UNIVERSITY OF BAHRAIN

College of Engineering

Department of Electrical & Electronics

Brain Tumor Classification and Segmentation Using Deep Learning Techniques

A Thesis Submitted in Partial Fulfillment of the Requirements for the M.Sc. Degree in Artificial Intelligence Systems

Submitted by:

Jawaher Adel Ali Albanki

20143551

Supervised by:

Prof. Ebrahim Mattar

Kingdom of Bahrain

May 2023

University of Bahrain

Collage of Engineering

Department of Electrical and Electronic Engineering

Brain Tumor Classification and Segmentation Using Deep Learning Techniques

A Thesis Submitted in Partial Fulfillment of the Requirement for the M.Sc. Degree in Artificial Intelligent Systems

Submitted by
Jawaher Adel Ali Albanki
20143551

Supervised by
Prof. Ebrahim Abdulla Mattar

May 2023

Defense Committee's Approval

University of Bahrain
College of Engineering
Department of Electrical and Electronics Engineering
Master of Science in Artificial Intelligent Systems

Student Jawaher Adel Ali Zayed Albanki ID number 20143551 has defended her thesis by which she has fulfilled the requirement for the degree of Master of Science in Artificial Intelligent Systems (Brain Tumor Classification and Segmentation Using Deep Learning Techniques) on Tuesday, May 16, 2023.

The committee recommends awarding of M.Sc. Degree with the grade of: Pass

(Defense Committee)

Name		Signature
Professor Ebrahim A Mattar	Supervisor	Ebrahim Mattar
Professor Hasan Al-Nashash	External examiner	Hasan Al-Nashash
Professor Mohab A. Mangoud	Internal examiner	M. Mangoud

Academic department's stamp

Abstract

Recent advancements in deep learning techniques have significantly improved the diagnosis of brain tumors by enabling the development of powerful models. This thesis aims to contribute to the advancement of brain tumor diagnosis systems by exploring and improving various recent deep learning techniques, with a focus on two tasks: classification and segmentation.

The first task comprises a sequence of three extensive studies, involving a thorough evaluation, analysis of results, and comparisons. In the first study, the performance of pre-trained and non-pretrained benchmark models such as VGG and ResNet is compared to custom-created CNN and Vision Transformer models. The second study applies the Model Soup algorithm to select highly performing models from the first study to improve accuracy. In the third study, a pre-trained CoAtNet model is employed for brain tumor classification and enhanced using techniques such as augmentations, large batch sizing, normalization, and exponentially decaying learning rate. Our model achieved the highest accuracy of 99.16%, surpassing state-of-the-art models, and our results were compared with previous literature results.

In the second task, we evaluate the performance of multiple U-Net models with a minimal skip-connections configuration on the segmentation of glioblastoma, a specific type of malignant brain tumor. The results showed that eliminating the first path results in the optimal U-Net skip-connection configuration.

Overall, this thesis employs a rigorous approach to explore the potential of deep learning techniques for brain tumor diagnosis, and the results have significant implications for improving the accuracy of brain tumor diagnosis.

Publications

Conference Publications

{1} Jawaher Albanki and Ebrahim Mattar. (2023), '*A Study on Effects of U-Net Skip-connections on Brain Tumor BraTS Dataset*', In proceeding of UKSim-AMSS 25th International Conference on Modelling & Simulation, Cambridge, UK, April-2023

Journal Publications

{1} Jawaher Albanki and Ebrahim Mattar. (2023), 'A Study on Effects of U-Net Skip-connections on Brain Tumor BraTS Dataset', International Journal of Simulation: Systems, Science & Technology, vol. 24(no. 1), April-2023, doi:10.5013/IJSSST.a.24.01.06

{2} Jawaher Albanki and Ebrahim Mattar. (2023), '*Classification of Brain Tumor Using CoAtNet Model*'—submitted to MDPI AI, under review.

{3} Jawaher Albanki and Ebrahim Mattar. (2023), '*Use of U-Net and its Variants in MRI Brain Tumors Segmentation*'—submitted to IEEE Access, under review.

Dedication

I would like to dedicate this work to my family.

Acknowledgments

I would like to take this opportunity to express my sincere gratitude to my esteemed thesis supervisor, Prof. Ebrahim Mattar, for his invaluable guidance and unwavering support throughout my thesis work. His proficient supervision has been instrumental in my success in this research journey. Under his guidance, I have been provided with essential resources and directed towards the right path to accomplish my work. Prof. Ebrahim Matter has been exceptionally generous with his time despite his busy schedule. During the course of my work, Prof. Ebrahim Mattar has always been a source of inspiration, encouraging me to think outside the box and have a researcher's mindset. He has urged me to write papers and publish them in reputable journals and conferences, which is invaluable to me as somebody who is in the beginning of their researcher track. I am immensely fortunate to have been one of his students, and greatly thankful for his patience and understanding throughout the process.

I would also like to extend my acknowledgement and appreciation to all of the distinguished doctors who taught me during my master's courses, namely, Prof. Mohab Mangood, Dr. Mohammad Alkhaldi, Dr. Zouhair Bahri, and once again, Prof Ebrahim Mattar. Their courses have been enriching, and they have all made sure that we read as many papers as possible and gave us plenty of practical projects and assignments that have enhanced our skills. The strong foundation in AI that we acquired has helped a lot in my research, reading, and especially programming.

Lastly, I would like to express my deepest gratitude and appreciation to my family, who have been a constant source of love, encouragement, and support. Their unwavering support has been a pillar of strength throughout to me without

which I would not have reached to this milestone. I am grateful for their sacrifices and commitment to my success.

In particular, I would like to dedicate this thesis to my beloved mother, who passed away three years ago. Her love and unwavering support have been a driving force behind my academic pursuits. Though she is no longer with us, her memory will forever live on in my heart, and I am grateful for the time we spent together. I would also like to express my gratitude to my father and siblings, who have been a constant source of love and support. Their guidance and encouragement have been instrumental in helping me achieve this milestone, and I am forever indebted to them.

Finally, I would like to thank all my friends and colleagues who have supported me throughout my academic journey. Their encouragement and support have been a constant source of motivation, and I am grateful for their contributions to my success.

Table of Contents

Defense Committee's Approval ..... A

Abstract ..... B

Publications C

Dedication ..... D

Acknowledgments ..... E

List of Tables ..... M

List of Figures ..... N

List of Abbreviations ..... S

List of Appendices ..... T

Chapter One

Introduction

1.1 Motivation 1

1.2 Aim of Thesis 2

1.3 Method 2

1.4 Research Question 5

1.5 Classification Dataset 6

1.6 Segmentation Dataset 8

1.7 Tools 10

1.8 Layout 10

Chapter Two

Theoretical Background and Literature Review

2.1 Computer vision 13

2.2 Multi-layer perceptron 14

2.3 Convolutional Neural Networks 17
2.3.1 Padding 19
2.3.2 Strides 20
2.3.3 Feature map 20
2.3.4 Pooling layers 20
2.3.5 Benchmarking and pre-trained models 21
2.4 Vision Transformers 27
2.4.1 Patch 29
2.4.2 Patch Embedding and flattening. 29
2.4.3 Positional Embedding 30
2.4.4 Multi-head Attention (MSA) 30
2.4.5 Multi-layer Perceptron MLP 31
2.4.6 Inductive Bias 31
2.4.7 Computational Cost 32
2.4.8 Swin Transformer 32
2.5 Loss Functions 33
2.5.1 Cross Entropy 33
2.5.2 Dice Loss 34
2.6 Optimization 34
2.6.1 Gradient descent 35
2.6.2 Adadelta 35
2.6.3 RMS-prop 36
2.6.4 ADAM 37
2.7 Metrices 38

2.7.1 Classification Metrices 38
2.8 Augmentation 40
2.9 Related Work 42
2.9.1 Use of Convolutional Neural Networks 43
2.9.2 Use of Capsule Neural Networks 45
2.9.3 Use of Transfer learning 47
2.9.4 Use of Generative Adversarial Networks 48
2.9.5 Use of Vision Transformers 51
2.10 Challenges 52

Chapter Three

Transfer Learning and Custom Models Studies

3.1 Setup 54
3.2 Non-pretrained 57
3.2.1 VGG16 58
3.2.2 ResNet101 59
3.2.3 ResNet50 60
3.2.4 Simple Custom CNN 61
3.2.5 Swin ViT 63
3.3 Pre-training 66
3.3.1 VGG16 67
3.3.2 ResNet50 67
3.3.3 ResNet101 68
3.4 Theory behind CoAtNet 69
3.4.1 MBConv block 69

3.4.2 Relative self-attention 70
3.4.3 Model architecture 70
3.5 Study Setup 73
3.6 Results of CoAtNet 74
3.6.1 Non-pretrained model 75
3.6.2 pre-trained model 77
3.7 Discussion 91

Chapter Four

Model Soups Algorithm

4.1 Introduction 93
4.2 Vision Transformers with Model Soup Algorithm 94
4.2.1 Results of soups 96
4.3 ResNet Souping Algorithm 101
4.3.1 Results of soups 102

Chapter Five

Discussion and Analysis on Classification of Brain Tumor

5.1 Discussion on initial studies 106
5.2 Discussion on Soup algorithm 109
5.3 Discussion on CoAtNet model 111
5.4 Comparison to previous literature work 113
5.5 Future work 114

Chapter Six

A Study on Effect of U-Net Skip-connection Alteration on Glioblastoma Segmentation

6.1 U-Net Model 118
6.2 Survey 120
6.2.1 Architecture Modification 120
6.2.2 Regularization 126
6.2.3 Ensemble Learning 128
6.3 Related work 129
6.4 Method 131
6.4.1 Skip-connection modifications 132
6.5 Setup 132
6.6 Results 133
6.6.1 Visualizations 133
6.6.2 Summary 143
6.6.3 Discussion 144

Chapter Seven

Conclusion

7.1 Summary and Conclusion 147
7.2 Direction for Future Research 148
7.2.1 Further Studies 148
7.2.2 Semi-supervised Learning 149
7.2.3 Dataset collection 150
7.3 Answer to Research Questions 150
7.4 Conclusion 152
References 153
Arabic Abstract 169

List of Tables

1.1 Classification core dataset 6

1.2 Classification extended dataset 8

4.1 Custom CNN results 62

4.2 Summary of custom CNN results 63

4.3 Custom Swin-ViT results 64

4.4 Summary of custom Swin-ViT results 66

4.5 Pre-trained VGG16 results 67

4.6 Pre-trained ResNet50 results 67

4.7 Pre-trained ResNet101 results 68

6.8 Samples of classification of highest performing CoAtNet 87

6.9 Summary of CoAtNet results 92

5.1 Model numbering convention 96

5.2 Results of custom Swin-ViT soups 97

5.3 Results of Pre-trained ResNet50 soups 102

7.1 Summary of Non-pretrained, Custom, and Pre-trained models' accuracy results 108

7.2 Summary of Model Soups algorithm accuracy results 110

7.3 Summary of CoAtNet accuracy results 112

7.4 Comparison our results with literature 113

8.1 U-Net variations dice metric and loss visualizations 134

8.2 Samples of segmentation 140

8.3 Summary and comparison of dice scores 143

List of Figures

1.1 Sample from classification dataset. 7

1.2 A sample of a manually segmented glioblastoma. 9

2.1 The basic architecture of the perceptron. 14

2.2 multi-layer perceptron. 16

2.3 Activation functions 17

2.4 An example of a convolutional operation. 18

2.5 (Top) a convolutional neural network, (Bottom) a dense neural network. 18

2.6 An example of shift invariance in a segmentation convolutional neural network. 19

2.7 Example of padding. 19

2.8 example of max pooling operation. 21

2.9 progression of models' accuracy on ImageNet dataset 22

2.10 AlexNet model architecture 23

2.11 VGG family of architectures 25

2.12 ResNet compared with other architectures. 26

2.13 A residual block 26

2.14 ResNet family of architectures. 26

2.15 Vision transformer architecture diagram 28

2.16 Window partitioning illustration. 33

2.17 CNN with parallel cascades proposed for brain tumor classification. 43

2.18 a convolutional-based network proposed for brain tumor classification..44

2.19 capsule network architecture performance on brain tumor classification. 47

2.20 Improved Bayesian capsule network brain tumor classification model. 47

2.21 images generated by proposed VAE-GAN architecture after 100, 500, 1000, 15,000, and 20,000 training epochs. 50

2.22 Proposed method to use GAN and VEA as augmenters. 51
2.23 ensemble vision transformer performance on brain tumor classification. 52
3.1 comparison of different optimizer performances on CIFAR10 dataset. 57
3.2 VGG without pre-training accuracy. 59
3.3 VGG without pre-training loss. 59
3.4 ResNet101 without pre-training accuracy 60
3.5 ResNet101 without pre-training loss 60
3.6 ResNet50 without pre-training accuracy 61
3.7 ResNet50 without pre-training loss 61
3.8 Accuracy of 2-layer CNN. 62
3.9 Loss of 2-layer CNN. 62
3.10 Accuracy of 3-layer CNN. 62
3.11 Loss of 3-layer CNN. 62
3.12 Accuracy of 4-layer CNN. 62
3.13 Loss of 4-layer CNN. 62
3.14 Accuracy of 5-layer CNN. 63
3.15 Loss of 5-layer CNN. 63
3.16 Accuracy of 6-layer CNN. 63
3.17 Loss of 6-layer CNN. 63
3.18 Accuracy of 1-block Swin-ViT. 64
3.19 Loss of 1-block Swin-ViT. 64
3.20 Accuracy of 2-block Swin-ViT. 64
3.21 Loss of 2-block Swin-ViT. 64
3.22 Accuracy of 3-block Swin-ViT. 65
3.23 Loss of 3-block Swin-ViT. 65
3.24 Accuracy of 4-block Swin-ViT. 65
3.25 Loss of 4-block Swin-ViT. 65
3.26 Accuracy of 5-block Swin-ViT. 65
3.27 Loss of 5-block Swin-ViT. 65

3.28 Accuracy of 6-block Swin-ViT. 65
3.29 Loss of 6-block Swin-ViT. 65
3.30 Accuracy of Pre-trained VGG-16. 67
3.31 Loss of Pre-trained VGG-16. 67
3.32 Accuracy of pre-trained ResNet-50. 68
3.33 Loss of Pre-trained ResNet-50. 68
3.34 Accuracy of Pre-trained ResNet-101. 68
3.35 Loss of Pre-trained ResNet-101. 68
3.36 CoAtNet Architecture and performance on ImageNet. 72
3.37 Properties reserved in CoAtNet. 73
3.38 Non-pre-trained CoAtNet Accuracy. 76
3.39 Non-pretrained CoAtNet Loss. 76
3.40 non-pre-trained CoAtNet confusion matrix. 76
3.41 non-pre-trained CoAtNet table of results. 76
3.42 plain pre-trained CoAtNet Accuracy. 78
3.43 plain pre-trained CoAtNet Loss. 78
3.44 plain pre-trained CoAtNet confusion matrix. 78
3.45 plain pre-trained CoAtNet table of results. 78
3.46 second scenario accuracy. 80
3.47 second scenario loss. 80
3.48 second scenario confusion matrix. 80
3.49 second scenario table of results. 80
3.50 third scenario accuracy. 82
3.51 third scenario loss. 82
3.52 third scenario confusion matrix. 82
3.53 third scenario results table. 82
3.54 fourth scenario accuracy. 83
3.55 fourth scenario loss. 83
3.56 fourth scenario confusion matrix. 83

3.57 fourth scenario results table. 84
3.58 fifth scenario accuracy. 85
3.59 fifth scenario loss. 85
3.60 fifth scenario confusion matrix. 85
3.61 fifth scenario results table. 85
3.62 sixth scenario accuracy. 86
3.63 sixth scenario loss. 86
3.64 sixth scenario confusion matrix. 86
3.65 sixth scenario results table. 87
3.66 CoAtNet performance summary with different modifications 92
4.1 confusion matrix for Swin-ViT uniform soup. 99
4.2 results table for Swin-ViT uniform soup. 99
4.3 confusion matrix for Swin-ViT greedy soup. 100
4.4 results table for Swin-ViT uniform soup. 100
4.5 confusion matrix for ResNet50 uniform soup. 104
4.6 results table for ResNet50 uniform soup. 104
4.7 Confusion matrix and results table for ResNet50 greedy soup. 105
4.8 Confusion matrix and results table for ResNet50 while eliminating below 90% models. 105
6.1 U-Net network arcitecture. Blue boxese represents multi-channel feature map. 119
6.2 TransBTS overall architecture. 123
6.3 Swin UNETR architecture. 124
6.4 BiTr-Unet architecture. 125
6.5 Evolution from U-Net to U-Net++. 126
6.6 SegResNet model with the variational auto-encoder implemented during training phase to add regularization. 128
6.7 ablation study to investigate skip-connection effect. Plot shows the dice score vs. model configuration. 131

6.8 U-Net training profile. 134
6.9 No-skips U-Net training profile. 135
6.10 L1-only training profile. 135
6.11 L2-only training profile. 136
6.12 L3-only training profile. 137
6.13 L4-only training profile. 137
6.14 No-L1 training profile. 138
6.15 No-L2 training profile. 139
6.16 No-L3 training profile. 139
6.17 No-L4 training profile. 140
6.18 Brain MRI sample. 140
6.19 ground truth segmentation. 140
6.20 U-Net segmentation. 141
6.21 None Segmentation. 141
6.22 L1-only segmentation. 141
6.23 L2-only segmentation. 141
6.24 L3-only segmentation. 142
6.25 L4-only segmentation 142
6.26 No-L1 segmentation. 142
6.27 No-L2 Segmentation. 142
6.28 No-L3 Segmentation. 143
6.29 No-L4 segmentation. 143
6.30 Average Dice Score vs U-Net model skip-connection modification. 144

List of Abbreviations

CNN	Convolutional Neural Network
AUG	Augmentation
ViT	Vision Transformer
NRM	Normalization
DRL	Decay Learning Rate
LB	Large Batch
NLP	Natural Language Processing
Swin-ViT	Swin Vision Transformer
GAN	Generative Adversarial Network
VAE	Variational Auto-Encoder

List of Appendices

Appendix (1)	Evaluation and comparison of used classification dataset

Chapter One

Introduction

Brain tumors are an abnormal growth of tissues in the brain. They can be challenging to diagnose and treat, yet they remain a pertinent condition that affects people around the world. With the advancement of modern technology, researchers have explored ways to improve diagnosis by using deep learning techniques. This thesis seeks to explore recent deep learning models developed for classifying brain tumors as well as evaluate them in terms of their accuracy.

Multiple studies and studies using a dataset of brain cancer images were conducted. Different deep learning models and techniques were tested. The models and techniques are all recent and inspired from successful efforts in literature and have not been used before in the classification of brain tumor. This research aims to assess the potential effectiveness of these techniques in the improvement of brain cancer diagnostics for future applications.

1.1 Motivation

The diagnosis of brain tumors is a critical task that affects patients' health and well-being. However, long waiting times for diagnosis and treatment can cause increased anxiety and delayed treatment of brain tumors and cancer, ultimately leading to poor patient outcomes. The potential of using deep learning techniques in the diagnosis of brain tumors presents an opportunity to address these challenges and improve patient care. By exploring and improving various recent deep learning techniques, this thesis aims to contribute to the advancement of brain tumor diagnosis

systems, potentially leading to quicker and more accurate diagnoses for patients. Ultimately, the work presented here has the potential to make a significant impact on patients' lives by reducing waiting times for diagnosis and improving the accuracy of brain tumor diagnoses.

In addition to reducing waiting times for patients, there is a growing need to improve the accuracy and reliability of brain tumor diagnosis, as human error in diagnosis can have severe consequences for patient outcomes. Advanced deep learning techniques have the potential to significantly improve the accuracy of brain tumor diagnosis and eliminate the risk of human error. By exploring and improving these techniques, this thesis aims to contribute to the development of more accurate and reliable brain tumor diagnosis systems, which can ultimately improve patient outcomes and help save lives.

1.2 Aim of Thesis

The aim of this thesis is to design an artificially intelligent system that is able to read brain scans and identify the type of tumor that is present. Once the system classifies the brain tumor, it will segment it to highlight its boundaries. This will involve the use of machine learning algorithms and specifically deep learning. We aim to achieve as high accuracy as possible and achieve results that are comparable to the state-of-the art. The system will be trained to distinguish between three brain tumors: glioma, meningioma, and pituitary.

1.3 Method

The research work began with a thorough overview and investigation of the literature done on the topic of interest. The first step was to read online as many related things as possible and skim through various and massive number of surveys and reviews on brain tumor classification and

segmentation to shape a vision and objective to the work. In this stage, certain things were ruled out such as the use of classical machine learning algorithm. It also became clear that our approach will rely on the use of deep learning methods. Then, specific papers of interest were studied in detail and had their results being compared to others along with the objective of finding a literature gap and state-of-the-art methods and techniques. The literature revision also included the revision of previous dissertations and thesis on the same or a related topic. Afterwards, the core of the work begins by exploring the code sources of the papers of interest which made their work publicly available.

When the dataset and path of research was determined, we proceeded by conducting an extensive testing and studies. As is common in the field of deep learning, much of our work was heuristic and based on trial and error, but also informed decision on what next step to take. We heavily relied on the experience of others who worked in a similar or related topic, that is what they did or propose as a means to improve results. We particularly conducted three different studies on the classification dataset, and one on the segmentation dataset.

The first study that was conducted was an initial assessment of different models and architecture performance on our dataset. Despite the use of models that were investigated with the same dataset in a previously published work, we still decided to use them again. That is because in deep learning, there are nearly infinite random configurations and setups that one can adjust to achieve different accuracies using the same dataset and model. For example, the shuffling of dataset, the variation of loss, the addition or removal of layers, the extension of dataset. As such, when using a previously used model, we clearly mention our setup and methodology and compare it later in the discussion with what others did and what had

possibly made our results better or worse than them. In this study, the main objective was to compare between pre-trained and non-pre-trained and custom models to decide which one to focus more on in the next stages.

In the second study, we processed by limiting our selection of the models which we found of potential in the previous study. We decided to use an approach called model soups algorithm in an attempt to test if it improves performance.

In the third study, we mainly hypothesized that transformers, pre-trained models and CNNs each have special abilities, if combined can be of great potential. This motivated our use of pre-trained CoAtNet model. We made a number of assessments of this model and additionally attempted ways to improve its performance by refereeing to literature.

Finally, a full section of thorough discussion is included. The discussion highlights the advantages and shortcomings of the studies and gives a clear comparison to existing literature work, and a clear insight to how this work can contribute to the body of research.

The fourth study which is the segmentation part involved the investigation of altering U-Net skip-connections performance on glioblastoma segmentation. The study is a spin off from the rest of the studies. The purpose of including this chapter in this document is to preserve the effort that was put into this study, as it was a work for nearly two months, as well as to recognize its contribution to research, as it is planned to be published. Segmentation is also complementary to classification, as our automated system can be architecture in a way to first classify a tumor type, and then segment its regions to offer a better perspective to practitioners.

1.4 Research Question

During the development of the research methodology, several research questions were formulated to guide the direction of this work. Answering those questions resulted in trials and studies. These questions are as follows:

1. Which type of model/method yields the highest performance in brain tumor classification: pre-trained, non-pre-trained, or custom models?
2. To what extent can the performance be improved by combining transformers, convolutional neural networks, and pre-training in one model for classifying brain tumors?
3. How does the performance of a pre-trained and non-pre-trained hybrid CNN and transformer model compare with each other, and how do they compare with other pre-trained and non-pre-trained models?
4. Does the model soup algorithm improve the performance of a model?
5. How does augmentation, normalization, batch size, and learning decay affect the final performance?
6. For the segmentation task, can future researchers improve the performance of glioblastoma segmentation without increasing training or inference time by simply altering the skip-connections of a U-Net?

Through the studies conducted in this thesis work, we will directly or indirectly answer these research questions. It is essential to note that each of these questions' answers is a valuable contribution to the research in brain tumor classification and segmentation.

The work of this thesis was originally intended for segmentation of brain tumors rather than classification. However, computational limitations and

time restrain pulled the focus towards classification of brain tumor which is less computationally expensive compared to segmentation.

1.5 Classification Dataset

The dataset used for this study is a combustion of three datasets of brain tumor classification. We originally intended to use a single dataset from these combined datasets. However, we anticipate that having a larger dataset would improve the results of the model, at the same time while maintaining generalizability to the intended 'core-dataset' and the others. Refer to appendix A in which we compare the use of each dataset.

Cheng Dataset

The core dataset is the basic dataset of our work. This dataset is originally created by researchers Cheg et al. [1] acquired from the Nanfang Hospital and General Hopital, Tianjing Medical University, China from 2005 to 2010. It is a brain tumor classification dataset composed of T1-weighted MRI images of three different classifications of tumors Gliomas, Meningiomas, and Pituitary. The images are arranged in a sagittal, coronal, and axial view, as shown in **Figure *1.1***.

They were originally created in MATLAB format. In an study we use a Kaggle version that is preprocessed in '.png' format. The total number of patients included in this dataset is 3064 and organized as show in the below table.

Table 1.1): Classification core dataset

Classification	Number of MRI images
Glioma	1426
Meningioma	708
Pituitary	930

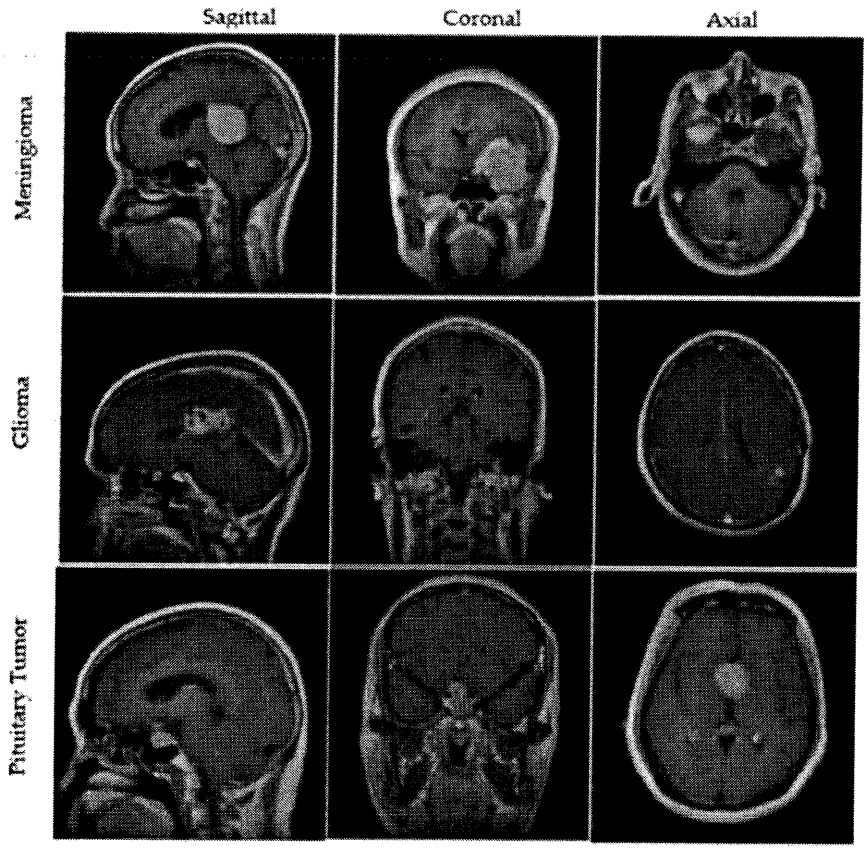


Figure 1.1): Sample from classification dataset.

(Image source: [2])

Extended Cheng Dataset (ours)

This dataset is obtained from Kaggle website [3] under the title 'Brain Tumor Classification (MRI). The dataset size is 158.6 MB and it under the public domain. We use this dataset throughout all of our studies. It consists of four different classifications, each ranging between 74 to 115 images, as shown in the table below.

Table 1.2): Classification extended dataset.

Classification	Number of MRI images (training + testing)
Glioma	1321 + 300
Meningioma	1339 + 306
Pituitary	1457 + 300
No Tumor	1595 + 405

In total this dataset contains 7022 images of human brain MRI. The dataset is originally combined from three different datasets Jun Cheng's dataset [1], Sartaj's dataset [4], and Br35H challenge [5]. Despite encompassing our core dataset within it, we avoided using it for our core work because there are no research works using it. This could make comparing the results of our model to literature difficult. Also, this dataset is pre-processed in a way to make it compact and less memory expensive compared to its original sources which makes it easy to use for initial studies.

1.6 Segmentation Dataset

The dataset used for segmentation is obtained from BraTS dataset from Decathlon library [6]. BraTS is an annual medical segmentation competition which involves the segmentation of glioblastoma which is a cancerous brain tumor [7] [8] [9]. Decathlon is also a companion for general medical image segmentation. They created their compilation of BraTS dataset in which they used 2016 and 2017 datasets and combined them. Also, their dataset is made facilitative in that it can be easily pre-processed and downloaded via a built-in MONAI application (refer to section 1.7). For this reason, it was used in this thesis.

The dataset consists of four different input MRI modalities for each patient sample. The modalities are T1-w, T2-w, T1gd, and FLAIR. There are 750 samples. The dataset is by default divided into 484 for training and 266 for validation. Since there is no testing set, we will use the validation results are our testing results. We justify this by the fact that validation dataset is not used to tune the hyperparameters of the model during training. Also, much of the literature available on BraTS is publications made on ArXiv as pre-prints with only validation results. The dataset has three different output modalities highlighting the boundaries of the tumor parts: tumor core (TC), whole tumor (WT), and enhancing tumor (ET).

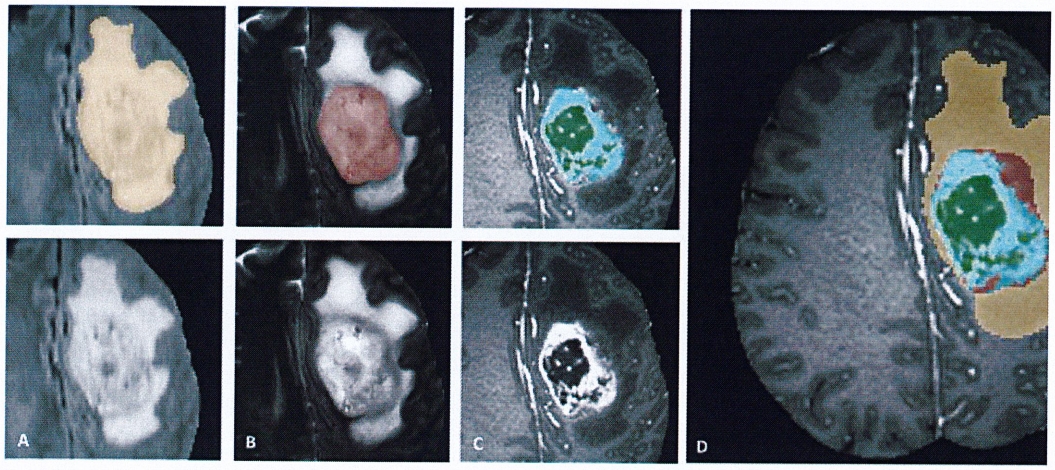


Figure (1.2): A sample of a manually segmented glioblastoma.

(A) FLAIR MRI modality showing the whole tumor boundaries (WT), (B) T2-w MRI modality showing the boundaries of the tumor core (TC), (C) T1c MRI modality showing the boundaries of necrotic component of tumor core (green) and the enhancing tumor (ET) in blue. (Image source: [7])

**Figure (*1.2* shows a sample of a brain MRI with a glioblastoma. The shown modalities are only three of which two, FLAIR and T2-w, in (A) and (B), respectively, are included in our dataset. The segmentation is done

manually to show what is expected from a segmentation model. In the shown sample, four segmentation boundaries are shown. Our dataset only consists three of those boundaries, the yellow, red, and blue segments which represent WT, TC, and ET, respectively.

1.7 Tools

The work is done entirely using Python programming language. As mentioned earlier, the work is divided into two parts: classification, and segmentation. The classification part is done entirely using TensorFlow and Keras which are packages for deep learning in Python created by Google. Both available at https://www.tensorflow.org/ and https://keras.io/, respectively. The segmentation part is done using PyTorch which is also a deep learning package in Python created by Meta AI. In addition, PyTorch was used along with MONAI package which is an extension package specialized for medical imaging deep learning. Both available at https://pytorch.org/ and https://monai.io/, respectively.

All studies were conducted in Google Colab Pro+ which is a virtual cloud-based notebook for deep learning research and scripting. Google Colab offers a Linux based environment with the capability to interpret Python code. Certain scientific and deep learning packages such as TensorFlow, Keras, and PyTorch are automatically installed. In addition, google Colab offers the option to use GPU, TPU, or CPU based computation. The GPU that Google Colab Pro+ offers is NIVIDIA V100, and for a more accelerated GPU is offers A100. However, it sets a computational limit on the usage which posed a limitation to our work.

1.8 Layout

This document is structured in a format to present the steps and results of all studies, for both the classification and segmentation tasks. However,

more emphasis on placed on to the classification as it is the core task of our research. As such, it dedicates separate chapters for its studies and literature review, whereas for segmentation they are all done in a single chapter. The document is organized into nine chapters as follows:

- **Chapter 1: introduction** – introduces the aim and context of the research and motivation behind it. It also clarifies the developed research questions, methodology, and contributions. It shows the tools and datasets that were used through the work.

- **Chapter 2: Theoretical Background and Literature Review** – gives a theoretical background necessary for this work. It includes a background on models serving as the basis of all computer vision deep learning architectures such as convolutional neural network, and the recent vision transformers. The chapter also discusses deep learning training and optimization algorithms, loss functions, and augmentation. Also, this chapter is dedicated to a thorough literature review on the brain tumor classification research specifically done on the dataset that is used in this work.

- **Chapter 3: Transfer Learning and Custom Model Studies** – shows the results of the first conducted studies, where we compare pre-trained and non-pre-trained benchmark models, and custom-built models' performance on brain tumor classification. In the next study, the CoAtNet model is introduced and the implementation of techniques such as batch size increase, augmentation, and use of exponential decay learning for enhancing performance of classification of brain tumors. The chapter compares the results of

different combinations and setups for CoAtNet while gradually increasing the implemented enhancing techniques.

- **Chapter 4: Model Soups Algorithm** – shows the results of implementing soups algorithm on models selected from previous chapters based on their performance.

- **Chapter 5: Discussion and Analysis on Classification of Brain Tumor** – Discusses and analysis the results of the previous three chapters' studies. The chapter also compares our results to other literature results. The chapter also offers future work directions.

- **Chapter 6: Study on the Effect of U-NET Skip-connection Alteration on the Segmentation of Glioma** – this chapter begins with an overview of U-Net model. Then, it gives a survey of using it on the segmentation of Glioma. It presents the related literature work from which we adopt the approach of altering skip-connection configuration. The results are then presented and compared to literature.

- **Chapter 7: Conclusion** – this chapter offers a summary and conclusion of the work done.

Chapter Two

Theoretical Background and Literature Review

Deep learning is a subtype of machine learning where artificial neural networks are used to achieve learning tasks. Artificial neural networks are inspired by the way a biological neural system works. A simple neural network is composed of a series of neurons which consist of an activation function and linked to other neurons via paths which have weights. Neural networks are stacked together in a series of layers. Typical the first and last layers are called input and output layers, respectively, and the layer in the middle are called hidden layers. As computational resources became more abundant and GPUs were developed, the simple neural network has developed into various derivative architectures such as RNN, LSTM, Transforms, CNNs, and more. Different architectures are specialized for different and specific tasks. RNNs and LSTMs were proposed for natural language processing; and CNNs are proposed for vision tasks. Our focus in this chapter is to introduce classification vision related deep learning models.

