PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

IBN KHALDOUN UNIVERSITY OF TIARET

Dissertation

PRESENTED TO:

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
DEPARTMENT OF COMPUTER SCIENCE

In order to obtain the degree of:

MASTER

Speciality: Software Engineering

Presented by:

Larbi ilyes louai

On the theme

Artificial Intelligence in Malignant Brain Tumor Detection

Publicly defended on 27 /05/2025 in Tiaret before the jury composed of:

Mr. BELARBI MOSTEFA	PR	Ibn Khaldoun University	Chairman
Mr. GAFOUR YACINE	MCA	Ibn Khaldoun University	Examiner
Mr. DJAFRI LAOUNI	MCA	Ibn Khaldoun University	Supervisor
Mr. BAGHDADI MOUHAMED	MCA	Ibn Khaldoun University	Co. Supervisor

2024-2025

Acknowledgements

First, I thank **God** Almighty for granting me the courage and will to undertake this work. I extend my sincere thanks, with pleasure and respect, to **Dr. Djafri Laouni** and **Dr. Baghdadi Mouhamed** for their efforts, patience, and continuous support in completing this work. I also extend my sincere thanks to the members of the jury who honored us by participating in the discussion of this modest work and enriching it with their suggestions: **Dr. Gafour Yacine** and **Pr. Belarbi Mostefa**. I would like to thank my **family**, especially my dear **mother**, who supported me morally and stood by me.

Dedicate

I dedicate this humble work to the soul of my dear **mother**, who supported me throughout my academic journey and was like my right arm.

To my **father**, who encouraged me emotionally despite the distance.

To all my **siblings**: Yasser, Fouad, and my angel sister Maram.

To all my **friends**: Ihab, Mostafa, Mohamed, Akram, Khaled, Ali, Imad, Kamal, and Chaima, especially Amin, who was like an older brother with his constant motivation.

And to all the members of my big **family**.

ملخص

أورام الدماغ مشكلة صحية خطيرة، خاصةً عندما لا تُكتشف مبكرًا. تُعد فحوصات التصوير بالرنين المغناطيسي مفيدة جدًا في اكتشاف هذه الأورام وتتبعها، إلا أن تحليل الصور يدويًا يستغرق وقتًا طويلًا وقد يؤدي إلى أخطاء، خاصةً عندما تكون الأورام صغيرة أو يصعب التمييز بينها.

وهنا يأتي دور التعلم العميق، فهو يُمكّن من تحليل كميات كبيرة من البيانات والعثور على أنماط تُساعد في مهام مثل تحديد نوع الورم وتحديد موقعه بدقة في الدماغ. هذا لا يُسهّل التشخيص ويجعله أكثر موثوقية فحسب، بل يُساعد الأطباء أيضًا على وضع خطط علاج أفضل لكل مريض.

في هذا المشروع، نهدف إلى بناء نظام تعلم عميق يُمكّن من القيام بوظيفتين رئيسيتين: تصنيف صور الرنين المغناطيسي للدماغ إلى أنواع من الأورام مثل (الورم الدبقي، الورم السحائي، ورم الغدة النخامية و عدم وجود ورم)، وتقسيم منطقة الورم بدقة. من خلال الجمع بين الخطوتين في أداة ذكية واحدة، نأمل في جعل عملية التشخيص أكثر كفاءة ودعم استخدام الذكاء الاصطناعي في الرعاية الطبية.

الكلمات المفتاحية: ورم الدماغ، التعلم العميق، الشبكات العصبية التلافيفية، التعلم الانتقالي، التصنيف، التجزئة، الورم الدبقي، الورم السحائي، ورم الغدة النخامية، التصوير الطبي، تطبيق الويب.

Abstract

Brain tumors are a serious health issue, especially when they're not detected early. Quick diagnosis is really important to improve the chances of survival. MRI scans are very useful for spotting and tracking these tumors, but analyzing the images manually takes a lot of time and can lead to mistakes, especially when the tumors are small or hard to tell apart.

That's where deep learning comes in. It can analyze big amounts of data and find patterns that help with tasks like identifying the type of tumor and showing exactly where it is in the brain. This doesn't just make diagnosis faster and more reliable, it also helps doctors create better treatment plans for each patient.

In this project, our goal is to build a deep learning system that can do two main things: classify brain MRIs into tumor types like (glioma, meningioma, pituitary tumor, or no tumor), and accurately segment the tumor area. By combining both steps in one smart tool, we hope to make the diagnosis process more efficient and support the use of AI in medical care.

Keywords: Brain Tumor, Deep Learning, CNN, Transfer Learning, Classification, Segmentation, Glioma, Meningioma, Pituitary Tumor, Medical Imaging, Web Application.

Résumé

Les tumeurs cérébrales constituent un problème de santé grave, surtout lorsqu'elles ne sont pas détectées tôt. Un diagnostic rapide est essentiel pour améliorer les chances de survie. Les IRM sont très utiles pour repérer et suivre ces tumeurs, mais l'analyse manuelle des images prend beaucoup de temps et peut entraîner des erreurs, surtout lorsque les tumeurs sont petites ou difficiles à distinguer.

C'est là qu'intervient l'apprentissage profond. Il permet d'analyser de grandes quantités de données et de dégager des schémas qui facilitent des tâches telles que l'identification du type de tumeur et sa localisation précise dans le cerveau. Cela permet non seulement d'accélérer et de fiabiliser le diagnostic, mais aussi d'aider les médecins à élaborer de meilleurs plans de traitement pour chaque patient.

Dans ce projet, notre objectif est de développer un système d'apprentissage profond capable de réaliser deux tâches principales : classer les IRM cérébrales par type de tumeur (gliome, méningiome, tumeur hypophysaire ou absence de tumeur) et segmenter précisément la zone tumorale. En combinant ces deux étapes dans un seul outil intelligent, nous espérons optimiser le processus de diagnostic et favoriser l'utilisation de l'IA dans les soins médicaux.

Mots-clés : tumeur cérébrale, apprentissage profond, CNN, apprentissage par transfert, classification, segmentation, gliome, méningiome, tumeur hypophysaire, imagerie médicale, application Web.

Contents

General introduction

1 Brain Tumor 2

Introduction 2

1.1 Tumor definition 2

1.2 Brain tumor definition 2

1.3 Types of brain tumors 3

1.3.1 Common Types of Brain Tumors 3

1.4 Impact of brain tumors on health 5

1.5 Causes and symptoms of brain tumors 5

1.5.1 Causes 5

1.5.2 Symptoms 6

Conclusion 7

2 Brain tumor detection techniques 8

Introduction 8

2.1 Traditional techniques 8

2.1.1 Magnetic Resonance Imaging 8

2.1.1.1 The different weightings 8

2.1.2 Biopsy 10

2.1.3 Electroencephalogram (EEG) 10

2.1.4 Fluorescence Imaging 11

2.2 Artificial Intelligence Techniques 11

2.2.1 Deep Learning 11

2.2.1.1 Artificial Neural Network (ANN) 11

2.2.1.2 Convolutional Neural Networks (CNN) 12

2.2.1.2.1 Principle of CNNs 12

2.2.1.2.2 Layers of a CNNs 12

2.2.1.2.3 The importance of CNNs in image classification 15

2.2.1.3 Training and Optimizing Deep Learning Models 15

2.2.1.3.1 Optimization Algorithms 16

2.2.1.3.2 Regularization techniques 16

2.2.1.4 Applications of Deep Learning in medical imaging 17

2.2.1.5 Some famous convolutional networks 18

2.2.2 Image classification and segmentation 18

2.2.2.1 Classification 19

2.2.2.1.1 Types of image classification 19

2.2.2.2 Classification models for brain tumors 19

2.2.2.2.1 Network used for classification 19

2.2.2.2.2 Comparison of architectures 24

2.2.2.3 Segmentation 25

2.2.2.3.1 Types of Segmentation 26

2.2.2.4 Segmentation models for brain tumors 27

2.2.2.4.1 Some semantic segmentation architecture 27

2.2.2.4.2 Segmentation evaluation methods 32

Conclusion 34

3 Conception and realization 34

Introduction 34

3.1 Presentation of the overall architecture of the model.....34
3.2 Image preprocessing.....37
3.2.1 Image cropping.....37
3.2.2 Image Resizing.....37
3.2.2 Image Normalization.....38
3.2.3 Data augmentation.....39
3.3 Choice of models for classification.....40
3.3.1 Ensemble learning.....40
3.3.1.1 Ensemble learning techniques.....41
3.3.1.1 Reason for use.....42
3.4 Choice of model for segmentation.....42
3.4.1 Reason for use 42
3.5 Dataset structuring.....43
3.5.1 Classification dataset.....43
3.5.1.1 Data splitting.....44
3.5.2 Segmentation dataset.....44
3.6 Project pipeline.....46
3.7 Development tools and environment.....46
3.7.1 The programming language used.....46
3.7.2 Kaggle.....46
3.7.3 Libraries used.....47
3.7.4 Google Colaboratory.....48
3.7.5 Hardware configuration.....48
3.8 Implementation of the classification model.....48

3.8.1 Training, optimization and evaluation metrics.....48
3.8.1.1 Dataset Preparation.....49
3.8.1.2 Hyperparameters.....50
3.8.1.3 Techniques for optimization.....52
3.8.1.4 Fully connected layer.....56
3.8.1.5 Activation function.....56
3.8.1.6 Ensemble Learning Approach.....57
3.8.1.6 Evaluation metrics.....57
3.8.2 Performance evaluation and analysis (discussion of results) 59
3.8.2.1 Xception.....59
3.8.2.2 InceptionV3.....60
3.8.2.3 ResNet50.....61
3.8.2.4 VGG19.....62
3.8.2.5 Confusion matrix.....63
3.8.2.6 Roc-Auc curve.....64
3.8.2.7 Stacking model.....65
3.9 Implementation of the segmentation model.....66
3.9.1 Training and optimization.....66
3.9.1.1 Dataset Preparation.....66
3.9.1.2 Hyperparameters.....67
3.9.1.3 Techniques for optimization.....68
3.9.1.4 Res-Unet architecture.....70
3.9.2 Performance evaluation and analysis (discussion of results) 72
3.9.2.1 Res-Unet.....72

3.9.2.2 Attention U-Net……73

3.9.2.3 U-Net……74

3.10 Comparison of the proposed model……75

3.10 Application interface……75

Conclusion 81

General conclusion

Bibliography

List of Figures

1.1 Brain tumor……3

1.2 Glioma tumor 1……4

1.3 Meningioma tumor 1……4

1.4 Pituitary tumor 1……4

2.1 Gray matter, White matter……9

2.2 Comparaison of T1 vs T2 vs Flair……9

2.3 EEG (Electroencephalogram) ……10

2.4 Fluorescence Imaging……11

2.5 Convolutional Neural Networks architecture……12

2.6 Convolution layer architecture……13

2.7 RELU Layer……13

2.8 Max pooling……14

2.9 Average pooling……14

2.10 Fully connected layer……15

2.11 Dropout regularization technique……17

2.12 Residual block……20

2.13 VGG-19 architecture……21

2.14 Inception module with dimension reduction……22

2.15 Original Depthwise Separable Convolution……23

2.16 Xception architecture……24

2.17 Semantic segmentation……26

2.18 Instance segmentation.....27

2.19 The architecture for U-NET.....28

2.20 Res-Unet architecture.....30

2.21 U-net++ architecture.....32

3.1 Architecture of the proposed model 37

3.2 Original image.....38

3.3 Cropped image.....38

3.4 Image preprocessing.....39

3.5 Some glioma tumor using data augmentation.....41

3.6 General ensemble learning method.....42

3.7 Glioma tumor 2.....44

3.8 Meningioma tumor 2.....44

3.9 No tumor.....44

3.10 Pituitary tumor 2.....44

3.11 Tumor 1.....46

3.12 Mask of Tumor 1.....46

3.13 Tumor 2.....46

3.14 Mask of Tumor 2.....46

3.15 Some augmented images with small tumor.....46

3.16 Masks of this images augmented with small tumor 47

3.17 Project pipeline.....47

3.18 Image resizing and normalization for classification task.....50

3.19 Splitting data into training, testing and validation data.....51

3.20 Optimizer, learning rate, loss……52

3.21 First 10 epochs……52

3.22 last 10 epochs……53

3.23 Callbacks (ReduceLROnPlateau, EarlyStopping) ……53

3.24 Dropout……54

3.25 Dataset augmentation……54

3.26 Dataset before data augmentation (imbalanced) ……55

3.27 Dataset after data augmentation (balanced) ……55

3.28 Data visualization of the classification dataset before balancing (imbalanced)……56

3.29 Data visualization of the classification dataset after balancing (balanced)……56

3.30 Implementation of the fully connected layer ……57

3.31 Implementation of the stacking model……58

3.32 Confusion matrix……59

3.33 The loss curve and the accuracy curve Xception……60

3.34 CSV Export of Training Logs for Xception……61

3.35 The loss curve and the accuracy curve for InceptionV3……61

3.36 CSV Export of Training Logs for InceptionV3……62

3.37 The loss curve and the accuracy curve for ResNet50……62

3.38 CSV Export of Training Logs for ResNet50……63

3.39 The loss curve and the accuracy curve for VGG19……63

3.40 CSV Export of Training Logs for VGG19……64

3.41 The confusion matrices for the four models……64

3.42 Roc/Auc curve for Xception……65

3.43 Roc/Auc curve for InceptionV3……65

3.44 Roc/Auc curve for ResNet-50……65

3.45 Roc/Auc curve for VGG-19……65

3.46 The confusion matrix for the stacking model……66

3.47 The ROC/Auc curve for the stacking model……66

3.48 The Training loss and accuracy curve for the stacking model……66

3.49 CSV Export for stacking model……66

3.50 Image resizing and normalization for segmentation task……67

3.51 Splitting data into training, testing and validation data……68

3.52 Hyperparameters used on segmentation task……69

3.53 Callbacks for segmentation task……69

3.54 Add batch normalization to our Res-Unet model……70

3.55 Number of images and masks augmented. ……70

3.56 Dataset before augmentation……71

3.57 Dataset after augmentation ……71

3.58 The creation of the model……72

3.59 The dice_coef curve and the accuracy curve for Res-Unet……73

3.60 CSV Export of Training Logs for Res-Unet……73

3.61 The dice_coef curve and the accuracy curve for Attention U-NET……74

3.62 CSV Export of Training Logs for Attention U-NET……74

3.63 The dice_coef curve and the accuracy curve for U-NET……75

3.64 CSV Export of Training Logs for U-NET……75

3.65 Application interface……78

3.66 An example for glioma tumor……79

3.67 An example for an ambiguous image……80

3.68 An example for no tumor……81

3.69 An example for no tumor 2……82

3.67 An example for no tumor 3……83

List of Tables

2.1 Comparison of architectures ……..24

3.2 The different input size for Xception, Inception, ResNet50, VGG-19……39

3.3 Comparison of the Proposed Model with State-of-the-Art Models for Multi-Class Brain Tumor Classification……76

3.4 Accuracy Comparison of Existing Brain Tumor Classification Models and the Proposed Model……76

List of Abbreviations

MRI	**M**agnetic **R**esonance **I**maging
EEG	**E**lectroencephalogram
CSF	**C**erebrospinal **F**luid
TE	**E**cho **T**ime
TR	**R**epetition **T**ime
FLAIR	**F**luid **A**ttenuated **I**nversion **R**ecovery
AI	**A**rtificial **I**ntelligence
ANN	**A**rtificial **N**eural **N**etwork
CNN	**C**onvolutional **N**eural **N**etwork
SGD	**S**tochastic **G**radient **D**escent
ADAM	**A**daptive **M**oment **E**stimation
DL	**D**eep **L**earning
IoU	**I**ntersection over **U**nion
ROI	**R**egion of **I**nterest
ResNet	**R**esidual **N**etwork
RELU	**R**ectified **L**inear **U**nit
TP	**T**rue **P**ositive
TN	**T**rue **N**egative
FP	**F**alse **p**ositive
FN	**F**alse **N**egative

General Introduction

The brain is one of the most complex and vital organs in the human body. It controls essential functions such as movement, memory, emotions, and thinking. However, this complexity makes it vulnerable to various disorders, among which brain tumors pose a particularly serious threat. These tumors result from abnormal and uncontrolled cell growth, which can disrupt normal brain activity and lead to serious health complications or even death if not detected and treated early.

Brain tumors are generally classified into two main types: benign and malignant. Among the most common types of brain tumors are gliomas, meningiomas, and pituitary tumors, each of which varies in location, behavior, and severity. Therefore, accurate and early diagnosis is crucial to improving treatment outcomes and survival rates.

Medical imaging, particularly magnetic resonance imaging (MRI), plays a key role in detecting and monitoring brain tumors. However, manually analyzing these images can be a challenging and time-consuming task for radiologists. The increasing volume of medical data, coupled with subtle visual differences between tumor types, often leads to delays or misdiagnoses.

In this context, deep learning has emerged as an effective tool to support the medical diagnostic process. Deep learning models, particularly convolutional neural networks (CNNs), are capable of learning from massive datasets and performing tasks such as image classification and segmentation with high accuracy. These models not only automatically detect and classify tumors but also help extract important morphological features, allowing for more accurate assessments.

To this end, we developed an intelligent system based on deep learning that performs three main tasks. First, it classifies brain MRIs into four categories: glioma, meningioma, pituitary tumor, or no tumor. Second, it segments the tumor area in the brain image to accurately locate the affected area. Third, it extracts tumor morphological features, such as surface, perimeter, width, height, and circularity.

This work consists of three chapters:

A general introduction

Chapter 1: Brain Tumor

In this chapter, we provide an overview of brain tumors, their types, and their effects on the human body.

Chapter 2: Brain tumor detection techniques

In this chapter, we focus on the main technologies used to detect brain tumors, including the use of artificial intelligence techniques.

Chapter 3: Conception and realization

In this final chapter, we present the design and implementation of our proposed system, which aims to support clinicians in making faster and more reliable diagnoses.

This thesis ends with **a general conclusion**

Chapter 1

Brain Tumor

Introduction

The brain is one of the most complex organs in the human body and the central organ of the human nervous system. It is responsible for processing sensory information, controlling movement, regulating vital functions, and enabling thought, memory, emotions, and decision-making. It works with billions of cells, sometimes these cells can be exposed to an uncontrolled division of cells and formed an abnormal group of cells around or inside the brain that's what we called brain tumors. It is a growth of cells in or near the brain. It can develop within brain tissue or in surrounding areas, such as nerves, the pituitary gland, the pineal gland, and the membranes covering the brain. It can begin in the brain. These are called primary brain tumors. Sometimes, cancer spreads to the brain from other parts of the body. These tumors are secondary brain tumors, also called metastatic brain tumors. Many different types of primary brain tumors exist. Some brain tumors aren't cancerous. These are called noncancerous brain tumors or benign brain tumors. Noncancerous brain tumors may grow over time and press on the brain tissue. Other brain tumors are brain cancers, also called malignant brain tumors. Brain cancers may grow quickly. The cancer cells can invade and destroy the brain tissue.

Brain tumors range in size from very small to very large. Some brain tumors are found when they are very small because they cause symptoms that you notice right away. Other brain tumors grow very large before they're found. Some parts of the brain are less active than others. If a brain tumor starts in a part of the brain that's less active, it might not cause symptoms right away. The brain tumor size could become quite large before the tumor is detected.

1.1 Tumor definition

A tumor, also known as a neoplasm, is an abnormal growth of tissue that occurs when cells begin to grow uncontrollably and fail to die in time. Tumors can appear almost anywhere in the body, including bones, skin, organs, glands, and tissues. While the majority of tumors are benign that is, noncancerous and do not spread to other parts of the body some may be malignant, or cancerous, multiplying rapidly, invading nearby tissue, or extending to distant organs. Benign tumors may or may not require treatment, depending on their size, location, and symptoms, while malignant tumors often require prompt and specific cancer treatments [1].

1.2 Brain tumor definition

A brain tumor is an abnormal mass of brain cells that develops in or near the brain. Brain tumors can be either benign or malignant, and the way they influence brain function depends a lot on their size, growth rate, and location within the brain. While some brain tumors develop slowly and do not have symptoms

for several years, others develop rapidly and may interfere with essential neurological functions and result in serious health issues [1][2].

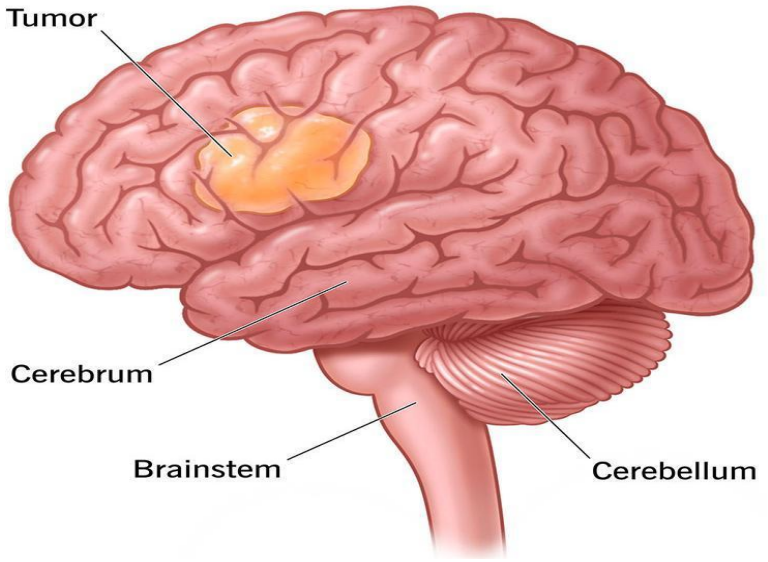


Figure 1.1 – Brain tumor.

1.3 Types of brain tumors

- **Benign:** These tumors do not multiply or spread, and in some cases, they grow very slowly. They are non-cancerous.
- **Pre-malignant:** Although these tumors are initially benign, they have the potential to become cancerous over time.
- **Malignant:** Composed of cancerous cells, these tumors grow rapidly and can invade nearby tissues. Cancer can develop in any part of the body.

1.3.1 Common Types of Brain Tumors

- **Gliomas:** Gliomas are a common type of malignant brain tumor that originate from glial cells the supportive cells that surround and protect neurons. They account for approximately 33 percent of all brain tumors and include subtypes such as astrocytes, oligodendrocytes and microglia cells depending on the specific glial cells affected [3].

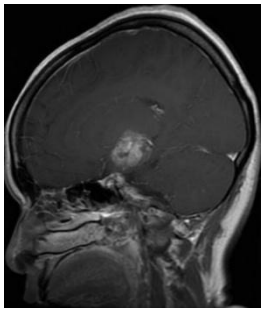


Figure 1.2 – Glioma tumor 1.

- **Meningiomas:** Meningiomas are tumors that begin in the layers of tissue (meninges), the protective membranes surrounding the brain and spinal cord. While most meningiomas are benign and slow-growing, some can become aggressive, requiring medical intervention such as surgery or radiation therapy [4].

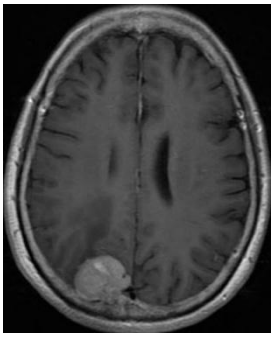


Figure 1.3 – Meningioma tumor 1.

- **Pituitary tumors:** Almost all pituitary tumors are benign (not cancer) glandular tumors called pituitary adenomas, these tumors don't spread to other parts of the body, like cancers can. Still, even benign pituitary tumors can cause major health problems [5].

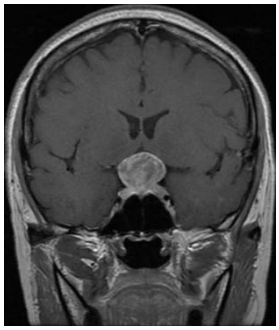


Figure 1.4 – Pituitary tumor 1.

1.4 Impact of brain tumors on health

Brain tumors significantly affect both physical and mental health, depending on their size, location, and aggressiveness. Here are some key impacts [6]:

Neurological Symptoms: Various neurological symptoms can result from brain tumors, depending on their location, size, and kind. Constant headaches, seizures, blurred vision, trouble speaking, and weakness in the muscles are common side effects. Usually, the tumor's pressure on the surrounding brain tissues or interference with the brain's regular processes causes these symptoms.

Cognitive Decline: The areas responsible for thinking and memory can be affected by brain tumors, leading to memory loss, shortened attention span, and difficulty with executive functions such as planning, organizing, and decision-making. This disorder often impacts the patient's ability to perform daily routine activities effectively.

Emotional and Psychological Effects: When parts of the brain that control emotions are involved, patients tend to develop mood changes, despair, and anxiety. In addition to the biological effects of the tumor, mental status and general well-being can also be affected considerably by the stress of therapy and diagnosis.

Hormonal Imbalances: The tumor affects the pituitary gland and has the potential to disrupt the normal secretion of growth hormone and other required hormones, leading to numerous physical changes.

Reduced Quality of Life: Daily life can be significantly impacted by sleep disturbances, chronic fatigue, and a loss of independence, making it harder for patients to maintain their usual routines and well-being.

1.5 Causes and symptoms of brain tumors

1.5.1 Causes

Until now, the main reason for the growth of these brain tumors is still unknown, but there are some factors that may increase the probability of developing them, which are:

- **Genetic Mutations:** When certain genes in a cell’s chromosomes become damaged, human brain tumor develops. These genes control cells growth, division and death sequentially. When they change, they give brain cells unusual orders that make them grow and divide without control [1].

- **Inherited Genetic Factors:** Some people may be born with altered DNA that raises their chances of getting a brain tumor. Brain cells can be genetically prone to tumor growth as they inherit genetic mutations that make them sensitive to change [1].

- **Environmental Triggers:** If you get exposed to a very high dose of radiation for a long time (from X-rays or for treatment of cancer), then it may damage the DNA of brain cells, which may initiate or promote tumorigenic processes. Sometimes the environment itself causes damage in a way that genes do not cause any [1][2].

1.5.2 Symptoms

Signs and symptoms of brain tumors are usually described as general or focal. Generally, a low-grade tumor corresponds to focal signs that generalize with the increase in grade or dimension of the tumor. General symptoms usually include [6]:

- **Vomiting:** The abrupt release of stomach contents, frequently brought on by elevated intracranial pressure.
- **Nausea:** Often associated with brain pressure or irritation and may lead to vomiting.
- **Headache:** Constant discomfort in the head, usually brought on by swelling or pressure from a tumor.
- **Sensory deficit:** A sensory deficit occurs when a tumor affects particular parts of the brain, resulting in a loss or diminution of feeling (touch, sight, hearing, etc.).
- **Impaired cognitive and emotional functions:** When a tumor interferes with brain function, it can lead to problems thinking, reasoning, or controlling emotions.
- **Memory loss:** A tumor that affects the brain's memory-related regions may cause memory loss, which is the inability to recall information.

Conclusion

In this chapter, we presented a general overview of what tumors are, then we specialized in brain tumors, where we saw a comprehensive definition and that there are malignant and benign ones. Then we touched on their famous types and saw their negative impact on human health. Finally, we took some of the causes that may contribute to causing these tumors because we said that the main cause for them has not been found, and we also listed some of the symptoms that occur in the person afflicted with them.

In the next chapter, we will talk about some modern methods in the field of deep learning to help doctors quickly and easily identify the type and location of these tumors.

Chapter 2

Brain tumor detection techniques

Introduction

Accurate detection of brain tumors is key in diagnosis and treatment. This field is dominated by conventional technologies such as Magnetic Resonance Imaging (MRI), biopsy, Electroencephalography (EEG), fluorescence imaging, etc. These methods offer useful information about the presence and type of tumor, but they have major drawbacks. These requires careful interpretation and may take a long time for novice doctors. Also, these methods may lack sufficient information, and precision. To help solve the issues faced by doctors due to radiological techniques, the AI-based techniques, especially Deep Learning, was used for detection of brain tumors. AI models can study large amounts of medical images and do this very accurately. They help radiologists to identify and segment the tumor conveniently. These techniques provide quicker processing times and enhanced consistency with lower reliance on human expertise.

In this chapter, we will discuss the traditional detection techniques about which we will present a brief overview. We will, however, be providing a detailed and in-depth overview of Artificial Intelligence techniques and their strength which is commonly used in modern medical diagnostics.

2.1 Traditional techniques

2.1.1 Magnetic Resonance Imaging

Magnetic resonance imaging (MRI) is a medical imaging technique that uses a magnetic field and computer-generated radio waves to create detailed images of organs and tissues in your body. Most of these devices are large tube-shaped magnets. When you lie inside an MRI machine, the magnetic field inside it works with radio waves and hydrogen atoms in your body to create cross-sectional images that can also produce 3D images that can be viewed from different angles [7].

2.1.1.1 The different weightings

MRI provides several types of image contrast, known as weights, which highlight different tissue features including [8][9]:

T1-weighted: Generated using short TE and TR values, In these images:

- Cerebrospinal Fluid (CSF) appears dark (black).
- Gray matter is outside.
- White matter is inside.

T2-weighted: Generated with longer TE and TR times, In this case:

- CSF appears bright (white).
- Gray matter is inside.
- White matter is outside.

FLAIR (Fluid-Attenuated Inversion Recovery): Similar to T2-weighted imaging, but with significantly longer TE and TR times and dark CSF.

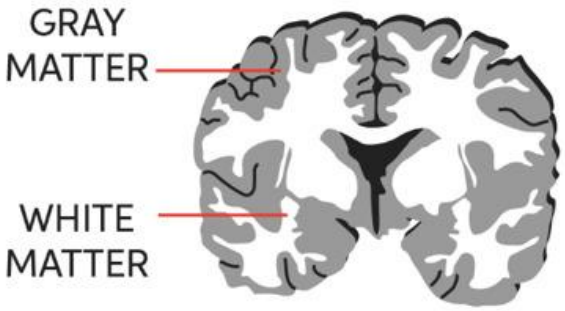


Figure 2.1 – Gray matter, White matter.

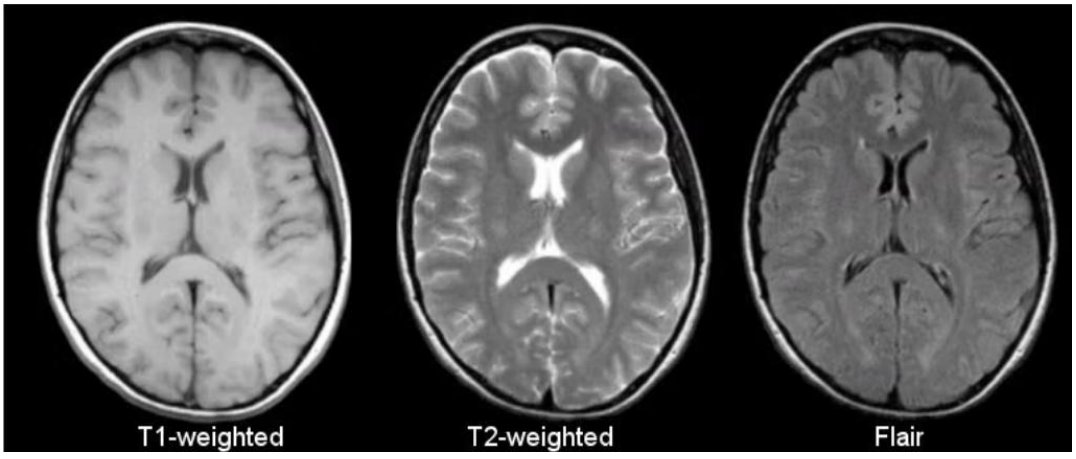


Figure 2.2 – Comparaison of T1 vs T2 vs Flair.

2.1.2 Biopsy

A biopsy is a procedure to remove cells, tissue or fluid for examination by a medical pathologist. Healthcare providers do biopsies when they identify areas of concern or if you have symptoms or signs of certain conditions. There are different types of biopsy procedures including [10]:

- **Bone marrow biopsy:** Healthcare providers take a small sample of bone marrow using a specialized biopsy needle and syringe to detect blood malignancies, blood disorders and cancers, and other diseases.

- **Excisional biopsy or incisional biopsies:** In order to remove tissue from inside your body, medical professionals perform these operations by making cuts or incisions. Whole masses or questionable regions are removed during excisional biopsies. Tissue samples are taken from lumps or other questionable regions during incisional biopsies.

