixel in an image with a corresponding class label. DeepLab has undergone several iterations, each of which added new features and advancements. The focus on merging deep convolutional networks with cutting-edge methods to improve both accuracy and efficiency in semantic segmentation is a prevalent theme among DeepLab architectures [58].

The following are some important characteristics and facets of DeepLab architectures:

- *Atrous (Dilated) Convolutions:* Using atrous convolutions, also referred to as dilated convolutions, is one of DeepLab's distinguishing characteristics. These convolutions make it possible to incorporate more context while keeping the number of parameters and spatial resolution relatively constant. To record information on several scales, atrous convolutions with various rates are frequently utilized.

- *Multi-Scale Context Integration:* Different techniques are incorporated into DeepLab systems to capture context at different scales. In order to capture global context, additional techniques, such as spatial pyramid pooling (ASPP), are also used in addition to atrous convolutions with various dilation rates.
- *DeepLabV1:* For complex prediction tasks such as semantic segmentation, atrous convolutions were first implemented in the first iteration of DeepLab [59]. It used a fully convolutional network with extensive convolutions in its final layers to predict class labels.
- *DeepLabV2:* The Atrous Spatial Pyramid Pooling (ASPP) module was added in this revision. To capture multi-scale context information, ASPP employs atrous convolutions with various dilation rates. It enhances the network's capacity to identify things at various scales.
- *DeepLabV3:* Extending to DeepLabv2, this version added new enhancements such the usage of feature fusion and batch normalization methods. Atrous separable convolutions, which are more parameter-efficient, were also introduced in DeepLabv3.
- *DeepLabV3+:* By incorporating Atrous convolutions with the encoder-decoder structure, DeepLabv3+ expanded the design even further. To improve the recognition of minute details, a modified version of the Xception architecture was used as the feature extractor. The segmentation map was improved by the decoder using skip connections and bilinear upsampling as shown in Figure 1.7.
- *DeepLabV3++:* This iteration of DeepLab included a new module to DeepLabv3+ to improve the segmentation map. The final predictions are generated using upsampled feature maps and a sequence of 1x1 convolutions.
- *DeepLabV4:* A better ASPP variant known as "ASPP with Image-Level Features" (ASPP+ILF) was introduced in DeepLabv4. To increase segmentation accuracy, it merged multi-scale context from ASPP with global context from image-level characteristics.

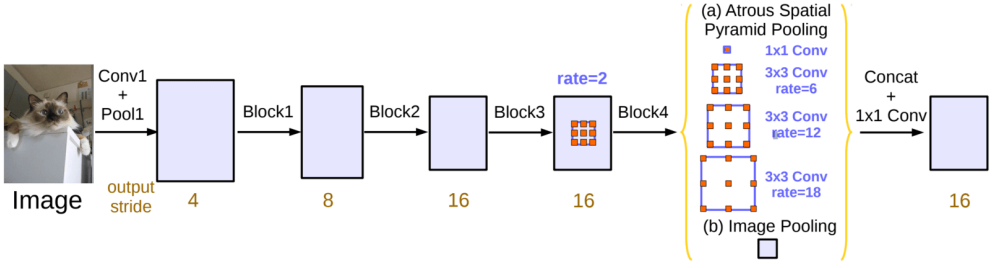


Figure 1.7 DeepLabV3 architecture [60]

- ***PSPNet (Pyramid Scene Parsing Network)***

The neural network architecture, PSPNet [61], or Pyramid Scene Parsing Network as presented in Figure 1.8, was created for semantic segmentation tasks. Researchers from the Chinese University of Hong Kong first proposed it in 2017. PSPNet's main innovation is implementing a pyramid pooling module to gather contextual data from various scales, improving the network's comprehension of scenes and objects.

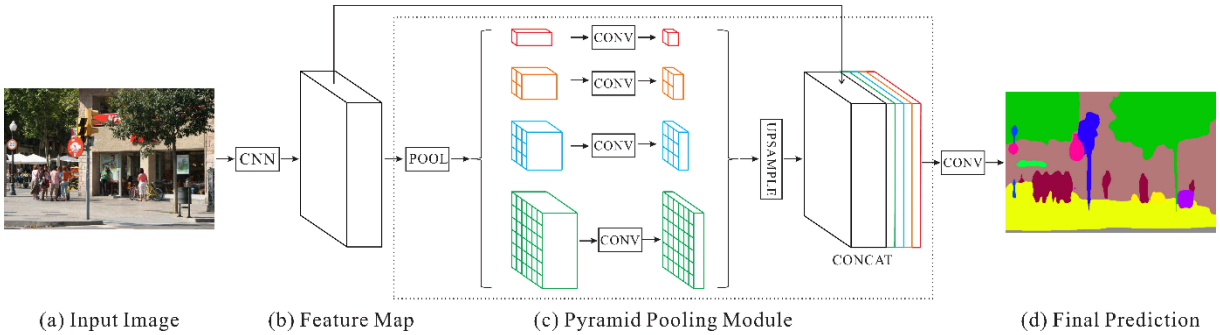


Figure 1.8 PSPNet architecture [62]

The PSPNet architecture's primary elements and characteristics are listed below:

- *Pyramid Pooling Module*: The main goal of PSPNet is to use a pyramid pooling module to capture context at various scales. This module combines local information at different sizes with global contextual data. It aids the network in comprehending the surrounds and context of items.
- *Encoder Network*: The PSPNet architecture [62] typically begins with an encoder network, which applies a number of convolutional layers and pooling operations to the input image to process it. This network records features at various abstraction levels.
- *Pyramid Pooling Module*: The pyramid pooling module is implemented after the encoder. This module conducts average pooling within each of the various-sized zones it has created from the feature map. A rich multi-scale contextual representation is created by concatenating the outcomes of pooling at several dimensions.
- *Decoder Network*: The decoder network improves the features and creates the final segmentation map after the pyramid pooling module. To regain spatial resolution, the decoder frequently uses a number of convolutional and upsampling layers.
- *Skip Connections*: Similar to U-Net and FCN architectures, PSPNet [63] also makes use of skip connections. The network can now capture fine details and context thanks to these links, which combine information from various layers of the encoder with comparable levels in the decoder.

- *Final Output Layer*: The pixel-wise segmentation map is created by the PSPNet architecture's final layer. With the softmax activation function, this layer is typically a 1x1 convolutional layer that generates class probabilities for each pixel.

The distinguishing characteristic of PSPNet is the Pyramid Pooling Module. PSPNet can precisely segregate objects in situations of various dimensions and complexity by recording multi-scale contextual information. This contextual awareness enhances the segmentation performance of the network, particularly in difficult settings where items may be obscured or small.

PSPNet has proven to perform well in many benchmark datasets for semantic segmentation and has made a significant contribution to the field. It's important to note that the principles underlying PSPNet have also been included into succeeding topologies, aiding in the continued advancement of cutting-edge segmentation techniques.

- ***ENet for semantic segmentation***

The neural network architecture known as ENet [64], or "Efficient Neural Network" as presented in Figure 1.9, is small and effective and was created for real-time semantic segmentation of images. In 2016, it was put forth by academics at the University of Waterloo. ENet is specially made to strike a compromise between accuracy and computational effectiveness, making it appropriate for circumstances where resources are limited, including embedded systems and mobile devices.

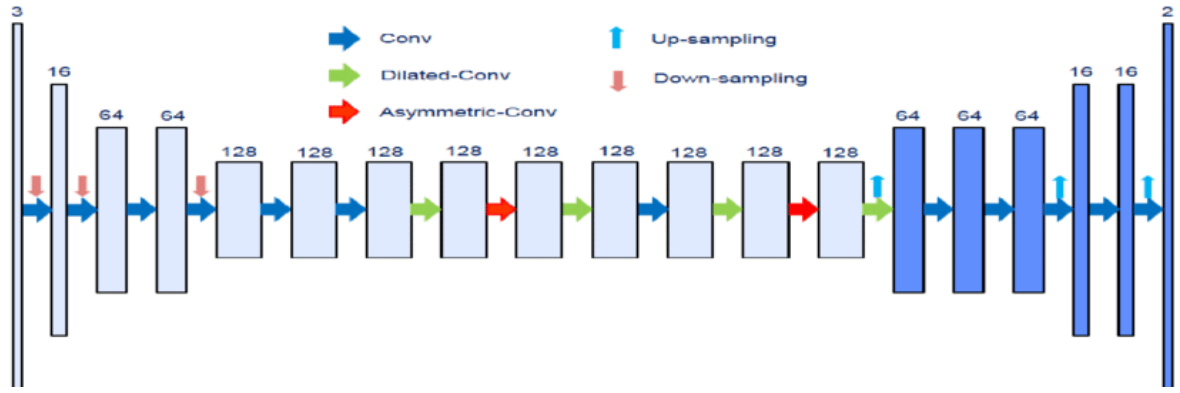


Figure 1.9 ENet architecture [65]

- *Input Layer*: The network accepts input images with any size.
- *Initial Block*: The first block of ENet processes the incoming image and gets it ready for more processing. In order to minimize the spatial dimensions, this block typically comprises of a number of convolutional layers followed by a max-pooling layer.

- *Bottleneck Blocks*: The bottleneck blocks that make up the core of the ENet architecture each comprise a combination of convolutional layers. The following sequence makes up the bottleneck block:
  a- 1x1 Convolution (Pointwise Convolution): By reducing the number of input channels, this convolutional layer aids in the reduction of computational complexity.
  b- 3x3 Depthwise Convolution: This depthwise convolution reduces computation while retaining spatial information by performing spatial convolution individually for each input channel.
  c- 1x1 Convolution (Pointwise Convolution): This convolutional layer adds more channels once more, regaining feature richness. The bottleneck blocks use 1x1 convolutions to decrease and raise the number of channels around the more costly 3x3 depthwise convolution, striking a compromise between computing economy and expressive capability.
  d- Skip Connections and aggregated upsampling: To connect early layers to subsequent layers directly, ENet uses skip connections. This enhances information flow and aids in the preservation of minute details. The ENet architecture also uses an aggregated upsampling strategy. The network can use multi-scale information by upsampling and combining the feature maps from several levels using element-wise summation.
  e- Decoder: A series of 1x1 convolutional operations are used in the decoder section of the ENet architecture, followed by upsampling. The feature maps' spatial resolution is continually increased by these upsampling layers until it reaches the original image size.
  f- Final Output Layer: In the ENet architecture, the final layer is a 1x1 convolutional layer with softmax activation that generates class probabilities for every pixel in the input image. A segmentation map with pixels per inch is the end product.

Efficiency is given priority in ENet's architecture [65] while segmentation accuracy is kept at a competitive level. For applications needing real-time semantic segmentation on low-resource devices, such as embedded systems, robotics, and mobile platforms, its small architecture makes it a good fit.

- ***SegNet for semantic segmentation***

SegNet [66] that has the presentation in Figure 1.10, is a convolutional neural network (CNN) architecture made for computer vision tasks requiring semantic segmentation. In the 2017 article "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation"

by Badrinarayanan et al., SegNet was suggested [67] as a method for image segmentation. While being computationally less demanding than some other deep learning segmentation designs, the architecture is renowned for its effectiveness and efficiency in semantic segmentation tasks. The major novelty of SegNet architecture, which is an encoder-decoder structure, is how it effectively manages the upsampling process utilizing pooling indices.

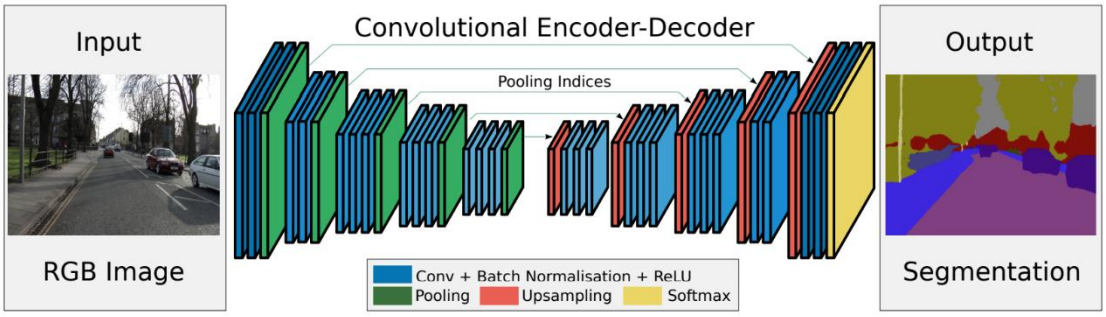


Figure 1.10 SegNet architecture [68]

- **Encoder structure**
  - a- *Input Image*: An input image serves as the SegNet architecture's initial input.
  - b- *Convolutional Layers*: To extract features from the input image, a succession of convolutional layers with batch normalization and rectified linear unit (ReLU) activations are used. These layers record traits that get increasingly abstract.
  - c- *Max Pooling*: To downsample the feature maps, max pooling is used after each pair of convolutional layers. For each pixel position, it's crucial to maintain the indices of the maximum values used in the pooling operation. The decoder will utilize these indices to carry out precise upsampling.
- **Decoder structure**
  - a- *Upsampling using Pooling Indices*: The decoder begins by using the pooling indices that were previously stored from the appropriate encoder layers. These indices show the locations of the maximum values during the pooling process. The max values are then positioned where they originally belonged, allowing the decoder to appropriately upsample the feature maps.
  - b- *Deconvolutional layers*: often referred to as transposed convolutional layers or upsampling layers, are part of the decoder that gradually boost the spatial resolution of the feature maps.

c- *Output Layer*: The semantic segmentation map is produced by the decoder's last layer. Each pixel on this map is given a class label designating the category to which it belongs, and its spatial dimensions are identical to those of the original image.

Due to the architecture's effective utilization of pooling indices, common problems such checkerboard artifacts that might occur with simple upsampling techniques are reduced. SegNet can produce segmentation results that are smoother and more accurate by repeating the precise locations of the maximum values. SegNet was created with the intention of achieving a compromise between computational effectiveness and segmentation accuracy, making it appropriate for real-time applications, particularly in settings when resources are few. Keep in mind that the description offered here is based on the initial SegNet paper released in 2017. Since then, the SegNet design may have undergone further modifications, advancements, or extensions.

- ***LinkNet for semantic segmentation***

For computer vision tasks requiring semantic segmentation, LinkNet [69] is another convolutional neural network architecture. It was first presented in the 2017 publication "LinkNet: Exploiting Encoder Representations for Efficient Semantic Segmentation" by Chaurasia and Culurciello. While preserving computational efficiency and ease of training, LinkNet seeks to achieve accurate semantic segmentation. The encoder representations are effectively used by the encoder-decoder structure of the LinkNet architecture to deliver high-quality segmentation results.

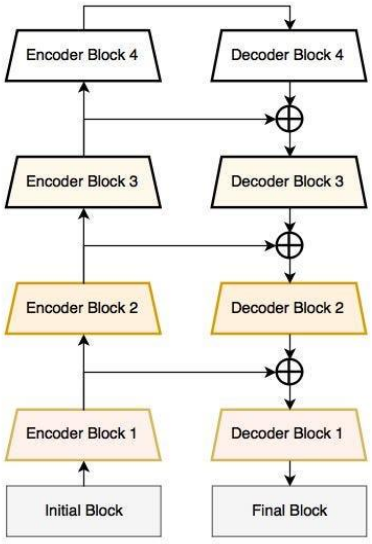


Figure 1.11 LinkNet architecture [70]

The architecture of LinkNet that was represented in Figure 1.11 has the following primary features:

- **Encoder Decoder structure**

The encoder-decoder structure of LinkNet is similar to the previous semantic segmentation designs. An initial encoder is used for feature extraction, and a decoder is used to upsample the data and create the final segmentation map. The encoder gradually reduces the spatial dimensions by extracting features from the input image using a sequence of convolutional layers. After each convolutional layer, batch normalization and ReLU activations are frequently utilized to improve training and performance. On the other hand, the decoder uses the encoded features and upsamples them to create a segmentation mask with the dimensions of the original input image. In order to enhance the segmentation outcomes, the decoder also incorporates data from the skip connections.

The encoder begins by extracting feature representations from the input image using a pretrained backbone network, such as ResNet. Convolutional blocks, which frequently include convolutions, batch normalization, and ReLU activations, are used by the encoder to further enhance the features. Furthermore, to minimize spatial dimensions, downsampling techniques (such max pooling or strided convolutions) are used.

- **Skip Connections**

Skip connections are used by LinkNet between the encoder and decoder layers. The decoder can use characteristics extracted at different levels of abstraction thanks to these links, which aid in the flow of information at varying resolutions.

- **Skip Pooling**

Skip pooling operations are used by LinkNet throughout the encoding process. These skip connections and skip pooling methods enhance information flow while downsampling the feature maps.

- **Bottleneck Design**

Utilizing a bottleneck design to lower the dimensionality of feature maps before to upsampling is the main concept behind LinkNet. This bottleneck design ensures that the decoder can effectively reconstruct the segmentation map while preserving computational efficiency.

- **Batch Normalization and Relu Activation**

To increase training stability and convergence, the design makes use of batch normalization and ReLU activation functions.

- **Efficient Upsampling**

To raise the spatial resolution of the feature maps while preserving computational efficiency, LinkNet employs effective upsampling methods such as transposed convolutional layers.

The LinkNet paper's authors highlight the architecture's simplicity, effectiveness, and efficiency in comparison to other, more sophisticated segmentation systems like U-Net, SegNet, and FCN (Fully Convolutional Network). The design decisions made by LinkNet [71], including skip connections, skip pooling, and the bottleneck architecture, help it to produce competitive segmentation results while being computationally effective.

- ***HRNet deep learning***

High-Resolution Network, or HRNet [72] for short, is a deep neural network architecture created to address the difficulties of handling high-resolution visual data in computer vision tasks. Unlike traditional networks that downsample the input image early in the architecture to reduce computational load, HRNet maintains high-resolution representations throughout the network, enabling it to capture fine-grained details and better address multi-scale information in the image.

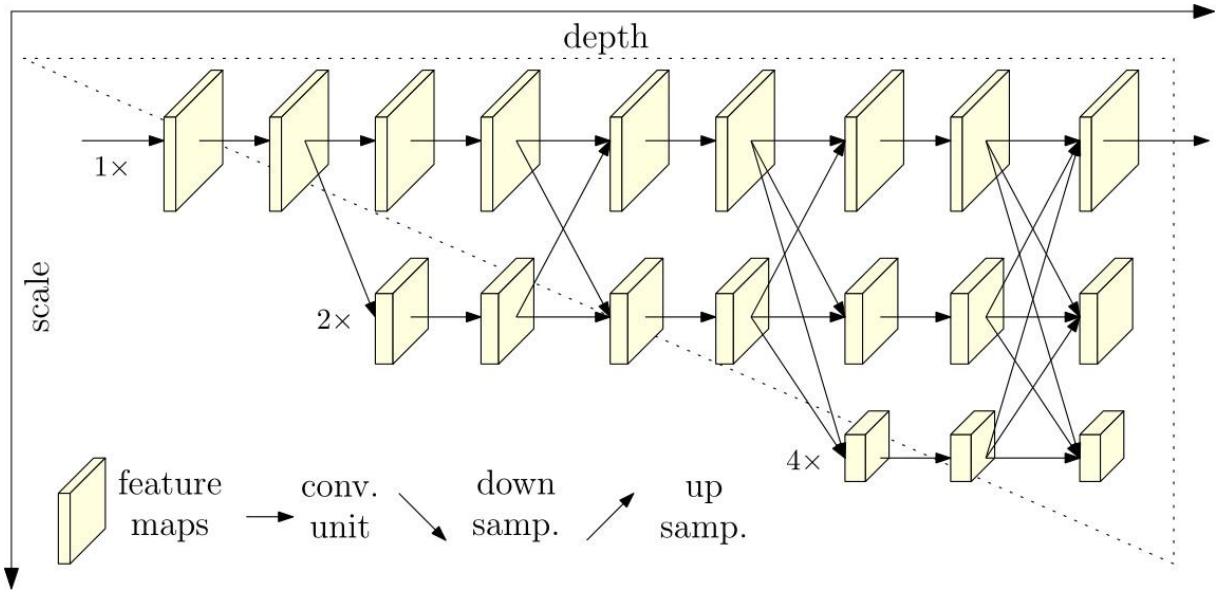


Figure 1.12 HRNet architecture [73]

The HRNet (High-Resolution Network) architecture is made to efficiently capture multi-scale data while maintaining high-resolution representations across the network. This is

accomplished by using several concurrent streams that combine data from these streams and process input images at various resolutions. An overview of the HRNet architecture [73] as you can see in Figure 1.12 is provided below:

a- *Stem Network*: A stem network first processes the input image by extracting the first features. The parallel streams are then given access to these functionalities.
b- *Parallel streams*: are used by HRNet, each of which processes the incoming image at a different level of resolution. These streams are built to properly handle little details and multi-scale context.
c- *Multi-Resolution Fusion Modules*: Multi-resolution fusion modules are used to combine data from various network levels inside each parallel stream. The network can meld high-resolution details with multi-scale information thanks to this fusion.
d- *High-resolution pooling operations*: are used to downscale the feature maps while keeping all of the small features. As opposed to conventional pooling techniques, which cause information loss in high-resolution data, this one does not.
e- *Cross-Resolution Supervision*: HRNet frequently makes use of signals for supervision at several scales. This ensures that the network learns to handle both fine-grained details and contextual information by training it with ground truth labels at varied resolutions.

The architecture of HRNet [74] is a good example of the value of preserving high-resolution data while also retaining multi-scale context, leading to better performance on jobs requiring precise localization and thorough analysis. Given that the subject of deep learning is still developing, it is crucial to consult the most recent research publications for the most up-to-date details on HRNet and its variations.

- ***Attention mechanisms***

Particularly in the areas of computer vision and natural language processing (NLP), attention mechanisms [75] are a key idea in deep learning. They give models the ability to process incoming data with a focus on particular portions of it, allowing the models to assign various levels of relevance to various input pieces.

The basic idea behind attention mechanisms is to mimic human attention. When reading a sentence or viewing an image, humans tend to concentrate more on particular words or areas based on the context, importance, or relevance. Similarly, when neural networks make predictions, attention mechanisms assist them by helping them "pay attention" to pertinent portions of the incoming data [76].

In deep learning, many attention strategies are employed:

- *Self-Attention (Scaled Dot-Product Attention):* The Transformer architecture, which transformed NLP tasks, frequently makes use of self-attention. Each query element is given a weighted sum of values, with the weights being determined by the dot product of the query and the keys. The model may take into account the connections between several items in the input sequence at once thanks to self-attention.
- *Multi-Head Attention*: To capture various facets of the relationships in the input data, numerous sets of projections of the query, key, and value are used. As a result, the model can concentrate on multiple areas of the input at once and gather different kinds of data.
- *Spatial Attention*: In computer vision tasks, spatial attention techniques are employed to concentrate attention on particular areas of an image. For instance, when captioning images, the model can be trained to focus on specific areas of the image and produce descriptions that correspond.
- *Channle Attention*: Convolutional layer-generated feature maps produced by different channels are the subject of computer vision's "channel attention" technique. Based on how well they serve the purpose, it learns to give various channels varying degrees of importance.
- *Global Average Pooling (GAP) and Global Max Pooling (GMP)*: are pooling procedures that determine the maximum or average value for each channel in a feature map. By summarizing the most crucial data from each channel, these operations can serve as a basic kind of attention.
- *Scaled Dot-product Attention*: calculates the similarity between the key and query vectors, which then establishes the weights for the related values. This method is used for various attention mechanisms.

- ***Transformer-based Models***

Natural language processing (NLP) [77] along with other related fields have been transformed by a family of deep learning models known as transformer-based models. They are highly suited for jobs like language translation, text production, sentiment analysis, and more because they are made to handle sequences of data.

The attention mechanism, which enables the model to consider the relative relevance of several words or tokens in a sequence when making predictions, is the main novelty of transformer models. Long-range dependencies in the data can now be captured by the model thanks to this

attention mechanism, which was previously impossible with models like recurrent neural networks (RNNs).

In the publication "Attention Is All You Need" published in 2017, Vaswani et al. first described the transformer design. The encoder and decoder components of the transformer model [78] each have several layers. The feedforward neural network and the multi-head self-attention mechanism are the two primary elements of each layer.

An outline of the architecture is given below:

- ***Encoder structure***

The encoder uses a stack of identical layers to process a sequence of input tokens (such as words or subwords). Two sublayers make up each layer:

a- A system called multi-head self-attention calculates attention scores between every pair of tokens in the input sequence. Various "heads" of attention are used to focus on various relationships within the sequence. When the model makes predictions for each token, it helps the model concentrate on pertinent portions of the input.

b- Feedforward Neural Network: The outputs are routed through a position-wise feedforward neural network after self-attention. For each place in the sequence, this network functions independently. Following layer normalization, both sub-layers have residual connections (skip connections) surrounding them. Deep network training and learning stabilization are both aided by this.

- ***Decoder structure***

Although the decoder and encoder are similar, the decoder contains an additional third sub-layer:

a- Masked Multi-Head Self-Attention: During training, the decoder makes autoregressive predictions one token at a time. The self-attention layer is covered with a mask to shield the model from future tokens. This makes sure that each token may only take care of positions that came before it in the sequence. Similar to those in the encoder, the additional sub-layers include a feedforward neural network and normalization.

- ***Positional encodings***

Positional encodings are included to the input embeddings since, unlike recurrent models, the transformer model does not already know the token order. These encodings give the model details about the token placements inside the sequence, enabling it to comprehend the order.

- **Input and Output embeddings**

Initial high-dimensional vector embeddings of the input tokens are usually scaled by the square root of the embedding dimension. The resulting embeddings are also the foundation for creating predictions in the decoder.

- **Connection to the Decoder**

The initial states for the decoder are taken from the final layer outputs of the encoder. Through this relationship, the decoder can focus on the necessary portions of the input sequence when producing output tokens.

The transformer architecture served as the basis for a variety of models, each of which was refined through time and tailored for particular needs. By extending or altering the transformer architecture, models such as BERT, GPT, T5, as well as XLNet have been created that deliver cutting-edge outcomes on a variety of natural language processing applications. The figure 1.13 below shows the transformers, GPT, BERT architecture [79].

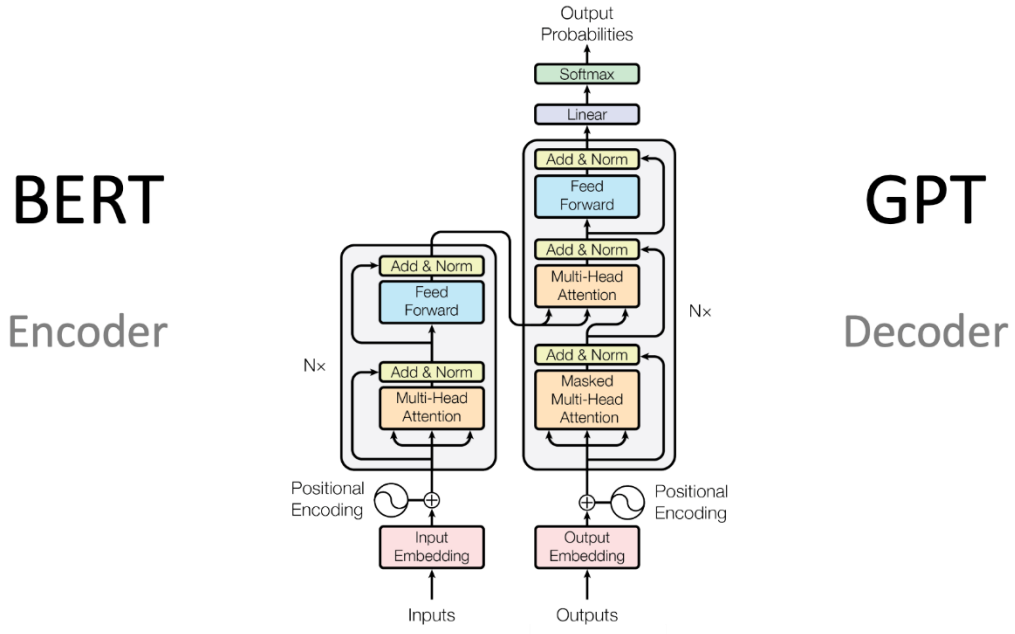


Figure 1.13 Transformers, GPT, BERT architecture [79]

1.4.4 Evaluation metrics

The effectiveness of algorithms and models that accomplish the task of segmenting an image into discrete regions belonging to various items or classes is evaluated using image

segmentation assessment metrics [80]. These metrics offer numerical evaluations of how well the segmentation output adheres to the segmentation map's manually annotated ground truth.