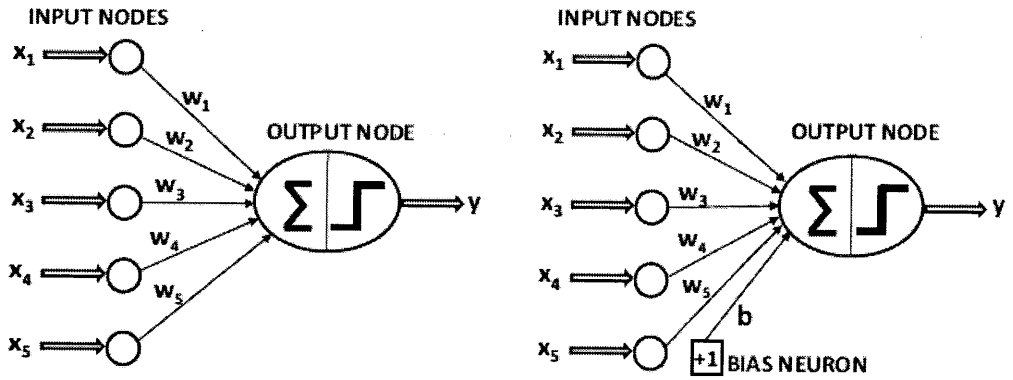
2.1 Computer vision

There are various computer vision tasks. The most famous one is classification, where a model is given a picture which it has to identify the content of that image and classify it accordingly. For example, if a model is capable of distinguishing animals, when given an image of a cat, it should be able to identify it as such. Another task which is also a computer vision task is detection and segmentation. Such tasks are also called dense

classification tasks because a model is required to classify each individual pixel of an image. If a detection model is given an image of a living room, for instance, it is supposed to locate the table, chair, curtains, and other furniture placed in that living room. A segmentation model does the same thing as detection, additionally, it highlights the exact boundaries of an object. In this thesis, we are concerned with classification tasks as we are concerned with classifying a tumor into its correct type. We are also concerned with segmentation in the case of the last conducted study where we segmented the glioblastoma tumor.

2.2 Multi-layer perceptron

The most basic form of a neural network architecture is the multi-layer perceptron, also known as dens or fully connected layer. A multi-layer perceptron is made up of a stack of single layer perceptron. A perceptron is arranged in a way that it contains a single input layer and an output layer.



(a) Perceptron without bias (b) Perceptron with bias

Figure (2.1): The basic architecture of the perceptron.

(a) a perceptron without bias. (b) a perceptron of bias. (Image source: [10])

Given an input vector of $\boldsymbol{X} = [x_1, \ldots, x_d]$, where d is the number of features, with weights $\boldsymbol{W} = [w_1, \ldots., w_d]$, and an output y which ranges between $y \in \{-1, +1\}$, the prediction is computed as

$$y = sign\{\boldsymbol{W}, \boldsymbol{X}\} \qquad (2.1)$$
$$= sing\left\{\sum_{j=1}^{d} w_j x_j\right\}$$

The $sign$ function maps a real value to either $+1$ or -1, which is typically appropriate for binary prediction. The $sign$ function serves as an activation function. Another version of the perceptron is that having a bias which is expressed as

$$y = sign\{\boldsymbol{W}, \boldsymbol{X}\} \qquad 2.2)$$
$$= sing\left\{\sum_{j=1}^{d} w_j x_j + b\right\}$$

The above function is especially important in the case of imbalance distribution of classes [10]. Once we stack multiple single layer perceptron behind each other, we form the multi-layer perceptron. The first layer of a multi-perceptron is the input layer, followed by a stack of layers called the hidden layer, the output layer is proceeded by a soft-max activation to convert the prediction probability into a class vector. Figure (***2.1*** shows a simple multi-layer perceptron. The multi-layer perceptron is also known as a feedforward neural network.

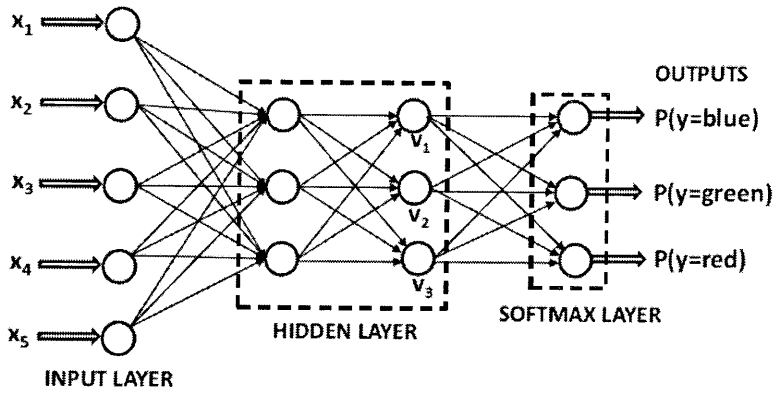


Figure (2.2): multi-layer perceptron.

(Image source: [10])

Activation functions

There are various activation functions that can be used in place of $sign$. In fact, choice of a proper activation function is a critical part of the design. $sign$ is appropriate where the predicted value is a binary value. In other situations, there are multiple classes that need to be predicted. Some of the common classical activation functions are sigmoid, and hyperbolic functions. Other common piece-wise functions are such as ReLU which have largely replaced sigmoid in modern neural networks because of the ease of training multi-layered networks using it. Figure (***2.3*** shows a graphical representation of the behavior of different activation functions.

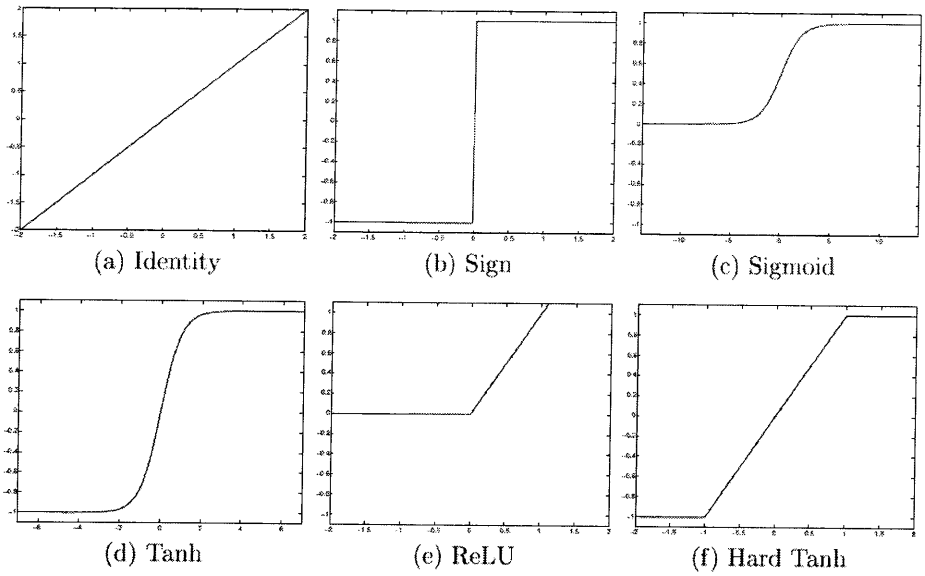


Figure (2.3): Activation functions

(Image source: [10])

2.3 Convolutional Neural Networks

Convolutional neural networks CNNs are the most popular computer vision choice of neural network models. They can also be thought of as neural networks with regularization. They are lighter in memory requirements and more efficient in terms of training and accuracy than dense neural networks (multi-layer perceptron). Ian Goodfellow in his book Deep Learning [11] defined CNNs as '*Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers*'. A convolutional operation for a 2-dimentional image is expressed mathematically as

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i-m, j-n)\, K(m,n) \qquad (2.3)$$

Where I is the input which we assume is a 2-D image, K is the kernel function which is also known as receptive field, and the output S is referred to as a feature map. An example of a convolutional operation is visualized in Figure (***2.4***. A convolutional neural network is a sparse network compared to a fully connected (dense) neural network which is what makes it more computationally efficient. The fact that it is sparse also means it is regularized which makes it more generalizable and less prone to over-fitting compared to a normal neural network.

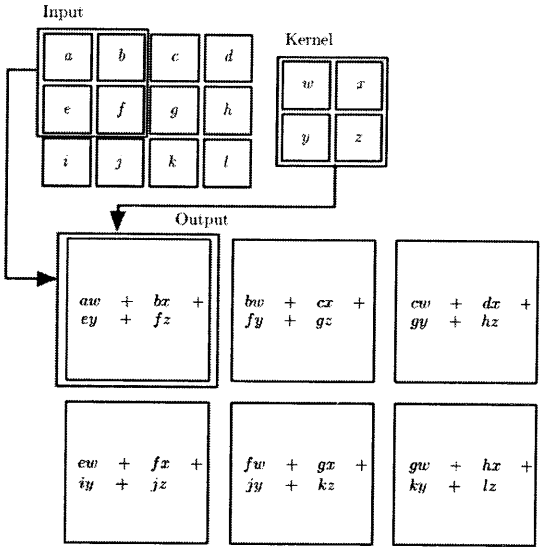


Figure (2.4): An example of a convolutional operation.

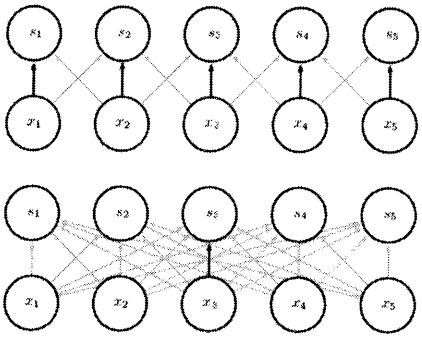


Figure (2.5): (Top) a convolutional neural network, (Bottom) a dense neural network.

(Image source: [11])

The way convolutional neural networks are sparse structures gains them the property of being *equivariant.* Mathematically speaking, if a function $f(x)$ is equuavariant to function g, it means that $f(g(x)) = g(f(x))$. In the context of CNNs and images, a neural network has the ability to identify pixels that belong to a class even if a transformation such as a shift in position is applied to that class of pixels as shown in below. Such a property is required in computer vision tasks such as detection.

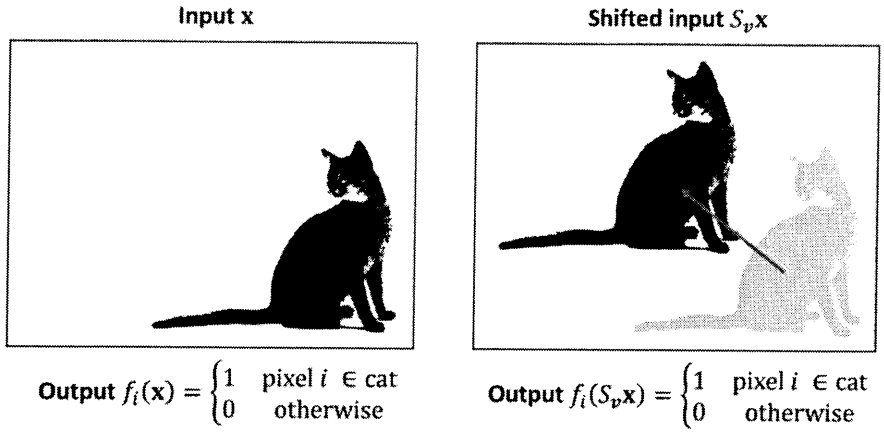


Figure 2.6): An example of shift invariance in a segmentation convolutional neural network.

(Image sources: [12])

2.3.1 Padding

In convolutional neural networks, each layer reduces the size of the next layer. This type of size reduction tends to leave behind some lost information. To resolve this problem, padding is used where we add $\frac{p-1}{2}$ pixels around the boarders of the feature map. The value of these paddings are set to 0. The padding boarder allows the convolution dot product to produce an image along the zero border without affecting the operation.

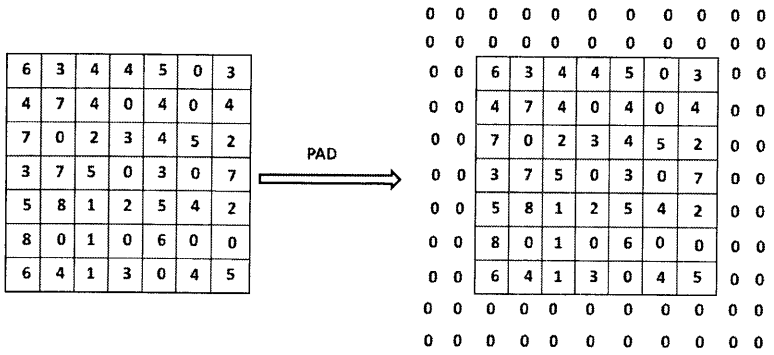


Figure (2.7): Example of padding.

(Image source: [10])

2.3.2 Strides

Strides defines the steps that a kernel (filter) jumps on an input. For instance, if the stride is set to 1, it means that the filter will move one pixel at a time. The size of the stride also affects the output size of the convolutional layer. The higher the stride size, the smaller the output feature map will be. Strides must be set to work hand-in-hand with padding to adjust the final output size.

2.3.3 Feature map

The output of a convolutional layer is called a feature map because it consists of extracted features or elements within an input image. The feature map size is smaller than the input size and it is calculated as $\frac{K(W-F+2P)}{S+1}$, where W is the input volume size, F is the receptive field (i.e., kernel or filter), S is the strides, P is the pooling, and K is the depth of the convolutional layer.

2.3.4 Pooling layers

convolutional neural networks have convolutional layers followed by detectors (nonlinear activation function), followed by pooling layers. A pooling layer is used to modify the output of the filter layer. One of the most used pooling techniques is called max *pooling* which is an operation that calculates the maximum of a value for patches of a feature map. The output is used to create a down sampled feature map. Other popular pooling function is the average-pooling which from its name takes the average of a patch of a feature map.

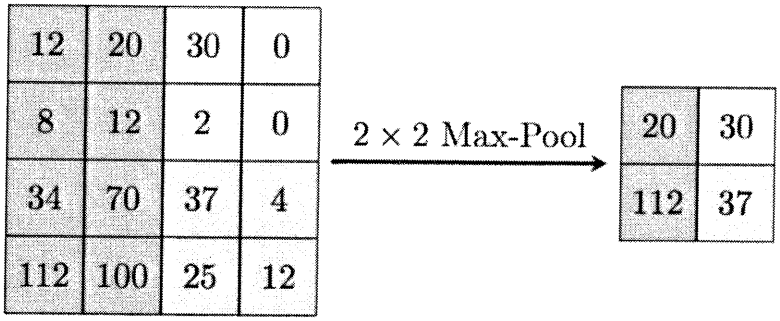


Figure 2.8): example of max pooling operation[1].

The advantage of pooling is to gain a CNN the property of being approximately *invariant* to small transformations. However, this does not mean CNNs is a robust invariant model. There is various literature which investigate the issues in CNN's invariance [13] [14] [15].

2.3.5 Benchmarking and pre-trained models

CNN models which are trained on large datasets serve as benchmark models and are used later on for various tasks beyond the datasets they are trained on. One of the most famous datasets that serve as a benchmark for computer vision field models is ImageNet [16]. This dataset consists of more than 14 million images with more than 20,000 categorists. When it was first published in 2012, many models were created over the years to achieve competitive accuracy levels. Most of which were CNN based models. Figure (***2.9*** shows accuracy progression of models across the years. The more recent models are actually mostly Vision Transformers [17] (discussed in details next section).

[1] Images source: https://computersciencewiki.org/index.php/File:MaxpoolSample2.png

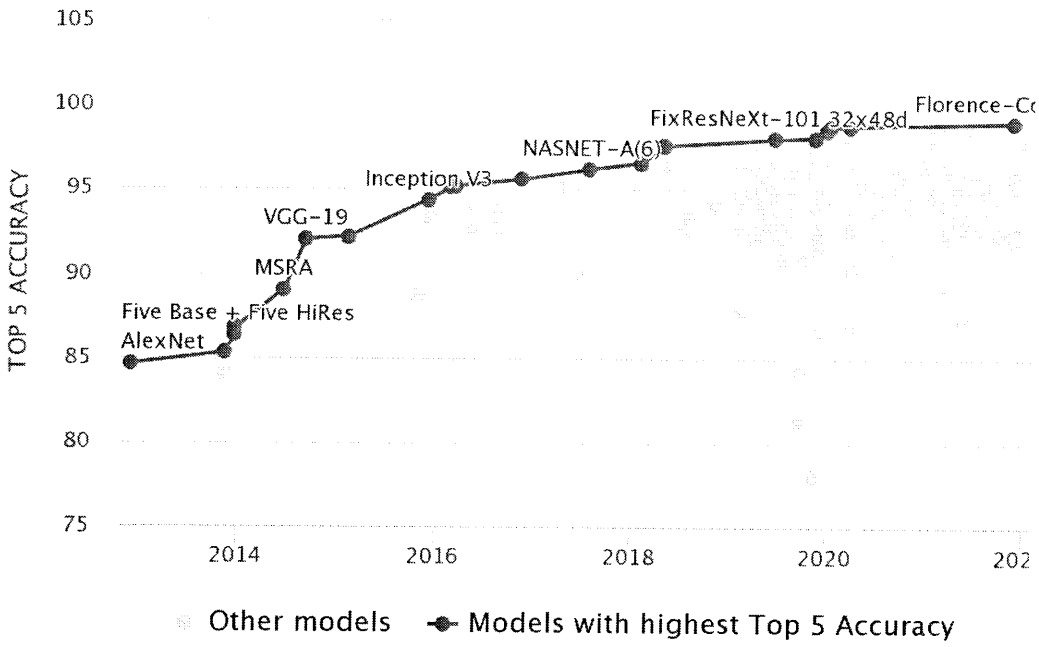


Figure (2.9): progression of models' accuracy on ImageNet dataset[2]

2.3.5.1 AlexNet

AlexNet was the first model to achieve the highest accuracy score in ImageNet [16] dataset. It was proposed by researchers Alex Krizhevisky et al. [18] from university of Toronto in 2012. It is a large convolutional neural network which has around 60 million parameters and 650,000 neurons, consisting of five convolutional layers with a final 1000-way SoftMax. The model was a revolutionary milestone in deep learning. It produced results the surpassed classical machine learning and feature extraction.

The model has many advantages when compared to traditional methods such as SVM. It can extract more semantic information from an image which makes classification even more accurate. Furthermore, it can process data quickly and efficiently. Additionally, AlexNet is easily trained to recognize new objects and categories.

The standard way to model the neuron's activation functions used to be the use if $\tanh(x)$ or $(1 + e^{-x})$. The researchers of AlexNet argue that the

[2] Image source: https://paperswithcode.com/sota/image-classification-on-imagenet

training time of these saturating functions with gradient descent is much slower than non-saturating functions. They proposed the use of activation function $\max(0, x)$ by Nair and HItin [19] which is rectified linear units. Alex Krizzhevisky et al. [18] referred to this function as ReLU which now became the mainstream name for it and a dominating choice for neural network activation functions. They demonstrated that this function is capable of minimizing training error to 25% by 5 epochs, whereas $\tan(x)$ takes around 35 epochs to achieve the same. They also argue that ReLU have the property of not requiring input normalization to prevent saturation. Nevertheless, it is still true that local normalization aids model generalization.

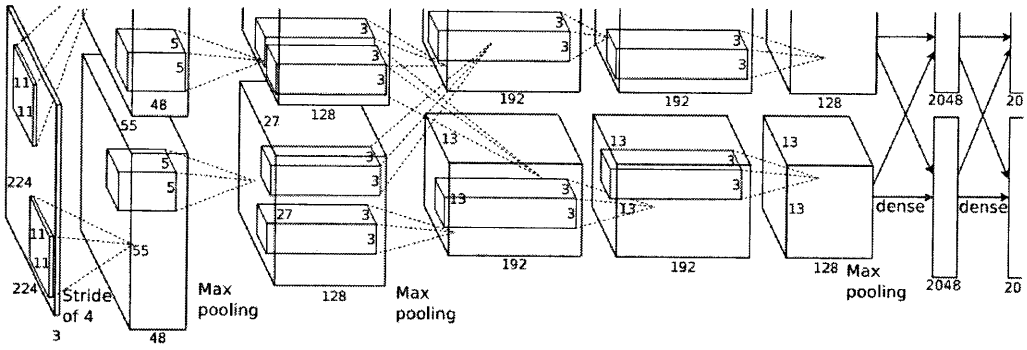


Figure (2.10): AlexNet model architecture

(Image source: [18])

Although their model is complex and consists of 60 million parameters, they still required to reduce overfitting by two primary approaches which are data augmentation and dropout method [20]. The dropout method consists of setting the output of each hidden neuron with probability 0.5 to zero. The neurons which are dropped out do not contribute to the forwards pass and do not participate in backpropagation. As for augmentations, they used two in particular; the first is horizontal reflection, and the second is alteration of RGB channels intensities. They mentioned that without the

first scheme, the network suffers from substantial overfitting which would force them to use smaller network size.

AlexNet was trained using stochastic gradient descent with a batch size of 128, momentum 0.9, and weight decay of 0.0005. The weight decay was used as a form of regularization.

2.3.5.2 VGG

VGG [21] was proposed by University of Oxford researchers in 2015 and scored an accuracy of 92% on ImageNet [16] which was the highest at its time. The researchers proposed this network after investigating the effect of increasing the depth of a CNN on the accuracy. They showed that a signification improvement is achieved when the depth is increased. Their model achieved state-of-the-art on ImageNet and other models and proved to be generalizable. They created several variations of their model as shown in the table shown in Figure (***2.11***.

The distinctive feature that allowed VGG to have many layers is the fact it have got a small receptive field of 3×3. In the same way as AlexNet, they also used ReLU as their activation function. However, they do not use local response normalization as AlexNet, as they argued that it increases memory consumption and computation time without improving the performance.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure (2.11): VGG family of architectures

(Image source: [21])

2.3.5.3 ResNet

Increasing CNN depth has a direct impact on accuracy performance. It increases the performance until up to some point, adverse impact start to happen. This impact was attributed to the degrading gradient decent. To mitigate this problem, K. He et al. [22] in 2016 proposed a new model called ResNet which stands for residual network. The idea behind this architecture is to create a residual block by mapping the input of that block (feeding it) to the output. Therefore, a residual block has an output of $F(x) + x$. This output retains some of the lost information from the feature extraction which if it accumulates would cause the phenomenon of gradient explosion.

The model training adopted random cropping and horizontal flipping as augmentations. Batch normalization [23]was also applied after each convolution before activation. The model was evaluated on ImageNet dataset an achieved various accuracy according to depth of architecture that were the state-of-the-art at the time. The highest architecture top-5 accuracy is ResNet152 which is 93.1%.

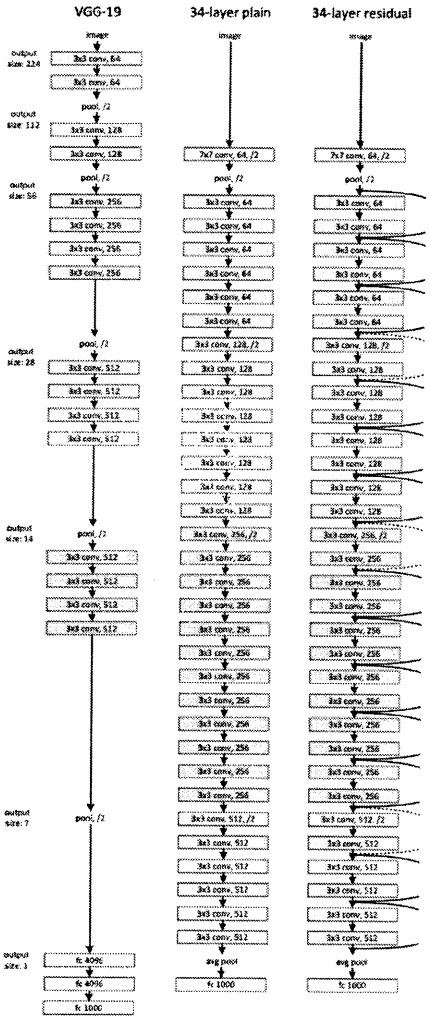


Figure (2.12): ResNet compared with other architectures.

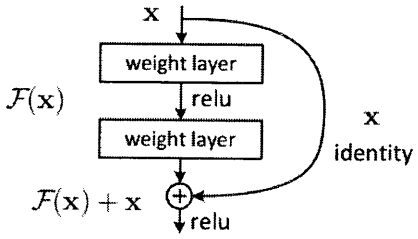


Figure (2.13): A residual block

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-l
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 6; 3×3, 6; 1×1, 2
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 1; 3×3, 1; 1×1, 5
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 25; 3×3, 25; 1×1, 10
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 5; 3×3, 5; 1×1, 20
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3x

Figure (2.14): ResNet family of architectures.

(Images source: [24])

2.4 Vision Transformers

Transformers are a deep learning model that adopts the attention mechanism. The model was first proposed by Google Brain researchers Vaswani et al. [25] in 2017 in a paper titled 'attention is all you need'. Similar to RNNs and LSTMs, transformers were primarily proposed for natural language processing tasks. It has quickly gained popularity and dominated the field due to its high performance. Later, in 2020, another researcher, Dosovitiskiy et al. [17], also from Google, proposed a transformer for computer vision tasks in a paper titled 'An image is worth 16x16 words. Their transformer was called vision transformer, or for short ViT, as it is used for computer vision tasks. There were some previous attempts to replace CNN with attention mechanism for computer vision tasks [26] [27] but these attempts were computationally expensive. On the other hand, using transformer from NLP as they are challenging because images contain more pixels than a sentence contains words. As such, it was not easy to implement transformers to images directly. A method needed to be proposed to lower the computational cost. In this respect, the authors of ViT proposed slicing an image into patches which is analogous to tokens in natural language processing.

Dosovitiskiy et al. [17] results showed that ViT achieves performance higher than the state-of-the-art CNN models on very large datasets. Transformers perform better than CNN but require large datasets and tend to also be computationally more expensive than CNNs. For smaller datasets, the authors suggest the use of strong regularization. Small datasets are something that is often encountered in medical imaging.

In natural language processing tasks, a transformer receives a 1-dimensinal sequence of tokens as an input. However, for 2-dimentional images, the equivalent of a token is a patch. The patch is projected linearly and flattened

to encode it; in addition, a positional encoding is also added to each encoded patch. The encoded patches are processed through the vision transformers block.

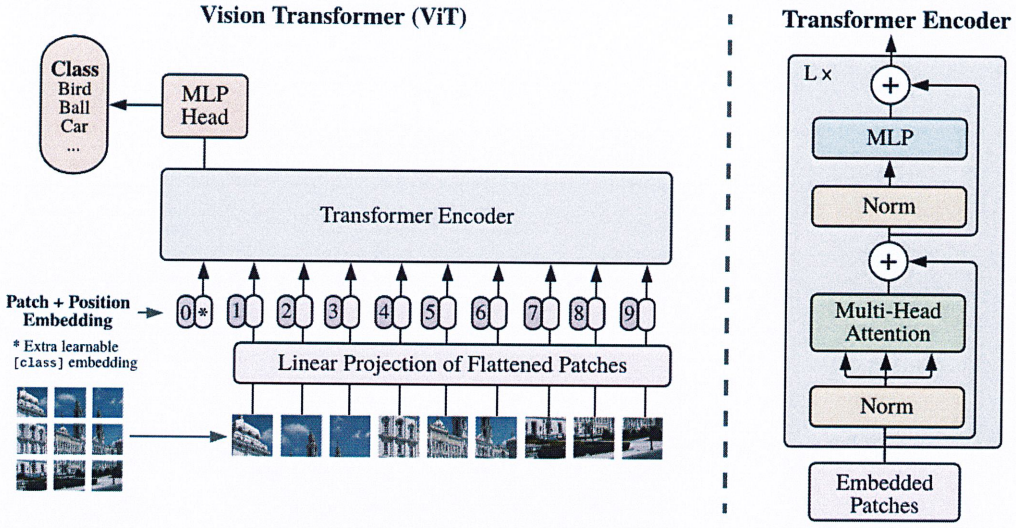


Figure (2.15): Vision transformer architecture diagram

(Image source: [17])

The transformer is made of a multi-head attention layer and a multi-layer perceptron block. Each of these blocks is preceded by a normalization layer which normalizes inputs and speeds up training. A skip connection passes the input into normalization layers into a concatenation node to recover any lost information and smooth the gradient descent. The transformer blocks equations are written as follows:

$$z_0 = \left[x_{class}; x_p^1 E; x_p^2 E; \dots; x_p^N E\right] + E_{pos}, \qquad E \in \mathbb{R}^{(N+1)\times D}$$

$$z'_l = MSA\left(LN(z_{l-1})\right) + z_{l-1}, \qquad l = 1 \dots L \qquad (2.4)$$

$$z_l = MLP\left(LN(z'_l)\right) + z'_l, \qquad l = 1 \dots L \qquad (2.5)$$

$$y = LN(z_L^0)$$

2.4.1 Patch

An image is sliced into patches before processing them in a transformer. This is done to reduce the computational cost. Given an input image $x \in \mathbb{R}^{H \times W \times C}$, such that $H \times W$ is the height and width of an image and C is the number of channels, the patches of such an input are $x_p \in \mathbb{R}^{N \times (P^2 \cdot C)}$. P is the resolution of the patch, and N is the resulting number of patches which is $N = \frac{HW}{P^2}$. Input has to be reshaped by flattening it.

The patch size is fixed, and it is selected by the user. Since the sequence length of the transformer is inversely proportional to the square of the patches size, selecting a smaller patch size makes computation more expensive. In the original paper, the authors proposed the use of 16×16 patches.

For certain tasks with 3D images such as brain tumor MRI, where we have an input $x \in \mathbb{R}^{H \times W \times D \times C}$, the patches are created such that $x_v \in \mathbb{R}^{N \times (P^3 \cdot C)}$, such that $\boldsymbol{N} = (H \times W \times D)/P^3$, where the patch resolution is $(P \times P \times P)$.

2.4.2 Patch Embedding and flattening.

In this sage, the patches are embedded by giving them an encoding. The patching and embedding steps can be performed as a single convolution operation. After patching an embedding, the patches of images need to be combined into a single 2D vector. This process is called flattening. The purpose of flattening is to accelerate computation.

Also, the embedding has to be done such that a learnable additional patch $z_0^0 = x_{class}$ is prepended to serve as the image representation.

2.4.3 Positional Embedding

When an image is transformed into patches, the position information of the patch is lost. As such, a positional encoding must be added to retain the position of each patch with respect to the original image. sin and cos were proposed as positional encoding functions.

$$PE_{pos,2i} = sin\left(\frac{\frac{pos}{10000^{2i}}}{d_{model}}\right) \quad (2.6)$$

$$PE_{pos,2i+1} = cos\left(\frac{\frac{pos}{10000^{2i}}}{d_{model}}\right) \quad (2.7)$$

sin is used for even i, whereas cos is used for odd i.

2.4.4 Multi-head Attention (MSA)

Multi-head attention is the block that performs the self-attention mechanism in the transformer. Attention enables a transformer to learn the relation between a sequence of inputs, enabling it to perform well on tasks with long-range dependencies.

The inputs to the MSA are known as the *query*, *key*, and *value*. These are automatically obtained from a matrix which is created by a dense layer. Each of them is represented as a vector. They are represented as $\boldsymbol{Q} = \boldsymbol{XW^Q}$, $\boldsymbol{K} = \boldsymbol{XW^K}$, and $\boldsymbol{V} = \boldsymbol{XW^V}$. These values are plugged into the MSA block to obtain the self-attention which is:

$$Z = Attention(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = softmax\left(\frac{Q\,K^T}{\sqrt{d_q}}\right) V \quad (2.8)$$

The above formula is also called dot-product attention. d_q is the dimension of the quarry which is equal to the dimensions of the keys d_k. The product of the query and key are scaled by multiplying them by $\frac{1}{\sqrt{d_q}}$ because the dot

product can produce large values in magnitude when d_q is large. Soft max tends to be saturated when it takes large values.

MSA performs the above-mentioned operation to all input patches in parallel. Therefore, the output is concatenated into a single matric $[Z_0, Z_1, \dots Z_{H-1}] \in \mathbb{R}^{n \times h.d_v}$. Self-attention enables transformers to learn both global and local features and provides adaptivity [17].

2.4.5 Multi-layer Perceptron MLP

This block is a basic fully connected multi-layer perceptron with one hidden layer. Given an input vector x, it is multiplied by weights vector W, and a bias term is added. Then a activation layer is used which is in the paper of ViT a GELU function. The full equation of a ViT MLP layer is expressed as

$$MLP(x) = GELU(xW_1 + b_1)W_2 + b_2 \qquad (2.9)$$

2.4.6 Inductive Bias

Machine learning models tend to have certain bias, meaning a unique pattern in performing prediction. For instance, CNNs are well known of having a local inductive bias, meaning that they tend to look at the center of an image to perform their prediction. They also have an equivariance bias, meaning that the position of an object does not change the prediction.

Transformers are different than CNN in that they have a rather global inductive bias. According to the paper of ViT this characteristic makes transformers require a large dataset to be trained. However, that makes them better in generalization, as when the dataset increases in size, they tend to configure their bias to do its predictions. As such they are preferred over CNN in terms of their performance.

2.4.7 Computational Cost

Transformers have a quadratic complexity $O(n^2)$ with respect to an image's size due to the self-attention. Such a complexity hinders its performance on large sized images that would require a longer sequence of patches. There are many works which attempted to solve this issue. One of the most famous ones is Swin transformer which is proposed by Microsoft researchers in 2021 [28] in which the self-attention mechanism is done with a shifting window, thereby reducing the computational complexity to a linear $O(n)$ complexity.

2.4.8 Swin Transformer

The large requirement of computational resources of Vision Transformers made it feasible to transfer to Swin-Transformer Architecture. As mentioned earlier, Swin-transformers [29] were proposed to enhance the speed and performance of the transformer by reducing the quadratic computational complexity $O(n^2)$ into a linear complexity $O(n)$. It proposes the use of hierarchical window shift in which the patches number is reduced in every stage. The computational complexity of a normal and a Swin-transformer on an $h \times w$ image is expressed as

$$\Omega(MSA) = 4hwC^2 + 2(hw)^2C \qquad (2.10)$$

$$\Omega(W\text{-}MSA) = 4hwC^2 + 2M^2hwC \qquad (2.11)$$

The letter M denotes the dimension of the patch which is usually fixed to 7. The first MSA is quadratic and depends on the hw which become expensive for large images. Whereas the window based self-attention has a linear complexity and can easily scale to large images.

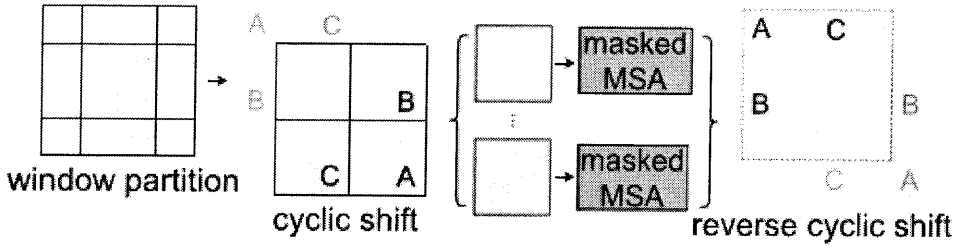


Figure 2.16): Window partitioning illustration.

(Image source: [28])

The above figure visualizes the window partitioning of a Swin Transformer. In each stage of a Swin-transformer, the window is shifted to enhance the modeling power because the MSA lacks connections across windows. The Swin-transformer equations are represented as follows.

$$z_l' = W\text{-}MSA\big(LN(z_{l-1})\big) + z_{l-1} \qquad 2.12)$$

$$z_l = MLP\big(LN(z_l')\big) + z_l', \qquad 2.13)$$

$$z_{l+1}' = SW\text{-}MSA\big(LN(z_l)\big) + z_l \qquad 2.14)$$

$$z_{l+1} = MLP\big(LN(z_{l+1}')\big) + z_{l+1}' \qquad 2.15)$$

2.5 Loss Functions

A loss function or cost function is a function that evaluates how well is a model fitting some dataset. The higher the loss, the more the more the errors a model produces. The objective of training a network is to reduce loss as much as possible.

2.5.1 Cross Entropy

The most widely used loss function for deep learning classification is the cross-entropy function. It computes the loss between the predicted values and target values, or true values. For classes in the range $[0, C)$ where c is the number of classes, the cross-entropy loss is represented as

$$l(x, y) = \boldsymbol{L} = (l_1, \dots, l_N)^T,$$

$$l_n = -w_{y_n} \log\left(\frac{\exp(x_{n,y_n})}{\sum_{c=1}^{C} \exp(x_{n,c})}\right)$$

$$l(x,y) = \begin{cases} \sum_{n=1}^{N} \frac{1}{\sum_{n=1}^{N} w_{y_n} \cdot 1} l_n, & if\ reduction = 'mean'; \\ \sum_{n=1}^{N} l_n, & if\ reduction = 'sum'. \end{cases} \quad (2.16)$$

Where x is the output, y is the target, w is the weight, C is the number of classes, and N spans the minibatch dimension as well as $d_1, \ldots, d_k$ for the k-dimensional case.