- **Liquid biopsy:** This blood test detects signs of cancerous cells or cancer cell DNA.

- **Needle biopsy:** Healthcare providers use needle biopsies to remove tissue, fluid, or cells. If a bulge or abnormal growth is noticed in your body, or if imaging tests reveal potential problems, a needle biopsy may be the solution.

2.1.3 Electroencephalogram (EEG)

An EEG is a technique used to detect abnormalities in the electrical impulses or brain waves. Electrodes are applied to the scalp during this examination. The tiny electrical impulses generated by the brain's neurons are picked up by these electrodes, which are tiny metal disks connected by thin cables. They can be printed on paper or magnified and shown as a visual representation on a computer screen. The EEG readings are then examined and interpreted by a medical expert [11].

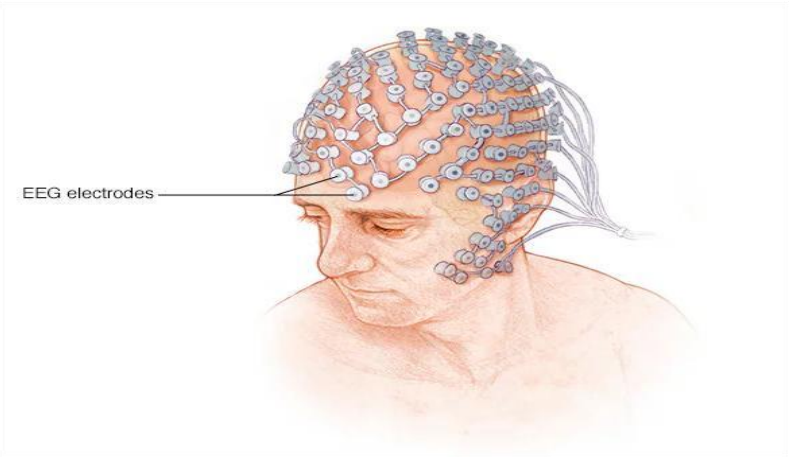


Figure 2.3– EEG (Electroencephalogram).

2.1.4 Fluorescence Imaging

Fluorescence imaging, which is frequently employed with fluorescent proteins or fluorochromes to observe cell compartments or molecules, is a technique that detects light that is released after being activated by particular wavelengths [12].

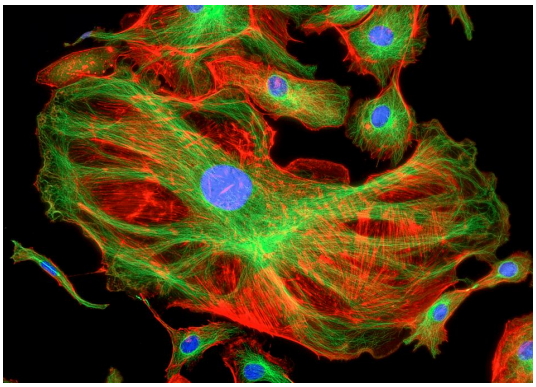


Figure 2.4 – Fluorescence Imaging.

2.2 Artificial Intelligence Techniques

2.2.1 Deep Learning

Deep learning is an artificial intelligence (AI) technology and an emerging field of machine learning that uses artificial neural networks, which mimic the human brain, to analyze and interpret knowledge. It can automate processes that typically require human intelligence, such as describing images and converting audio to text, using deep learning techniques.

There are two basic techniques or types in deep learning:

2.2.1.1 Artificial Neural Network (ANN)

Architecture: Artificial neural networks consist of layers of interconnected neurons: an input layer, one or more hidden layers, and an output layer.

Neurons: All neurons are fully connected to all neurons in the previous layer and the next layer.

Examples of use cases: simple or binary classification, regression problems, and also prediction of numerical results [13].

2.2.1.2 Convolutional Neural Networks (CNN)

2.2.1.2.1 Principle of CNNs

Convolutional neural networks (CNNs) are a class of deep neural networks used primarily in the field of image processing. Their functionality is based on extracting local features like edges, textures, and patterns by applying convolutional filters to the model's input data. They are very effective at discovering complex patterns in visual information. They are also a mathematical construct that typically consist of three types of layers: convolutional layers, pooling layers, and fully connected layers. We will see these types in detail [14][15].

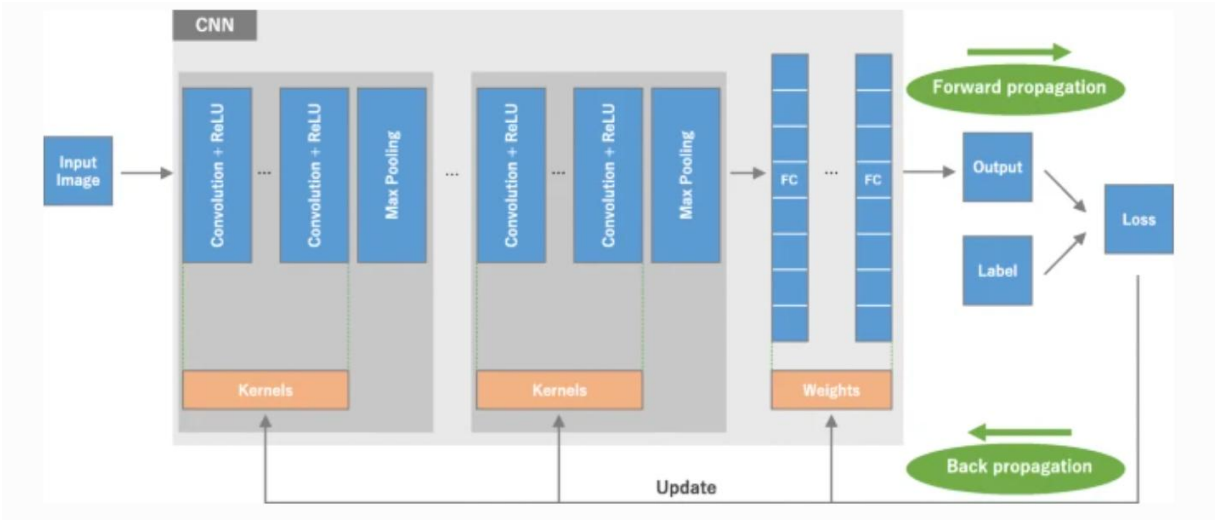


Figure 2.5 – Convolutional Neural Networks architecture.

2.2.1.2.2 Layers of a CNNs

- **Convolution layer (Conv + RELU):**

Convolution is a linear operation used to extract features, by applying a small set of numbers called a filter to the input, which is a set of numbers called a tensor, where the product of each element of the kernel and the input tensor at each location of the tensor is calculated and summed to get the output value at the corresponding location of the output tensor, which is called a feature map. This procedure is repeated by applying multiple kernels to form a random number of feature maps, which represent different properties of the input tensor. Thus, different kernels can be considered different feature extractors. There are two main parameters that define the convolution operation, the size and the number of kernels. The former is 3 × 3, but sometimes 5 × 5 or 7 × 7. The latter is random, and determines the depth of the output feature maps [15].

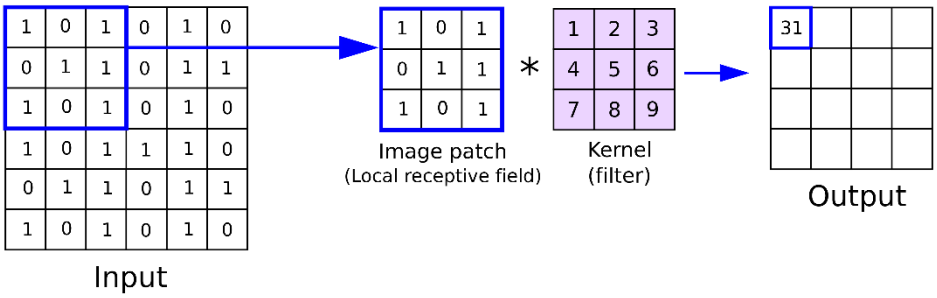


Figure 2.6 – Convolution layer architecture.

- **Activation Function:**

An activation function, usually "ReLU", is used after convolution to provide nonlinearity, ensure that the model can learn more complex associations, and convert negative results to zeros [14].

$$f(x) = \max(0, x)$$

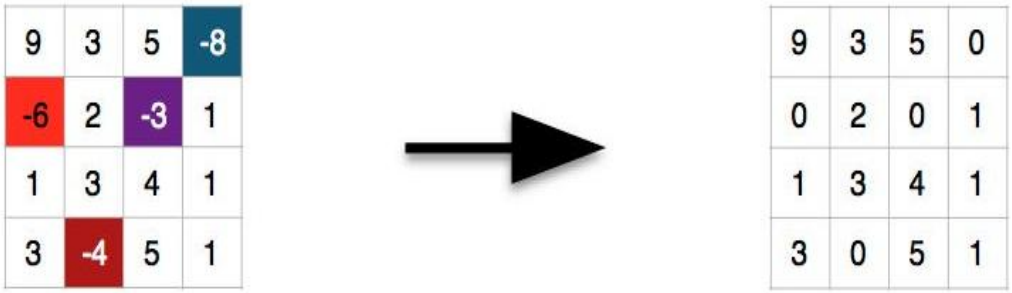


Figure 2.7 – RELU Layer.

- **Pooling layer:**

Pooling operations are used to down sample the feature maps and reduce the dimensions while retaining important features, it can be of different types:

1. **Max Pooling:** The maximum element from the area of the feature map that the filter covers is chosen by max pooling. Therefore, a feature map with the most noticeable features from the prior feature map would be the output following the max-pooling

layer. In most situations, the max pooling layer offers superior performance while maintaining the most crucial elements (textures, edges, etc.) [16].

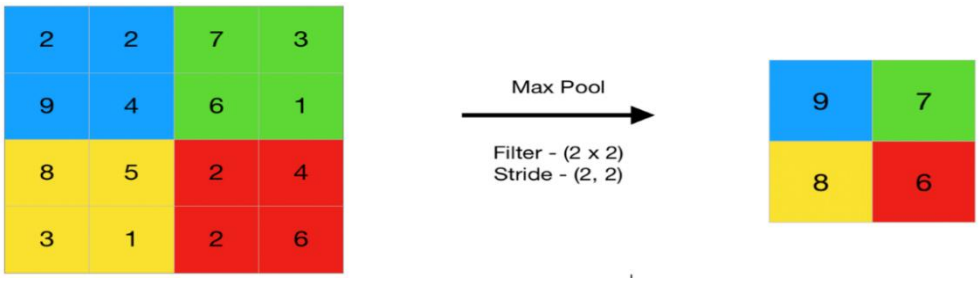


Figure 2.8 – Max pooling.

2. **Average Pooling:** The average of the elements found in the feature map area that the filter covers is calculated using average pooling. Therefore, average pooling provides the average of the features in a patch, whereas max pooling provides the most noticeable feature in a specific patch of the feature map [16].

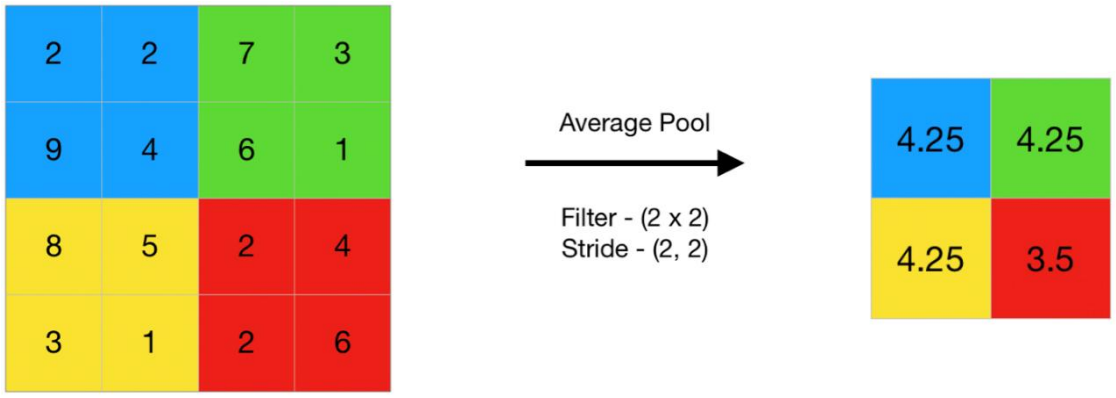


Figure 2.9 – Average pooling.

- **Fully connected layer:**

After applying convolution and pooling, the output is a multidimensional matrix where the data must be flattened into a one-dimensional vector to be the input to fully connected layers, so named because every neuron in one layer is connected to every neuron in the previous layer, creating a highly interconnected network [14][17].

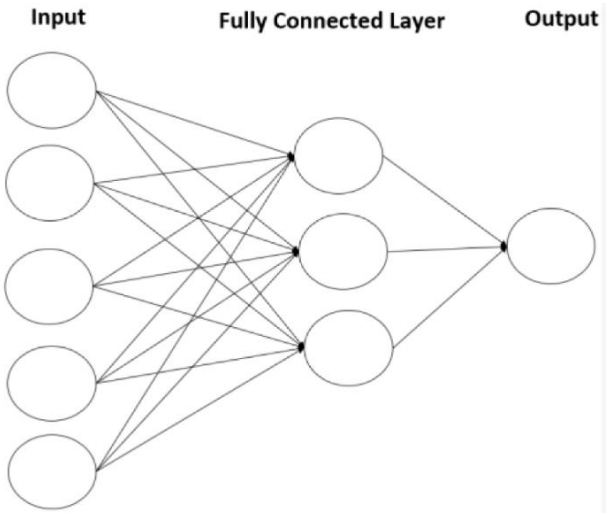


Figure 2.10 – Fully connected layer.

- **Output layer:**

  The final fully-connected layer obtains the output probabilities using an activation function like sigmoid or SoftMax.

2.2.1.2.3 The importance of CNNs in image classification

Convolutional neural networks have become essential in the field of image classification because they provide a powerful way to analyze and recognize visual patterns. Convolutional neural networks automatically learn features from images through multiple layers of processing. They efficiently capture spatial hierarchies by detecting edges, textures, and complex shapes, enabling them to recognize objects with high accuracy and making them widely used in areas such as medical imaging, facial recognition, and autonomous vehicles. Their ability to reduce computational complexity while improving performance has made them the preferred choice for modern image classification tasks.

2.2.1.3 Training and Optimizing Deep Learning Models

Deep learning models are trained by feeding large datasets through neural networks to optimize weights and reduce errors, in order to avoid large discrepancies between the predicted outputs and the truth output. This is usually done using the following steps [18]:

- **Forward propagation:** the process where the input moves forward through the network layer by layer to generate an output, his goal compute the predicted output and measure the error.

- **Backward propagation:** the process of updating weights using the error computed in forward propagation, his goal minimizing the loss function by updating model parameters.

2.2.1.3.1 Optimization Algorithms

The optimization algorithm used determines how well deep learning models are trained, some popular algorithms include:

- **Stochastic Gradient Descent (SGD):**

is a deep learning model training optimization algorithm that adjusts model weights to minimize the loss function, it is a Gradient Descent variant, but instead of using the entire dataset to compute gradients, SGD updates weights based on one randomly chosen data point or a small batch at a time [18].

$$W_{\text{new}} = W_{\text{old}} - \eta \cdot \nabla L(W, x_i)$$

- W_{new} is the new values of the model's weights after an update.
- W_{old} is the current set of weights before applying the update.
- η (Learning Rate) is a small positive number that controls how much the weights change in each update.
- $\nabla L(W, x_i)$ measures how much the loss function changes with respect to the weights.

- **Adam (Adaptive Moment Estimation):**

An advanced technique that effectively trains complex models by combining the benefits of both SGD and adjustable learning rates [18].

2.2.1.3.2 Regularization techniques

Regularization techniques in deep learning are techniques used to prevent overfitting by adding constraints or modifications to the model while training. They ensure that the model generalizes well to new, unseen data instead of memorizing the training data.

- **Dropout:** some neurons are randomly dropped or "dropped out" temporarily for each iteration. It avoids overfitting by allowing more powerful learning and less reliance on individual neurons [21][22].

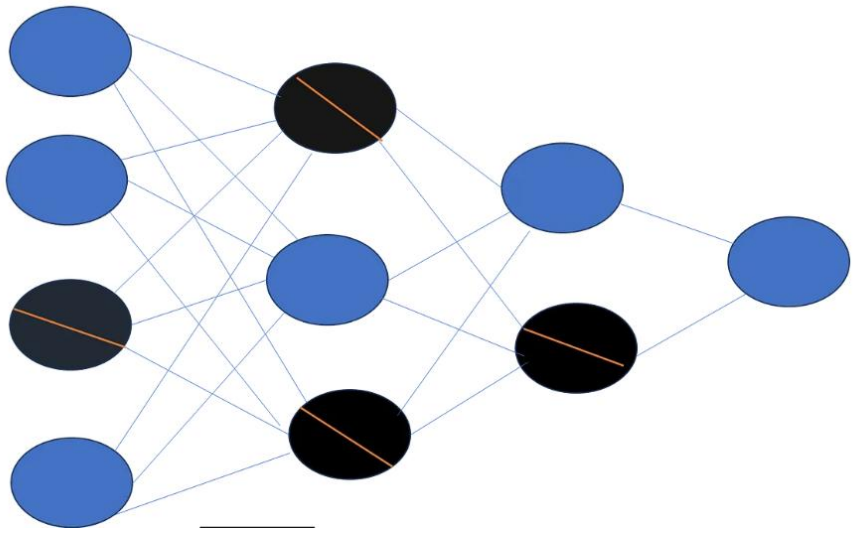


Figure 2.11 – Dropout regularization technique.

This image illustrates the dropout regularization technique in a neural network, where some neurons (shown in black) are randomly "dropped out" or deactivated during training.

- **Batch normalization:** uses normalization of inputs on every layer in a mini-batch by subtraction of the mean and division by standard deviation. Normalization enables fixing issues such as internal covariate shift so that the input to every layer gets centered as well as keeps a similar magnitude while being trained [22].

2.2.1.4 Applications of Deep Learning in medical imaging

Medical imaging has the ability to provide doctors with a lot of information to come up with the most precise diagnosis possible. However, current medical imaging diagnosis is primarily reliant on manual interpretation, which increases the workload of physicians and leads to wrong decisions. Computer-aided diagnosis has been shown to be a reliable method to reduce the workload of physicians and reduce the time required to evaluate medical images, especially deep learning [23].

One of the most popular applications of deep learning in medical imaging is classification. AI models can automatically classify medical images into different groups, for example, distinguishing between abnormal and normal tissue in MRI scans or classifying tumor types, such as in our case, brain tumor classification, with large datasets where models pick up patterns that are not easily visible to the naked eye. In addition to classification, deep learning is also pivotal in image segmentation, where techniques such as U-Net and Mask R-CNN allow for accurate recognition and identification of structures in medical images. For example, in our topic of brain tumor detection, segmentation models can identify the tumor area in MRI scans so that doctors can assess its size, shape, and growth. The application of deep learning technology in medical imaging continues to enhance diagnostic

accuracy, automate doctors' clinical work, and accelerate medical research, thus emerging as a key device in modern medicine.

2.2.1.5 Some famous convolutional networks

Convolutional neural networks have revolutionized the field of computer vision, enabling machines to achieve human-like performance in image classification, object detection, and segmentation tasks. Various convolutional neural network architectures have been developed over the years, becoming standards in the field of deep learning. We will briefly review some of them.

- **LeNet:** developed in 1998, is the 'Hello World' in the field of convolutional neural networks. One of the first CNN architectures, composed of convolutional and pooling layers, mainly used for handwritten digit recognition [24].

- **AlexNet:** Is a classic convolutional neural network architecture. It consists of convolutions, max pooling, and dense layers as its basic building blocks. It was the first architecture that used a graphics processing unit (GPU) to boost performance [25].

- **VGG-NET:** Created by the Visual Engineering Group (VGG) at Oxford University in 2014. It is a simple yet powerful architecture that applies small 3x3 convolutional filters and very deep networks (VGG-16, VGG-19) and has gained widespread popularity in image classification and transfer learning, due to its outstanding performance on large-scale image datasets such as ImageNet [26].

- **DenseNet:** It consists of dense connections between layers, with each layer directly connected to the next. To maintain feedforward functionality, each layer takes additional inputs from all of its previous layers and then passes its feature maps to all of its subsequent layers [27].

- **EfficientNet:** derived from a method called "compound scaling" that is an approach to solving the age-old trade-off between model size, accuracy, and computational cost. The idea behind compound scaling is to scale three basic dimensions of a neural network: width, depth, and resolution [28].

2.2.2 Image classification and segmentation

Image segmentation and classification are two essential components of computer vision and artificial intelligence for interpreting visual information. We use these methods in many applications, including brain tumors, as in our project. It is the process of classifying an entire image into a predefined class, enabling models to distinguish between different objects or patterns. This is accomplished using deep learning models, such as convolutional neural networks, which have significantly improved classification accuracy. On the other hand, image segmentation is an advanced stage of classification, dividing the image into objects or regions of interest. This is particularly useful in medical imaging, where accurate identification of abnormalities, such as tumors in MRI images, is

critical for diagnosis and treatment planning. Many segmentation networks enable accurate pixel-level analysis, making them essential for applications requiring high spatial resolution.

2.2.2.1 Classification

Among computer vision tasks is image classification, the main goal of which is to accurately assign input images to predefined classes. This requires image analysis, where convolutional neural networks automatically assign raw image pixels to class labels. Classification is used in tasks such as face recognition to identify individuals from facial images, and disease diagnosis in medical imaging [29].

2.2.2.1.1 Types of image classification

There are different types of image classification methodologies to be employed including [30]:

- **Binary classification:** In order to categorize unknown data points into two groups, binary classification uses an "either-or" logic. Binary classification provides yes-or-no solutions for a wide range of situations, including identifying benign and malignant cancers and evaluating product quality to identify faults.

- **Multiclass classification:** While binary classification is used to distinguish between two classes of objects, multiclass classification, as the name suggests, classifies objects into three or more classes. It is very useful in many fields, such as in this project, where we will use it to classify tumors into multiple classes.

- **Multilabel classification:** Unlike multiclass classification, where each image is assigned to exactly one class, multilabel classification allows the item to be assigned to multiple labels.

- **Hierarchical classification:** Is the task of organizing classes into a hierarchical structure based on their similarities, where a higher-level class represents broader categories and a lower-level class is more concrete and specific.

2.2.2.2 Classification models for brain tumors

Brain tumor classification uses deep learning techniques, specifically convolutional neural networks, to distinguish between tumor types from MRI images. Some of the most common architectures include ResNet, VGG-19, Xception, and InceptionV3, which are often used in ensemble learning to improve accuracy. They classify tumors into types such as glioma, meningioma, pituitary tumor, or no tumor.

2.2.2.2.1 Networks used for classification

- **ResNet:** When neural networks become deeper, they are plagued by the vanishing gradient problem, in which gradients are too small while backpropagating, so it becomes difficult to train.

The traditional deep networks do not learn well when layers are increased, leading to accuracy degradation. ResNet which is one of the common architectures of CNN solves this issue [31].

1. **Definition:** Residual networks, or "ResNets," resemble networks with convolution, pooling, activation, and fully-connected layers almost exactly. Convolutional and identity blocks, which link the output of one layer with the input of a previous layer (skip connection), are the fundamental building blocks of ResNets [31].

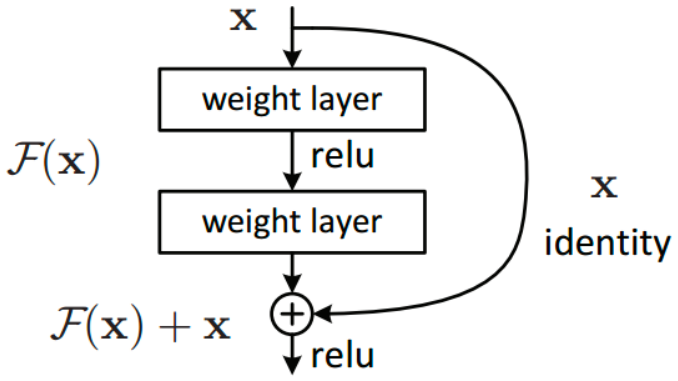


Figure 2.12 – Residual block.

2. **Residual block:** A key element of ResNet, which is intended to address the vanishing gradient issue in deep neural networks, is a residual block. It presents skip connections, which enable a layer's input to be added straight to the output without passing through one or more layers. a residual block compute: $Y = F(X) + X$

   X: represents the input feature map to the residual block**.**

   $F(X)$: represents the series of operations applied to inside the residual block, typically includes convolution layers, batch normalization, and activation functions (ReLU).

   Y: The final output after summing X and $F(X)$.

The Res-Net architecture, have different variations despite originating from the same source. The only difference between these architectures is the number of layers: ResNet-50, Resnet-101, Resnet-110… [31].

3. **ResNet-50:** composed of 50 layers, structured into five stages. Each ResNet design uses 7×7 and 3×3 kernel sizes for initial convolution and max-pooling, respectively. There are three convolution layers in each convolution block and three convolution layers in each identity block. A fully connected layer with 1000 neurons (ImageNet class output) follows an Average Pooling layer [32].

- **VGG-NET:** Is a Convolutional Neural Network based on AlexNet, developed by the Visual Geometry Group at Oxford University. It processes RGB images of size 224×224 pixels and employs small 3×3 convolutional filters with a stride of 1, along with 2×2 max-pooling layers. The network includes three fully connected layers and comes in two main variants: VGG-16 and VGG-19 [33].

  1. **VGG-19:** As we can see in the picture below it contains: 19 layers (16 convolutional layers + 3 Fully connected layers), the architecture is easier to comprehend and apply since it adheres to a simple, repeating pattern [34].

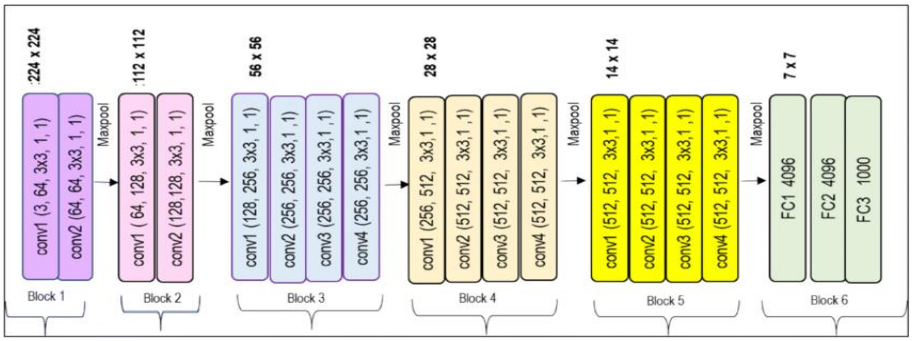


Figure 2.13 – VGG-19 architecture.

The key components of the VGG-19 architecture are:

Convolutional Layers: 3x3 filters with a stride of 1.

Activation Function: ReLU applied after each convolutional layer to introduce non-linearity.

Pooling Layers: max pooling with a 2x2 filter and a stride of 2 to reduce the spatial dimensions.

Fully Connected Layers: 3 layers + SoftMax.

- **Inception:** Is a convolutional neural network introduced by GoogleNet to improve computational efficiency and accuracy by using parallel convolutional filters of different sizes within the same layer, and also solves the overfitting problem [35].

The most famous versions of this architecture are: Inception v1, inception v2, inception v3, inception v4.

  1. **Inception module:** The Inception module is the fundamental building block in the Inception network. It employs multiple filters (1×1, 3×3, and 5×5) in parallel to enable the network to capture spatial features at various scales. It also includes an additional 1×1 convolution to compress the feature map, lowering the efficiency cost of the model [36].

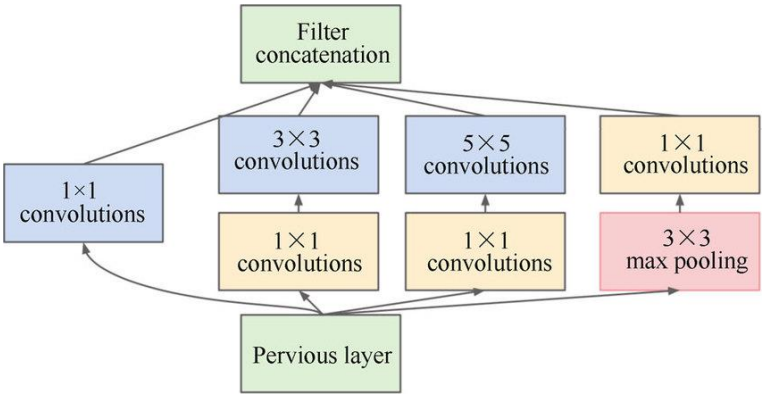


Figure 2.14 – Inception module with dimension reduction.

2. **Inception v3:** The architecture of an Inception v3 network is progressively built, step-by-step, as explained below:

   Factorized Convolutions: reduce the computational efficiency as it reduces the number of parameters involved in a network. It also keeps a check on the network efficiency.

   Smaller convolutions: replacing big convolutions with little convolutions makes training faster. 5 × 5 filter has 25 parameters but 3 × 3 filters replacing a 5 × 5 convolution has just 18 parameters [36].

   Asymmetric convolutions: in this one, a 3×3 convolution is replaced with a 3×1 followed by a 1×3 convolution to improve efficiency.

   Auxiliary classifier: an auxiliary classifier is a small CNN inserted in the middle of layers during training, and the loss incurred is added to the main network loss. it applied as regularizer.

   Grid size reduction: for smoother down sampling, it uses convolutions and pooling layers in addition to max-pooling.

- **Xception:** It is an extension of the Inception architecture based entirely on depthwise separable convolution layers, and it use residual connections to help gradient flow and stabilize deep networks.

Since this architecture is more powerful than Inception, we call it Xception, which stands for "Extreme Inception" [37].

1. **Original Depthwise Separable Convolution:** The original depthwise separable convolution is the depthwise convolution followed by a pointwise convolution.

2. **Depthwise Convolution:** Deep convolution is an n×n spatial convolution for each channel. Let's assume in the figure-14 that we have 5 channels, so we will have 5 n×n spatial convolutions.

3. **Pointwise convolution:** After the depthwise convolution, a pointwise convolution is applied. This is a 1×1 filter that combines the output of the depthwise convolution into a single feature map.

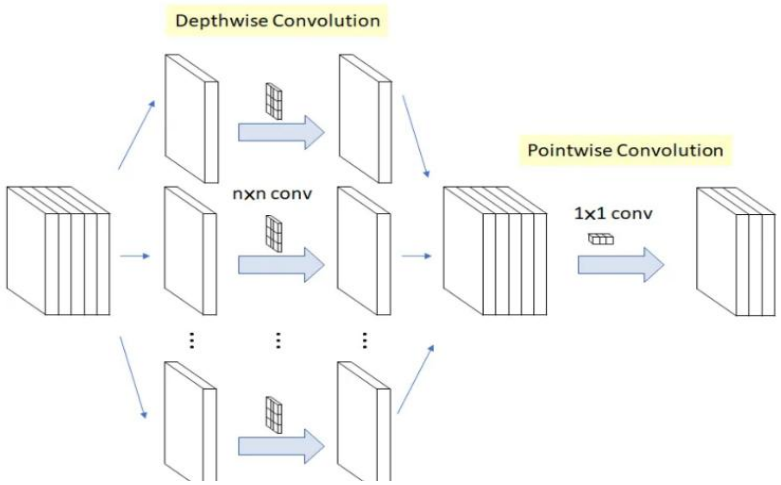


Figure 2.15 – Original Depthwise Separable Convolution.

The Xception model consists of three main parts:

- **Entry flow:** responsible for extracting low-level features from the input image, composed of a sequence of convolutional layers followed by depthwise separable convolutions.

- **Middle flow:** the core of the Xception architecture is where most of the computations are performed, and consists of eight repeated depthwise separable convolution blocks.

- **Exit flow:** is responsible for toggling between feature extraction and classification. It also compresses the spatial dimensions and deepens the feature maps.