- *Pixel Accuracy (PA):* Pixel accuracy determines the proportion of successfully classified pixels in the segmented image [81]. Although it is a straightforward and understandable metric, it might not be appropriate for classes that are unbalanced or when the boundaries between objects matter.
- *Mean Pixel Accuracy (MPA):* This measure is comparable to pixel accuracy in that it determines the accuracy for each class independently before averaging those accuracies [82].
- *Mean Intersection over Union (mIoU):* IoU, sometimes referred to as the Jaccard Index, determines the amount of overlap between each class's expected and actual areas. The average IoU for all classes is then determined by the mIoU. It's one of the most used measures for measuring the effectiveness of image segmentation.

  IoU is defined as: (Intersection of predicted and ground truth regions) / (Union of predicted and ground truth regions)

  mIoU = (IoU for class 1 + IoU for class 2 + ... + IoU for class N) / N
- *Frequency Weighted IoU (FWIoU):* Similar to mIoU but weighting classes with bigger regions more heavily, frequency weighted IoU (FWIoU) helps equalize the effect of classes with different area sizes.
- *Dice Coefficient (F1-score):* The Dice coefficient, which takes into account both precision and recall, calculates the overlap between the predicted and ground truth areas. Medical image segmentation jobs frequently employ it.

  Dice Coefficient = 2 * (Intersection) / (Total number of pixels in both regions)

- *Boundary F1-Score:* This statistic measures how effectively the model segments object boundaries. For activities where exact border delineation is crucial, it is very helpful.

$$F1 - score = 2\frac{Precision * Recall}{Precision + Recall} \tag{1.15}$$

$$F1 - score = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{1.16}$$

- *Precision and recall:* By treating the predicted and ground truth areas as sets of pixels, these traditional binary classification metrics may be used to segmentation. While recall assesses the proportion of properly identified pixels in the region of the ground truth, precision assesses the proportion of correctly categorized pixels in the anticipated region.

$$F1 - score = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{1.17}$$

$$F1 - score = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{1.18}$$

$$Precision = \frac{TP}{TP + FP} \tag{1.19}$$

- *Global Consistency Error (GCE):* Taking into account pixel-wise correlations and connectivity, GCE determines the discrepancy between expected and actual segmentations.
- *Contours-based Metrics:* These metrics evaluate the smoothness and quality of the object borders, frequently by comparing anticipated contours to actual contours.
- *Boundary Recall and Precision:* These measures assess how well object borders are segmented.

The peculiarities of the segmentation job, the significance of various classes, and the attributes of segmentation performance that are most pertinent to the application should all be taken into consideration when selecting an assessment measure [83]. Using a variety of measures to have a thorough knowledge of the model's performance is also a smart idea.

1.5 Object detection

Identifying and detecting objects inside an image or a video stream is a computer vision problem known as object detection [84]. Identifying present items and pinpointing their exact placements within the visual input are the two objectives of object detection. The importance of this problem may be seen in a number of applications, including autonomous driving, surveillance, robotics, medical imaging, and others as well.

Typically, object detection entails the following crucial steps:

- *Image preprocessing*: In order to improve the quality of the input image, equalize the colors, and maybe shrink it to a standard size for processing, it is typically preprocessed.
- *Feature extraction*: Convolutional Neural Networks (CNNs) are frequently employed for feature extraction from input images. Hierarchical representations that capture pertinent patterns and features can be learned by these networks.
- *Region Proposal*: In this stage, potential areas of interest that could contain objects are suggested. These regions are produced using a variety of methods, including anchor-based methods, region proposal networks (RPNs), and selective search [85].
- *Classification*: The object detection system classes each suggested region according to the sort of object it believes to be present there. Frequently, softmax-based categorization is used for this.
- *Bounding Box Regression*: The model not only categorizes the objects but also improves the bounding box coordinates surrounding each object that is observed. The bounding box will be perfectly aligned with the object's real shape and location as a result.
- Non-Maximum Suppression is a post-processing technique used to eliminate redundant detections and maintain only the most certain and non-overlapping ones. This is done because many zones may overlap and detect the same object.
- *Output*: The labels of the discovered objects and their related bounding box coordinates are included in the final output of an object detection system.

Figure 1.14 bellow shows the main steps for object detection

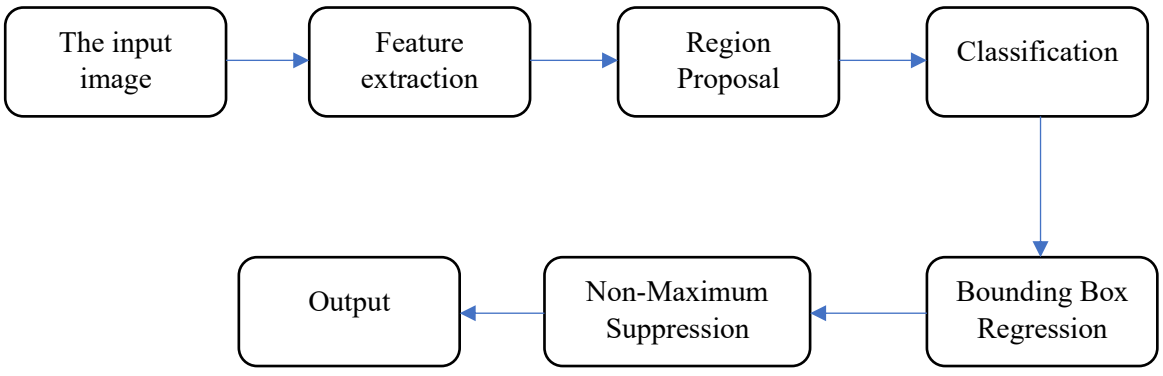


Figure 1.14 Object detection flowchart

For object detection, there are a number of well-liked methods and structures, including:

- *Faster R-CNN*: A well-liked object detection approach in computer vision is faster R-CNN (Region-CNN) [86]. Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun first

discussed it in their 2015 paper titled "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." Through the incorporation of a region proposal network (RPN) into the detection pipeline, Faster R-CNN aims to increase the object detection pipeline's speed and accuracy.

- *YOLO*: Another well-liked object detection approach in computer vision is called "You Only Look Once," or YOLO [87]. Ross Girshick, Ali Farhadi, Santosh Divvala, and Joseph Redmon first discussed it in their 2016 work titled "You Only Look Once: Unified, Real-Time Object Detection." In contrast to Faster R-CNN, YOLO adopts a different strategy, aiming for real-time object identification by directly predicting bounding boxes and class probabilities in a single neural network pass.
- *SSD*: or Single Shot MultiBox Detector. Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg introduced SSD [88], also known as "Single Shot MultiBox Detector," as another well-liked object detection algorithm in computer vision in their 2016 paper titled "SSD: Single Shot MultiBox Detector." Similar to YOLO, SSD is made with an emphasis on effectiveness and accuracy for real-time object recognition.
- *RetinaNet:* Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár presented RetinaNet, a cutting-edge object recognition system ([87], [88]), in their 2017 publication titled "Focal Loss for Dense Object Detection." RetinaNet was created to address the difficulty of detecting objects at various scales while avoiding the "vanishing gradient" issue that sometimes arises when deep neural networks are trained for object detection.

The effectiveness of algorithms and models that locate and identify objects and items inside images is evaluated using metrics for object detection. These measures make it possible to measure how closely the projected bounding boxes match the actual annotations on the ground. Here are some often-used metrics for object detection evaluation:

- *Intersection over Union (IoU):* IoU calculates the amount of overlap between the predicted and actual bounding boxes. It is determined by dividing the overlapped area by the united area of the two bounding boxes [88]. When determining if a forecast is true positive, false positive, or false negative, IoU is frequently utilized.

$$\text{IoU} = (\text{Area of Overlap}) / (\text{Area of Union}) \tag{1.20}$$

- *Average Precision (AP):* The statistic known as Average Precision (AP) is frequently employed in object detection. By computing the precision-recall curve's area under the curve, it takes trade-offs into account. Recall is the proportion of genuine positives to all ground truth positives, whereas precision measures [89] the proportion of true positives to all anticipated positives. By altering the confidence threshold for object detection and computing the precision and recall values at each threshold, the precision-recall curve is produced. The AP score is determined by the area under this curve.
- *Mean Average Precision (mAP):* The mean of AP values computed across several classes is known as mAP. It offers a comprehensive evaluation of the model's effectiveness across multiple object types.
- *Precision-Recall Curve:* As the confidence threshold for object detection changes, the precision-recall curve illustrates the trade-off between precision and recall. It aids in comprehending how the model performs at various operating points.
- *F1-Score:* The harmonic mean of recall and accuracy is the F1-score. It is helpful when the class distribution is unequal or when both recall and accuracy are equally crucial.
- *The Fowlkes-Mallows Index (FMI):* counts the true positives, false positives, and false negatives to determine how close the anticipated and actual bounding boxes are.
- *Log-Average Miss Rate (LAMR):* The detection performance for various IoU thresholds is summarized using the LAMR. It offers information on how well the model functions at various localization accuracy levels.
- *CorLoc (Correct Localization):* With the help of the ground truth annotation, CorLoc calculates the percentage of images for which at least one of the identified instances has an IoU that is higher than a set threshold. It assesses how well the model can locate items.
- *Precision at Different IoU Thresholds:* Precision may be computed at a variety of IoU thresholds, including 0.5 (which is frequently used in the Pascal VOC challenge), 0.7, etc. This aids in analyzing the model's performance when localization strictness is varied.
- *Recall at Different IoU Thresholds:* Recall may also be measured at various IoU thresholds in order to assess how effectively the model catches items at various levels of overlap.
- *AP@[IoU Threshold]:* In order to determine how well the model works at a specified amount of overlap, this measure estimates the average precision particularly at a predetermined IoU threshold.

The specifics of the job, such as the number of classes, the existence of tiny or big objects, and the optimal trade-off between precision and recall, should all be taken into account when assessing object detection models. A thorough insight of the model's performance may be obtained by combining these indicators.

1.6 Image classification

The fundamental goal of computer vision known as image classification [90] entails classifying an incoming image into one of many predetermined classes or categories. In order to effectively classify images into the appropriate categories, image classification aims to train a machine learning model to recognize patterns, features, and properties within images. Applications for this task are numerous and include object identification, information filtering, medical diagnosis, and more.

Here is a summary of the primary steps in image classification:

- *Dataset preparation*: A dataset with annotated images is used to train an image classification algorithm. Each image has a unique class label attached to it that describes its content.
- *Data Preprocessing*: Before training, the dataset's images are frequently preprocessed. To increase the variety of the training data and enhance the generalization of the model, preprocessing may entail downsizing images to a constant resolution, standardizing pixel values, and applying data augmentation techniques.
- *Feature extraction:* Convolutional Neural Networks (CNNs), which can automatically identify hierarchical characteristics in images [91], are frequently employed for image categorization. Convolutional, pooling, and fully connected layers, which extract and process features at various levels of abstraction, are all parts of CNNs.
- *Model Architecture*: The positioning of the layers, the total number of neurons, and the connection patterns are all determined by the architecture of the CNN. Architectures that are frequently used include AlexNet, VGG, ResNet, Inception, and many more. Multiple convolutional layers are frequently used in these architectures, which are subsequently followed by fully connected layers for classification ([92], [93]).
- *Training*: During training, the model learns to reduce the discrepancy between its predictions and the actual class labels in the training dataset by adjusting its internal parameters (weights and biases). Both backpropagation and forward propagation are used in this procedure to update parameters using gradient descent.

- *Loss Function*: The discrepancy between anticipated class probabilities and actual class labels is measured by a loss function. Categorical cross-entropy and softmax loss are typical loss functions for classification applications.
- *Optimization*: To iteratively change the model's parameters in the direction that minimizes the loss function, optimization methods like stochastic gradient descent (SGD) or its variations are utilized.
- *Validation and Testing*: Following training, the model's effectiveness is assessed using a different validation dataset in order to adjust the hyperparameters and avoid overfitting. A second test dataset is used to evaluate the model's performance in the actual world after it has been finalized.
- *Inference*: The trained model is applied to infer the class label [93] of fresh, unlabeled images. After the model processes the image, it produces a probability distribution over the various classes.

Image classification flowchart can be represented in Figure 1.15 as follows:

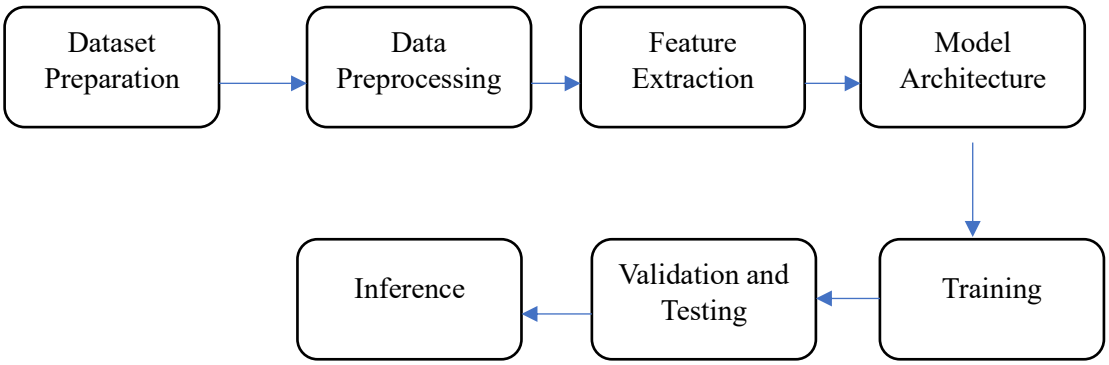


Figure 1.15 Image Classification steps flowchart

With the emergence of deep learning and CNNs, image classification has made great strides. It is simpler to get acceptable performance in a variety of applications when using pretrained models, which are models trained on huge datasets like ImageNet.

1.7 Text recognition

Optical character recognition (OCR), commonly referred to as text recognition in image processing, is a technology that converts extracted text from images or scanned documents into editable and searchable text. Numerous processes, including document digitization, robotic data input, language translation, and others, heavily rely on this method. Moreover, Tesseract, PyTorch, TensorFlow, and OpenCV are well-known OCR libraries and frameworks that offer tools and APIs to execute these processes and accomplish precise text recognition in image

processing applications. An outline of the procedures involved in word recognition using image processing is provided below:

Step 1: Image acquisition: Obtaining an image with the text to be recognized by using image acquisition. This image may have been derived from scanned papers, photographs, or screenshots, among other sources.

Step 2: Preprocessing: To increase the precision of word recognition we need to improve the quality of the captured image by using image scaling and resizing preprocessing methods, binary conversion of an image, as well as image averaging and noise reduction.

Step 3: The areas of the image with text should be identified. To separate the text from the rest of the image, we must do these important steps below:

- Edge detection which is one of the methods for localizing text.
- Detecting measuring contours.
- Linked component analysis.

Step 4: (Optional). If necessary, separate the discovered text sections into individual characters or phrases using character segmentation. If character-level recognition is desired or the text is densely packed, this stage can be necessary.

Step 5: Extraction of Features: Take features from the segments or localized text areas. Shape, size, stroke width, and other pertinent traits that aid in character differentiation could be included in these features.

Step 6: Character Recognition: To identify the characters in the segmented regions, we use machine learning models or deep learning architectures. Due to their efficiency in image recognition tasks, convolutional neural networks (CNNs) are frequently utilized for this purpose.

Step 7: Postprocessing: Refine the detected characters and fix any mistakes that may have happened during the recognition process. To increase the accuracy of the detected text, this stage could use language modeling, spell-checking, or utilizing the context.

1.8 Conclusion

In conclusion, image segmentation and object identification are crucial tasks within computer vision, each one of them addresses a different set of issues and serves a different function. The fundamental principle of computer vision, which enables machines to perceive and interpret

visual input, is shared by image segmentation and object detection. Their numerous industrial uses progress technology and improve human interactions with both the real and digital worlds. On the other hand, Optical Character Recognition (OCR) technology, which makes text recognition possible, has a variety of uses in different fields. In this chapter, we gave a very precise state of the art of image segmentation, object detection, image classification as well as text recognition in the image processing field by affording all the methods and techniques that are used in every area.

Chapter 2:

Mathematical tools and algorithms for image segmentation, object detection and text recognition

2.1 Introduction

Important tasks in computer vision include image segmentation and object detection, which frequently serve as the first steps in a variety of applications such as autonomous vehicles, medical imaging, object tracking, and more. For these tasks, a wide variety of algorithms are used, each with unique strengths and useful use cases. In this chapter we'll describe a few important object identification and image segmentation algorithms.as well as discussing the main methods that have been used in the contributions of this thesis. For this, we will start by a general introduction for each method, subsequently, mathematical tools and algorithms will be given in detail.

2.2 Algorithms used in the first proposed method: "A brain tumor segmentation and detection technique based on Birch and Marker watershed"

2.2.1 Median Filter:

Median filter [94] is defined as a popular and efficient method for reducing noise and raising the quality of an image. Salt-and-pepper noise, isolated white and black pixels that can happen for a variety of causes during image capture or transmission, is particularly helpful for eliminating with this technique. Moreover, the median filter's main objective is to reduce noise while keeping the image's borders and features intact. The median filter [95] replaces the pixel value with the median value of the pixel's neighborhood as opposed to linear filters (such as Gaussian or mean filters), which replace the pixel value with an average of nearby pixels. Because outliers have a tendency to have a major influence on the performance of mean-based filters, the median filter is particularly resistant to them. Furthermore, the median filter moves a window [96], often referred to as a kernel or mask, over the image. The pixel values inside the

window are gathered at each place, and the median value is then computed from these pixel values. The middle value obtained after sorting all of the pixel values in the window is known as the median value. The original pixel value in the window's center has been replaced by this median value. This median value has replaced the original pixel value in the window's center of each pixel in the image, this procedure is repeated.

- ***Step 1***: Select a kernel or window that is an unusual size. Sizes like 3x3, 5x5, 7x7, etc. are typical. There is a distinct central pixel thanks to the strange size.
- ***Step 2:*** Put the window at the image's very first pixel.
- ***Step 3***: Gather and sort the window's pixel data in ascending order.
- ***Step 4***: Set the median value from the sorted list of pixel values as the window's central pixel value.
- ***Step 5***: To process the complete image, move the window to the subsequent pixel position and repeat steps 3 and 4 there.

The median filter flowchart can be presented in Figure 2.1 as follows:

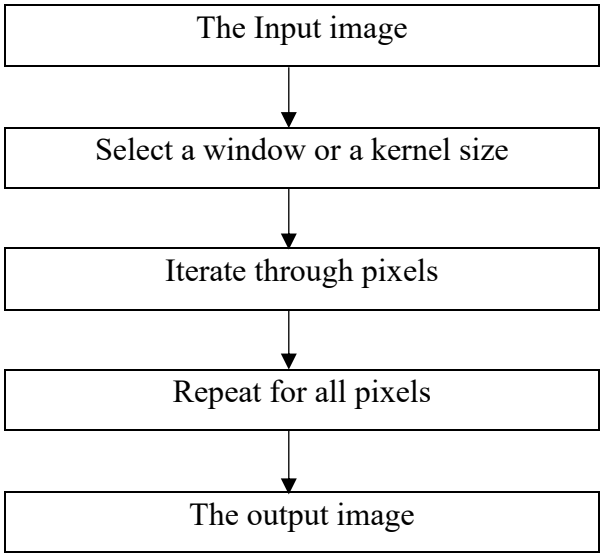


Figure 2.1 The median filter flowchart

Median's filter advantages [97] are several and one of them is that it's considered as an efficient noise reduction system, especially for salt-and-pepper noises, moreover, it's better at preserving edges and minute features than mean-based filters. On the other hand, Median filter is less affected by extreme values, making it robust against outliers. However, given that it doesn't consider the image's actual composition, the median filter might result in blurring along sharp edges. Additionally, it's slower than mean-based filters, particularly for bigger window widths.

A window, often of odd width, is stepped along a signal by the median filter one sample at a time. The sample values within are ordered accordingly to their magnitude at each point of the window, and the middle element of this ranking is known as the output. Typically, it is considered that the window has a width of 2N + 1, where N is any positive integer. Assume that the input sequence's kth sample serves as the window's center and that the vector specifies the window's 2N + 1 points in order of time.

$$(x_{k-N}, x_{k-N+1}, \ldots, x_k, \ldots, x_{k+N}) \tag{2.1}$$

It's required to find y_k where:

$$y_k = \text{median}\,(x_{k-N}, x_{k-N+1}, \ldots, x_k, \ldots, x_{k+N}) \tag{2.2}$$

- **Median's Filter algorithm:**

The following Median filter algorithm for a n x m window:

Algorithm 2.1: Median filter algorithm

$for\ x_0 = 1\ to\ X\ do$
  $for\ y_0 = 1\ to\ Y\ do$
    $for\ x_i = x_0 - \frac{n-1}{2}\ to\ x_0 + \frac{n-1}{2}\ do$
      $for\ y_i = y_0 - \frac{m-1}{2}\ to\ y_0 + \frac{m-1}{2}\ do$
        $load\ input\ [x_i, y_i]\ into\ array\ A;$
      end for
    $output\ A[\frac{N-1}{2}]$
  $end\ for$
$end\ for$

2.2.2 Fuzzy c-means:

A clustering method called fuzzy C-means (FCM) [98] is used to divide a set of data points into clusters. It extends the traditional K-means technique and allows data points to be assigned to clusters with variable levels of membership, or "fuzziness." When a data point's placement in a particular cluster is questionable, FCM is very helpful [99].

Each data point in K-means is solely assigned to one cluster center depending on that center's proximity. The degree to which each data point is a member of each cluster is indicated by the membership value that each data point in FCM has assigned to each cluster center. The weights used to represent these membership values for each data collection add up to 1, and Membership and cluster centers are adjusted based on the following formula after each iteration:

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{c} (\frac{d_{ij}}{d_{ik}})^{\frac{2}{m-1}}} \tag{2.3}$$

$$v_j = \frac{(\sum_{i=1}^{n} (\mu_{ij})^m)}{(\sum_{i=1}^{n} (\mu_{ij})^m)}, \forall j = 1, 2, \dots c \tag{2.4}$$

Where:

- 'n' represents the number of data points
- 'm' represents the fuzziness index $m \in [1, \infty]$
- 'μ_{ij}' is the membership of ith data to jth cluster center
- 'v_j' is the jth cluster center
- 'c' is the number of cluster center
- 'd_{ij}' is the Euclidean distance between ith data and jth cluster center

The fuzzy c-means algorithm's primary goal is to minimize:

$$J(U, V) = \sum_{i=1}^{n} \sum_{j=1}^{c} (\mu_{ij})^m ||x_i - v_j||^2 \tag{2.5}$$

Where:

'$||x_i - v_j||$' represents the Euclidean distance between ith data and jth cluster center.

The FCM algorithm works following the mean steps bellow [100]:

- Initialization; Initialize the cluster centers and membership values at random after choosing the number of clusters (K).
- Membership Values Update; based on the separations between each data point and each cluster center, determine new membership values for every point of data. A data point's membership value for a given cluster increases with proximity to the cluster center.

- Re compute the cluster centers using the new membership values to update them. Each cluster center is determined by averaging the data points using membership values as the weights.
- Replicate Steps 2 and 3 until the convergence requirement is satisfied, iterating between changing membership values and cluster centers. Monitoring how much the membership values and cluster centers change between iterations is typically how convergence is assessed.
- As a result, we will get a collection of cluster centers and membership values that represent the fuzzy membership of every point of data to each cluster after the algorithm converges. Figure 2.2 shows the steps of the Fuzzy c-means algorithm.

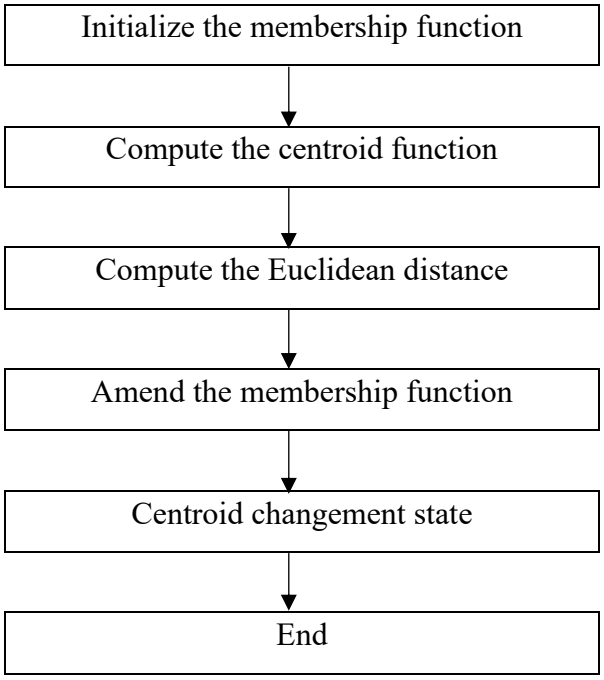


Figure 2.2 Fuzzy c-means flowchart

In pattern recognition, image processing, and data analysis, FCM has a variety of uses. When there is intrinsic ambiguity in the data or when a data point may partially or completely belong to more than one cluster, it is especially advantageous when it may provide some fuzziness to the clustering process.

FCM does, however, have significant limitations, including sensitivity to the initial cluster center placements, the requirement to predetermine the number of clusters (K), and somewhat slower convergence than the sharp K-means method. In addition, non-convex clusters and outliers may be difficult for FCM to handle. Fuzzy C-means, in general, is a potent clustering

method when it comes to allowing for variation in data point assignments, but it necessitates careful parameter adjustment and a grasp of the underlying data.

Algorithm 2.2: Fuzzy c-means algorithm

Initialize the fuzzy cluster number, F and the cluster centers, $c = \{c_1, \ldots, c_j, \ldots, c_p\}$.
Set iteration time q = 0;
While $\|W^{(q)} - W^{(q-1)}\| \geq \varepsilon$ **do**
  for j=1 to F^{th} cluster **do**
    for i=1 to N^{th} image pixel **do**
      calculate μ_{ij}, i.e. the membership of pixel x_i to the jth cluster;
      if $\| x_i - c_j \| = 0$ **then**
        $\mu_{ij} = 1$;
        Reset other membership of pixel i to 0;
      end if
    end for
    update c_j;
  end for
  calculate objective function $W^{(q)}$;
  q = q+1;
end while

2.2.3 Multi Otsu'

A variation on the traditional Otsu's thresholding technique [101] used for image segmentation is the multi-Otsu's approach, sometimes referred to as Otsu's multilevel thresholding. Using pixel intensity values, Otsu's approach automatically chooses an ideal threshold for binarizing an image, thereby dividing it into two classes (often foreground and background).

With Multi-Otsu's approach, the principle is expanded upon in order to divide an image into numerous classes (more than two) depending on intensity levels. When you wish to segment an image into finer details or when you have an image with more than two unique sections, this is quite helpful.