2.5.2 Dice Loss

Some datasets suffer from an imbalance which means a certain class is represented more than others. Such datasets bias the model towards the majority class with a tendency to classify the minority as an outlier. Dice loss [30] is a widely used loss function in medical imaging segmentation where data imbalance is a problem. It is mathematically expressed as follows:

$$l(x,y) = 1 - \frac{2}{n} \sum_{n=1}^{N} \frac{\sum_{i=1}^{I} x_{i,j}\, y_{i,j}}{\sum_{i=1}^{I} x_{i,j}^2 + \sum_{i=1}^{I} y_{i,j}^2} l_n \quad (2.17)$$

Where x is the output probability, y is the target probability, w is the weight, I is the number of voxels, and N is the number of classes.

2.6 Optimization

Training a neural network involves the process of tuning its parameters to achieve desired results. Once the loss of a network is found, this loss needs to be minimized in an iterative manner. The minimization is done by tuning the parameters which in return yield better performance. This process of tunning parameters to minimize loss is called optimization. The are many different types of optimization techniques such as gradient descent, stochastic gradient descent, Adam [31], and many others. The objective of

these techniques is to reach the global minimum of a given function $f(x)$ which in the case of deep learning represents the space of the loss function given the intended model.

2.6.1 Gradient descent

This method produces sets the search in the space of a stochastic scalar function $f(x)$ in the negative gradient $\Delta x = -\nabla f(x)$.

Algorithm 1 *Gradient descent method*

given *a starting point* $x \in dom\ f$

repeat

1. *Update weights* $\Delta\theta := -\nabla f(\theta)$
2. *Line search. Choose step size* t *via exact or backtracking line search.*
3. *Update.* $\boldsymbol{\theta} := \boldsymbol{\theta} + t\boldsymbol{\nabla\theta}$

until *stopping criterion is satisfied.*

---

the criteria to stop is$\| \nabla f(x)\|_2 \leq \eta$, where η Is small and positive [32].

2.6.2 Adadelta

Adadelta is an optimization algorithm with an adaptive learning rate. It was proposed by Zeiler and Matthew [33]. The method is able to adapt dynamically over time using first order information and is said to have minimal computational requirement. It requires no manual tuning and is able to tune its learning rate by itself. It is robust to noisy gradient information.

Algorithm 2 *Adadelta*

Require: *Decay rate* ρ, *constant* ϵ

Require: *initial parameter* θ_i

Initialize accumulation variable $E[g^2]_0 = 0, E[\Delta\theta^2]_0 = 0$

For $t = 1{:}T$ ***do*** *loop over # of updates*

1. *Compute gradient:* g_t
2. *Accumulate gradient:* $E[g^2]_t = \rho e[g^2]_{t-1} + (1-\rho)g_t^2$
3. *Compute update* $\Delta\theta_t = -\frac{RMS[\Delta\theta]_{t-1}}{RMS[g]_t}\, g_t$
4. *Accumulate updates:* $E[\Delta\theta^2]_t = \rho E[\Delta\theta^2]_{t-1} + (1-\rho)\Delta\theta_t^2$
5. *Apply update* $\boldsymbol{\theta_{t+1}} = \boldsymbol{\theta_t} + \boldsymbol{\Delta\theta_t}$

End

2.6.3 RMS-prop

RMS-prop is an optimization algorithm that is proposed by Professor Hinton [34]. It stands for root mean square propagation. The aim of RMS-prop is to maintain a moving average of the square of gradients and divide the gradient by the root of this average. Usually, when implementing gradient descent, we may encounter oscillations before reaching a convergence point. This algorithm keeps an exponentially moving average to reduce the oscillations. The prob updates the weights such that $\theta = \theta - \alpha\frac{\Delta\theta}{\sqrt{s\,\Delta\theta}}$ and the bias is updated as $b = b - \alpha\frac{\Delta b}{\sqrt{s\,\Delta b}}$. This prop updates speeds up convergence meanwhile damping oscillations.

Algorithm 3 *RMS-prop*

Require: *Decay rate* ρ, *constant* ϵ

Require: *initial parameter* θ

Initialize β, ρ, T *and* θ

For $t = 1{:}T$ ***do*** *loop over # of updates*

1. *Compute gradient:* g_t
2. *Update parameter* $\boldsymbol{\theta_{t+1}} = \boldsymbol{\theta_t} - \alpha \frac{\Delta\theta_t}{\sqrt{s\,\Delta\theta_t}}$

End

2.6.4 ADAM

ADAM [31] algorithm is another optimization method that has been a common choice among many deep learning researchers. The algorithm is based on adaptive estimation of lower-order moments. It is computationally more efficient and has smaller memory requirements. It is well suited for large datasets and models with large parameters. The method combines the advantages of two other well-known optimization methods which are AdaGrad [35] and RMS-prop.

Algorithm 4 *Adam*

Require: α: *stepsize*

Require: $\beta_1, \beta_2 \in [0,1)$: *Exponential decay rates for the moment estimates*

Require: $f(\theta)$: *Stochastic objective function with parameters* θ

Require: θ_0: *Initial parameter vector*

$m_0 \leftarrow 0$ *(Initialize 1st moment vector)*

$v_0 \leftarrow 0$ *(Initialize 2nd moment vector)*

$t \leftarrow 0$*(Initialize timestep)*

while θ_t *not converged,* ***do:***

$t \leftarrow t + 1$

$g_t \leftarrow \nabla\_\theta f_t(\theta_{t-1})$ *(Get gradients w.r.t. stochastic objective at timestep t)*
$m_t \leftarrow \nabla\_\theta f_t(\theta_{t-1})$ *(Update biased first moment estimate)*
$v_t \leftarrow \nabla\_\theta f_t(\theta_{t-1})$ *(Update biased second raw moment estimate)*
$\widehat{m_t} \leftarrow \nabla\_\theta f_t(\theta_{t-1})$ *(Compute bias-corrected first moment estimate)*
$\widehat{v_t} \leftarrow \nabla\_\theta f_t(\theta_{t-1})$ *(Compute bias-corrected second raw moment estimate)*
$\theta_t \leftarrow \nabla\_\theta f_t(\theta_{t-1})$ *(Update parameters)*

end while

return θ_t

2.7 Metrices

Metrics allow as to measure the performance of a model. They also make the performance comparable in an objective way with other research work. For different deep learning tasks, different types of metrics are used. For different deep learning tasks, there are different performance metrices. In this section, we shall discuss classification and segmentation metricizes.

2.7.1 Classification Metrices

. In classification, evaluating a model performance is essentially done by measuring its Accuracy which tells us how correctly a model is classifying a dataset. True positives TP represent results that are correctly classified, false positives FP represent results that are incorrectly classified such that they are classified into the positive class when the ground truth is negative; false negatives FN represent results that are falsely classified as negative when the ground truth is positive. These classifications typically fit a binary classification problem. However, in multiple-classes problems, they can still be valid for per class basis.

Accuracy

This metric is the most looked at during training and it shows the percentage of total correct classifications. The closer it is to 1 or 100%, the better the results are.

$$ACC = \frac{TP+TN}{TP+TN+FP+FN} \qquad (2.18)$$

Specificity

Represents the proportion of correctly classified negative samples to all of the negative samples in data. The aim is to get this metric as close to zero as possible.

$$SPE = \frac{TN}{TN+FP} \qquad (2.19)$$

Precision

Precision represents the ratio of true positives to all positives.

$$PRE = \frac{TP}{TP+FP} \qquad 2.20)$$

Sensitivity

Sensitivity measures the ratio of true positive samples to the total actual positives.

$$SEN = \frac{TP}{TP+FN} \qquad 2.21)$$

F1-Score or Dice Similarity Coefficient (DSC)

F1-Score metric is suitable for imbalanced data. This metric considers both precision and sensitivity, it is most widely used to assess models trained on imbalanced datasets. Imbalanced datasets are those with a higher number of certain classes compared to other classes. The F1-score is calculated as

$$F1\text{-}Score = 2 \times \frac{PRE \times SEN}{PRE+SEN} \qquad 2.22)$$

In medical imaging segmentation tasks especially, this metric is known as the dice similarity metric [30]. This is not to confuse dice similarity with dice loss, as the former is a loss function which play role in tuning a model hyperparameters, whereas the later is a performance metric that merely measures performance of a model. is mathematically expressed as

$$DSC = \frac{TP}{\frac{1}{2}(2TP+FP+FN)} \qquad (2.23)$$

In segmentation, this metric is used to the spatial overlap between ground truth and predicted annotation.

2.8 Augmentation

To reduce the overfitting of a computer vision model, a common trick to do is to augment data. Data augmentation extends the size of a dataset and allows the model to train on a wider variation of input and be more robust and flexible. Augmentations can be applying transformations to an image such as rotation, reflection, patch extraction, or it can be applied distortions or noise to an image or even changing its brightness and color. The following are some common augmentations in deep learning:

- ***Cropping:*** random cropping of images.
- ***Zoom:*** random zooming of an image
- ***Flips:*** random flipping of an image on axis x, y, z.
- ***Gaussian Nose:*** random Gaussian noise with certain mean and standard deviation.
- ***Gaussian Blur:*** Gaussian blurring with standard deviation of the Gaussian Kernel
- ***Brightness:*** brightness adjustment of images
- ***Contrast:*** contrast adjustment of images

Such augmentations also allow the model to recognize an object even with different orientations. Augmentation is sometimes also known as data perturbation. However, depending on the application, one must practice caution in applying augmentations as, for example, dealing with datasets

such as MINST [36] where we have numbers requires us to avoid certain augmentations such as flipping because if we flip a number like '9' it will become a '6'. Such an augmentation will hinder the performance of our model. Also, in brain tumor classification, cropping is not a good choice as it may crop out a tumor from a brain making it seem as if it is healthy, meanwhile, the label says otherwise which could create a confusion to the model and possible negatively affect training. In the contrary, in segmentation, if cropping is applied to both the label and input in an identical matter, cropping technique does no harm, in fact it would be just as good as any other augmentation technique.

Classification of brain tumor assists clinicians in identifying the type of tumor that is shown in a CT or MRI scan. The ultimate method to diagnose a brain tumor is to perform a biopsy in which a piece of the tumor is resected and taken for pathological examination. However, such a procedure requires caution and may put a patient at risk of hemorrhage and injury that could result in a permanent damage to the brain. In addition, in some cases, contraindications to biopsy may prevent a patient from getting it, meaning due to a certain medical condition, symptom, or any other reason the patient is not able to get a biopsy which necessitates the diagnosis through scans. For this reason, if physicians are able to identify with high accuracy whether a tumor is cancerous or not and what type it is though mere observation, a biopsy can be avoided. In fact, the automation of this process can also reduce waiting time for a patient, as no surgery is needed (for biopsy) and a medical practitioner that is not specialized in radiology can get results with high confidence.

There are various classification techniques that have been proposed for identifying tumor type. The body of literature is massive, ranging from classical machine learning techniques to deep learning techniques. In this

chapter, we will review in detail the work that has been done in the classification of brain cancer using deep learning. The aim is to find a research gap to be filled and contribute to filling it through this work. Also, the information in this chapter will help guide the rest of the work in this thesis and will help in comparison against existing work and methods.

In past works that used the same dataset (core dataset, see chapter Classification Dataset1.5), including the original creators of the dataset [37], the attempts that are made to classify the tumors frequently used classical machine learning techniques. However, their main disadvantage is that they require prior feature engineering and data preparation before training a machine learning model which is time consuming [38]. With the advent of deep learning and CNNs, it became more feasible to approach such a problem using deep learning rather than classical machine learning. It also made the problem as data-driven, rather than feature-driven. As such, the choice of this work is to limit the approach of the task with deep learning methods rather than classical machine learning.

2.9 Related Work

There is various literature research done on the dataset used in this work. Most of which is based on convolutional neural networks. They have been extensively used due to their learning capacity and robustness with computer vision tasks. Other more recent models are based on vision transformers which also achieved results comparable to CNN. There were also attempts to use capsule networks which did not prove to be as robust. Additionally, transfer learning using benchmark models also showed great results. This section will present a literature review of the different techniques that were used.

2.9.1 Use of Convolutional Neural Networks

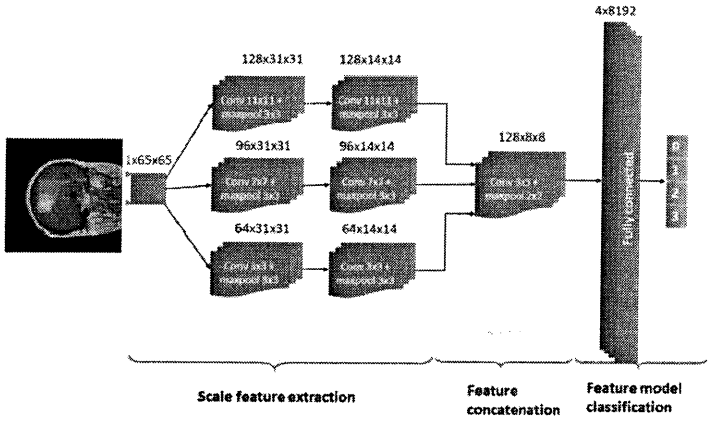


Figure (2.17): CNN with parallel cascades proposed for brain tumor classification.

(Image source: [38])

Francisco Javier Diaz-Pernas et al. [38] proposed a CNN based model which they adopted for two different tasks: classification and segmentation. They proposed a technique of preprocessing input images in three spatial scales along different processing pathways. Their technique is inspired by human visual system operation. Their model has three blocks: scales feature extraction, feature concatenation, and feature model classification. The input layer is a sliding window. In the feature extraction and concatenation blocks, each convolutional layer is followed by a max-pooling layer. The three pathways in the scale feature extraction are design such that they extract large, medium, and small features – such a three-paths enable an enhanced extraction of image features to recognize it on a global and a local level. They also employed a dropout mechanism between concatenation and fully connected stage. Their model achieves very high accuracy on classification which is 97.3%.

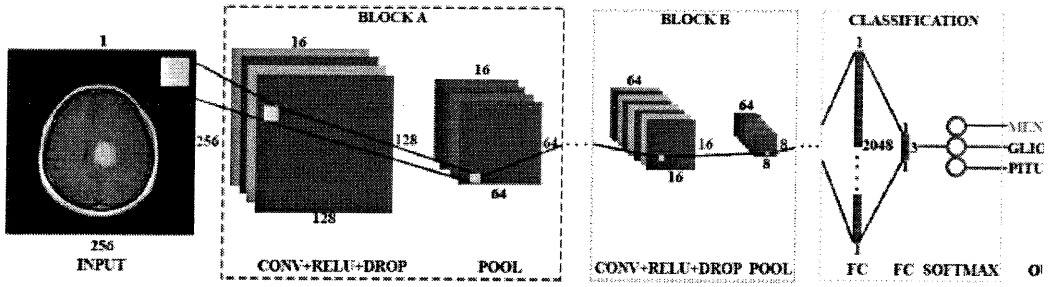


Figure (2.18): a convolutional-based network proposed for brain tumor classification.

(Image source: [39])

Milica and Marko [39] proposed a CNN base model. They applied two image pre-processing augmentations: rotation and flipping. Their model is divided into block A, block B, and classification block. In each block convolutional layers are used with ReLU activations and dropout, followed by pooling. The filters reduce in block A. In the classification layer, a fully connected layer is used. They used 10-fold cross-validation. The weights were initialized using Glorot initializer [40]. They achieved an overall accuracy of 95.4%. The authors showed that a small structure of CNN can have a performance comparable to other more complex models.

Another simple CNN architecture has been proposed by Hossam Sultan et al. [41]. The model is composed of a pre-processing stage where the image is resized into a smaller size image for computational speed. Then, augmentations were applied, specifically, salt noise and image rotation. The model has 16 CNN layers with ReLU activations, normalization layer, a pooling layer, a dropout layer, a fully connected layer, and a softmax at the end. This model has achieved an accuracy of 96.13%.

A CNN graph built with random generation model is proposed by Zhinguan haun et al. [42]. The model is proposed to have a modified activation function GeLU [43] which is new than ReLU and has the tendency to be

more robust as it is able to activate in the negative region, where as ReLU cannot. GeLU is calculated as:

$$GeLU(x) = -.5x(1 + \tanh[\sqrt{2/\pi}\,(x + 0.044715x^3)]) \qquad (2.24)$$

The GeLU nonlinearity weights are inputted by their magnitude rather than signs as in ReLU. The authors achieved with this network the highest accuracy of 94.85% in testing set. In training set they achieved 100% accuracy. The authors skipped visualizing precision and F1-score which are especially important in evaluating such imbalanced dataset and could give a better clue of the model performance. However, it is quite a common practice in literature of this dataset to only evaluate a model by looking at its accuracy.

2.9.2 Use of Capsule Neural Networks

Despite their success, CNNs still suffer with some draw backs. One of the well-known drawbacks is that they are not robust to certain transformations and struggle to take special relationship within the image into consideration. For example, an image maybe completely misinterpreted by a CNN network simply because the color is changed, or some noise is added. That is why augmentation is often used by authors to enhance CNN models. Another drawback is that it requires quite a large dataset to learn from. Medical images datasets tend to be small, as such they are more likely to encounter overfitting CNNs due to them not meeting the size requirements. In an attempt to overcome these CNN short-comings, Hinton et al [44] proposed Capsule networks architecture.

Among the first attempts to use capsule neural networks on brain cancer dataset classification is a model proposed by used by Afshar et al. [45] for brain tumor classification. They explored potential architectures of capsule networks. However, they still found a problem with over-fitting due to small

data size. For this reason, they adopted early stopping technique to smooth out overfitting. They attempted different architectures with different depths as shown in **Figure (*2.19*** below. The best accuracy they achieved was 86.56%. They used 10 epochs and used Caps-Net loss and MSE loss. In a follow-up work, Afshar et al. [46] proposed an improvement on their previous Capsule network by adding a Bayesian framework that can provide not only the mean predictions but also an entropy as a measure of uncertainty. Their argument is that neural network models even though they produce results needs further assessment than simply their accuracy, as in many times it can be misleading. Adding entropy can show us how certain the results are. The authors observed that filtering out results which are over 0.25 uncertainty yielded an accuracy of 71.3%, and by removing over 0.2 uncertainty yielded 72.6%, and by removing uncertainties over 0.14 and 0.1 got an accuracy of 73.6%, and 73.9% respectively. All of which are lower than their original paper results in [45]. Their finding confirms that many of the models presented in literature may be subjected to the same question even if they seem to yield high accuracies. As some degree of overfitting may give misleading results and actually memorize the data rather than generalize to other unseen images.

Capsule Network Architecture	Prediction Accur
Original architecture	82.30%
Two convolutional layers with 64 feature maps each	81.97%
One convolutional layer with 64 feature maps	86.56%
One convolutional layer with 64 feature maps and 16 primary capsules	83.61%
One convolutional layer with 64 feature maps and 32 primary capsules of dimension 4	82.30%
Three fully connected layers with 1024, 2048 and 4096 neurons	83.93%

Figure (2.19): capsule network architecture performance on brain tumor classification.

(Image source: [45])

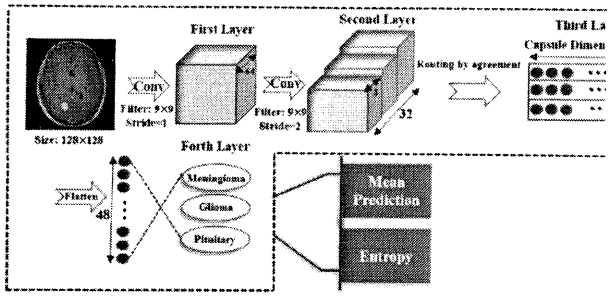


Figure (2.20): Improved Bayesian capsule network brain tumor classification model.

(Image source: [46])

2.9.3 Use of Transfer learning

Reducing the requirement of large datasets can be achieved by transfer learning, especially if the pre-training dataset is very large. Rehman et al. [47] investigated the use of transfer learning on three different famous benchmark models (see chapter 2) which are AlexNet [18], GoogLeNet [48], VGG [21]. nevertheless, they still applied several data augmentations to increase the size of the dataset which include rotations and flipping. They explored many scenarios by varying the epochs from 7 to 10, and trying different loss functions such as ADAM, SGDM, RMS-prop. The best results that they were able to achieve are 97.39%, 98.04%, 98.69% for AlexNet, GoogLeNet, and VGG, respectively. Their results are as expected since the most advanced of all these networks is VGG. However, they did not study with the more advanced and famous pre-trained model ResNet [24], despite being already proposed at the time the paper was written.

Swati et al. [49] proposed the use of content-based brain tumor retrieval which is applied to measure the similarity distance between images. The features of the images were initially extracted from a pre-trained VGG19. They achieved a final accuracy of 96.79%. They did not use any

augmentations which explains the lower accuracy compared to Rehman et al. [47]. Also, they used VGG19, whereas the latter used VGG16 which has a less complex architecture, as such less likely to overfit the data.

Models that use transfer learning generally score much higher than those that do not. In fact, the highest score we were able to find in literature for accuracy is 99.02% in which transfer learning was used before training ResNet50 as proposed by Özlem and Chafer [50]. The model was pre-trained on ImageNet dataset.

2.9.4 Use of Generative Adversarial Networks

Generative Adversarial Networks, or GANs for short, we first proposed by Goodfellow et al. [51] in 2014. A GAN consists of two fundamental elements: a generator and a discriminator. The generator attempts to create outputs that mimic real inputs or images in such a way that it fools the discriminator which tries to distinguish real from fake. If we assume D to be the discriminator, and G a generator, the two are playing a minimax game with value function or loss function represented as follows:

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[log\, D(x)] + E_{z\sim p_z(x)}\left[log\left(1 - D\big(G(z)\big)\right)\right] \quad 2.24)$$

GANs were used numerously in the generation of images and deep fake applications [52]. In medicine, GANs are a useful tool to generate artificial medical images where datasets are limited [53]. Bilal Ahmad et al. [54] proposed the combination of variational autoencoders VAE and generative adversarial networks GANs to generate realistic looking MRIs with brain tumors. Basically, the GAN and VAE would act as an augmentation layer to extend the dataset. They used a ResNet50 as their classifier and achieved an accuracy of 96.25%. However, they do not mention clearly whether it is pre-trained or not. Nevertheless, they mentioned training it for 30 epochs. The authors choose Adam optimizer with a learning rate of 0.0001 in

training the ResNet. They also split their data into 60% training, 20% validation, and 20% testing. The authors also attempted the training of ResNet50 under different settings: (a) without classic augmentations and without generative images they achieved 72.63% accuracy, (b) with classical augmentation and without generative images they achieved 77.52%, (c) without classical augmentation and with generative images they achieved 92.3%, and (b) with generative images and with classical augmentation they achieved 96.25%.

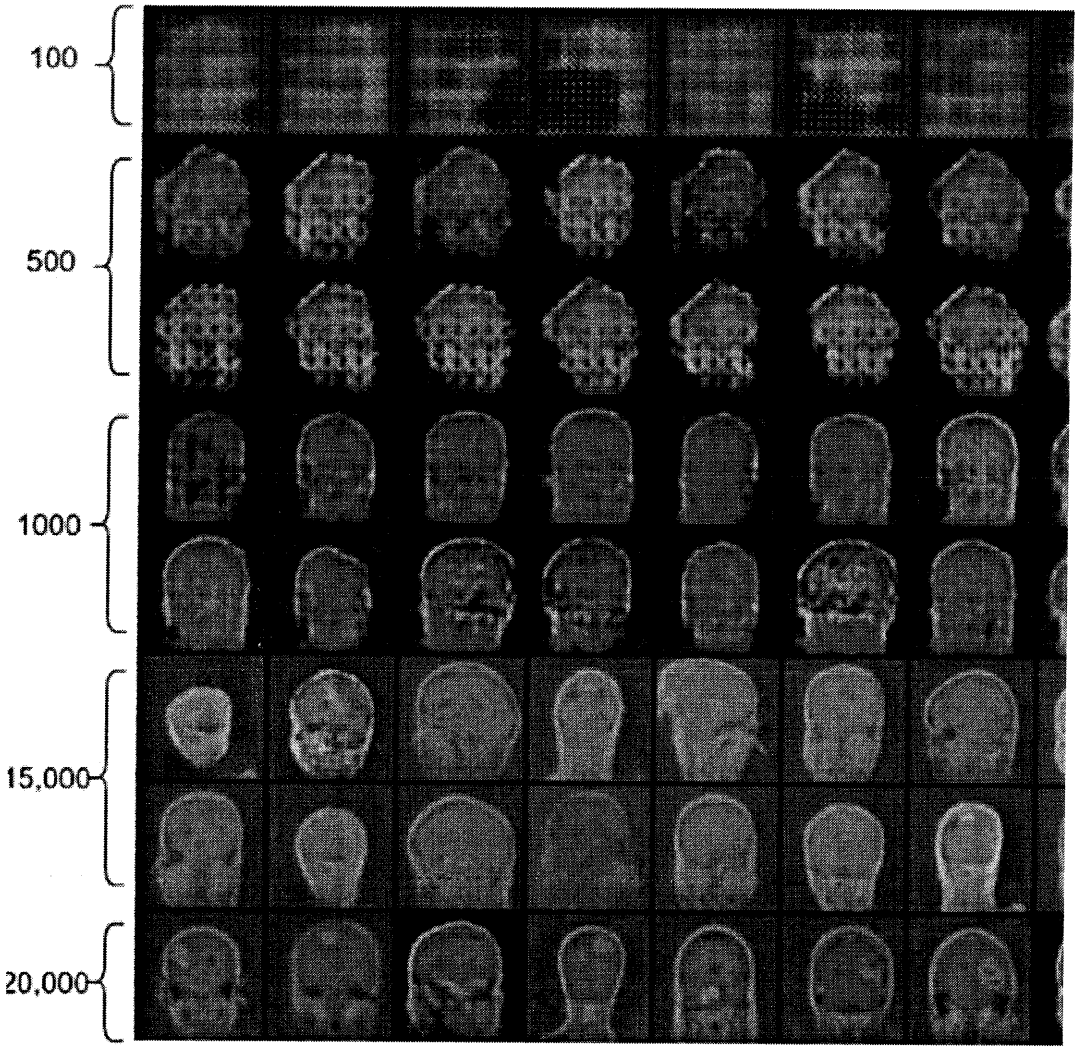


Figure 2.21): images generated by proposed VAE-GAN architecture after 100, 500, 1000, 15,000, and 20,000 training epochs.

(Image source: [53])

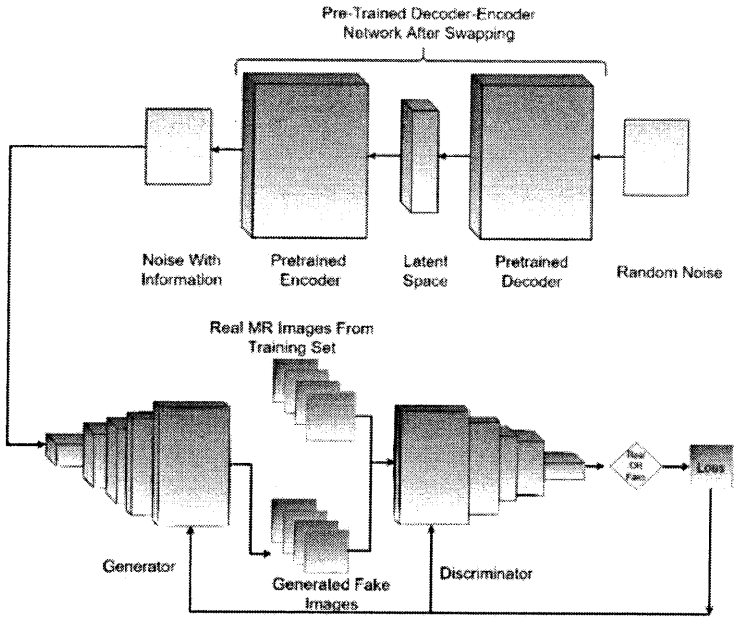


Figure (2.22): Proposed method to use GAN and VEA as augmenters.

(Image source: [54])

2.9.5 Use of Vision Transformers

Sudhakar et al. [2] train an ensemble of vision transformers to classify brain tumors. They fine-tuned B/16, B/32, L/16, and L/32 [17] which were pretrained on ImageNet [16]. The ensemble technique they used was to take the SoftMax output of each ViT model, dot-product them, and take their average. Their models were made to be trained on two different resolutions; some models were trained on 224X224 and the others are trained on 384X384. The SoftMax and Ensembling is mathematically represented as:

$$softmax = \frac{1}{N}\sum_{i=1}^{N} softmax_i \qquad (2.25)$$

$$final\ prediction = argmax(softmax_e) \qquad (2.26)$$

They also trained their different models on various sets of optimizers and hyperparameters. The optimizers they used were RMS-prop, Adam, and

Adadelta. They also varied their epochs in 15, 20, and 25. The learning rate was varied as 0.0001, 0.000005. 0.1, and 0.05. The batch size was set as 16, 32, and 8. The table shows the different combinations and set of accuracies they got.

Resolution	Optimizers and Hyperparameters	Validation Accuracy in Percentage		
		ViT-B/16	ViT-B/32	ViT-L/16
224 × 224	*RMSprop* $lr = \mathbf{0.0001}$, $ne = \mathbf{25}$, $mbs = \mathbf{16}$ $lr = 0.0001$, $ne = 20$, $mbs = 32$ $lr = 0.00005$, $ne = 15$, $mbs = 32$	96.20 96.41 97.06	**97.28** 97.01 96.47	96.10 96.47 95.92
	Adam $lr = \mathbf{0.0001}$, $ne = \mathbf{25}$, $mbs = \mathbf{16}$ $lr = 0.0001$, $ne = 20$, $mbs = 32$ $lr = 0.00005$, $ne = 15$, $mbs = 32$	**97.83** 97.55 96.47	95.92 96.74 96.74	96.82 96.40 96.50
	Adadelta $lr = \mathbf{0.1}$, $ne = \mathbf{15}$, $mbs = \mathbf{16}$ $lr = 0.1$, $ne = 20$, $mbs = 32$ $lr = \mathbf{0.05}$, $ne = \mathbf{15}$, $mbs = \mathbf{32}$	97.25 97.01 97.55	96.01 96.01 96.20	97.28 97.25 **97.55**
384 × 384	*RMSprop* $lr = 0.0001$, $ne = 15$, $mbs = 8$ $lr = 0.0001$, $ne = 10$, $mbs = 16$ $lr = 0.00005$, $ne = 10$, $mbs = 8$	97.31 96.60 97.63	97.55 97.21 96.74	97.40 96.95 97.60
	Adam $lr = 0.0001$, $ne = 15$, $mbs = 8$ $lr = 0.0001$, $ne = 10$, $mbs = 16$ $lr = 0.00005$, $ne = 10$, $mbs = 8$	97.30 97.54 96.90	97.11 96.65 97.01	96.82 97.40 97.70
	Adadelta $lr = \mathbf{0.1}$, $ne = \mathbf{10}$, $mbs = \mathbf{8}$ $lr = 0.1$, $ne = 15$, $mbs = 16$ $lr = \mathbf{0.05}$, $ne = \mathbf{10}$, $mbs = \mathbf{8}$	97.10 97.80 **98.10**	**98.04** 97.83 96.84	97.90 97.50 **98.64**

Figure (2.23): ensemble vision transformer performance on brain tumor classification.

2.10 Challenges

There are various challenges facing the field of brain tumor classification using deep learning. One of the main challenges lies in the fact that data available is scarce, meanwhile deep learning architectures such as CNNs require large number of annotated data to have high performance [55]. When dataset is not large enough, a CNN has a high tendency to overfit, affecting its capacity to generalizer [56]. Another challenge is the data mismatch in which a model that is well trained and shows high performance fails to generalize to real-world clinical data. It could be that the training

dataset is different than clinical data [55]. Another challenge that could be faced with datasets is the class imbalance where a certain class is represented much more than others. For example, positive cases are more than negative. This makes a network biased towards the majority class [57]. In addition, data quality may impact the performance of a classifier. The data that is acquired may contain some noise. In an MRI acquisition process, the source of this noise could be the scanner or external sources that could induce electricity in the receiver coil of the MRI, generating image artifacts. Also, if the patient moves during the MRI scanning, this may affect the acquisition, especially during the image reconstruction [58]. These factors may reduce the quality of images, thereby affecting classifier performance.

Chapter Three

Transfer Learning and Custom Model Studies

A series of benchmarking studies were conducted on the brain tumor classification dataset. These studies involve the use of benchmark state-of-the-art models from computer vision classification literature and training them to process our models. This chapter will show the full details of those studies. The aim is to choose a model that perfectly suites our data and make it our starting point. Further enhancements to the model will be made in the next chapter. The basis upon which we select our model is how good it performs on testing data.

The purpose of this study is to build a foundation for the rest of the work. Our aim is to find a model to begin with, as in the later studies we will aim to improve on this model. We will compare three schemes: non-pretrained benchmark models, pre-trained benchmark models, and custom-made models. Then, we will compare all results to conclude on which model should we select to proceed with in the rest of our work.

3.1 Setup

For all studies the number of epochs is chosen to be 40 epochs, number of batches per step is 35. Unless stated otherwise, on all occasions the epochs and batch number will stay the same. We will study with the following models and train each one at a time:

- VGG16 [21]
- ResNet50 [22]
- ResNet101 [22]

- Swin-ViT [29]
- Custom simple CNN

The first two models will be studied with first by non-pretraining and then by pretraining them. Then, we will compare the difference between them in further details in the discussion chapter.

In each epoch, the data will be normalized by batch normalization which will normalize the pixels of the image and enhance the training speed and convergence of the model. Then a number of augmentations is performed which are as follows:

- Rescaling – pixels of an image will be rescaled.
- Random flipping – images will specifically be flipped horizontally.
- Random brightness – color brightness and intensity will be varied.

Augmentations help the model generalize better and prevents the model from overfitting and memorizing the data [59]. CNNs, especially, due to their learning bias tend to collapse sometimes due to slight variations in an image such as flipping an image, or changing the location of an object, or even the color, or adding noise. In adversarial attacks, one may fool CNN by giving it an image with a noise injected. This was demonstrated by DeepFool model proposed by Moosavi-Dezfooli et al. which finds the minimum possible noise injected needed to cause a misclassification [60]. Augmentations in this case can serve as a valuable tool to improve this weakness. However, we will not go as far as using augmentation as a defense against adversarial attacks in our work, as this is not our focus.

In addition to the augmentations and the base model training, three layers will be added at the end that will perform the prediction at the end of the training pipeline. Those are global averaging, drop out, and an additional dense multi-perceptron layer with an output of 4. Each output will resemble

a hot-encoded classification which correspond to No-tumor, Glioma, Meningiomas, and Pituitary tumor. Also, the image sizes will be rescaled to 128. Smaller sizes tend to be less computationally expensive. We split the images into 90% training and 10% testing. Within the training set we let 10% be for validation which is important in analyzing the results and interpreting them.

For this specific study, we used brain tumor dataset from Kaggle. The reason we use this dataset is that it is also a brain tumor dataset, but it has got a smaller size which makes it easy to conduct bulk studies and perform and test adjustments layer on. Also, since we face a limitation on computational resources, and we need to train and test multiple models, this dataset is more efficient. Although different datasets tend to have different signals according to their shapes, distribution, and number of positives and negatives, this dataset it expected to mimic the dataset which we will be using later on, in terms of distribution and ratio of positive to negative images, i.e., percentage of each classification representation in the dataset.

The loss function that will be used is categorical cross entropy and the optimizer is Adam [31]. The optimizer was kept at default rates as per Keras built-in Adam model with a 0.001 learning rate which are the same as the paper. In the original paper that proposed ADAM algorithm [31], it was shown that it outperforms other algorithms such as SGD and AdaGrad. It was tested on CIFAR10 images training a convolutional neural network, as can be seen in the figure below. For this reason, we made ADAM as our first choice.

TensorFlow and Keras were used in this study and the models are obtained from built-in Keras pre-trained models where available. However, pre-training is set to false. We use NIVIDIA V100 GPUs of 15 GB size obtained from Google Colab Pro+.