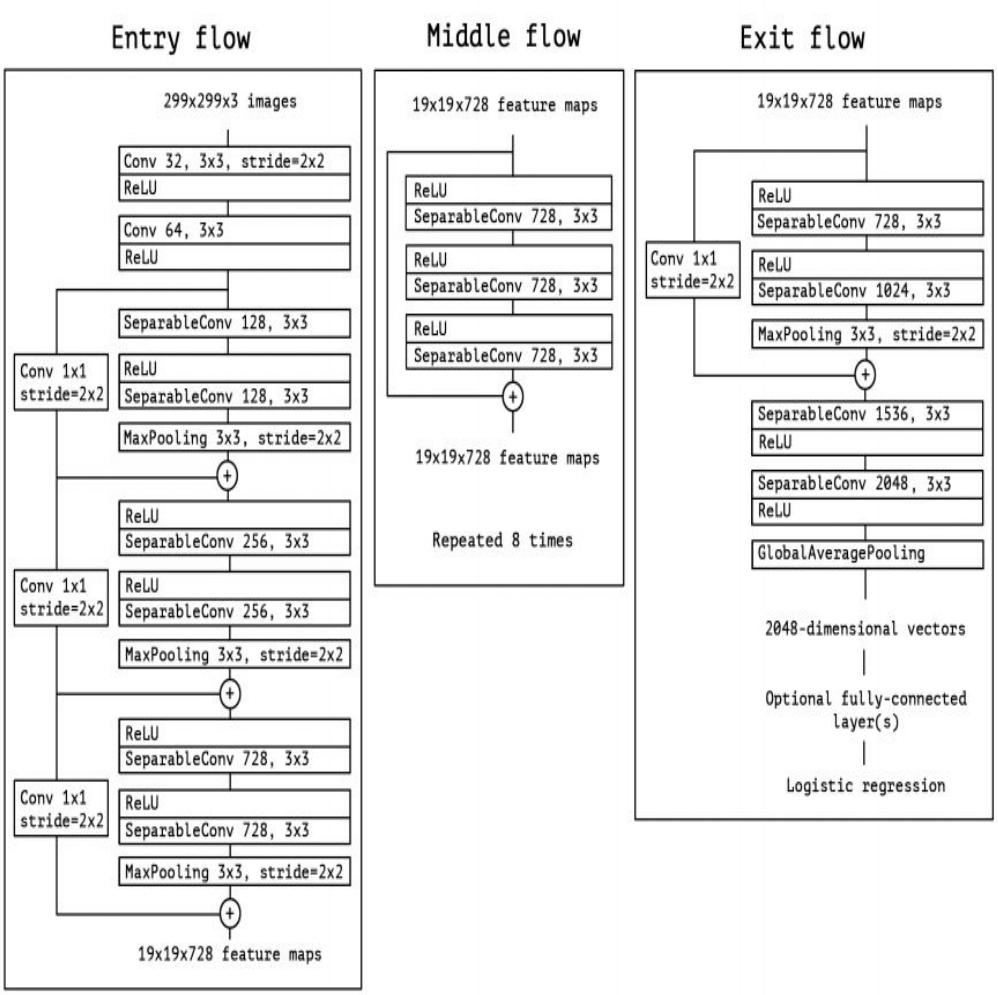


Figure 2.16 – Xception architecture.

2.2.2.2.2 Comparison of architectures

Feature	ResNet-50	VGG-19	InceptionV3	Xception
Year Introduced	2015	2014	2016	2017
Developed By	Microsoft Research	Visual Geometry Group (VGG)	Google Brain	Google Brain
Number of Layers	50	19	48	71

Feature	ResNet-50	VGG-19	InceptionV3	Xception
Number of Parameters	~25.6M	~143.7M	~23.8M	~22.9M
Input Image Size	224 × 224	224 × 224	299 × 299	299 × 299
Architecture Type	Deep Residual Network	Traditional CNN	Inception Module	Depthwise Separable CNN
Key Innovation	Skip (Residual) Connections	Deep stack of 3x3 convolutions	Factorized convolutions, 1x1 convs	Depthwise Separable Convolutions
Training Speed	Faster due to residual learning	Slow due to large number of parameters	Faster than VGG-19 but complex	Faster than InceptionV3
Regularization Techniques	Batch Normalization, Dropout	Dropout	Label Smoothing, Batch Normalization	Batch Normalization
Inference Time	Fast	Slow	Fast	Faster than InceptionV3
Disadvantages	More complex than traditional CNNs	Very large model, slow inference	Complex architecture, requires optimization	Higher computational cost than ResNet

Table 2.1 – Comparison of architectures

2.2.2.3 Segmentation

Image segmentation is a fundamental component of many visual perception systems. It involves dividing images (or video frames) into multiple parts or objects and labeling each part. This occurs at the pixel level to define the precise outlines of the object within its frame and class. These outlines, also known as outputs, are characterized by one or more colors, depending on the type of segmentation. They play a pivotal role in a wide range of applications, including medical image analysis (like extracting tumor boundaries and measuring tissue volume), autonomous vehicles (e.g., detecting surfaces and pedestrians), and video surveillance. Deep learning (DL) networks have produced a new generation of image segmentation models, with significant performance improvements—often achieving the highest accuracy rates on common performance benchmarks, leading to what many consider a paradigm shift in the field. Segmentation is divided into three main types, which we will discuss in the next section [38][39].

2.2.2.3.1 Types of Segmentation

- **Semantic segmentation:** In a task of semantic segmentation, segmentation masks are actually fully labeled images. It is an indication that each pixel of the image should belong to some category, whether or not they belong to the same instance. however, each pixel of the same category will be one segment. If two pixels are labeled as "people" then pixel values in the segmentation mask for both of them will be the same [39].

  As we can see in the image below, all teeth are the same green color, and the upper and lower jaw bones are the same yellow color.

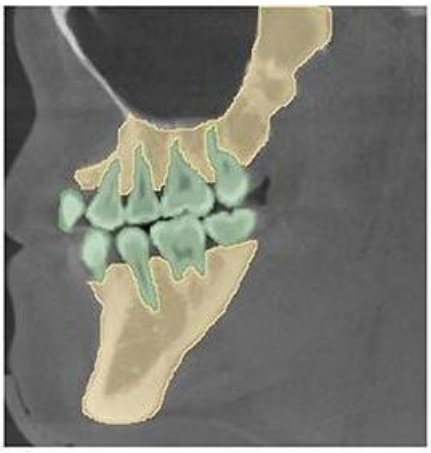


Figure 2.17 – Semantic segmentation.

- **Instance segmentation:** Instance segmentation is one of the most important and challenging tasks. It aims to predict class labels and instance masks for each pixel to locate varying numbers of instances in images, where each new object is classified as a different instance, even within the same class [40].

  As we can see in the image below, each tooth and bone are marked with a different color, even if they belong to the same category.

Figure 2.18 – Instance segmentation.

- **Panoptic segmentation:** Is a computer vision task that combines semantic segmentation and instance segmentation to provide an end-to-end comprehension of the scene. Panoptic segmentation seeks to segment the image into semantically meaningful parts or regions and detect and classify individual instances of objects in the regions [41].

2.2.2.4 Segmentation models for brain tumors

Medical imaging can provide a wealth of information to help doctors make the best possible diagnosis. However, current medical imaging diagnostics rely primarily on manual interpretation, which increases the workload of physicians and leads to incorrect assessments. Computer-aided diagnosis has proven to be a reliable tool for reducing the burden on physicians and shortening the time it takes to evaluate medical images. Among these areas, automatic segmentation of medical images is an important element of our time and can be used in many areas of medicine, such as brain tumors, where it helps pinpoint the exact location of a tumor, even when it is not clearly visible. This helps doctors identify and diagnose quickly [42].

2.2.2.4.1 Some semantic segmentation architecture

1. U-NET architecture:

The U-Net is one of the most popular segmentation architectures, primarily designed for image segmentation tasks. It is widely used in medical image segmentation using convolutional neural networks (CNNs). It features an encoder-decoder architecture with skip connections that allow combining low-level details from the encoder with high-level semantic features from the decoder. The shrinkage encoder pipeline consists of recurrent convolution layers, batch normalization layers, and max-pooling layers to extract abstract representations. The expanded decoder pipeline consists of shifted convolution layers and upsampling layers to restore the original resolution. Skip connections connect the encoder and decoder features at each level, preventing the loss of spatial information and enabling accurate boundary identification. The network starts with 64 filters in the first layer, and

the number gradually increases in deeper layers to enhance hierarchical learning. After four decoder layers, the output passes through a final fully connected convolution layer, where the kernel size is adjusted based on the number of classes in the mask to meet the needs of a specific task. The output is then validated using a function that varies with the number of tags, ensuring customized results for each task [43].

Below is the architecture of U-Net:

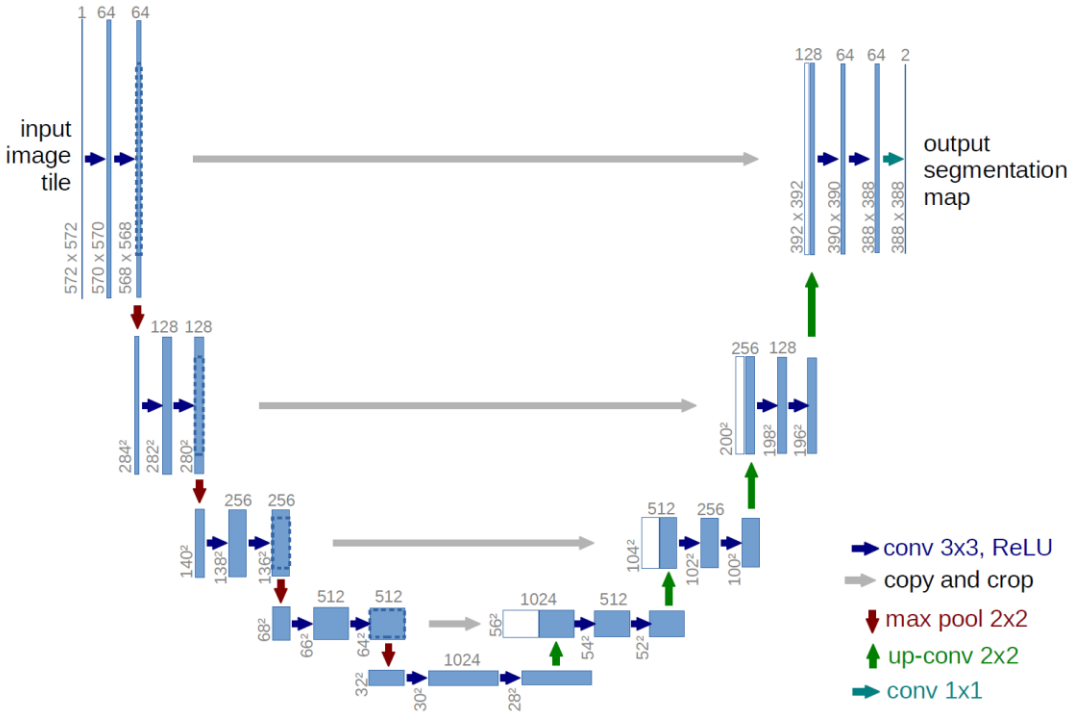


Figure 2.19—The architecture for U-NET.

- **Encoder:** intends to capture important information about the image while simultaneously reducing the image's spatial size [44].
  - Consists of multiple convolutional blocks.
  - Each block has two convolutional layers.
  - Uses ReLU activation after each convolution.
  - Followed by MaxPooling (2×2) to reduce spatial dimensions by half.
- **Decoder:** aims to upsample the feature map and produce a relevant segmentation map using the patterns learnt in the contracting path [44].
  - Each upsampling step consists of Up-convolution.

- Followed by two convolutional layers (3×3, ReLU).
- Skip connections concatenate encoder features of the same resolution.

- **Skip connections:** A key feature of U-Net is the skip connections, which link encoder layers directly to decoder layers of the same resolution [44].
  - Helps recover spatial details lost during max pooling.
  - Prevents the vanishing gradient problem in deep networks.
  - Enables better localization of segmented objects.

2. Res-Unet:

It is a semantic segmentation neural network that combines the strengths of both U-Net and residual neural networks. This combination achieves two benefits: first, the residual unit makes training the network easier, and second, the skip connections within the residual unit and between the low and high levels of the network facilitate information propagation without degradation. This enables a neural network to be designed with much fewer parameters, potentially achieving significantly better performance in semantic segmentation. The Res-Unet architecture uses seven levels to extract the path space and consists of three parts: the encoder, the concatenation, and the decoder. The first part encodes the input image into compact representations, while the last part reconfigures the representations into a pixel-wise classification, i.e., semantic segmentation. The middle part acts as a bridge between the encoder and decoder paths. All three parts are built with residual units consisting of two 3×3 convolution blocks and an identity mapping layer. Each convolution block includes a BN layer, a ReLU activation layer, and a convolution layer. The identity mapping connects the unit's inputs and outputs [45].

The encoder path contains three residual units. In each block, instead of using pooling to reduce the feature map size, a two-step process is applied to the first convolution block to reduce the feature map by half. In contrast, the decoding path also consists of three remaining blocks. Before each block, the feature maps from the lower level are up sampled and concatenated with the feature maps from the corresponding encoding path. After the last level of the decoding path, a 1×1 convolution layer and a sigmoid activation layer are used to project the multi-channel feature maps into the desired segmentation [45].

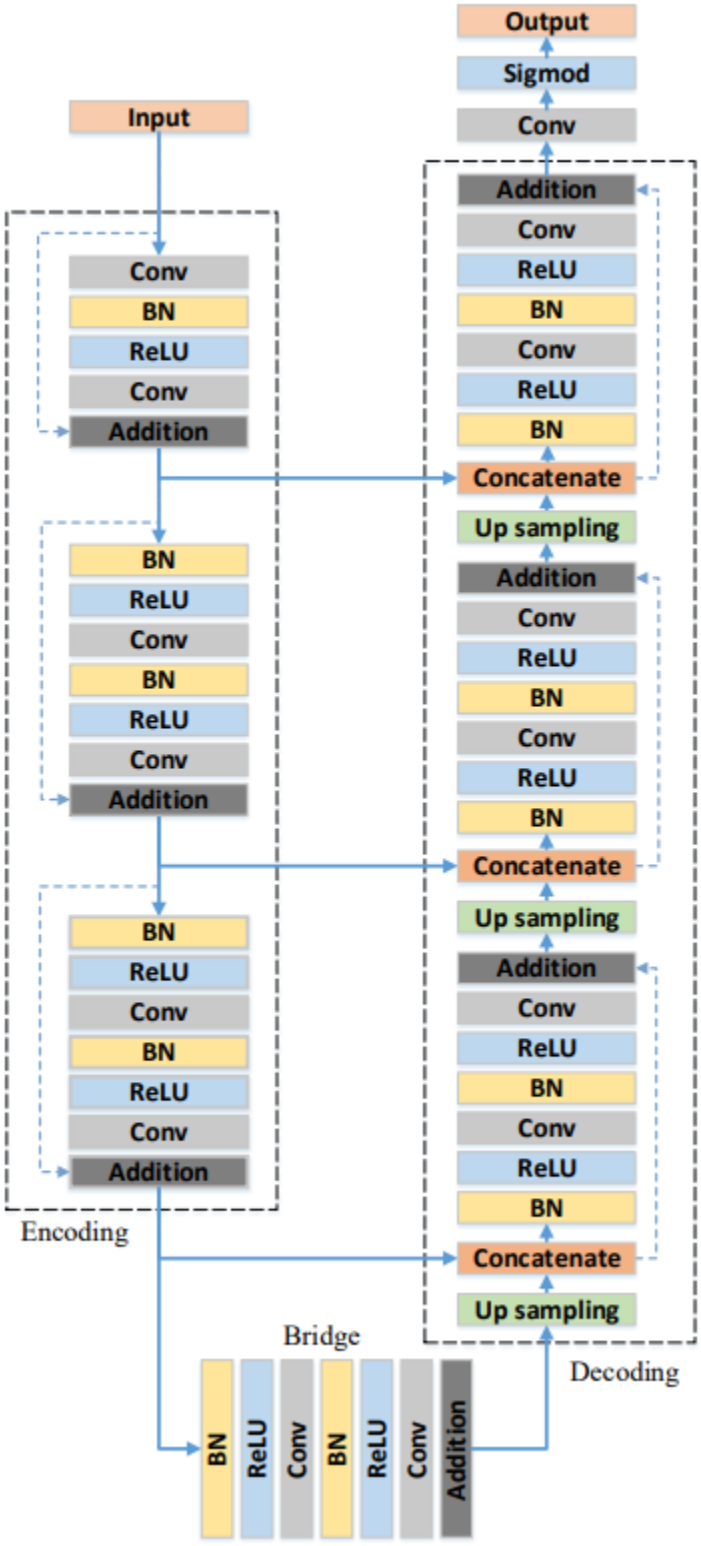


Figure 2.20—Res-Unet architecture.

3. Attention U-Net:

Attention, in image segmentation, is a technique used to highlight only the relevant activations during training. This reduces wasted computation on irrelevant activations, allowing the network greater generalization capacity. Essentially, the network is capable of applying "attention" to certain regions of the image [46].

Attention comes in two forms:

- **Hard attention:** Hard attention works on the basis of highlighting relevant regions by cropping the image or proposing a repeating region. Because fine-tuning can only select one region of an image at a time, it has two consequences: it is non-differentiable and requires reinforcement learning for training. Because it is non-differentiable, it means that for a given region of the image, the network can either "pay attention" or "pay no attention," without any intermediate steps. As a result, standard backpropagation cannot be performed. Because accuracy depends on how well the samples are taken, other techniques such as reinforcement learning are necessary to make the model effective [46].

- **Soft attention:** works by weighting different parts of the image. Areas of high relevance is multiplied with a larger weight and areas of low relevance is tagged with smaller weights. As the model is trained, more focus is given to the regions with higher weights. Unlike hard attention, these weights can be applied to many patches in the image.

  Due to the deterministic nature of soft attention, it remains differentiable and can be trained with standard backpropagation. As the model is trained, the weighting is also trained such that the model gets better at deciding which parts to pay attention [46].

4. U-net++:

Named also Nested U-Net, is a deep learning architecture introduced in 2019 with in U-Net++. In U-Net, the encoder captures high-level features from the input image through a series of convolution and pooling layers, while the decoder refines these features to create a dense segmentation map. However, a semantic gap may exist between the encoder and decoder features, meaning the decoder may struggle to reconstruct fine details and produce an accurate segmentation [47].

U-Net++ (Nested U-Net) is an enhanced version of U-Net designed to improve segmentation accuracy by providing dense skip connections and deep supervision to bridge this semantic gap. It adds additional skip connections between the encoder and decoder blocks at multiple resolutions. These connections allow the decoder to access and combine low- and high-level features from the encoder, providing a more detailed and comprehensive understanding of the image.

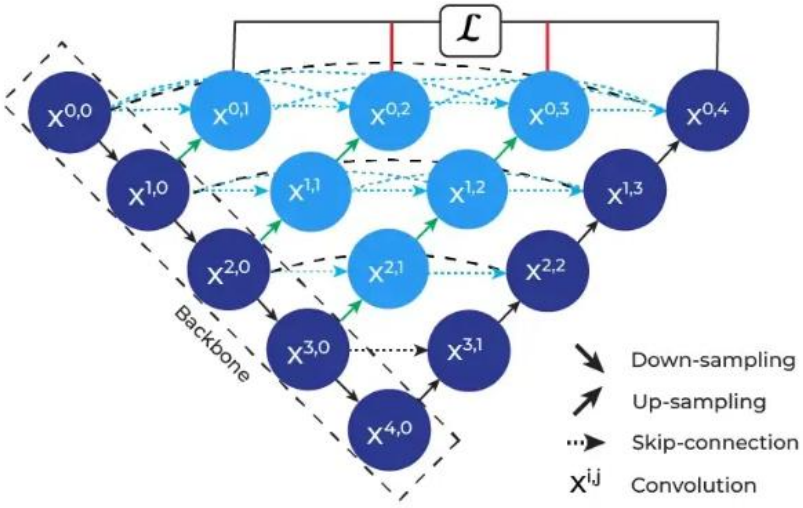


Figure 2.21—U-net++ architecture.

The figure 2.21 illustrates the design of the U-Net++ architecture. We see the nested encoder-decoder architecture of the U-Net++ architecture. Instead of a traditional skip connection, the lower-level feature map is concatenated with the higher-level feature, and the new merged feature data is passed through.

The black dotted skip connection indicates the original skip connection in the U-Net architecture, while the blue dotted skip connection indicates the new nested skip connection. Before concatenating the lower-level feature map, its precision is increased to match the number of channels at that level. The figure also shows how deep supervision is applied to the outputs of nodes X0,1, X0,2, X0,3, and X0,4 to improve model learning during training. The model is optimized on the outputs of nodes X0,1, X0,2, X0,3, and X0,4 by calculating the total loss on the predicted outputs based on the outputs of each of these nodes [47].

2.2.2.4.2 Segmentation evaluation methods

1. **Dice coefficient:** The Dice Coefficient is a measure of similarity of two sets and is often applied on segmentation problems to compare the predicted with the ground truth masks. It can be mathematically expressed as:

$$Dice\ coeff = \frac{2 \mid A \cap B \mid}{\mid A \mid + \mid B \mid}$$

- A : Predicted segmentation mask.

- B : Ground truth segmentation mask.

- $\mid A \cap B \mid$: Number of overlapping pixels between prediction and ground truth.

-$\mid A \mid + \mid B \mid$: Total number of pixels in both masks.

Where **1** means perfect and **0** means no.

2. **Dice Loss:** The Dice loss is derived from the Dice coefficient and is used as a loss function in segmentation models to maximize overlap between predicted and actual regions.

   It is defined as:

$$L_{Dice}(P, G) = 1 - \frac{2\sum_{i,j} p_{ij}\, g_{ij} + \epsilon}{\left(\sum_{i,j} p_{ij}^2\right) + \left(\sum_{i,j} g_{ij}^2\right) + \epsilon}$$

   • P are our predictions.

   • G is the ground truth.

   • In practice each g_i will either be 0 or 1.

   • ϵ is a small number that is added to avoid division by zero.
   The Dice loss ranges between:

   • 0: perfectly matching the ground truth distribution g.

   • 1: complete mismatch with the ground truth.

3. **Intersection over Union (IoU):** measures the overlap between predicted and ground truth masks relative to their union.

$$IoU = \frac{|\, A \cap B \,|}{|\, A \cup B \,|}$$

   $|\, A \cap B \,|$: Intersection (common pixels between prediction and truth).

   $|\, A \cup B \,|$: Union (total pixels in both prediction and truth).

Conclusion

In this chapter, we discussed some key points, took a comprehensive look at what MRI is, and discussed various modern techniques that enable us to detect and segment brain tumors.

Now that we have an understanding of the above, we can move on to the practical side of our project, which we will cover in the next chapter.

Chapter 3

Conception and realization

Introduction

Developing an effective brain tumor detection system requires a well-designed architectural framework that integrates data preprocessing, model selection, and implementation strategies. This chapter summarizes the composition and layout of our proposed method and describes the key components involved in the classification and segmentation processes.

We begin with the general structure of our model, followed by the preprocessing techniques applied to MRI images, including normalization, rescaling, and data augmentation. We continue by discussing the reasons for choosing specific deep learning architectures for classification and segmentation, and their advantages in medical image analysis. We then discuss the dataset structure, its origin, and how to partition the data into training, validation, and testing datasets.

This chapter also discusses the tools and development environment, including the libraries and hardware configurations used. Finally, we demonstrate how to implement the classification and segmentation models, comparing and analyzing their performance metrics. We conclude our chapter with a discussion of the front-end of our website. This is done to facilitate and illustrate the results and how the system performs in real-world scenarios.

3.1 Presentation of the overall architecture of the model

The first step in our project was to download the dataset from Kaggle, which contained MRI images classified into four categories: glioma, meningioma, pituitary tumor, and no tumor. The first step was to preprocess the dataset, which involved images pre-processing. We started by cropping and resizing all images to ensure consistent input dimensions. We normalized the images using scaling to approximate pixel values to a standard range, which improved the efficiency of model training. Since the number of images in each category was unbalanced, we applied data augmentation techniques to balance the dataset and improve model generalization. Next, we split the dataset into three subsets: training, validation, and testing, to properly train the model and evaluate its performance.

The model consists of three main components: classification, segmentation, and feature extraction.

1. **Classification:** When an MRI image enters the model, it is first processed by the classification module, which utilizes an ensemble learning approach combining Xception, InceptionV3, ResNet-50, and VGG-19. If the classifier determines that there is no tumor,

the process stops immediately, as no further analysis is needed. However, if a tumor is detected, the model classifies it into the correct category (glioma, meningioma, or pituitary tumor).

2. **Segmentation:** If a tumor is detected, the image moves to the segmentation module, where a ResUNet model is used to segment the exact region of the tumor. This module generates two output images:

   The first image is the original MRI scan overlaid with a colored segmentation mask (green), highlighting the tumor region.

   The second image is a binary mask (black and white), where the tumor appears in white, and the background remains black.

3. **Feature Extraction**: After segmentation, the model proceeds to the final step: feature extraction. In this stage, important morphological characteristics of the tumor are calculated, including:

   - Surface area (size of the tumor region)

   - Perimeter (boundary length of the tumor)

   - Width and height (dimensions of the tumor)

   - Circularity (shape analysis to determine if the tumor is more elongated or rounded)

The image below shows the project architecture:

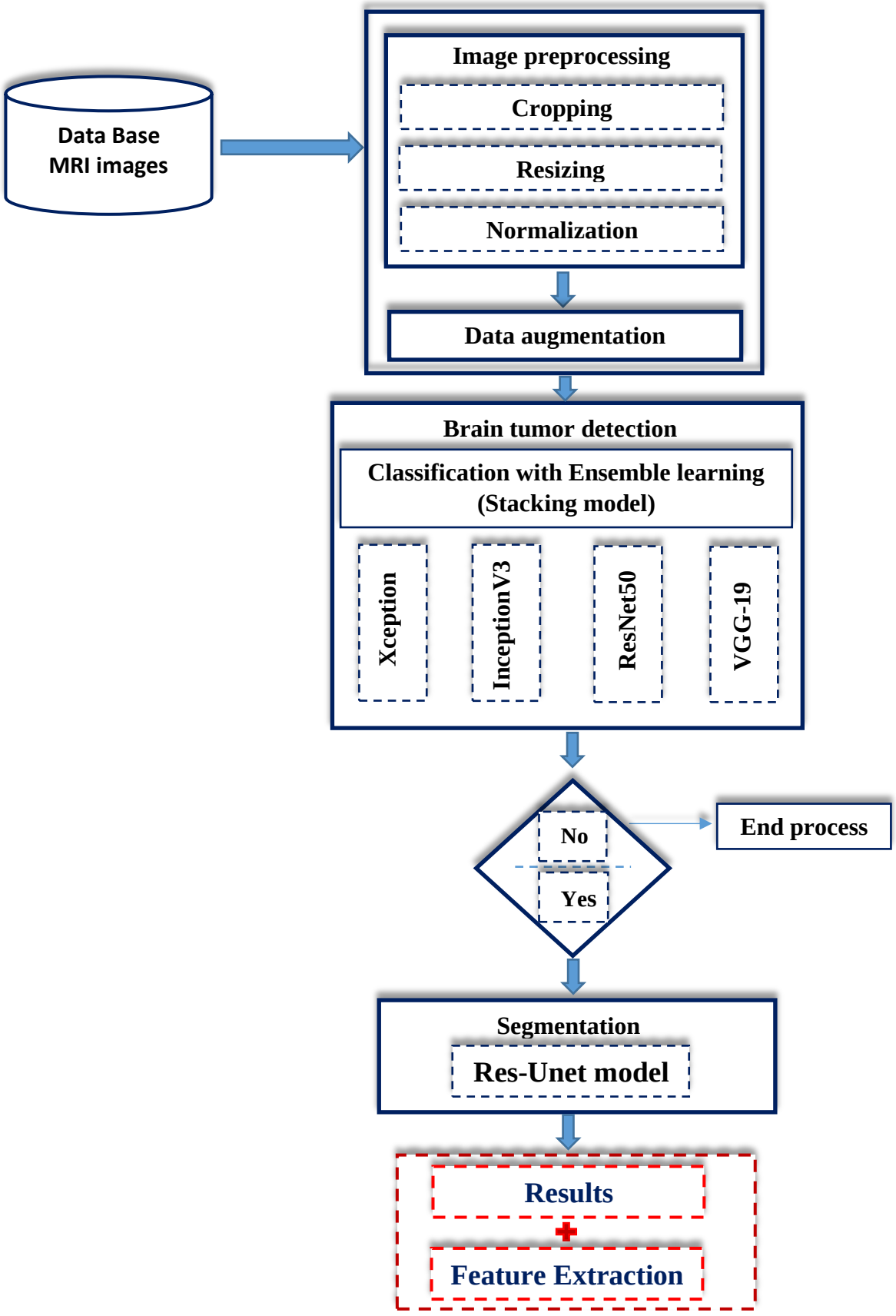


Figure 3.1— Architecture of the proposed model.

3.2 Image preprocessing

Image preprocessing is a fundamental step in image processing for deep learning. It transforms raw image data into a format that is easier to analyze before feeding it into a neural network. It addresses various distortions and improves key image properties, such as contrast, resolution, and noise levels, making it more suitable for training and improving model performance. Preprocessing ensures consistent image format, size, and quality, helping neural networks learn more effectively. Some of the most common image preprocessing operations we implemented in my project include [48]:

3.2.1 Image cropping

Is a preprocessing technique that involves the removal of unnecessary outer areas of an image leaving the most significant area. In deep learning, image cropping is typically used to focus on the region of interest (ROI), remove unnecessary background noise. Cropping aims to improve computational efficiency by reducing the data load that must be processed by a model and also can further increase the accuracy of models by limiting analysis to only the important elements of an image [49].

As in our project, we cropped all the images in the dataset because they contain excess, inconsequential black areas surrounded by the brain. The images are shown below:

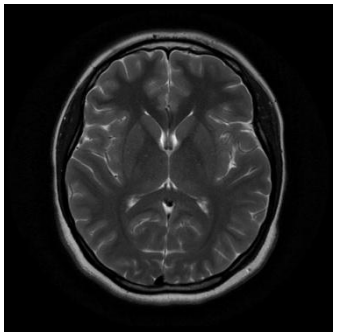


Figure 3.2—Original image.

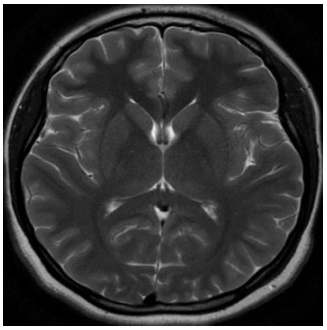


Figure 3.3—Cropped image.

3.2.2 Image Resizing

Resizing images refers to changing its size but retaining its basic characteristics. Resizing comes into play for image classification when applying deep learning, as the models use different sizes as their input. For example, in our project we used ensemble learning in classification, where several architectures such as Xception, InceptionV3, VGG-19, and ResNet-50 are averaged out. Every model uses a specific input size.

Model	Input size
Xception	299 x 299
InceptionV3	299 x 299
VGG-19	224 x 224
ResNet-50	224 x 224

Table 3.2 – The different input size for Xception, Inception, ResNet50, VGG-19.

Since these models anticipate different image sizes, resizing ensures that all images are in the necessary format, enabling consistent feature extraction and classifier performance. This step helps maintain the spatial relationships in the image without causing distortions that can affect model accuracy.

3.2.2 Image Normalization

In deep learning, image normalization is a preprocessing method that improves neural network performance and stability by scaling pixel values to a standard range. Normalization makes ensuring that raw pixel values are mapped to a smaller range, like [0,1] or [-1,1], as they usually vary from 0 to 255. This helps models converge more quickly during training by lowering numerical instability [50][51].