Similar to the original Otsu's approach, the fundamental principle behind Multi-Otsu's method [102] is to repeatedly discover several thresholds that minimize the intra-class variation. How compact and distinct the pixel values are within each class is shown by the variance within that class. The method finds appropriate thresholds to segment the image into numerous classes by maximizing the inter-class variance (difference between classes) and reducing the intra-class variance.

$$\sigma_{\omega}^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t) \tag{2.6}$$

Where ω_0 and ω_1 represents the likelihoods of the two classes separated by a threshold threshold t, and σ_0^2 as well as σ_1^2 represent the variances of these two classes.

The class likelihood $\omega_{\{0,1\}}(t)$ is calculated from the histogram's L bins [103]:

$$\omega_0(t) = \sum_{i=0}^{t-1} p(i) \tag{2.7}$$

$$\omega_1(t) = \sum_{i=t}^{L-1} p(i) \tag{2.8}$$

Minimizing intra-class variance for two classes is the same as maximizing inter-class variance:

$$\omega_1(t) = \sum_{i=t}^{L-1} p(i) \tag{2.9}$$

$$\begin{aligned} \sigma_b^2(t) &= \sigma^2 - \sigma_{\omega}^2(t) \\ &= \omega_0(t)(\mu_0 - \mu_T)^2 + \omega_1(t)(\mu_1 - \mu_T)^2 \\ &= \omega_0(t)\omega_1(t)[\mu_0(t) - \mu_1(t)]^2 \end{aligned} \tag{2.10}$$

which is a class likelihood that is stated ω and class means μ, where the class means μ_0, μ_1 and μ_T are expressed bellow:

$$\mu_0(t) = \frac{\sum_{i=0}^{t-1} ip(i)}{\omega_0(t)} \tag{2.11}$$

$$\mu_1(t) = \frac{\sum_{i=t}^{L-1} ip(i)}{\omega_1(t)} \tag{2.12}$$

$$\mu_T = \sum_{i=0}^{L-1} ip(i) \tag{2.13}$$

Following are the steps in multi-Otsu's approach as shown in Figure 2.3:

- Determine the image's histogram, which displays the occurrence rates for various pixel intensities.
- Utilize the histogram to determine the probability for each intensity level.
- Set the threshold values to the required number of classes at the beginning.
- Go through the threshold values repeatedly:
  a. Based on the current threshold, determine the intra-class variance for each class.
  b. Determine the variance between neighboring classes.
  c. To obtain a composite variance value for the segmentation, add the intra-class and inter-class variances.
- Pick the cutoff points that reduce the composite variance.

Segment images by categorizing them into classes according to threshold values.

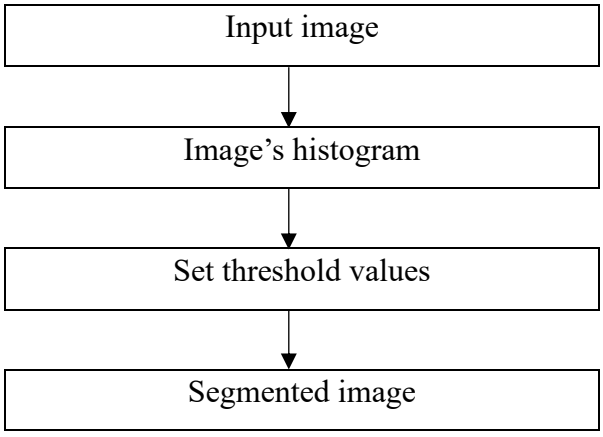


Figure 2.3 Multi Otsu's flowchart

For multi-level thresholding, when the image is divided into several discrete sections, repeat these procedures for the appropriate number of classes. When we wish to automatically segment an image's various items or areas with varied levels of intensity, multi-Otsu's approach comes in handy. Applications like computer vision, medical imaging, and image analysis frequently employ it. Although Multi-Otsu's technique [104] offers automatic thresholding, it's crucial to remember that the outcomes might not always be ideal for complicated images. In circumstances when there is substantial overlap between the intensity levels of several areas, manual adjustment or post-processing might still be necessary to perfect the segmentation.

- **Multi Otsu's algorithm**

Algorithm 2.3: Multi Otsu's algorithm
Compute grayscale histogram H(i) Compute cumulative distribution function C(i) Compute mean grayscale intensity value μ **for** T = 0 to 255 do compute P_0(T) and P_1(T) compute m_0(T) and m_1(T) compute between-class variance var(T) **end** find the optimal threshold value T_{opt} apply threshold T_{opt} to the grayscale image I to obtain binary image

2.2.4 Evaluation criteria

a- Dice coefficient:

The Dice coefficient is a statistical metric frequently used to determine how similar or overlapped two sets of binary data are. The Dice coefficient is frequently used in image processing to assess how well image segmentation methods work. It provides a quantitative evaluation of how well the output of the algorithm matches the actual objects or regions of interest in the image. It does this by measuring the similarity between the anticipated segmentation result and the ground truth segmentation. It's frequently used for applications like comparing the similarity of two strings in natural language processing, image segmentation, and medical image analysis.

Dice coefficient formula can be written as follow:

$$Dice\ coefficient = \frac{2 * intersection}{union + intersection} = \frac{2TP}{2TP + FN + FP} \quad (2.14)$$

Or: $$Dice\ coefficient = 2\frac{precision * recall}{precision + recall} \quad (2.15)$$

The Dice coefficient is in the range of 0 to 1, where:

- 0 shows no resemblance or overlap between the sets.
- 1 shows exact resemblance or overlap between the sets.

The Dice coefficient, for instance, is used to gauge how closely the predicted segmentation mask resembles the original segmentation mask in the context of image segmentation. It aids in assessing how well the expected and actual regions match up. The Dice coefficient can compare two strings in natural language processing, especially for goals like text similarity, paraphrased detection, or named entity identification. However, Dice coefficient has limits, particularly when working with unbalanced data or situations where a high degree of overlap is not the main metric of interest. Other measures like the Jaccard Index or F1-score may be more suited in such circumstances.

b- Jaccard index:

Another widely used measure is the Jaccard Index, sometimes referred to as the Jaccard similarity coefficient or intersection over union (IoU) in a number of disciplines, such as image processing and natural language processing. The Jaccard Index, which measures similarity between two sets like the Dice coefficient does, is frequently used to gauge how well segmentation algorithms function or to compare two sets of data.

Jaccard index is computed using the following formula:

$$Jaccard\ index = \frac{|\ intersection\ of\ sets\ |}{|\ union\ of\ sets\ |} \tag{2.16}$$

$$Jaccard\ index = \frac{|\ A \cap B\ |}{|\ A \cup B\ |} \tag{2.17}$$

With A and B are two random sets.

Where:

$|\ A \cap B\ |$ is the size (cardinality)of the intersection of A and B.

$|\ A \cup B\ |$ is the size (cardinality) of the union of A and B.

You would have two sets in the case of segmenting an image, one having the anticipated segmentation mask's pixels and the other including the ground truth segmentation mask's pixels. When comparing the total number of pixels that appear in both masks (union) to the percentage of pixels that are properly identified in both masks (intersection), the Jaccard Index is calculated. Similar to this, when comparing two sets of words from two texts using natural language processing, the Jaccard Index will calculate the percentage of shared words (intersection) in comparison to the total number of unique words in both sets (union).

The Jaccard Index value that results varies from 0 to 1:

- 0 shows no resemblance or overlap between the sets.
- 1 shows exact resemblance or overlap between the sets.

Higher Jaccard Index values demonstrate more resemblance or overlap among the sets under comparison.

c- Sensitivity:

Sensitivity is a measure that is frequently used in statistics, machine learning, and medical diagnostics to assess how well a binary classification or detection system is doing. It is sometimes referred to by the names true positive rate, recall, or hit rate. It gauges the percentage of positive occurrences that the system properly recognizes. In other words, the capacity of a system to accurately detect the true positive instances among all the positive cases in the dataset is represented by sensitivity. It measures the system's capacity to prevent missing instances that ought to have been labeled positively.

The sensitivity calculation formula is as follows:

$$Sensitivity = \frac{TP}{TP + FN} \tag{2.18}$$

Where:

TP: The situations where the algorithm accurately recognizes positive instances are known as True Positives (TP).

FN: When the system fails to recognize positive occurrences that are truly there, False Negatives (FN) occur.

The range of sensitivity values is from 0 to 1:

- When the system fails to detect any positive occurrences, the sensitivity value is 0 (high rate of false negatives).
- A sensitivity rating of 1 means that all positive cases are appropriately identified by the system (no false negatives).

Sensitivity is crucial in circumstances when omitting positive evidence has major repercussions, for as in the field of medical diagnostics, where failure to detect a condition might cause treatment to be delayed. To gain a more complete knowledge of the system's performance, you might wish to take both specificity and sensitivity into account in some circumstances.

d- Specificity:

Another crucial parameter used in statistics, machine learning, and medical diagnostics to assess the effectiveness of a binary classification or detection system is specificity, often known as the true negative rate. It gauges the percentage of incidents that the system accurately classifies as negative. Moreover, the system's ability to accurately avoid categorizing negative situations as positive is measured by its specificity. In other words, it shows how well the system prevents false alerts for false positives. Therefore, specificity formula is defined as bellow:

Where:

- TN: When the system successfully detects negative occurrences, they are known as True Negatives (TN).
- FP: When the system mistakenly classifies negative instances as positive ones, it produces false positives (FP).

Additionally, specificity values vary from 0 to 1:

$$Specificity = \frac{TN}{TN + FP} \tag{2.19}$$

- The system mistakenly labels all negative cases as positive when the specificity value is 0, which implies a high proportion of false positives.
- The system successfully classifies all negative cases as negative (no false positives), according to a specificity value of 1.

In cases like airport security scans or quality control procedures, where false alarms for harmful events might have serious repercussions, specificity is especially crucial. Similar to sensitivity, specificity is a crucial parameter for evaluating a classification or detection system's overall performance. When contemplating the trade-off between accurately recognizing positives and negatives and preventing false alarms and false negatives, you may wish to strike a balance between sensitivity and specificity in some circumstances.

2.3 Algorithms used in the second proposed method: “A hybrid skin lesions segmentation approach based on image processing methods

2.3.1 Birch segmentation

Clustering algorithms are employed in data mining and machine learning, and one of these algorithms is called BIRCH (Balanced Iterative Reducing and Clustering utilizing Hierarchies) [105]. The technique may be used with many other sorts of data, including image data, even

though it is not specifically an image processing algorithm. Furthermore, large datasets may be efficiently clustered using BIRCH by building a hierarchical structure of clusters. To do this, it combines partitioning and hierarchical techniques. The approach is especially helpful when dealing with huge datasets because conventional clustering algorithms may be computationally or memory-intensive. When using the BIRCH method for image clustering, related images are grouped together by first encoding the image data as feature vectors. Moreover, the Clustering Feature Tree (CF Tree), which is the basis of BIRCH's operation [106], is initially built. In order to enable effective clustering, this tree aids in summarizing the dataset. The method creates this tree slowly before iteratively honing the clusters. Furthermore, using the following equations, we first define the centroid, radius, and diameter of a cluster in BIRCH:

$if\ \{x_i\}\ where\ i\ =\ 1,2,\ldots,N\ are\ the\ data\ points.$

$$The\ centroid \quad x_0\ =\ \frac{\sum_{i=1}^{N} x_i}{N} \tag{2.20}$$

$$the\ radius \qquad R\ =\ \sqrt{\frac{\sum_{i}^{N}(x_i - x_0)^2}{N}} \tag{2.21}$$

$$The\ diameter\ D\ =\ \sqrt{\frac{\sum_{i}^{N}\ \sum_{j}^{N}\ (x_i - x_j)^2}{N(N-1)}} \tag{2.22}$$

The tightness of the cluster around the centroid is shown by both R, the average distance between member points and the centroid, and D, the average pairwise distance inside a cluster.

There are then 5 more distances that may be used to compare the proximity of two clusters.

$let's\ assume\ that\ x_{0_1}\ and\ x_{0_2}\ are\ the\ centroid\ of\ two\ clusters$

The centroid Euclidean distance D_0 is:

$$D_0\ =\ \sqrt{(x_{0_1} - x_{0_2})^2} \tag{2.23}$$

$$D_1\ =\ |x_{0_1} - x_{0_2}| \quad\ =\ \sum_{t=1}^{d} |x_{0_1}{}^{(t)} - x_{0_2}{}^{(t)}| \tag{2.24}$$

The centroid Manhattan distance $D_1\ is$:

The average Inter-Cluster distance D_2:

$$D_2\ =\ \sqrt{\frac{\sum_{i,j=1}^{N}\ \sum_{N_1+1}^{N_1+N_2}\ (x_i - x_j)^2}{N_1 N_2}} \tag{2.25}$$

The average Intra-Cluster distance D_3:

$$D_3 = \sqrt{\frac{\sum_{i,j=1}^{N_1+N_2} \sum_{N_1+1}^{N_1+N_2} (x_i - x_j)^2}{(N_1+N_2)(N_1+N_2-1)}} \tag{2.26}$$

The variance increase distance D_4:

$$D_4 = \sum_{k=1}^{N_1+N_2} (x_k - \frac{\sum_{l=1}^{N_1+N_2} x_l}{N_1+N_2})^2 - \sum_{i=1}^{N_1} (x_i - \frac{\sum_{l=1}^{N_1} x_i}{N_1})^2 - \sum_{i=1}^{N_2} (x_j - \frac{\sum_{l=N_1+1}^{N_1+N_2} x_l}{N_1+N_2})^2 \tag{2.27}$$

Furthermore, clustering feature and clustering feature tree are the two fundamental ideas of BIRCH. These ideas are used to present a cluster hierarchy and summarize clusters. A three-dimensional vector called a clustering feature (CF) contains information about clusters. A cluster's CF is described as follows:

$$\mathrm{CF} = (\mathrm{n}, \mathrm{LS}, \mathrm{SS}) \tag{2.28}$$

Where:

n represents how any points are in the cluster.

LS is the linear sum of the n points.

SS represents the square sum of the data points.

Birch algorithm works by following the main steps below:

- ***Step 1: Initialization.*** Select the Clustering Feature Tree (CF Tree)'s branching factor ("B") and the maximum number of entries ("L") that each node may accommodate. Furthermore, Create a root node at the start of a blank CF Tree.
- ***Step 2: Reading the data.*** Read the feature vectors (input data points) that we wish to cluster.
- ***Step 3: Insertion.*** For every data point:

- Use the BIRCH insertion technique to add the data point to the CF Tree.
- To keep the tree balanced and small, update the nodes' parameters.

- ***Step 4: clustering.*** To create clusters, traverse the CF Tree. At different levels of the system, clusters can emerge. On the other hand, subclusters are represented by the CF Tree's leaf nodes.
- ***Step 5: Finalizing clusters.*** If more clustering is required, proceed by merging subclusters that are strongly connected.

- ***Step 6: Cluster analysis.*** Examine the subclusters' sizes, distributions, and other features as part of the analysis of the generated clusters.
- ***Step 7: Evaluation.*** Utilize the right metrics for your dataset and application to assess the clusters' quality. The Davies-Bouldin index, the silhouette score, and domain-specific metrics are examples of common metrics.

The three following steps are optional:

- ***Step 8: Visualization.*** Use methods like PCA, t-SNE, or other dimensionality reduction techniques to display the clusters and their distribution, if appropriate.
- ***Step 9: Post Processing.*** We might need to carry out further post-processing processes, such as labeling clusters or using the clusters for subsequent operations, depending on the application.
- ***Step 10: Iterative refinement.*** We may refine the procedure by changing the parameters, the feature extraction strategy, or the clustering algorithm depending on the outcomes and the needs, as shown in figure 2.4.

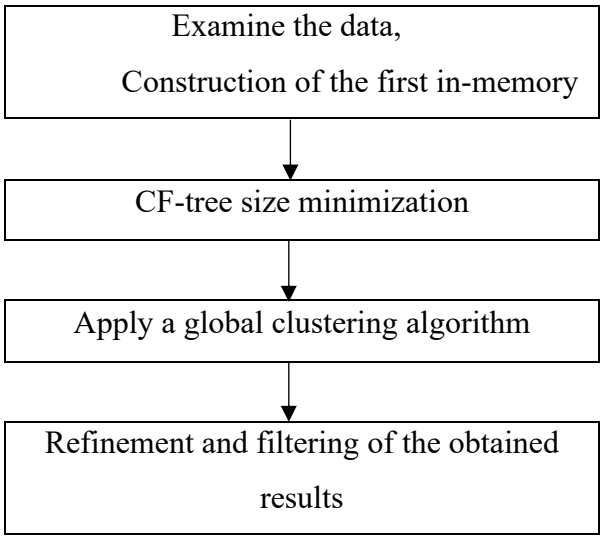


Figure 2.4 Birch algorithm diagram

It's vital to remember that the specifics of the dataset and use case may influence the implementation details and parameter selections. When dealing with data that doesn't fit into memory, the BIRCH method is very helpful since it is made to efficiently cluster enormous datasets by constructing a hierarchical structure of clusters. It’s also important to know that BIRCH needs good parameter selection [107], suitable feature extraction, and perhaps post-processing to produce meaningful clusters. The following is the pseudo code of Birch algorithm:

Algorithm 2.4: Birch algorithm

Input:

D = {t_1, t_2, …, t_n} //a set of elements

T is the threshold for CF tree construction.

Output:

K is a set of clusters.

Birch clustering algorithm:

for each $t_i \in D$

determine the correct leaf node for t_i insertion;

if the threshold condition is not violated then add t_i to the cluster and update CF triples;

else

if room to insert t_i then

insert t_i as single cluster and update CF triples;

else

split leaf node and redistribute CF features

2.3.2 marker watershed

The Watershed Algorithm [108] is a common image segmentation method that divides an image into useful segments or areas in computer vision and image processing. It's very helpful for separating things that are touching or overlapping in images. The idea behind the method is to consider the image as a topographic map, with the intensity values of the image standing in for the heights of the terrain. The plan is to flood this "terrain" with water from various "basins" or "markers" until the watershed lines, which represent the segment borders, are reached [109].

The Watershed Algorithm uses markers in the following ways:

- ***Step 1: Marker Generation.*** In this stage, you put down the first markers that denote the regions you want to divide. These indicators are frequently obtained by preprocessing of some kind or user input. Each marker has a distinctive label that designates a separate location.
- ***Step 2: Distance Transform.*** Calculate the image's distance transform. This gives each pixel a value based on how far away from the closest marker it is.
- ***Step 3: Gradient magnitude.*** Determine the image's gradient magnitude by computing the gradient magnitude. This draws attention to the corners and divisions between various geographic areas.

- ***Step 4: Marker propagation.*** Start flooding the image with markers to spread them. Both the distance transform and the gradient magnitude serve as indicators for the flooding process. The algorithm mimics the filling of the basins symbolized by the marks with water. Waters from many basins begin to mix as they get closer to one another.
- ***Step 5: Watershed Line Formation.*** The watershed lines are where the flooded zones from several markers finally converge. The targeted segment boundaries are along these watershed lines. They identify the locations when flooding fronts from various basins converge.
- ***Step 6: Label Assignment.*** Label each pixel to reflect the marker from which it was flooded. The labels of pixels that belong to the same region will match.

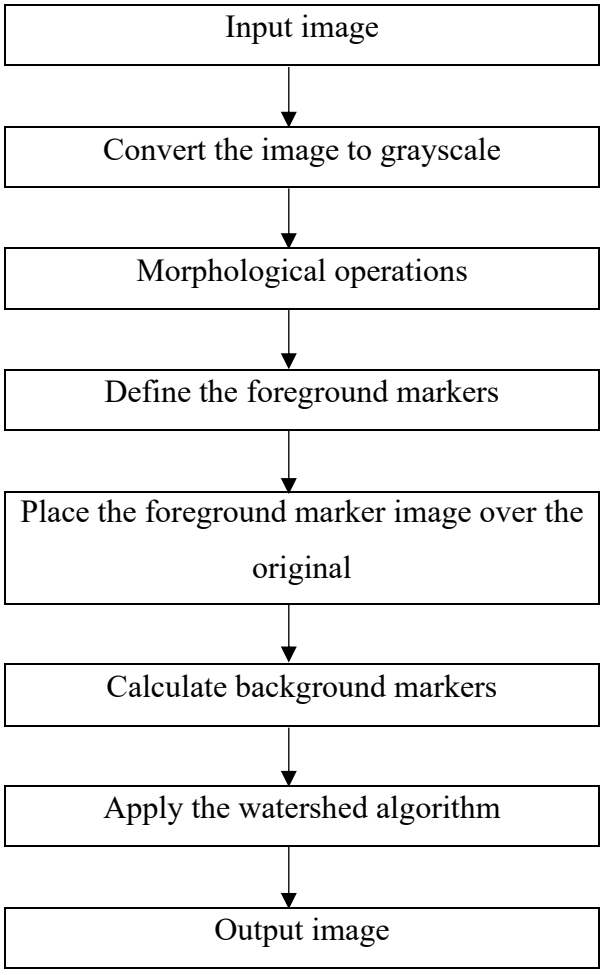


Figure 2.5 Marker watershed flowchart

Effective algorithms were created by Beucher and Meyer that model the flooding process using a priority queue, where a priority is the value of a relief element, and a lower value denotes a greater priority. A detailed flowchart is given in Figure 2.5.

The pseudocode used by Beucher and Meyer to describe all the phases that are involved in the marker-controlled watershed algorithm is illustrated in the example below:

Algorithm 2.5: Marker watershed algorithm

$Require: |R| = |M|$

$function\ Watershed\ (R, M)$

$PQ \leftarrow \emptyset$

$bg \leftarrow 0$

$for\ (i, r_i) \in R \wedge (i, m_i) \in M \wedge m_i \neq bg\ do$

$for (j, m_i) \in N_D(i, m_i) \wedge m_j = bg\ do$

$push\ (r_i, i, PQ)$

$break$

$end\ for$

$end\ for$

$while\ |PQ| \neq 0\ do$

$i \leftarrow pop(PQ)$

$for\ (j, m_j) \in N_D(i, m_i) \wedge m_j = bg\ do$

$M \leftarrow (M\ (j, m_j)) \cup (j, m_i)$

$push(max(r_i, r_j), j, PQ)$

$end\ for$

$end\ while$

$return\ M$

$end\ function$

There are several ways to apply the Watershed Algorithm, and one of them uses markers as starting seeds to direct the segmentation process. These indicators may be acquired manually (manual annotation) or via the use of methods such as morphological procedures or object detection algorithms. Though the Watershed Algorithm is a potent tool, it may occasionally be susceptible to noise and could result in too segmented output if improperly configured. As a result, it is frequently used in conjunction with other segmentation techniques or post-processing procedures to enhance the outcomes. The Watershed Algorithm is available for practical usage in several applications thanks to the OpenCV library's implementations.

2.3.3 Thresholding segmentation

A basic image processing method called thresholding divides images into discrete sections depending on pixel intensity levels. It's a straightforward yet effective technique for removing details or objects from the background of an image.

Thresholding involves setting a threshold value and comparing each pixel in the image to that value. The foreground or background class of a pixel is determined by whether its intensity is above or below the threshold.

There are several thresholding methods, including:

- ***Global Thresholding:*** In this straightforward method, the entire image is subjected to a single threshold setting. Foreground pixels are those with intensities above the threshold, whilst background pixels are those with intensities below the threshold.
- ***Adaptive Thresholding:*** Different areas of an image could have various contrasts or lighting conditions. According to the characteristics of each location, adaptive thresholding locally modifies the threshold value. This aids in adjusting for differences in illumination throughout the image.
- ***Otsu's Thresholding:*** Using Otsu's approach, a threshold value that minimizes the intra-class variation in the foreground and background is determined to be the "ideal" value. When the histogram of intensity values exhibits bimodal behavior, it is especially helpful.
- ***Hysteresis Thresholding:*** This method is employed for image edge detection. Two threshold values—a high threshold and a low threshold—are used. Edges are unquestionably those pixels that are higher than the high threshold. When coupled to high-threshold pixels, pixels between the high and low thresholds are referred to as edges.
- ***Color thresholding:*** the thresholding method used for color images is called color thresholding. To divide areas into color-based categories, thresholds must be specified independently for each color channel (such as red, green, and blue).

For many different applications, including object identification, image segmentation, and image enhancement, thresholding may be quite helpful. It does, however, have certain drawbacks. It is sensitive to changes in image quality, noise, and illumination. Thresholding might not be the ideal option when the foreground and background areas' intensity levels considerably overlap. Furthermore, thresholding occasionally produces "hard" or "binary" segmentation, in which each pixel is entirely either a part of the foreground or a part of the background. Other

segmentation approaches, such as region expanding, watershed segmentation, or deep learning-based methods, may be more suited in situations when more subtle or slow transitions are required.

In reality, many image processing libraries, like OpenCV, offer simple functions for different thresholding techniques, making it usable even by individuals who are unfamiliar with the underlying mathematics.

- **Thresholding pseudo-code is as follow:**

Algorithm 2.6: Thresholding algorithm
If f (x, y) < T then f (x, y) = 0 else f (x, y) = 255 where f (x, y) = Coordinate Pixel Value T = Threshold Value.

The thresholding formula is as follow:

$$g(x,y) = 0 \text{ if } f(x,y) < T \quad \text{and} \quad g(x,y) = 1 \text{ if } f(x,y) \geq T \tag{2.29}$$

There are several shades of grey that correspond to area 1:

$$g(x,y) = 0 \text{ if } f(x,y) < T_1 \quad \text{OR} \quad f(x,y) > T_2 \text{ and } g(x,y) = 1 \text{ if } T_1 \leq f(x,y) \leq T_2 \tag{2.30}$$

2.3.4 morphological operations

The term "morphology" often refers to a field of biology that examines the shape and form of animals and plants. To extract the image components helpful for describing area form, borders,

etc., we often utilize the same word in "mathematical morphology". Morphology is a broad range of image processing techniques that manipulate images according to their forms. A structuring element is added by morphological processes to an input image to produce an output image of the same size. When performing a morphological operation, each output pixel's value is determined by comparing it to its neighbors in the input image.

Image segmentation and morphology [110] have a little amount of overlap. Morphology comprises of techniques that may be applied either as a post-processing technique on the output of the Image Segmentation stage or as a pre-processing technique on the input data. In other words, when segmentation is finished, morphological procedures may be applied to the segmented image to correct flaws and provide details about the structure and shape of the image.

The aforementioned terms form the foundation for all morphological processing processes [111]. A matrix or a small-scale template is referred to as a structuring element and is used to navigate a image. In order to compare the structural element with the associated pixels, it is placed in every position that is feasible inside the image. Any form is possible.

- *Fit:* is when all of the pixels in the structuring element completely encircle the pixels of the target object.
- *Hit:* We refer to a situation in which at least one structural element pixel completely engulfs an object pixel.
- *Miss:* We refer to a situation in which the object's pixels are not covered by any pixels in the structuring element.

Morphological image processing and spatial filtering are fundamentally comparable [112]. Every pixel in the original image has the structural element moved across it to create a pixel in the newly processed image. The morphological process used determines the value of this new pixel. Erosion and dilation are the two most often employed procedures.

a- Erosion:

Erosion includes the areas in an image's borders getting smaller. A structuring element is also moved across the image, and a pixel is only changed to 1 in the output image if all of its associated pixels in the input image are likewise 1. Erosion is helpful for shrinking items, eliminating minute noise, and dismantling interconnected systems.

Erosion reduces the size of the image's pixels, or it eliminates pixels at object boundaries. To conduct an erosion operation, we first traverse the structuring element across the image object. The following equation is used to determine the values of the output pixels.

$$\begin{aligned} \text{Pixel (output)} &= 1 \ \{\text{if FIT}\} \\ \text{Pixel (output)} &= 0 \ \{\text{otherwise}\} \end{aligned} \tag{2.31}$$

The erosion operation may be described mathematically as follows:

Let's refer to the structural element (kernel) as B and the input image as A. A tiny binary matrix serving as the structural element is usual, with the 1s denoting the form that will be taken into account throughout the erosion procedure. Furthermore, at a certain pixel position (x, y), the erosion of the input image A by the structuring element B may be estimated as:

$$(A \ominus B)(x, y) = min_{(i,j)\in B}\{A(x + i, y + j)\} \tag{2.32}$$

Where:

- (x, y) is the location of the pixel in the input image.
- (i, j) indicates the position of the pixel in the structuring element B.
- $A(x + i, y + j)$ is the value of the pixel in the input image at the position $(x + i, y + j)$
- $min_{(i,j)\in B}$ determines which pixel value in the area that the structuring element covers has the lowest value.