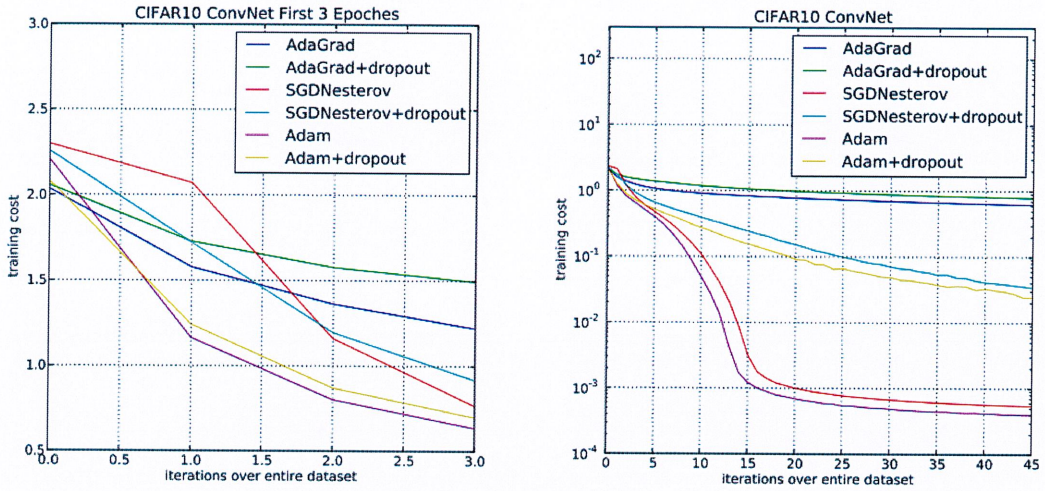


Figure (3.1): comparison of different optimizer performances on CIFAR10 dataset.

(Image source: [31])

The settings mentioned in this subsection are followed in all trainings of this chapter for the sakes of fair comparison, unless otherwise stated. We will call those settings the conventional settings throughout this chapter. Initially models were done following those setups and adjustments were made later according to results, if needed.

Be noted in the following section that when we display the training and validation curves, we will follow the convention of coloring the training curves in red and the validation curves in green. As for the final accuracy, we consider the accuracy obtained from the testing dataset, not the training or validation. This convention will be followed in presenting the results of this chapter and all other chapters.

3.2 Non-pretrained

In this section, we will train some deep learning CNN based benchmark models without pre-training. Typically, these models are not trained from

scratch, as we will do in this sub-section. The practice is to pre-train them on a very large dataset and then fine-tune them on a smaller dataset in a process known as transfer learning. We will keep transfer learning for the next subsection. As for this one, the training is done completely from scratch.

3.2.1 VGG16

Training VGG16 model following the conventional setting shows that the accuracy of training and validation become constant at 28%. This indicates that the model is experiencing a plateau in which the loss function arrives to a local minimum and is not able to get out of that saddle point, as can be seen in the figure. For this reason, different attempts were made to fit the data to the model without reaching a local minimum such as changing the learning rate into higher numbers ranging from 0.01 to 2.5 and the optimizer itself was changed from Adam to SGD. Nevertheless, the same problem continued to occur. It could indicate that the model is too big for the dataset.

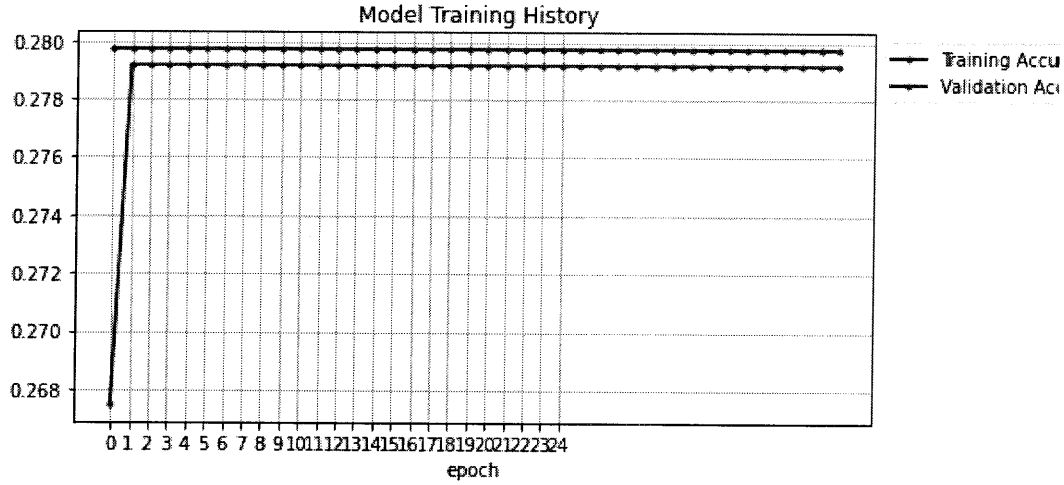


Figure (3.2): VGG without pre-training accuracy.

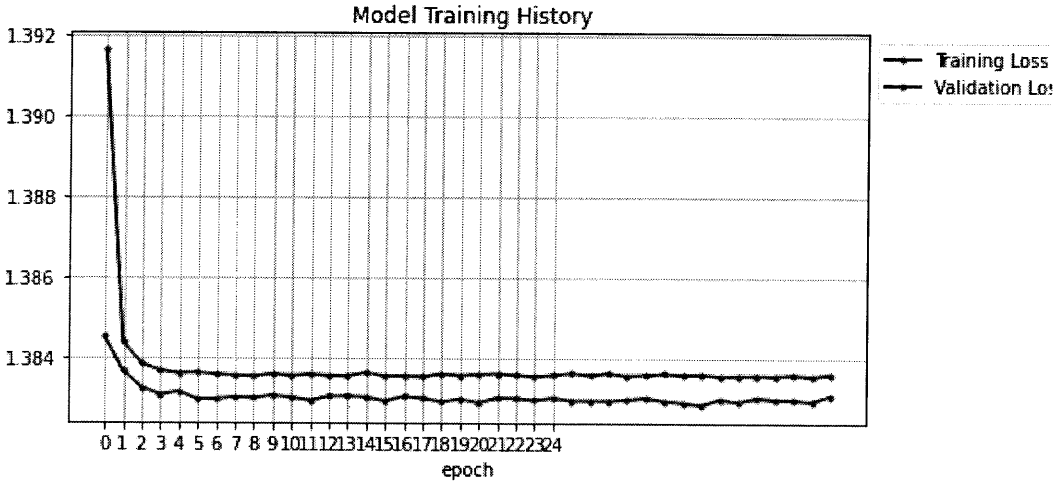


Figure (3.3): VGG without pre-training loss.

3.2.2 ResNet101

The ResNet101 network showed a pretty good training accuracy profile, but the validation, although reached more than 90% in some epochs showed a rather fluctuating behavior. We attribute this fluctuation to the fact that augmentations were also included during the validation phase. Because the way the augmentation layers were integrated with the model itself and were not frozen during the validation phase. We neglected freezing them during validation because our final ideal aim is to reach a model that it invariant to perturbations. Nevertheless, we inferred the final training results from the testing set. The test results gave a final accuracy of 23% which is very low.

We interpret that the reason behind high training accuracy is the memorization of training set. This is quite common when the dataset is small compared to the network. In the next subsection we try using a smaller network which is ResNet50 to confirm this hypothesis.

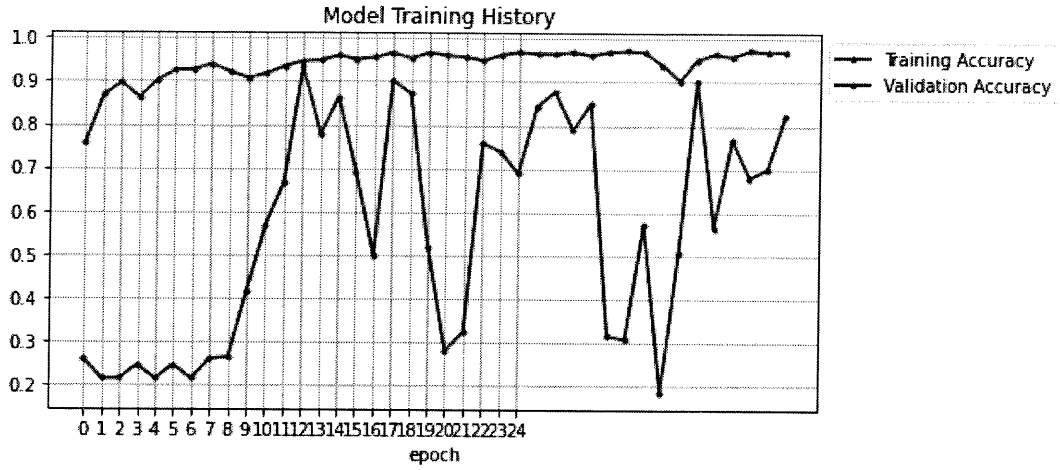


Figure 3.4): ResNet101 without pre-training accuracy

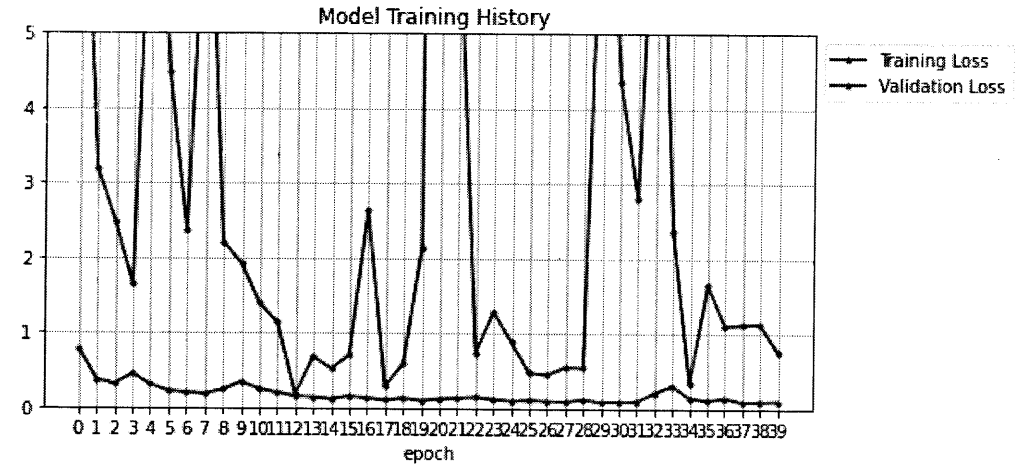


Figure (3.5): ResNet101 without pre-training loss

3.2.3 ResNet50

The training was repeated on ResNet50 to test if reducing the size would change the results. We have found unfortunately that the results are still the same. Training seems to be memorized by the model. The test accuracy result is 23%.

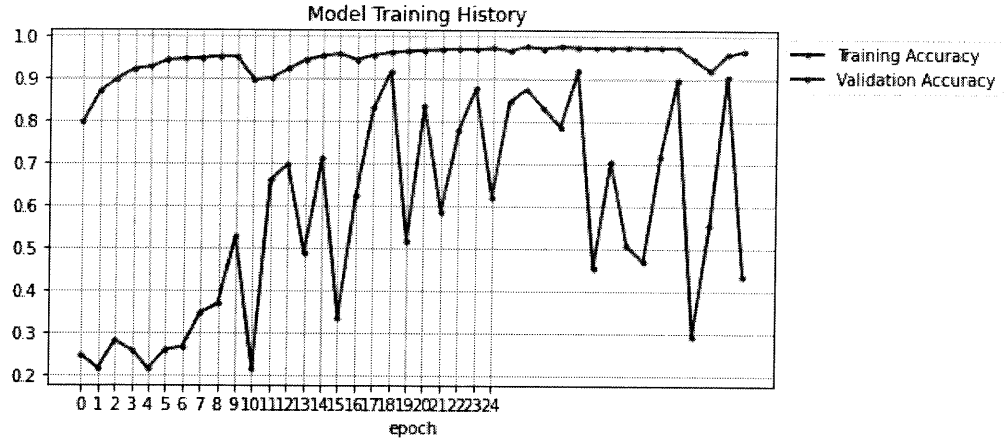


Figure (3.6): ResNet50 without pre-training accuracy

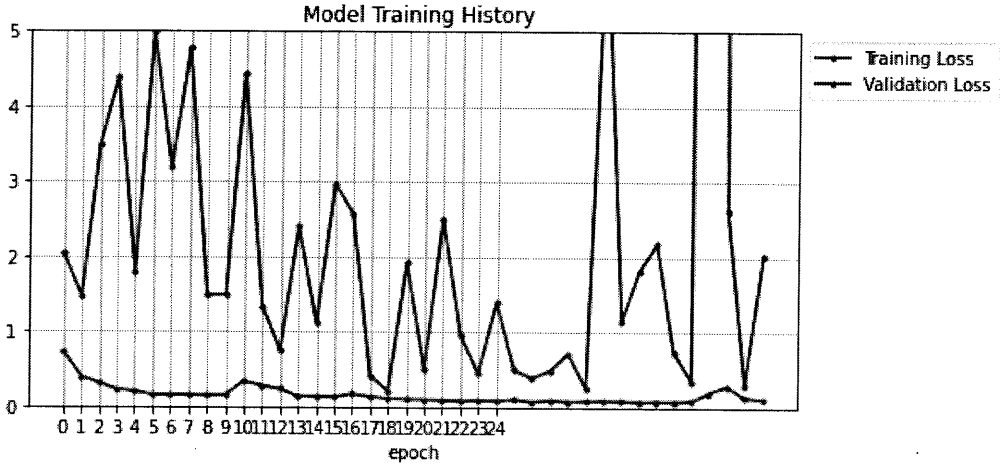
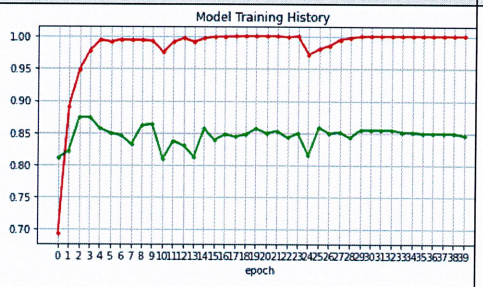
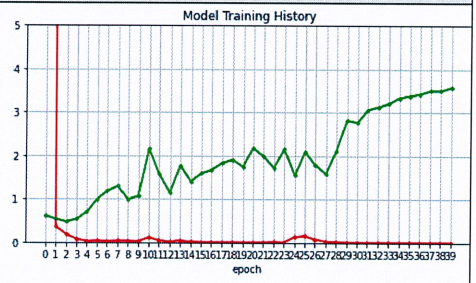
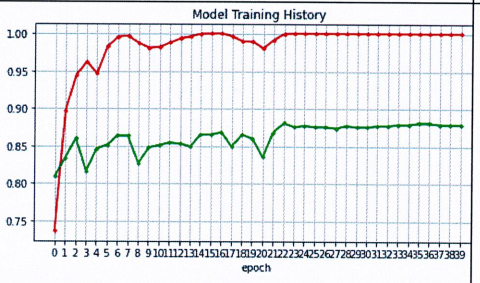
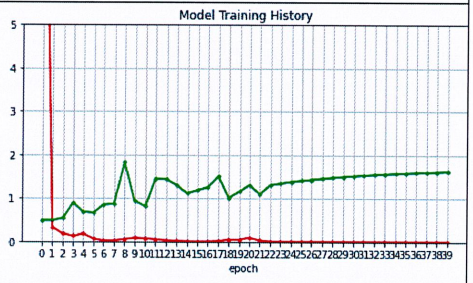
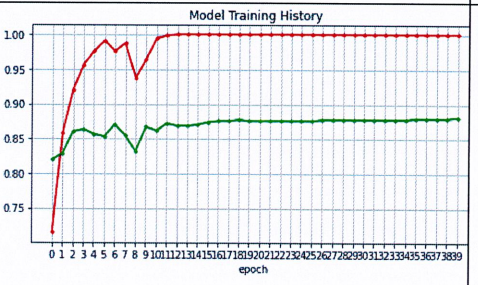
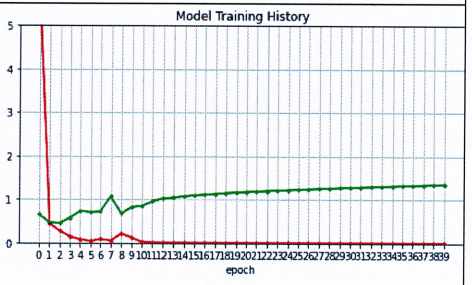


Figure (3.7): ResNet50 without pre-training loss

3.2.4 Simple Custom CNN

We decided to design a rather simple and custom CNN network to train on the data. The reason we use a small network is because the dataset is small, and we want to avoid memorization of training dataset as was done by the previously trained large benchmark models which are by default designed to accommodate large datasets. We will try multiple different custom designs of simple CNNs. We will use the setup configurations. However, the final 2 layers which are the global averaging, drop-out might be verified. Each network design will be presented.

Table 3.1): Custom CNN results

	Training (red) and Validation (green) Plots	
CNN layers	**Accuracy**	**Loss**
2	 **Figure (3.8): Accuracy of 2-layer CNN.**	 **Figure (3.9): Loss of 2-layer CNN.**
3	 **Figure 3.10): Accuracy of 3-layer CNN.**	 **Figure (3.11): Loss of 3-layer CNN.**
4	 **Figure 3.12): Accuracy of 4-layer CNN.**	 **Figure (3.13): Loss of 4-layer CNN.**

5

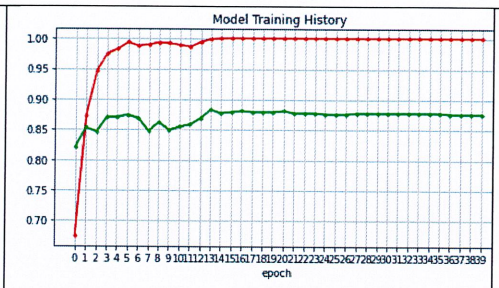


Figure 3.14): Accuracy of 5-layer CNN.

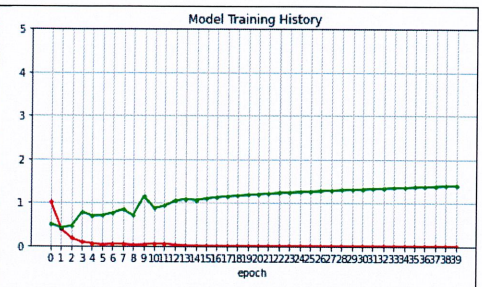


Figure (3.15): Loss of 5-layer CNN.

6

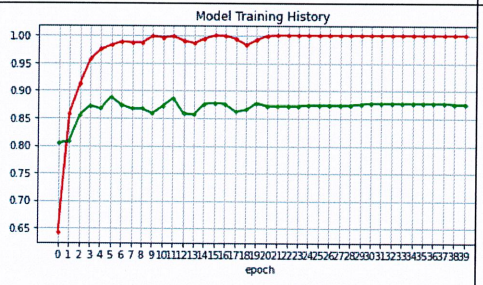


Figure (3.16): Accuracy of 6-layer CNN.

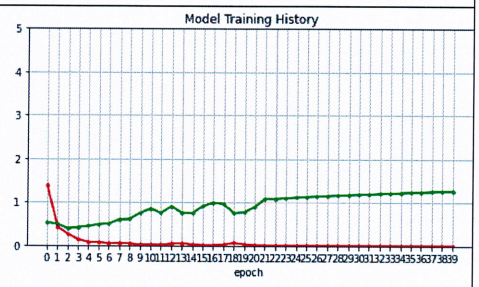


Figure 3.17): Loss of 6-layer CNN.

The models were tested by using the testing dataset and the results of the accuracy, precision, recall, and F1-score are shown below. It can be seen that the model which gives optimal results is the 4 layers model.

Table 3.2): Summary of custom CNN results

CNN Layers	Accuracy	Precision	Recall	F1-Score
2	0.85	0.87	0.88	0.87
3	0.89	0.91	0.95	0.98
4	0.91	0.93	0.95	0.94
5	0.90	0.91	0.95	0.93
6	0.89	0.91	0.96	0.94

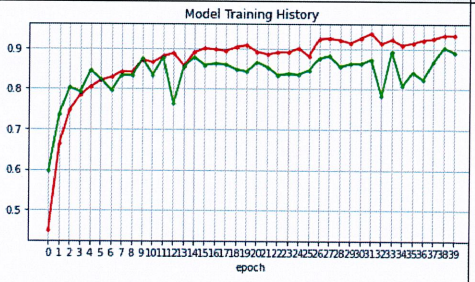
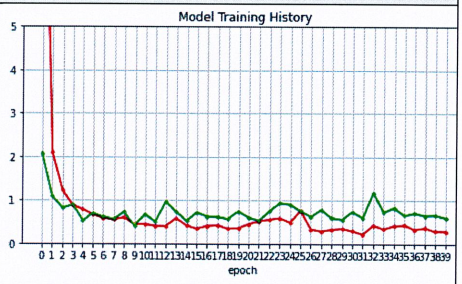
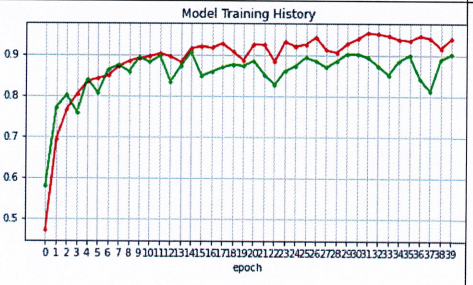
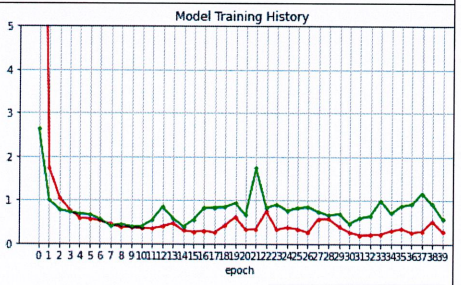
3.2.5 Swin ViT

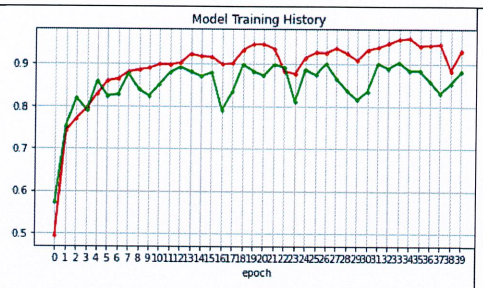
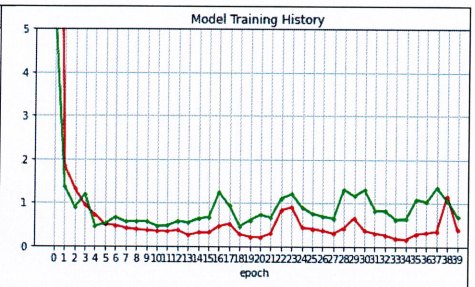
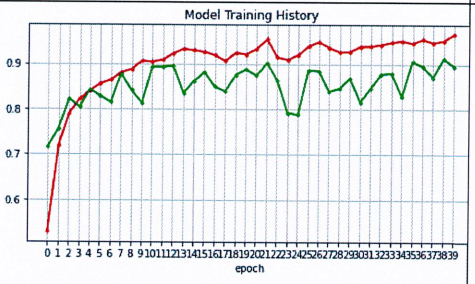
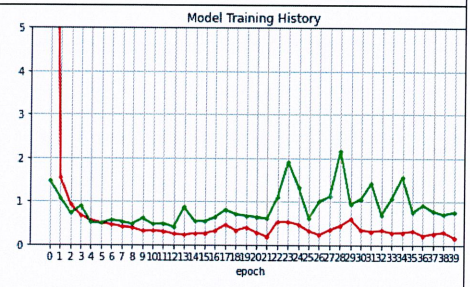
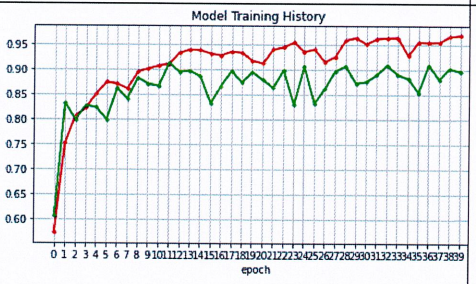
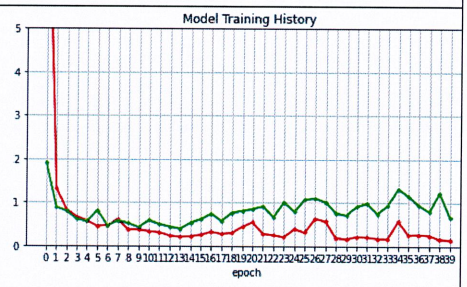
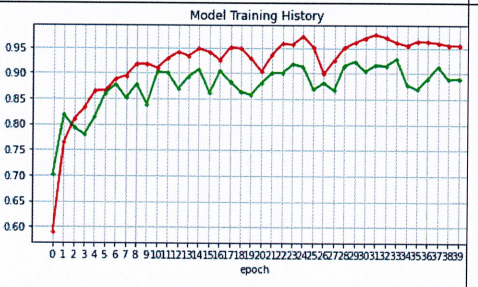
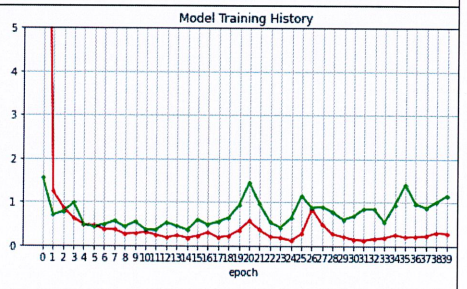
Each Swin-ViT block consists of 2 layers of transformers. The first layer is non-shifted window; the second layer is shifted window, as per Swin-

Transformer [29] design. The reason we choose Swin-Transformer instead of the plain Transformer ViT is because it is faster to train and less computationally expensive. As for performance, it should not be very different than the plain ViT architecture.

In this study, we increased the number of blocks gradually and recorded the results for each model. The models were trained as per the set-up section details. The results for the training and validation curves are show in the next table.

Table 3.3): Custom Swin-ViT results

Training (red) and Validation (green) Plots		
Swin-Vit Blocks	**Accuracy**	**Loss**
1	 **Figure (3.18): Accuracy of 1-block Swin-ViT.**	 **Figure (3.19): Loss of 1-block Swin-ViT.**
2	 **Figure (3.20): Accuracy of 2-block Swin-ViT.**	 **Figure (3.21): Loss of 2-block Swin-ViT.**

3		
	Figure 3.22): Accuracy of 3-block Swin-ViT.	**Figure (3.23): Loss of 3-block Swin-ViT.**
4		
	Figure (3.24): Accuracy of 4-block Swin-ViT.	**Figure (3.25): Loss of 4-block Swin-ViT.**
5		
	Figure 3.26): Accuracy of 5-block Swin-ViT.	**Figure (3.27): Loss of 5-block Swin-ViT.**
6		
	Figure (3.28): Accuracy of 6-block Swin-ViT.	**Figure (3.29): Loss of 6-block Swin-ViT.**

The below table shows the accuracy, precision, recall, and F1-Score of all evaluated Swin-Transformer models. The optimal model in accuracy is with 4 layers and also 5 layers. They seem to have equal testing accuracies.

However, 5 is more precise than 4 and 4 has higher recall and F1-score than 5. Also, it is smaller in size as such faster to train and more suitable for our small dataset.

Table 3.4): Summary of custom Swin-ViT results

Swin-ViT Blocks	Accuracy	Precision	Recall	F1-Score
1	0.86	0.88	0.78	0.84
2	0.91	0.94	0.98	0.96
3	0.90	0.89	0.99	0.94
4	0.92	0.96	0.98	0.97
5	0.92	0.98	0.95	0.96
6	0.90	0.97	0.90	0.94

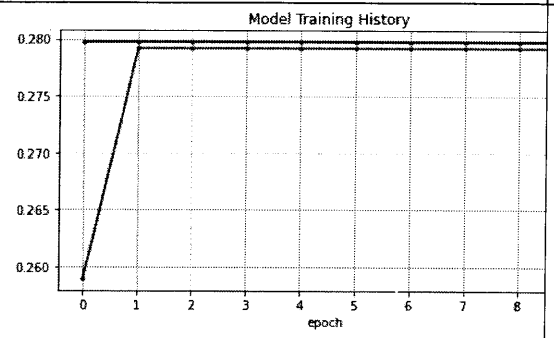
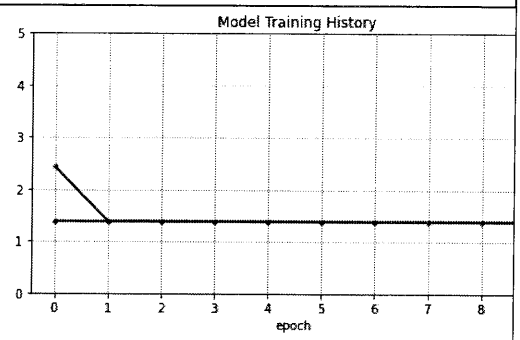
3.3 Pre-training

We will fine tune the pre-trained versions of the previous benchmark models which we used in the first section ResNet50, ResNet101, and VGG16. The models used in this section were all pretrained on ImageNet dataset. All models were trained for 10 epochs and augmentation as per this chapter's setup section. During the pre-training for all models, we froze the base model layers, and only trained the last three layers. Also, we added a dropout of 30% , a dense layer with 128 layers and ReLU activation, a dropout of 20%, and with a soft-max activation to perform the classification.

3.3.1 VGG16

An accuracy of 30.89% was obtained for VGG16. It still seems not able to converge despite the pre-training. The below curve shows the training of VGG16. The red represents the training and green the testing.

Table 3.5): Pre-trained VGG16 results

VGG16 fine-tuned	
Accuracy	Loss
 **Figure 3.30): Accuracy of Pre-trained VGG-16.**	 **Figure (3.31): Loss of Pre-trained VGG-16.**

3.3.2 ResNet50

An accuracy of 95.73% was obtained for ResNet50. The training and validation curves are shown below.

Table 3.6): Pre-trained ResNet50 results

ResNet50	
Accuracy	Loss

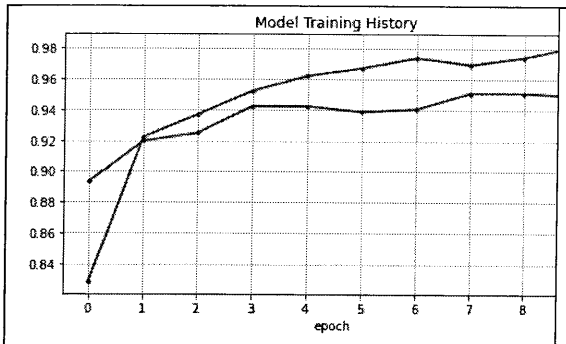


Figure 3.32): Accuracy of pre-trained ResNet-50.

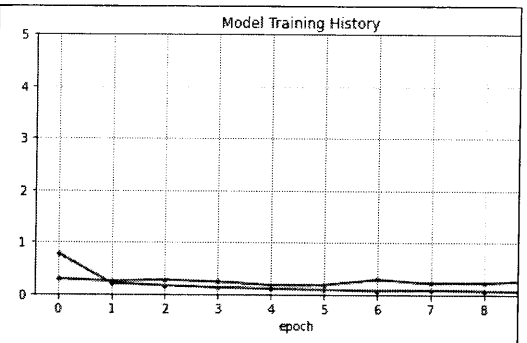


Figure (3.33): Loss of Pre-trained ResNet-50.

3.3.3 ResNet101

An accuracy of 96.72% was obtained for ResNet101. The training and validation curves are shown below, in red and green, respectively.

Table 3.7): Pre-trained ResNet101 results

ResNet101	
Accuracy	Loss

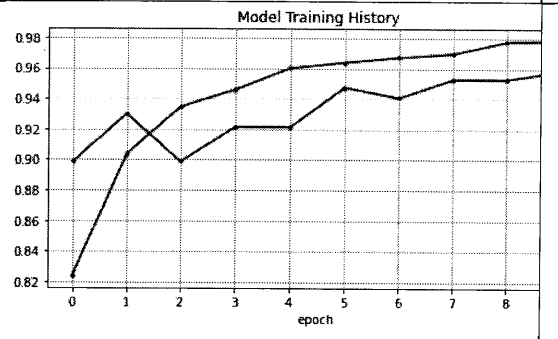


Figure (3.34): Accuracy of Pre-trained ResNet-101.

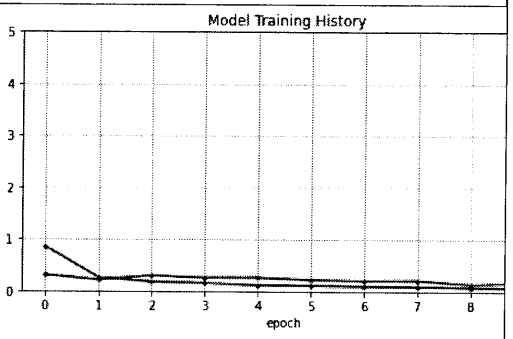


Figure (3.35): Loss of Pre-trained ResNet-101.

In the previous studies, the highest obtained accuracy is 97% by ResNet. Although it is very high, we aim in this work to get as high as possible to the highest accuracy in literature 99.02%. In this chapter, we explore the use of a different architecture which is known as CoAtNet [61]. This architecture combines both the local abilities of a convolutional neural network and the global abilities of a transformer. When pre-trained on ImageNet, the model achieves an accuracy of 88.56% on top 1. The reason

we choose this architecture is due to its enhanced scalability on all different sizes of data while combining the strengths of both convolutions and transformers. Zihang dai et al. [61] conducted many studies to test their performed model and measures its size, parameters, flops, and accuracy on ImageNet, as shown in Figure (***3.36*** (a) . They created four different variants in the model, the difference is that they increase in size. They found their model attains results that are either higher or comparable to the state of the art.

3.4 Theory behind CoAtNet

The architecture of a basic CoAtNet is shown in (b)

Figure (*3.36*. CoAtNet mainly consists of two different kinds of blocks: an MBConv [62] block and a relative attention block.

3.4.1 MBConv block

MBConv is an inverted residual convolutional block meaning that it follows the format of narrow-wide-narrow input size which is opposite to residual convolutions, as in [24]. The first narrow layer represents a convolutional layer that expands the input dimensions and passes it to the second layer. The second layer which is a wide layer represents a depth-wise convolutional layer. Such layers do not change the size of the input. The third layer is known a point-wise convolutional layer which performs a 1x1 convolution to reduce the dimension of the input features. The benefit of an inverse residual convolution is that it reduces computation by almost a factor of k^2. Although the computational cost is reduced, this reduction is on the expense of a slight loss of accuracy. Nevertheless, the main reason it is selected in the architecture of CoAtNet is that it facilitates for the incorporation of the proposed attention mechanism. The way they process

an image is similar to the way fully connected neural networks at the end of transformers do. In Figure 1.1, the yellow blocks represent MBConv blocks.

3.4.2 Relative self-attention

The attention mechanism implemented in CoAtNet is slightly different from that in a transformer. In transformers, it is a self-attention mechanism, whereas in CoAtNet, it is a relative attention mechanism which has also been proposed in previous literature work with many variations [61]. The variation present in CoAtNet is similar to the one proposed in [63]. The problem with self-attention in transformers is that they do not explicitly model relative or absolute position information in its structure. Relative attention, on the other hand, extends the self-attention to efficiently consider representations of relative positions of distance between sequence elements. Meaning, for example, if we have an image where there is in the foreground a man riding a horse in a park with trees in the background, a traditional transformer will be able to tell that there is a man riding a bike in a park with trees, but it will not be able to tell which is in the background and which is in the foreground of an image.