This Figure shows the stages of image preprocessing:

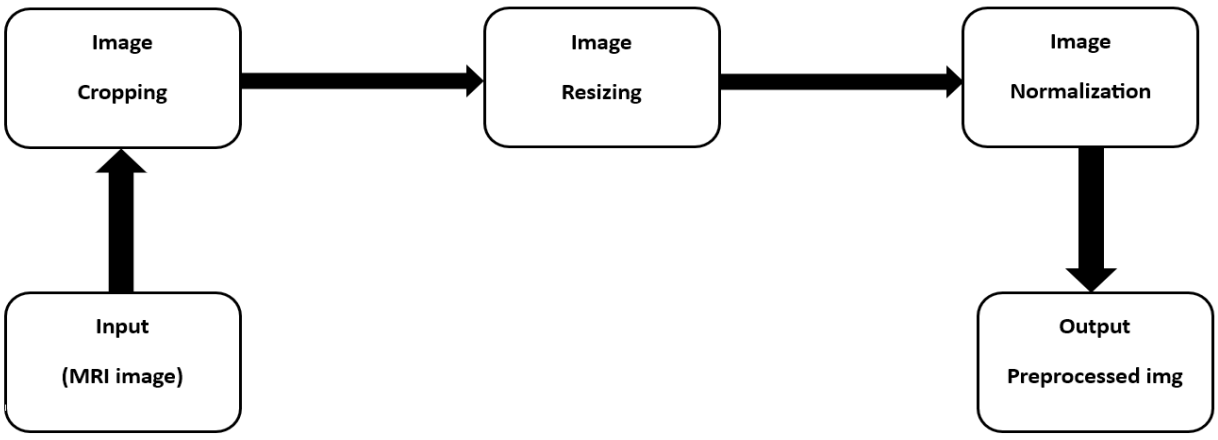


Figure 3.4—Image preprocessing.

3.2.3 Data augmentation

The simplest way to reduce overfitting, generalize the model, and enhance robustness is to expose the model to various variations of the same data, known as data augmentation. This is widely used in training deep CNNs. Data augmentation aims to artificially expand the training dataset from existing data using various transformations, such as shifting, rotating, flipping, clipping, adding noise, etc. [52].

Care must be taken. In the medical application, especially in the identification of brain tumors, data augmentation must be done carefully since MRI scans carry critical anatomical details, and excessive transformation results in distortion of the shape of the tumor and lead to incorrect classifications. Therefore, we have selected some appropriate transformations to apply to the image set within the database with positive effects including:

- **Rotation:** Helps with slight orientation variations (10°).
- **Width & Height shift:** are transformations that move an image horizontally or vertically within a given range, this helps simulate variations in object positioning and makes the model more robust to slight changes in object location (0.1).
- **Zoom:** refers to scaling an image in or out to simulate different levels of magnification (0.1).
- **Brightness adjustment:** modifies the intensity of pixel values to make the image brighter or darker ([0.8, 1.2]).
- **Horizontal flip:** reverses the image left to right, creating a mirrored version.

As we said before, some transformations and their values should be chosen carefully. What should be avoided for example is:

- **Large rotation:** for example, ±90° or more, cause tumors are not randomly oriented; extreme rotations could make the tumor appear in unrealistic locations
- **Strong zooming:** (> 15%), might crop out parts of the tumor, affecting classification and segmentation.
- **Excessive brightness adjustments:** (> ±30%), can alter tumor visibility, making it too bright or too dark, which may confuse the model.

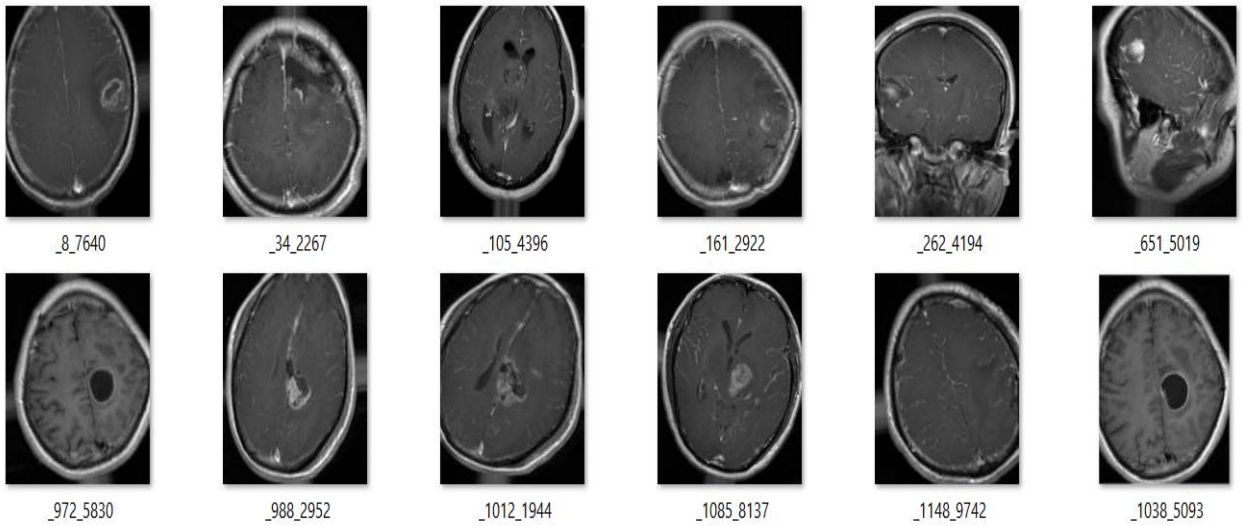


Figure 3.5—Some glioma tumor using data augmentation.

3.3 Choice of models for classification

There are multiple deep learning models for brain tumor classification, but instead of relying on a single model, we used ensemble learning.

3.3.1 Ensemble learning

Ensemble learning is a widely used machine learning technique that combines multiple base learners to form an ensemble learning model to construct a robust and accurate prediction model. A base learner is a single model that can easily suffer from noise, bias, and variance in the data when making predictions. Therefore, ensemble learning is applied to reduce generalization errors and improve classifier performance. The basic idea behind the ensemble learning framework is to assemble base learner classifiers (c1, c2, ...) to predict a single output using a dataset of size n and feature dimension m. The output prediction is calculated based on the ensemble method using weighted mean, maximum weight, or majority voting. Figure 3.6 illustrates the general framework of ensemble learning techniques, which can be visualized and realized through four properties: data sampling, clustering rules, heterogeneity, and voting. First, data sampling refers to the process of dividing a training dataset into subsets to achieve better accuracy and diversity through independent and dependent strategies. The independent sampling strategy involves subsets that are not interdependent and are not affected by the performance of the previous subset. The dependent sampling strategy involves subsets that are dependent on the performance of the previous subset. Therefore, to avoid the difficulty of achieving diversity through data sampling techniques, the optimal size of each subset and the maximum number of samples must be determined. Second, ensemble rules refer to the method of combining two base classifiers using parallel ensemble and sequential ensemble techniques. Parallel ensemble techniques are used to train base classifiers simultaneously without interclassifier dependency or data sampling to increase diversity between base classifiers. A common parallel ensemble algorithm is the bagging algorithm. Alternatively, sequential ensemble techniques are used to train base classifiers sequentially due to the resampling of data. Sequential

methods are used to correct errors made by the previous base classifier in each iteration. A common sequential ensemble algorithm is the boosting algorithm. Third, the heterogeneity characteristics depend on the type of algorithms used for each base classifier in the ensemble process, which can be further classified as homogeneous or heterogeneous. A homogeneous ensemble method consists of a number of base classifiers that use the same algorithms to build the model, while a heterogeneous ensemble method consists of a number of base classifiers that use different algorithms. Finally, voting methods are applied in the final stage of both classification and regression tasks to improve ensemble prediction. The voting methods used in clustering and boosting can be classified into majority voting, average voting, and weighted average voting. First, majority voting, also known as max voting, is the most common ensemble prediction method and is based on the largest number of votes for each labeled class. The average vote is then calculated by dividing the average of the sum of predictions by the total number of predictions. Finally, weighted average voting is based on assigning different weights to each base classifier [53].

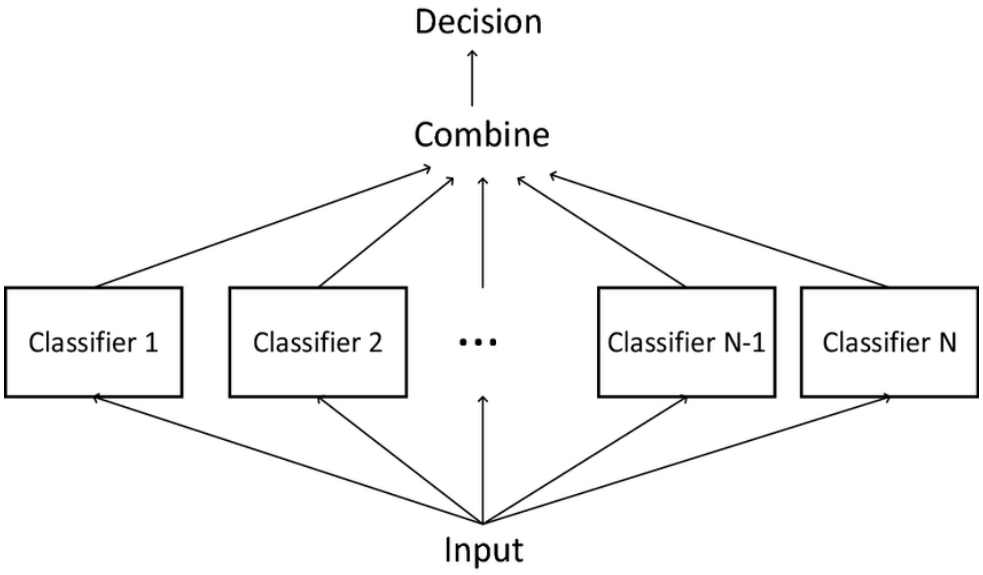


Figure 3.6—General ensemble learning method.

3.3.1.1 Ensemble learning techniques

The three most common techniques for ensemble learning are: bagging, boosting, and stacking.

- **Bagging:** involves training multiple models on different subsets of the training data and combining their predictions to improve performance. In classification tasks, it uses majority voting, where each model votes for a class, and the most frequent class is chosen as the final prediction. In regression tasks, it applies average voting, where the predictions from all models are averaged to obtain the final result. A well-known example of bagging is the Random Forest algorithm [54].

- **Boosting:** Sequentially trains models, where each new model focuses on the errors made by the previous ones. This method aims to correct mistakes and improve overall accuracy, as seen in algorithms like AdaBoost and Gradient Boosting [54].

- **Stacking:** An effective ensemble learning strategy that trains multiple models and then uses another model (meta-model) to combine their outputs. This leverages the strengths of different algorithms to improve performance [54].

3.3.1.1 Reason for use

In our project, we selected four classification models: Xception, InceptionV3, VGG-19, and ResNet-50, based on two main criteria. First, their architectural diversity, each of these models has a unique structure and feature extraction strategy: Xception (Depthwise Separable Convolutions), InceptionV3 (Multi-Scale Convolutions), VGG-19 (Deep Sequential Layers), and ResNet-50 (Residual Connections), allowing them to capture different aspects of the input images. By combining them using ensemble learning, we create a comprehensive and robust classification system, as each model contributes its own strengths and analysis techniques. Second, their proven performance in image classification, as these models are among the most powerful CNN architectures, consistently ranking highly on ImageNet benchmarks.

After selecting these models, we needed to choose the most suitable ensemble learning approach. Boosting prioritizes high accuracy by focusing on difficult samples, but it tends to overfit, which is problematic for medical imaging, where misclassification could have serious consequences. Bagging works best with high-variance models, such as decision trees, by reducing their overfitting through averaging multiple predictions. However, in our case, we are using pre-trained deep learning models, which are already optimized and do not suffer from high variance in the same way. Instead of reducing variance, our goal is to leverage the complementary strengths of different architectures, making stacking a more suitable choice for improving classification performance. By using a meta-learner, stacking ensures that the final prediction is based on the most relevant patterns extracted from each model, making it the ideal approach for brain tumor classification, where it also yielded excellent results.

3.4 Choice of model for segmentation

3.4.1 Reason for use

We used Res-Unet for the segmentation task in our project for two reasons. The first is its superior feature modeling capability, which is essential for accurate brain tumor segmentation. The U-Net model, with its encoder-decoder architecture, is popular in medical image segmentation due to its ability to assimilate spatial information with high accuracy. However, traditional U-Net models are unable to extract deeper features, especially in complex medical images such as MRI scans.

To improve segmentation accuracy, Res-Unet adds residual connections from ResNet to help solve the gradient vanishing problem and enable deeper networks to learn more efficiently. These residual connections enable gradients to flow more easily during training, improving feature propagation and enabling the model to assimilate both low and high-level features. Brain tumor segmentation is a highly sensitive process, requiring precise localization to avoid misclassification. Res-Unet's ability to preserve fine tumor details, as well as capture contextual information, makes it an ideal choice. It has also been used in a number of medical image segmentation tasks,

demonstrating excellent performance when dealing with complex structures, such as tumors with irregular shapes and densities. The second reason is that we trained the database on several segmentation models, including U-Net, Res-Unet, Attention U-Net. However, of these, only Res-Unet gave me good results in terms of accuracy and DiceCoef. Therefore, Res-Unet was the ideal choice for our project, combining U-Net's advantages in localization accuracy with Res-Net's ability to extract deep features, providing effective brain tumor segmentation.

3.5 Dataset structuring

3.5.1 Classification dataset

In this work, we used a brain tumor MRI dataset downloaded from the Kaggle [55] platform. The dataset contains two folders, one for training and one for testing. Each folder contains four categories: glioma, meningioma, pituitary gland, and no tumor. The images are shown below.

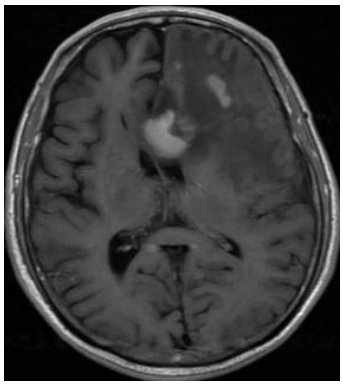


Figure 3.7—Glioma tumor 2.

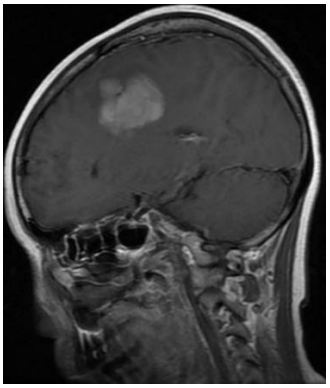


Figure 3.8—Meningioma tumor 2.

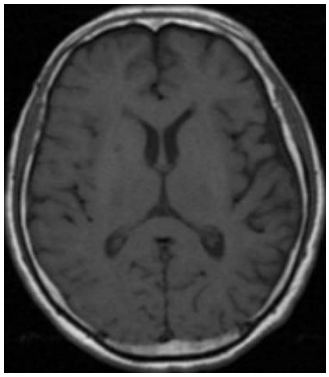


Figure 3.9—No tumor.

Figure 3.10—Pituitary tumor 2.

The database consists of 7028 images, divided into 1307 test images and 5721 training images. Each category has a specific number of images: glioma (1330), meningioma (1339), pituitary gland

(1457), and no tumor (1595). However, this is called an imbalanced dataset, which reduces the efficiency of the classification model. To solve this problem, we applied a process called data augmentation, where we increased the number of images in each category to 2000, bringing the total number of training images to 8000. This method resulted in a balanced dataset.

3.5.1.1 Data splitting

For model training, we need to split the data into:

- **Training data:** refers to the data set that is used to train and evaluate a ML model. Training data consists of a large number of input examples, which include the features as well as the corresponding output or target values [56].

- **Testing data:** is used to evaluate the final performance of a machine learning model. Unlike training and validation data, testing data is only used once the model has been fully trained and optimized. This data helps determine how well the model generalizes to new, unseen examples and provides an unbiased assessment of its accuracy, precision, and overall reliability [57].

- **Validation data:** is a critical part of the machine learning process, used to fine-tune and optimize the model after it has learned from the training data. Unlike training data, validation data is not used to teach the model but to evaluate its performance during the development phase. This helps in adjusting parameters, known as hyperparameters, such as learning rates, to improve model accuracy and prevent overfitting [57].

In the dataset there are two folders, one for training which contains 8000 images, this is data intended for training the model. The other folder is for testing and contains 1307 images. we divided it into two parts: one for testing which contains 654 images, and the other for validation which contains 653 images. In this way, we divided our data into training, testing and validation data.

3.5.2 Segmentation dataset

For the segmentation tasks, we used the "Brain Tumor Segmentation" dataset on Kaggle [58], which contains two folders. The first folder contains 3064 brain tumor images. The second one also contains the same number of images, but contains tumor mask images. This means that each brain tumor image in the first folder corresponds to its mask image in the second folder, respectively.

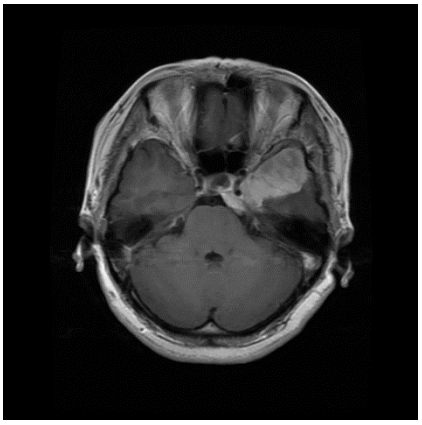


Figure 3.11—Tumor 1.

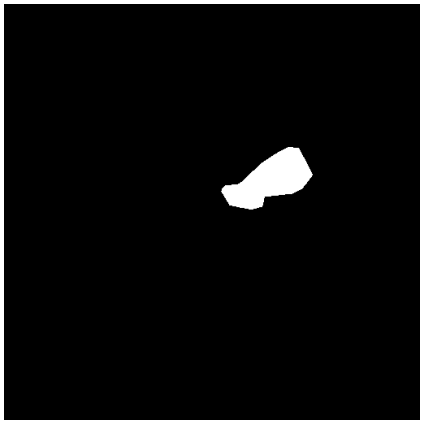


Figure 3.12—Mask of Tumor 1.

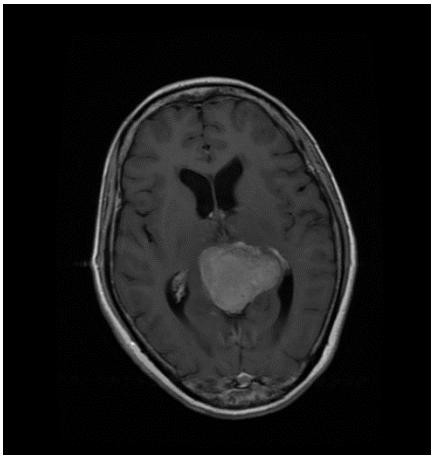


Figure 3.13—Tumor 2.

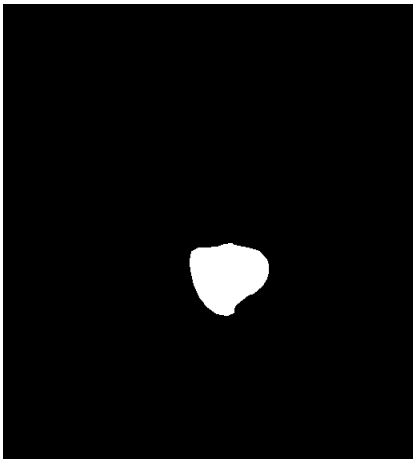


Figure 3.14—Mask of Tumor 2.

- **Data augmentation**

We selected images with small tumors and applied a data augmentation process on them to train the model more robustly for segmenting small tumor images. The process provided 780 images, each having a mask, which amounts to 780 masks. This brings the images now to 3844.

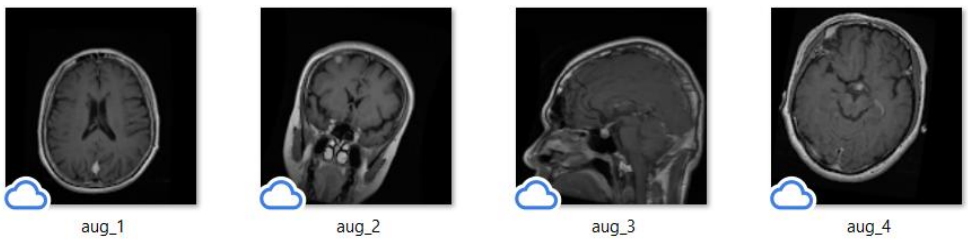


Figure 3.15—Some augmented images with small tumors.

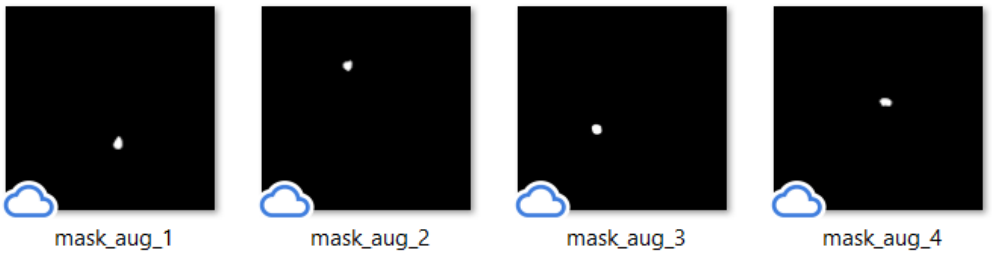


Figure 3.16— Masks of these images augmented with small tumors.

3.6 Project pipeline

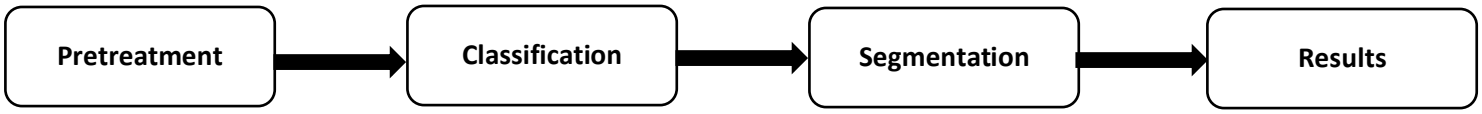


Figure 3.17—Project pipeline.

3.7 Development tools and environment

3.7.1 The programming language used

- **Python**

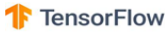
is an interpreted, object-oriented, high-level programming language designed for simplicity and readability. It provides advanced data structures, dynamic typing, and dynamic binding, making it ideal for rapid application development and scripting. Its freely and open-source availability across major platforms.it used in artificial intelligence, where its extensive libraries like TensorFlow and Keras facilitate machine learning and deep learning applications.

We used Python 3.12.4 for this work. It is the most recent major version of Python, with numerous optimizations and new features [59].

3.7.2 Kaggle

Kaggle is a data science and machine learning environment in which users compete with each other to create the most accurate model for a given problem or data set. Kaggle also has a community feature in which users can collaborate on projects, exchange data sets and code, and learn from other users' projects [60].

3.7.3 Libraries used

- **TensorFlow** 

is an open-source machine learning system developed by Google for building and deploying AI models in broad and diverse environments. It provides a comprehensive ecosystem of tools, libraries, and community resources for developing deep learning models, and supports both CPU and GPU acceleration for scalable computation [61].

- **Keras** 

Keras is a deep learning API written in python that provides an easy-to-use interface for building and training neural networks. Designed to be modular, flexible, and easy to use, it focuses on debugging speed, code elegance, and brevity, making it ideal for rapidly prototyping deep learning models. Keras runs on many machines learning frameworks, including TensorFlow [62].

- **OpenCV** 

OpenCV is an open-source library for machine learning and computer vision, used for real-time video and image processing. OpenCV supports multiple programming languages, including C++ and Python. It provides a wide range of tools and algorithms for tasks such as object detection and face recognition [63].

- **NumPy** 

The foundational package for scientific computing in Python is called NumPy (Numerical Python). It offers a collection of mathematical functions for working with these data structures in addition to support for arrays and big multidimensional arrays. Because of its effectiveness and compatibility with other libraries, it is frequently utilized in data analysis, machine learning, and deep learning [64].

- **Matplotlib** 

Matplotlib is a Python 2D plotting library that can produce high-quality figures. It supports both interactive and non-interactive usage and can save images in a variety of output formats (PNG, PS, etc.). It also offers many types of graphs (lines, columns, pie charts, histograms, etc.). It is also highly customizable, flexible, and easy to use [65].

- **Pandas** 

A python library that includes rich data structures and tools for working with structured datasets common in statistics, finance, and the social sciences, and is widely used in artificial intelligence, particularly machine learning. The library provides integrated and intuitive routines for performing common data processing and analysis operations. It aims to be the foundation for the future of statistical computing in Python [66].

- **Streamlit** 

is an open-source python framework for creating dynamic web apps for data science and machine learning. It is a popular option for quick prototyping and machine learning model visualization since it enables users to develop and implement data-driven apps with little coding knowledge [67].

3.7.4 Google Colaboratory 

Colab is a cloud-based platform that allows users to write and execute Python code in a Jupiter notebook environment. It provides free access to computing resources, including GPUs and TPUs, making it a popular tool for machine learning, data science, and deep learning projects. Colab facilitates collaboration by enabling users to share and edit notebooks in real time via Google Drive [68].

3.7.5 Hardware configuration

Deep learning is a term that refers to all machine learning techniques, in other words a form of learning based on mathematical approaches used to model data, with intense computational requirements and resource availability especially GPU 'graphical processing unit', is more powerful and can perform specific tasks very quickly, offers massive parallelism ideal for repetitive and highly parallel tasks.

My work is carried out on:

- A PC with the following configuration:
  - Laptop: HP EliteBook 830 G5, (RAM): 16.00 GB.
  - Microprocessor: Intel(R) Core (TM) i7-8650U CPU @ 1.90GHz 2.11GHz.
  - Hard drive: 512 GB.
  - System type: 64-bits operating system.
  - Google Compute Engine backend (GPU): Intel(R) UHD Graphics 620.
  - Windows: 11 pro.

3.8 Implementation of the classification model

3.8.1 Training, optimization and evaluation metrics

In this part, we implemented a brain tumor classification model using ensemble learning, combining four deep learning architectures: Xception, InceptionV3, VGG-19, and ResNet-50. These models are widely used in medical image analysis due to their powerful feature extraction capabilities.

3.8.1.1 Dataset Preparation

- **Image preprocessing**

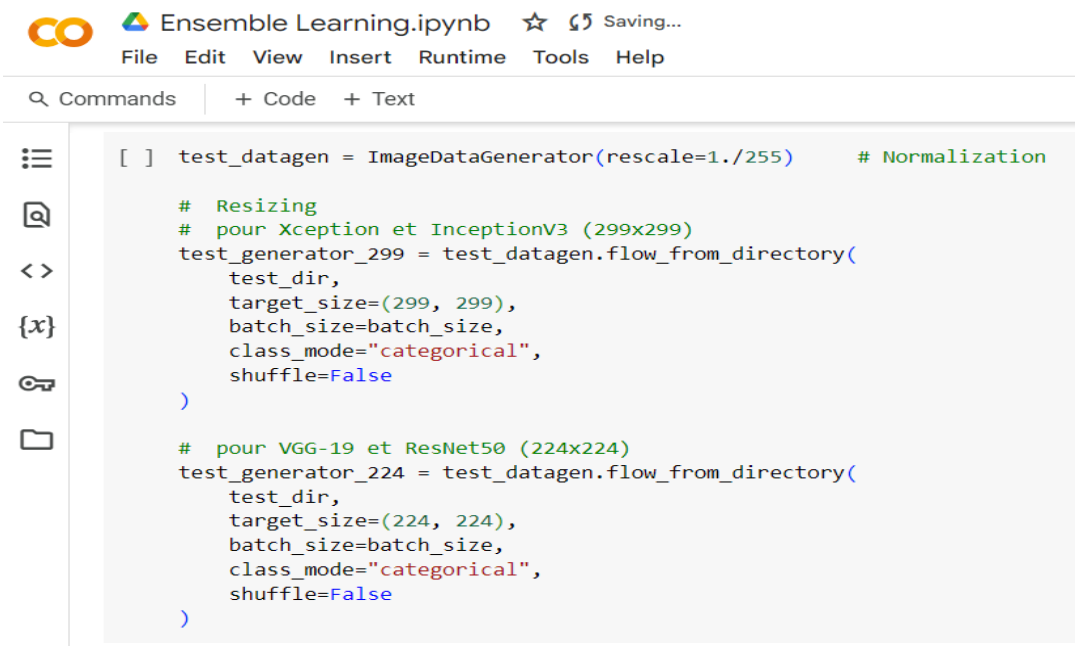


Figure 3.18—Image resizing and normalization for classification task.

We resized the images because each model accepts a specific input size, such as Xception and Inception which accept 299x299 while Res-Net and VGG-19 which accept 224x224. We also performed normalization by dividing the pixels by 255.

- **Data splitting**

Figure 3.19—Splitting data into training, testing and validation data.

Here, we divide the dataset into three parts: a training set used to train the model, a validation set used to fine-tune the model and evaluate its performance during training, and a test set dedicated to evaluating the final performance of the model on unseen data.

3.8.1.2 Hyperparameters

- **Optimizer (adamax):** Adamax is a variant of the Adam optimizer, based on the infinity norm, it's used to adjust the weights during training to minimize the loss function.

- **Learning rate:** controls how big each step the optimizer takes and we used 0.001 because it is a commonly used default value.

- **Loss:** we utilized "categorical_crossentropy" as our labels are one hot encoding for a multi-class classification.

- **Number of epochs:** In the model training process, we adopted a two-step strategy. First, we completely froze the base model and trained only the upper layers for 10 epochs. This step allowed the model to learn high-level, task-specific features while preserving the knowledge previously trained from ImageNet. Next, we improved the model by unfreezing the upper layers and resuming training for an additional 10 epochs, bringing the total number of epochs to 20. This step helped our model fine-tune its deep layers to better recognize patterns in the dataset, improving its accuracy and performance on new data.

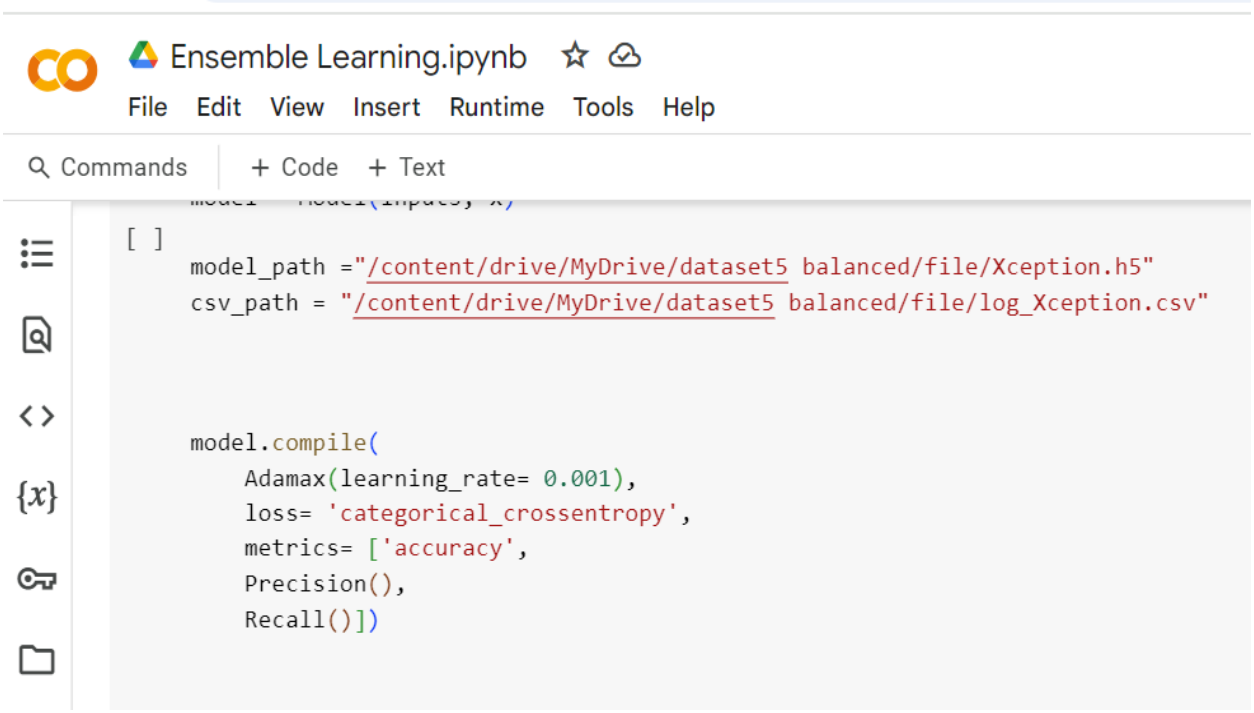


Figure 3.20—Optimizer, learning rate, loss.