The eroded image $(A \ominus B)$ that is produced will contain pixel values that correspond to the lowest values found in the areas that the structuring element covers as it advances across the input image. Moreover, that the binary images with black (0) or white (1) pixels correspond to the use of this erosion algorithm. Similar techniques are frequently used for the erosion procedure on grayscale images, only intensity values are used instead of binary data.

b- Dilation:

Dilation entails extending the borders of image portions. In order to achieve this, a structuring element a preset form, frequently a tiny binary matrix is moved across the image, and if at least one pixel in the structuring element is 1 in the corresponding region of the input image, the pixel in the output image is set to 1. Dilation assists in object enlargement, gap filling, and connecting nearby structures.

Pixels in the image are enlarged by dilation, or object boundaries are given more pixels. To conduct a dilation operation, we first traverse the structuring element across the image object. The following equation is used to determine the values of the output pixels.

$$\begin{aligned} \text{Pixel (output)} &= 0 \ \{\text{if FIT}\} \\ \text{Pixel (output)} &= 1 \ \{\text{otherwise}\} \end{aligned} \tag{2.33}$$

The dilation method can be described mathematically as follows:

Let's refer to the structural element (kernel) as B and the input image as A. A tiny binary matrix serving as the structural element is usual, with the 1s denoting the form being taken into account for the dilation process.

At a particular pixel point (x, y), the structuring element B can dilate the input image A as follows:

$$(A \oplus B)(x, y) = max_{(i,j)\in B}\{A(x - i, y - j)\} \tag{2.34}$$

Where:

- (x, y) is the location of the pixel in the output image (dilated image).
- (i, j) indicates the position of the pixel in the structuring element B.
- $A(x - i, y - j)$ is the value of the pixel in the input image at the position $(x - i, y - j)$
- $max_{(i,j)\in B}$ determines the maximum value among all the pixel values in the area the structuring element covers.

The pixel values in the resulting dilated image $(A \oplus B)$ indicate the highest values found in the areas that the structuring element covers as it advances across the input image. This dilation equation holds true for binary images just like it does for erosion. Similar techniques are used to conduct the dilation procedure for grayscale images, only intensity values are used instead of binary data.

c- Compound operations:

In order to accomplish a more complex or higher-level operation, various elementary operations or processes are combined or composed to form compound operations. Dilation and erosion can be used as the foundation for more intricate procedures, such as:

- ***Opening:***

In morphological image processing, the opening operation combines an erosion operation with a dilatation operation that both use the same structural element (kernel). In binary or grayscale

images, opening is generally used to eliminate minute noise, minute features, and tenuous relationships between objects. It is beneficial to keep the larger buildings and demolish the smaller ones.

The opening operation can be described mathematically as:

$$Opening\ (A, B) = (A \ominus B) \oplus B \tag{2.35}$$

Where:

- A represents the input image.
- B represents the structuring element or Kernel.
- $\ominus$ is the Erosion operation.
- $\oplus$ is the Dilation operation.

The following steps are involved in the opening operation:

- Use the structural element B to erode the input image A.
- Apply the same structural element B to the erosion's resultant dilation.

The result of the opening operation is the removal of tiny and thin structures and the preservation of bigger structures. It works particularly well in applications where eliminating tiny structures and reducing noise is crucial. The main anatomical components can be preserved while removing minor speckles, for instance, in medical imaging.

- ***Closing:***

  In morphological image processing, the closure operation combines a dilatation operation with an erosion operation that both use the same structural element (kernel). In binary or grayscale images, closing is generally used to blend neighboring structures, close minor gaps, and smooth object borders.

  The closing operation can be mathematically described as:

$$Closing\ (A, B) = (A \oplus B) \ominus B \tag{2.36}$$

  The following steps are involved in the closing operation:

- Utilizing the structural element B, dilate the input image A.
- Utilizing the identical structural element B, do erosion of the dilation's outcome.

The closing method can be used to combine neighboring structures that are separated by small gaps, close tiny gaps between objects, and smooth out object borders. Applications like object identification and segmentation, where it's vital to join disjointed structures and make more cohesive areas, may benefit from its use.

- ***Morphological Gradient:***

In image processing, the morphological gradient is a morphological method that is used to emphasize object borders or edges. It is produced by employing a structural element (kernel) to calculate the difference between the dilation and erosion of an image. The intensity transitions or shifts between various sections of an image are highlighted by this process. The morphological gradient operation can be described mathematically as follows:

$$Morphological\ Gradient\ (A,B) = (A \oplus B) - (A \ominus B) \tag{2.37}$$

The steps involved in the morphological gradient process are as follows:

- Employing the structural element B, dilate the input image A.
- Apply erosion to the original input image A using structuring element B.
- Calculate the difference between the outcomes of erosion and dilation element-by-element.

Indicating the presence of object borders or edges, the generated image will emphasize regions where there are abrupt intensity variations. As a preprocessing step for more complex image processing tasks like object identification, image segmentation, and feature extraction, this technique is particularly helpful for edge detection.

The degree of sensitivity to various edges or structures within the image may be controlled by changing the size and form of the structuring element. Smaller structural components will draw attention to more minute features, whilst bigger components will highlight wider transitions.

- ***Top Hat and Black Hat:***

A combination of an image and its morphological opening or closure is used in the top hat and black hat procedures, two morphological image processing approaches. These processes are used to emphasize particular aspects of a image, such as small, bright structures against a backdrop that is darker or small, dark structures against a background that is lighter.

a- Top Hat Operation:

Top hat refers to a particular kind of morphological operation employed in image processing and computer vision in the context of morphological operations. The top hat operation is described as the difference between an image's opening and the input image when a structuring element is used. It may be stated mathematically as:

$$Top\ Hat\ (A,B) = A - Opening\ (A,B) \tag{2.38}$$

Small, brilliant structures or elements that were left behind during the opening operation are highlighted by the top hat operation. The finer features are preserved while the backdrop is essentially removed from the original image. The top hat operation has applications in the detection of minute details, such as microcalcifications in medical images or minute surface scratches in industrial inspection.

Furthermore, the top hat operation is helpful for emphasizing particular aspects of an image, such as spotting tiny bright areas against darker backgrounds or emphasizing details that stand out from the surrounding areas.

b- Black Hat Operation:

The discrepancy between the input image's closure and the input image itself is known as a black hat operation. It may be written mathematically as follows:

$$Black\ Hat\ (A,B) = Closing\ (A,B) - A \tag{2.39}$$

The closure process leaves behind certain tiny, dark structures or features, which are highlighted by the black hat operation. It basically removes the original image's smoothed foreground, exposing the darker elements against a lighter backdrop. The black hat operation can be used to identify abnormalities like material fractures or manufacturing process flaws. Furthermore, in situations when improving particular kinds of characteristics or structures is vital, top hat and black hat operations are both helpful. The qualities of the components you wish to emphasize or remove from the image will influence the structuring element and operation you choose. Therefore, morphological operations are flexible image processing procedures with a wide range of uses, including character identification, industrial inspection, medical imaging, and more. They are especially useful for editing binary (black and white) images, but by taking intensity values into account, they may also be used to grayscale images.

2.3.5 Evaluation metrics

The evaluation metrics used in the second contribution are the same used in the first one, which are the dice coefficient, Jaccard index, sensitivity as well as specificity. Therefore, all these techniques are well explained in the previous section.

2.4 Algorithms used in the third proposed method: "License plate text recognition using deep learning, NLP, and image processing techniques"

Text extraction and recognition are procedures that entail taking significant data from a variety of text sources, including scanned documents, images, PDFs, and even handwritten text, and transforming it into a digital format so that it may be processed, analyzed, and used to a variety of tasks. Fundamental to many applications, such as document digitization, automated data entry, information retrieval, and natural language processing (NLP), are these processes.

Locating and extracting text from various documents or images is known as text extraction. This includes locating and removing specific fields like names, dates, addresses, or invoice numbers as well as paragraphs, phrases, and even individual words. Text extraction methods include:

- OCR, or optical character recognition, which is a technology that turns scanned text into editable, searchable text. Text may be easily extracted from scanned documents, PDFs, and photos using this method.
- Regular expressions and pattern matching, that are two methods for identifying and extracting particular sorts of text by using established patterns or rules.
- Techniques for Natural Language Processing (NLP), where a text can be extracted using NLP models in accordance with linguistic patterns, named entities, or certain environmental cues.

Moreover, the process of text recognition entails locating and transforming text from an image or document into machine-readable text. This is the first step in the process of extracting text.

Text recognition methods include:

- OCR (as previously mentioned).
- Models for deep learning: To identify and extract text from images or scanned documents, neural networks and deep learning models, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), are frequently utilized.

2.4.1 *OCR algorithm*:

Algorithm 2.7: OCR algorithm

```
FUNCTION OCR (image):
# Preprocessing
    preprocessed_image = preprocess(image)
# Text extraction
   extracted_text = extract_text(preprocessed_image)
# Post-processing
   processed_text = postprocess(extracted_text)
   RETURN processed_text

FUNCTION preprocess(image):
# Image enhancement (e.g., noise reduction, contrast enhancement)
   enhanced_image = enhance(image
# Binarization (convert to binary image)
   binary_image = binarize(enhanced_image)
   RETURN binary_image

FUNCTION extract_text(image):
# Optical Character Recognition (OCR) to recognize text and symbols
   recognized_text = apply_ocr(image)
   RETURN recognized_text

FUNCTION postprocess(text):
# Post-processing steps such as error correction, formatting, etc.
   processed_text = clean_and_format(text)
   RETURN processed_text

FUNCTION enhance(image):
# Implement image enhancement techniques (e.g., denoising, sharpening)
# Return the enhanced image
```

```
FUNCTION binarize(image):
# Convert the image to binary format using thresholding or adaptive methods
# Return the binarized image

FUNCTION apply_ocr(image):
# Use a specialized OCR model to recognize characters, symbols, and text
# Return the recognized text

FUNCTION clean_and_format(text):
# Perform post-processing steps like error correction, formatting, etc.
# Return the cleaned and formatted text
```

Each of the functions (enhance, binarize, apply_ocr, clean_and_format) in a practical implementation would have a more particular algorithm or group of algorithms suited to the relevant task. These procedures would be properly included by the OCR function to extract text from the provided image.

2.5Algorithms used in the fourth proposed method: "Reinforcement learning"

A type of machine learning called reinforcement learning (RL) [113] teaches an agent to make decisions by interacting with its surroundings. The agent's objective is to develop a policy and a method for choosing actions that maximizes a cumulative reward signal over time.

The following are crucial elements of reinforcement learning [114]:

- *Agent:* The person who makes decisions and engages with the environment. The agent makes activities and develops the ability to tailor its behavior to meet predetermined objectives.
- *Environment:* The system outside of the agent that it communicates with. Based on the agent's behaviors, the environment gives the agent feedback in the form of incentives.
- *State:* The agent utilizes this as a representation of the current state of the environment for making decisions. The state gives details about the environment that the agent is working in.
- *Action:* The decisions the agent takes to change the surroundings. Future states and rewards for the agent can be affected by an action in both the short- and long-term.

- *Reward:* A scalar number that represents the immediate cost or benefit of performing a given action in a certain situation. To maximize the cumulative reward over time is the agent's objective.
- *Policy:* A plan that links conditions to responses. The agent's goal is to figure out the best course of action to take in order to maximize the projected cumulative payoff.
- *Value Function:* A function that calculates the anticipated total gain from a specific state or state-action pair. Value functions evaluate the attractiveness of various states or actions to assist the agent in making decisions.

The agent acts in the environment, receives rewards, and uses these experiences to modify its policy as part of the reinforcement learning process. Different techniques are used by reinforcement learning algorithms for both exploitation (selecting actions that are known to produce large rewards) and exploration [115] (performing novel actions to learn more about the environment). Exploration to find better policies and exploitation of information already known to maximize rewards are frequently trade-offs made during the learning process. Numerous industries, including robots, gaming, banking, healthcare, autonomous cars, recommendation systems, and more have found use for reinforcement learning. Popular RL algorithms include Actor-Critic techniques, Proximal Policy Optimization (PPO), Deep Q Networks (DQN), and Q-learning. These algorithms employ a number of strategies to deal with issues such as the credit distribution problem, the exploration-exploitation conundrum, and managing continuous state and action spaces. Reinforcement Learning (RL) may be used to automate command and control processes for a variety of image processing issues. Here are some examples of how RL may be applied to image processing:

- *Autonomous Navigation and Object Tracking:* RL algorithms may be used to teach robots or other agents to independently navigate using visual input. The agent has the ability to decipher visual cues and take appropriate action to follow pathways, avoid hazards, or follow moving objects.
- *Image restoration and denoising:* RL may be used to create agents that can learn to fix noisy or damaged images. The agent picks up a technique for applying the right filters or changes to improve image quality while retaining crucial information.
- *Super-Resolution Imaging:* RL algorithms may be used to boost finer features in low-resolution images by increasing image resolution. Based on low-resolution inputs, the agent learns to anticipate high-resolution information.

- *Image segmentation and object recognition:* RL can help in the training of agents to conduct image segmentation, where the objective is to distinguish objects from their background. The agent gains the ability to recognize areas of interest and define object borders.
- *Adaptive image compression:* By choosing the best compression settings depending on the image content and striking a balance between compression ratios and image quality, RL algorithms may learn to compress images successfully.
- *Image-Based Game Playing:* RL agents may learn to play games like Atari or board games that need visual input. The agent watches the game screen and picks up tips on how to play to win or gain high points.
- *Image-Based Medical Diagnostics:* RL may be used to create tools that examine X-rays or MRIs as well as other medical images to help diagnose disorders. The agent gains the ability to recognize patterns linked to particular situations.
- *Content creation and Style Transfer: RL* algorithms are capable of creating new images by learning from previously collected visual input. RL may also be used to direct style transfer procedures to modify the style of an image to another.

Deep neural networks are frequently used in these applications' real-time learning to interpret images and reach choices based on the visual data. The RL agent selects behaviors that result in desired outcomes after receiving input images, processing those images using convolutional neural networks (CNNs) or other architectures. The agent learns through trial-and-error exploration to identify behaviors that provide larger cumulative rewards, which frequently necessitates the creation of suitable reward functions to direct the agent's activity. Therefore, in numerous sectors, productivity, and accuracy may be increased by combining RL with image processing to create automated and intelligent systems that can carry out difficult tasks using just visual input.

2.6 Conclusion

In this chapter, we've presented a detailed explanation of every used algorithm in this thesis, such as k-means clustering, deep learning methods, thresholding, Birch algorithm and more. Moreover, we provided the mathematical operations as well as the algorithms that we used for the development of the contributions mentioned above.

Chapter 3:

Image Segmentation and object Detection - Text extraction and recognition

3.1 Introduction:

Image segmentation is defined as a computer vision task that entails splitting an image into relevant and unique segments or areas. Image segmentation aims to break down an image's representation into easier-to-understand, analyze-able chunks. Typically, each segment in an image represents a different object, area, or interesting element. The basic goal is to isolate pixels from those from other segments and to group pixels with similar characteristics together. As a result of the richness and variety of real-world images, image segmentation is a challenging problem in computer vision. For segmentation findings to be accurate and reliable, several obstacles must be overcome. On the other hand, identifying and localizing things of interest inside an image or video frame is the basic goal of object detection in computer vision. Object detection seeks to locate things precisely within the visual data as well as categorize them. Bounding boxes are drawn around the observed items to do this. For many applications, including as image interpretation, robotics, surveillance, autonomous cars, and more, object detection is an essential building element. The subject of object recognition is one that is always evolving, with gains in accuracy, speed, and the capacity to handle challenging settings and situations. For more complex activities like instance segmentation and tracking, it serves as a basis.

Our main objective in this thesis is to developpe new methods for image segmentation as well as the detection of several objects in images. Moreover, these methods were elaborated using some already existing methods in the literature. The rest of this chapter will be organized as follows:

- ***Section II):*** A Brain Tumor Segmentation and Detection Technique Based on Birch and Marker Watershed.
- ***Section III):*** A hybrid skin lesions segmentation approach based on image processing methods.
- ***Section IV):*** Text extraction and recognition method for license plates.

3.2 A Brain Tumor Segmentation and Detection Technique Based on Birch and Marker Watershed.

3.2.1 Introduction:

medical imaging has a radical role in all types of brain tumor extraction, it can identify and detect brain tumors in their early stage, and it even allows doctors to have a classification of the tumor detected. Many medical imaging techniques are used in different circumstances; like nuclear medicine, magnetic resonance imaging, ultrasound, and several techniques. When discussing medical imaging, we are certainly looking for a method or an operation of visual representation of the internal of the human corpus. Medical imaging is employed in the case of clinical analysis as well as medical involvement, such as uncovering any hidden structure by skin or bones. It is also very useful when surgeons are trying to identify and treat illnesses. Over the past 10 to 15 years, the hardest part of acquiring these images has improved tremendously, in resolution, quality, signal-to-noise ratio, and the way doctors can look inside each patient's body and try to figure out what is wrong and what to do. Image processing is employed in various regions of computer vision, specifically, image segmentation, 3D reconstruction, the Auto-calibration camera. Image segmentation represents the operation of subdividing an image into various meaningful segments or partitions that will be easier to examine. Segmentation takes its name from the segmentation or division of the pixels in the image into regions of interest, otherwise known as ROIs. In some software, this step may be called segmentation, but it could also be called object identification or detection. The regions of interest may be displayed using outlines, bounding boxes, or as arbitrary colored collections of pixels called a matrix of labels. Because each separate entity receives a different digital label.

Among the major applications in the medical area is to easily detect brain tumors or to identify abnormalities in general. We can also use it for blood cell segmentation in the case of leukemia patients, where there are infected blood cells. Among the existing medical image segmentation methods that we have, we can list the Watershed algorithm, the graph-based method, region growth, binary segmentation, and the Fuzzy c-means algorithm. In this article, a novel method based on a Watershed algorithm is introduced. The watershed algorithm considers the gradient of the required image. The main idea of this algorithm is to build barriers between the different regions of the image to avoid fusion. In the end, these barriers will present the result of the segmentation. The second algorithm is Birch, which is used in the case of large data with limited storage space. The rest of the paper will be presented as follows; starting with the part of the

related work, where we will discuss the different methods and approaches already proposed in the field of detection and segmentation of brain tumors. Followed by the third part devoted to the proposed method. Then comes the fourth part of the article, where we present the results obtained from our proposed method, followed by a comparative study between the proposed method and the two used techniques (Birch and Watershed) by employing several evaluation metrics. Furthermore, we mentioned a set of some proposed methods in the state-of-the-art (SOTA) and made a comparison with our proposed method. At the end of the article, we discussed the reason behind choosing the traditional image processing techniques over the deep learning models in terms of training dataset size, computer power, and training time. The obtained results are very favorable and satisfactory. However, researchers are still facing many difficulties and challenges owing to the huge processes of brain tumor datasets. It has always been very challenging to come out with a method or an algorithm that can deal with the immense size of the datasets. That was the primary motive behind choosing the Birch algorithm over the other segmentation techniques, due to its capacity to handle the huge number of images in the dataset. Furthermore, because of the different types of brain tumors, it's always so challenging to recognize and extract the tumor from the MRI image.

3.2.2 Proposed method

In this current section, we will present our suggested method which presents two essential steps. The first phase of the algorithm starts by applying a filter to attenuate the noise of the given image. The one we used here is the Median filter which is a non-linear filter employed to eliminate the existing noise in the original images, this will help us to have clearer results later. The resulting image will then be passed through the renowned Birch segmentation algorithm, with its ability to process huge databases, and also its performance in terms of execution time and results obtained in just one scan of the dataset. The other main reason behind choosing Birch over the other existing methods for image segmentation is its capability to deal with non-uniform gray sections. Birch algorithm will be explained in detail in the next second subsection. This first stage of the proposed method is called image processing, which contains two main tasks. The first task is to pre-process the input image to remove artifacts, the second task is to apply a segmentation method to locate the tumor.

The second part of the algorithm is to use the Watershed method for the first output image. In this phase of the program and after preprocessing the image and applying the Birch segmentation algorithm, we move on to the last step of tracing the contours around the detected

tumor. The diagram in Figure 3.1 above explains the sequence in detail of the steps of the proposed method:

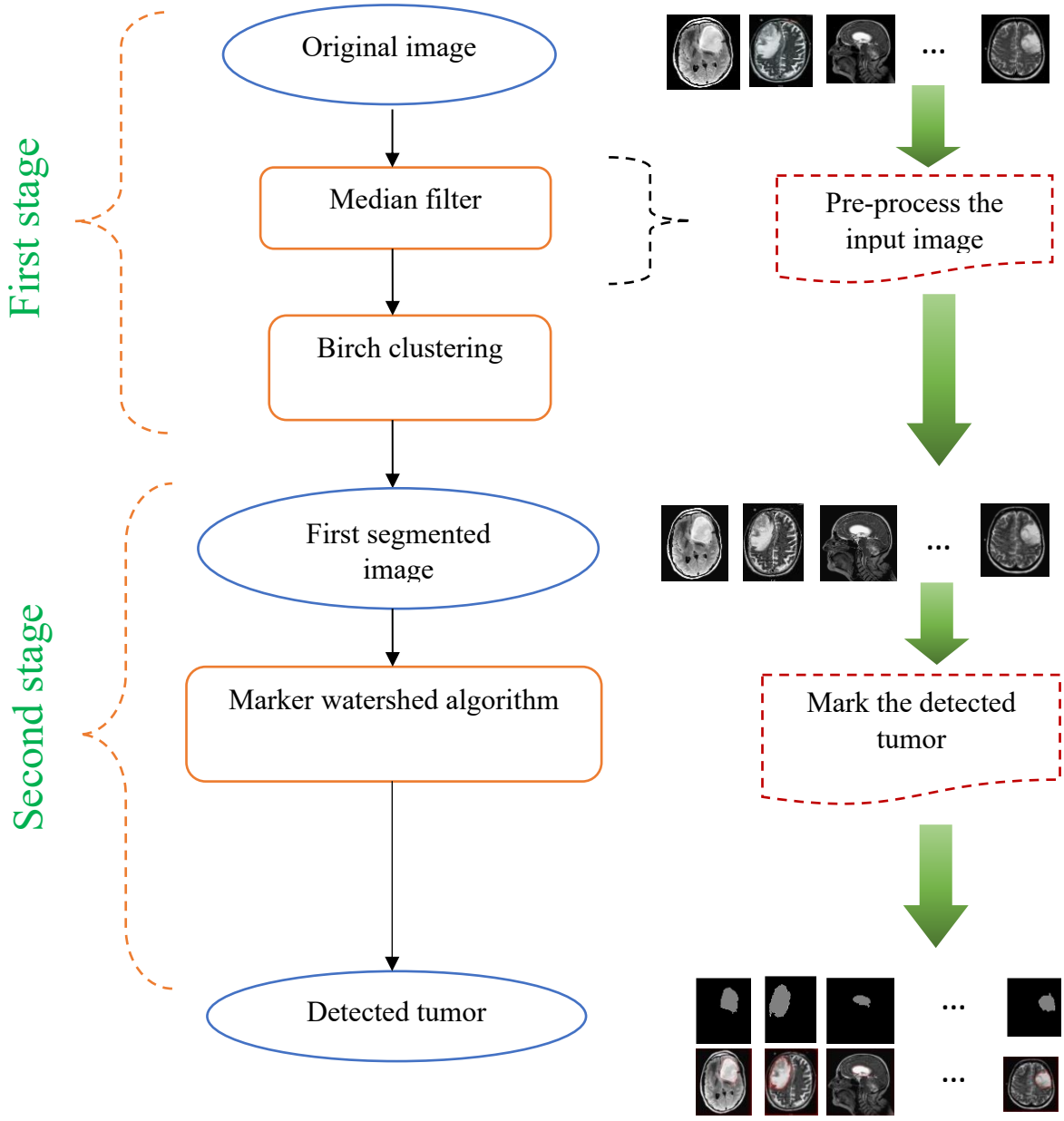
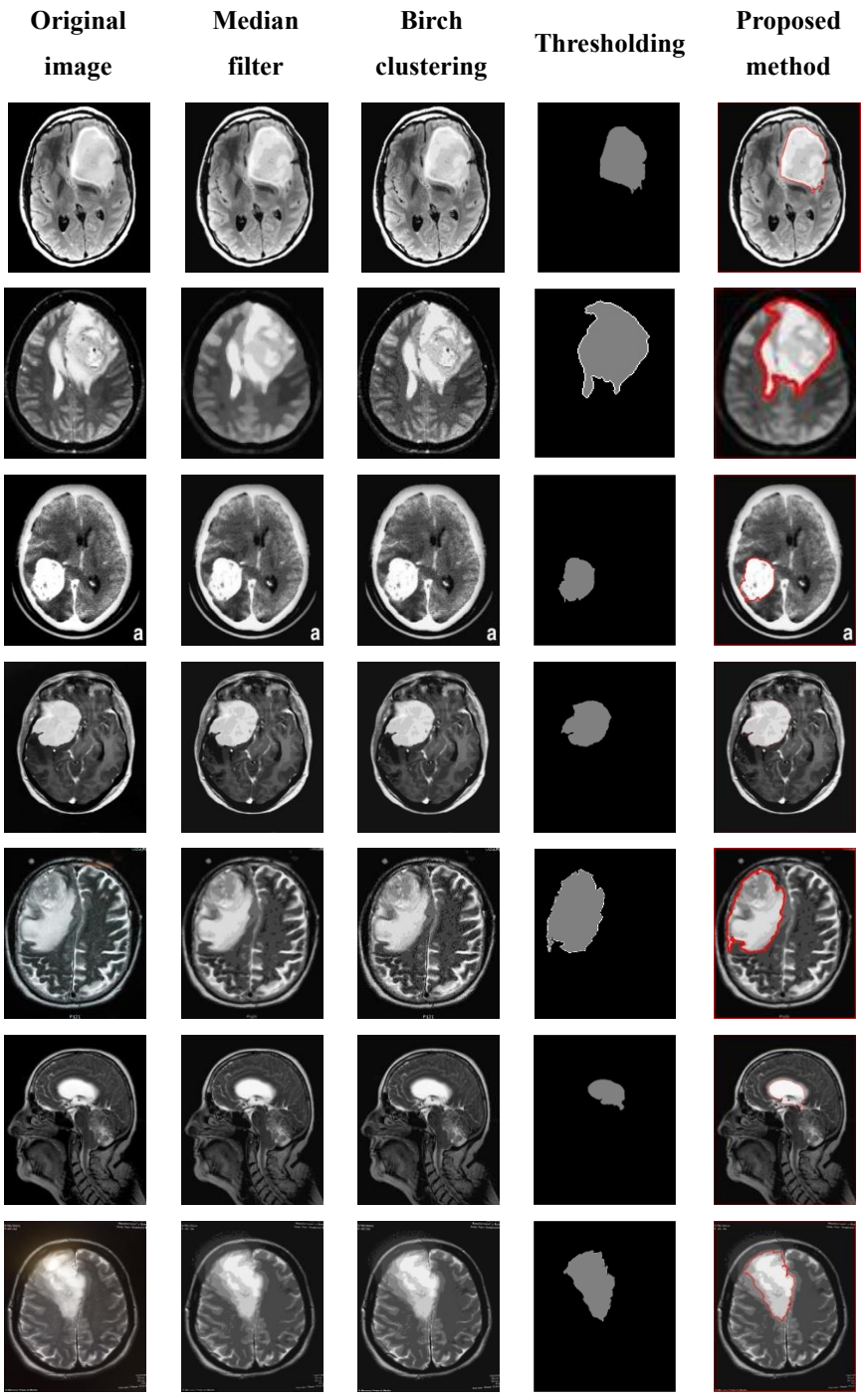


Figure 3.1 Flowchart of the first proposed method

3.2.3 Results

The MRI images used in this part of the experiment belong to Kaggle datasets. Showing different types of tumors located in different locations. We used 155 images in the previously mentioned dataset to train our algorithm. The first step of the proposed technique is to do a preprocessing for the MRI images. This step is very important, as the original images contain

noise. For this, we used the Median filter which is highly recommended in the case of medical images. The Birch clustering algorithm will be then applied to the processed image to partition it into subgroups or clusters, which will subsequently help us to better segment the tumor. In the last step, we'll apply the watershed algorithm to mark the detected tumor. The following Figure 3.2 represents some of the results obtained from our proposed algorithm following the steps explained above.