3.4.3 Model architecture

A convolution depends on a fixed kernel size to gather information from a local receptive field. A depth wise convolution is represented as:

$$y_i = \sum_{j \in \mathcal{L}(i)} w_{i-j} \odot x_j \qquad (3.1)$$

Where $x_i, y_i \in \mathbb{R}^D$ are the input and output at position i respectively, and $\mathcal{L}(i)$ denotes a local neighborhood of i. Self-attention, on the other hand, allows the receptive field to be the entire spatial locations and computes the

weights based on the re-normalized pairwise similarity between the pair (x_i, x_j):

$$y_i = \sum_{j \in \mathcal{G}} \underbrace{\frac{\exp(x_i^T x_j)}{\sum_{k \in \mathcal{G}} \exp(x_i^T x_j)}}_{A_{i,j}} x_j \tag{3.2}$$

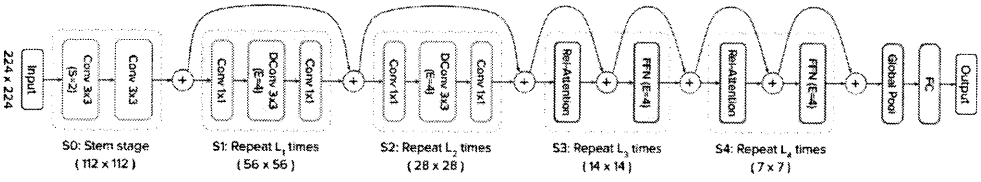
Where $\mathcal{G}$ is the global spatial space. In the convolutional operation, the depth-wise convolutional kernel w_{i-j} is an input-independent parameter of fixed value. Whereas in the attention operation, the attention weight $A_{i,j}$ is dynamic and it depends on the representations of the input. This characteristic makes it easier for attention to capture complicated relations between different spatial positions which is a desirable property when processing global or high-level features. However, when the data is limited, such flexibility comes at the risk of easily overfitting data. Also, convolutions have the property of being translationally equivalent which has been found to improve their generalizability. Transformers lack this property due to their usage of absolute positional embeddings. Given this comparison, combining the equations above gives:

$$y_i^{pos} = \sum_{j \in \mathcal{G}} \left(\frac{\exp(x_i^T x_j)}{\sum_{k \in \mathcal{G}} \exp(x_i^T x_j)} + w_{i-j} \right) x_j \tag{3.3}$$

$$y_i^{pre} = \sum_{j \in \mathcal{G}} \frac{\exp(x_i^T x_j + w_{i-j})}{\sum_{k \in \mathcal{G}} \exp(x_i^T x_j + w_{i-j})} x_j \tag{3.4}$$

Models		Eval Size	#Params	#FLOPs	ImageNet Top-1	
					1K only	21
Conv Only	EfficientNet-B7	600^2	66M	37B	84.7	
	EfficientNetV2-L	480^2	121M	53B	85.7	
	NFNet-F3	416^2	255M	114.8B	85.7	
	NFNet-F5	544^2	377M	289.8B	**86.0**	
ViT-Stem TFM	DeiT-B	384^2	86M	55.4B	83.1	
	ViT-L/16	384^2	304M	190.7B	-	
	CaiT-S-36	384^2	68M	48.0B	85.0	
	DeepViT-L	224^2	55M	12.5B	83.1	
Multi-stage TFM	Swin-B	384^2	88M	47.0B	84.2	
	Swin-L	384^2	197M	103.9B	-	
Conv+TFM	BotNet-T7	384^2	75.1M	45.8B	84.7	
	LambdaResNet-420	320^2	-	-	84.8	
	T2T-ViT-24	224^2	64.1M	15.0B	82.6	
	CvT-21	384^2	32M	24.9B	83.3	
	CvT-W24	384^2	277M	193.2B	-	
Conv+TFM (ours)	CoAtNet-0	224^2	25M	4.2B	81.6	
	CoAtNet-1	224^2	42M	8.4B	83.3	
	CoAtNet-2	224^2	75M	15.7B	84.1	
	CoAtNet-3	224^2	168M	34.7B	84.5	
	CoAtNet-0	384^2	25M	13.4B	83.9	
	CoAtNet-1	384^2	42M	27.4B	85.1	
	CoAtNet-2	384^2	75M	49.8B	85.7	
	CoAtNet-3	384^2	168M	107.4B	85.8	
	CoAtNet-4	384^2	275M	189.5B	-	
	+ PT-RA	384^2	275M	189.5B	-	
	+ PT-RA-E150	384^2	275M	189.5B	-	
	CoAtNet-2	512^2	75M	96.7B	85.9	
	CoAtNet-3	512^2	168M	203.1B	**86.0**	
	CoAtNet-4	512^2	275M	360.9B	-	
	+ PT-RA	512^2	275M	360.9B	-	
	+ PT-RA-E150	512^2	275M	360.9B	-	

(a)



(b)

Figure (3.36): CoAtNet Architecture and performance on ImageNet.

(Image source: [61])

CoAtNet is designed in a way to retain desired properties in convolutions and attention, as shown in Figure 3.37. Convolutions has the translation equivariance property, as mentioned before, which is a desirable property in computer vision because it allows the recognition of an object in different locations within an image. For example, given an image of an apple, if we shift the apple to the right, CNN is able to detect it. In self-attention, two important properties that we desire are the input-adaptive weighting and global receptive fields. The first property means that attention is able to focus on the most relevant part of an image an ignore the rest or noisy information. The second property, global receptive field, allows an attention to capture global relationships between elements.

Properties	Convolution	Self-Attention
Translation Equivariance	✓	
Input-adaptive Weighting		✓
Global Receptive Field		✓

Figure 3.37): Properties reserved in CoAtNet.

(Image source: [61])

3.5 Study Setup

There is a number of variations for CoAtNet. We selected CoAtNet-0 as it is the smallest one and most suitable for small datasets. Also, for TensorFlow, it is the only variation of CoAtNet which we found a pre-trained model for on ImageNet. The studies were performed on Google Colab Pro+ with GPU NIVIDIA V100. Various studies were conducted, and various scenarios were explored. They all had a common setup. Unless otherwise mentioned, the selected loss function is cross-categorical loss entropy, the optimizer is ADAM with default Keras settings, and the

number of epochs is 50. The dataset is kept as its original split of testing and training (refer to chapter 1 under the dataset section). A 10% from training set is allocated to validation. On all occasions, we trained the model end to end, meaning that no layers were frozen. Also, in no scenario did we add any additional layers to the model.

The training was first done on the non-pretrained model. Then, the rest of the studies were on the pre-trained model that we found it has better potential than the non-pretrained. In the first two studies, we did not add any augmentations and we used default settings for the optimizer, as per TensorFlow, such as the learning rate was kept as 0.0001. In the pre-trained model study, we gradually added augmentations to observe closely their effect on enhancing model performance. However, we avoided certain augmentations such as cropping, as for this dataset it may create confusion to the model when it crops out the tumor, for example, causing the model to possibly diverge. Also, the learning rate was gradually further decreased, as it has been found that for this particular model and dataset, a smaller learning rate is better. Later, a scheduled exponential decay learning rate which is expressed as:

$$I_r = I_{r0}\, exp(-kt) \qquad (3.5)$$

Where I_{r0} is the initial learning rate, t is the iteration or epoch, k is a hyperparameter called decay rate which we set as 0.96, as per Keras default settings.

3.6 Results of CoAtNet

We conducted two initial studies. In the first one, the model was trained from scratch. In the second one, the model was pretrained on Image-Net dataset. The following shows the results of CoAtNet-0 pre-trained. Be noted that for this chapter, the results are displayed in the format of curves

for loss and accuracy of both training and validation. The convention we follow is similar to previous chapters. The red represents the training curve, the green represents the validation curve. Also, we added a performance report which shows the accuracy, F1-Score, Precision, and Recall. Keep in mind that this table rounds up values, as such the figures are not exact. Also, we will display a confusion matrix to give us an idea where the model is classifying correctly, and where it is not. The convention of the confusion metric is as follows:

- 0 – Glioma
- 1 – Meningioma
- 2 – No tumor
- 3 – Pituitary

3.6.1 Non-pretrained model

The following are the results of the non-pretrained model. We can observe that despite the model reaching 88% final accuracy, it could have reached higher results, but clearly it suffered from fluctuating training. The possible cause of this fluctuation is large network size compared to small dataset. Increasing the batch size.

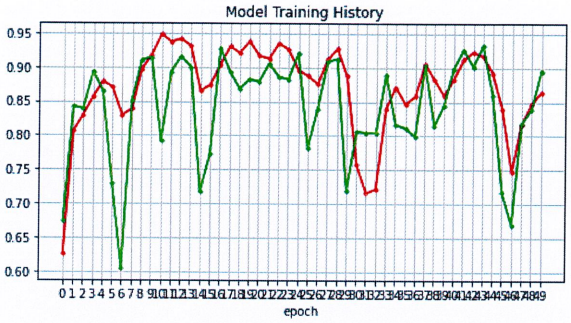


Figure (3.38): Non-pre-trained CoAtNet Accuracy.

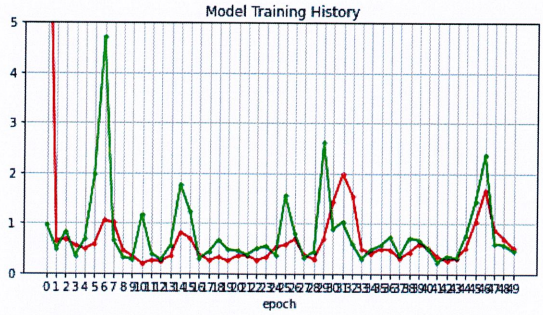


Figure (3.39): Non-pretrained CoAtNet Loss.

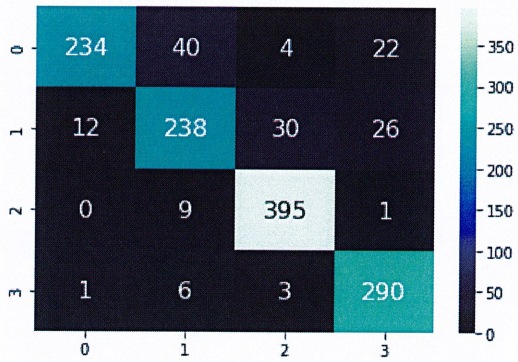


Figure (3.40): non-pre-trained CoAtNet confusion matrix.

	precision	recall	f1-score	support
0	0.95	0.78	0.86	300
1	0.81	0.78	0.79	306
2	0.91	0.98	0.94	405
3	0.86	0.97	0.91	300
accuracy			0.88	1311
macro avg	0.88	0.87	0.88	1311
weighted avg	0.88	0.88	0.88	1311

Figure 3.41): non-pre-trained CoAtNet table of results.

3.6.2 pre-trained model

A pre-trained model on Image-Net was used. The model was trained end-to-end, meaning that no layers were frozen during training. The model seems relatively more stable than the previous one in terms of training. Also, it has reached an equal value to our highest achieved score so far 97%.

Multiple scenarios were studied in which the model performance was improved. To summarize the scenarios for easy reference, they are as follows:

- scenario 1 – plain pre-training training
- scenario 2 – pre-training with increased batch size (LB)
- scenario 3 – pre-trained with increased batch size and one augmentation (LB+AUG)
- scenario 4 – pre-trained with increased batch size, 1 augmentation and decay learning (LB+1AUG+DLR)
- scenario 5 – Pretrained with increased batch, 1 augmentations, normalization, and decay learning (LB+AUG+NRM+DLR)
- scenario 6 – Pretrained with increased batch, 2 augmentations, and decay learning (LB+2AUG+NRM+DLR)

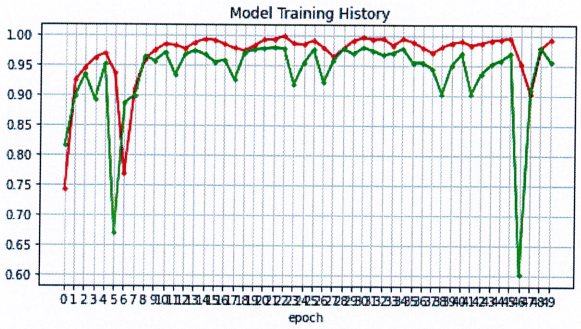


Figure (3.42): plain pre-trained CoAtNet Accuracy.

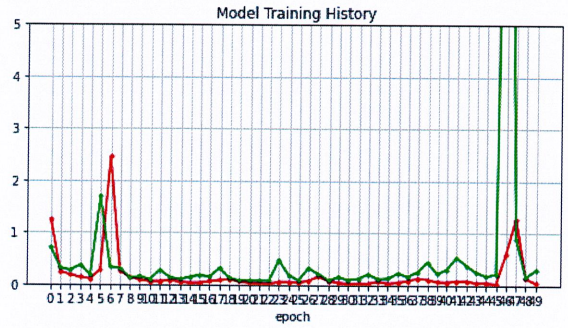


Figure (3.43): plain pre-trained CoAtNet Loss.

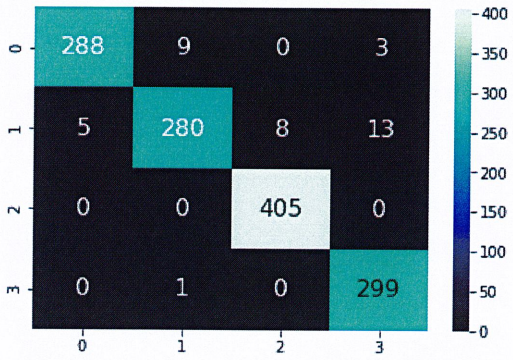


Figure (3.44): plain pre-trained CoAtNet confusion matrix.

	precision	recall	f1-score	support
0	0.98	0.96	0.97	300
1	0.97	0.92	0.94	306
2	0.98	1.00	0.99	405
3	0.95	1.00	0.97	300
accuracy			0.97	1311
macro avg	0.97	0.97	0.97	1311
weighted avg	0.97	0.97	0.97	1311

Figure (3.45): plain pre-trained CoAtNet table of results.

Since the results seem satisfactory, we repeated training for pre-trained mode and increased the number of batches to 100 to attempt to achieve higher performance and reduce the fluctuation of the training [64]. Large batch size reduces fluctuations because it reduces noise in SGD [64]. It also reduces the variance of the SGD which is in our case necessary given the large size of the model compared to the size of the dataset; thereby decreasing overfitting. The results are shown in the proceeding figures. We can see a more stable training which proves the fact that increasing the batch size stabilizes the training and reduces fluctuations. The accuracy was higher than before, and it is equal to 98.32% (the numbers shown in the figure are rounded up).

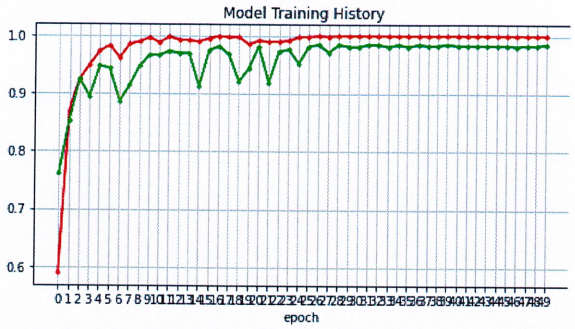


Figure (3.46): second scenario accuracy.

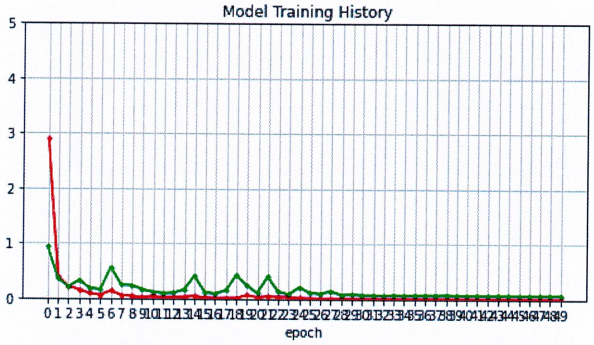


Figure 3.47): second scenario loss.

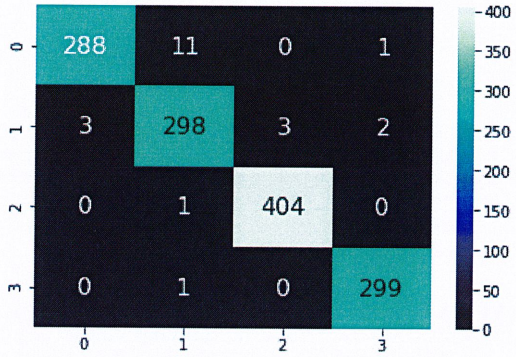


Figure 3.48): second scenario confusion matrix.

	precision	recall	f1-score	support
0	0.99	0.96	0.97	300
1	0.96	0.97	0.97	306
2	0.99	1.00	1.00	405
3	0.99	1.00	0.99	300
accuracy			0.98	1311
macro avg	0.98	0.98	0.98	1311
weighted avg	0.98	0.98	0.98	1311

Figure (3.49): second scenario table of results.

Another attempt to increase the accuracy is by adding augmentations. The augmentation added is random horizontal flip. This is supposed to decrease the gap between training and testing which would lead to higher accuracy. The results are shown in figures 3.50 to 3.53 with accuracy equal to 97.4%.

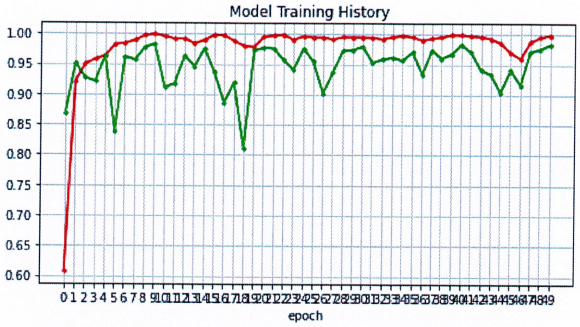


Figure (3.50): third scenario accuracy.

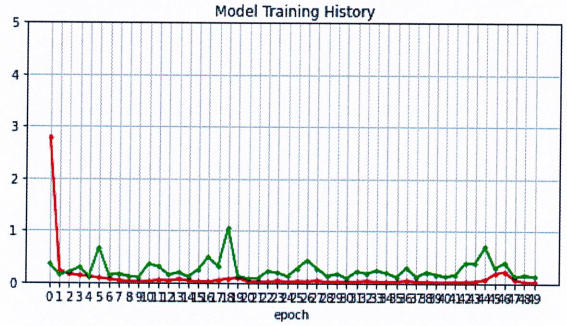


Figure 3.51): third scenario loss.

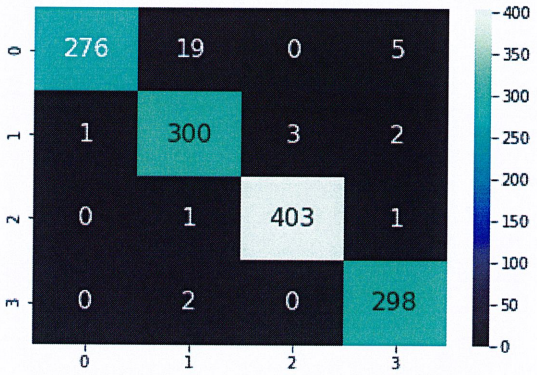


Figure 3.52): third scenario confusion matrix.

	precision	recall	f1-score	support
0	1.00	0.92	0.96	300
1	0.93	0.98	0.96	306
2	0.99	1.00	0.99	405
3	0.97	0.99	0.98	300
accuracy			0.97	1311
macro avg	0.97	0.97	0.97	1311
weighted avg	0.97	0.97	0.97	1311

Figure (3.53): third scenario results table.

Although the training curve seems stable, the validation curve is fluctuating. Adding augmentations to the dataset may have delayed proper convergence. In an attempt to improve validation profile, we apply exponential decay for the learning with an initial rate of 0.0001. Results show a drastic improvement in accuracy. The obtained accuracy is 99.08%.

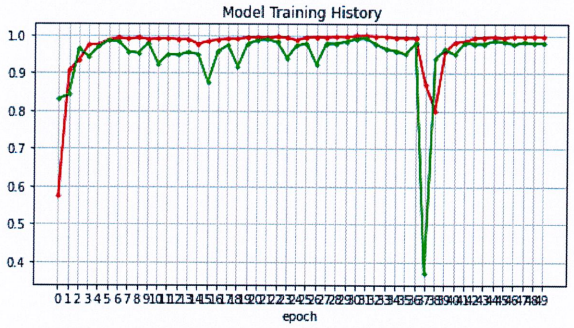


Figure (3.54): fourth scenario accuracy.

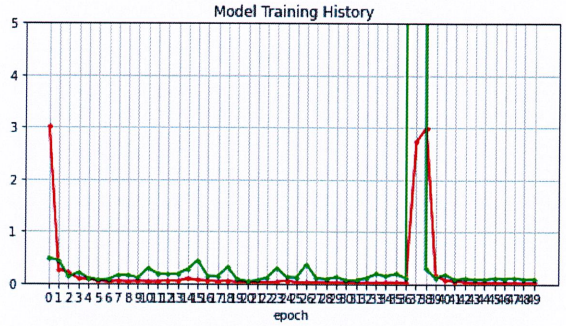


Figure 3.55): fourth scenario loss.

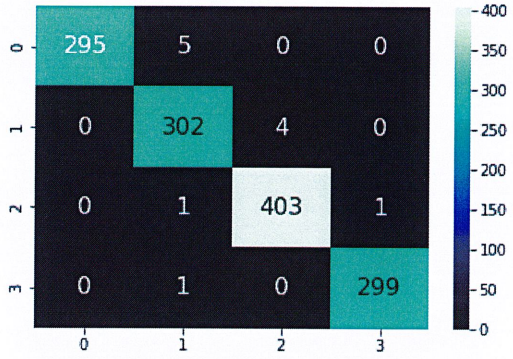


Figure 3.56): fourth scenario confusion matrix.

	precision	recall	f1-score	support
0	1.00	0.98	0.99	300
1	0.98	0.99	0.98	306
2	0.99	1.00	0.99	405
3	1.00	1.00	1.00	300
accuracy			0.99	1311
macro avg	0.99	0.99	0.99	1311
weighted avg	0.99	0.99	0.99	1311

Figure (3.57): fourth scenario results table.

We applied normalization in addition to random flipping augmentation and exponential learning rate decay. We got the following learning profile with accuracy 98.86%. The accuracy is lower than before, but the training seemed more stable against sudden dips.

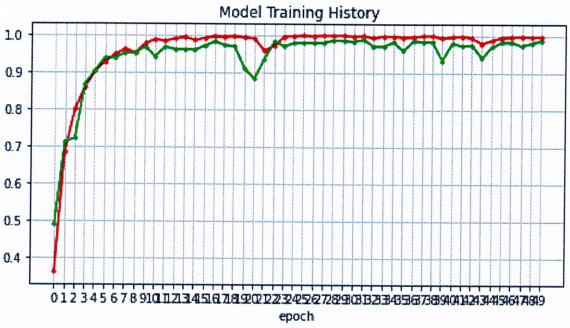


Figure (3.58): fifth scenario accuracy.

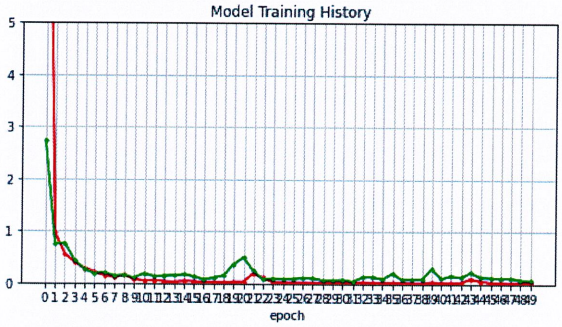


Figure 3.59): fifth scenario loss.

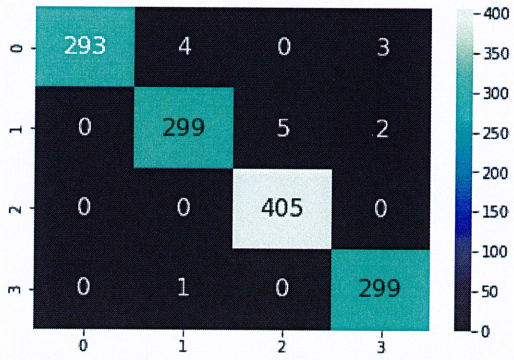


Figure 3.60): fifth scenario confusion matrix.

	precision	recall	f1-score	support
0	1.00	0.98	0.99	300
1	0.98	0.98	0.98	306
2	0.99	1.00	0.99	405
3	0.98	1.00	0.99	300
accuracy			0.99	1311
macro avg	0.99	0.99	0.99	1311
weighted avg	0.99	0.99	0.99	1311

Figure (3.61): fifth scenario results table.

One more augmentation was added to further improve model generalizability which is random rotation by factor 0.2. The accuracy improved to 99.16%. results are shown below. We also notice a greater gap between validation and training which indicates. Nevertheless, the accuracy is satisfactory and the highest so far in literature.

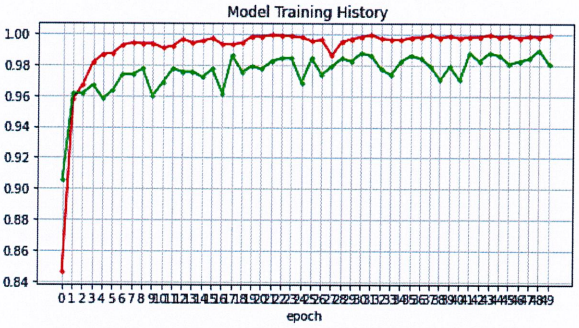


Figure (3.62): sixth scenario accuracy.

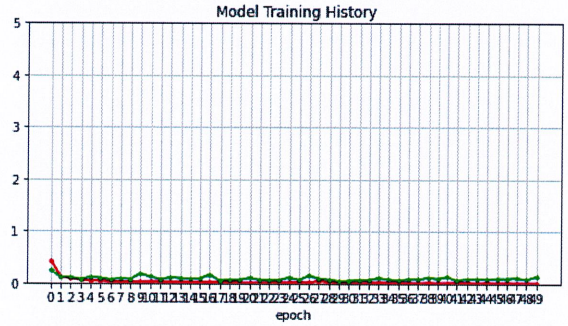


Figure (3.63): sixth scenario loss.

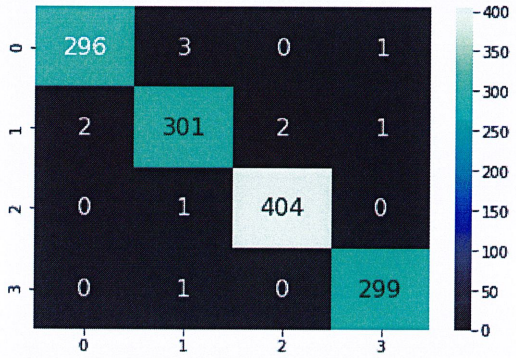


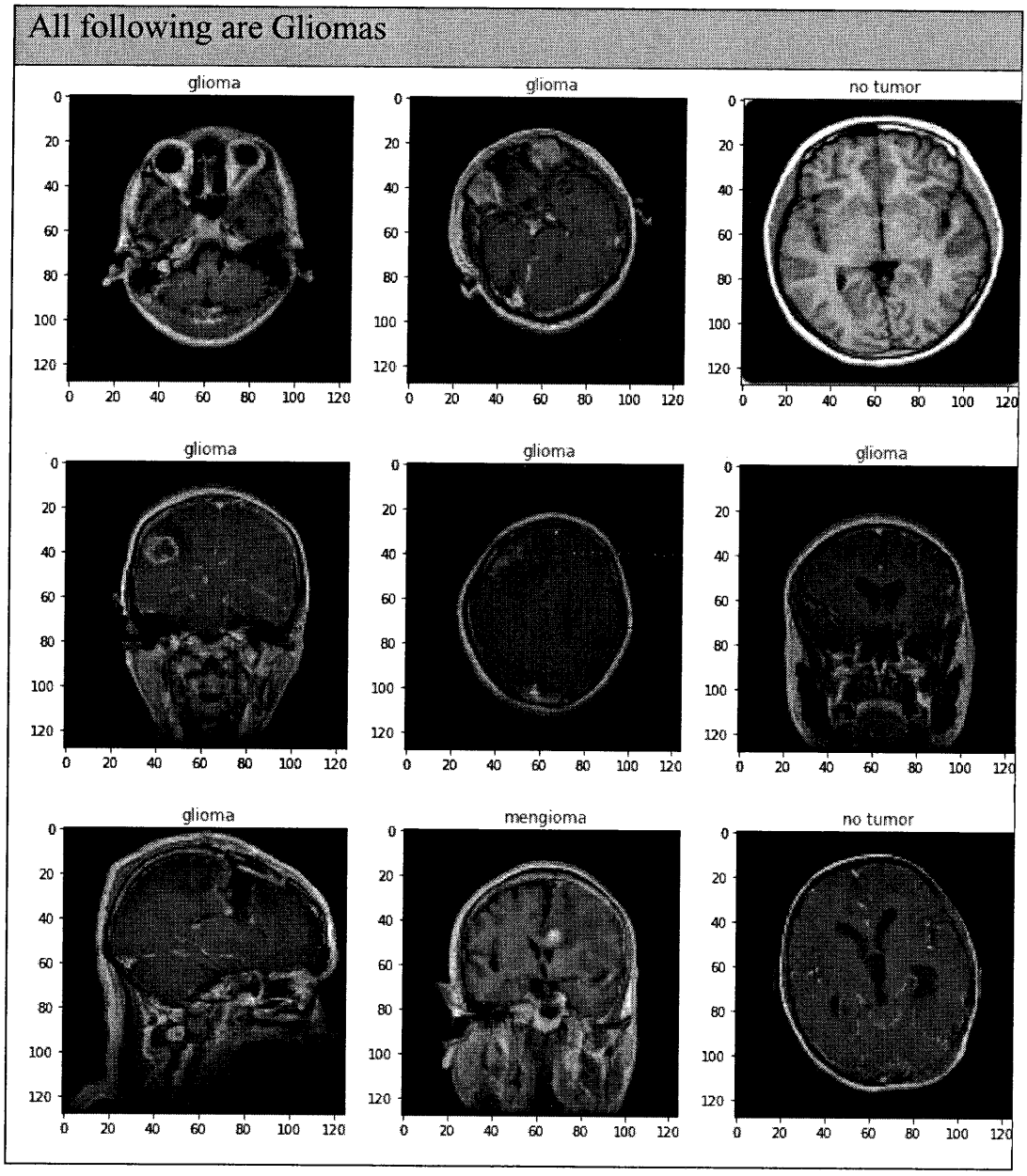
Figure (3.64): sixth scenario confusion matrix.

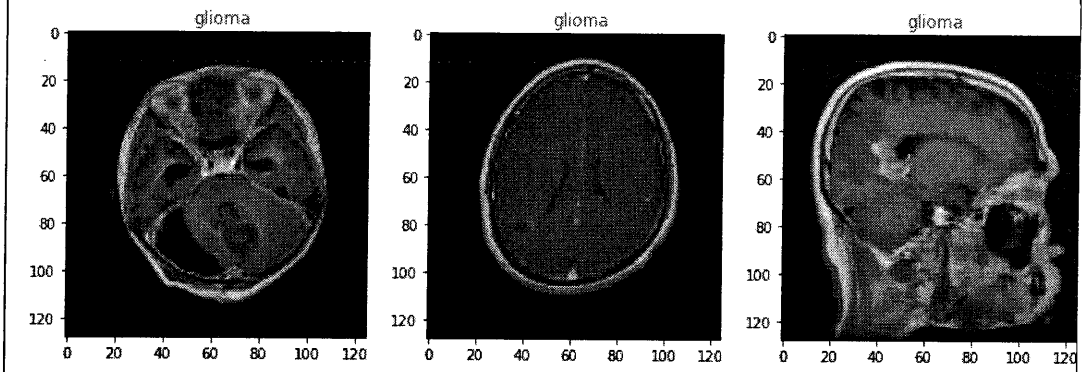
	precision	recall	f1-score	support
0	0.99	0.99	0.99	300
1	0.98	0.98	0.98	306
2	1.00	1.00	1.00	405
3	0.99	1.00	1.00	300
accuracy			0.99	1311
macro avg	0.99	0.99	0.99	1311
weighted avg	0.99	0.99	0.99	1311

Figure (3.65): sixth scenario results table.

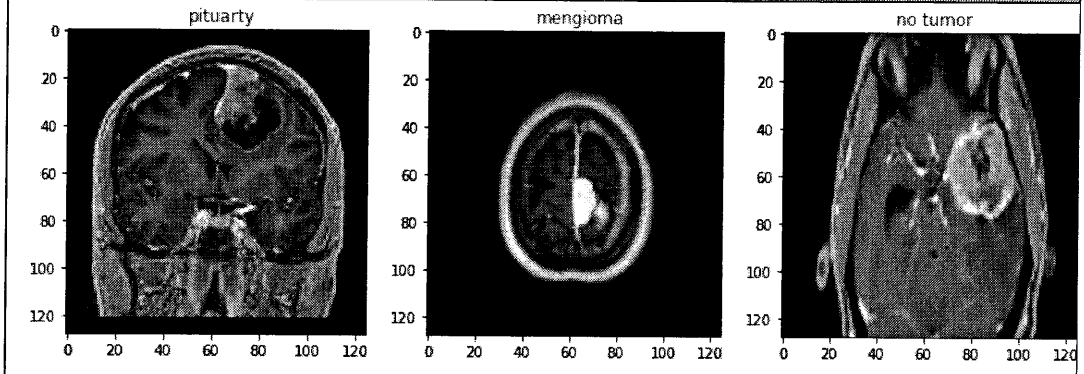
Below are some examples of test samples classified by the highest accuracy model. In green, correctly classified, and in red, misclassified.