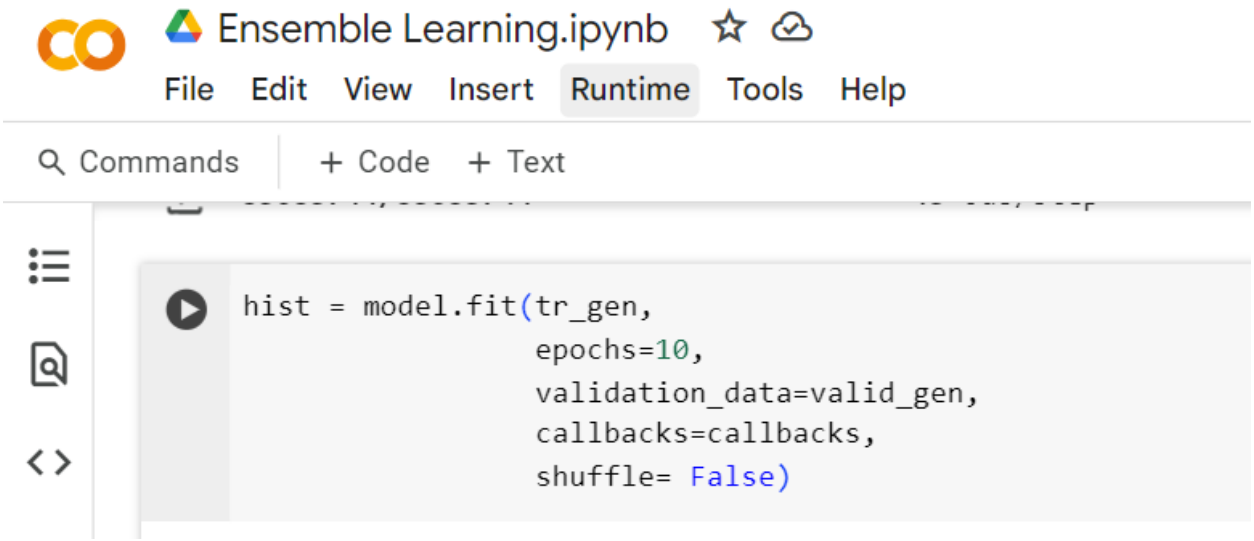


Figure 3.21—First 10 epochs.

Figure 3.22—Last 10 epochs.

3.8.1.3 Techniques for optimization

- **Learning rate scheduling:** To improve the learning process, we used a learning rate scheduling technique with the ReduceLROnPlateau callback function. This automatically reduces the learning rate when the validation loss optimization stops, helping the model converge better and avoid getting stuck during training.

- **Early stopping:** we used the Early Stopping technique to prevent overfitting and reduce training time, it monitors the validation loss and stops the training process if no improvement is observed for 5 epochs and the model restores the best weights achieved during training, ensuring optimal performance.

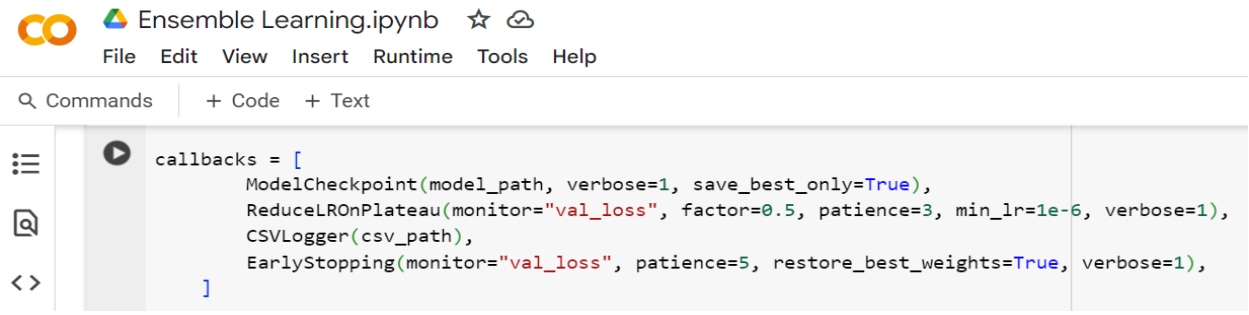


Figure 3.23—Callbacks (ReduceLROnPlateau, EarlyStopping).

- **Dropout**

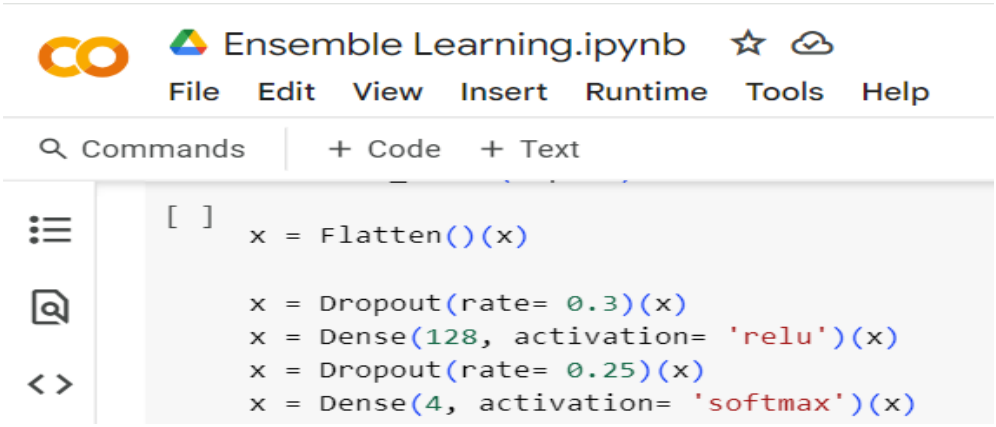


Figure 3.24—Dropout.

We used dropout in the fully connected layer twice to avoid overfitting.

- **Data augmentation**

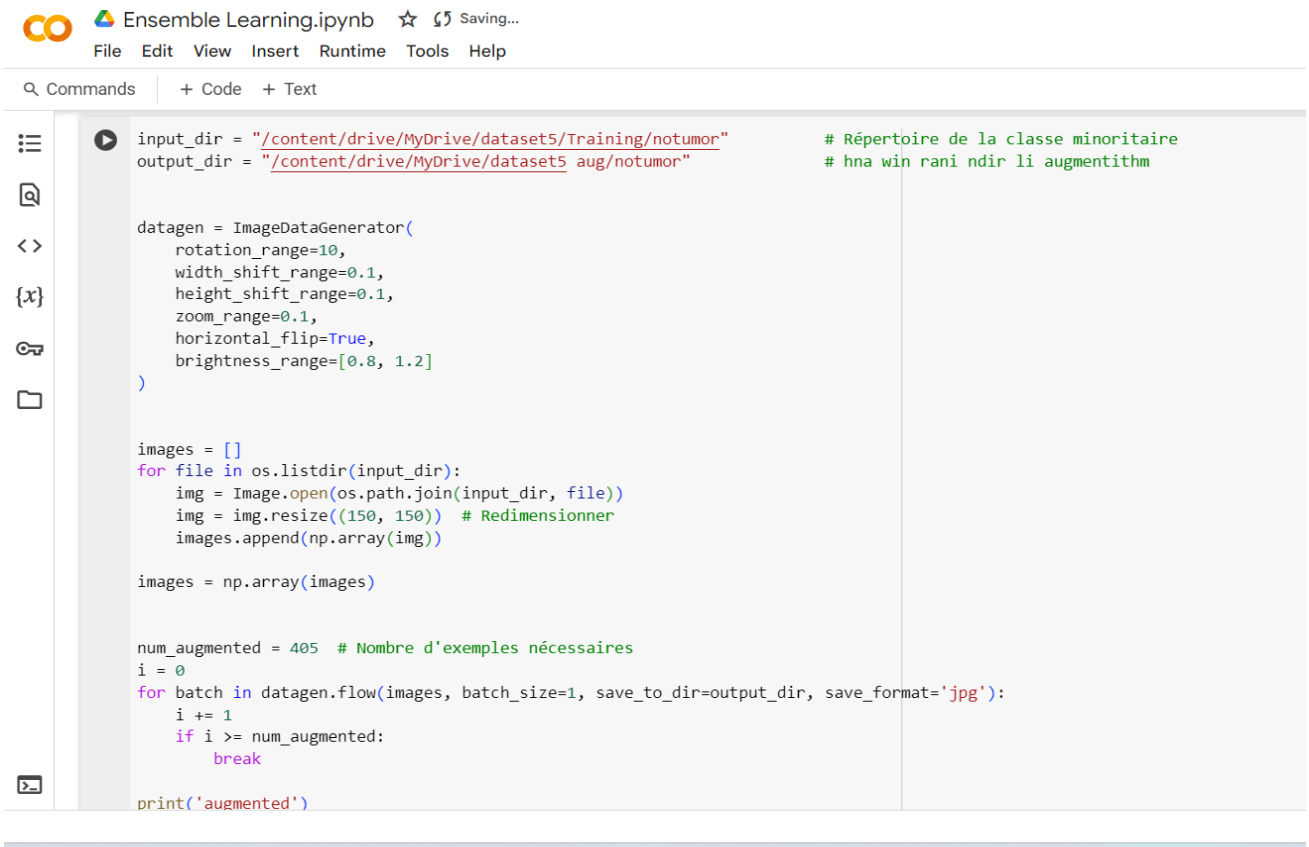


Figure 3.25—Dataset augmentation.

In this image we have created some transformations that we will apply to the dataset to make it balanced.

```
[ ]  glioma_path=os.path.join('/content/drive/MyDrive/dataset5/Training','glioma')
     meningioma_path=os.path.join('/content/drive/MyDrive/dataset5/Training','meningioma')
     pituitary_path=os.path.join('/content/drive/MyDrive/dataset5/Training','pituitary')
     no_tumor_path=os.path.join('/content/drive/MyDrive/dataset5/Training','notumor')

     glioma =os.listdir(glioma_path)
     meningioma =os.listdir(meningioma_path)
     pituitary =os.listdir(pituitary_path)
     no_tumor =os.listdir(no_tumor_path)

     print('Glioma :'+str(len(glioma)))
     print('meningioma :'+str(len(meningioma)))
     print('pituitary :'+str(len(pituitary)))
     print('no_tumor :'+str(len(no_tumor)))

     Glioma :1330
     meningioma :1339
     pituitary :1457
     no_tumor :1595
```

Figure 3.26—Dataset before data augmentation (imbalanced).

```
[ ]  glioma_path=os.path.join('/content/drive/MyDrive/dataset5 balanced/Training','glioma')
     meningioma_path=os.path.join('/content/drive/MyDrive/dataset5 balanced/Training','meningioma')
     pituitary_path=os.path.join('/content/drive/MyDrive/dataset5 balanced/Training','pituitary')
     no_tumor_path=os.path.join('/content/drive/MyDrive/dataset5 balanced/Training','notumor')

     glioma =os.listdir(glioma_path)
     meningioma =os.listdir(meningioma_path)
     pituitary =os.listdir(pituitary_path)
     no_tumor =os.listdir(no_tumor_path)

     print('Glioma :'+str(len(glioma)))
     print('meningioma :'+str(len(meningioma)))
     print('pituitary :'+str(len(pituitary)))
     print('no_tumor :'+str(len(no_tumor)))

     Glioma :2000
     meningioma :2000
     pituitary :2000
     no_tumor :2000
```

Figure 3.27—Dataset after data augmentation (balanced).

In both images 3.26 and 3.27 we can see the difference between the number of images before and after applying data augmentation.

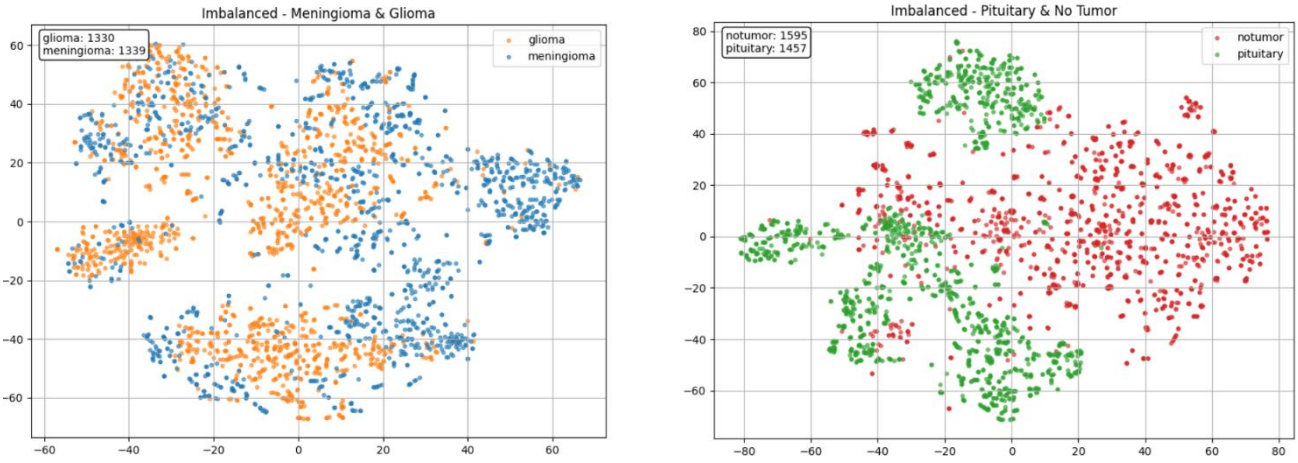


Figure 3.28—Data visualization of the classification dataset before balancing (imbalanced).

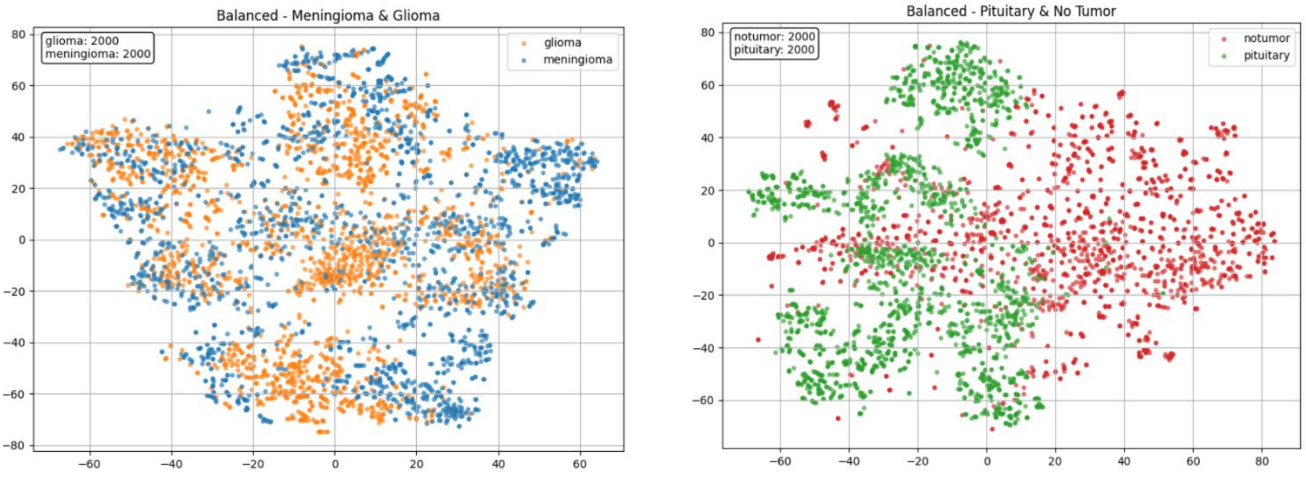


Figure 3.29—Data visualization of the classification dataset after balancing (balanced).

Figures 3.28 and 3.29 illustrate the t-SNE (t-distributed Stochastic Neighbor Embedding) visualization of the training dataset before and after class balancing for brain tumor classification.

3.8.1.4 Fully connected layer

Since these architectures are built on transfer learning, we profited their pre-trained weights on ImageNet while modifying the final layers to suit the classification task. We removed the original fully connected layers and replaced them with ones more suitable for my task and we applied these changes to all models.

The modifications included:

- Flattening the extracted features from the base model (converts into 1D vector).
- Adding dropout layers to prevent overfitting.
- We added a dense layer for feature refinement and we used 128 neurons to compress the data instead of passing all features to the final classification layer.
- We used a final dense layer for multi-class classification and 4 neurons because we have 4 classes (glioma, meningioma, pituitary, no tumor).

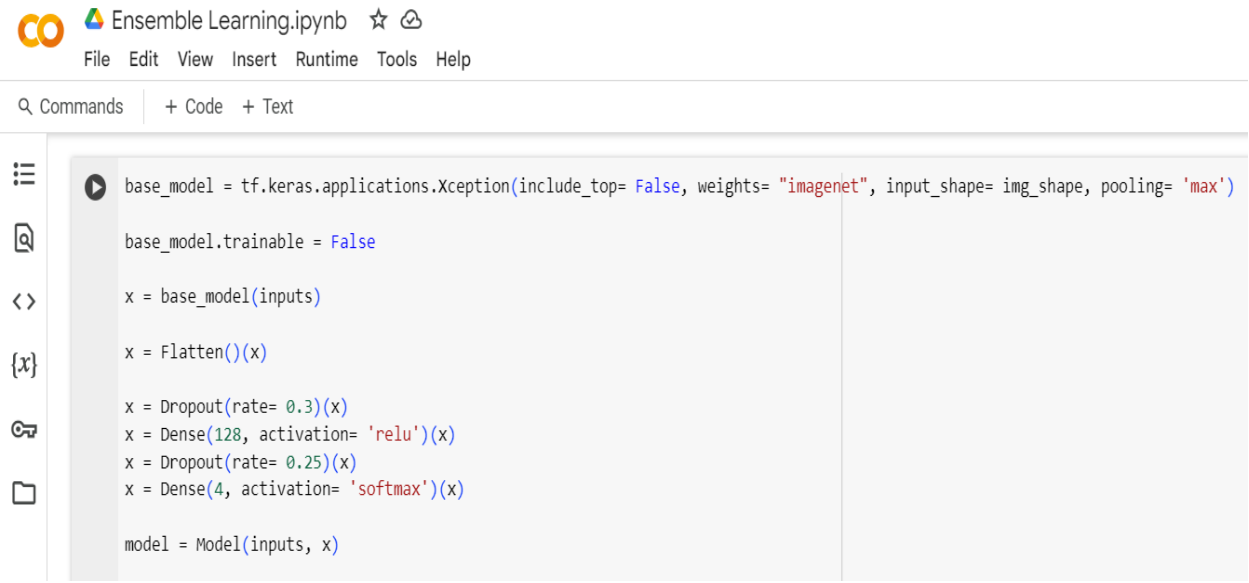


Figure 3.30—Implementation of the fully connected layer.

3.8.1.5 Activation function

We used the ReLu function in the hidden dense layer to Helps learn complex patterns, avoids vanishing gradient, speeds up training and adds non-linearity, in the other side we have used SoftMax function in the output layer to Converts outputs into probabilities for multi-class classification and make class predictions. Figure 3.28.

3.8.1.6 Ensemble Learning Approach

Ensemble Learning.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.callbacks import ModelCheckpoint, CSVLogger
import numpy as np
from sklearn.ensemble import GradientBoostingClassifier

X_train = np.hstack([xception_preds, inception_preds, resnet_preds, vgg_preds])  # hna Shape : (1307, 16)
y_train = test_generator_224.classes  # w hna Shape : (1307,)



# Definir un petit reseau de neurone
stacking_model = Sequential([
    Dense(64, activation='relu', input_shape=(16,)),
    Dense(32, activation='relu'),
    Dense(4, activation='softmax')  # psq 3ndi 4 classes
])


stacking_model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])


model_path = "/content/drive/MyDrive/dataset5 balanced/file/Ensemble_learning.h5"
csv_path = "/content/drive/MyDrive/dataset5 balanced/file/log_Ensemble_learning.csv"