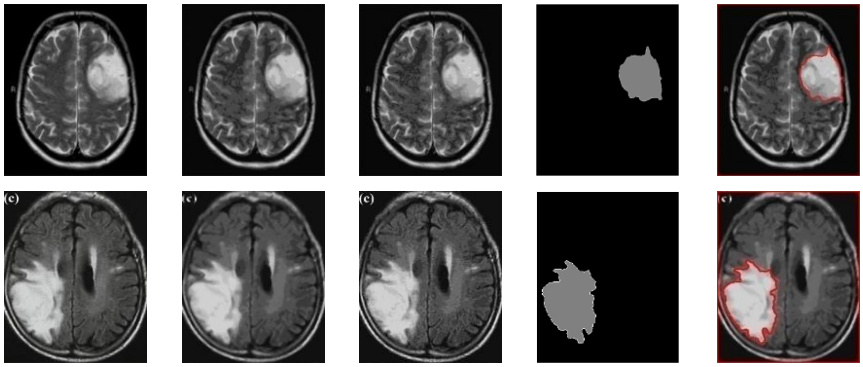


Figure 3.2 Obtained results for the first proposed method

To check and validate the suggested technique, we used several evaluation metrics as Sensitivity, Dice coefficient, Specificity, Jaccard Index, and Shannon Entropy.

Image 1	Dice coefficient	Sensitivity	Specificity	Jaccard index
The proposed method	0.77	0.99	0.98	0.65
Birch	0.76	0.99	0.15	0.61
Watershed	0.75	0.98	0.16	0.61

Table 3.1 Comparison of the proposed technique and the two segmentation methods for the first image in Table 3.1

Image 2	Dice coefficient	Sensitivity	Specificity	Jaccard index
The proposed method	0.97	0.98	0.96	0.94
Birch	0.94	0.98	0.1	0.93
Watershed	0.94	0.95	0.43	0.94

Table 3.2 Comparison of the suggested technique and the two segmentation methods for the second image in Table 3.1

From the tables above: Table 3.1 and Table 3.2, it can be deduced that the suggested method gives remarkable results compared to the other two segmentation methods. Furthermore, the result images of the proposed algorithm show a great convergence in tumor detection, that is, the tumor is very well marked and surrounded. Otherwise, Birch clustering alone didn't perform well in some metrics, which can be caused by the principle of reducing the number of data points

in the dataset. As said before, some clustering features work on reducing and tightening the size of the clusters. On the other hand, we notice that the second algorithm which is marker watershed, when applied alone, didn't perform well in some evaluation metrics, for example in the first image it had only 0.16 in specificity. On the other hand, by comparing our proposed method with the already existing ones, we deduce that we have better results. In [116] for example, the authors presented a hybrid method based on Berkeley Wavelet Transformation (BWT) and SVM. This algorithm achieved 97.72 % for the sensitivity and a specificity of 94.2 %, while our proposed method obtained 99% in sensitivity and 98% in specificity. The same thing in [117] where authors have achieved a specificity of 90% and 95 % for the sensitivity. as shown in Table 3.3 bellow.

	Sensitivity	Specificity	Dice coefficient	Jaccard index
The proposed method	99%	98%	96%	94%
Paper [19]	96%	-	88%	87%
Paper [20]	97.72%	94.2%	-	-
Paper [21]	95%	90%	-	-
Paper [38]	-	91%	86%	-
Paper [39]	94%	-	92%	-
Paper [40]	-	-	88%	-
Paper [41]	-	-	89%	-
Paper [42]	-	-	91%	-
Paper [43]	-	-	89%	-

Table 3.3 Comparison of the suggested approach and some other proposed methods

3.2.4 Discussion

The choice of image processing techniques over deep learning methods ([118], [119], [120]) wasn't random. In many computer-vision problems, classic image processing is chosen for cases like edge detection and computing, like in this study where the main objective is to extract the tumor area that is based on pixel intensity variation. Also, it's a preferred alternative in our case with its simple color thresholding technique. Furthermore, the features that are learned from the deep neural network which are specific to the training dataset should be well constructed, otherwise, it won't proceed well for images that are different from the training dataset. On the other hand, by using the hybridization between the two already existing methods

which are Marker watershed and Birch algorithms we achieved better results as shown in Table 3.1 and Table 3.2. By comparing both traditional and deep learning techniques, where most of the times users employ a cascade of convolutional neural networks (C-CNNs) ([121], [122]), we figured out that deep neural networks require a large training data besides high stockpiling and computing power, unlike the classic image processing techniques that need only a small amount of data to verify the performance of the proposed method. In addition to that, deep learning needs a long time for training and annotation, contrary to traditional image processing. It also gives transparency and flexibility to computer-vision engineers to adjust the parameters in the algorithm to achieve better results. Another important factor is the training time that is so short in the case of the traditional image processing techniques compared to deep learning models. The proposed method gives to doctors a good brain tumor segmentation and detection with low computer power, small training dataset and short training time that was within 8 seconds to give the final detected tumor. Moreover, the proposed method was compared to SOTA (state-of-the-art) methods ([123], [124], [125]) that have participated in the BraTS 2020 challenge [124] for brain tumor segmentation, and by comparing our obtained results with the Whole Tumor Target scores of the three approaches, we conclude that we achieved satisfactory results for the brain tumor detection.

3.3 A hybrid skin lesions segmentation approach based on image processing methods

3.3.1 Introduction:

Skin cancer occurs when cells in the epidermis have turned into cancerous cells. Different types of skin lesions are determined by which cells in the epidermis turn cancerous. Among the skin cancer types, we can cite the nonmelanoma and the melanoma types. The nonmelanoma will include basal cell carcinoma, and squamous cell carcinoma including actinic keratosis which can turn into squamous cell carcinoma. Furthermore, the melanoma type remains one of the most highly aggressive types of skin cancer of all the types listed before. Non-melanomaskin cancers happen on the extremities, face, and any place we get a lot of chronic sun damage. Chronic meansthat the more sun exposure we get, the higher the risk will be. Melanoma happens also in other areas where thesun doesn't shine as constantly, like on the legs. The other characteristic we want to look for is changing, so an area that's changing more than the rest of the body.We all have spots and lumps that are relatively stable, but cancer generally isn't. So, it'll continue to evolve

and change. Everybody at least is on the list of being at risk of skin cancer, and early detection means that you'll have a smaller scar while the surgical repair, or sometimes we can treat very early non-melanoma skin cancers with non-surgical techniques such as creams or another kind of ablative techniques. The best ways to protect ourselves especially in the case of having fair skin, when the risk for UV damage is high is by avoiding the sun between 10 and 2 PM, wearing sun-protective clothing, applying sunscreens appropriately. Medical imaging is a fascinating area, the challenges are incredible and so are the rewards. When we solve problems in medical imaging, we're helping humanity, and that's a great reward. Medical imaging is very different than many other areas of image processing. One of the differences is that very often we start the investigation with a problem presented to us such as brain tumor, leukemia, and skin cancer detection and segmentation. Image processing can be used in many different types of computer vision, particularly image segmentation, 3D reconstruction, the auto-calibration camera. Image segmentation is the mechanism that gives the capacity to move apart objects from the background to make these objects easier to analyze later. Image segmentation helps to identify individual structures within the image, like in our case, finding regions of the skin that contain tumors. In other words, we'll classify each pixel in the skin image as a tumor or not tumor. To segment an image, we can choose over many approaches, depending on the nature of the problem to solve. Among these approaches, we can list pixel-based methods like thresholding and clustering. The main idea of these methods is to group pixels in the input image based on color similarities or the intensity of pixels. The second example is the method based on edges when the algorithm's main objective is to identify the change of state of the intensity between pixels. Region-based algorithms are the third type of image segmentation when the purpose is to create larger and homogeneous areas by comparing every pixel with its neighbor. The last type of image segmentation is the hybrid approach when we use to combine two already existing methods to give birth to a new one with better results and performances. In this article, we'll be presenting a novel technique for detecting and segmenting skin tumors, based on the hybridization between fuzzy c-means algorithm and marker watershed method using Otsu's algorithm. The work will be organized as follows; we'll start by listing the previous and different methods that have been proposed in the area of skin tumor detection. The third part will be consecrated to discuss our proposed method, followed by the part when we'll compare the obtained results with the two methods that have been used using several evaluations metrics such as Shannon entropy, Jaccard Index, Specificity, Sensitivity and dice coefficient that all have shown

satisfactory results.

3.3.2 Proposed method

In this unit, we will be talking about our proposed process for skin cancer detection and segmentation. For this,the first task was to gather images from the web and datasets such as the Kaggle dataset. The second step is the preprocessing phase when we apply the median filter. It's a non-linear filter that is used for reducing the noise in the original image. In the next step, we opted for fuzzy c-means clustering which is a soft clustering with the ability to give to a pixel the capacity to be in one or more clusters at the same time. The resulting image willbe after that going through the next phase of the proposed method. The next step is presented by Multi Otsu's algorithm that looks at the histogram of the image, including the pixel values and the property to segment the object that we want. After that, we'll use the marker watershed algorithm for the segmentation. The following Figure 3.3 shows in detail the sequence of the different stages in the proposed method.

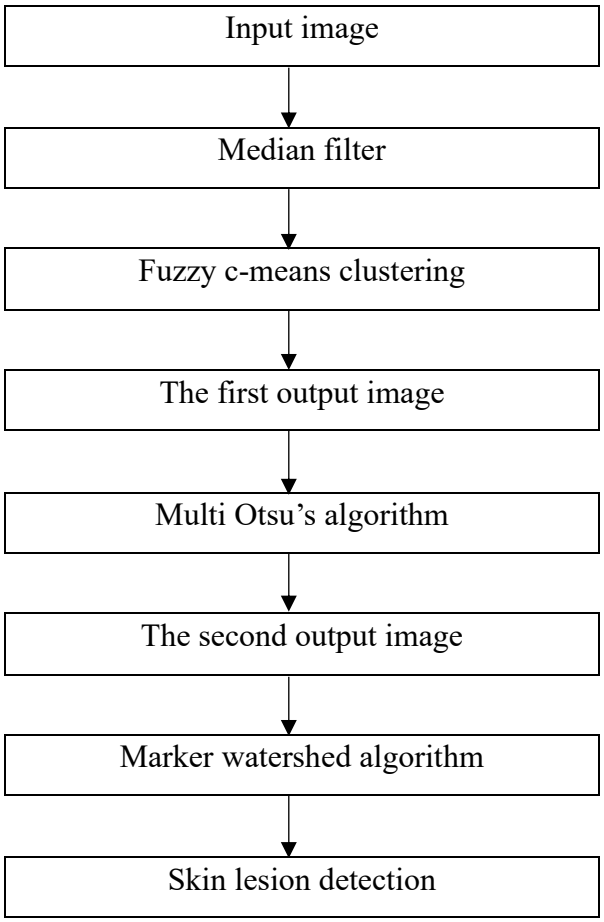


Figure 3.3 Flowchart of the second proposed method

3.3.3 Results and discussion

In this part of the paper, we will discuss the proposed process which is made of three phases. The first thing wehad to do is to gather the medical images showing skin lesions. For this, we had to collect about 300 images to test our proposed algorithm. The second stage is to go through the preprocessing phase. At this stage, the main purpose is to clarify the image, which means to remove noises so that the results will be more efficient. Here, we've chosen the median filter over the other filters because it has good results for pepper and salt enhancement. In addition to that, it doesn't blur edges which is very interesting and advantageous. After that, we opted for fuzzy c-means segmentation which is a soft clustering that as said before, gives the chance for a pixel to be in one or more clusters at the same time. This resulting image will then be used as an input image to the next method, the marker watershed algorithm. The purpose behind choosing the marker watershed algorithm is to finally mark the boundaries of the detected skin lesion. The Figure 3.4 below shows the obtained result of the proposed process.

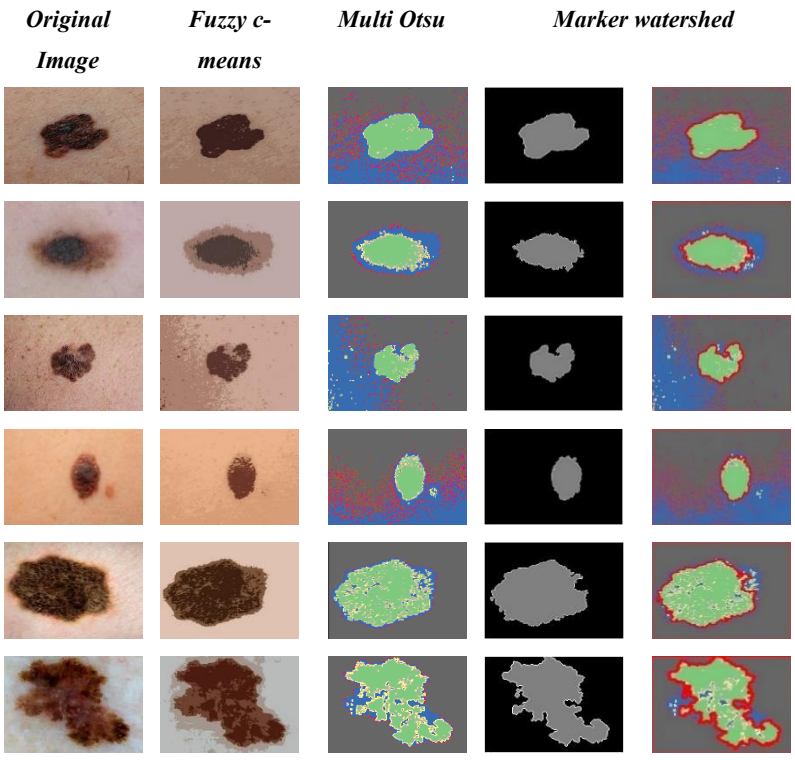


Figure 3.4 Obtained results for the second proposed method

To assess the obtained result of the submitted method, we opted for several evaluation metrics like dice coefficient, Jaccard index, Sensitivity as well as Specificity. All the

obtained results showed very satisfactory results of the given method compared to the two already existing algorithms, which are Fuzzy c means as well asthe Watershed algorithm. Which means a good detection of skin lesions. Furthermore, the six tables below show the result of the six images shown at the top, using the four different measurements. As we can see, the proposed method achieved better results compared to fuzzy c-means besides marker watershed.

Image 1	Dice coefficient	Jaccard index	Sensitivity	Specificity
The proposed method	0.89	0.91	0.82	0.87
FCM	0.6	0.43	0.9	0.1
Marker watershed	0.59	0.42	0.97	0.04

Table 3.4 Obtained results for image 1

Image 2	Dice coefficient	Jaccard index	Sensitivity	Specificity
The proposed method	0.86	0.86	0.79	0.72
FCM	0.7	0.54	0.9	0.1
Marker watershed	0.69	0.53	0.98	0.04

Table 3.5 Obtained results for image 2

Image 3	Dice coefficient	Jaccard index	Sensitivity	Specificity
The proposed method	0.76	0.95	0.88	0.99
FCM	0.14	0.07	0.99	0.1
Marker watershed	0.13	0.07	0.98	0.79

Table 3.6 Obtained results for image 3

Image 4	Dice coefficient	Jaccard index	Sensitivity	Specificity
The proposed method	0.82	0.95	0.89	0.98
FCM	0.2	0.11	0.97	0.98
Marker watershed	0.19	0.11	0.94	0.22

Table 3.7 Obtained results for image 4

Image 5	Dice coefficient	Jaccard index	Sensitivity	Specificity
The proposed method	0.8	0.93	0.77	0.95
FCM	0.3	0.17	0.8	0.97
Marker watershed	0.3	0.17	0.9	0.81

Table 3.8 Obtained results for image 5

Image 6	Dice coefficient	Jaccard index	Sensitivity	Specificity
The proposed method	0.93	0.97	0.95	0.99
FCM	0.3	0.18	0.9	0.1
Marker watershed	0.29	0.17	0.95	0.17

Table 3.9 Obtained results for image 6

3.4 License plate text recognition using deep learning, NLP, and image processing techniques

3.4.1 Introduction

A variety of devices support optical character recognition ([159], [160], [161]), or OCR ([162], [163]). With OCR, each character on a page is scanned individually. There are many various OCR solutions that have been used over the years, and the idea of turning a text into an electronic format is broad. The Optophone, which was developed way back in 1914, is one of the earliest electric OCR machines ([164], [165]). It depended on selenium's unique capacity to conduct electricity differently in the light and the dark. The Optophone identified the black links of text and lighter vacant spaces as it scanned the words on a page. A mechanism that could convert was created later in 1931. Moreover, one of the earliest methods of converting printed characters into electrical impulses rather than sounds. another way, object detection is a sort of image recognition that identifies and pinpoints the presence of an object in an image. This provides the bounding box information as well as the class. Moreover, the term "text processing" describes the study and manipulation of textual material in order to extract relevant information, alter the data, or carry out other operations on it. It is a basic problem in both data analysis and natural language processing (NLP). Text processing may be carried out using a variety of methods and tools, and it has a number of uses, such as:

- *Text cleaning*: entails eliminating extraneous characters, formatting, or background noise from text data. Eliminating punctuation, changing all text to lowercase, and dealing with unusual characters are typical text cleaning duties.
- *Tokenization*: is the process of dividing text into smaller tokens, which are often words or phrases. Tokenization, which divides text into digestible chunks for analysis, is a crucial step in many NLP activities.
- *Stopword Removal*: Stopwords are frequent words (such as "the," "and," and "in") that frequently have little significance and can be eliminated from text to decrease analytical noise.
- *Stemming and lemmatization*: these processes, known as stemming and lemmatization, break down words into their most basic or root forms. Lemmatization converts words to their dictionary or lemma form, whereas stemming attempts to eliminate prefixes and suffixes. This can aid in standardizing text and eliminating word variances.
- *Text vectorization*: For many machine learning and data analysis activities, text must be transformed into numerical vectors. For this, methods like TF-IDF (Term Frequency-Inverse Document Frequency) and one-hot encoding are employed.
- *Named Entity Recognition (NER)*: is a technique for finding and classifying named entities in text, such as names of individuals, companies, events, and places.
- *Sentiment analysis*: is the process of identifying the emotional undertone of a text and classifying it as either good, negative, or neutral. It is frequently used to examine client reviews and postings on social media.
- *Text classification*: Text categorization refers to the process of labeling or categorizing a piece of text. It is utilized in many different applications, including sentiment analysis, subject categorization, and spam detection.
- *Part-of-Speech Tagging (POS)*: POS tagging entails identifying the components of speech (such as noun, verb, and adjective) of each word in a phrase. It aids in comprehension of text's grammatical structure.
- *Text summarizing*: Text summarizing techniques are designed to reduce a larger text to a more manageable, cohesive summary while retaining all of the important details.
- *Language translation*: text is translated from one language to another using machine translation tools like Google Translate.
- *Information extraction*: is the process of separating structured information from unstructured text, such as from news articles or resumes.

- *Regular Expressions*: For pattern matching and text manipulation, regular expressions (regex) are effective tools. They are frequently employed for text search and pattern extraction.
- *Natural Language Generation (NLG)*: NLG systems produce language that is legible by humans from structured data. This material is helpful for producing reports and generating chatbot replies.

A key stage in many data analysis and NLP jobs is text processing, which frequently calls for a combination of methods and tools to get the necessary outcomes. Different text processing methods and packages, including NLTK, spaCy, and scikit-learn in Python, can be used to process and analyze text data efficiently, depending on the work at hand and the desired results.

3.4.2 Proposed method

In the first step of the suggested method and after collecting images from the internet, we used deep learning to detect the license plate in the input image. Moreover, we applied the Yolo v7 deep learning method to detect the license plate in order to get a well-segmented license plate and, therefore, a well-segmented character. Consequently, and by using the opening morphological operation and thresholding, we had well-detected characters.

Subsequently, in the next step of the proposed method we applied some required enhancements like resizing and cropping in order to keep only the essential part of the detected license plate. Afterward, we used EasyOcr as a technique to read the text from the image and save it in a text file. According to the results, we conclude that the suggested approach provides satisfactory detection and text recognition results. The Figure 3.5 below represents a detailed explanation of the proposed method:

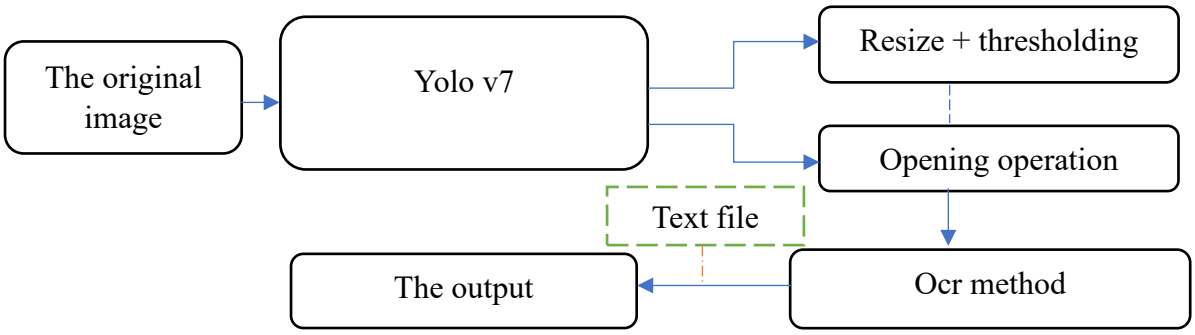


Figure 3.5 Third proposed method

3.4.3 Results

To validate the proposed method, we used the accuracy metric and tested the algorithm on several images. the Table 3.10 below shows the obtained results:

Original image	Yolo v7	Opening + Thresholding	OCR Accuracy
			95%
			96%
			98%
			97%

Table 3.10 Obtained results

3.5 Conclusion

This third chapter was dedicated to introduce the first and the second contribution which main objective was about image segmentation. In the first contribution we have presented a method for brain tumor detection and segmentation. As we know Brain tumor detection remains a very serious field in medical imaging. A brain tumor can even be a malignant type that is cancerous or a benign type that is non-cancerous. In both cases, the earlier detection helps a lot of doctors to help heal the patient. Image segmentation plays an essential role in medical imaging; for this, we used two algorithms which are Birch clustering as well as the marker-watershed algorithm. The obtained results showed that the suggested algorithm compared to the other two approaches gives good results and accurate detection of the brain tumor located in different positions and shapes. Moreover, in the second contribution, we've been discussing the proposed process for skin lesion detection. For this, we used a hybridization between the two already existing methods which are fuzzy c-means as well as marker watershed algorithm. Also, we used Multi Otsu's thresholding. The proposed method was evaluated using about 300 gathered and collected images. The obtained results show very satisfactory results compared to the two other

methods. Therefore, this chapter was dedicated to explain in detail the two first contributions we proposed where we solved some of the image segmentation problems in medical imaging. For the purposes of diagnostic analysis, therapy planning, and surgical guiding in medical imaging, image segmentation is used to recognize and isolate regions of interest in medical images. Therefore, the first proposed method entitled "A brain tumor segmentation and detection technique based on Birch and Marker watershed" where we used a hybridization between Birch and Marker watershed algorithms. Our method achieved satisfactory results. In the second contribution, we proposed a novel technique for skin lesion detection and segmentation based on image segmentation techniques. Furthermore, in this section, we presented a new technique for text extraction and recognition in the case of license plates. For this, we first applied the yolo v7 method to extract the license plate from the input image. Secondly, we applied the opening and the thresholding techniques in order to segment the extracted license plate. Afterward, we added some enhancements like resizing and cropping in order to limit the area. EasyOcr was afterward applied for text extraction, consequently, a text file will be generated containing the text or the character. The proposed method obtained satisfactory results for detection, text extraction, and recognition. This technique can be used in any use case that requires text detection and recognition as well. In future work, we’re planning to integrate deep learning in this process and generalize it to different types of license plates.

Chapter 4:

Reinforcement Learning

4.1 Introduction

Reinforcement learning is considered a sort of machine learning that acquires knowledge of solving problems using the trial-and-error technique. The process starts with the main actor that is the agent interacting with a given environment and attempting to achieve a multi-step goal within this environment. The environment is characterized by a state that the agent detects and examines. On the other hand, due to the agent's several actions, the environment's state changes according to these modifications. Eventually, and at this stage, the agent gets reward signals as it proceeds nearer to its goal. The agent uses these rewards signals to determine which actions were successful and which actions were not. The state action is then repeated and the reward is looped until the agent learns how to operate effectively within the environment using the trial-and-error concept. The agent's main objective is to learn how to always choose the right action given any state of the environment that leads it closer to its goal. In this chapter, we gathered all the methods used in the literature. Multi-armed bandits, the Markov decision process, Monte Carlo methods, dynamic programming as well as temporal-difference learning are some of the corresponding methods used to solve reinforcement learning issues. Recently, reinforcement learning [116] has exploded through the birth of many challenging projects. Robotic arm manipulation, 1v1 Dota, and vintage Atari games are one of the most important implementations using reinforcement and deep reinforcement learning. Moreover, the winnings of supervised deep learning [117] have continued to cumulate, giving the example of the ImageNet classification challenge in 2012. In addition, researchers from many areas have been involved in deep neural networks to solve an important range of new projects such as comprehending intelligent attitudes and actions within a convoluted dynamic environment. Reinforcement learning is considered as a subfield of machine learning [118] and is considered one of the most favorable and helpful orientations to attain a high level of intelligence in robotics behavior. Nearly every researcher uses supervised learning [119] in machine learning projects and applications ([120], [121]), where an input is given to the neural network model while knowing exactly what output the model should produce, subsequently, gradients will be calculated by utilizing the backpropagation method to train the network to output the results. Unfortunately, by using supervised learning

elements should be collected in the created dataset. This step of the process is not always easy for doing, regarding the amount number of elements that should be gathered in the dataset. Furthermore, the neural network will be trained to straightforwardly emulate the human player records and actions, knowing that an agent can never perform well at playing compared to a human. Thereafter, reinforcement learning ([122], [123], [124]) remains the idealistic choice to handle the issue of an agent being a better player than a human gamer, and also to learn how to play it by itself with no human interaction. Reinforcement learning and the normal framework of supervised learning are similar, we constantly have an input frame that is run through certain neural network models, and afterward, the network generates an output action. In reinforcement learning, and owing to the absence of a dataset it'll be impossible to recognize the target label, unlike supervised learning. Likewise, a Policy Network is defined as a network that modifies input frames to output actions, and among the uncomplicated ways or techniques to train the Policy Network is by employing the Policy Gradient. Moreover, the output of the network will consist of two numbers, the probability of each decision, and what to do while training. In reinforcement learning we want to authorize the agent to acquire knowledge completely by itself [125], the only given feedback, in this case, is the scoreboard. The agent in this process receives a reward of (+1) whenever it achieves to mark or score a goal, or else, in the case when the opponent was the first to score the goal, the agent is then rewarded with a penalty of (-1). The noteworthy aim of the agent is to optimize its policy to increase the amount received rewards. The policy network will be trained by first of all gathering a bunch of experiences by running a whole set of game frames through the network. Then, pick out random actions and feed them back into the system as shown in Figure 4.1.