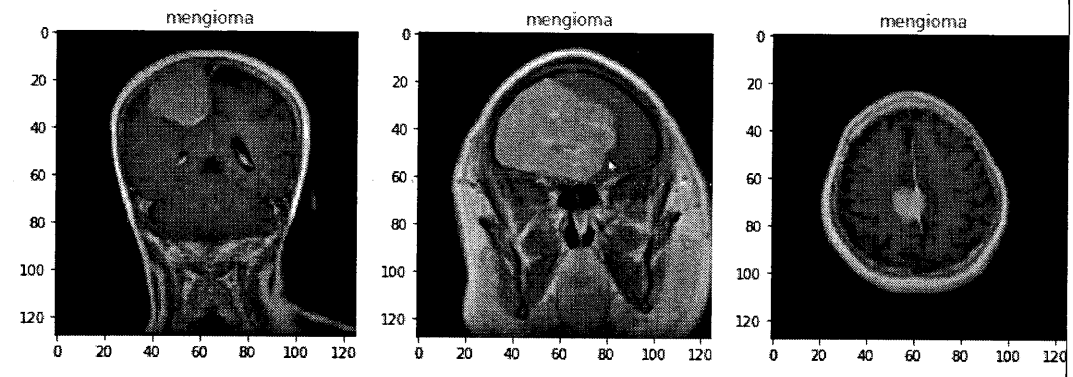
Table 3.8): Samples of classification of highest performing CoAtNet

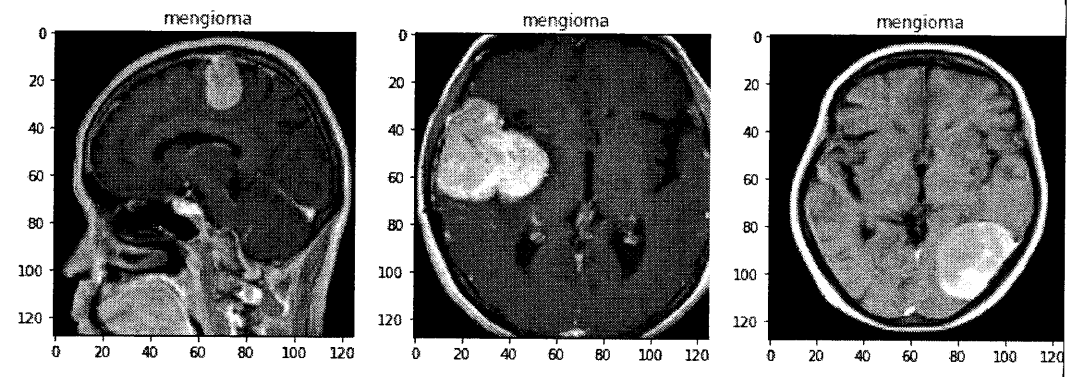


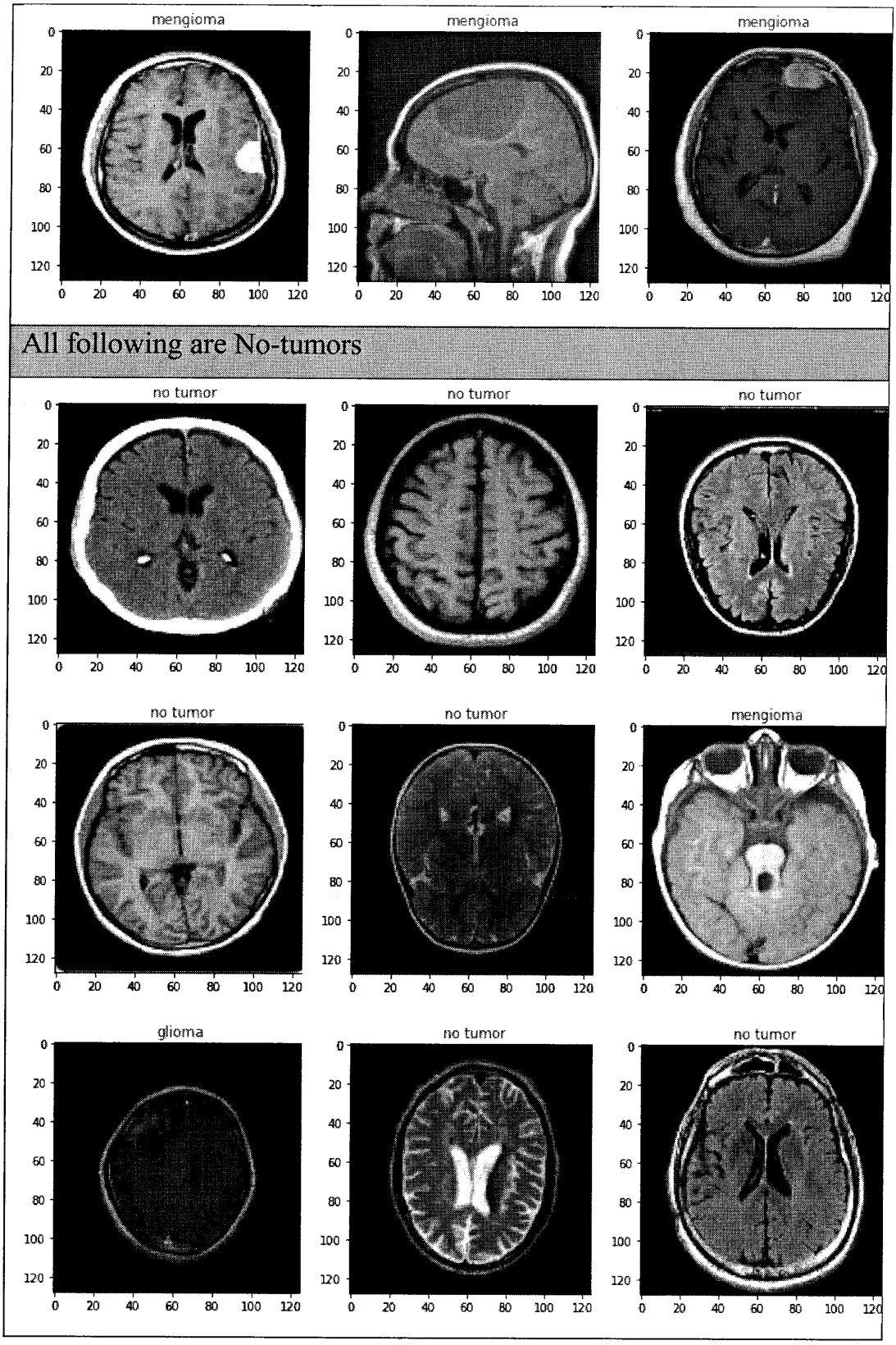


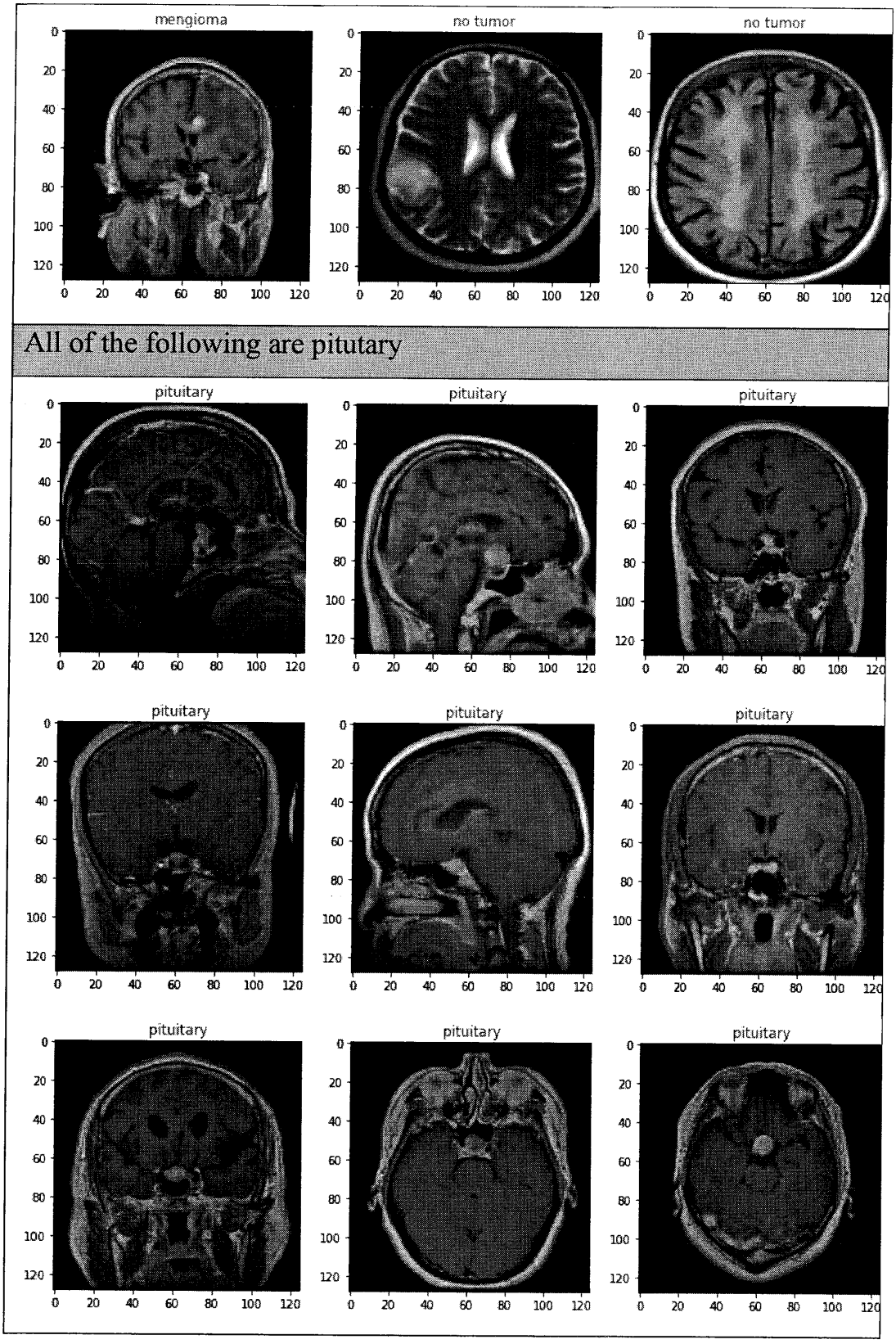
All following are Menegiomas

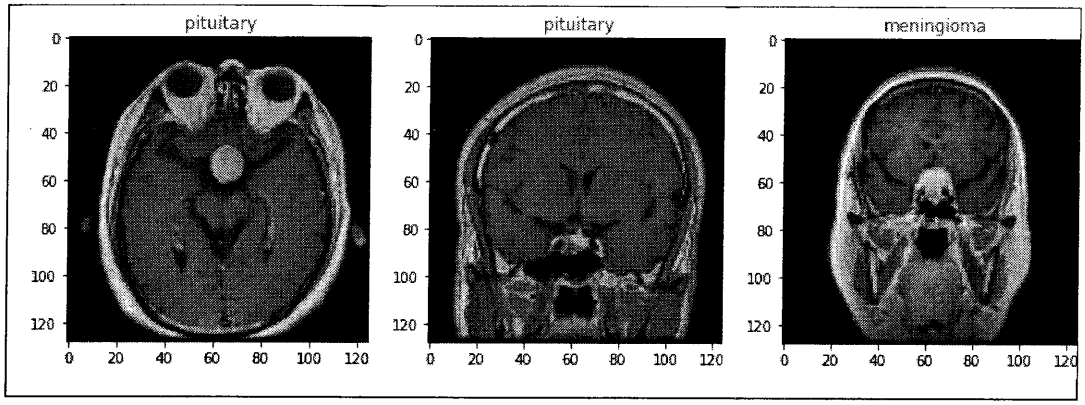







mengioma
mengioma
mengioma
All following are No-tumors
no tumor
no tumor
no tumor
no tumor
no tumor
mengioma
glioma
no tumor
no tumor


mengioma
no tumor
no tumor
All of the following are pituitary
pituitary
pituitary
pituitary
pituitary
pituitary
pituitary
pituitary
pituitary
pituitary



3.7 Discussion

In the proceeding studies conducted on CoAtNet we observed a clear relation between accuracy and the pretraining of a model. We noticed that the model jumped from 88% to 97% from non-pretrained to pre-trained, respectively. This is expected because first CoAtNet is quite a large model, even though we used the smallest size variation which is CoAtNet-0, and second the dataset is considered as rather small which is why it is more likely to overfit the dataset causing low accuracies.

During our studies, we also found that to enhance a model performance adding augmentations plays a very important role. Augmentation by nature extends a dataset, thereby serving the effect of adding regulation to the dataset. The other thing we found was that tweaking the learning rate of the model further enhances performance. In addition, increasing the batch size clearly stabilizes the model and eliminates oscillation during training. However, when we increased the batch size it took from the GPU memory a large space. In fact, during conducting our studies, it was intended to increase the batch size up to as large batch as possible. Nevertheless, our GPU RAM was overwhelmed by the size of the batch and limited out ability to consider such a scenario. Nevertheless, the purpose of this model is to also be practical for industrial or medical implementation. As such, limiting

the number of batches to 100 is not an issue. The full range of accuracies that we obtained is shown in Figure (*3.66*.

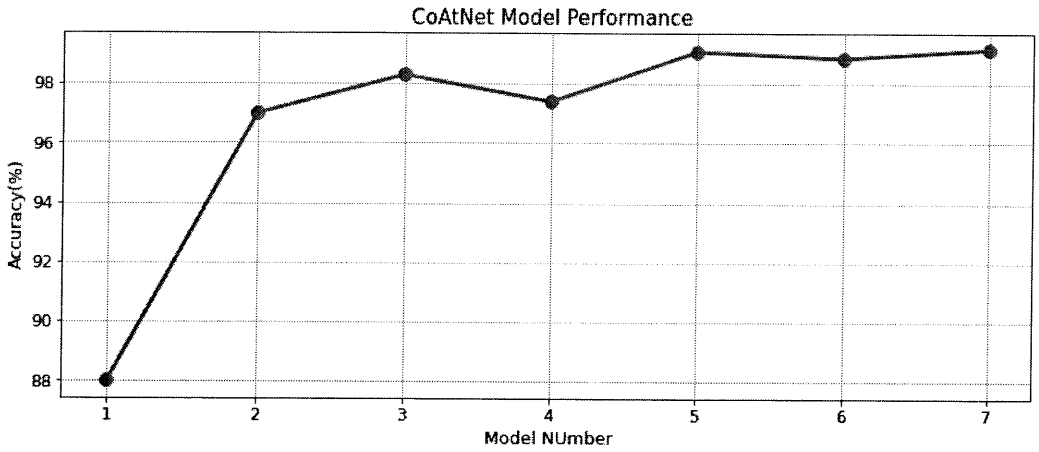


Figure (3.66): CoAtNet performance summary with different modifications

It can be seen that the best result achieved is on the greatest number of augmentation and large batch size and implementation of exponential decay rate. Below table also gives a clearer view.

Table 3.9): Summary of CoAtNet results

Number	Model	Accuracy (%)
1	Non-pretrained	88.00
2	Pre-trained	97.00
3	pre-trained + LB	98.30
4	pre-trained + LB + AUG	97.40
5	pre-trained + LB + AUG + DLR	99.08
6	pre-trained + LB + AUG + NRM+ DLR	98.86
7	pre-trained + LB + 2AUG + NRM + DLR	99.16

Since the model and method of this chapter turn out to be successful, they will be considered as our final results.

Chapter Four

Model Soups Algorithm

In this chapter, we expand on the models we designed in previous chapters by using model soups algorithm [65]. We specifically choose this algorithm for the high score it achieved on larger datasets such as ImageNet. As of now, the algorithm has achieved a top-1 accuracy of 90.88% which is the second highest.

4.1 Introduction

Model soup algorithm is a recently proposed model, and it has shown a promising success on many different large datasets such as ImageNet. The usual way to maximize a model's accuracy is to train multiple models with various hyper parameters and pick the individual model which has the highest performance, while either discarding the remaining models which makes it a waste of computational resources, or keeping them, or at least the highest performing of them, and applied to this group of models ensembling. However, ensembling tends to require more time and larger memory during inferencing which could also pose a computational challenge in practical applications. If we suppose for instance that this program was implemented as part of an MRI software, it will make it really slow and expensive if the model running in the background is actually an ensemble of model. As such, Soup Algorithm revisits this procedure by showing that averaging the models' weights into a single model could give high accuracy. Unlike ensembling, model soup could average tens of models without incurring any additional inference of memory cost.

The algorithm has a very simple concept. Multiple model which must be of identical architecture are trained with different hyper parameters, for example, different loss functions, optimizers, epochs, or other techniques and variations. Then, the weights of these models is taken and is averaged. The average weight overwrites the original model weights.

The weights of each trained model were recorded to apply model soups which involves the averaging of model weights into a single weight, expressed as follows.

$$\boldsymbol{\theta_S} = \frac{1}{N}\sum_{i \in N} \boldsymbol{\theta_i} \qquad 4.1)$$

The uniform soup model is expressed as below.

$$f(x, \boldsymbol{\theta_S}) \qquad 4.2)$$

4.2 Vision Transformers with Model Soup Algorithm

In [2], Sudhakar Tummala et al. used Ensembling to improve their vision transformer results. Ensembling techniques improved their overall accuracy from 98.2% to 98.7%. They used the soft-max average of all model's prediction and found the $argmax$ of the final output to produce their ensembled prediction which is mathematically expressed as follows:

$$pred = argmax(\frac{1}{N}\sum_{i=1}^{N} softmax_i) \qquad (4.3)$$

Where N represents the individual model. However, recent publication by Mitchell Wortsman et al. [65] proposes the use of a more efficient technique which they called 'Model Soups'. The algorithm tends to be much better than Ensembling in performance, also it does not increase the inferencing time because the prediction will only be taken from a single model rather than multiple models at the same time. For this reason, we will choose the optimal Swin-ViT architecture which scored 92% and train a number of variations from it. We created multiple combinations and trained them by varying the optimizer, loss function, and learning rate. In some instances, we used exponentially decaying learning rate. The various loss functions used are:

- Adam
- Stochastic Gradient Descent
- Adadelta
- RMS prop

The learning rates used are 0.00001, 0.0005, and an exponentially decaying learning rate of initial value 0.01, with a decay rate of 0.9. The various loss function used are:

- Categorical cross entropy
- Poisson
- KL-Divergence

The number of epochs was kept as 40 epochs. The models are numbered as follows:

Table 4.1): Model numbering convention

Categorical cross entropy loss				
Learning Rate	**Adam**	**SGD**	**Adadelta**	**RMS-prop**
0.00001	1	2	3	4
0.0005	5	6	7	8
Exponential decay	9	10	11	12
Poisson loss				
Learning Rate	**Adam**	**SGD**	**Adadelta**	**RMS-prop**
0.00001	13	14	15	16
0.0005	17	18	19	20
Exponential decay	21	22	23	24
KL-Divergence loss				
Learning Rate	**Adam**	**SGD**	**Adadelta**	**RMS-prop**
0.00001	25	26	27	28
0.0005	29	30	31	32
Exponential decay	33	34	35	36

4.2.1 Results of soups

The results of each individual transformer model are shown in Table 4.2): Results of custom Swin-ViT **soups**. We observe that under some combinations of loss function, optimizer, and learning rate the accuracy is much higher than in others. The optimal training model was number 4 which had cross-categorical loss and RMS prop optimizer with a learning rate of 0.000001. We also noticed that once the loss function and learning rate are change, the results are un-satisfying.

Table 4.2): Results of custom Swin-ViT soups

Model number	accuracy
1	0.91
2	0.23
3	0.41
4	0.93
5	0.91
6	0.71
7	0.35
8	0.92
9	0.42
10	0.23
11	0.67
12	0.51
13	0.23
14	0.31
15	0.23
16	0.23
17	0.23
18	0.23
19	0.38
20	0.23
21	0.23
22	0.63
23	0.23
24	0.23
25	0.23

26	0.23
27	0.34
28	0.32
29	0.23
30	0.23
31	0.31
32	0.23
33	0.32
34	0.23
35	0.41
36	0.23

Despite the low performances obtained in some models of the table above, we still took the uniform soup of them all. We obtained the following performances for the Swin-transformer model soup.

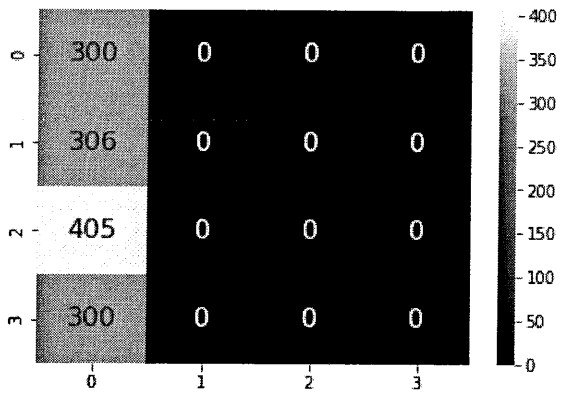


Figure (4.1): confusion matrix for Swin-ViT uniform soup.

	precision	recall	f1-score	support
0	0.23	1.00	0.37	300
1	0.00	0.00	0.00	306
2	0.00	0.00	0.00	405
3	0.00	0.00	0.00	300
accuracy			0.23	1311
macro avg	0.06	0.25	0.09	1311
weighted avg	0.05	0.23	0.09	1311

Figure (4.2): results table for Swin-ViT uniform soup.

The overall accuracy was 0.23 which is expected because multiple models showed this low accuracy. In [65], they discussed using an improved version of model soup algorithm called greedy soup. They suggested that this improvement adjusts the model to only accept higher resulting accuracies with every soup integration. The algorithm is as shown next.

Algorithm 5 *Greedy Model Soup Algorithm*

Given input: *Potential soup* $\{\theta_1, \dots, \theta_k\}$ *(sorted in decreasing order of* $ValAcc(\theta_i)$*).*

for $i = 1$ ***to*** k ***do***

if $ValAcc\big(average(model\ parameter \cup \{\theta_i\})\big) \geq$

$ValAcc(average(model\ parameter))$ ***then***

return $average(model\ parameter)$.

Applying the greedy soup algorithm produced the next results.

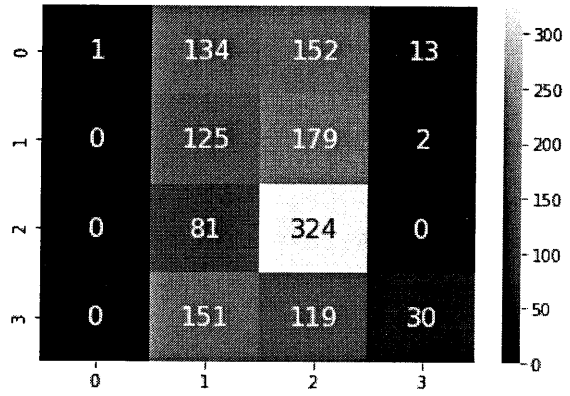


Figure (4.3): confusion matrix for Swin-ViT greedy soup.

	precision	recall	f1-score	support
0	1.00	0.00	0.01	300
1	0.25	0.41	0.31	306
2	0.42	0.80	0.55	405
3	0.67	0.10	0.17	300
accuracy			0.37	1311
macro avg	0.58	0.33	0.26	1311
weighted avg	0.57	0.37	0.28	1311

Figure (4.4): results table for Swin-ViT uniform soup.

The results again, despite being higher than before, do not meet expectations. We attribute that to the fact that the performances of the other

variants are inclining the model towards low performance. In addition, during the execution of the greedy soup algorithm, we observed that despite initializing the algorithm with model number 1 which is among the most highly performing models, the uniform averaging of this model with model number 2 which is also highly performing produces a model that is far from and below their accuracies.

The comparison between the souping of our models and the Ensembling of [2] models may not be fair. First, the model used in their paper is a pre-trained vision transformer model. It was pretrained on Image-Net dataset. Ours is not pre-trained which explains the lower performance. Also, although Ensembling has produced for them higher accuracy, it was only by a 0.2 percent which is marginal. The model soup technique which we used on model maybe further investigate on pre-trained models. Also, given our results, it is advised that future studies avoid the use of model soup if using different loss functions and optimizers shows very different accuracies.

We will consider the highest performance for Swin-transformer model to be 93% as per model 4. However, we will perform a second trial of model soup on a pre-trained ResNet to see if we get better results. We chose ResNet [22] [50] instead of transformers for the fact that a pre-trained model is freely available and easily accessible in Keras. Also, because past literature work showed the highest performance while using them, even higher than transformers. More details are presented in the next sections.

4.3 ResNet Souping Algorithm

The paper proposing the Soup Model Algorithm [65] originally proposed its use on pre-trained models. For this reason, instead of creating a model from scratch, we will use a pre-trained model. We choose ResNet because

from previous literature it showed a very high score of 99.02% [50]. We specifically choose ResNet50 variation, following their convention. We used the same model combinations and numbering convention as in Table 4.3.

4.3.1 Results of soups

The accuracies we obtained for induvial ResNet models is shown in Table 4.3.

Table 4.3): Results of Pre-trained ResNet50 soups

ResNet Model number	accuracy
1	0.95
2	0.95
3	0.79
4	0.96
5	0.96
6	0.92
7	0.76
8	0.97
9	0.92
10	0.95
11	0.91
12	0.91
13	0.44
14	0.95
15	0.79
16	0.95
17	0.54

18	0.89
19	0.75
20	0.96
21	0.41
22	0.96
23	0.91
24	0.71
25	0.52
26	0.96
27	0.81
28	0.68
29	0.92
30	0.74
31	0.75
32	0.91
33	0.23
34	0.95
35	0.93
36	0.50

We observe from Table 4.3 that the accuracies of different ResNet50 variations are much higher than those of Swin-ViT. Also, the highest obtained score is 97% which belongs to model number 8. This gives an indication that this dataset tends to achieve higher score with RMS prop optimizer and cross categorical loss entropy. When we apply uniform model soup to the models obtained, we get the following results.

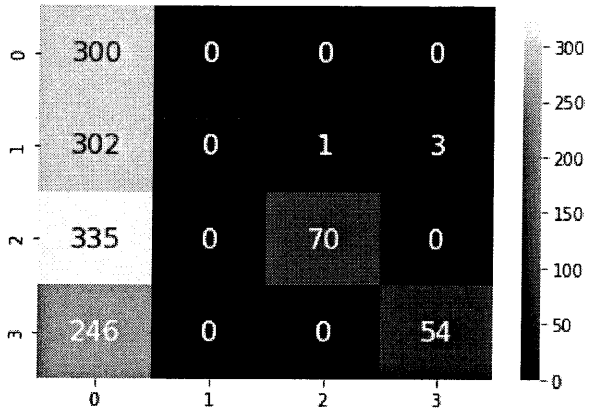


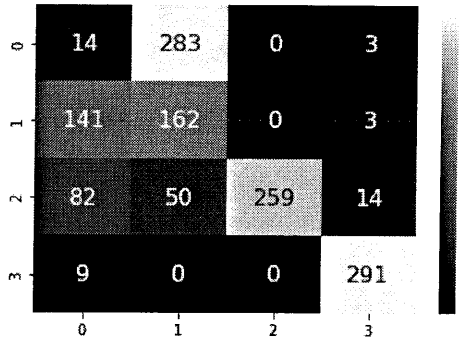
Figure 4.5): confusion matrix for ResNet50 uniform soup.

	precision	recall	f1-score	support
0	0.25	1.00	0.40	300
1	0.00	0.00	0.00	306
2	0.99	0.17	0.29	405
3	0.95	0.18	0.30	300
accuracy			0.32	1311
macro avg	0.55	0.34	0.25	1311
weighted avg	0.58	0.32	0.25	1311

Figure (4.6): results table for ResNet50 uniform soup.

The accuracy of 32% is again unexpectedly low. From the confusion metric it can be seen that the model mistakenly classifies many images as being glioma. The fact that the real one is classified correctly is highly doubtable as being a mere matter of chance. There is a big indication that the soup model is ether unconverging or converges to a local minimum. This also could indicate the Individual models are converging to points that are rather far away, they are not converging to the same global minimum.

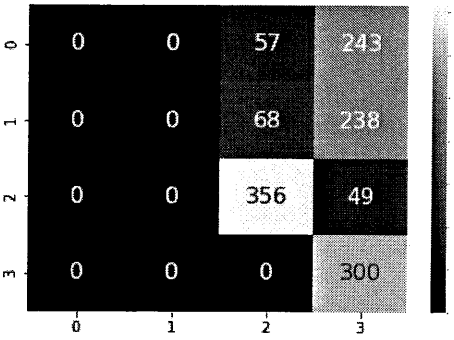
We made a second attempt by using the greedy soup algorithm to see if we can improve results. The following is obtained. Despite the accuracy being higher than before, still it is lower than the highest accuracy we obtained.



	precision	recall	f1-score	sup
0	0.06	0.05	0.05	
1	0.33	0.53	0.40	
2	1.00	0.64	0.78	
3	0.94	0.97	0.95	
accuracy			0.55	
macro avg	0.58	0.55	0.55	
weighted avg	0.61	0.55	0.57	

Figure 4.7): Confusion matrix and results table for ResNet50 greedy soup.

In another attempt to achieve better results, we eliminated all models below 90%. We got an optimal greedy soup as follows:



	precision	recall	f1-score	sup
0	0.00	0.00	0.00	
1	0.00	0.00	0.00	
2	0.74	0.88	0.80	
3	0.36	1.00	0.53	
accuracy			0.50	
macro avg	0.28	0.47	0.33	
weighted avg	0.31	0.50	0.37	

Figure 4.8): Confusion matrix and results table for ResNet50 while eliminating below 90% models.

The results of 50% is less than the accuracy of model number 8 which is the highest, and even less than 90% which is the minimum result we anticipated before setting a minimum threshold of 90%. This means applying model soup did not improve our accuracy.

Chapter Five

Discussion and Analysis on Classification of Brain Tumor

In this chapter, we will discuss the results of all three studies conducted in this thesis.

5.1 Discussion on initial studies

The initial studies consisted of four different studies the first study was on the use of benchmarking models such as VGG and ResNet without pre-training them, later we used the same models with pre-training. We also studied with custom made CNN and Swin-ViT models.

We observed that with non-pre-trained models, the smaller custom-made architectures CNN and Swin-ViT give better results than the bigger architectures. In fact, the big architectures ResNet and VGG were never able to converge. We explain this by the fact that the training sample is very small. This makes the model very likely to memorize and over fit the data. In the case of resNet101 and ResNet50, the training and validation graphs show a very clear pattern of memorization and overfitting where the training accuracy is very close to 100% while the validation is fluctuating at low accuracies and the testing scores are extremely low. Whereas in the case of VGG the model converges at a local minimum.

To improve the training profile of VGG, we anticipate that changing the optimizer might help the model to converge at a better point. Nevertheless, we still expect an overfitting behavior. Not to mention, the architecture of ResNet, with its residual connections, is far more robust in terms of gradient

flow compared to VGG. As such, we did not proceed with any further studies on VGG. In the case of a custom CNN, we observed still that the data is quite close from overfitting the model. However, the accuracy of validation and testing is much better than the benchmarking large models. The minimum that a custom CNN architecture was able to reach is 85% on testing accuracy and the maximum is 91%. In case of a Swin-ViT, the overfitting behavior is less noticeable compared to a CNN. Despite ViT demand of big training data, on our small data it has shown accuracies on testing up to 92% maximum and a minimum of 86%. We also noticed that in case of a CNN, the data is very quickly memorized, and the training accuracy hits 100% in all architectures by around 10^{th} epoch. We did not see a similar behavior in ViT. Also, we observed a more stable, less fluctuating curves in the CNN compared to ViT. Nevertheless, what encouraged us to choose the ViTs to proceed with further studies was that it had a training and validation losses and accuracies that are quite close. Also, the ViT has features that a CNN does not have which makes it more generalizable and less biased.

In addition to the non-pre-trained and custom models, we studied pre-trained models. These are ResNet101, ResNet50, and VGG16 which are pre-trained on the famous ImageNet dataset. For the VGG, again, we observed a convergence at a local minimum. Whereas for the ResNet50 and ResNet101 we achieved very good results on testing accuracy which are 95.73% and 96.72%, respectively. Also, we reached very good results with a very small number of epochs which is 9. In addition, the accuracy and loss curves of training and validation were as close as possible which indicates a good balance between bias and variance. In addition, the training seemed just as flexible as CNN training in the custom architecture we created. The summary of this study's results is shown in the table below.

Table 5.1): Summary of Non-pretrained, Custom, and Pre-trained models' accuracy results

Model	Accuracy
Non-pretrained and Custom models	
VGG16	28%
ResNet50	23%
ResNet101	23%
Custom CNN 2	85%
Custom CNN 3	89%
Custom CNN 4	91%
Custom CNN 5	90%
Custom CNN 6	89%
Swin-Vit 1	86%
Swin-Vit 2	91%
Swin-Vit 3	90%
Swin-Vit 4	92%
Swin-Vit 5	92%
Swin-Vit 6	90%
Pretrained models	
VGG16	30.89%
ResNet50	95.73%
ResNet101	**96.72%**

This study showed us that pre-trained models preform much better than custom or non-pretrained models. It also showed that the bigger the models the worse they are for our small dataset.

There are two other instances in literature in which the ResNet50 and VGG16. Ozlem and Cahfer [50] also used a ResNet50 and were able to achieve 99.02%. The first difference between our results and theirs is that we used an extended dataset which adds more samples for Glioma,

Meningioma, and Pituitary, and a fourth classification of no tumor. The fact that their dataset is smaller than ours indicates that despite lower accuracy, our model maybe more generalizable and practically more accurate that theirs. The other difference the added layers after feature extraction. We added dropout, while they did not which indicates that their model is less generalizable than ours. Also, they do not mention weather they trained their model end-to-end or not. In our case, the model was trained by freezing all layers of ResNet except the last three, and without freezing the manually added layers. Also, we trained the model for 9 epochs and a batch of 20, and they trained it for 50 epochs with a batch size of 16. These hyperparameters, in addition to the larger set we trained on indicated that this model despite the lower accuracy could be more reliable than their model.

Bilal Ahmad et al. [54], contrary to our finding, achieved an accuracy of 72.63% with ResNet50 without augmentations, and 77.52% with augmentations. Their results suggest a much lower accuracy in ResNet50 than ours which is expected since we have an extended dataset.

5.2 Discussion on Soup algorithm

In the next study, we selected the models that seemed promising from the previous study in both the pre-trained models and non-pretrained/custom models. We selected Swin-ViT-4 from the custom models and the ResNet50 model from the pre-trained models. We varied the optimizer, loss, and learning rate and created 36 different variations of each model. We observed that varying these hyperparameters greatly affects the performance of the model. There are some instances in which the model converged at a local minimum as low as 23% for the Swin-ViT, and 50% for the ResNet50. We noticed that Swin-ViT is more likely to converge to the local minima compared to the pre-trained ResNet50. Then, for both

models we applied Soup Algorithm which according to its authors is able to increase performance. However, the authors specifically stated that the method works for fine-tuned models, that is pre-trained models. Nevertheless, we tried applying them for both. The results are summarized in the table below. We tried both uniform soup model algorithm and greedy soup model which is said to give better performance. Since the results were very low in accuracy, we attempted to add a threshold to the greedy soup algorithm before applying it in which we only choose models that are above 90%. This gave the best results. We hypothesis that the reason why we got very low performance is because the individual models converge at very different points and have weights that are deviated even though the overall model gives good results.

Table 5.2): Summary of Model Soups algorithm accuracy results

Model	Accuracy
Uniform – Swin-ViT	23%
Greedy – Swin-ViT	37%
Uniform – ResNet50	32%
Greedy – ResNet50	55%
Greedy90 – ResNet50	50%

In brain cancer classification, we referred to other authors attempting to use model soups algorithm to see if they experienced the same issue. However, Roman C. Maron et al. [66] used it for the classification of skin cancer. Contradictory to our findings, they found it to increase the performance of their model. Nevertheless, they showed that ensemble still scores higher accuracies than soup algorithm. Also, all results they got were 90% or higher regardless of using ensemble or ViT.

Although it is clear that model soups performed badly in our dataset, knowing the exact cause of that requires further in-depth analysis of the selected models, dataset, and even the consistency and robustness of the algorithm itself. It is beyond the scope of this thesis to spend time on that, as such we decided to move forward and investigate other techniques. However, we associate such low performance to the dataset itself. We think that it is likely that noise in the dataset is causing such behavior. In fact, as will be seen in the next study, using large batch size drastically improves performance. This is likely due to the noise in the dataset. If the dataset is noisy, it can cause individual models to learn incorrect patterns, leading to a lack of correlation between their weights.

5.3 Discussion on CoAtNet model

In this study, we decided to combine all the desirable features of the models we studied. Specifically, we saw that pre-training gives high performance, CNN gives stable training and quick convergence, and ViT gives more generalizability. We also decide to avoid soup algorithm to improve results as it was not performing as expected, also it was taking long time and computational resources to train many models to end up with a model that is not as good as an individual highly performing model. Combining CNN and ViT motivated us to use a model CoAtNet in our study which was never attempted in previous literature work, In addition, we used a pre-trained version to get higher performance. To double check, we also attempted to train a non-pretrained model of CoAtNet, for mere sakes of comparison which proved the inferred conclusion of the first study to be correct. Although we obtained satisfactory results, we still decided to tweak the training by increasing the batch-size, augmentations, and use an exponential learning rate. All of these increased the accuracy of the CoAtNet and

showed to be important complementizes to the training of a model. The table in below summarizes results.

Table 5.3): Summary of CoAtNet accuracy results

Model	Accuracy (%)
Non-pretrained	88.00
Pre-trained	97.00
pre-trained + LB	98.30
pre-trained + LB + AUG	97.40
pre-trained + LB + AUG + DLR	99.08
pre-trained + LB + AUG + NRM+ DLR	98.86
pre-trained + LB + 2AUG + NRM + DLR	99.16

The main drawback when we compare CoAtNet to ResNet50 is the training time. One may argue that if we use same hyperparameters adjustment for ResNet50 instead of CoAtNet, we may get results that are just as good. However, our main reason behind selecting CoAtNet instead is the non-pretraining results, where ResNet50 clearly shows an overfitting behavior. Also, the pre-trained CoAtNet without any augmentations achieves an accuracy of 97% which is higher than ResNet 96.72% with augmentations.

It should be noted that the number of batches we used for this study is quite high. In our search in literature, we found from the highest performing models that the batch sizes were for Ozlem and Cahfer [50] 16, for Arshia Rehman et al. [47] was 7-10, and Sudhakar Tummala [2] was 16-32. In our case, we used a batch size of 100 which is extremely high, and it showed high performance. Also, from the mentioned literature works, we are the only work that implemented an exponentially decaying learning rate.

5.4 Comparison to previous literature work

The results of the final study were considered as our final results, as they showed the highest performance. The result 99.16% which was the highest even exceeded all literature work. Below we compare our work with previous literature work.

Table 5.4): Comparison our results with literature

Author	Method	Accuracy (%)
Cheng et al. [37]	BoW – SVM	91.28%
Ismael and Abdel-Qader [67]	DWT – Gabor – NN	91.90%
Abiwinanda et al. [68]	Two-layer CNN	84.19%
Afshar et al. [69]	CapsNet	90.89%
Pashaei et al. [70]	CNN + KELM	93.68%
Phaye et al. [71]	Diverse CapsNet	97.50%
Seetha and Selvakumar [72]	CNN	97.50%
Avşar and Salçın [73]	Region based CNN	91.66%
Zhou et al. [74]	LSTM + DenseNet	92.13%
Anaraki et al. [75]	CNN + GA	94.20%
Gumaei et al. [76]	Hybrid PCA – NGIST + RELM	94.23%
Sultan et al. [77]	Deep NN	96.13%
Deepak and Ameer [78]	CNN + SVM	98.00%
Kaplan et al. [79]	nLBP + KNN	95.56%
Francisco Javier [38]	CNN	97.30%
S. Deepak [78]	CNN + SVM + kNN	98.00%
S. Tummala et al. [2]	ViT + Ensemble	98.70%

Ozlem P. and Cahfer Gungen [50]	Fine-tuned ResNet50	99.02%
Proposed study	Fine-tuned CoAtNet-0 + LB	98.30%
	Fine-tuned CoAtNet-0 + LB + 1AUG + DLR	99.08%
	Fine-tuned CoAtNet-0 + LB + 2AUG + NRM + DLR	**99.16%**

5.5 Future work

There are many aspects that can be improved as future work on the model that is built. We will discuss them in this section.