callbacks = [
        ModelCheckpoint(model_path, verbose=1, save_best_only=True),
        CSVLogger(csv_path, append=True),
        ReduceLROnPlateau(monitor="loss", factor=0.5, patience=3, min_lr=1e-6, verbose=1),
        EarlyStopping(monitor="loss", patience=3, restore_best_weights=True, verbose=1),
    ]
```

Figure 3.31—Implementation of the stacking model.

We implemented a stacking ensemble model by combining the predictions of multiple base classifiers. On top of this ensemble, we added a lightweight neural network as a meta-learner, which was trained to make the final predictions based on the outputs of the base models.

3.8.1.6 Evaluation metrics

- **Confusion matrix:** In statistical classification, we create models or algorithms to classify or predict data into a limited number of classes. Since the models are not perfect, there will be some misclassified data points. A confusion matrix is basically a tabular representation of how well a model performs and can be used in both binary classification and multi-class classification problems [19].

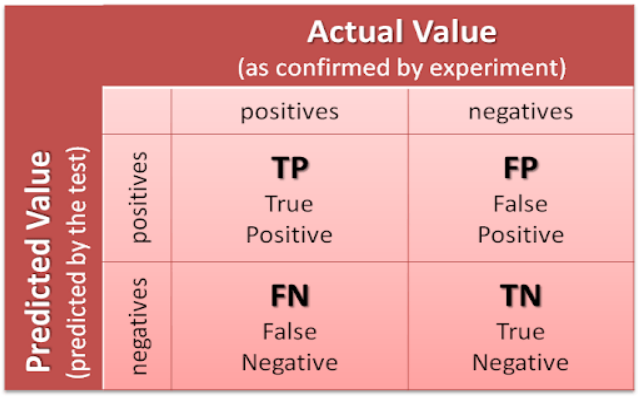


Figure 3.32 – Confusion matrix.

True Positive (TP): The model correctly predicts a positive case.

True Negative (TN): The model correctly predicts a negative case.

False Positive (FP): The model incorrectly predicts a positive case.

False Negative (FN): The model incorrectly predicts a negative case.

There are several metrics that can be calculated from the matrix, including:

1. **Accuracy:** This measure may be the simplest, most effective and most straightforward for classification tasks as it measures the number of correct predictions to the total number of predictions [20].

   it can be misleading if your classes are imbalanced.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

2. **Precision:** precision measures how many of the predicted positive cases are actually correct, it's the ratio of true positives to all positives [20].

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

3. **Recall:** measures how many of the actual positive cases were correctly predicted [20].

$$Recall = \frac{TP}{TP + FN}$$

1. **F1-Score:** F1 score is harmonic mean of precision and recall, it tries to achieve a trade-off between precision and recall for a classification task [20].

   Is more useful when you have class imbalance because it tries to achieve a trade-off between precision and recall**.**

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

3.8.2 Performance evaluation and analysis (discussion of results)

We will talk about the statistics and results provided by each model and we will start with:

3.8.2.1 Xception

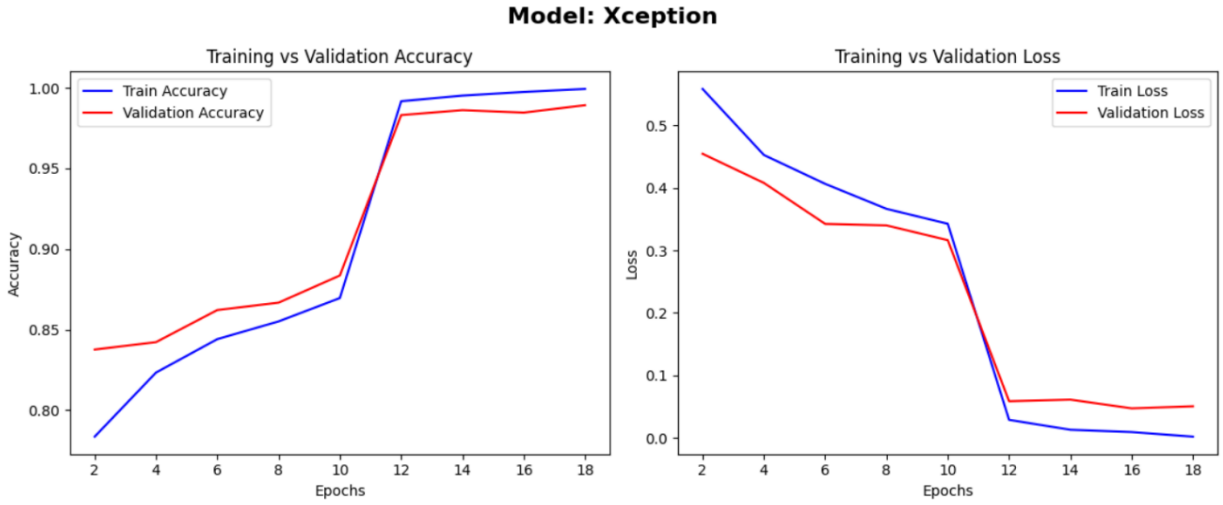


Figure 3.33— The loss curve and the accuracy curve Xception.

No Overfitting Yet: Since the validation loss doesn't start increasing significantly while the training loss continues to drop, there's no clear sign of overfitting in this case.

Good Generalization: Both the training and validation losses are decreasing, which suggests the model is learning and generalizing well to new data.

epoch	accuracy	learning_rate	loss	precision	recall	val_accuracy	val_loss	val_precision	val_recall
1	0.6865000129	0.001000000047	0.7967743874	0.7429305911	0.6141250134	0.817764163	0.510060966	0.8583617806	0.7702909708
2	0.7836250067	0.001000000047	0.5578293204	0.821695745	0.7413750291	0.8376722932	0.4542316198	0.8783333302	0.8070443869
3	0.8106250167	0.001000000047	0.4921119511	0.8456924558	0.7754999995	0.8499234319	0.4156242311	0.8727858067	0.8300153017
4	0.8233749866	0.001000000047	0.4523415864	0.8512008786	0.7929999828	0.8422664404	0.4077157378	0.8721311688	0.8147013783
5	0.8403750062	0.001000000047	0.4222299457	0.8644562364	0.8147500157	0.860643208	0.3692053258	0.8700475693	0.8407350779
6	0.8441249728	0.001000000047	0.4062346518	0.8624539971	0.8206250072	0.8621745706	0.3422723413	0.8767772317	0.8499234319
7	0.851000011	0.001000000047	0.3863767684	0.8735178113	0.8287500143	0.8744257092	0.3457946479	0.8874801993	0.8575804234
8	0.85512501	0.001000000047	0.3663553894	0.8733150363	0.8341249824	0.8667687774	0.3399190307	0.882825017	0.8422664404
9	0.8616250157	0.001000000047	0.3509121835	0.8803977966	0.8410000205	0.8805512786	0.3152655959	0.8947368264	0.8591117859
10	0.8696249723	0.001000000047	0.3424623311	0.8881164789	0.8463749886	0.8836140633	0.3162717223	0.905844152	0.8545176387
11	0.9315000176	0.001000000047	0.2020722777	0.9431236982	0.9223750234	0.9754977226	0.05719103292	0.9799692035	0.9739663005
12	0.9917500019	0.001000000047	0.02934352495	0.9923626184	0.9907500148	0.9831546545	0.05906296521	0.9831546545	0.9831546545
13	0.9955000281	0.001000000047	0.01479046792	0.9957484007	0.9953749776	0.990811646	0.02650929801	0.990811646	0.990811646
14	0.9952499866	0.001000000047	0.01349552441	0.9958719015	0.9951249957	0.9862174392	0.06163296476	0.9862174392	0.9862174392
15	0.9955000281	0.001000000047	0.01402648073	0.9961225986	0.9955000281	0.981623292	0.07913708687	0.981623292	0.981623292
16	0.9975000024	0.001000000047	0.009812013246	0.9976243973	0.9973750114	0.9846860766	0.04768224806	0.9846860766	0.9846860766
17	0.9986249804	0.0005000000237	0.003882636316	0.9986249804	0.9986249804	0.990811646	0.04056061804	0.990811646	0.990811646
18	0.9993749857	0.0005000000237	0.002489642007	0.999499917	0.9993749857	0.9892802238	0.05096396431	0.9892802238	0.9892802238

Figure 3.34— CSV Export of Training Logs for Xception.

In the Xception model, we observe that EarlyStopping stopped the training process because the val_loss did not improve for 5 consecutive epochs. It selected the 13th epoch as the best one, with an accuracy of **0.995**, a loss of **0.014**, a validation accuracy of **0.990**, and a validation loss of **0.026**.

3.8.2.2 InceptionV3

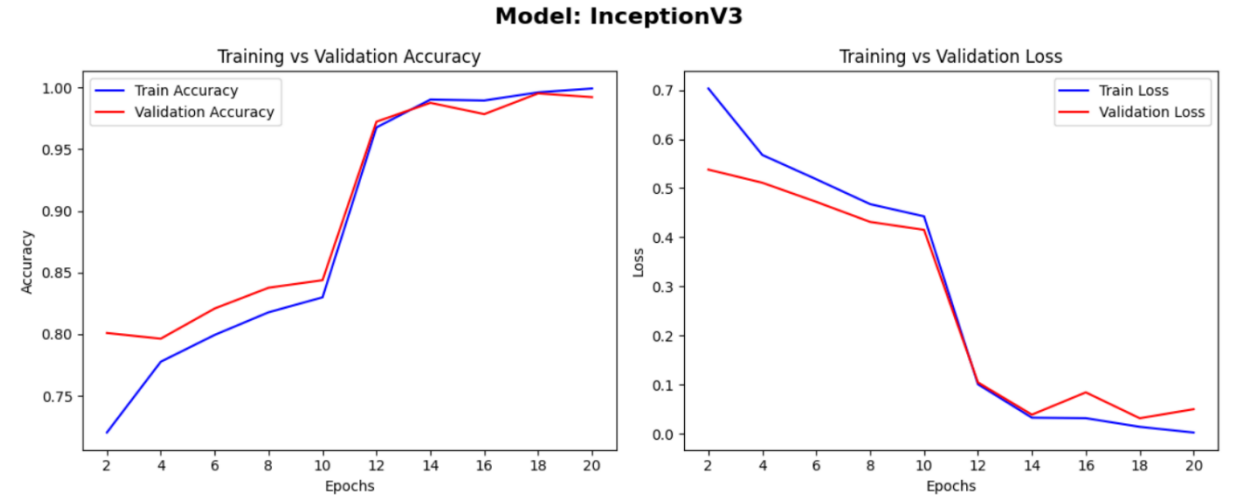


Figure 3.35— The loss curve and the accuracy curve for InceptionV3.

epoch	accuracy	learning_rate	loss	precision	recall	val_accuracy	val_loss	val_precision	val_recall
1	0.5914999843	0.001000000047	1.044712901	0.6703083515	0.4754999876	0.7825421095	0.6048266292	0.8458646536	0.6891270876
2	0.7201250196	0.001000000047	0.7032022476	0.7796403766	0.6395000219	0.8009188175	0.537972331	0.8324872851	0.7534456253
3	0.7555000186	0.001000000047	0.6184465885	0.8052790761	0.6978750229	0.8116385937	0.5167790651	0.8439597487	0.7702909708
4	0.7776250243	0.001000000047	0.5678141713	0.8237885237	0.7246249914	0.7963246703	0.5112496018	0.8313856721	0.7626339793
5	0.7906249762	0.001000000047	0.5457733274	0.8286154866	0.7391250134	0.836140871	0.4531199634	0.8486312628	0.8070443869
6	0.7993749976	0.001000000047	0.518360734	0.8395403624	0.7580000162	0.8208269477	0.4723235071	0.862244904	0.7764165401
7	0.8105000257	0.001000000047	0.4944218695	0.8449793458	0.7678750157	0.8346095085	0.4479523301	0.8552845716	0.8055130243
8	0.8177499771	0.001000000047	0.4676565826	0.8497830629	0.7835000157	0.8376722932	0.431417048	0.8613376617	0.808575809
9	0.8224999905	0.001000000047	0.4619686007	0.8551098108	0.7886250019	0.8376722932	0.4317254126	0.860655725	0.8039816022
10	0.8298749924	0.001000000047	0.4427320361	0.8593243361	0.7948750257	0.8437978625	0.4152666628	0.86721313	0.8101071715
11	0.8945000172	0.001000000047	0.3104938567	0.9235787392	0.8671249747	0.9525268078	0.1241055056	0.9670846462	0.9448698163
12	0.9677500129	0.001000000047	0.1006610617	0.9727134705	0.9624999762	0.972434938	0.1044195592	0.9723926187	0.9709035158
13	0.981374979	0.001000000047	0.05578585342	0.9833103418	0.9794999957	0.9555895925	0.1293546706	0.964341104	0.9525268078
14	0.9903749824	0.001000000047	0.03269348666	0.9914829731	0.9894999862	0.9877488613	0.03881817684	0.9892638326	0.9877488613
15	0.9906250238	0.001000000047	0.03490126505	0.9916113615	0.9900000095	0.9464012384	0.2546699643	0.9491525292	0.9433384538
16	0.9896249771	0.001000000047	0.03163143992	0.9908567071	0.9888749719	0.9785605073	0.08422095329	0.9800613523	0.9785605073
17	0.9908750057	0.001000000047	0.03143722564	0.991735518	0.9900000095	0.9846860766	0.06046072766	0.9846153855	0.9800918698
18	0.9962499738	0.0005000000237	0.01404568739	0.9964982271	0.9959999919	0.9954057932	0.03153498471	0.9954057932	0.9954057932
19	0.9986249804	0.0005000000237	0.005901292898	0.9986248016	0.9984999895	0.9923430085	0.04176981747	0.9938650131	0.9923430085
20	0.9993749857	0.0005000000237	0.00236038235	0.999499917	0.9993749857	0.9923430085	0.04991571605	0.9923430085	0.9923430085

Figure 3.36— CSV Export of Training Logs for InceptionV3.

Now in this model as we see he chose the 18th epoch with **0.9962** of accuracy, **0.0140** of loss, **0.9954** of validation accuracy and **0.0315** of validation loss.

3.8.2.3 ResNet50

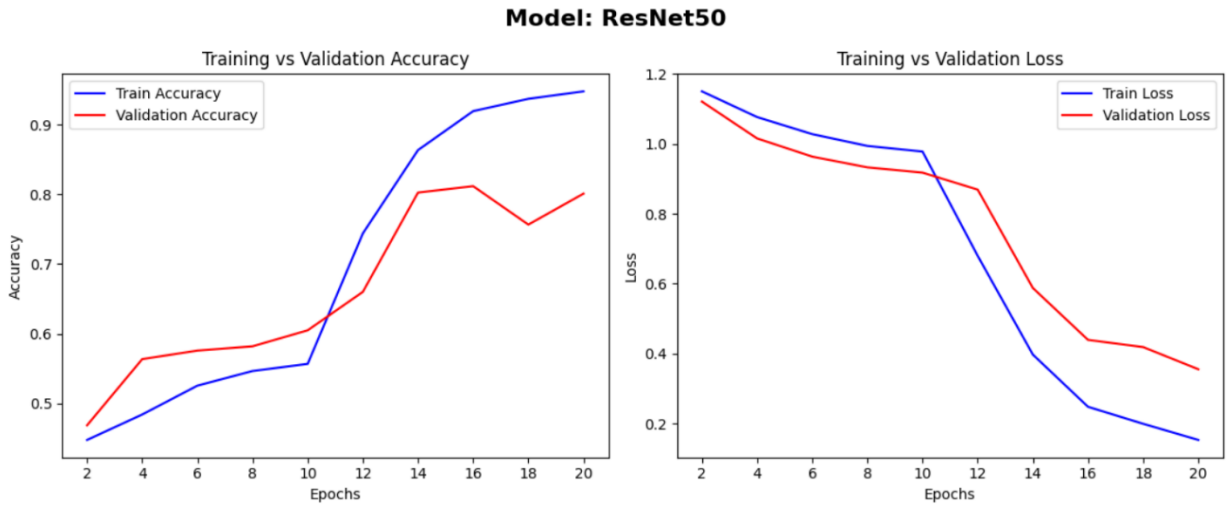


Figure 3.37— The loss curve and the accuracy curve for ResNet50.

epoch	accuracy	learning_rate	loss	precision	recall	val_accuracy	val_loss	val_precision	val_recall
1	0.3841249943	0.001000000047	1.427197695	0.5021398067	0.175999999	0.4701378345	1.158992767	0.8508771658	0.1485451758
2	0.4476250112	0.001000000047	1.1502949	0.7607601881	0.1701249927	0.4686064422	1.12101531	0.8644067645	0.1562021375
3	0.4668749869	0.001000000047	1.10555017	0.7769308686	0.1911250055	0.51301682	1.061478257	0.8476821184	0.1960183829
4	0.4841249883	0.001000000047	1.076740742	0.7369520068	0.2206249982	0.5635528564	1.015638351	0.8019801974	0.2480857521
5	0.5148749948	0.001000000047	1.048853517	0.7430124283	0.2392500043	0.5574272871	1.005553603	0.718978107	0.3016845286
6	0.5254999995	0.001000000047	1.027889609	0.7281752825	0.2658750117	0.5758039951	0.9635557532	0.7896825671	0.3047473133
7	0.535374999	0.001000000047	1.010488153	0.7164964676	0.2893750072	0.5788667798	0.9842656255	0.7147766352	0.3185298741
8	0.5465000272	0.001000000047	0.9941601753	0.7188872695	0.3068749905	0.5819295645	0.9327847958	0.7241379023	0.3859111667
9	0.5478749871	0.001000000047	0.9868047237	0.7242069244	0.31674999	0.583460927	0.9231871367	0.6919192076	0.4196018279
10	0.5567499995	0.001000000047	0.9779644608	0.7223922014	0.3246249855	0.6049004793	0.9177509046	0.7684887648	0.3660030663
11	0.5686249733	0.001000000047	1.503089547	0.7065482736	0.397875011	0.2817764282	0.9065353775	0.340816319	0.2557427287
12	0.7438750267	0.001000000047	0.6795935035	0.8245558143	0.6497499943	0.6600306034	0.8690680861	0.7330895662	0.6140888333
13	0.8121250272	0.001000000047	0.5270063877	0.8631167412	0.7456250191	0.7779479623	0.6866332293	0.8231173158	0.7197549939
14	0.8633750081	0.001000000047	0.3971581757	0.9017795324	0.8171250224	0.8024502397	0.5876773596	0.8341625333	0.7702909708
15	0.8907499909	0.001000000047	0.3248457909	0.9168216586	0.8625000119	0.8545176387	0.5394589305	0.8634222746	0.8422664404
16	0.9192500114	0.001000000047	0.2477966696	0.9383651018	0.8982499838	0.8116385937	0.4392292106	0.8238993883	0.8024502397
17	0.9298750162	0.001000000047	0.2110958397	0.943785429	0.9150000215	0.7595711946	0.4386358953	0.8228980303	0.7044410706
18	0.9368749857	0.001000000047	0.1991960406	0.9523132443	0.9211249948	0.75650841	0.4186266851	0.7940199375	0.7320061326
19	0.9506250024	0.001000000047	0.151694566	0.9585347176	0.9419999719	0.9402756691	0.390067504	0.9559054971	0.9295558929
20	0.9476249814	0.001000000047	0.1529649943	0.9584929943	0.9381250143	0.8009188175	0.35522861	0.8119122386	0.7932618856

Figure 3.38— CSV Export of Training Logs for ResNet50.

The ResNet model gave us these results as it chose the last epochs: accuracy **0.9476**, loss **0.1529**, validation accuracy **0.8009** and validation loss **0.3552**.

3.8.2.4 VGG19

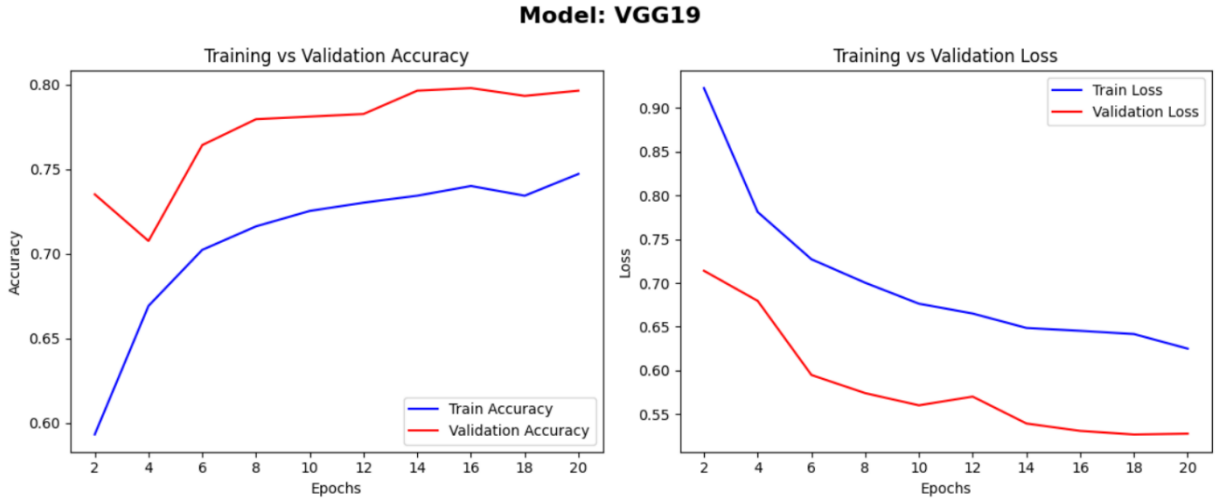


Figure 3.39— The loss curve and the accuracy curve for VGG19.

epoch	accuracy	learning_rate	loss	precision_3	recall_3	val_accuracy	val_loss	val_precision_3	val_recall_3
1	0.4742499888	0.001000000047	1.16253233	0.5974806547	0.2608749866	0.6891270876	0.8217537999	0.8987854123	0.3399693668
2	0.5931249857	0.001000000047	0.9226165414	0.7258937955	0.4187499881	0.7350689173	0.7138989568	0.8256658316	0.5222052336
3	0.6458749771	0.001000000047	0.8332029581	0.7353755236	0.4981249869	0.7059724331	0.7114751339	0.767123282	0.6003062725
4	0.6691250205	0.001000000047	0.7810131311	0.7431684732	0.554125011	0.7075038552	0.6794682145	0.7835820913	0.6431853175
5	0.6848750114	0.001000000047	0.7581948042	0.7625578046	0.5768749714	0.765696764	0.6109752059	0.8159851432	0.6722818017
6	0.7022500038	0.001000000047	0.7270371914	0.7639634013	0.605250001	0.7641654015	0.5948117375	0.8327067494	0.678407371
7	0.7016249895	0.001000000047	0.7123214006	0.7680453062	0.6104999781	0.7488514781	0.6079118848	0.7897526622	0.6845329404
8	0.7161250114	0.001000000047	0.7002208233	0.7739966512	0.6340000033	0.7794793248	0.574070394	0.8245614171	0.7197549939
9	0.7251250148	0.001000000047	0.6789327264	0.7807494998	0.6458749771	0.7871363163	0.5562054515	0.8410714269	0.7212863564
10	0.7252500057	0.001000000047	0.676261127	0.7791429162	0.6499999762	0.781010747	0.5601370335	0.8263888955	0.7289433479
11	0.7189999819	0.001000000047	0.6827843189	0.7719771862	0.6432499886	0.7534456253	0.5850449204	0.8059440851	0.7059724331
12	0.73012501	0.001000000047	0.6650435328	0.7786619663	0.66049999	0.7825421095	0.5701300502	0.8209219575	0.7090352178
13	0.7335000038	0.001000000047	0.6604516506	0.7838680744	0.6596249938	0.7886676788	0.553587079	0.8368794322	0.7228177786
14	0.7342500091	0.001000000047	0.6485632062	0.7878832221	0.6713749766	0.7963246703	0.5393920541	0.8440207839	0.7457886934
15	0.7392500043	0.001000000047	0.6478963494	0.7846243978	0.6685000062	0.7963246703	0.5388951302	0.8385416865	0.7396630645
16	0.7400000095	0.001000000047	0.6452735066	0.7847715616	0.6763749719	0.7978560328	0.5309044123	0.8304794431	0.7427259088
17	0.7433750033	0.001000000047	0.6375150681	0.7902523875	0.6809999943	0.781010747	0.5349113941	0.8336282969	0.7212863564
18	0.7342500091	0.001000000047	0.6415902376	0.783157289	0.6672499776	0.7932618856	0.5268341303	0.8397212625	0.7381317019
19	0.7473750114	0.001000000047	0.6203888655	0.7921574116	0.6893749833	0.7871363163	0.54758811	0.8237287998	0.7442572713
20	0.7471250296	0.001000000047	0.624907732	0.7923900485	0.6846250296	0.7963246703	0.5277609229	0.8373287916	0.7488514781

Figure 3.40— CSV Export of Training Logs for VGG19.

These results were presented by the VGG19 model with low accuracy **0.7342**, loss **0.6415**, validation accuracy **0.7932** and validation loss **0.5268**.

3.8.2.5 Confusion matrix

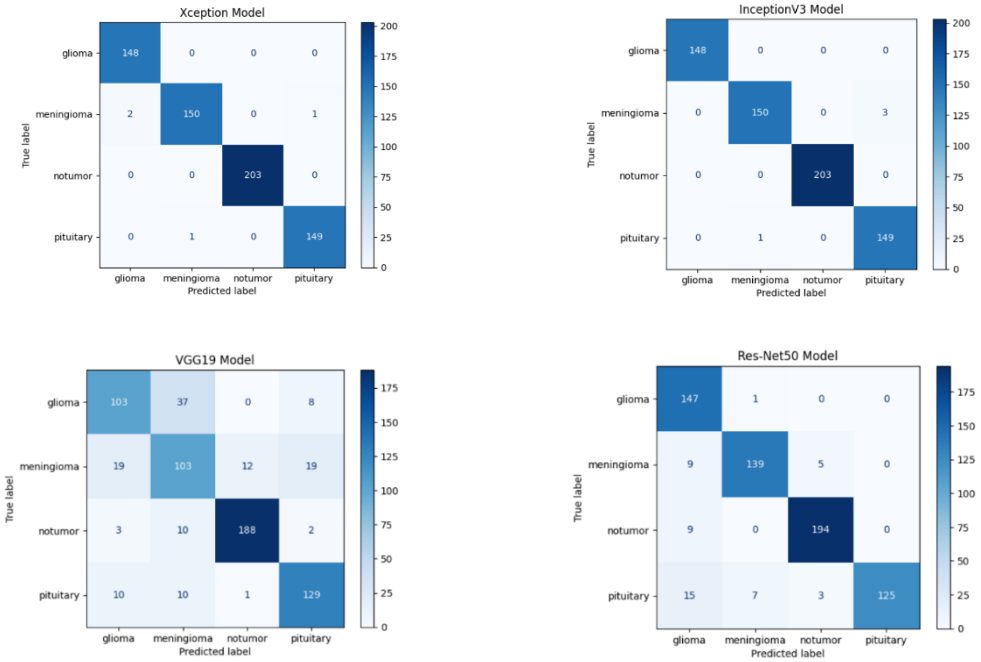


Figure 3.41— The confusion matrices for the four models.

Confusion matrices indicate that the Xception and InceptionV3 models have excellent classification performance, correctly classifying **148** glioma, **150** meningioma, **203** no-tumor, and **149** pituitary tumor cases, with a small misclassification of **1** to **3** cases. ResNet50 correctly classified most cases, but incorrectly classified **9** no-tumor cases and **15** pituitary tumor cases as glioma, and **9** meningioma cases as glioma, showing some overlap in the learned features. On the other hand, the VGG19 model suffers from severe confusion, especially between glioma and meningioma, classifying **37** glioma cases as meningioma and **19** meningioma cases as glioma, and some cases between pituitary and other categories, reflecting weak feature extraction capabilities.

3.8.2.6 Roc-Auc Curve

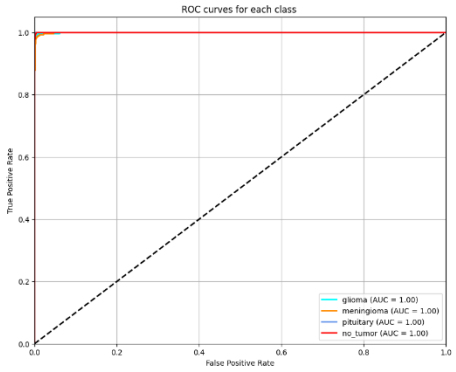


Figure 3.42— Roc/Auc curve for Xception.

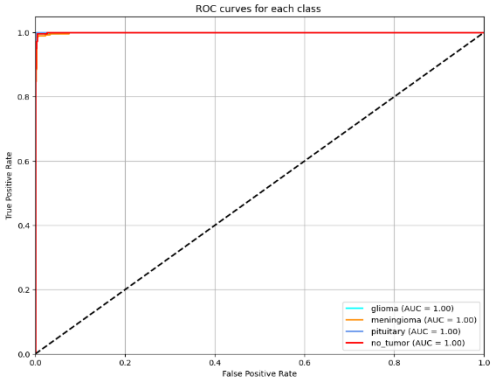


Figure 3.43— Roc/Auc curve for InceptionV3.

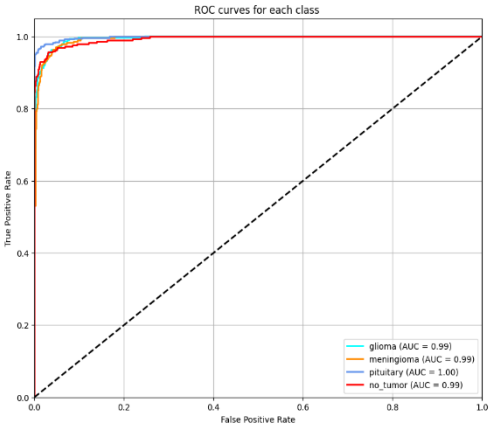


Figure 3.44— Roc/Auc curve for ResNet-50.

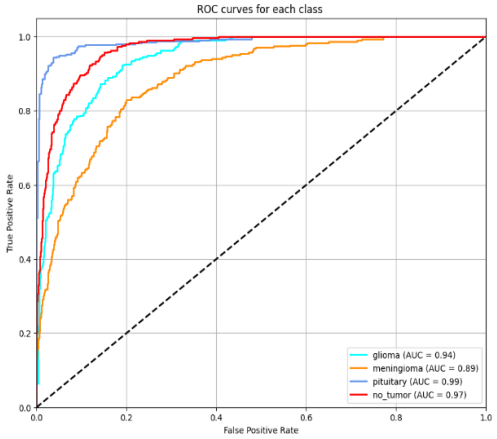


Figure 3.45— Roc/Auc curve for VGG-19.

Receiver operating characteristic (ROC) curves show how well a model can separate classes, a higher AUC (close to 1) means better prediction performance.

The images above demonstrate the classification performance of the four baseline models: Xception, InceptionV3, ResNet-50, and VGG-19 across four tumor classes (glioma, meningioma, pituitary, and no tumor).

Xception and InceptionV3 performed almost perfectly, with AUC scores close to **1.00** for all classes.

ResNet-50 performed very well, with AUC values ranging between **0.99** and **1.00**.

VGG-19 shows low AUC values, especially for categories such as meningioma and no tumor (AUC ≈ **0.89** - **0.97**).

These results indicate that the three models: Xception, InceptionV3, and ResNet-50 are able to discriminate and classify tumors with high accuracy, while the VGG-19 model faces some difficulties.

3.8.2.7 Stacking model

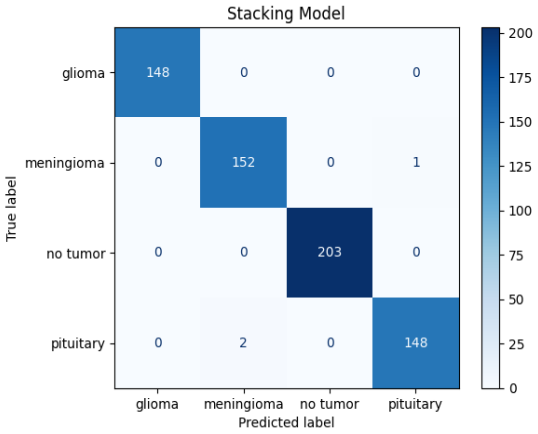


Figure 3.46— The confusion matrix for the stacking model.

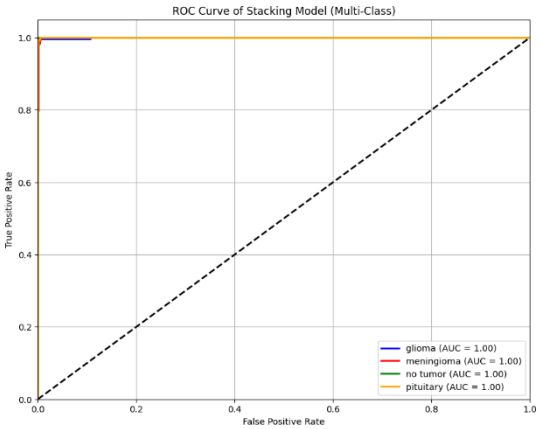


Figure 3.47— The ROC/Auc curve for the stacking model.

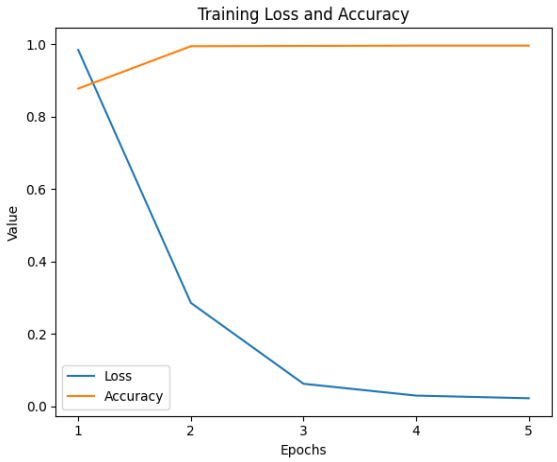


Figure 3.48— The Training loss and accuracy curve for the stacking model.

epoch	accuracy	loss
1	0.877582252	0.9844990373
2	0.9946442246	0.2860683203
3	0.9954093099	0.06272928417
4	0.9961744547	0.03007725999
5	0.9961744547	0.02269619144

Figure 3.49— CSV Export for stacking model.

The figure 3.46 shows the confusion matrix for the stacking model, which achieved almost perfect classification, such as 148/148 for glioma and 203/203 for no tumor, and very few misclassifications, totaling 3 images.

Figure 3.47 shows the ROC/AUC curve. It shows that the model reached an AUC value of 1.00 for all categories (glioma, meningioma, no tumor, pituitary), indicating perfect classification.