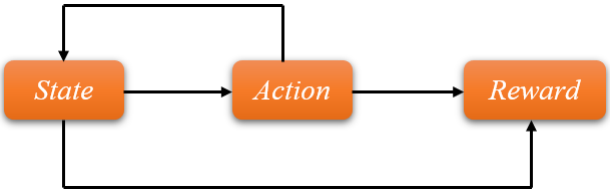


Figure 4.1 Policy optimization process

The agent is randomly going to select a whole succession of actions that lead to scoring a goal. Multiple sequences of actions will be selected randomly by the agent to score goals, subsequently, it will receive a reward. A key thing to retain here is that for each episode, and in any case, we can calculate the gradients to make the agent's action, either we need a positive or a negative reward. Ordinarily, the normal gradients are chosen to increment the likelihood of the taken actions in the future. The same gradient will also be used but this time multiplied

by (-1) whenever we obtain an unfavorable reward. In this case, and by using this minus sign, every action that is taken in a bad episode will be less likely. Consequently, during the policy network training all the actions that lead to an unfavorable or negative reward will be filtered out, thereafter, the positive ones will be more likely. However, the problem with policy gradients is that if in an episode the agent was lucky by taking good actions, it did badly in the last one. In this case, the gradient policy is going to presume that since that episode is missed, every taken action must be bad and reduce the probability of taking those actions in the future [126]. In reinforcement learning, this is called the "Credit Assignment problem". This issue is entirely related to the fact that we have what it's called a "Sparse Reward setting". Rather than gaining a reward for every single action, we only acquire a reward posterior to an entire episode, and the agent needs to find out in what part of its action sequence the reward has been created. In reinforcement learning, algorithms ([127], [128]) must have a training time before they can learn some useful behavior. In some extreme cases, the sparse reward setting fails; a famous example is the Montezuma's Revenge game ([129], [130]), where the main motivation of the agent is to navigate a bunch of stairs, leap over the skull, snatch a key and navigate to the way out to get to the following stage. The problem here is that by taking random actions, the agent is never going to see a singular reward; and that is due to the succession of actions that it needs to take to get that reward that is so complicated. That means that the policy gradient is never going to see a single positive reward. The traditional approach to solve this issue of sparse rewards has been the employment of 'Reward Shaping' which is the process of manually designing the reward function which needs to guide the policy to some required attitude. For the Montezuma's game, the agent could be rewarded every single time it manages to avoid the skull or reach the key. Posteriorly, these additional rewards will guide the policy to some desired behavior. Nevertheless, there are some significant downsides to reward shaping. Firstly, reward shaping is a custom process that needs to be redone for every novel environment, and the second problem is that reward shaping suffers from what is called alignment.

4.2 Reinforcement learning models

4.2.1 Model-based in reinforcement learning

The first biggest dichotomy is between the first type which is model-based and the second one which is model-free reinforcement learning. Initially, the model-based RL ([131], [132]) is the simplest form of reinforcement learning, in this version, we know everything about the environment and there will be no need for exploration ([133], [134]). In particular, what we

know is the complete likelihood distribution of the following state, with an already specified (s_t, a_t) we know the likelihood of reaching any possible state s_{t+1} besides the reward R_{t+1}. In this situation, the value function is given as follows:

$$V^*(s_t) = \max_{a_t}(E[r_{t+1}] + \gamma \sum_{s_{t+1}} P(s_{t+1}|s_t, a_t)V^*(s_{t+1})) \tag{4.1}$$

This equation can be solved and then calculate the values of the value function V where $E[r_{t+1}]$ represents the expected reward. The optimal policy starting from state s_t is going to be as shown in the following equation the moment that we have the value function:

$$\Pi^*(s_t) = arg\ \max_{a_t}(E[r_{t+1}|s_t, a_t] + \gamma \sum_{s_{t+1}} P(s_{t+1}|s_t, a_t)V^*(s_{t+1})) \tag{4.2}$$

The important thing here is to figure out the value function for each one of the states using an algorithm with the steps:

- Step 1: Initialize the value functions to arbitrary values
- Step 2: Repeat this part of the algorithm until $V_{(s)}$ converge

For all s ∈ S

For all a ∈ A

$$Q(s, a) \leftarrow E[r|s, a] + \gamma \sum_{s' \in S} P(s'|s, a)V(s') \tag{4.3}$$

$$V(s) \leftarrow \max_{a} Q(s, a) \tag{4.4}$$

Where E[r|s,a] refers to the predictable reward in the given state ‘s’ while choosing the action ‘a’ The second we obtain the value function; it will be able to compute the optimal policy in any state. The following algorithm allows us to estimate the optimal policy directly without going necessarily via the estimate to the value function. This algorithm is called the policy iteration, and it follows

the steps below:

Step 1: Arbitrarily initialize a policy Π

Step 2: Repeat this part of the algorithm until convergence

$\Pi \leftarrow \Pi'$

Calculate the values using Π by determining the linear equations:

$$V^{\Pi}(s) = E[r|s, \Pi(s)] + \gamma \sum_{s' \in S} P(s'|s, \Pi(s)) V^{\Pi}(s') \quad (4.5)$$

Upgrade the policy in each state:

$$\Pi'(s) = arg\ \max_{a}(E[r|s, a] + \gamma \sum_{s' \in S} P(s'|s, a) V^{\Pi}(s')) \quad (4.6)$$

4.2.2 Model-free in reinforcement learning

In reinforcement learning, model-free [135] is the opposite of model-based. It's an algorithm that does not require the use of the transition likelihood distribution, along with the reward function that are associated with the Markov decision process, which means that we don't know the P (s', r | s, a). Model-free methods [136] are trial-and-error trainees that try to optimize the value function along with the policy without trying to assess the transition and the reward function. Furthermore, and in this case, everything will be based on trying to estimate some average values over time without keeping track of what has been done before. The average reward at the state 's' in time 't' is:

$$E_t(s) = (r_1 + r_2 + ... + r_t)/t \quad (4.7)$$

$$E_{t+1}(s) = (r_1 + r_2 + ... + r_t + r_{t+1})/(t+1) \quad (4.8)$$

Subsequently:

$$E_{t+1}(s) = t/(t+1) E_t(s) + r_{t+1}/(t+1) \quad (4.9)$$

$$E_{t+1}(s) = (t+1)/(t+1) E_t(s) - E_t(s)/(t+1) + r_{t+1}/(t+1) \quad (4.10)$$

$$E_{t+1}(s) = E_t(s) + 1/(t+1)[r_{t+1} - E_t(s)] \quad (4.11)$$

Can be similar to:

$$E_{t+1}(s) = E_t(s) + \delta[r_{t+1} - E_t(s) \quad (4.12)$$

This allows us to perform an incremental estimation of the average. If δ is small it will converge to the real average very slowly, otherwise, if δ is large it will oscillate around the average. The parameter δ is called the learning rate.

4.2.3 Main two models' differences

In model-based, the environment is represented just as Markov Decision Process using a

subsequent element. Given a collection of states where a single state is designated by s, and a series of actions that are disposable in every state, where a single action is indicated by a. a transition probability function from the actual state which is s to the upcoming state denoted as s' within the action a T (s, a, s'). The reward function represents the instant reward obtained while transitioning starting with the state's achieving s' using the action a. The model-based reinforcement learning algorithms assume that we already know the model of our environment, and that model is Markov Decision Process where we know all the states and the actions that are available in the states, as well as the transition probability function including the immediate reward function. With this knowledge, there are two mutual approaches to coming up with optimal policy involving the recursive relation of the Bellman equation, the first one is called value iteration, and the second one is known as the policy iteration. Moreover, in the value iteration method, the ideal policy is gotten by picking out the action that increases the ideal state value function. Moreover, the perfect state value function will be obtained by employing an iterative algorithm, which is the reason behind the name value iteration. Furthermore, the policy function is not used during iteration, rather, the ideal state value function is updated by choosing the action that increases the estimate for the optimal state value function. The second algorithm for the model-based reinforcement learning to get the perfect policy is recognized as policy iteration. In the policy iteration method, the ideal policy is gained by discovering iteratively a preferable estimates of optimal policy function. In this approach, there're two major steps, the first one is policy evaluation as well as policy improvement. For policy evaluation case, the state value function is evaluated for the present policy, while in policy amelioration the policy is improved using the assessment of the state value function. Moreover, the key limitation of the model-based in reinforcement learning technique is that it works only when we know the model of the environment which means the Markov Decision Process corresponds to that environment when we already know all the states, actions as well as the transition probability function in addition to the reward function. However, that is not possible for all reinforcement learning games. In model free reinforcement learning case, the simulation model of the environment is given, and by reacting with it, the necessary information about this environment is gathered, which means that we can learn about the game only if we play the game. Model-free reinforcement learning techniques don't need information about the Markov Decision Process. The most common technique for model-free reinforcement learning is the Monte Carlo method. Furthermore, the main idea here is to play the game repeatedly for a large number of times and in each play of the game, we keep track of the states that we're in, the actions that we take in those states, and the rewards that we receive for every taken action in

each state. By using this information, we can assess the action value function that is the Q-value, as well as the state value function. The Monte Carlo method for reinforcement learning works by following two main steps. The first one is policy evaluation where the policy is evaluated, and the second one is policy improvement where we improve the policy. Unfortunately, Monte Carlo has many limitations, the first one is that is practicable for games that have only a few states and actions, which means in cases where the game doesn't last for a long time. Another shortcoming is that the procedure may pass a lot of time assessing suboptimal policies, which signifies that for a specified state 's' only one of the actions will correspond to optimal policy and all the other actions will correspond to suboptimal policy. In addition to that, the Monte Carlo method works only for episodic issues. Table 4.1 below shows the two different models of reinforcement learning in detail:

	RL model	
	Model-free	Model-based
Policy	-Use sampling to set up a policy as well as value function. -Model dynamics are disregarded. -The model can foretell the optimal policy without using reward function besides the transition function. -In the environment, the agent can't reach a model	-Employ the model to get the optimal policy. -It's not required to develop a policy explicitly. -Employ the reward function besides the transition function to assess the optimal policy. -The agent is capable to access an environment model.
Strength	-Has reduced assumptions that labor enormous collection of tasks -Show performance at learning intricate policies. -In some tasks, the policy may be propagated better	-Self-trained -Sample effectual -The mastered dynamic model is portable
Weakness	- Conduct mediocre decisions. -In the case of a complex model, that might be exposed to overfit.	-Requires retraining in order to optimize the controller for a certain task. -The policy is not immediately

		optimized. -Further hypothesis and not suitable for all tasks. -A model might be more intricate than a policy
Applications	-Real-world applications -Self-driving cars -Atari -AlphaStar -Robotics -Open AI	-Chess games -Applications in robotics (Motion control, surgical, kicking, walking, balancing, tracking, . . .) -Autonomous vehicles -Sanford Helicopter -Poker -AlphaGo -MuZero

Table 4.1 Reinforcement learning models

4.3 Multi-armed bandits

Reinforcement learning is evaluating the actions rather than instructing, what the correct actions are with the correct set of labels. If we imagine the situation of having four different buttons to press, and by pushing each of these buttons we get a reward. The main purpose here is to detect a way to smartly explore and exploit which buttons to push. The k-armed bandit [137] is similar to the button or slot machines with different slot levers. The purpose here is to increase the reward through a given number of times steps. Moreover, the agent sends an action to push one of the four previous buttons, and the environment sends back a reward. In the situation of the k-armed bandit, it's a contrast to other reinforcement learning problems, where the state is going to be the same environment every time, and each time the agent has four buttons to push, and at all times it's going to be the same regardless of which buttons it pushes. This is an interesting characteristic of the k-armed bandit ([138], [139]) problem. The distribution can be stationary, which means that pushing one of the four buttons would have the same expected reward for the first 50-time steps. Or it could be non-stationary, for example, the reward distribution changes over. One good example of a non-stationary reward distribution [140] could be if there's a play against an opponent in an adversarial like tic-tac-toe; if the player keeps on making the same

moves, the opponent might catch on to this and then change its strategy accordingly. Therefore, the reward distribution for making that move is non-stationary, it changes over time. The idea of the k-armed bandits is to maximize the reward by having the estimation below of each

button's reward:

$$Q_t(a) = \frac{\sum_{i=1}^{t-1} R_i I_{A_{i=a}}}{\sum_{i=0}^{t-1} I_{A_{i=a}}} \tag{4.13}$$

By a time, step noted as 't', the assessed value of action 'a' is given as $Q_t(a)$. where 'Ipredicate' indicates the arbitrary variable. 'Ipredicate' receives 1 in case the predicate is true, in other ways it gets a 0. The Greedy action selection rule would be to take the maximum button value that returned the most reward given by the equation of the number of rewards achieved over time.

$$A_t = \text{argmax} Q_t(a) \tag{4.14}$$

We've just defined the Greedy action option and selected the button with the maximum value function estimated by our agent. ε-Greedy ([135], [137]) selection is going to have a probability ε of not selecting the maximum utility button, but rather sampling another pressed button with the uniform distribution or a uniform probability. And, just as the number of steps rises with the ε-Greedy policy, each action is going to be sampled an indefinite number of times. Q_t which is the value function concerning the action will converge to q*(a)which means the optimal value function for an action. There're different situations in which the Greedy algorithm [135] is advantageous to the ε-Greedy, in cases where the reward variance is 0 the Greedy selection only needs to take the action one time to know the true reward given by that action, however, ε-Greedy algorithms [137] do much better when there are noisier rewards like having more variance in the distribution away from the mean of the reward on that action. It also performs much better with non-stationary rewards, as the reward distribution changes over time the ε-Greedy algorithm [138] will venture out. The simple bandit algorithm is the idea of either taking the maximum action with a likelihood of (1-ε) or taking a haphazard out action with a likelihood of ε and then updating the sample average for that action.

Initialization, for a=1 to k we get:

Q(a)receives 0

N(a)receives 0

Loop continually:

R receives bandit(A)

N(A)receives N(A)+1

$$Q(A) \leftarrow Q(A) + \frac{1}{N(A)}[R - Q(A)] \tag{4.15}$$

Adjusting step size is important in the case of non-stationary rewards. For example, by pushing a button, the reward given by each button is changing over time with the underlying distribution. Here we're having an δ parameter that weighting how much the expectation is updated for each pressed button.

$$Q_{n+1} = Q_n + \delta[R_n - Q_n] \tag{4.16}$$

This idea of unevenly weighting the most recently obtained rewards compared to the overall average rewards is known as the "Exponential Recently-Weighted Average".

$$\begin{aligned} Q_{n+1} &= Q_n + \delta[R_n + Q_n] \\ &= \delta R_n + (1-\delta)\,Q_n \\ &= \delta R_n + (1-\delta)[\delta R_{n-1} + (1-\delta)\,Q_{n-1}] \\ &= \delta R_n + (1-\delta)\,\delta R_{n-1} + (1-\delta)^2 Q_{n-2} \\ &= \delta R_n + (1-\delta)\,\delta R_{n-1} + (1-\delta)^2 Q_{n-2} + \ldots + (1-\delta)^{n-1}\delta R_1 + (1-\delta)^n Q_1 \end{aligned} \tag{4.17}$$

$$Q_{n+1} = (1-\delta)^n Q_1 + \sum_{i=1}^{n} \delta(1-\delta)^{n-i} R_i \tag{4.18}$$

From step to step and by varying the parameter of the step size:

$$Q_{n+1} = Q_n + \delta[R_n - Q_n] \tag{4.19}$$

Another interesting hyperparameter is the initialization of action values in this algorithm, which helps to provide prior knowledge of reward expectations. This optimistic initialization promotes exploration and it's efficient for stationary problems, doesn't matter with non-stationary rewards. Strategic initialization is often a waste of effort in practice because in this case, we're dealing with non- stationary reward distributions, so things concerning the initial behavior of the agent don't matter that much for most real problems. Upper-Confidence-Bound (UCB) is another technique for balancing exploration and exploitation of ε-Greedy [139], it's about selecting randomly another action uniformly after deciding not to take the greedy action. How could we do a better technique of this and weigh actions based on whether they are almost

greedy or if they haven't been tested. There's a lot of uncertainty about the action, this is done with the UCB equation below:

$$A_{t=arg\,\max_a}[Q_t(a) + c\sqrt{\frac{Int}{N_t(a)}}] \tag{4.20}$$

Where $Q_t(a)$ is the estimation of the action.

$c\sqrt{\frac{Int}{N_t(a)}}$ is the UCB term, and 'c' is the parameter that weights the balance between (Int) and $N_t(a)$. Int represents the ordinary logarithm of t. $N_t(a)$ presents for how many times action 'a' has been chosen before the time t. However, UCB is more complicated than the ε-Greedy to expand over bandits to more common reinforcement learning problems mainly due to non-stationary problem reward distributions and large state spaces. Another interesting way of balancing exploration and exploitation is the Gradient Bandit Algorithm. The main idea here is that it's preferred to take one action compared to the other actions, and what we have is a Soft-max distribution over our actions. Whenever the preference is large, the action will be taken more often.

Where:

$\Pi_t(a)$ at a time 't' presents the likelihood of picking up an action 'a'.

$$Pr\{A_t = a\} = \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} = \Pi_t(a) \tag{4.21}$$

$\Pi_t(a)$ presents the preference for each action 'a'. Afterward, updating using gradient ascent:

$$H_{t+1}(A) = H_t(A_t) + \delta(R_t - \bar{R}_t)(1 - \Pi_t(A_t)) \tag{4.22}$$

And:

$$H_{t+1}(a) = H_t(a) + \delta(R_t - \bar{R}_t)\Pi_t(a) \tag{4.23}$$

Where:

$a \neq A_t$

$\bar{R}_t$is the average for all rewards.

So far, we've been talking about the agent that attempts to find a unique best action when it's about a stationary task or attempts to keep track of the best action when it varies through a time when it's a non-stationary task. however, there are several situations, and the purpose is to

acquire knowledge of a policy or a mapping to the finest actions from situations. Here we're talking about the associative search task that includes trial and error to explore the finest actions and the combination of these actions together with the best status. In the literature, this is called the contextual

4.4 Markov Decision Process (MDP)

Most problems that are solved using reinforcement learning are based on formalization. This concept of formalization was given by the Markov Decision Procedure or Process. In this case, the agent ([140], [141]) that reacts to the environment is the decision-maker and it interacts with the given environment. The agent receives a description of the environment at any time step and depending on this description the agent picks out an action to pick up. Afterward, the environment moves into some novel state, moreover, and as an outcome of its foregoing action, the agent will gain a reward. To summarize, the elements of an MDP include the environment and the agent besides all the possible environment states in addition to all the actions that the agent takes, and all the rewards that the agent acquires from taking actions in the environment. The procedure of picking out a particular action from a specific state moving to a new one and obtaining a reward occurs in a sequential way repeatedly, which produces what it's called a trajectory. The trajectory presents the concatenation of state actions and rewards all over the procedure ([141], [142]). Moreover, in reinforcement learning, the principal objective of the agent is to increase the entire number of recompenses that it obtains from picking up actions in certain states of the environment, which means that during the entire process, the agent concentrates on maximizing the instant rewards as well as the accumulated rewards earned overtime. In the Markov Decision Process, a group of states 'S' as well as a collection of actions 'A' and a series of rewards 'R' has a restricted number of elements. The agent will have for every time step t=0, 1, 2, . . . a clear description of the environment state $S_t \in S$, according to this, the agent picks up an action $A_t \in A$ consequently, we will have the state action pairs noted as (A_t, A_t) At). Thereafter, time is increased and switched from t to t+1 then the environment is changed to a novel state $S_{t+1} \in S$, at this stage of the process, the agent acquires a numeral reward $R_{t+1} \in R$ from the taken action A_t from the state S_t. Generally, we can kind of think of this procedure of gaining a reward as being a random function:

$$f(S_t, A_t) = R_{t+1} \tag{4.24}$$

The trajectory is showing the consecutive procedure of picking out an action from a state, thereafter moving to a novel state and gaining a reward can be schematized as:

S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, ...

Figure 4.2 below illustrates this entire idea.

Step 1: The environment begins is in the state S_t

Step 2: The agent spots the actual state and picks up an action A_t

Step 3: The environment proceeds until the state S_{t+1} and grants to the agent a recompense R_{t+1}.

Step 4: The agent starts again for the upcoming time step t+1.

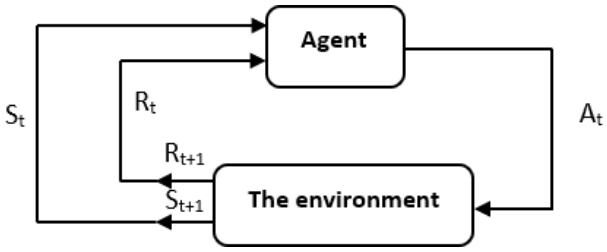


Figure 4.2 Process of the agent-environment

Since the set of states 'S' and the set of rewards 'R' are limited, the casual variables Rt and S_t that represent the reward in the state at time T, have competently described likelihood distributions. This means that all the probable values that can be allocated to Rt and St have some related likelihood. These allocations rely on the preceding state as well as the action that took place in the precedent time step t-1. Moreover, it requires a process to combine and formalize these accumulating rewards, for this, the predictable return of the rewards is used at a specified time step. Presently, we can consider the return as the total of the upcoming recompenses as shown below:

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + ...R_T \tag{4.25}$$

Here T represents the latest time step.

The idea of the predictable return is very significant because it's the agent's main purpose to increase the foreseeable return which drives the agent to make the decisions that it makes now. The agent-environment interplay splits up into subsequences noted as episodes [143], like the game pong where every novel round or cycle of the game can be counted as an episode, furthermore, the ending time step of each episode happens when a player gains a point. Every episode is finished in the final state at time T that is succeeded by resetting the environment to several starting state standards, or a random pattern from an allocation of probable beginning states. The following episode starts independently relying on how the preceding episode took

the end. There exist several types of tasks whereas the agent-environment interconnections don't split up naturally into episodes, but instead, proceed without restriction. These types of assignments are noted as continuing tasks [144], consequently, the definition of the return at each time T problematic becomes problematic for the reason that the final time step T would be equivalent to infiniteness, therefore the return may be infinite, due to this it's required to make the way of working with the return clear. On the other hand, the expected discounted return of rewards will be maximized, specifically, the agent will be picking out an action for every time step to increase the expected return. However, the main purpose of this discounted return [144] is to push the agent to be more conscious about the instant reward than the future rewards for the reason that it will be heavily discounted. The immediate rewards will have more influence on the agent's decision for choosing an action while it considers the rewards to gain in the future. The discounted return is described as follows:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \tag{4.26}$$

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{4.27}$$

Where γ represents the discount rate that's a number between 0 and 1.

Furthermore, the question to be asked here, is how can an agent be capable to select a specific action given a particular state. This is where policies take place. In addition to that, it's required to know how the chosen action in a given state might be benefic over another one in terms of the agent's return. To be conscious of this, the agent will have an advantage in choosing the right action in a specific state to increase its expected return, at this stage, the value function becomes functional. The symbol Π represents the policy function that delineates a particular status to probabilities of choosing every conceivable action from that state to indicate policy [145]. While speaking about policies officially, we that time Π(a s) represents the likelihood that the action at time t is At = 'a′ if the state at a precised time t S t = s. That signifies that at time T using the policy the likelihood of choosing an action 'a' within a given state 's' is Π(a s), knowing that for every state s S , Π is considered as a probability distribution over s A(s). Value functions [146] represent functions of states or of what it's called state- action pairs that evaluate how efficient it's for an agent to be in a specific state, or how functional for an agent to execute a specific action in a particular state. This concept is stated in terms of the predicted return. Since value functions are determined concerning the predicted return, this means that value functions are described concerning particular techniques of acting since the policy that

it's followed by the agent is influencing directly the ways the agent takes an action, thereafter, we conclude that value functions are therefore determined for policies. Therefore, value functions are defined as being the functions of states or state action pairs, and there are two sorts of value functions, the first one is a state-value function, while the second one is the state-value function. In the case of the state-value function that is using the policy, Π is designated as Π, and it informs us on how efficient any particular state is for an agent using policy Π, moreover, we'll be given the value of a state. The foreseeable return is described as the value of state denoted 's' using a policy Π at a time 't'. Subsequently, Π(s) is defined using the expression bellow:

$$\pi(s) = E[G_t|S_t = s] \tag{4.28}$$

$$\Pi^{(s)} = E[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s] \tag{4.29}$$

Likewise, q represents the action-value function following the policy Π, informing us on how efficient for an agent to pick up any specific action from any specified state using a policy Π. subsequently, it gives us the value of an action under Π. Formally, the value of the action 'a' at a specific state 's' at a time 't' is defined as:

$$q\Pi(s, a) = E[G_t|S_t = s, A_t = a] \tag{4.30}$$

$$q_{\Pi}(s, a) = E\sum_{k=0}^{\infty} R_{t+k+1}|S_t = s, A_t = a] \tag{4.31}$$

The notation q refers to the action-value function which can be mentioned as a Q-function. Moreover, Q-value represents the function's return in the case of any given state-action pair, knowing that Q indicates the quality of picking up a particular action in a specified state.

4.5 Dynamic programming

Dynamic programming ([145], [146], [147]) is a technique in computer science frequently used for things like genetic sequence alignment or spell checking. Dynamic programming refers to the idea of not using a separate array that holds the previous value function estimates at time step k. What should be done rather is to update the state values in a single sweep. This might cause a little bit of noise in the updates; because updating the states might have a higher magnitude of updating state. Another interesting extension to dynamic programming that makes it applicable to reinforcement learning problems like chess and backgammon is the idea of synchronous dynamic programming. For example, in the backgammon game ([148], [149])

there are 1020 states, and to do a complete state space sweep and update the value pushed estimates of all 1020 states, would take 1000 years at a speed of 1 million states per second. Asynchronous dynamic programming refers to the idea of updating the value estimates of a subgroup of states rather than the entire set of states at every policy iteration. Asynchronous dynamic programming can also be distributed across machines.

4.6 Monte Carlo methods

The Monte Carlo method ([150], [151]) is a way to learn without prior awareness of the state to the next state transition given our actions. Monte Carlo methods are employed to create samples of episodes and then update the value estimates of the state-action pairs according to the actual returns that are received by experiencing the world by doing this model-free approach of trial-and-error learning. Monte Carlo generally [151] describes randomized algorithms, here it's used to describe randomly sampling episodes in the environment. To use the Monte Carlo algorithm [152], we follow the following steps:

Step 1: first take in the policy Π, then we'll have the value function which is either initialized randomly or inherited from the previous iteration of the generalized policy iteration loop shown in Figure 3.

Step 2: initialize a Returns (s) array that is an empty list, which is going to aggregate all the returns every time we traverse through a given state in the state set. Returns (s) ← an emptied list, for every $s \in S$

Step 3: Loop generating episodes following the policy. We'll have these sequences of actions and then different states that we traverse given the policy that's determining the state-to-action mapping.

$\Pi = S_0, A_0, R_1, S_1, A_1, R_2, ..., S_{T-1}, A_{T-1}, R_T$

Step 4: Aggregate the achieved rewards append to the Returns array.