Model Expansion

The model can be expanded to detect other types of brain tumors. This can be done by expending the dataset. Also, to make the model more advanced, it should be designed in such a way to accept multiple MRI modalities at the same time. There are certain features in different brain tumor types that requires a radiologist to look at multiple MRI modalities at the same time before being able to determine the specific brain tumor type. This is because certain features appear better in certain modalities. These kinds of extensions are not difficult to implement. The only barrier is the availability of datasets. There are very few and limited publicly available medical datasets. In this respects, future work might need to focus on creating more datasets for more types of brain tumors that are publicly available and that are also of high quality, as low-quality data can heart model performance. In addition to brain tumors, the model can be generalized to other brain lesions and diseases such as multiple-sclerosis, Alzheimer's, strokes etc.

Further than the brain, it can also be expended for other body parts such as lungs, intestine, uterus etc.

Model Accuracy Further Improvement

This work has demonstrated the effect of increasing the batch size and augmentations and learning rate variation on the performance of training which directly impacted the accuracy and performance of the model. Future work can explore other means to enhance performance. Increasing the batch size may not be easy when computational resources are limited, especially if the dataset it limited. For this reason, other techniques must also be explored.

Also, we used CoAtNet model for its strength in that it combines both CNNs and Transformers characteristics. Other models also need to be investigated. The drawback of this model compared to ResNet, for example, is speed of training. This is an open area of future work and improvement.

Generalization of Model to Other Medical Classification Tasks

To prove the generalizability and effectiveness of the model and technique we followed in this work, we must apply it to other tasks. Future work can duplicate this technique to other medical classification tasks and compare it to corresponding literature work.

Investigation of Non-pretrained Models

In this work, we have shown a clear pattern that it is difficult to train a model with such a small dataset and achieve high accuracy. The best accuracy we were able to achieve without any pretraining was 93% with the Swin-Transformer. Otherwise, this is even clear from literature, high accuracy is hardly achieved without any pre-training. The difficulty with pre-training is that we require huge computational resources and huge datasets such as

ImageNet to train a model. Typically, we use readymade pre-trained model and download the weights of the model. However, even though a lot of the pre-trained models are made publicly available, some are not. Also, we may want to design small models that are specifically for the intended task. For these reasons, future work needs to focus on designing non-pretrained models with accuracies comparable to pretrained models.

Further Investigation on Soup Algorithm

Model Soups algorithm showed unexpectedly low results in our studies. Further investigation into the reasons must be considered for future work. The soup algorithm is expected to yield results that are equal to or higher than the highest performing model. However, in our case it was even worse than the lowest performing model. This poses some questions on whether such an algorithm always works better than Ensambling on all datasets and requires further empirical investigation.

Chapter Six

A Study on Effect of U-Net Skip-connection Alteration on Glioma Segmentation

After developing a system that is able to classify brain tumors, it is required to extend this system by enabling the capacity to go a step further and segment a tumor. This chapter explores a different task that is segmentation of brain tumor. Specifically, we develop a model that segments high- and low-grade gliomas. The reason we do not include other types of tumors is because there are no datasets available for them. However, the ability to segment gliomas in itself opens the future potential to extend this model.

Prior to 2014, random forest was among the most popular techniques used for segmentation of brain tumor [7] [80]. It was successful in achieving high performance in brain tumor segmentation competitions. However, in 2015, deep learning began to take over and since then, convolutional neural networks-based models have been the top-ranking performers. U-Net, proposed by Ranneberger et al. [81] in 2015, has become a very prominent and widespread model for brain tumor segmentation and many other biomedical segmentation tasks. section.

This chapter will begin by introducing the model U-Net which we will be using for this task. Then, a section is dedicated to the survey done on this topic along with the related work to our study. Our aim is to add the minimum development to the U-Net model without lengthening its training

or inferencing timing, that is we aim to propose the most practical and efficient addition to the model.

6.1 U-Net Model

The U-Net model proposed by Ranneberger et al. [81] is a deep learning architecture specifically designed for biomedical image segmentation. It is based on the Fully Convolutional Network (FCN) model [82], but with the addition of skip-connections. The U-shaped symmetrical architecture of U-Net consists of three main parts: an encoder, a decoder, and skip-connections. The encoder is responsible for extracting features from the input image, while the decoder is used to reconstruct the segmented image. The skip-connections are used to bridge the encoder and decoder, allowing for the transfer of information between the two. In addition to the skip-connections, the authors proposed to use strong augmentation to further improve the performance of the model.

The U-Net model has been widely used in biomedical image segmentation tasks and has achieved state-of-the-art results. It has been used to segment a variety of medical images, such as MRI, CT, and X-ray images. The model has also been used in other applications, such as object detection and image classification. The success of the U-Net model is due to its ability to capture both local and global features, as well as its ability to transfer information between the encoder and decoder.

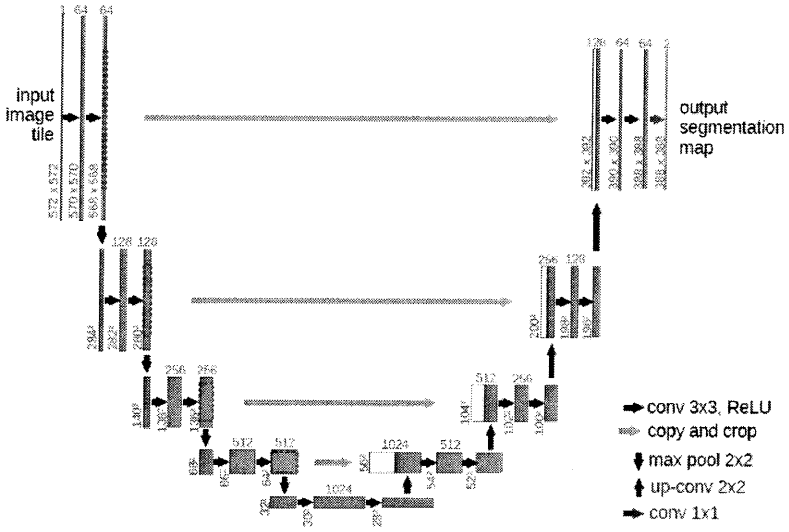


Figure (6.1): U-Net network arcitecture. Blue boxese represents multi-channel feature map.

The blocks to the left represent the encoder. The blocks to the right represent the decoder. The block in the middle is the bottlenick. The grey lines are skip-connections. (original source

(Image source: [81])

The encoder and decoder of U-Net are composed of blocks of convolutional neural networks (CNNs) stacked behind each other. The encoder blocks are referred to as down-sampling blocks, while the decoder blocks are called up-sampling blocks. A bottleneck block is placed in the middle of the two paths, along with skip connections extending from each encoder block to the corresponding decoder block. The encoder path is known as the contracting path, where the features of the image are compressed and extracted. It consists of three CNNs with 3x3 kernel size, followed by a ReLU activation function and a 2x2 max pooling with two strides. The decoder path is referred to as the expanding path. It includes two 2x2 convolutions with half the number of feature channels, a concatenation with the corresponding feature map from the encoder, two 3x3 convolutions, each followed by a ReLU activation, and a 1x1 convolution at the final

layer. This produces a mask, which is segmentation or dense prediction. The skip connections are pathways from the end of the encoder blocks to the decoder blocks, which help to smooth the training process by recovering lost information from the encoder to the decoder. U-Net is designed for 2D images but can be modified for 3D segmentation. Two famous adaptations of U-Net for 3D images are 3D U-Net [83] and V-Net [84].

6.2 Survey

Since its introduction in 2015, U-Net has become a popular architecture for medical image segmentation tasks. As a result, many researchers have sought to improve the model by modifying its architecture or training method. In this section, we will provide a survey of the different kinds of modifications that have been implemented on U-Net to segment brain tumors. We will describe modifications emphasizing the U-Net architecture and others on the training scheme. Each will be discussed in a separate subsection.

When the training method is modified, a regularization technique is used to enhance the model performance. Regularization is a technique used to reduce overfitting by adding a penalty to the loss function. Common regularization techniques include dropout, weight decay, and data augmentation.

6.2.1 Architecture Modification

When the architecture is modified, one or more of the following components can be changed: the encoder, the decoder, the bottleneck, and the skip-connection. The encoder is responsible for extracting features from the input image, while the decoder is responsible for reconstructing the segmentation mask from the extracted features. The bottleneck is a convolutional layer that reduces the number of feature maps, and the skip-

connection is a connection between the encoder and decoder that helps to preserve spatial information.

Architecture modification on U-Net creates a new variant of the model with an enhanced performance. The majority of work on brain tumor segmentation using U-Net has proposed some form of architecture modification. We organized them into two categories: core model modification and skip connection modification and discuss each one in the following subsections.

6.2.1.1 Core model modification

Core model modifications include modifications done on either one or both of the encoder and decoder of U-Net model. In addition, we also consider the bottleneck network between the decoder and encoder to be part of the core model. There are many approaches in modifying U-Net's core architecture. We will discuss three: attention mechanism, cascade, and vision transformer.

Attention mechanism. The small size of kernel in CNN limits the shape of the receptive view which cause CNN to be weak at capturing global information. Attention mechanism [85] is able to enhance CNN's global performance captioning. It was first proposed for Recurrent Neural Networks in machine translation. Zhang et al. [86]proposed AResU-Net which combines between attention mechanism and residual networks [24].

Cascaded Structure. A number of models modified U-Net by designing a cascaded structure. Jiang et al. [87] used a two-stage cascaded approach in which the first stage computes a coarse segmentation which is refined in the second stage. Their model won first place in BraTS 2019 competition. Bukhari et al [88] proposed E1D3 U-Net model by duplicating the decoder

in U-Net into three parallel decoders each one to perform the segmentation of a specific area, namely, WT, ET, and CT.

Vision Transformers. Recently, since the advent of transformers in the field of computer vision, it became a common practice to replace CNN blocks with vision transformer blocks, not merely on classification tasks, but also in segmentation. As mentioned earlier, Vision transformers were proposed by google researchers Dosovitskiy et al. [17] in a paper released in 2020. It is an adaptation of transformers used in Natural Language Processing which were proposed by Vaswani et al. [25]. More details are available in chapter 2 of the background. Generally, the advantage of transformers over CNNs is that they are able to capture long-range dependencies making them perform better than CNN in capturing global information. Vision transformers work by splitting an image into a vector of 1D patches, encodes those patches, feeds them to a block of attention followed by multi-layer perceptron, and finally outputs the classification. The main difference in behavior between a Vision Transformer and a CNN is that CNNs have inductive bias whereas transformers do not which makes them more generalizable than CNNs since they do not make an prior assumption about an image. Given the extreme differences in shape and form of brain tumors, transformer's lack of inductive bias gives transformers an advantage over CNN and makes them a promising choice for U-Net modifications.

The first adaptation of transformers for brain tumor segmentation is TransBTS model, proposed by Chen et al. [89]. It was tested on BraTS 2019 and 2020 datasets and showed scores that are higher or comparable to the previous state-of-the-art. Their model utilizes a transformer in the bottleneck of U-Net. They kept the CNNs to capture the local context

information, meanwhile the transformers captured global information. The model is shown in figure 3. Hatamizadeh et al. [90] proposed UNETR model in which they replaced the CNN encoder with vision transformer encoder. However, transformers suffer the disadvantage of computational complexity, they have a square complexity with respect to image size.

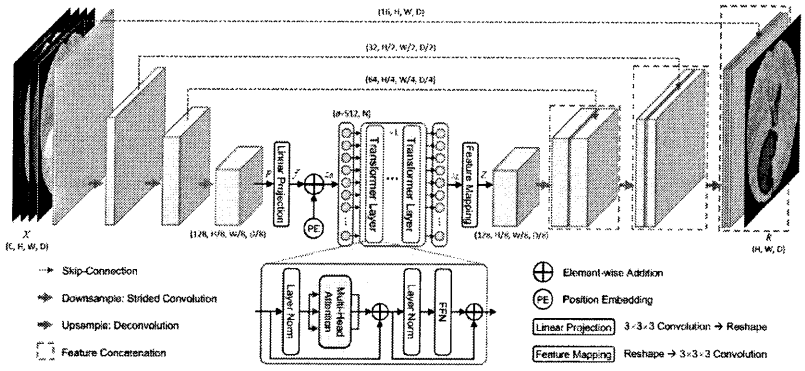


Figure 6.2): TransBTS overall architecture.

(Image source: [89])

To overcome the disadvantage of transformer complexity, Liu et al. [29] proposed the use of window-shifting technique to reduce the complexity into linear complexity and named their model Swin Transformer. Hatamizadeh et al. replaced vision transformers with Swin transformers in their previous UNETR design and named their model Swin UNETR [91]. In addition, they used residual convolutional neural networks, as proposed in [24], at the end of each down-sampling block. They achieved in validation dice score of 0.858, 0.926, and 0.885 for ET, WT, and TC, respectively. They trained their models on 800 epochs.

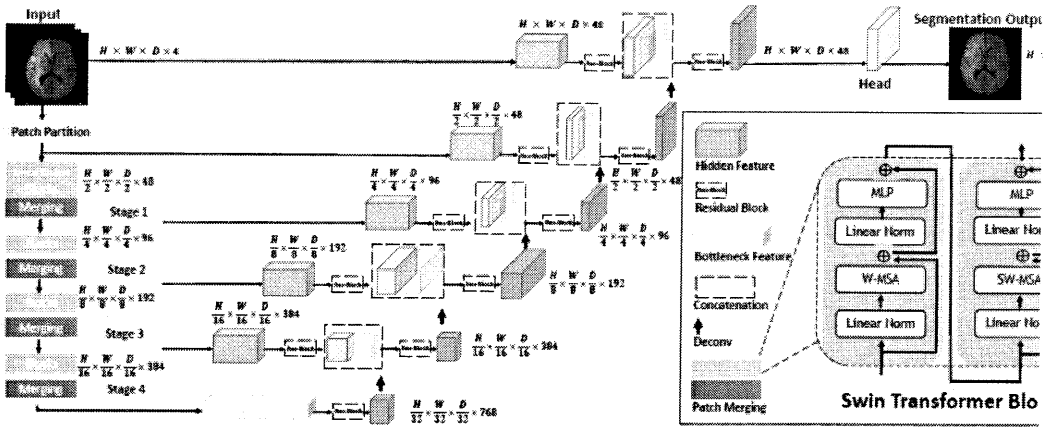


Figure 6.3): Swin UNETR architecture.

(Image source: [91])

A recent transformer-based U-Net variant proposed by Jia and Shu [92] was designed specifically for BraTS dataset. The model was named BiTr-Unet. The design uses two bottleneck transformer blocks between the encoder and decoder, as shown in figure 5. They also improved the convolutional blocks by adding attention module CBAM as proposed in [93] and it was expanded for 3D. It is an efficient and lightweight model. Given feature map F, 3D CBAM produces the channel attention map$M_c \in \mathbb{R}^{1\times H\times W\times D}$. The 3D spatial attention map is expressed as follows:

$$F' = M_c(F) \otimes F \qquad (6.1)$$

$$F'' = M_c(F) \otimes F' \qquad (6.2)$$

Where $\otimes$ is the element-wise multiplication between maps, and F'' represents the refined feature map after CBAM 3D block. BiTr-Unet model achieved in validation 0.8187, 0.9097, 0.8434 for ET, WT, and TC mean dice score with around 7050 epochs.

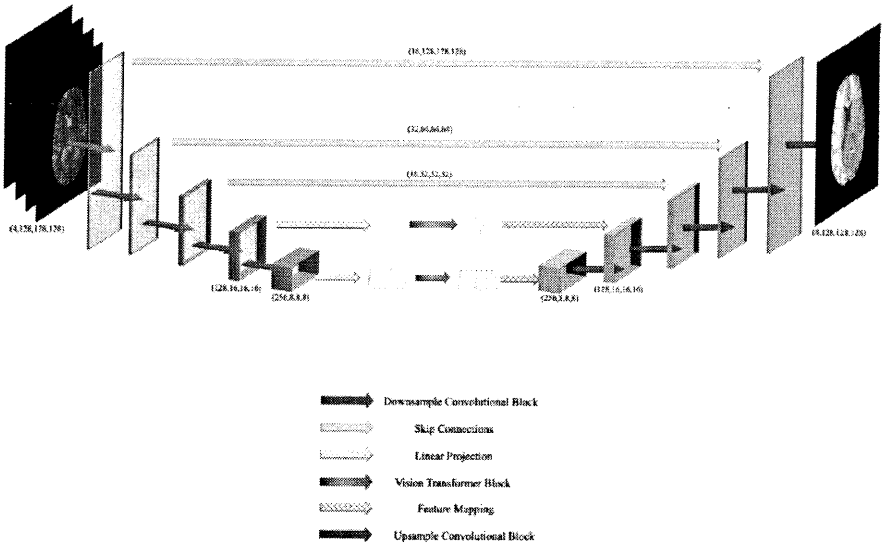


Figure (6.4): BiTr-Unet architecture.

(Image source: [92])

Despite the major success of transformers in computer vision field, they tend to require very large datasets to achieve results compatible with CNNs. This fact was also pointed out in the original paper [17]. To alleviate the need for large dataset, one needs to either apply strong augmentation or run the model for a big number of epochs.

6.2.1.2 Skip-connection Modification

Skip-connection modification entails the redesigning of skip connections to enhance the performance of U-Net. Compared to core architecture modification, there has been little work done on skip-connection modification. Zhoue et al. [94] proposed UNet++ in which they reinforce the skip connections architecture by applying deep supervisor. Their model overcomes the limitation of having to do a deep search for an optimal U-Net depth, as enhancing the skip connections promotes better training. They trained their model on BraTS 2013 dataset an obtained an average IoU score of (2±0.87). Huang et al. [95] argued that UNet++ is insufficient to obtain information from multiple scales, as such proposed an improved model U-

Net3+. The model was then further enhanced by adding residual blocks by Qin et al. [96] The model was trained on BraTS 2019 dataset. The residual connections [24] which they used are also a form of skip-connection modification that has been in fact widely used by other U-Net variants to enhance their performance (e.g., Swin UNETR [91]). Residual connections enhance the gradient flow of a model. They are simply a skip-connection from an input point to a concatenation point, as illustrated in Figure 6.5.

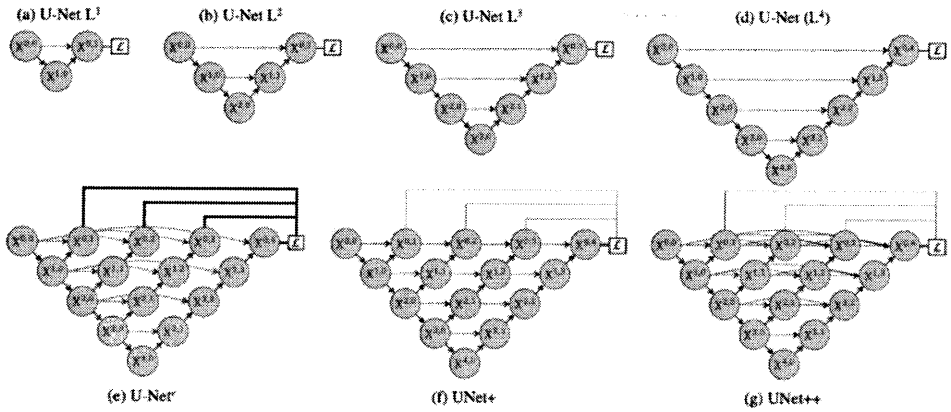


Figure 6.5): Evolution from U-Net to U-Net++.

(Image source [94])

6.2.2 Regularization

Regularization is important in enhancing a model's performance. It helps reduce over-fitting and improve the generalizability of a model. There are many different techniques in applying regularization. We will cover the main ones used in brain tumor segmentation.

6.2.2.1 Augmentation

Augmentation can be thought of as a form of regularization. As mentioned in Chapter 2, there are various augmentation techniques that one can use to enhance performance. It works by expanding the dataset and altering shapes to extend the range of a model's perception.

It has been a standard practice to apply augmentation before the training of model on brain tumor segmentation. Augmentation is the most frequently used regularization technique in many of the papers reviewed in this article [81] [83] [24] [87] [88] [89] [90] [94].

Augmentation is especially meant to enhance the invariance of a model. It enables a model to detect a tumor and segment it properly even if it changes shape or location.

6.2.2.2 Model Structure

Regularization can be achieved by model structure. This is done by using an additional block that is only used during training and removed during testing. Myronenko [97] proposed adding a variational auto-encoder to the structure of U-Net. The model is known as SegResNet. It won the BraTS 2018 challenge and achieved 0.8145, 0.9043, and 0.8596 for ET, WT, and TC, respectively.

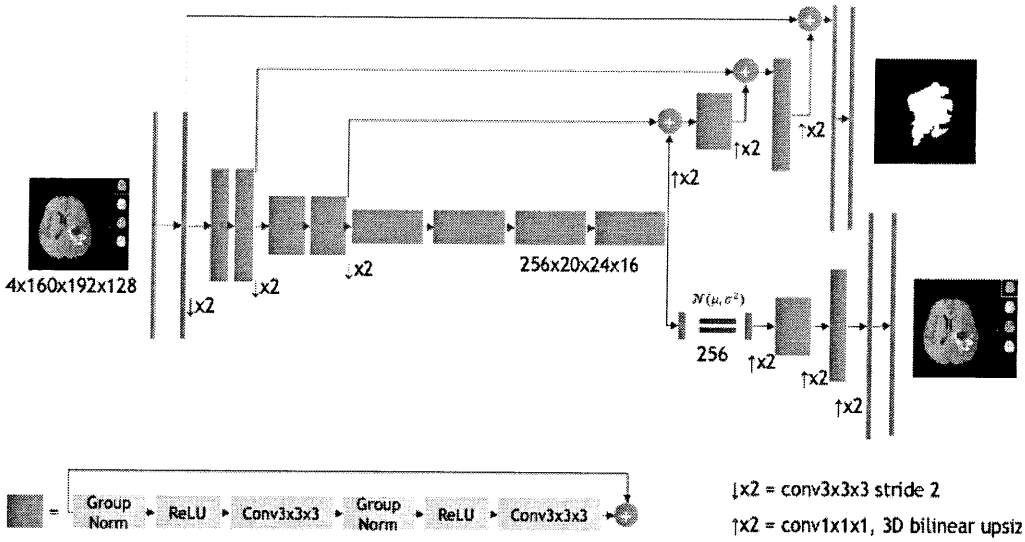


Figure 6.6): SegResNet model with the variational auto-encoder implemented during training phase to add regularization.

(Image source: [97]*)*

6.2.3 Ensemble Learning

Ensembling technique involves the combination of multiple models with strong predictions to aggregate their predictions and produce enhanced results. Kamnitsas et al. [98] proposed an ensemble model and used it for brain tumor segmentation in which they ensembled three strong models, namely, FCN [82], 3D U-Net [81], and DeepMedic [99]. Their model was the winner of BraTS 2017 challenge. Also, a common ensembling practice observed in multiple brain tumor segmentation papers is to use a single model but with different settings [100]. In fact, this kind of segmentation is seen in the majority of models' final results reviewed in this article. By combining multiple models, the variance is reduced, which reduces the amount of error that is introduced when the models are applied to new data.

6.3 Related work

In 2018, Myronenko et al. [97] implemented a U-Net which has an additional variational auto encoder branch that was only used during training to regularize the model and achieved the highest score to win the compaction. BraTS 2019's winner proposed implementing a cascaded structure of U-Net with two encoders. In 2020 and 2021 versions of the competition, the models that scored the highest applied no explicit modification to the architecture of U-Net model, rather they modified training setting, augmentation, and loss function. They only modified certain setting and were able to successfully configure a highly performing model.

In 2020, Isensee et al. [100] proposed model nnU-Net which stands for no new net. Their model is not very different from the original U-Net, except they proposed using region-based deeply supervised training, data augmentation, and only minor modifications such as the use of ReLU activation with leakiness 0.01 and used instance normalization instead of batch normalization. In 2021, Futrega et al. [101] run an extensive ablation study to test on U-Net deep supervision loss, focal loss, decoder attention. Drop block, and residual connections. They have found that the optimal model is U-Net with deep supervision which can be improved by adding an additional input channel with one-hot encoding for foreground, increasing depth and designing a post-processing strategy.

Although their ablation study encompassed many simple modifications to the U-Net architecture, we argue that U-Net itself with its default number of skip-connection layers is not necessarily the optimal configuration for the nature of BraTS dataset. The purpose of skip-connections is to recover lost information from earlier layers; thereby bridging the semantic gap between the encoder and decoder and enhancing speed of training and

convergence and even performance of the model. Wang et al. [102] showed the effect of varying the number of skip-connections built into U-Net on three different datasets: Synapse, Glas, and MoNuSeg. They observed that different layers of skip-connections have different contributions to the overall dice score, as shown in figure 1. Their results implied that full connections of all U-Net layers are not necessarily the optimal configuration for all datasets. In fact, for GlaS, the optimal performance was achieved by omitting all skip-connections. This is because not each skip connection setting is effective due to the issue of incompatible feature sets of encoder and decoder stage, even some skip connection may result in negatively affecting the performance. Inspired by their study, we will repeat their study of U-Net skip-connection analysis on BraTS dataset. The optimal configuration that we find in this study should serve as a starting point base model to future researchers interested in implementing U-Net architecture.

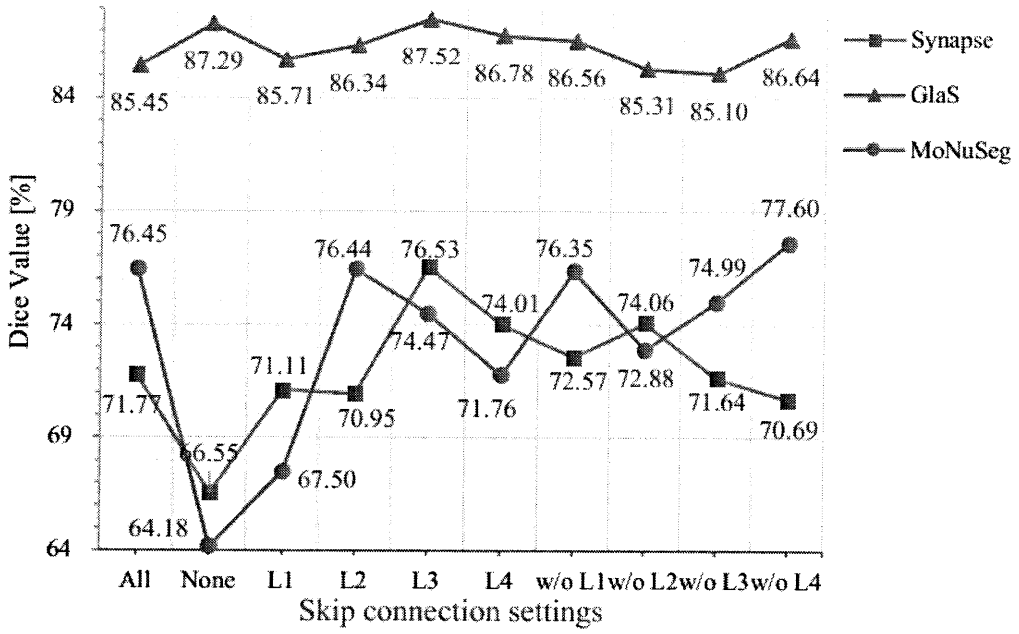


Figure 6.7): ablation study to investigate skip-connection effect. Plot shows the dice score vs. model configuration.

(Image source: [102])

6.4 Method

The network architecture we used is as proposed on the original paper of U-Net [81]. The model is made of an extracting and a contracting path. Along with skip-connections in between, as shown in figure 2. The contracting path is called an encoder which consists of four blocks, each with two convolutional layers with kernel size 3 followed by ReLU activation and a max-pooling layer at the end with strides 2 and kernel size 2. After each block, the number of feature channels is doubled. Therefore, the assuming a feature of n, the input channel size of each block is n, $2n$, $4n$, and $8n$, respectively. The expensive path is a decoder which is consistent of four blocks that are arranged in a symmetrical manner with the encoder blocks. Each block consists of an up-sampling with kernel size of 2 and two convolutional layers of kernel size 3 followed by ReLU activation. A cropping is applied which is necessary due to the loss of border pixels in

every convolution. At the final layer, a convolution of size 1. The network was originally designed for 2D images; thus, we modified it into 3D network as done in [83].

The advantage of our proposed modifications is that they do not involve any major architecture alteration, nor any increase in training or inferencing time. There has been a lot of previous work which focused on the modification of U-Net without effecting its architecture majorly or inferencing time [100] [103].

6.4.1 Skip-connection modifications

In addition to the original architecture, we designed another ten architectures all of which have only the skip connections being modified. In the first variation, we remove all skip connection layers which results in a network identical to FCN model as proposed in [82]. We designed four other networks that keep only a single layer of skip-connection the first, second, third, and fourth which we denote as L1, L2, L3, and L4, respectively. We also designed four other variants to include all skip-connection layers, except one layer. In each variant we removed a different skip-connection layer and denoted the models as no-L1, no-L2, no-L3, and no L4.

6.5 Setup

Our study is implemented using PyTorch and MONAI and trained using Google Colab Pro+ which uses a V100 NIVIDIA GPU limited by a 500 number of compute units which if a user exceeds may experience a drop in performance causing a limitation in our computational resources, thereby we limited the number of epochs to 100 and frequency of validation to 10.

The input has a 48-feature size with image re-sized to 128x128x128 pixels. The optimizer used is ADAM optimizer with learning weight of 0.0001 and

weight decay of 0.00001. The training inputs to the model were normalized to speed up training. The images were resampled to a pixel-dimension (1,1,1) with interpolation mode being bilinear and nearest. For augmentation, random cropping, random flipping with probability 0.5, random scaling intensity of factor 0.1 and probability 1, and random intensity shifting of off-set 0.1 and probability 1 were applied to the training dataset. The dataset used is taken from Decathlon library [6]. A seed of 0 for deterministic output was set by MONAI built-in determinism function to make the results of the study reproducible and consistent.

6.6 Results

All models were trained on the same dataset with the same training settings. The aim of this study is examining the effect of skip-connection and each layer's contribution on BraTS dataset. The results are shown in Table 1. Table shows validation results. The first column indicates the model being trained. To illustrate what each model label indicates, "U-Net" is the original model with full connection, "None" is no skip-connections, "L1 only" is only first skip-connection is included while all others are not, and "No L1" is all skip-connection layers are included expect L1. Same terminology convention applies to L2, L3, and L4. In the table, TC indicates tumor core, WT whole tumor, ET enhancing tumor, and final column is the average dice score.

6.6.1 Visualizations

For each model we visualized five parameters: average loss, mean dice, mean TC dice, mean WT, and mean ET dice against epochs. In this section, we will show all visualizations.

Table (6.1): U-Net variations dice metric and loss visualizations

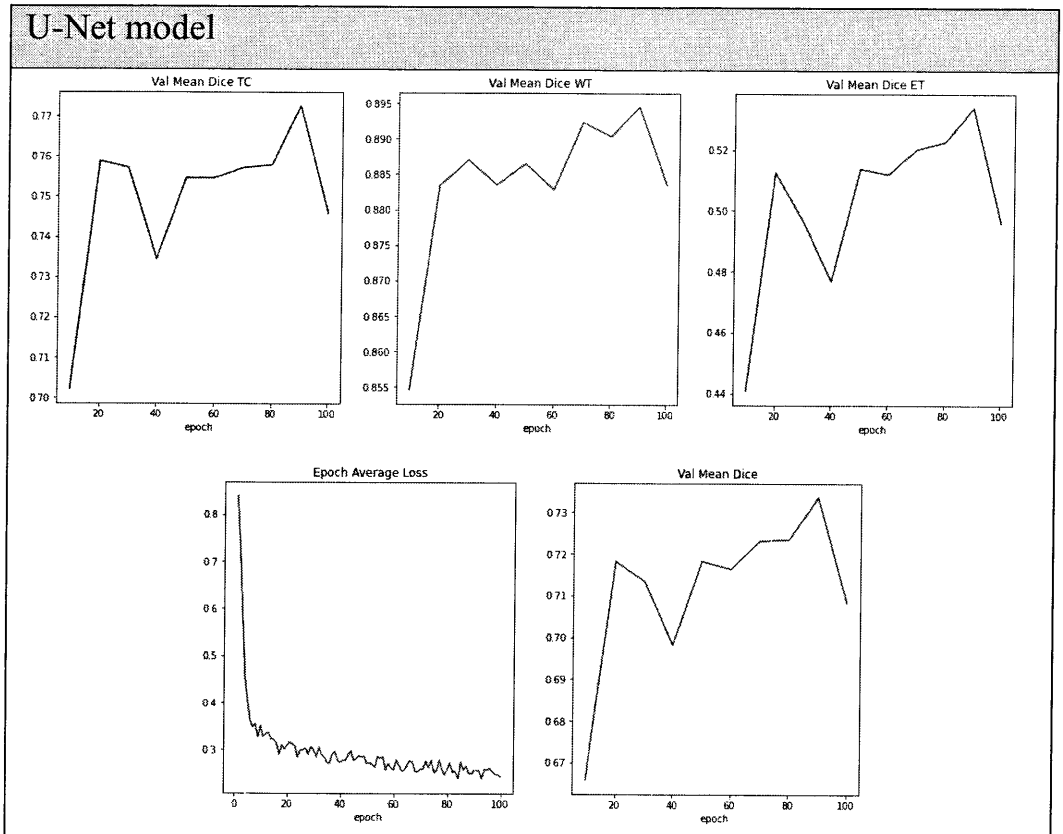
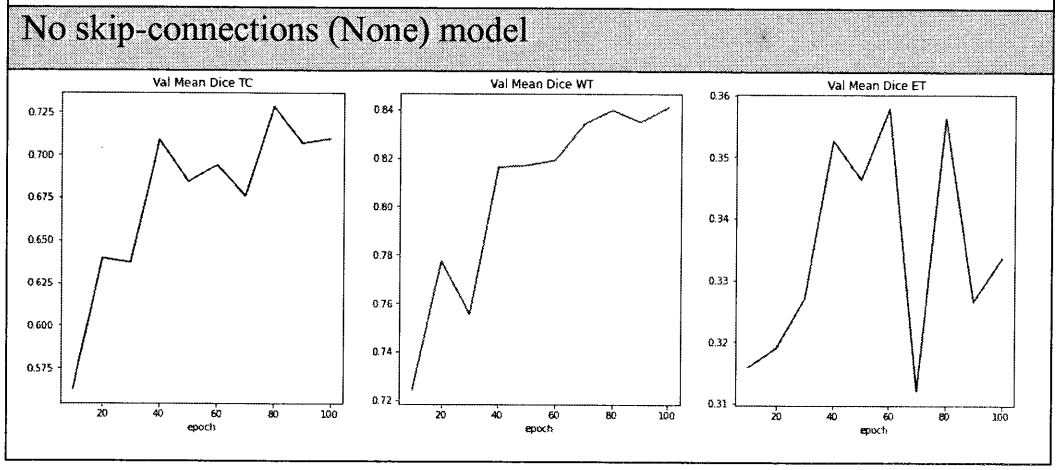


Figure 6.8): U-Net training profile.



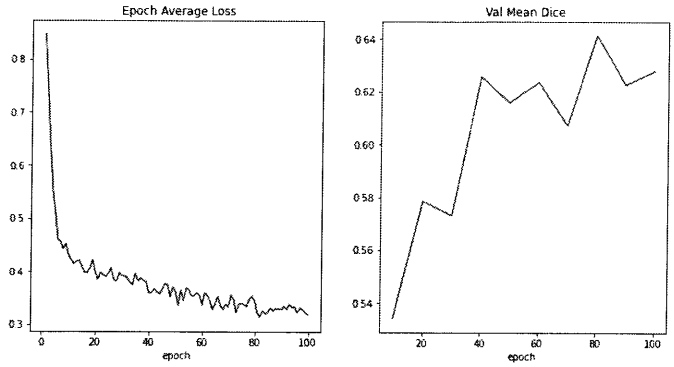


Figure (6.9): No-skips U-Net training profile.