The training curves shown in the figure 3.48 show that the loss dropped to nearly 0 and the accuracy reached 100% after only 5 epochs, which proves the excellent performance. Since the stacking model is trained on predictions of pre-trained base models rather than directly on raw images, using a separate validation set was not essential, as the base models had already been validated individually.

Figure 3.49 shows the CSV export of the stacking model where we see that in only 5 epochs it gave us excellent results and the best values were taken and they are: **0.996** of accuracy and **0.022** of loss.

3.9 Implementation of the segmentation model

3.9.1 Training and optimization

3.9.1.1 Dataset Preparation

- **Image preprocessing**

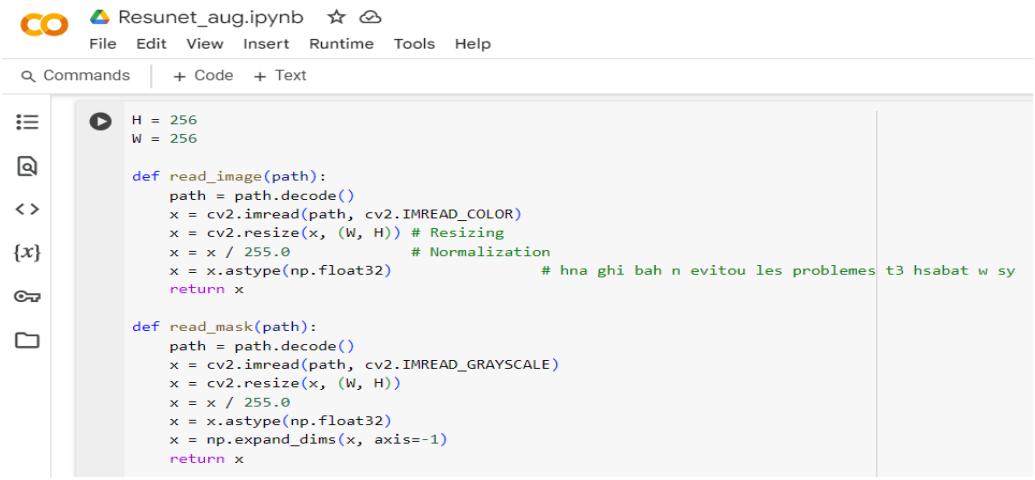


Figure 3.50—Image resizing and normalization for segmentation task.

Here in the segmentation task, we also resized the images to 256×256 because the Res-Unet model accepts this size. In addition, we did normalization as well.

- **Data splitting**

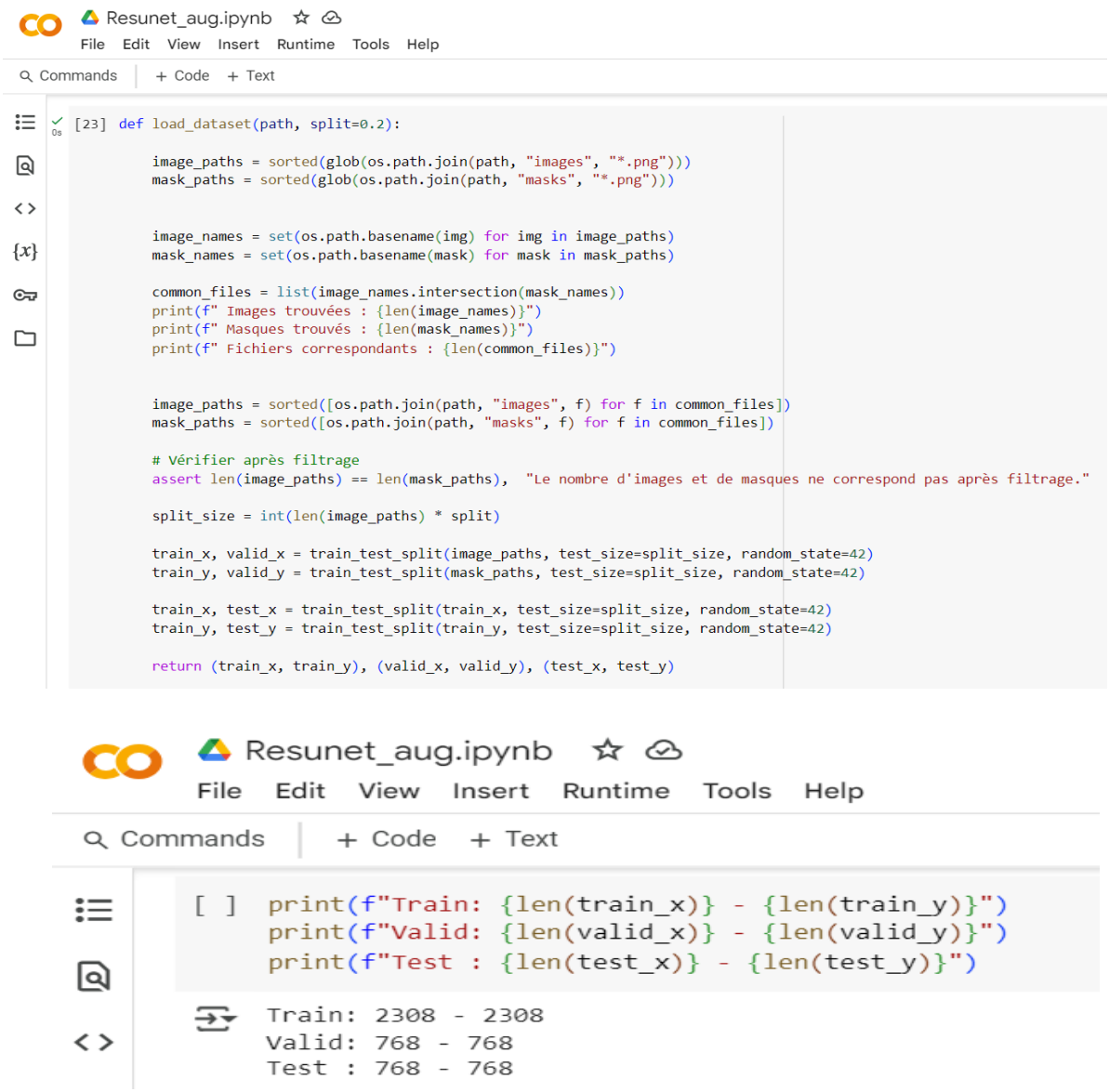


Figure 3.51—Splitting data into training, testing and validation data.

We split the data into 60% for training, 20% for testing, and 20% for validation.

3.9.1.2 Hyperparameters

- **Optimizer:** I used Adam optimizer.
- **Learning rate:** I used the popular value which is 0.001.

- **Loss:** in loss function I utilized dice_loss because it optimizes the model for better segmentation.
- **Number of epochs:** 30 epochs.

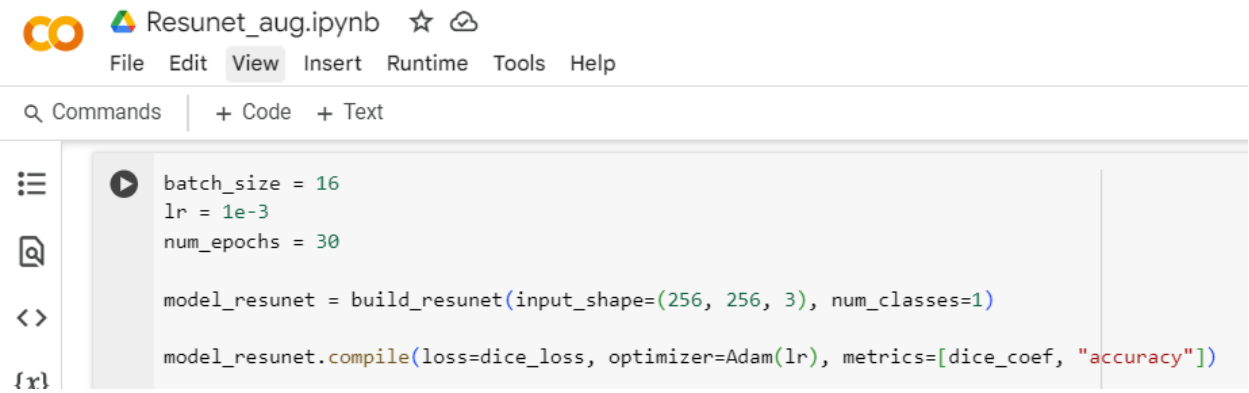


Figure 3.52—Hyperparameters used on segmentation task.

3.9.1.3 Techniques for optimization

- **Learning rate scheduling:** if there is no improvement in the val_dice_coef for 5 consecutive epochs, the learning rate is reduced by a factor of 0.1, this helps model to learn more precisely.
- **Early stopping:** the val_dice_coef metric is monitored. If no improvement is observed after 10 epochs, training is stopped and only best weights are retained.

Figure 3.53—Callbacks for segmentation task.

- **Batch Normalization:** We also used this method as an optimization technique to normalizes the inputs of each layer to have a consistent distribution of values during training.

Figure 3.54— Add batch normalization to our Res-Unet model.

- **Data augmentation**

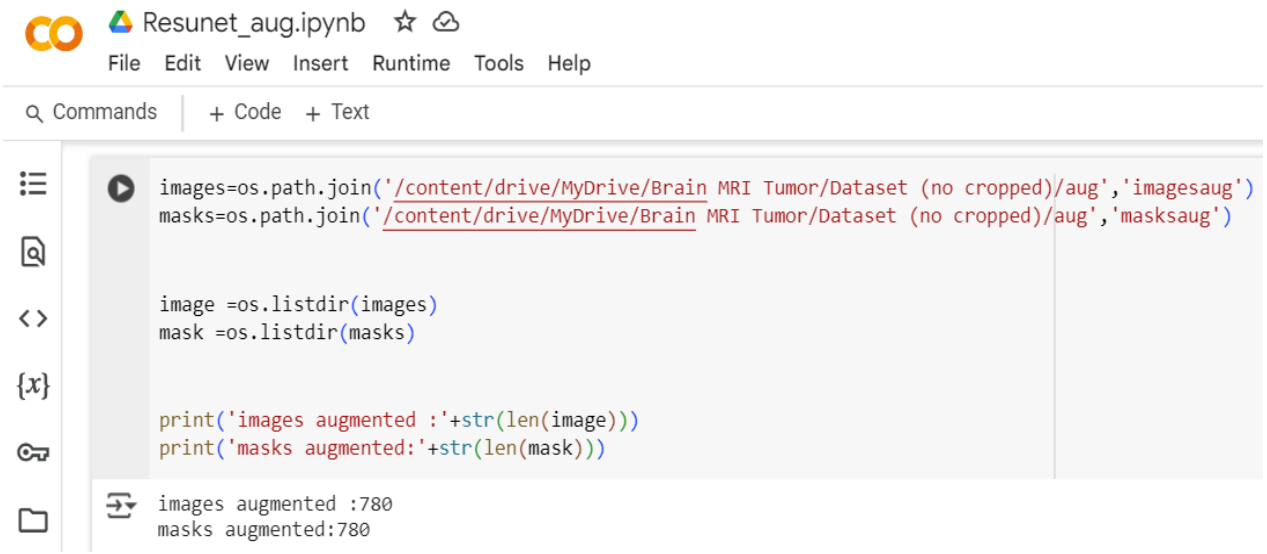


Figure 3.55— Number of images and masks augmented.

Here we have added 780 images using data augmentation method and as we said before we have selected only the images that contain small tumors and applied this method to them.

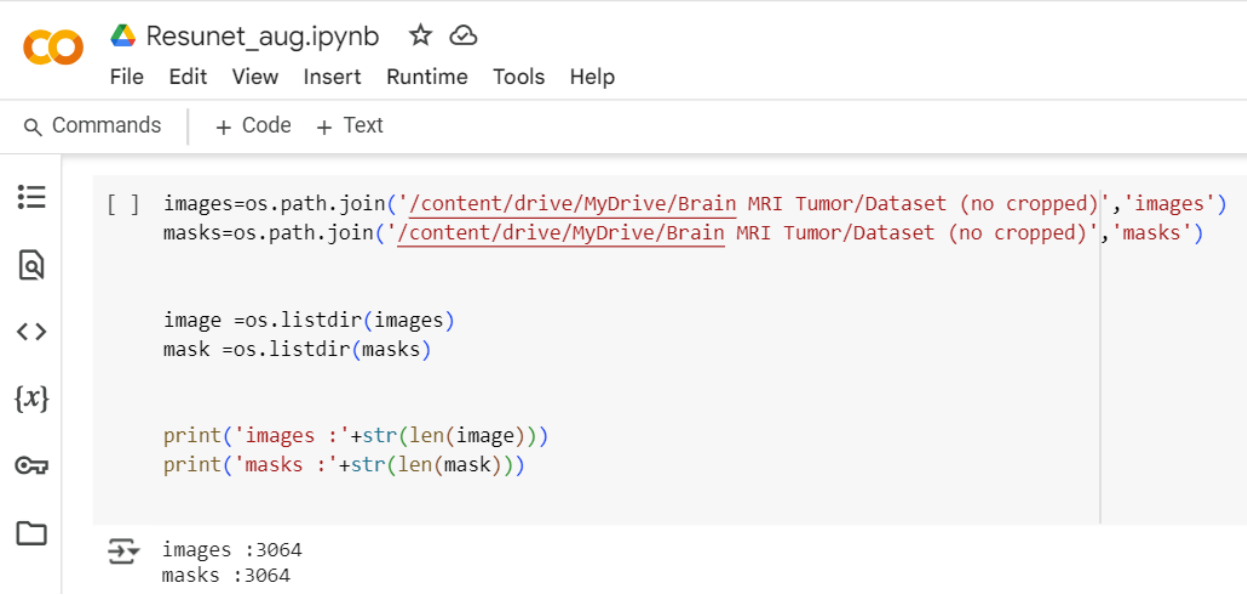


Figure 3.56— Dataset before augmentation.

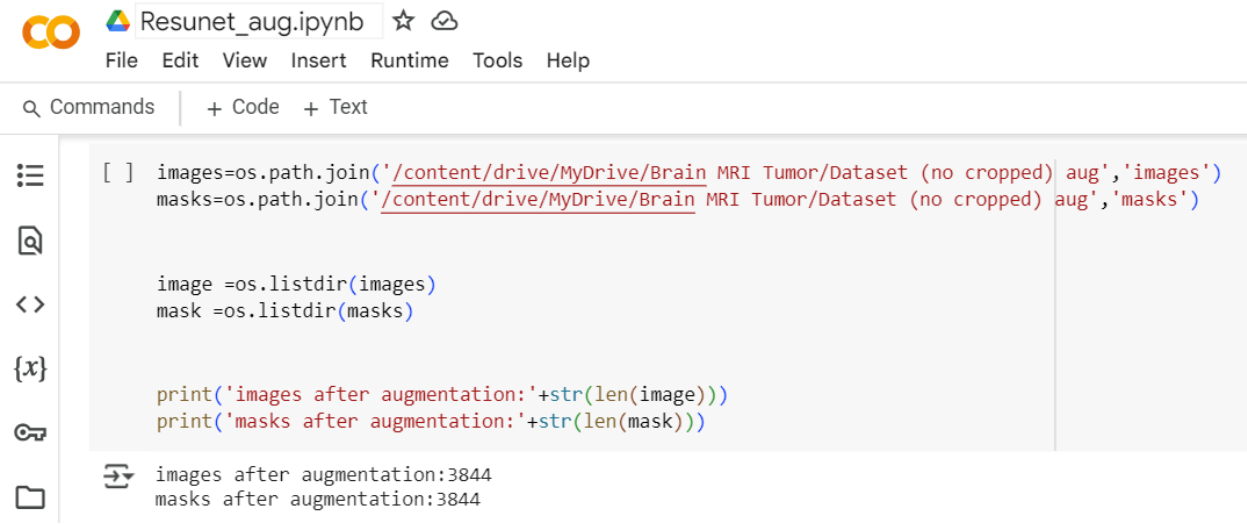


Figure 3.57— Dataset after augmentation.

We notice in these two images (3.45, 3.46) the difference between the number of images in the dataset before and after the data augmentation process.

3.9.1.4 Res-Unet architecture

This is the creation of our Res-Unet model.

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 256, 256, 3)	0	-
conv2d (Conv2D)	(None, 256, 256, 64)	1,792	input_layer[0][0]
batch_normalization (BatchNormalization)	(None, 256, 256, 64)	256	conv2d[0][0]
activation (Activation)	(None, 256, 256, 64)	0	batch_normalization[0…
conv2d_1 (Conv2D)	(None, 256, 256, 64)	36,928	activation[0][0]
batch_normalization_1 (BatchNormalization)	(None, 256, 256, 64)	256	conv2d_1[0][0]
conv2d_2 (Conv2D)	(None, 256, 256, 64)	256	input_layer[0][0]
add (Add)	(None, 256, 256, 64)	0	batch_normalization_1… conv2d_2[0][0]
activation_1 (Activation)	(None, 256, 256, 64)	0	add[0][0]
max_pooling2d (MaxPooling2D)	(None, 128, 128, 64)	0	activation_1[0][0]
conv2d_3 (Conv2D)	(None, 128, 128, 128)	73,856	max_pooling2d[0][0]
batch_normalization_2 (BatchNormalization)	(None, 128, 128, 128)	512	conv2d_3[0][0]
activation_2 (Activation)	(None, 128, 128, 128)	0	batch_normalization_2…
conv2d_4 (Conv2D)	(None, 128, 128, 128)	147,584	activation_2[0][0]
batch_normalization_3 (BatchNormalization)	(None, 128, 128, 128)	512	conv2d_4[0][0]

add_7 (Add)	(None, 128, 128, 128)	0	batch_normalization_1… conv2d_23[0][0]
activation_15 (Activation)	(None, 128, 128, 128)	0	add_7[0][0]
up_sampling2d_3 (UpSampling2D)	(None, 256, 256, 128)	0	activation_15[0][0]
concatenate_3 (Concatenate)	(None, 256, 256, 192)	0	up_sampling2d_3[0][0], activation_1[0][0]
conv2d_24 (Conv2D)	(None, 256, 256, 64)	110,656	concatenate_3[0][0]
batch_normalization_16 (BatchNormalization)	(None, 256, 256, 64)	256	conv2d_24[0][0]
activation_16 (Activation)	(None, 256, 256, 64)	0	batch_normalization_1…
conv2d_25 (Conv2D)	(None, 256, 256, 64)	36,928	activation_16[0][0]
batch_normalization_17 (BatchNormalization)	(None, 256, 256, 64)	256	conv2d_25[0][0]
conv2d_26 (Conv2D)	(None, 256, 256, 64)	12,352	concatenate_3[0][0]
add_8 (Add)	(None, 256, 256, 64)	0	batch_normalization_1… conv2d_26[0][0]
activation_17 (Activation)	(None, 256, 256, 64)	0	add_8[0][0]
conv2d_27 (Conv2D)	(None, 256, 256, 1)	65	activation_17[0][0]

Total params: 33,146,433 (126.44 MB)
Trainable params: 33,134,657 (126.40 MB)
Non-trainable params: 11,776 (46.00 KB)

Figure 3.58— The creation of the model.

3.9.2 Performance evaluation and analysis (discussion of results)

Although we used one model, Res-Unet, we will see the results of some of the models we tried before choosing Res-Unet.

3.9.2.1 Res-Unet

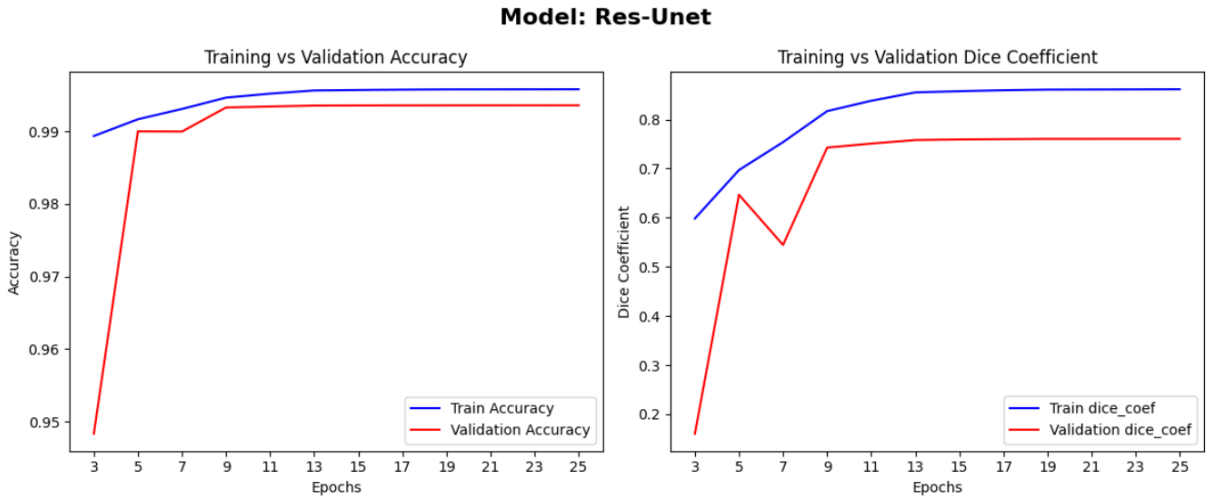


Figure 3.59— The dice_coef curve and the accuracy curve for Res-Unet.

epoch	accuracy	dice_coef	learning_rate	loss	val_accuracy	val_dice_coef	val_loss
1	0.9803817868	0.4159392715	1.00E-04	0.582484901	0.9855237007	0.00126303907	0.9987368584
2	0.9881899357	0.5369921327	1.00E-04	0.4609683454	0.9854490757	0.0003369442129	0.9996630549
3	0.989364624	0.5984022021	1.00E-04	0.4007888734	0.9483339787	0.160428822	0.8395712376
4	0.9907278419	0.6573613286	1.00E-04	0.3420932293	0.9896895885	0.6028084755	0.3971914351
5	0.9916724563	0.6969386935	1.00E-04	0.302205354	0.990008831	0.6466234326	0.353376627
6	0.9924497008	0.7284687757	1.00E-04	0.2708572745	0.9905836582	0.5495477319	0.4504523277
7	0.9930924773	0.7537019849	1.00E-04	0.2453817725	0.989980042	0.5445572734	0.455442667
8	0.9939797521	0.7924262285	1.00E-05	0.2072347254	0.9931152463	0.7314459682	0.2685538828
9	0.9946555495	0.816845417	1.00E-05	0.1830002517	0.9932975769	0.7426063418	0.2573936284
10	0.9949541688	0.8283029795	1.00E-05	0.1715873629	0.9933587909	0.7480530739	0.2519468963
11	0.995200634	0.8378329277	1.00E-05	0.1620656103	0.9934301972	0.7507312298	0.2492686957
12	0.9954252243	0.8465353847	1.00E-05	0.1533658057	0.9935030341	0.7530965805	0.2469034344
13	0.9956419468	0.8549273014	1.00E-06	0.1449421048	0.9935606122	0.7580091357	0.2419908792
14	0.9956774116	0.8562839627	1.00E-06	0.1435866654	0.9935700893	0.7588572502	0.2411425114
15	0.9957040548	0.8573205471	1.00E-06	0.1425529569	0.9935756326	0.7592098117	0.2407902479
16	0.9957306981	0.8583304882	1.00E-06	0.1415448189	0.9935829639	0.7595002055	0.2404998392
17	0.9957573414	0.859347105	1.00E-06	0.1405301094	0.9935891628	0.7597899437	0.2402100414
18	0.995785594	0.8604825139	1.00E-07	0.1393900067	0.9935907722	0.7602718472	0.2397280782
19	0.9957892299	0.8605974317	1.00E-07	0.1392749101	0.9935914874	0.7603671551	0.2396326661
20	0.9957920909	0.8607133627	1.00E-07	0.1391593516	0.9935924411	0.7603853345	0.2396147996
21	0.9957957864	0.860830605	1.00E-07	0.1390423477	0.9935930371	0.760391295	0.2396087199
22	0.9957989454	0.8609494567	1.00E-07	0.1389233619	0.9935936928	0.7603986263	0.2396013886
23	0.9958019257	0.8610707521	1.00E-07	0.1388023198	0.9935941696	0.7604115605	0.2395885587
24	0.9958055019	0.8611941338	1.00E-07	0.1386791021	0.9935953021	0.7604268193	0.2395731807
25	0.9958082438	0.8613194823	1.00E-07	0.1385539919	0.9935966134	0.760446012	0.2395540625

Figure 3.60— CSV Export of Training Logs for Res-Unet.

These are the results of the Res-Unet model, which gave an **0.9958** of accuracy, **0.1385** of loss, training dice_coef **0.8613**, and validation dice_coef **0.7604**.

3.9.2.2 Attention U-Net

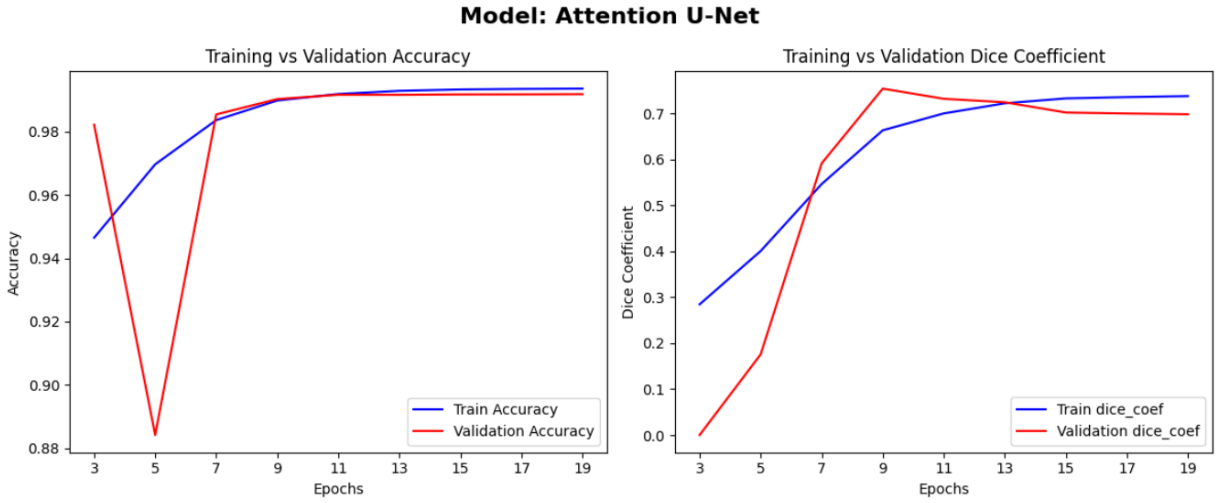


Figure 3.61— The dice_coef curve and the accuracy curve for Attention U-NET.

epoch	accuracy	dice_coef	learning_rate	loss	val_accuracy	val_dice_coef	val_loss
1	0.8450098634	0.1461230069	1.00E-04	0.8538768888	0.9822565317	0.0113776084	0.988612175
2	0.9228816032	0.2260159254	1.00E-04	0.7739840746	0.9822412729	0.0006222074153	0.9993770719
3	0.9464949965	0.2847048938	1.00E-04	0.715295136	0.9822107553	0.0002038845414	0.9997957349
4	0.9601147175	0.3449630439	1.00E-04	0.6550364494	0.9304416776	0.08670932055	0.9123817682
5	0.969694972	0.4007241428	1.00E-04	0.5992760658	0.8841253519	0.1758239418	0.8224113584
6	0.9782437682	0.4773896933	1.00E-04	0.5226103067	0.986004591	0.6070013046	0.3909461498
7	0.9836688638	0.5466272831	1.00E-04	0.4533727467	0.9854739308	0.5917394161	0.4053502679
8	0.986575067	0.6018304825	1.00E-04	0.3981696963	0.9829621911	0.6078193784	0.3947688937
9	0.9898670316	0.6635155678	1.00E-05	0.3364844024	0.9903169274	0.754291594	0.2479742914
10	0.9912266135	0.6875550747	1.00E-05	0.3124449253	0.9903558493	0.7274800539	0.2727259099
11	0.9919146299	0.7003290653	1.00E-05	0.2996711135	0.9916796684	0.7320810556	0.267924279
12	0.992369175	0.7121605873	1.00E-05	0.2878392041	0.9917556643	0.7236577868	0.2760605514
13	0.9929248095	0.722294271	1.00E-05	0.2777053714	0.9916632175	0.724550724	0.2755188644
14	0.9932439327	0.7303920984	1.00E-06	0.2696079016	0.9916355014	0.7073665261	0.2926014066
15	0.9933639765	0.7330074906	1.00E-06	0.2669925392	0.9917860031	0.70228827	0.2974368632
16	0.993478775	0.7346146703	1.00E-06	0.2653851211	0.9918431044	0.7007926702	0.2988384962
17	0.9935302734	0.7358373404	1.00E-06	0.2641626	0.9918052554	0.7001417875	0.2995234132
18	0.9935480952	0.7369724512	1.00E-06	0.2630276382	0.9918882847	0.7012093663	0.2984319925
19	0.9936259389	0.737953186	1.00E-07	0.2620467246	0.9918646812	0.698536396	0.3011035919

Figure 3.62— CSV Export of Training Logs for Attention U-NET.

When training this model, the number of epochs was not completed because the early stopping process stopped the training because of the absence of progress in the validation dice coefficient after 5 epochs. The best result was in epoch 11 where the accuracy was **0.9919**, the loss was **0.2996**, the training dice_coef was **0.7003**, and the validation dice_coef was **0.7320**.

3.9.2.3 U-Net

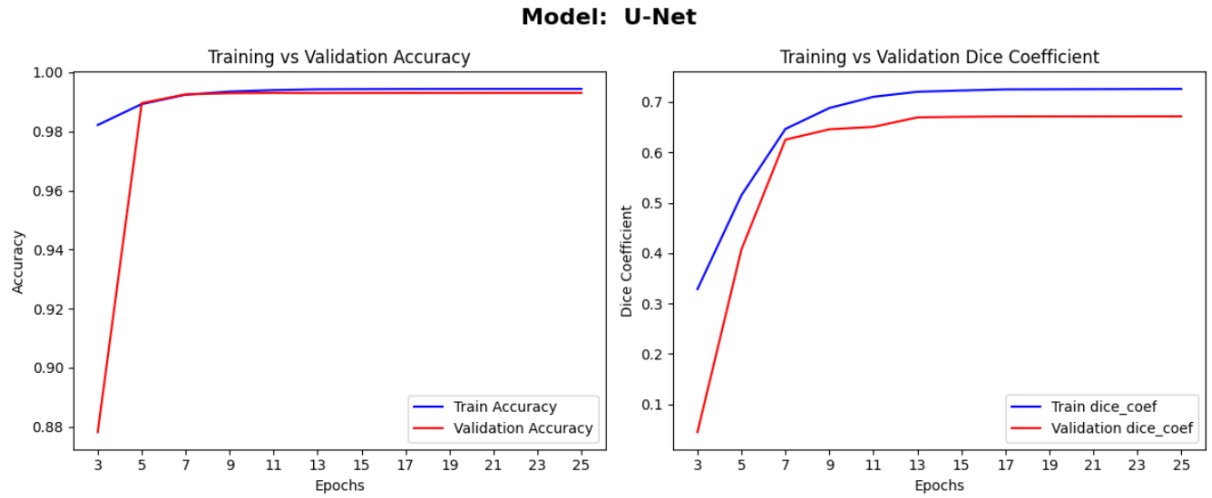


Figure 3.63— The dice_coef curve and the accuracy curve for U-NET.

epoch	accuracy	dice_coef	learning_rate	loss	val_accuracy	val_dice_coef	val_loss
1	0.8755033612	0.1268693805	1.00E-04	0.8737280965	0.666917026	0.0101078162	0.9898920655
2	0.9735860825	0.2358973771	1.00E-04	0.7641083598	0.4701127112	0.01231832802	0.9876815677
3	0.9821898937	0.3287226856	1.00E-04	0.6712055206	0.8782060742	0.04496941343	0.9550305009
4	0.9875227809	0.4348286986	1.00E-04	0.5652271509	0.9857260585	0.3037154377	0.6962844729
5	0.9892819524	0.515104413	1.00E-04	0.4848868847	0.9895787239	0.4073216915	0.5926782489
6	0.9906331897	0.5828794837	1.00E-04	0.4168939292	0.9896375537	0.4658126533	0.5341873765
7	0.9924548268	0.646556437	1.00E-05	0.3531814516	0.9925859571	0.6253197193	0.3746802509
8	0.993191123	0.6752011776	1.00E-05	0.324431777	0.9928584099	0.6384974718	0.3615025282
9	0.9935232997	0.6881361604	1.00E-05	0.3114761412	0.992950201	0.6459271908	0.3540728986
10	0.9938025475	0.7000811696	1.00E-05	0.2995295227	0.9929558635	0.654538393	0.3454615176
11	0.9940174818	0.710392952	1.00E-05	0.2891717851	0.9930544496	0.6507397294	0.3492602408
12	0.9942512512	0.7180898786	1.00E-06	0.2814793289	0.9930226207	0.6676859856	0.3323140144
13	0.9942882657	0.7203966975	1.00E-06	0.2791729271	0.9930126071	0.6695123315	0.3304876983
14	0.9943166375	0.7217175364	1.00E-06	0.2778517604	0.9930288196	0.6700818539	0.3299181163
15	0.9943423271	0.7228963375	1.00E-06	0.2766736746	0.9930410981	0.6706821918	0.3293178976
16	0.9943660498	0.7239501476	1.00E-06	0.2756189406	0.9930565953	0.6710495353	0.3289504945
17	0.9943892956	0.7251230478	1.00E-07	0.2744401991	0.9930527806	0.6713522077	0.3286477625
18	0.9943955541	0.7252514958	1.00E-07	0.2743109763	0.9930543303	0.6714098454	0.3285900652
19	0.9943995476	0.7253727317	1.00E-07	0.2741890252	0.9930558801	0.6714277864	0.3285724223
20	0.9944042563	0.725490272	1.00E-07	0.2740711272	0.993057251	0.671444416	0.3285555542
21	0.9944077134	0.7256054282	1.00E-07	0.2739554048	0.9930593371	0.671467483	0.3285326958
22	0.9944109917	0.7257195711	1.00E-07	0.2738411427	0.9930599332	0.6714954376	0.328504473
23	0.9944140911	0.7258334756	1.00E-07	0.273727566	0.9930601716	0.6715285778	0.3284713924
24	0.9944169521	0.7259465456	1.00E-07	0.2736143768	0.9930618405	0.6715640426	0.3284359276
25	0.9944197536	0.7260594964	1.00E-07	0.2735013366	0.993062973	0.6716036201	0.3283964097

Figure 3.64— CSV Export of Training Logs for U-NET.

These are the outcomes of the U-Net model, which provided training dice_coef **0.7260**, validation dice_coef **0.6716**, accuracy **0.9944**, and loss **0.2735**.

3.10 Comparison of the proposed model

We will compare our proposed classification model with the latest models and modern studies. This comparison includes seeing the results of the performance metrics in two cases: the first for classes only.

Classes/Papers->	**Proposed model**				Model 1 [69] (2024)				Model 2 [70] (2025)			
	accuracy	Precision	Recall	F1-Score	accuracy	Precision	Recall	F1-Score	accuracy	Precision	Recall	F1-Score
Meningioma	0.9954	0.9870	0.9935	0.9902	0.9623	0.93	0.92	0.92	0.9923	0.9803	0.9852	0.9828
Glioma	1.0000	1.0000	1.0000	1.0000	0.9689	0.97	0.96	0.97	0.9934	0.9905	0.9952	0.9929
Pituitary	1.0000	1.0000	1.0000	1.0000	0.9869	0.97	0.98	0.98	0.9945	0.9965	0.9863	0.9914
No tumor	0.9954	0.9933	0.9867	0.9900	1.0000	1.0000	1.0000	1.0000	-	-	-	-

Table 3.3 – Comparison of the Proposed Model with State-of-the-Art Models for Multi-Class Brain Tumor Classification.

In the table 3.3, our proposed model outperformed the other two in classifying glioma and pituitary tumors, while the second model performed better in classifying no_tumor. The performance of the third model was relatively close to that of both our model and the second model.

Ref.	Model	Dataset	Accuracy
[71]2021	Transfer learning with MobileNetV2, VGG19, InceptionV3	4600 brain images (Kaggle dataset)	MobileNetV2: 92.00% VGG19: 88.22% InceptionV3: 91.00%
[72]2020	Customized ResNet50 model	253 MRIs (Kaggle dataset)	97.01%
[73]2020	AlexNet, VGG16, AlexNet+VGG16 with hyper-column techniques and SVM	253 MRIs from Kaggle	AlexNet: 92.47% VGG16: 90.32% AlexNet+VGG16: 96.77%
[74]2021	Multi-scale CNN	Figshare	97.3%
[75]2022	InceptionV3 with Quantum classification	Kaggle data, 2020-BRATS, local collected images	99.44%, 90.91%, 93.33%
[76]2022	Ensemble of vision transformers	Fighshare	98.7%
[77]2022	Inception-ResnetV2 with ADSCFGWO algorithm	BRaTS 2021	99.98%
[78]2023	ResNet with hyper-parameter optimization	Figshare	98.6%
[79]2021	Improved ResNet50 Model	551 healthy and diseased images Kaggle and http://radiopaedia.org/	98.59%
[80]2021	Feature level ensemble of 3 CNNs with PCA feature reduction	Figshare	98.37%
[81]2022	Custom deep neural network model with less hyper-parameters	Figshare	99.18%
[82]2022	3D U-Net and 16-layer CNN	BRATS2020 for segmentation model Kaggle dataset 3264 images for classification model	99.06% 90%
2025	**Proposed model**	**Kaggle Brain tumor MRI dataset**	**99.61%**

Table 3.4 – Accuracy Comparison of Existing Brain Tumor Classification Models and the Proposed Model.

And in this table 3.4, we see the accuracy compared to some models and their different techniques and different datasets used with our proposed model.

3.10 Application interface

We wanted to present our models, which consist of convolutional neural networks, with a graphical interface so we could visualize their performance. We used the streamlit library for this purpose.

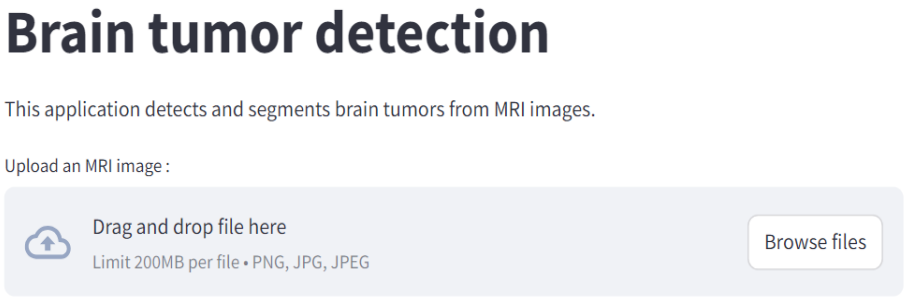


Figure 3.65— Application interface.

In these images we see the interface of our application, where we deliberately placed this image because it contains a small tumor in the pituitary gland, to see how powerful our model is in classifying and segmenting small tumors.

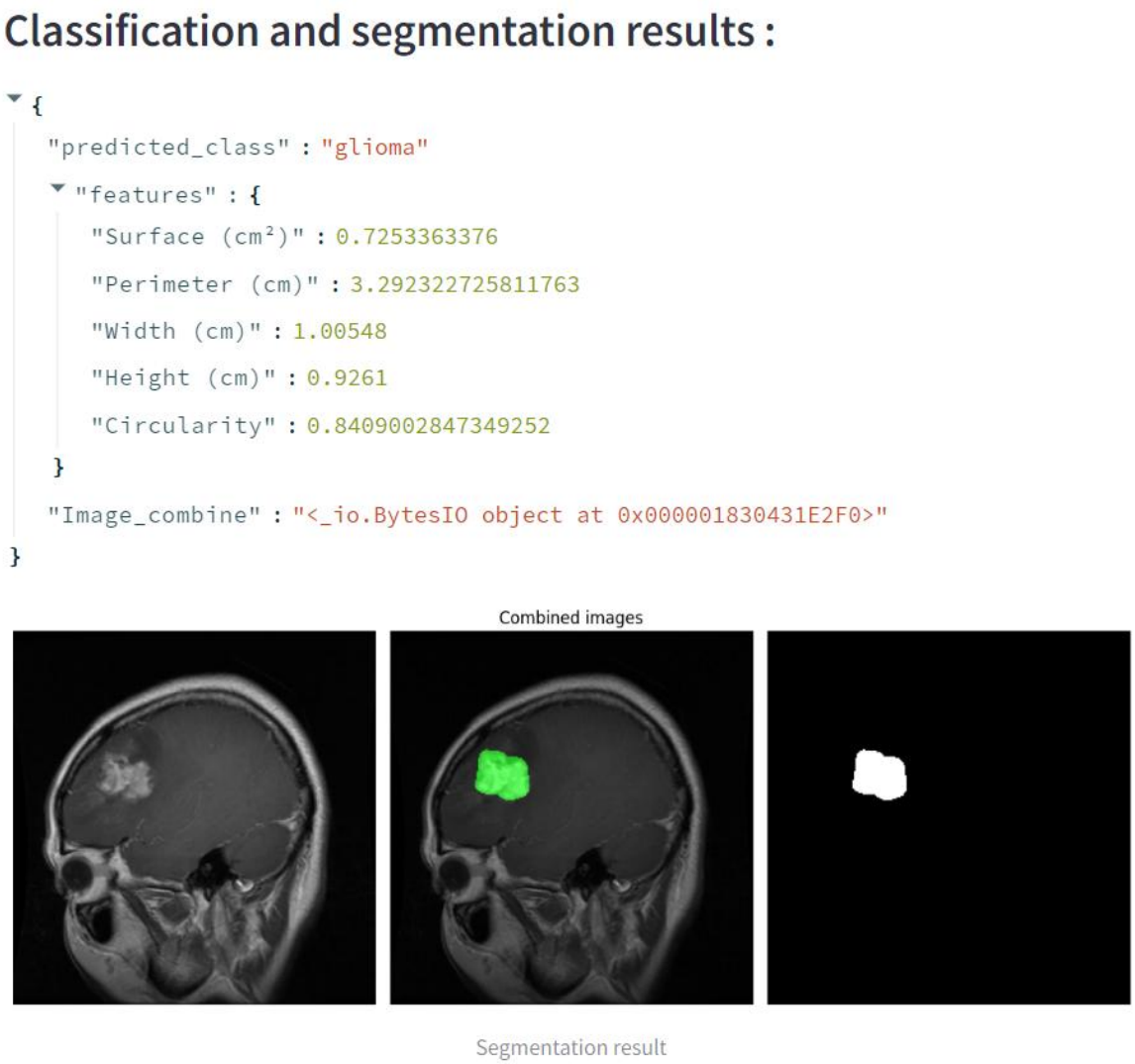


Figure 3.66— An example for glioma tumor.

Here we take an example of a medium-sized glioma.

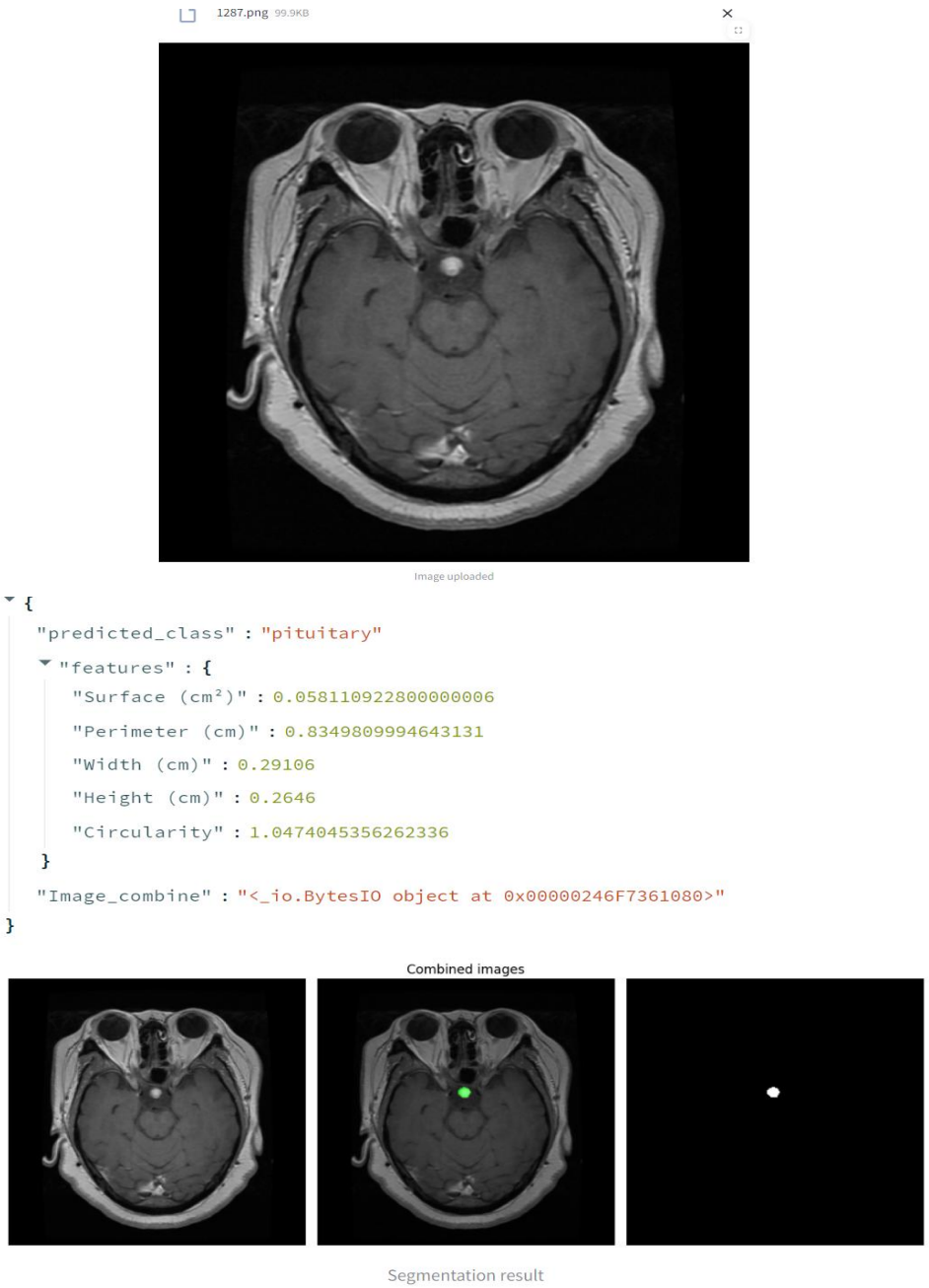


Figure 3.67— An example for an ambiguous image.

We notice the precise segmentation of our model in this ambiguous image, as it contains a tumor, but in the bottom, there is a white substance that is not a tumor, which the model correctly ignored and only highlighted the tumor area.

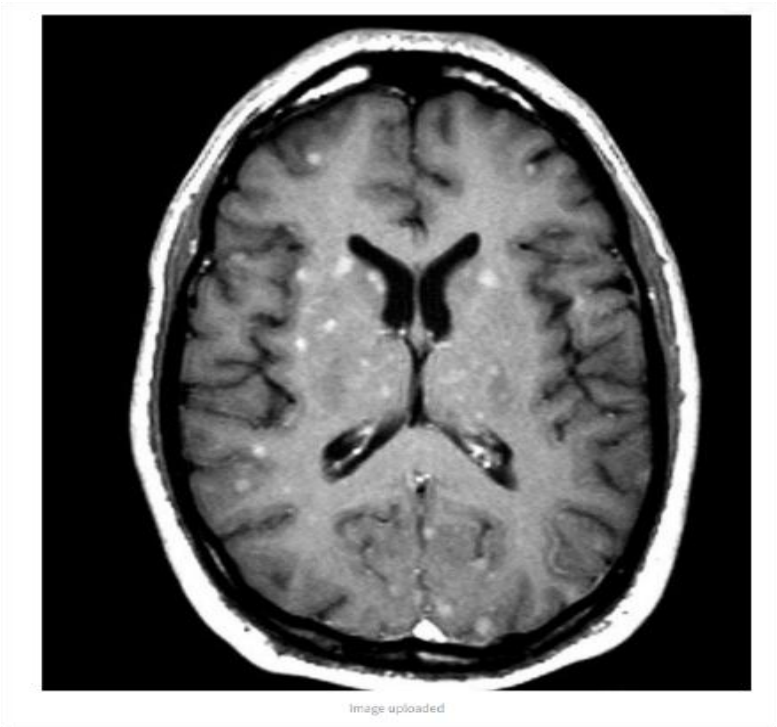


Figure 3.68— An example for no tumor.

We have included this image, which does not contain a tumor, but contains small white dots that some might think are a tumor, but in fact they are chronic, minute lesions in the blood vessels resulting from poor brain perfusion, and they often appear on T2 or FLAIR MRI images, and they are not tumors, and this indicates the strength of our model in classifying tumors.

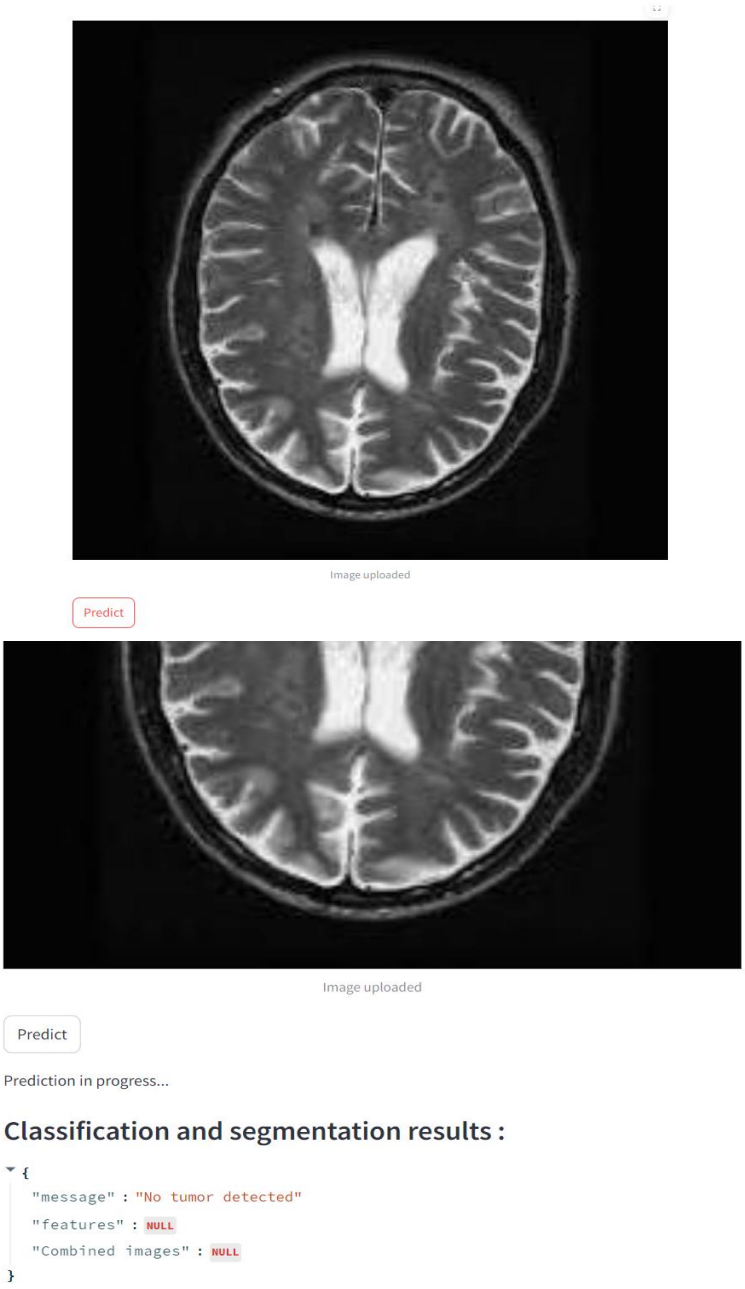


Figure 3.69— An example for no tumor 2.

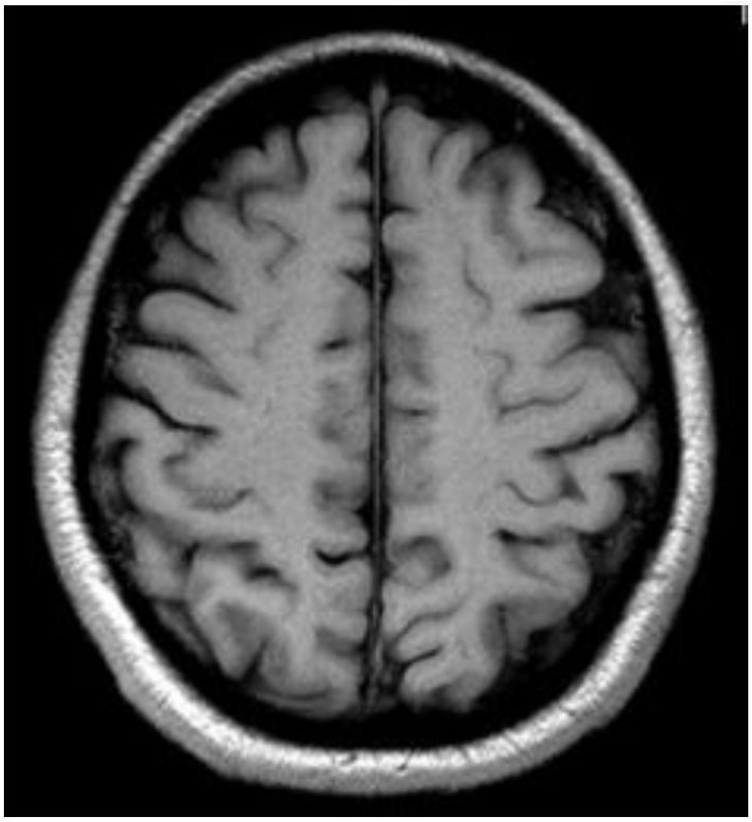


Image uploaded

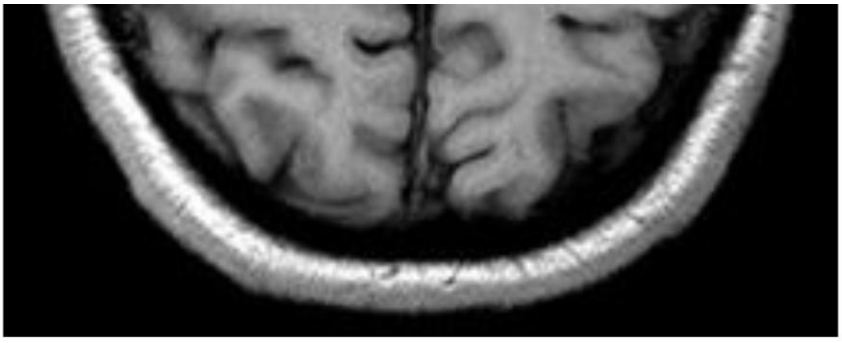


Image uploaded

Predict

Prediction in progress...

Classification and segmentation results :