Step 5: Set the new values of every state as the of the returns in the Returns array. The average is going to be the sum of the Returns divided by how many times each of the Transitioning from model-based learning to model-free

learning, we'll be interested in Q functions compared to the value functions, because we don't know how the state- action pairs necessarily translate into the s' or the follow- ing state. The value of these state-action pairs should be explicitly evaluated. Putting this together, we'll have

the generalized policy iteration framework with Monte Carlo control. According to Figure 4.3, the policy is used to assess the values of the state-action pairs or Q functions using this policy, then, grouping the policy and making it greedy concerning the state-action pair table. The blackjack card game is a good motivating example for using Monte Carlo control to find the preferable state-action value estimates as well as the optimal policy. In blackjack, we have got the dealer's demonstrating card, our hand, and then whether or not we have a usable ace. The decisions from these states or whether we want to hit and receive a new card, or we want to stick and hold the fort down with our current configuration, hoping that the dealer doesn't have a greater sum than we do, or can achieve it by taking the hit action. And because the estimates of every state-action pair are coming from experience rather than by using a table, it'll be a need to have a way of balancing exploitation and exploration. One way of doing this is through exploring starts, which is initializing the environment with each state-action pair, such that the entire state-action pairs are experienced throughout the succession of episodes. The Monte Carlo with exploring starts works fairly for blackjack. We can imagine this algorithm scaling very well to most of the interesting problems in reinforcement learning. Another way of adding exploration to Monte Carlo control differently from exploring starts is to use -soft policies, this is similar to the k-armed bandits where we either take the optimal action or the probability epsilon and randomly sample another action from the conceivable actions given the current state. Probably the most interesting way to achieve exploration in Monte Carlo learning algorithms is with off-policy learning. The way this is done is by keeping a target policy that refers to the policy that will try to behave optimally, and then we'll have a behavior policy 'b' which is like the exploration policy. Here, episode samples are used from the behavior policy to explore the environment and then use this to bring the target policy up to date. At this stage, Π and 'b' must be aligned with importance sampling to weigh how realistic these trajectories are for the target policy as they're experienced by the behavior policy. This significance sampling ratio is used to weigh the returns achieved by the behavior policy when used to bring the target policy up to date. This ratio is calculated by taking the product that's relating the probabilities of choosing a particular action given a certain state from the target policy [153].

$$\begin{aligned} &Pr(A_t, S_{t+1}, A_{t+1}, \ldots, S_T | S_t, A_{t:T-1} \sim \Pi) \\ &\quad = \Pi(A_t|S_t)p(S_{t+1}|S_t, A_t)\Pi(A_{t+1}|S_{t+1})\ldots p(S_{T-1}, A_{T-1}) \\ &\quad = \prod_{k=1}^{T-1} \frac{\Pi(A_k|S_k)}{b(A_k|S_k)} \end{aligned} \tag{4.32}$$

When computing the importance sampling ratio for off- policy learning, it's computed over the trajectory of every action. There're two ways of averaging out how we're going to overall update the value estimate of the state. It's neither weighted by the length of the sequence T, nor it can be weighted by the same summation of the importance sampling ratios.

$$V(s) = \frac{\sum_{t\in\tau(s)} \rho_{t:T(t)-1} G_t}{|\tau(s)|} \tag{4.33}$$

$$V(s) = \frac{\sum_{t\in\tau(s)} \rho_{t:T(t)-1} G_t}{\sum_{t\in\tau(s)} \rho_{t:T(t)-1}} \tag{4.34}$$

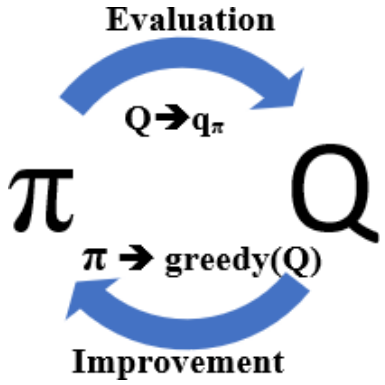


Figure 4.3 Monte Carlo prediction

4.7 Temporal-difference learning

The main idea of temporal difference ([154], [155]) is to combine learning from sampled experience with the bellman equations. The bellman equations are shown where we can assess the value of the actual state depending on the values of the states that are reached after taking certain actions and transitioning into those states. The crucial idea of temporal-difference learning is to improve the way to do model-free reinforcement learning or learning from experience. For the Monte Carlo update, the value estimate is brought of the state up to date by sampling an episode and then receiving the return Gt. Furthermore, take the error of the return minus the value under that state or the return achieved from that state as shown in the equation below:

In temporal-difference learning, the same value estimate of the return is expected from the state but with the use of bootstrapping or the technique of introducing the bellman equations.

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \quad (4.36)$$

$$G_t \cong R_{t+1} + \gamma V(S_{t+1}) \quad (4.37)$$

$$V(S_t) \leftarrow V(S_t) + \delta[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] \quad (4.38)$$

Using this idea, we introduce the full TD (0) [156] algorithm that is used to assess the value functions given a particular policy. Furthermore, this algorithm requires a step size $\delta \in (0, 1]$, and an initial assess of the value function V(s).

Step 1: TD takes the policy Π to evaluate as input

$$V(S_t) \leftarrow V(S_t) + \delta[G_t - V(S_t)] \quad (4.35)$$

Step 2: Every episode begins in some initial state 's'. At this stage, looping through taking actions given by the policy Π for a given state. Furthermore, observing the reward in the following state S', along with bringing the value of the state up to date based on the achieved return R at that step plus the discount factor γ times the value estimate of the subsequent state V(S') reached, then subtracting this by the original prediction of what it was thought it would achieve from the given state V(S).

$$V(S) \leftarrow V(S) + \delta[R + \gamma V(S') - V(S)] \quad (4.39)$$
$$S \leftarrow S'$$

Step 3: Reiterate step 2 up till the episode's ending.

For this reason, the backup diagram of the TD (0) or the one step ahead temporal-difference learning algorithm looks like what you can see in Figure 4.4. It starts in state S then it picks out an action A, and it finishes in state S'. Throughout temporal-difference learning, we'll have the temporal-difference error which is where we have the reward achieved at the time step, and then we add it to the discount factor times the value approximate of the subsequent state, then we subtract this by the original value estimate of that state.

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \quad (4.40)$$

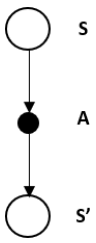


Figure 4.4 TD (0)

One more important characteristic of free reinforcement learning algorithms is that we can save the experience by using patch updating. The main idea here is to repeatedly process all the available experiences with the novel value function to make a new gain data batch. Monte Carlo converges differently than the temporal-difference learning algorithm in these batch updating. Moreover, the exploration and exploitation need to be balanced with temporal-difference learning using different algorithms like SARSA, Q-learning, Expected Sarsa [157], and Double Q-learning. Practically in these cases, we're interested in evaluating the Q-function q (s, a) rather than the value function of states V(s) and that's because it's a model-free reinforcement learning where it's not known how states transition into the next states, and this is because there's no way to access the environmental dynamics. For this reason, it'll be interesting to evaluate the pairs of state- action and use the value estimates of the state-action pairs to improve the policy. SARSA is a shortened form of the State Action Reward State Action that represents the On-Policy temporal-difference learning method, therefore, to balance exploration-exploitation it uses ε-greedy policy. The probability of ε will pick up the action that achieved the highest estimate with the likelihood of (1- ε). Then selecting another action from the actions that are available at that state. Figure 4.5 shows how the trajectories of experience have this structure of action reward: In temporal- difference learning update, a time step with the policy is taken or the Q function mapping the state to action, and then to randomly sample action from the next state achieved, and bootstrapping.

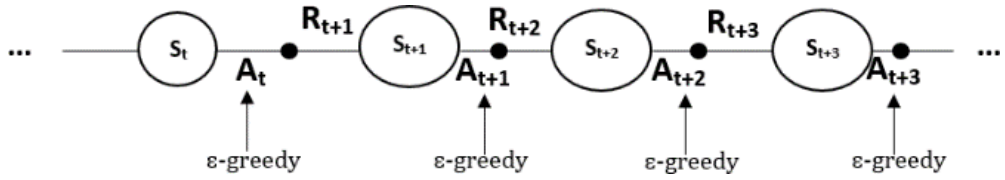


Figure 4.5 Trajectory of an experience

This happens by first starting with an initial state, then applying one action according to the -greedy policy. After this, we'll be in the next state. By observing the reward and accordingly, feeding the values until they converge. However, the updates are done only after producing every training ($[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$) and bring up to date the Q value of the preceding state as shown in the equation:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \delta[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t) \tag{4.41}$$

The full algorithm for the SARSA On-policy temporal- difference control for estimating the value functions of state- action pairs is as shown below:

Step 1: Initialize the state-action pairs with random values in the Q-table.

Step 2: In every state S that is encountered, the first thing to do is to execute the action A that comes from ε-greedy policy and get the reward R. Then processing to state S'.

Step 3: Presently, as being in state S' and planning to get to the next action A' using the ε-greedy policy at step S'.

Step 4: Referencing to the Q-table to get the value of Q (S', A')

Step 5: Update

$$Q(S,A) \leftarrow Q(S,A) + \delta[R + \gamma Q(S',A') - Q(S,A)$$
$$S \leftarrow S'; A \leftarrow A' \tag{4.42}$$

Which means considering the next state as a new initial state

Step 6: Reiterate from step 2 unto S is terminal. Q-learning is very similar to SARSA except for one difference in Step 5 in the SARSAS algorithm [158], where instead of sampling another action from our policy, we pick out the maximum action together with the highest Q-function for that state action.

$$Q(S,A) \leftarrow Q(S,A) + \delta[R + \gamma \max_a Q(S',a) - Q(S,A)] \tag{4.43}$$

SARSA outperforms Q-learning because it's learning concerning its ε-greedy policy, and that's because it's an on-policy learning algorithm. It knows that its policy is going to make these random decisions. Although the Q- learning algorithm isn't learning concerning its ε-greedy policy, rather it's learning for the optimal action it would take. Another extension to SARSA is Expected SARSA which turns SARSA from On-policy to Off-policy by rather than taking the maximum action in Q-learning, expected SARSA is going to weight the Q-value of each next state- action pair by the likelihood of taking that action given that the next, that is determined by the policy.

$$\begin{aligned} &Q(S_t,A_t) \leftarrow Q(S_t,A_t) + \delta[R_{t+1} + \gamma \mathrm{E}\pi[Q(S_{t+1},A_{t+1})\ S_{t+1}] - Q(S_t,A_t)] \\ &\leftarrow Q(S_t,A_t) + \delta[R_{t+1} + \gamma \textstyle\sum_a\ \Pi(a|S_{t+1})Q(S_{t+1},a) - Q(S_t,A_t)] \end{aligned} \tag{4.44}$$

The term $(a|S_{t+1})$ represents the probability distribution of actions given the current state, and $Q(S_{t+1})$ refers to the Q-value of each of the state-action pairs. A sum-up is then employed over all the actions, and that's what it's used for the bootstrap. In Q-learning, the temporal-difference method updates or the bootstrap value estimate of the following state has a maximization or a positive bias because it's taking the action that has the current highest value estimate, and this is solved with double Q-learning. Furthermore, the double Q-learning algorithm is going to maintain two separate tables $Q_1$and Q_2 of the state-action pairs, and then with a probability, using the Q_2 estimate of the Q_1 taking the action. Fundamentally use the Q_1 table to take the maximum action while doing the bootstrap step, and then evaluate that state-action pair with the Q_2 table or vice versa.

$$Q_1(S,A) \leftarrow Q_1(S,A) + \delta(R + \gamma Q_2(S', \underset{a}{argmax}\, Q_1(S', a)) - Q_1(S,A)) \tag{4.45}$$

Or

$$Q_2(S,A) \leftarrow Q_2(S,A) + \delta(R + \gamma Q_1(S', \underset{a}{argmax}\, Q_2(S', a)) - Q_2(S,A) \tag{4.46}$$

4.8 Conclusion

Reinforcement learning is a domain of machine learning that puts its focus on how an agent might take an action in the environment to maximize the recompenses. The main interest is to extend some of the algorithms cited into real life, which is a challenging task. The hardness comes from running a policy until termination because, in the real world, termination means crashing and destroying things by training and simulation, and modern simulators do not accurately depict the real world. Furthermore, they don't transfer to the real world when we deploy them, this means that in simulation the simulators work very well unlike when we deploy that policy in the real world. One of the most interesting applications in the real world is autonomous cars, where the car represents the agent and the environment is a 3d track. The possible actions here are to move left or right or to do nothing. Another interesting scope is robotics manipulation. Among the main advantages of reinforcement learning is that it's highly adaptive to the surrounding environment. Robots operating highly dynamic and ever-changing environments that's impossible to predict what's going to happen next, because of this, reinforcement learning provides a huge advantage to be accounting for these different scenarios and making sure that the robot's function is robust enough and able to handle all of the possible scenarios. Besides that, many other broad applications in real life are using reinforcement learning, like in healthcare, natural language processing, news recommendation, trading and finance, and many others.

General conclusion and perspectives

In this thesis we've been focusing on image segmentation, object detection, as well as text extraction and recognition. Our main purpose was to find and to propose new methods to enhance and improve image segmentation and object detection. Moreover, we focused as well on text extraction and recognition. The first task that it had to be done is to make a general and an informative state of the art of the existing methods, in order to make a comparison and find out the pros and cons of each algorithm. Therefore, we presented two contributions concerning image segmentation and tumors detection in MRI images. Additionally, a third contribution takes place in this thesis which is about text extraction and classification. Moreover, and in the fourth contribution we focused on giving a reinforcement learning review by making very concise research of all the existing methods in the field. Additionally, in the last part of this thesis we gave our current work which is based on text extraction and recognition:

- *The first proposed method*: We outlined our recommended methodology in this part, which consists of two crucial phases. Applying a filter to reduce the noise in the provided image is the first step of the algorithm. The Median filter, a non-linear filter used to remove noise from the source images, is the one we applied in this case. Doing so will help us get clearer results in the future. The generated image will then be run via the well-known Birch segmentation algorithm, which is known for its performance in terms of execution time and results received after only one scan of the dataset. Birch is capable of processing enormous databases. Furthermore, Birch's capacity to handle non-uniform gray sections is the other primary factor that led to its selection above the other image segmentation techniques currently in use. The second part after this one will go into further depth on the Birch algorithm. Image processing, the first step in the suggested strategy, consists of two key activities. The first step is to clean up the artifacts in the input image, and the second is to use a segmentation technique to find the tumor. Additionally, the Watershed approach is used in the algorithm's second step to create the first output image. In this stage of the program, we proceed to the last step of drawing the outlines around the discovered tumor after preprocessing the image and using the Birch segmentation technique. Moreover, the experiment's utilization of images from an MRI comes from Kaggle datasets. displaying several tumor forms that are situated in various places. The dataset described earlier had 155 images that we utilized to train our system on. Preprocessing the MRI images is the first stage in the suggested method. Thus, we employed a number of assessment measures, including Sensitivity, Dice

coefficient, Specificity, Jaccard Index, and Shannon Entropy, to verify and confirm the recommended approach. In comparison to the other two segmentation approaches, the recommended method therefore produces impressive results. Additionally, the suggested algorithm's output images demonstrate a strong convergence in tumor detection, meaning the tumor is clearly defined and surrounding.

- *The second proposed method:* This section will cover the subject of our suggested method for segmenting and detecting skin cancer. Gathering images from the internet and databases like the Kaggle dataset was the initial challenge for this. The median filter is used in the second step, which is the preprocessing stage. This non-linear filter is employed to lessen the noise in the original image. In the following phase, we used fuzzy c-means clustering, a soft clustering method that allows pixels to simultaneously be in one or more groups. Afterward, the final image will then go to the following stage of the suggested procedure. Multi Otsu's approach, which is provided in the next stage, examines the histogram of the image, taking into account the pixel values and the property to segment the desired item. The segmentation will then be done using the marker watershed algorithm. We used a number of assessment criteria, including dice coefficient, Jaccard index, sensitivity, and specificity, to evaluate the results of the proposed technique. Subsequently, all of the acquired findings demonstrated extremely good outcomes for the suggested technique when compared to the two pre-existing algorithms, the Watershed algorithm and fuzzy c-means. This denotes accurate skin lesion identification.
- *The third work:* After gathering images from the internet, we employed deep learning to identify the license plate in the input image as the first phase of the proposed method. In order to obtain a well-segmented license plate and, consequently, a well-segmented character, we also performed the opening and the thresholding techniques on the identified license plate. Consequently, we have well-detected characters thanks to the use of the opening and thresholding techniques. Then, in the following stage of the suggested procedure, we added some necessary improvements like scaling and cropping to retain only the crucial portions of the identified license plate. After that, we read the text from the image using EasyOcr and saved it in a text file. Subsequently, the nation of the detected license plate is then shown. We draw the conclusion that the outcomes of the offered technique for detection and text recognition are good. OCR Accuracy is the process of comparing the OCR output with the original text that served as the basis

for the comparison (the "ground truth"). Finally, we tested the algorithm on numerous images and used the accuracy score to validate the suggested approach.

- *The fourth contribution:* The fourth contribution, which discusses a reinforcement learning review, was designated for this part. Additionally, we gathered every approach that was employed in the literature. To address problems with reinforcement learning, comparable techniques such as multi-armed bandits, the Markov decision process, Monte Carlo methods, dynamic programming, and temporal-difference learning are sometimes utilized.

By proposing the previous contributions, we had several ideas about developing other approaches in the future such as:

- Including deep learning in the medical imaging field, and to propose solutions for the 3D MRI.
- Using reinforcement learning in image processing solutions.
- To use other NLP methods for text processing

Publications

International journals:

1. Moussaoui, H., El Akkad, N. & Benslimane, M. A Brain Tumor Segmentation and Detection Technique Based on Birch and Marker Watershed. *SN COMPUT. SCI.* **4**, 339 (2023). https://doi.org/10.1007/s42979-023-01802-4
2. H. Moussaoui, Nabil El Akkad, and Mohamed Benslimane, "A Hybrid Skin Lesions Segmentation Approach Based on Image Processing Methods", *Stat., optim. inf. comput.*, vol. 11, no. 1, pp. 95-105, Jan. 2023.
3. MOUSSAOUI, Hanae, BENSLIMANE, Mohamed, *et al.* Reinforcement Learning: A review. *International Journal of Computing and Digital Systems*, 2023, vol. 13, no 1, p. 1-1.
4. "License plate text recognition using deep learning, NLP, and image processing techniques". Journal: "Statistics, Optimization & Information Computing", 2023
5. Submitted article: "New method of text detection and recognition based on deep learning, image processing algorithms, and OCR-Use case license plates", Scientific Reports journal.

International conferences:

1. MOUSSAOUI, Hanae, EL AKKAD, Nabil, et BENSLIMANE, Mohamed. A Review of Video Summarization. Digital Technologies and Applications: Proceedings of ICDTA'23, Fez, Morocco, Volume 1, 2023, p. 516-525.
2. MOUSSAOUI, Hanae, BENSLIMANE, Mohamed, et EL AKKAD, Nabil. Image segmentation approach based on hybridization between K-means and mask R-CNN. In : WITS 2020: Proceedings of the 6th International Conference on Wireless Technologies, Embedded, and Intelligent Systems. Springer Singapore, 2022. p. 821-830.
3. MOUSSAOUI, Hanae, EL AKKAD, Nabil, et BENSLIMANE, Mohamed. Moroccan Carpets Classification Based on SVM Classifier and ORB Features. In : Digital Technologies and Applications: Proceedings of ICDTA'22, Fez, Morocco, Volume 2. Cham : Springer International Publishing, 2022. p. 446-455.
4. MOUSSAOUI, Hanae, BENSLIMANE, Mohamed, et EL AKKAD, Nabil. A novel brain tumor detection approach based on fuzzy c-means and marker watershed

algorithm. In : International Conference on Digital Technologies and Applications. Cham : Springer International Publishing, 2021. p. 871-879.

5. MOUSSAOUI, Hanae, EL AKKAD, Nabil, et BENSLIMANE, Mohamed. Text extraction and recognition method for license plates. In : E3S Web of Conferences. EDP Sciences, 2023. p. 00069.

National Events:

1. Summer School At « Al Akhawayn University » in collaboration with MoroccoAI. « Ru'ya: Innovative Accident Severity & Recommendation App for Moroccan Road Safety ».
2. Journée de Recherche Inter-Laboratoires 2020 (JRIL'20), « Image segmentation techniques ».
3. Journée de Recherche Inter-Laboratoires 2023 (JRIL'23). « Date fruit diseases detection and segmentation ».

References

[1] GUO, Meng-Hao, XU, Tian-Xing, LIU, Jiang-Jiang, *et al.* Attention mechanisms in computer vision: A survey. *Computational visual media*, 2022, vol. 8, no 3, p. 331-368.

[2] SZELISKI, Richard. Computer vision: algorithms and applications. Springer Nature, 2022.

[3] HOPKINS, Emily, et al. Machine learning tools, algorithms, and techniques. Journal of Self-Governance and Management Economics, 2022, vol. 10, no 1, p. 43-55.

[4] WANG, Risheng, LEI, Tao, CUI, Ruixia, et al. Medical image segmentation using deep learning: A survey. IET Image Processing, 2022, vol. 16, no 5, p. 1243-1267.

[5] KHERADMANDI, Narges et MEHRANFAR, Vida. A critical review and comparative study on image segmentation-based techniques for pavement crack detection. Construction and Building Materials, 2022, vol. 321, p. 126162.

[6] DHANACHANDRA, Nameirakpam et CHANU, Yambem Jina. A survey on image segmentation methods using clustering techniques. European Journal of Engineering and Technology Research, 2017, vol. 2, no 1, p. 15-20.

[7] RAMESH, K. K. D., KUMAR, G. Kiran, SWAPNA, K., et al. A review of medical image segmentation algorithms. EAI Endorsed Transactions on Pervasive Health and Technology, 2021, vol. 7, no 27, p. e6-e6.

[8] XU, Shuyuan, WANG, Jun, SHOU, Wenchi, et al. Computer vision techniques in construction: a critical review. Archives of Computational Methods in Engineering, 2021, vol. 28, p. 3383-3397.

[9] KAUL, Vivek, ENSLIN, Sarah, et GROSS, Seth A. History of artificial intelligence in medicine. Gastrointestinal endoscopy, 2020, vol. 92, no 4, p. 807-812.

[10] BHATT, Dulari, PATEL, Chirag, TALSANIA, Hardik, et al. CNN variants for computer vision: History, architecture, application, challenges and future scope. Electronics, 2021, vol. 10, no 20, p. 2470.

[11] KAUL, Vivek, ENSLIN, Sarah, et GROSS, Seth A. History of artificial intelligence in medicine. Gastrointestinal endoscopy, 2020, vol. 92, no 4, p. 807-812.

[12] MAHADEVKAR, Supriya V., KHEMANI, Bharti, PATIL, Shruti, et al. A review on machine learning styles in computer vision-techniques and future directions. IEEE Access, 2022.

[13] GUPTA, Abhishek. Current research opportunities for image processing and computer vision. Computer Science, 2019, vol. 20, p. 387-410.

[14] HOESER, Thorsten et KUENZER, Claudia. Object detection and image segmentation with deep learning on earth observation data: A review-part i: Evolution and recent trends. Remote Sensing, 2020, vol. 12, no 10, p. 1667.

[15] HOUSSEIN, Essam H., EMAM, Marwa M., ALI, Abdelmgeid A., et al. Deep and machine learning techniques for medical imaging-based breast cancer: A comprehensive review. Expert Systems with Applications, 2021, vol. 167, p. 114161.

[16] MORID, Mohammad Amin, BORJALI, Alireza, et DEL FIOL, Guilherme. A scoping review of transfer learning research on medical image analysis using ImageNet. Computers in biology and medicine, 2021, vol. 128, p. 104115.

[17] ZHANG, Caiming et LU, Yang. Study on artificial intelligence: The state of the art and future prospects. Journal of Industrial Information Integration, 2021, vol. 23, p. 100224.

[18] ISMAIL FAWAZ, Hassan, LUCAS, Benjamin, FORESTIER, Germain, et al. Inceptiontime: Finding alexnet for time series classification. Data Mining and Knowledge Discovery, 2020, vol. 34, no 6, p. 1936-1962.

[19] CHATTERJEE, Saptarshi, DEY, Debangshu, et MUNSHI, Sugata. Integration of morphological preprocessing and fractal based feature extraction with recursive feature elimination for skin lesion types classification. Computer methods and programs in biomedicine, 2019, vol. 178, p. 201-218.

[20] Moussaoui, H., El Akkad, N. & Benslimane, M. A Brain Tumor Segmentation and Detection Technique Based on Birch and Marker Watershed. SN COMPUT. SCI. 4, 339 (2023). https://doi.org/10.1007/s42979-023-01802-4

[21] H. Moussaoui, Nabil El Akkad, and Mohamed Benslimane, “A Hybrid Skin Lesions Segmentation Approach Based on Image Processing Methods”, Stat., optim. inf. comput., vol. 11, no. 1, pp. 95-105, Jan. 2023.

[22] WANG, Zhaobin, WANG, E., et ZHU, Ying. Image segmentation evaluation: a survey of methods. Artificial Intelligence Review, 2020, vol. 53, p. 5637-5674.

[23] DU, Getao, CAO, Xu, LIANG, Jimin, et al. Medical image segmentation based on u-net: A review. Journal of Imaging Science and Technology, 2020.

[24] MINAEE, Shervin, BOYKOV, Yuri, PORIKLI, Fatih, et al. Image segmentation using deep learning: A survey. IEEE transactions on pattern analysis and machine intelligence, 2021, vol. 44, no 7, p. 3523-3542.

[25] WANG, Risheng, LEI, Tao, CUI, Ruixia, et al. Medical image segmentation using deep learning: A survey. IET Image Processing, 2022, vol. 16, no 5, p. 1243-1267.

[26] HOU, Shou-Ming, JIA, Chao-Lan, WANGA, Ya-Bing, et al. A review of the edge detection technology. Sparklinglight Transactions on Artificial Intelligence and Quantum Computing (STAIQC), 2021, vol. 1, no 2, p. 26-37.

[27] BHATTI, Uzair Aslam, MING-QUAN, Zhou, QING-SONG, Huo, et al. Advanced color edge detection using clifford algebra in satellite images. IEEE Imagenics Journal, 2021, vol. 13, no 2, p. 1-20.

[28] ABDEL-GAWAD, Ahmed H., SAID, Lobna A., et RADWAN, Ahmed G. Optimized edge detection technique for brain tumor detection in MR images. IEEE Access, 2020, vol. 8, p. 136243-136259.

[29] WU, Fangsheng, ZHU, Changan, XU, Jinxiu, et al. Research on image text recognition based on canny edge detection algorithm and k-means algorithm. International Journal of System Assurance Engineering and Management, 2021, p. 1-9.

[30] LI, Xiaofeng, JIAO, Hongshuang, et WANG, Yanwei. Edge detection algorithm of cancer image based on deep learning. Bioengineered, 2020, vol. 11, no 1, p. 693-707.

[31] LIU, Xiangbin, SONG, Liping, LIU, Shuai, et al. A review of deep-learning-based medical image segmentation methods. Sustainability, 2021, vol. 13, no 3, p. 1224.

[32] UZIEL, Roy, RONEN, Meitar, et FREIFELD, Oren. Bayesian adaptive superpixel segmentation. In : Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019. p. 8470-8479.

[33] GAMARRA, Margarita, ZUREK, Eduardo, ESCALANTE, Hugo Jair, et al. Split and merge watershed: A two-step method for cell segmentation in fluorescence microscopy images. Biomedical signal processing and control, 2019, vol. 53, p. 101575.

[34] RAMESH, K. K. D., KUMAR, G. Kiran, SWAPNA, K., et al. A review of medical image segmentation algorithms. EAI Endorsed Transactions on Pervasive Health and Technology, 2021, vol. 7, no 27, p. e6-e6.

[35] ABDULATEEF, Salwa Khalid et SALMAN, Mohanad Dawood. A Comprehensive Review of Image Segmentation Techniques. Iraqi Journal for Electrical & Electronic Engineering, 2021, vol. 17, no 2.

[36] KOLLEM, Sreedhar, REDDY, Katta Rama Linga, et RAO, Duggirala Srinivasa. A review of image denoising and segmentation methods based on medical images. International Journal of Machine Learning and Computing, 2019, vol. 9, no 3, p. 288-295.