```
{
  "message" : "No tumor detected"
  "features" : NULL
  "Combined images" : NULL
}
```


Figure 3.70— An example for no tumor 3.

Finally, as we can see, we extracted the following features to better analyze the detected tumor:

Surface area: The number of segmented pixels translated into square centimeters, indicates the size of the tumor. This helps assess the extent of the tumor and monitor its progression over time.
Perimeter: Determines how long the tumor's contour is. This characteristic is helpful in spotting asymmetrical boundaries, which are frequently linked to cancerous growths.

Width: refers to the tumor's greatest horizontal distance. It provides information on the tumor's horizontal axis spread, which is crucial for designing focused treatments.

Height: It is the maximum vertical distance of the tumor. It provides accurate measurements of the tumor, which is important for surgical planning and treatment decisions.

Circularity: Describes the shape of the tumor based on the ratio between its surface area and its perimeter. A perfectly round tumor has a circularity close to 1. Irregular shapes (low circularity) can indicate potential abnormalities that require further investigation.

Conclusion

In this chapter, we presented the implementation and application of our system. Our model shows better accuracy in both training and validation for both classification and segmentation, with these values tending to 1 at a given time. The Dice coefficient in segmentation was good. We used and developed various optimization and regularization techniques such as batch normalization, dropout, and data augmentation to optimize and robust our model. The proposed CNN model can successfully recognize and accurately segment brain images containing tumors and normal brain images, adding some important tumor features that are helpful to the doctor. It has no variability and is fast to learn.

General conclusion

This project has highlighted the effectiveness of artificial intelligence in the detection and segmentation of brain tumors, by offering an automated, fast and accurate solution that can be a valuable tool to help doctors in diagnosis and decision-making. It has also provided me with a rich and motivating personal experience, strengthening my theoretical knowledge, developing my design and programming skills, and allowing me to work on a concrete project with meaning and clear objectives.

Perspectives

The project is currently still under development. We plan to add 3D segmentation to handle all types of MRI scans, improve segmentation accuracy, and integrate other features into the web application, such as tumor stage or progression estimation, to better support physicians in their decision-making.

Bibliography

[6] Chieffo, D. P. R., Lino, F., Ferrarese, D., Belella, D., Della Pepa, G. M., & Doglietto, F. (2023). Brain tumor at diagnosis: From cognition and behavior to quality of life. Diagnostics, 13(3), 541. https://doi.org/10.3390/diagnostics13030541

[14] Rakotoarimanana, R., Randriamaroson, R., & Rastefano, E. (2025). Global temperature prediction by convolutional neural network (CNN). European Journal of Applied Science, Engineering and Technology, 3(1), 99–109. https://ejaset.com/index.php/journal/article/view/177/139

[15] Yamashita, R., Nishio, M., Do, R., & Togashi, K. (2018). Convolutional neural networks: An overview and application in radiology. Insights into Imaging, 9, 611–629. https://doi.org/10.1007/s13244-018-0639-9

[18] Blessing, M., & Surisetti, S. (2024). Optimizing deep learning models for enhanced performance in artificial intelligence systems.

[**21**] Soumare, H., Benkahla, A., & Gmati, N. (2021). Deep learning regularization techniques to genomics data. Array, 11, 100068.

[**23**] Xu, Y., Hou, S., Wang, X., Li, D., & Lu, L. (2023). A medical image segmentation method based on improved UNet 3+ network. Diagnostics, 13(3), 576. https://doi.org/10.3390/diagnostics13030576

[**25**] Papers with Code. (n.d.). AlexNet. Retrieved March 14, 2025. https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[**26**] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556. https://arxiv.org/pdf/1409.1556

[**27**] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 4700-4708).

[**29**] Cheng, H. (2025). Advancements in image classification: From machine learning to deep learning. ITM Web of Conferences, 70, 02016. https://www.itm-conferences.org/articles/itmconf/pdf/2025/01/itmconf_dai2024_02016.pdf

[**31**] Hasanah, S. A., Pravitasari, A. A., Abdullah, A. S., Yulita, I. N., & Asnawi, M. H. (2023). A deep learning review of ResNet architecture for lung disease identification in CXR images. Applied Sciences. https://doi.org/10.3390/app132413111

[**32**] Kumar, V., Prabha, C., Sharma, P., Mittal, N., Askar, S. S., & Abouhawwash, M. (2024). Unified deep learning models for enhanced lung cancer prediction with ResNet-50–101 and EfficientNet-B3 using DICOM images. BMC Medical Imaging, 24.

[**33**] Yang, Z. (2021). Classification of picture art style based on VGGNET. Journal of Physics: Conference Series, 1774(1), 012043. IOP Publishing.

[**35**] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the Inception architecture for computer vision. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2818-2826. https://doi.org/10.1109/CVPR.2016.308

[**37**] Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1251-1258.
https://doi.org/10.1109/CVPR.2017.195
[**38**] Minaee, S., Boykov, Y., Porikli, F. M., Plaza, A. J., Kehtarnavaz, N., & Terzopoulos, D. (2021). Image segmentation using deep learning: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 44(7), 3523-3542.
https://arxiv.org/pdf/2001.05566
[**40**] Liu, S., Qi, L., Qin, H., Shi, J., & Jia, J. (2018). Path aggregation network for instance segmentation. In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 8759-8768).
https://openaccess.thecvf.com/content_cvpr_2018/papers/Liu_Path_Aggregation_Network_CVPR_2018_paper.pdf
[**41**] Papers with Code. (2025). Panoptic segmentation. Papers with Code. Retrieved March 16, 2025.
https://paperswithcode.com/task/panoptic-segmentation
[**42**] Xu, Y., Hou, S., Wang, X., Li, D., & Lu, L. (2023). A medical image segmentation method based on improved UNet 3+ network. Diagnostics, 13(3), 576.
https://doi.org/10.3390/diagnostics13030576
[**43**] Huang, L., Miron, A., Hone, K., & Li, Y. (2024). Segmenting medical images: From UNet to Res-UNet and nnUNet. arXiv preprint arXiv:2407.04353.
https://arxiv.org/pdf/2407.04353
[**45**] Zhang, Z., Liu, Q., & Wang, Y. (2017). Road extraction by deep residual U-Net. IEEE Geoscience and Remote Sensing Letters, 15, 749-753.
[**49**] Mishra, B. K., Thakker, D., Mazumdar, S., & et al. (2020). A novel application of deep learning with image cropping: A smart city use case for flood monitoring. *Journal of Reliable Intelligent Environments, 6*(1), 51–61.
https://doi.org/10.1007/s40860-020-00099-x
[**52**] Zhong, Z., Zheng, L., Kang, G., Li, S., & Yang, Y. (2017). Random Erasing Data Augmentation. *ArXiv, abs/1708.04896*.
https://arxiv.org/pdf/1708.04896
[**53**] Alotaibi, A. (2025). Ensemble deep learning approaches in health care: A review. *Computers, Materials & Continua, 82*(3), 3741–3771.
https://www.techscience.com/cmc/v82n3/59945/pdf
[**54**] Oye, E., Frank, E., & Owen, J. (2024). *Improving APT detection with ensemble learning.*
[**55**] Msoud Nickparvar. (2021). Brain Tumor MRI Dataset [Data set]. Kaggle.
https://doi.org/10.34740/KAGGLE/DSV/2645886
[**61]** Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Zheng, X. (2016). {TensorFlow}: a system for {Large-Scale} machine learning. In 12th USENIX symposium on operating systems design and implementation (OSDI 16) (pp. 265-283).
https://arxiv.org/pdf/1605.08695
[64] Harris, C.R., Millman, K.J., Walt, S.V., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M.H., Brett, M., Haldane, A., R'io, J.F., Wiebe, M., Peterson, P., G'erard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., & Oliphant, T.E. (2020). Array programming with NumPy. *Nature, 585*, 357 – 362.
[65] Tosi, S. (2009). Matplotlib for Python developers. Packt Publishing Ltd.
[66] Mckinney, Wes. (2011). pandas: a Foundational Python Library for Data Analysis and Statistics. Python High Performance Science Computer.

[69] H. Sadr, M. Nazari, S. Yousefzade-Chabok, et al., Enhancing brain tumor classification in MRI images: A deep learning-based approach for accurate classification and diagnosis, Image and Vision Computing (2024).
https://doi.org/10.1016/j.imavis.2025.105555
[70] Chandni, M., Sachdeva, M., & Kushwaha, A. K. S. (2025). AI-based intelligent hybrid framework (BO-DenseXGB) for multi-classification of brain tumor using MRI. Image and Vision Computing, 154, 105417.
https://doi.org/10.1016/j.imavis.2025.105417
[71] Tazin, T., Sarker, S., Gupta, P., Ayaz, F., Islam, S., Khan, M. M., Bourouis, S., Idris, S., & Alshazly, H. (2021). A robust and novel approach for brain tumor classification using convolutional neural network. Computational Intelligence and Neuroscience, 2021, 1–11.
https://doi.org/10.1155/2021/2392395
[72] Çinar, A., & Yildirim, M. (2020). Detection of tumors on brain MRI images using the hybrid convolutional neural network architecture. Medical Hypotheses, 139, 109684.
https://doi.org/10.1016/j.mehy.2020.109684
[73] Togaçar, M., Comert, Z., & Ergen, B. (2020). Classification of brain MRI using hyper column technique with convolutional neural network and feature selection method. Expert Systems with Applications, 149, 113274.
https://doi.org/10.1016/j.eswa.2020.113274
[74] Díaz-Pernas, F. J., Martínez-Zarzuela, M., Antón-Rodríguez, M., & González-Ortega, D. (2021). A deep learning approach for brain tumor classification and segmentation using a multiscale convolutional neural network. Healthcare, 9(2), 153.
[75] Amin, J., Anjum, M. A., Sharif, M., Jabeen, S., Kadry, S., & Moreno-Ger, P. (2022). A new model for brain tumor detection using ensemble transfer learning and quantum variational classifier. Computational Intelligence and Neuroscience, 2022.
[76] Tummala, S., Kadry, S., Bukhari, S. A. C., & Rauf, H. T. (2022). Classification of brain tumor from magnetic resonance imaging using vision transformers ensembling. *Current Oncology, 29*(10), 7498–7511.
[77] ZainEldin, H., Gamel, S. A., El-Kenawy, E. S. M., Alharbi, A. H., Khafaga, D. S., Ibrahim, A., & Talaat, F. M. (2022). Brain tumor detection and classification using deep learning and sine-cosine fitness Grey wolf optimization. Bioengineering, 10(1), 18.
https://doi.org/10.3390/bioengineering10010018
[78] Mehnatkesh, H., Jalali, S. M. J., Khosravi, A., & Nahavandi, S. (2023). An intelligent driven deep residual learning framework for brain tumor classification using MRI images. Expert Systems with Applications, 213, 119087 .
https://doi.org/10.1016/j.eswa.2023.119087
[79] Li, L., Li, S., & Su, J. (2021). A multi-category brain tumor classification method based on improved ResNet50. CMC-Computers, Materials & Continua, 69(2), 2355–2366.
[80] Aurna, N. F., Yousuf, M. A., & Taher, K. A. (2021). Multi-classification of brain tumors via feature level ensemble of convolutional neural networks. In 2021 3rd International Conference on Sustainable Technologies for Industry 4.0 (STI) (pp. 1–6).
[81] Polat, O., Dokur, Z., & Olmez, T. (2022). Brain tumor classification by using a novel convolutional neural network structure. International Journal of Imaging Systems and Technology, 32(5), 1646–1660.
[82] Agrawal, P., Katal, N., & Hooda, N. (2022). Segmentation and classification of brain tumor using 3D-UNet deep neural networks. International Journal of Cognitive Computing and Engineering, 3, 199–210.

Webography

[1] Cleveland Clinic. (n.d.). Tumor Retrieved February 27, 2025.
https://my.clevelandclinic.org/health/diseases/21881-tumor
[2] Mayo Clinic. (2024, December). Brain tumor – Symptoms and causes. Retrieved March 2, 2025.
https://www.mayoclinic.org/diseases-conditions/brain-tumor/symptoms-causes/syc-20350084
[3] Johns Hopkins Medicine. (n.d.). Gliomas: Symptoms, diagnosis, and treatment. Retrieved February 27, 2025.
https://www.hopkinsmedicine.org/health/conditions-and-diseases/gliomas
[4] Cancer Research UK. (n.d.). Meningioma – A type of brain tumor. Retrieved February 27, 2025.
https://www.cancerresearchuk.org/about-cancer/brain-tumours/types/meningioma
[5] American Cancer Society. (n.d.). What is a pituitary tumor? Retrieved February 27, 2025.
https://www.cancer.org/cancer/types/pituitary-tumors/about/what-is-pituitary-tumor.html
[7] Mayo Clinic. (n.d.). Magnetic resonance imaging (MRI). Mayo Clinic. Retrieved March 2, 2025.
https://www.mayoclinic.org/tests-procedures/mri/about/pac-20384768
[8] Case Western Reserve University. (n.d.). MRI basics. Case Western Reserve University School of Medicine. Retrieved March 2, 2025.
https://case.edu/med/neurology/NR/MRI%20Basics.htm
[9] Medicosis Perfectionalis. (2021, March 11). MRI sequences: T1 vs T2 vs FLAIR - Radiology basics [Video]. YouTube. Retrieved March 2, 2025.
https://www.youtube.com/watch?v=ka-HBXP5ERA
[10] Cleveland Clinic. (n.d.). Biopsy overview. Retrieved March 2, 2025.
https://my.clevelandclinic.org/health/diagnostics/15458-biopsy-overview
[11] Johns Hopkins Medicine. (n.d.). Electroencephalogram (EEG). Retrieved March 2, 2025.
https://www.hopkinsmedicine.org/health/treatment-tests-and-therapies/electroencephalogram-eeg
[12] ScienceDirect. (n.d.). Fluorescence Imaging. Retrieved March 2, 2025.
https://www.sciencedirect.com/topics/nursing-and-health-professions/fluorescence-imaging
[13] Jain, A. (n.d.). Difference Between ANN and CNN. Medium. Retrieved March 2, 2025.
https://medium.com/@abhishekjainindore24/difference-between-ann-and-cnn-ada6303171df
[16] GeeksforGeeks. (2025, March 5). CNN – Introduction to pooling layer. GeeksforGeeks.
https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/
[17] Zilliz. (2025, March 5). What is a fully connected layer in deep learning? Zilliz AI FAQ.
https://zilliz.com/ai-faq/what-is-a-fully-connected-layer-in-deep-learning
[19] ScienceDirect. (n.d.). Confusion matrix. Retrieved March 6, 2025.
https://www.sciencedirect.com/topics/computer-science/confusion-matrix
[**20**] Nerdjock. (n.d.). Deep Learning Course Lesson 11: Model Evaluation Metrics. Medium. Retrieved March 6, 2025.
https://medium.com/@nerdjock/deep-learning-course-lesson-11-model-evaluation-metrics-d85d0b85bcca.
[**22**] Data Science Journey. (2025, March 6). Regularization techniques in deep learning. Medium.
https://medium.com/@datasciencejourney100_83560/regularization-techniques-in-deep-learning-3de958b14fba

[**24**] Bhandari, S. (2020, October 8). LeNet-5 architecture explained. Medium. Retrieved March 14, 2025. https://medium.com/@siddheshb008/lenet-5-architecture-explained-3b559cb2d52b
[**28**] Roboflow. (n.d.). What is EfficientNet? Roboflow Blog. Retrieved March 14, 2025. https://blog.roboflow.com/what-is-efficientnet/
[**30**] SuperAnnotate. (2023). Image classification basics. SuperAnnotate Blog. Retrieved March 12, 2025. https://www.superannotate.com/blog/image-classification-basics
[**34**] GeeksforGeeks. (2025, March 6). VGG-Net architecture explained. GeeksforGeeks. https://www.geeksforgeeks.org/vgg-net-architecture-explained/
[**36**] DigitalOcean. (2025, March 7). Popular deep learning architectures: ResNet, InceptionV3, SqueezeNet. DigitalOcean Community. https://www.digitalocean.com/community/tutorials/popular-deep-learning-architectures-resnet-inceptionv3-squeezenet#inception-v3-2015
[**39**] SuperAnnotate. (2023, November 22). Image segmentation detailed overview [Updated 2024]. SuperAnnotate Blog. Retrieved March 16, 2025. https://www.superannotate.com/blog/image-segmentation-for-machine-learning
[**44**] Ito Aramendia, A. (2023, October 30). Decoding the U-Net: A complete guide. Medium. Retrieved March 17, 2025. https://medium.com/@alejandro.itoaramendia/decoding-the-u-net-a-complete-guide-810b1c6d56d8
[**46**] Ito Aramendia, A. (2021, February 21). A detailed explanation of the Attention U-Net. Medium. Retrieved March 17, 2025. https://medium.com/data-science/a-detailed-explanation-of-the-attention-u-net-b371a5590831
[**47**] GeeksforGeeks. (2023, September 13). U-Net++ Architecture Explained. Retrieved March 17, 2025. https://www.geeksforgeeks.org/unet-architecture-explained/
[**48**] Keylabs. (n.d.). *Best practices for image preprocessing in image classification.* Keylabs. Retrieved March 24, 2025. https://keylabs.ai/blog/best-practices-for-image-preprocessing-in-image-classification/
[**50**] GeeksforGeeks. (n.d.). *Normalize an image in OpenCV - Python.* Retrieved March 24, 2025. https://www.geeksforgeeks.org/normalize-an-image-in-opencv-python/
[**51**] Kashyap, P. (n.d.). *Image normalization in PyTorch: From tensor conversion to scaling.* Medium. Retrieved March 24, 2025. https://medium.com/@piyushkashyap045/image-normalization-in-pytorch-from-tensor-conversion-to-scaling-3951b6337bc8
[56] Hopsworks. (n.d.). *Training data*. Hopsworks AI. Retrieved March 28, 2025. https://www.hopsworks.ai/dictionary/training-data
[**57**] Eastgate. (2019, April 30). *Understanding training, validation and testing data in ML*. Medium. Retrieved March 28, 2025. https://medium.com/@eastgate/understanding-training-validation-and-testing-data-in-ml-c297e53d6096
[**58**] Tomar, N. (n.d.). *Brain tumor segmentation* [Data set]. Kaggle. Retrieved March 28, 2025. https://www.kaggle.com/datasets/nikhilroxtomar/brain-tumor-segmentation
[**59**] Python Software Foundation. (n.d.). *General Python FAQ: What is Python?* Python.org. Retrieved April 1, 2025. https://www.python.org/doc/essays/blurb/

[**60**] Coursera. (n.d.). *What is Kaggle? Your guide to machine learning competitions*. Retrieved April 1, 2025.
https://www.coursera.org/articles/kaggle
[**62]** Chollet, F. (2015). *Keras: Deep learning for humans*. GitHub. Retrieved April 1, 2025.
https://keras.io
[**63]** OpenCV.org. (n.d.). *About OpenCV*. Retrieved April 1, 2025.
https://opencv.org/about/
[67] DataCamp. (n.d.). *Streamlit tutorial: A guide to building interactive web apps with Python*. Retrieved April 1, 2025.
https://www.datacamp.com/tutorial/streamlit
[68] Google. (n.d.). *Google Colaboratory*. Retrieved April 1, 2025.