[37] KOTARIDIS, Ioannis et LAZARIDOU, Maria. Remote sensing image segmentation advances: A meta-analysis. ISPRS Journal of Imagegrammetry and Remote Sensing, 2021, vol. 173, p. 309-322.

[38] BALAJI, V. R., SUGANTHI, S. T., RAJADEVI, R., et al. Skin disease detection and segmentation using dynamic graph cut algorithm and classification through Naive Bayes classifier. Measurement, 2020, vol. 163, p. 107922.

[39] KIM, Wonjik, KANEZAKI, Asako, et TANAKA, Masayuki. Unsupervised learning of image segmentation based on differentiable feature clustering. IEEE Transactions on Image Processing, 2020, vol. 29, p. 8055-8068.

[40] MOUSSAOUI, Hanae, BENSLIMANE, Mohamed, et EL AKKAD, Nabil. Image segmentation approach based on hybridization between K-means and mask R-CNN. In : WITS 2020: Proceedings of the 6th International Conference on Wireless Technologies, Embedded, and Intelligent Systems. Springer Singapore, 2022. p. 821-830.

[41] MOUSSAOUI, Hanae, EL AKKAD, Nabil, et BENSLIMANE, Mohamed. Moroccan Carpets Classification Based on SVM Classifier and ORB Features. In : Digital Technologies and Applications: Proceedings of ICDTA'22, Fez, Morocco, Volume 2. Cham : Springer International Publishing, 2022. p. 446-455.

[42] MOUSSAOUI, Hanae, BENSLIMANE, Mohamed, et EL AKKAD, Nabil. A novel brain tumor detection approach based on fuzzy c-means and marker watershed algorithm. In : International Conference on Digital Technologies and Applications. Cham : Springer International Publishing, 2021. p. 871-879.

[43] PARE, S., KUMAR, A., SINGH, Girish Kumar, et al. Image segmentation using multilevel thresholding: a research review. Iranian Journal of Science and Technology, Transactions of Electrical Engineering, 2020, vol. 44, p. 1-29.

[44] HOUSSEIN, Essam H., HELMY, Bahaa El-din, OLIVA, Diego, et al. A novel black widow optimization algorithm for multilevel thresholding image segmentation. Expert Systems with Applications, 2021, vol. 167, p. 114159.

[45] DHAL, Krishna Gopal, DAS, Arunita, RAY, Swarnajit, et al. Nature-inspired optimization algorithms and their application in multi-thresholding image segmentation. Archives of Computational Methods in Engineering, 2020, vol. 27, p. 855-888.

[46] ROUDITCHENKO, Andrew, ZHAO, Hang, GAN, Chuang, et al. Self-supervised audio-visual co-segmentation. In : ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2019. p. 2357-2361.

[47] LIU, Weide, LIN, Guosheng, ZHANG, Tianyi, et al. Guided co-segmentation network for fast video object segmentation. IEEE Transactions on Circuits and Systems for Video Technology, 2020, vol. 31, no 4, p. 1607-1617.

[48] LU, Zhoumin, XU, Haiping, et LIU, Genggeng. A survey of object co-segmentation. IEEE Access, 2019, vol. 7, p. 62875-62893.

[49] HAO, Shijie, ZHOU, Yuan, et GUO, Yanrong. A brief survey on semantic segmentation with deep learning. Neurocomputing, 2020, vol. 406, p. 302-321.

[50] LATEEF, Fahad et RUICHEK, Yassine. Survey on semantic segmentation using deep learning techniques. Neurocomputing, 2019, vol. 338, p. 321-348.

[51] LIU, Zhenqing, CAO, Yiwen, WANG, Yize, et al. Computer vision-based concrete crack detection using U-net fully convolutional networks. Automation in Construction, 2019, vol. 104, p. 129-139.

[52] WANG, Jianfeng, SONG, Lin, LI, Zeming, et al. End-to-end object detection with fully convolutional network. In : Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021. p. 15849-15858.

[53] REN, Yupeng, HUANG, Jisheng, HONG, Zhiyou, et al. Image-based concrete crack detection in tunnels using deep fully convolutional networks. Construction and Building Materials, 2020, vol. 234, p. 117367.

[54] SIDDIQUE, Nahian, PAHEDING, Sidike, ELKIN, Colin P., et al. U-net and its variants for medical image segmentation: A review of theory and applications. Ieee Access, 2021, vol. 9, p. 82031-82057.

[55] DU, Getao, CAO, Xu, LIANG, Jimin, et al. Medical image segmentation based on u-net: A review. Journal of Imaging Science and Technology, 2020.

[56] WANG, Hongyi, XIE, Shiao, LIN, Lanfen, et al. Mixed transformer u-net for medical image segmentation. In : ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2022. p. 2390-2394.

[57] LIU, Chenxi, CHEN, Liang-Chieh, SCHROFF, Florian, et al. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In : Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019. p. 82-92.

[58] ZHANG, Yifei, SIDIBÉ, Désiré, MOREL, Olivier, et al. Deep multimodal fusion for semantic image segmentation: A survey. Image and Vision Computing, 2021, vol. 105, p. 104042.

[59] LIU, Chenxi, CHEN, Liang-Chieh, SCHROFF, Florian, et al. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In : Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019. p. 82-92.

[60] BERTASIUS, Gedas, TORRESANI, Lorenzo, YU, Stella X., et al. Convolutional random walk networks for semantic image segmentation. In : Proceedings of the IEEE conference on computer vision and pattern recognition. 2017. p. 858-866.

[61] HAN, Shuangquan et XI, Zhihong. Dynamic scene semantics SLAM based on semantic segmentation. IEEE Access, 2020, vol. 8, p. 43563-43570.

[62] ZHU, Xiliang, CHENG, Zhaoyun, WANG, Sheng, et al. Coronary angiography image segmentation based on PSPNet. Computer Methods and Programs in Biomedicine, 2021, vol. 200, p. 105897.

[63] LI, Hanchao, XIONG, Pengfei, AN, Jie, et al. Pyramid attention network for semantic segmentation. arXiv preprint arXiv:1805.10180, 2018.

[64] LI, Huiyi. Image semantic segmentation method based on GAN network and ENet model. The Journal of Engineering, 2021, vol. 2021, no 10, p. 594-604.

[65] JHA, Debesh, RIEGLER, Michael A., JOHANSEN, Dag, et al. Doubleu-net: A deep convolutional neural network for medical image segmentation. In : 2020 IEEE 33rd International symposium on computer-based medical systems (CBMS). IEEE, 2020. p. 558-564.

[66] SYED, Arsal et MORRIS, Brendan Tran. SSeg-LSTM: semantic scene segmentation for trajectory prediction. In : 2019 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2019. p. 2504-2509.

[67] SYED, Arsal et MORRIS, Brendan Tran. SSeg-LSTM: semantic scene segmentation for trajectory prediction. In : 2019 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2019. p. 2504-2509.

[68] JADON, Shruti. A survey of loss functions for semantic segmentation. In : 2020 IEEE conference on computational intelligence in bioinformatics and computational biology (CIBCB). IEEE, 2020. p. 1-7.

[69] XIONG, Wei, JIA, Xiuhong, YANG, Dichun, et al. DP-LinkNet: A convolutional network for historical document image binarization. KSII Transactions on Internet and Information Systems (TIIS), 2021, vol. 15, no 5, p. 1778-1797.

[70] AKYEL, Cihan et ARICI, Nursal. LinkNet-B7: noise removal and lesion segmentation in images of skin cancer. Mathematics, 2022, vol. 10, no 5, p. 736.

[71] ZHU, Qingtian, ZHENG, Yumin, JIANG, Yulai, et al. Efficient multi-class semantic segmentation of high resolution aerial imagery with dilated LinkNet. In : IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium. IEEE, 2019. p. 1065-1068.

[72] SEONG, Seonkyeong et CHOI, Jaewan. Semantic segmentation of urban buildings using a high-resolution network (HRNet) with channel and spatial attention gates. Remote Sensing, 2021, vol. 13, no 16, p. 3087.

[73] ZHANG, Jing, LIN, Shaofu, DING, Lei, et al. Multi-scale context aggregation for semantic segmentation of remote sensing images. Remote Sensing, 2020, vol. 12, no 4, p. 701.

[74] SEONG, Seonkyeong et CHOI, Jaewan. Semantic segmentation of urban buildings using a high-resolution network (HRNet) with channel and spatial attention gates. Remote Sensing, 2021, vol. 13, no 16, p. 3087.

[75] GUO, Meng-Hao, XU, Tian-Xing, LIU, Jiang-Jiang, et al. Attention mechanisms in computer vision: A survey. Computational visual media, 2022, vol. 8, no 3, p. 331-368.

[76] ZHU, Xizhou, CHENG, Dazhi, ZHANG, Zheng, et al. An empirical study of spatial attention mechanisms in deep networks. In : Proceedings of the IEEE/CVF international conference on computer vision. 2019. p. 6688-6697.

[77] ALSHEMALI, Basemah et KALITA, Jugal. Improving the reliability of deep neural networks in NLP: A review. Knowledge-Based Systems, 2020, vol. 191, p. 105210.

[78] FENG, Steven Y., GANGAL, Varun, WEI, Jason, et al. A survey of data augmentation approaches for NLP. arXiv preprint arXiv:2105.03075, 2021.

[79] KLEIN, Tassilo et NABI, Moin. Learning to answer by learning to ask: Getting the best of gpt-2 and bert worlds. arXiv preprint arXiv:1911.02365, 2019.

[80] WANG, Zhaobin, WANG, E., et ZHU, Ying. Image segmentation evaluation: a survey of methods. Artificial Intelligence Review, 2020, vol. 53, p. 5637-5674.

[81] EELBODE, Tom, BERTELS, Jeroen, BERMAN, Maxim, et al. Optimization for medical image segmentation: theory and practice when evaluating with dice score or jaccard index. IEEE Transactions on Medical Imaging, 2020, vol. 39, no 11, p. 3679-3690.

[82] JHA, Debesh, RIEGLER, Michael A., JOHANSEN, Dag, et al. Doubleu-net: A deep convolutional neural network for medical image segmentation. In : 2020 IEEE 33rd International symposium on computer-based medical systems (CBMS). IEEE, 2020. p. 558-564.

[83] WANG, Risheng, LEI, Tao, CUI, Ruixia, et al. Medical image segmentation using deep learning: A survey. IET Image Processing, 2022, vol. 16, no 5, p. 1243-1267.

[84] ZOU, Zhengxia, CHEN, Keyan, SHI, Zhenwei, et al. Object detection in 20 years: A survey. Proceedings of the IEEE, 2023.

[85] ZAIDI, Syed Sahil Abbas, ANSARI, Mohammad Samar, ASLAM, Asra, et al. A survey of modern deep learning based object detection models. Digital Signal Processing, 2022, vol. 126, p. 103514.

[86] KIM, Jeong-ah, SUNG, Ju-Yeong, et PARK, Se-ho. Comparison of Faster-RCNN, YOLO, and SSD for real-time vehicle type recognition. In : 2020 IEEE international conference on consumer electronics-Asia (ICCE-Asia). IEEE, 2020. p. 1-4.

[87] HE, Zhenwei et ZHANG, Lei. Multi-adversarial faster-rcnn for unrestricted object detection. In : Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019. p. 6668-6677.

[88] XIE, Xingxing, CHENG, Gong, WANG, Jiabao, et al. Oriented R-CNN for object detection. In : Proceedings of the IEEE/CVF international conference on computer vision. 2021. p. 3520-3529.

[89] ZAIDI, Syed Sahil Abbas, ANSARI, Mohammad Samar, ASLAM, Asra, et al. A survey of modern deep learning based object detection models. Digital Signal Processing, 2022, vol. 126, p. 103514.

[90] SHEYKHMOUSA, Mohammadreza, MAHDIANPARI, Masoud, GHANBARI, Hamid, et al. Support vector machine versus random forest for remote sensing image classification: A meta-analysis and systematic review. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 2020, vol. 13, p. 6308-6325.

[91] HONG, Danfeng, HAN, Zhu, YAO, Jing, et al. SpectralFormer: Rethinking hyperspectral image classification with transformers. IEEE Transactions on Geoscience and Remote Sensing, 2021, vol. 60, p. 1-15.

[92] AZIZI, Shekoofeh, MUSTAFA, Basil, RYAN, Fiona, et al. Big self-supervised models advance medical image classification. In : Proceedings of the IEEE/CVF international conference on computer vision. 2021. p. 3478-3488.

[93] YADAV, Samir S. et JADHAV, Shivajirao M. Deep convolutional neural network based medical image classification for disease diagnosis. Journal of Big data, 2019, vol. 6, no 1, p. 1-18.

[94] BOMMERT, Andrea, SUN, Xudong, BISCHL, Bernd, et al. Benchmark for filter methods for feature selection in high-dimensional classification data. Computational Statistics & Data Analysis, 2020, vol. 143, p. 106839.

[95] ELHOSENY, Mohamed et SHANKAR, K. Optimal bilateral filter and convolutional neural network based denoising method of medical image measurements. Measurement, 2019, vol. 143, p. 125-135.

[96] QUAID, Majid Ali Khan et JALAL, Ahmad. Wearable sensors based human behavioral pattern recognition using statistical features and reweighted genetic algorithm. Multimedia Tools and Applications, 2020, vol. 79, p. 6061-6083.

[97] CHEN, Baoshi, ZHANG, Lingling, CHEN, Hongyan, et al. A novel extended Kalman filter with support vector machine based method for the automatic diagnosis and segmentation of brain tumors. Computer Methods and Programs in Biomedicine, 2021, vol. 200, p. 105797.

[98] TANG, Yiming, REN, Fuji, et PEDRYCZ, Witold. Fuzzy C-means clustering through SSIM and patch for image segmentation. Applied Soft Computing, 2020, vol. 87, p. 105928.

[99] CAI, Weiwei, ZHAI, Bo, LIU, Yun, et al. Quadratic polynomial guided fuzzy C-means and dual attention mechanism for medical image segmentation. Displays, 2021, vol. 70, p. 102106.

[100] KUMAR, Manoj, ALSHEHRI, Mohammed, ALGHAMDI, Rayed, et al. A de-ann inspired skin cancer detection approach using fuzzy c-means clustering. Mobile Networks and Applications, 2020, vol. 25, p. 1319-1329.

[101] AL-RAHLAWEE, Anfal Thaer Hussein et RAHEBI, Javad. Multilevel thresholding of images with improved Otsu thresholding by black widow optimization algorithm. Multimedia Tools and Applications, 2021, vol. 80, no 18, p. 28217-28243.

[102] CHEN, Chengcheng, WANG, Xianchang, HEIDARI, Ali Asghar, et al. Multi-threshold image segmentation of maize diseases based on elite comprehensive particle swarm optimization and otsu. Frontiers in plant science, 2021, vol. 12, p. 789911.

[103] MA, Guoyuan et YUE, Xiaofeng. An improved whale optimization algorithm based on multilevel threshold image segmentation using the Otsu method. Engineering Applications of Artificial Intelligence, 2022, vol. 113, p. 104960.

[104] SRIKANTH, R. et BIKSHALU, K. Multilevel thresholding image segmentation based on energy curve with harmony Search Algorithm. Ain Shams Engineering Journal, 2021, vol. 12, no 1, p. 1-20.

[105] MITTAL, Himanshu, PANDEY, Avinash Chandra, SARASWAT, Mukesh, et al. A comprehensive survey of image segmentation: clustering methods, performance parameters, and benchmark datasets. Multimedia Tools and Applications, 2021, p. 1-26.

[106] TOMAR, Rohan et SHARMA, Abhilasha. K-Means and BIRCH: A Comparative Analysis Study. In : Inventive Communication and Computational Technologies: Proceedings of ICICCT 2022. Singapore : Springer Nature Singapore, 2022. p. 281-294.

[107] YIN, Shoulin, LI, Hang, LIU, Desheng, et al. Active contour modal based on density-oriented BIRCH clustering method for medical image segmentation. Multimedia Tools and Applications, 2020, vol. 79, p. 31049-31068.

[108] ZHANG, Hu, TANG, Zhaohui, XIE, Yongfang, et al. A watershed segmentation algorithm based on an optimal marker for bubble size measurement. Measurement, 2019, vol. 138, p. 182-193.

[109] LIN, Yuefei, DIAO, Yong, DU, Yongzhao, et al. Automatic cell counting for phase-contrast microscopic images based on a combination of Otsu and watershed segmentation method. Microscopy Research and Technique, 2022, vol. 85, no 1, p. 169-180.

[110] SAMADI, Farnaam, AKBARIZADEH, Gholamreza, et KAABI, Hooman. Change detection in SAR images using deep belief network: a new training approach based on morphological images. IET Image Processing, 2019, vol. 13, no 12, p. 2255-2264.

[111] VINCENT, Luc. Current topics in applied morphological image analysis. In : Stochastic Geometry. Routledge, 2019. p. 199-283.

[112] SERRA, Jean. Mathematical morphology. In : Encyclopedia of Mathematical Geosciences. Cham : Springer International Publishing, 2022. p. 1-16.

[113] MOERLAND, Thomas M., BROEKENS, Joost, PLAAT, Aske, et al. Model-based reinforcement learning: A survey. Foundations and Trends® in Machine Learning, 2023, vol. 16, no 1, p. 1-118.

[114] ZHU, Zhuangdi, LIN, Kaixiang, JAIN, Anil K., et al. Transfer learning in deep reinforcement learning: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2023.

[115] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, “Safe learning in robotics: From learning- based control to safe reinforcement learning,” Annual Review of Control, Robotics, and Autonomous Systems, vol. 5, pp. 411–444, 2022.

[116] D. Yarats, R. Fergus, A. Lazaric, and L. Pinto, “Reinforcement learning with prototypical representations,” in International Conference on Machine Learning. PMLR, 2021, pp. 11 920–11 931.

[117] C. Janiesch, P. Zschech, and K. Heinrich, “Machine learning and deep learning,” Electronic Markets, vol. 31, no. 3, pp. 685–695, 2021.

[118] P. P. Shinde and S. Shah, “A review of machine learning and deep learning applications,” in 2018 Fourth international conference on computing communication control and automation (ICCUBEA). IEEE, 2018, pp. 1–6.

[119] R. Kirk, A. Zhang, E. Grefenstette, and T. Rockta¨schel, “A survey of generalisation in deep reinforcement learning,” arXiv preprint arXiv:2111.09794, 2021.

[120] A. Oliver, A. Odena, C. A. Raffel, E. D. Cubuk, and I. Goodfellow, “Realistic evaluation of deep semi-supervised learning algorithms,” Advances in neural information processing systems, vol. 31, 2018.

[121] M. Naeem, S. T. H. Rizvi, and A. Coronato, “A gentle introduction to reinforcement learning and its application in different fields,” IEEE Access, vol. 8, pp. 209 320–209 344, 2020.

[122] J. Shin, T. A. Badgwell, K.-H. Liu, and J. H. Lee, “Reinforcement learning–overview of recent progress and implications for process control,” Computers & Chemical Engineering, vol. 127, pp. 282– 294, 2019.

[123] Y. Qian, J. Wu, R. Wang, F. Zhu, and W. Zhang, “Survey on reinforcement learning applications in communication networks,” Journal of Communications and Information Networks, vol. 4, no. 2, pp. 30–39, 2019.

[124] M. Lanctot, E. Lockhart, J.-B. Lespiau, V. Zambaldi, S. Upadhyay,

[125] J. Pe´rolat, S. Srinivasan, F. Timbers, K. Tuyls, S. Omidshafiei et al., “Openspiel: A framework for reinforcement learning in games,” arXiv preprint arXiv:1908.09453, 2019.

[126] J. Oh, M. Hessel, W. M. Czarnecki, Z. Xu, H. P. van Hasselt,

[127] S. Singh, and D. Silver, “Discovering reinforcement learning algorithms,” Advances in Neural Information Processing Systems, vol. 33, pp. 1060–1070, 2020.

[128] M. M. Afsar, T. Crump, and B. Far, “Reinforcement learning based recommender systems: A survey,” ACM Computing Surveys, vol. 55, no. 7, pp. 1–38, 2022.

[129] T. Salimans and R. Chen, “Learning montezuma’s revenge from a single demonstration,” arXiv preprint arXiv:1812.03381, 2018.

[130] L. Canese, G. C. Cardarilli, L. Di Nunzio, R. Fazzolari, D. Giardino, M. Re, and S. Spano`, “Multi-agent reinforcement learning: A review of challenges and applications,” Applied Sciences, vol. 11, no. 11, p. 4948, 2021.

[131] D. Mehta, “State-of-the-art reinforcement learning algorithms,” International Journal of Engineering Research and Technology, vol. 8, pp. 717–722, 2020.

[132] H. Fei, X. Li, D. Li, and P. Li, “End-to-end deep reinforcement learning based coreference resolution,” in Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019, pp. 660–665.

[133] E. R. Watters, “Factors in employee motivation: Expectancy and equity theories,” Journal of Colorado Policing, vol. 970, p. 4, 2021.

[134] B. Recht, “A tour of reinforcement learning: The view from continuous control,” Annual Review of Control, Robotics, and Autonomous Systems, vol. 2, pp. 253–279, 2019.

[135] K. Sharma, B. Singh, E. Herman, R. Regine, S. S. Rajest, and V. P. Mishra, “Maximum information measure policies in reinforcement learning with deep energy-based model,” in 2021 International Conference on Computational Intelligence and Knowledge Econ- omy (ICCIKE). IEEE, 2021, pp. 19–24.

[136] Dunjko, N. Friis, N. C. Harris, M. Hochberg, D. Englund et al., “Experimental quantum speed-up in reinforcement learning agents,” Nature, vol. 591, no. 7849, pp. 229–233, 2021.

[137] H. Helskyaho, J. Yu, and K. Yu, “Introduction to machine learn- ing,” in Machine Learning for Oracle Database Professionals. Springer, 2021, pp. 1–22.

[138] C. Li and M. Qiu, Reinforcement Learning for Cyber-Physical Sys- tems: with Cybersecurity Case Studies. Chapman and Hall/CRC, 2019.

[139] A. G. Barto, “Reinforcement learning: Connections, surprises, and challenge,” AI Magazine, vol. 40, no. 1, pp. 3–15, 2019.

[140] T. M. Moerland, J. Broekens, and C. M. Jonker, “Model-based reinforcement learning: A survey,” arXiv preprint arXiv:2006.16712, 2020.

[141] L. Kaiser, M. Babaeizadeh, P. Milos, B. Osinski, R. H. Campbell,

[142] K. Czechowski, D. Erhan, C. Finn, P. Kozakowski, S. Levine et al., “Model-based reinforcement learning for atari,” arXiv preprint arXiv:1903.00374, 2019.

[143] C. Sun, J. Orbik, C. M. Devin, B. H. Yang, A. Gupta, G. Berseth, and S. Levine, “Fully autonomous real-world reinforcement learn- ing with applications to mobile manipulation,” in Conference on Robot Learning. PMLR, 2022, pp. 308–319.

[144] A. S. Polydoros and L. Nalpantidis, “Survey of model-based reinforcement learning: Applications on robotics,” Journal of In- telligent & Robotic Systems, vol. 86, no. 2, pp. 153–173, 2017.

[145] P. Swazinna, S. Udluft, and T. Runkler, “Overcoming model bias for robust offline deep reinforcement learning,” Engineering Applications of Artificial Intelligence, vol. 104, p. 104366, 2021.

[146] A. Balakrishnan, D. Bouneffouf, N. Mattei, and F. Rossi, “Using multi-armed bandits to learn ethical priorities for online ai sys- tems,” IBM Journal of Research and Development, vol. 63, no. 4/5, pp. 1–1, 2019.

[147] T. Lykouris, M. Simchowitz, A. Slivkins, and W. Sun, “Corruption- robust exploration in episodic reinforcement learning,” in Confer- ence on Learning Theory. PMLR, 2021, pp. 3242–3245.

[148] D. Bouneffouf, I. Rish, and C. Aggarwal, “Survey on applications of multi-armed and contextual bandits,” in 2020 IEEE Congress on Evolutionary Computation (CEC). IEEE, 2020, pp. 1–8.

[149] R. N. Boute, J. Gijsbrechts, W. Van Jaarsveld, and N. Vanvuchelen, “Deep reinforcement learning for inventory control: A roadmap,”

[150] Z. Wang and T. Hong, “Reinforcement learning for building controls: The opportunities and challenges,” Applied Energy, vol. 269, p. 115036, 2020.

[151] Y. Liu, Y. Chen, and T. Jiang, “Dynamic selective maintenance optimization for multi-state systems over a finite horizon: A deep reinforcement learning approach,” European Journal of Operational Research, vol. 283, no. 1, pp. 166–181, 2020.

[152] G. Farina and T. Sandholm, “Model-free online learning in un- known sequential decision making problems and games,” in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, no. 6, 2021, pp. 5381–5390.

[153] T. Ding, Z. Zeng, J. Bai, B. Qin, Y. Yang, and M. Shahidehpour, “Optimal electric vehicle charging strategy with markov decision process and reinforcement learning technique,” IEEE Transactions on Industry Applications, vol. 56, no. 5, pp. 5811–5823, 2020.

[154] B. Zhang, G. Zhang, W. Sun, and K. Yang, “Task offloading with power control for mobile edge computing using reinforcement learning-based markov decision process,” Mobile Information Sys- tems, vol. 2020, 2020.

[155] X. Xiang and S. Foo, “Recent advances in deep reinforcement learning applications for solving partially observable markov decision processes (pomdp) problems: Part 1—fundamentals and applications in games, robotics and natural language processing,” Machine Learning and Knowledge Extraction, vol. 3, no. 3, pp. 554–581, 2021.

[156] S.Padakandla, P. KJ, and S. Bhatnagar, “Reinforcement learning algorithm for non-stationary environments,” Applied Intelligence, vol. 50, no. 11, pp. 3590–3606, 2020.

[157] Y. Lin, J. McPhee, and N. L. Azad, “Comparison of deep reinforcement learning and model predictive control for adaptive cruise control,” IEEE Transactions on Intelligent Vehicles, vol. 6, no. 2, pp. 221–231, 2020.

[158] XU, Zheng, KAMRUZZAMAN, M. M., et SHI, Jinyao. Method of generating face image based on text description of generating adversarial network. Journal of Electronic Imaging, 2022, vol. 31, no 5, p. 051411.

[159] XU, Yiheng, LI, Minghao, CUI, Lei, et al. Layoutlm: Pre-training of text and layout for document image understanding. In : Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2020. p. 1192-1200.

[160] CHANDRA, Sushant, SISODIA, Saurav, et GUPTA, Preeti. Optical character recognition-A review. International Research Journal of Engineering and Technology (IRJET), 2020, vol. 7, no 4, p. 3037-3041.

[161] NGUYEN, Thi Tuyet Hai, JATOWT, Adam, COUSTATY, Mickael, et al. Survey of post-OCR processing approaches. ACM Computing Surveys (CSUR), 2021, vol. 54, no 6, p. 1-37.

[162] WANG, Jing, TANG, Jinhui, YANG, Mingkun, et al. Improving OCR-based image captioning by incorporating geometrical relationship. In : Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021. p. 1306-1315.

[163] CHANDRA, Sushant, SISODIA, Saurav, et GUPTA, Preeti. Optical character recognition-A review. International Research Journal of Engineering and Technology (IRJET), 2020, vol. 7, no 4, p. 3037-3041.

[164] THORAT, Chhanam, BHAT, Aishwarya, SAWANT, Padmaja, et al. A detailed review on text extraction using optical character recognition. ICT Analysis and Applications, 2022, p. 719-728.

[165] BALDOMINOS, Alejandro, SAEZ, Yago, et ISASI, Pedro. A survey of handwritten character recognition with mnist and emnist. Applied Sciences, 2019, vol. 9, no 15, p. 3169.