L1-only model

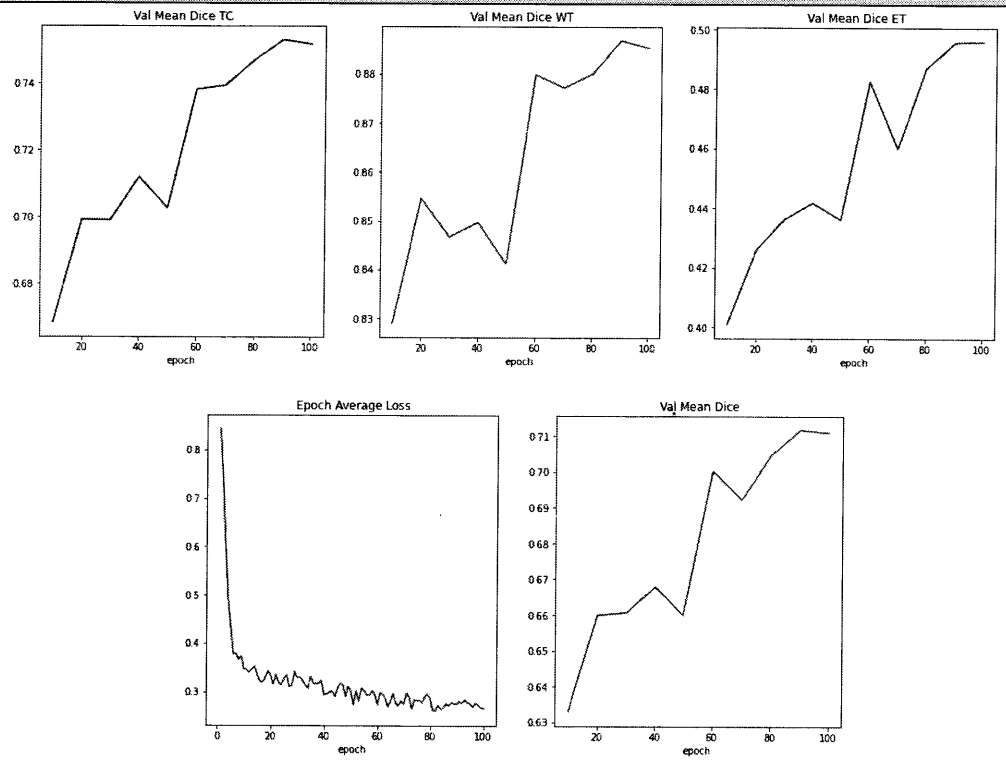


Figure (6.10): L1-only training profile.

L2-only model

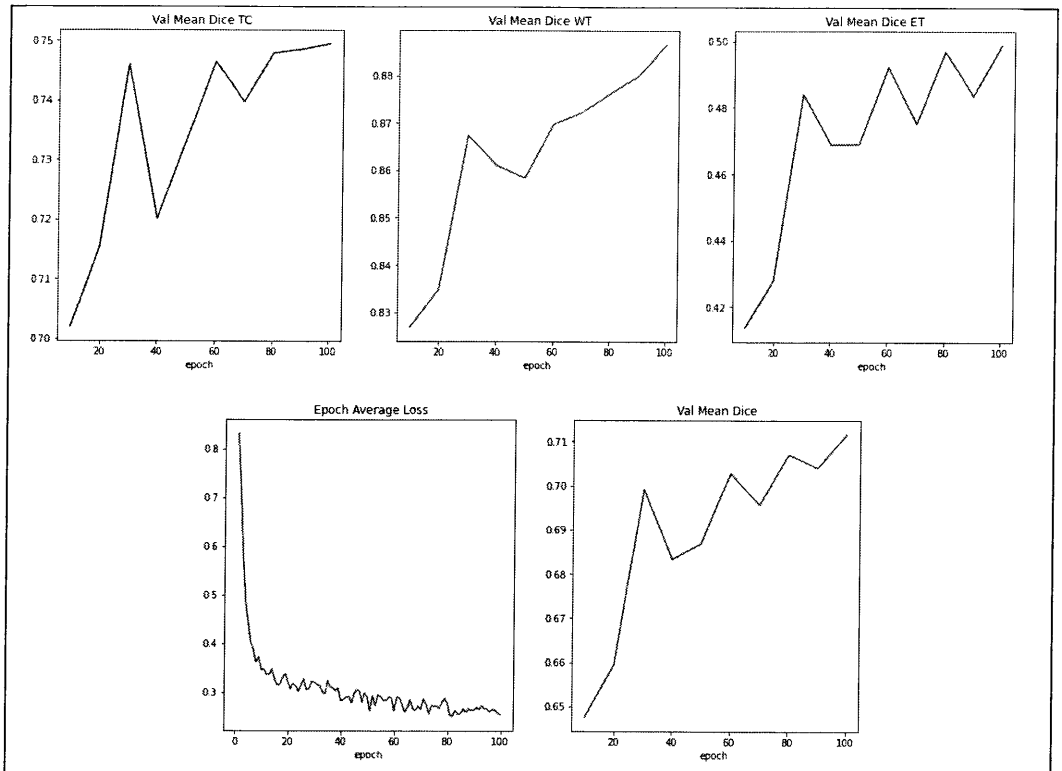
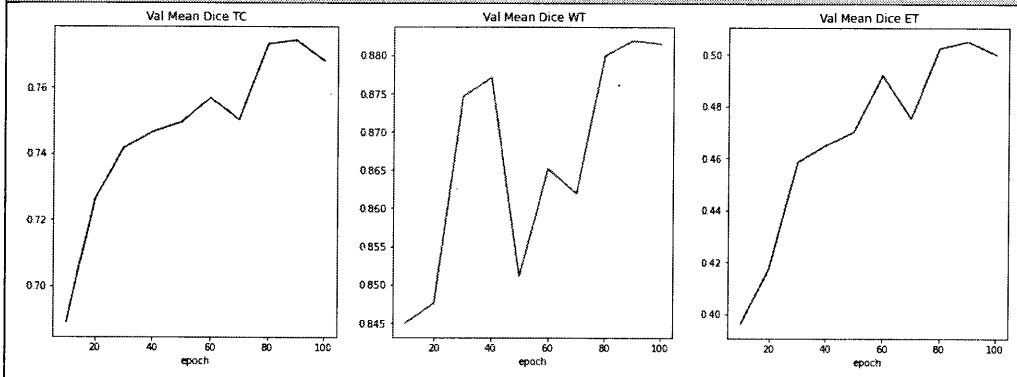


Figure (6.11): L2-only training profile.

L3-only Model



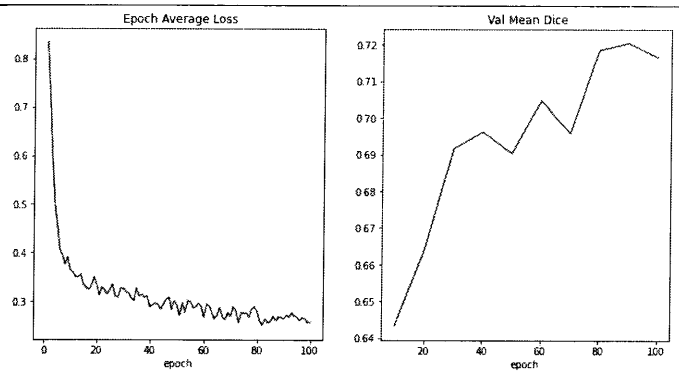


Figure 6.12): L3-only training profile.

L4-only Model

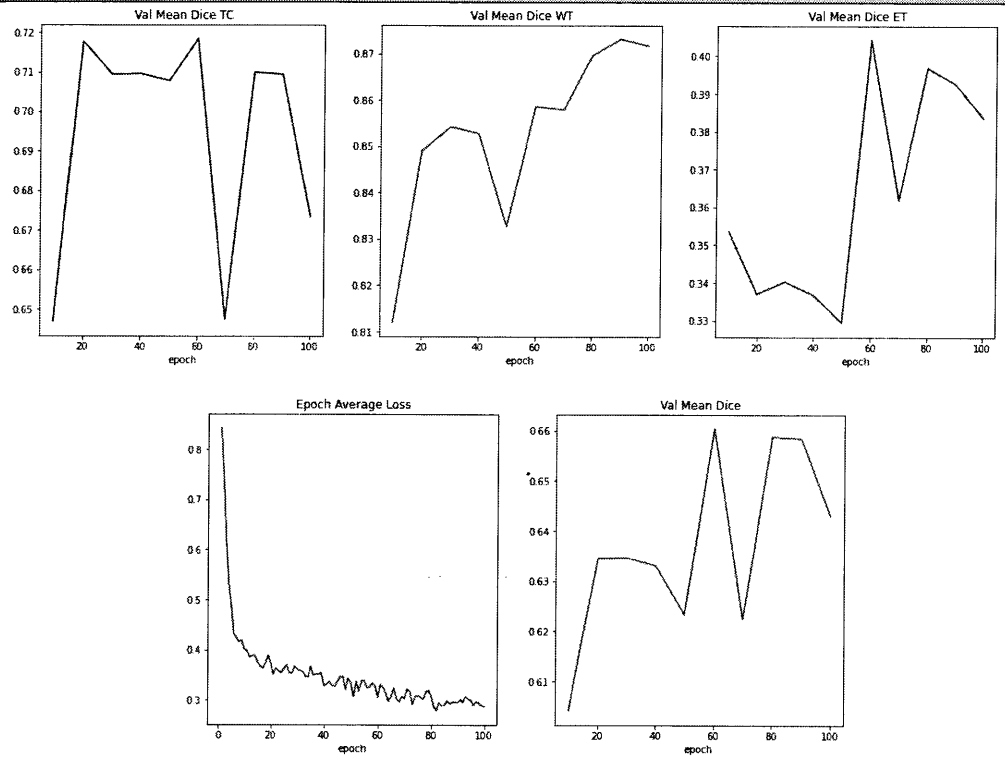


Figure (6.13): L4-only training profile.

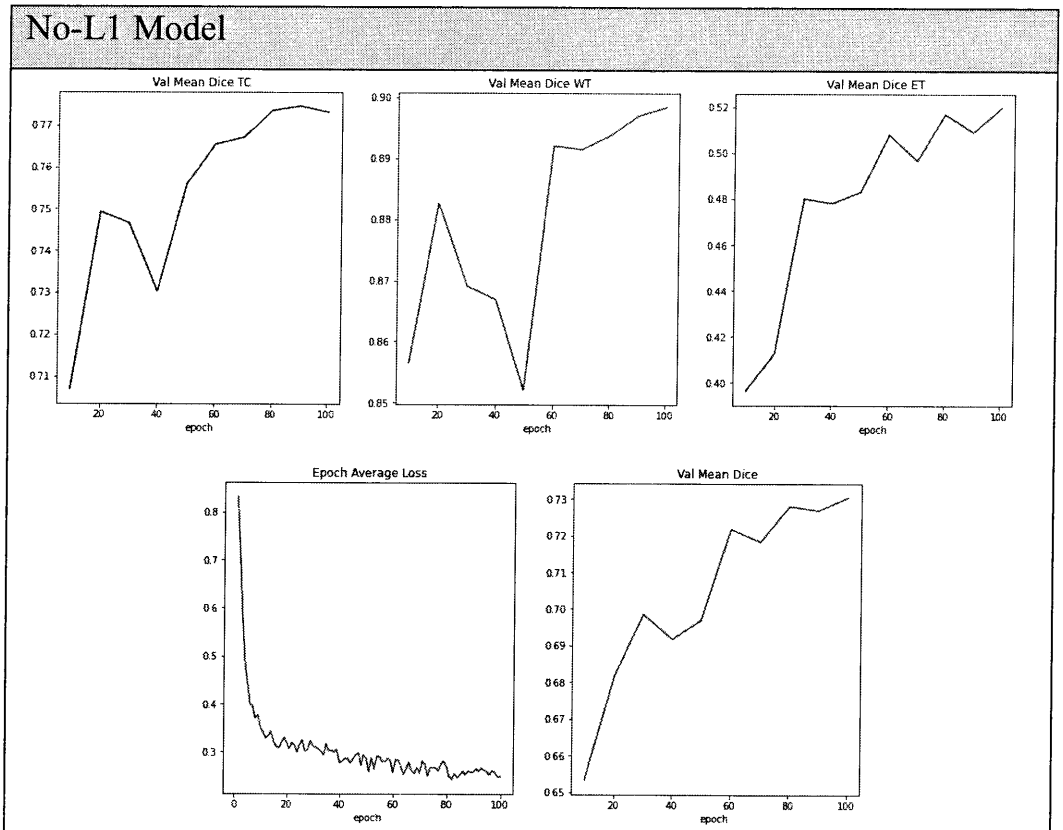
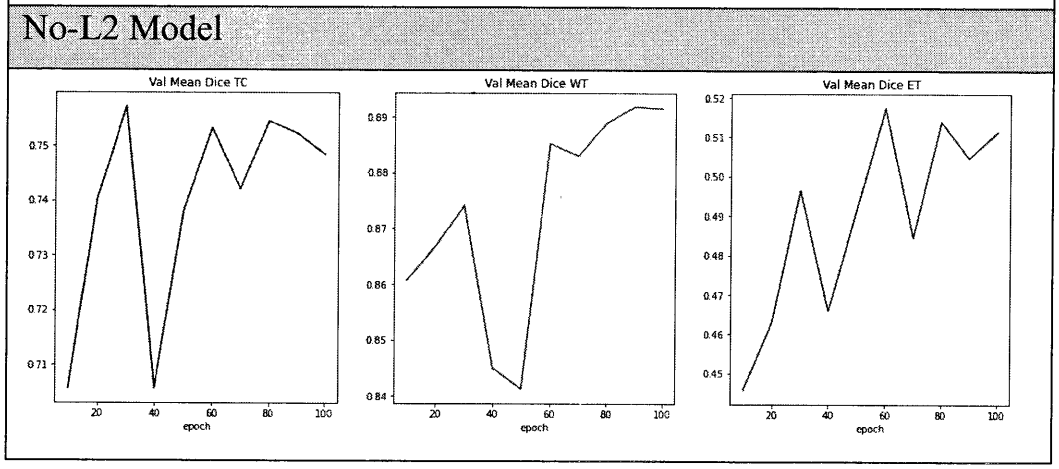


Figure (6.14): No-L1 training profile.



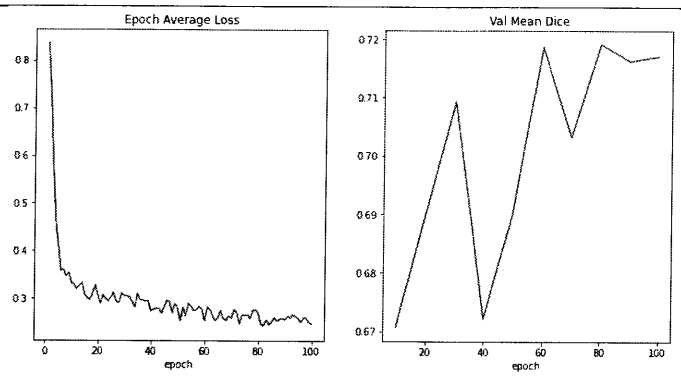


Figure (6.15): No-L2 training profile.

No-L3 Model

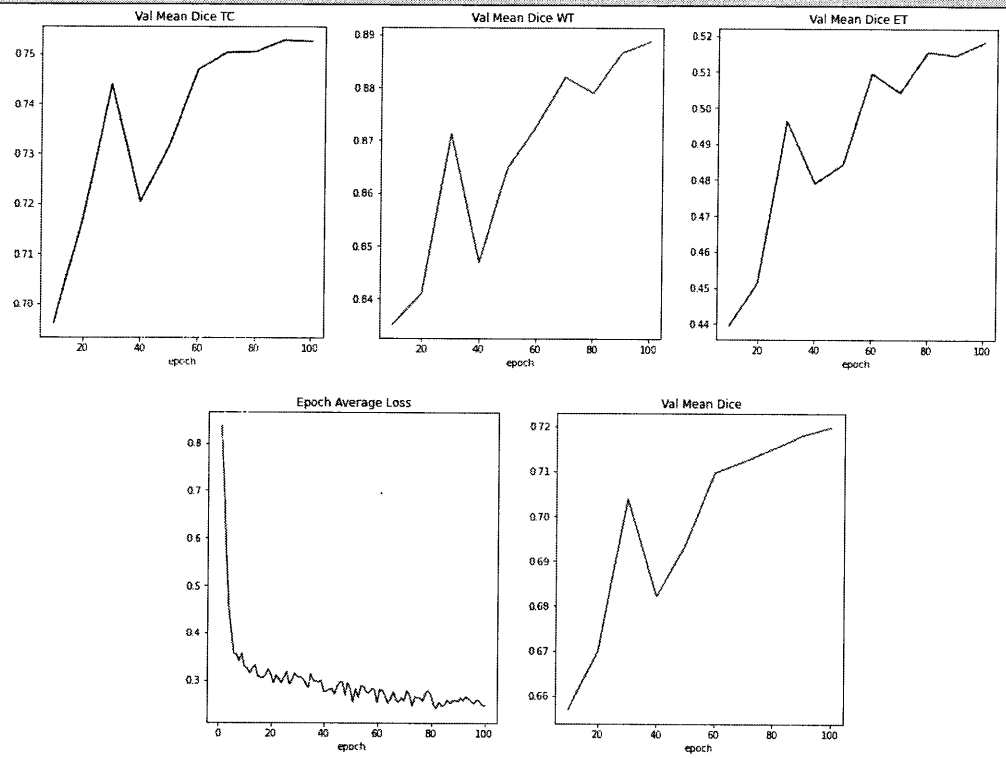
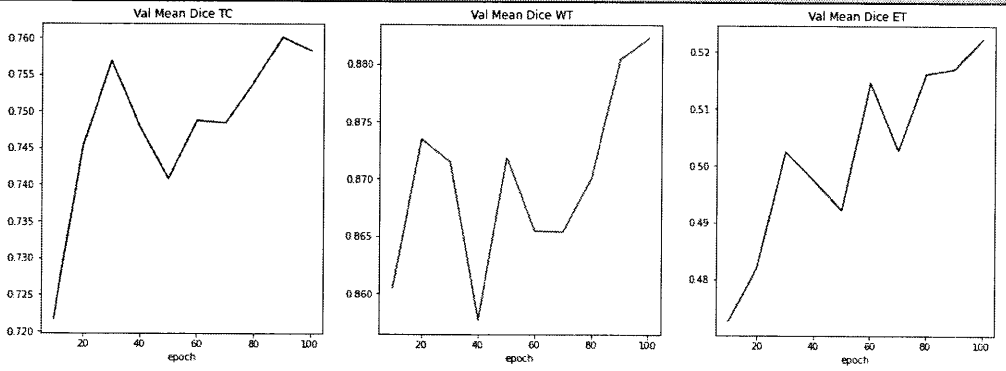


Figure 6.16): No-L3 training profile.

No-L4 Model



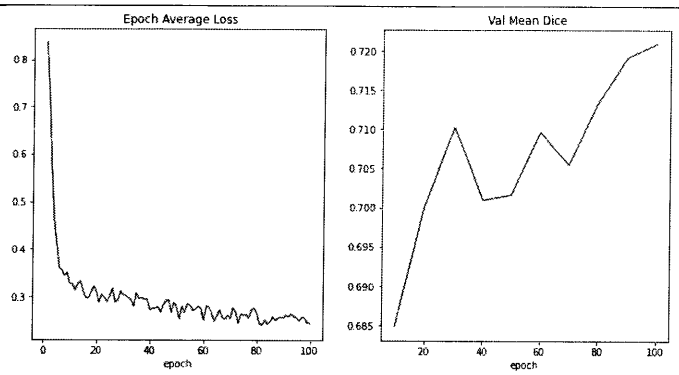


Figure (6.17): No-L4 training profile.

A random MRI sample is selected from the dataset, the four modalities' labels are shown with the predicted segmentation of each model for sakes of comparison, as in table below.

Table 6.2): Samples of segmentation

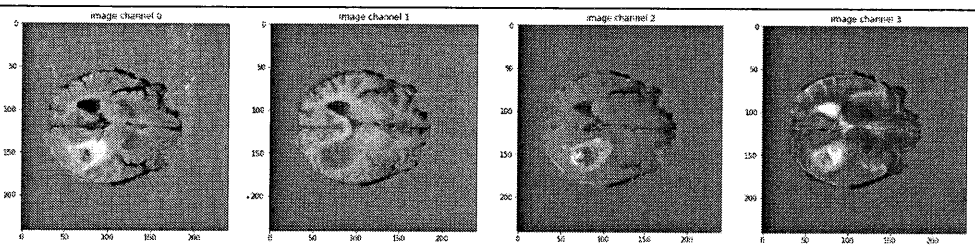


Figure (6.18): Brain MRI sample.

Ground truth

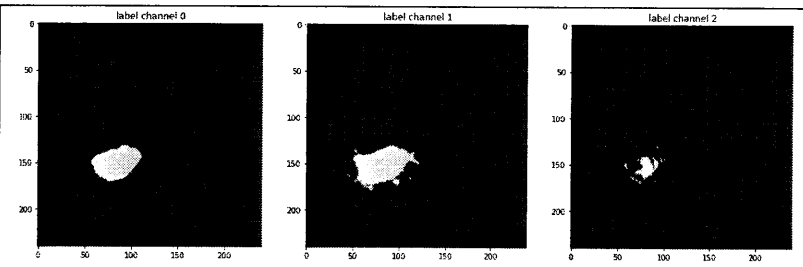


Figure (6.19): ground truth segmentation.

U-Net

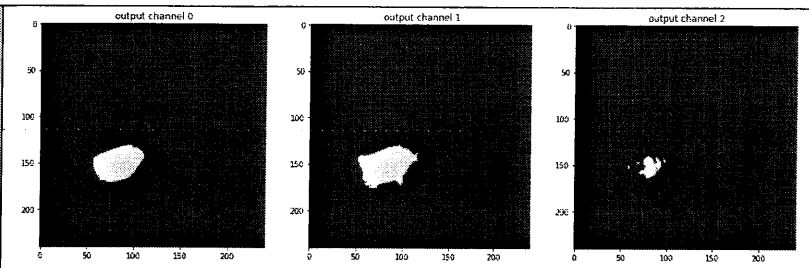


Figure 6.20) U-Net segmentation.

None

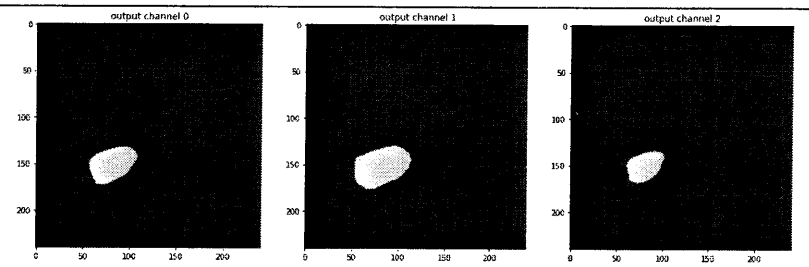


Figure (6.21): None Segmentation.

L1-only

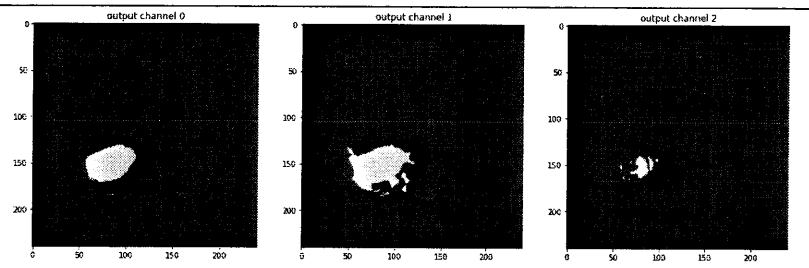


Figure 6.22): L1-only segmentation.

L2-only

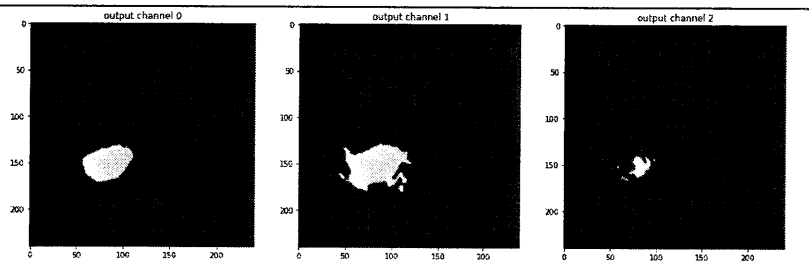


Figure 6.23): L2-only segmentation.

L3-only

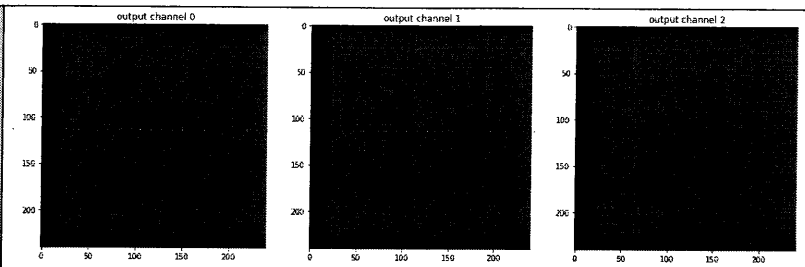


Figure (6.24): L3-only segmentation.

L4-only

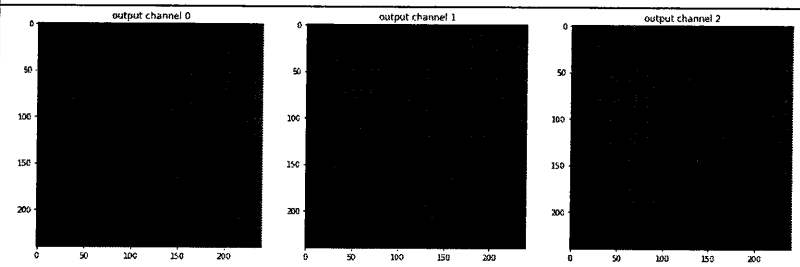


Figure (6.25): L4-only segmentation.

No-L1

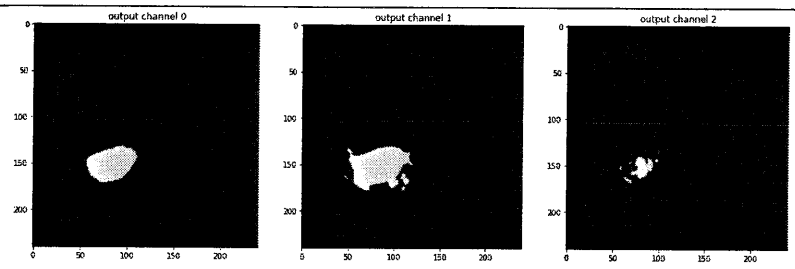


Figure (6.26): No-L1 segmentation.

No-L2

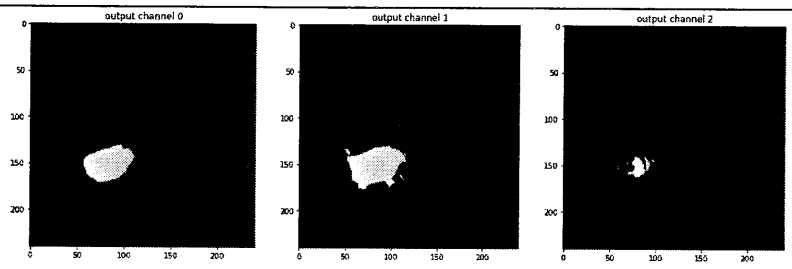


Figure (6.27): No-L2 Segmentation.

No-L3

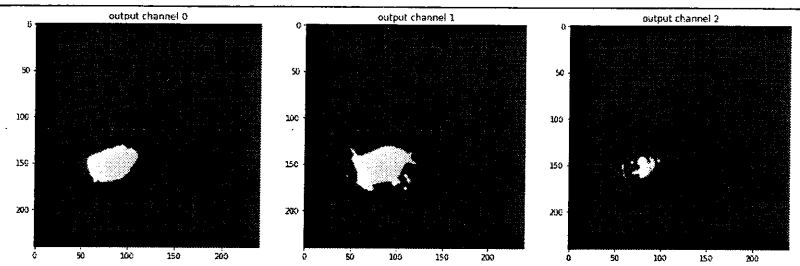


Figure (6.28): No-L3 Segmentation.

No-L4

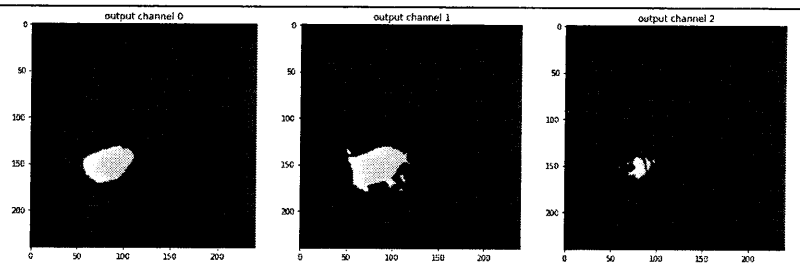


Figure (6.29): No-L4 segmentation.

6.6.2 Summary

The table below summarizes all results in terms of final dice score result of TC, WT, ET, and average. The results of average dice in Table 6.3 are plotted as a graph in Figure 6.30.

Table 6.3): Summary and comparison of dice scores

Modification	Dice score			
	TC	WT	ET	Average
U-Net (original) [81]	0.7458	0.8836	0.4959	0.7084
None	0.7093	0.8409	0.3336	0.6279
L1 only	0.7518	0.8853	0.4960	0.7110
L2 only	0.7495	0.8867	0.4990	0.7117
L3 only	0.7681	0.8815	0.5001	0.7166

Modification	Dice score			
	TC	WT	ET	Average
L4 only	0.6735	0.8717	0.3836	0.6429
No L1	**0.7732**	**0.8985**	0.5196	**0.7304**
No L2	0.7484	0.8915	0.5113	0.7171
No L3	0.7525	0.8888	0.5182	0.7198
No L4	0.7583	0.8823	**0.5222**	0.7209

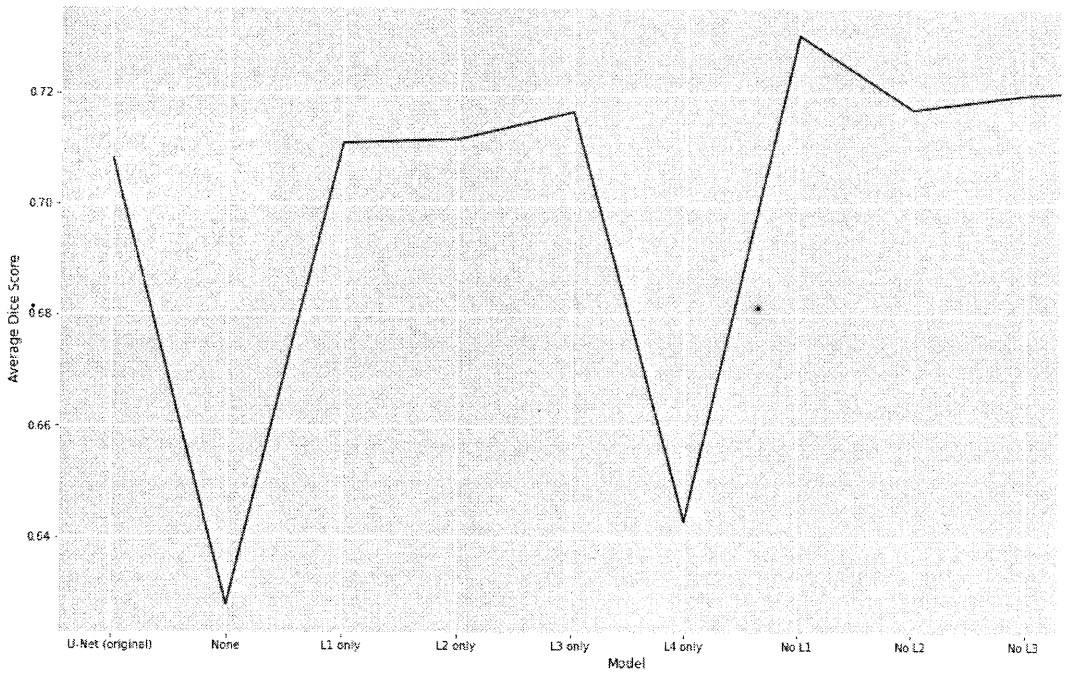


Figure 6.30): Average Dice Score vs U-Net model skip-connection modification.

6.6.3 Discussion

It is clear from Table 1 that the original U-Net is not an optimal design for BraTS dataset in terms of skip-connection. Generally, the tumor of least dice score is the enhancing tumor ET. And the tumor that is easiest to detect is the whole tumor WT which is normally the case in all models trained on

BraTS due to the imbalance in the dataset which is expected in medical imaging dataset. Though it can be noticed that for model "no L4" the ET is at its highest score indicating that the fourth layer of skip-connection may have a negative impact on this mask type.

Meanwhile, with "no L1" model we notice the highest performance in both TC and WT. With the observation of such a variation between layers alteration on TC, WT, and ET, it can be inferred that to achieve optimal performance on all masks, one needs to build rather a cascaded network than a single one to classify each different network. For example, an optimal configuration may be using two decoder branches one for TC and WT segmentation, and the other branch for ET segmentation, where the first branch feeds from all layers except the first, and the second branch feeds from all layers except the fourth. In fact, cascading U-Net structure has been proposed several times in previous works, such as [88] and [87], but in none of these works did the authors attribute the problem to the skip connection, rather models target the parameters of the CNNs which will be updated differently with training. However, the use of a cascaded U-Net may compromise computational efficiency and speed of training. As such, to balance between high performance and computational cost, U-Net with "No L1" itself offers the highest average score. In addition, it can be used as a base model rather than U-Net itself with the advantage that there is no computational difference between the two.

It is important to mention though that our study faced some limitations. We had a very scarce computational resources upon conducting this study. Therefore, study was done on quite a small number of epochs which is 100. A larger number of epochs could give more accurate results of each model's performance. After longer epochs, we may or may not observe performances reflecting table 1 as the results may tend to stabilize further

and give different outcome. Also, the validation frequency was such that validation is applied every 10 epochs. An ideal frequency of validation is 1. However, again due to computational limitation we had to set it to 10.

It would be interesting to see if the same performance with the same modifications applies to BraTS datasets from years other than 2016 and 2017. It is not expected that a major difference would result as all BraTS datasets have the same nature of imbalance. Nevertheless, to confirm this hypothesis, the same ablation study must be repeated on other BraTS as well.

Due to the computational resource limitation of this study, we leave to future studies to investigate the implementation of a UCTransNet as proposed in [102] instead of the skip-connections and prove whether it is more effective than skip-connections. Though, it must be noted that transformers suffer the expensive computation, even if they perform better than CNNs and are able to capture long-range dependencies.

Also, we propose for future work to implement this analysis on previous U-Net variants that were used on BraTS datasets to confirm the consistency of the study results on other modification of U-Net. Finally, we leave for future work the implementation of the same ablation studies on datasets of BraTS from years other than 2016 and 2017 which were used in this study to confirm whether the same thing applies to other versions of the dataset, as datasets tend to have different signals resulting in different responses from a model.

Chapter Seven

Conclusion

7.1 Summary and Conclusion

In this study, our objective was to develop a system that can aid clinicians in diagnosing brain tumors by classifying tumors and segmenting them. We utilized different datasets for each task, employing a dataset consisting of MRI images of three tumor types, namely glioma, meningioma, and pituitary, for the classification task, and the famous BraTS dataset for the segmentation task.

For the classification task, we conducted three different studies. In the initial study, we compared the performance of pre-trained and non-pretrained models and found that pre-trained models provided better results. Furthermore, we created custom CNN and ViT models that showed promising performance, particularly on smaller datasets. The highest accuracy that we achieved in this study was 96.27% using a pre-trained ResNet101 model.

In the second study, we aimed to increase the performance of select models from the previous study using a recent algorithm called Model Soups algorithm. However, we observed that applying this algorithm caused the performance to either drop or collapse. Consequently, we employed a different approach in the third study, which involved combining pre-training, CNN, and ViTs in one model. For this purpose, we selected the pre-trained CoAtNet-0 which intrinsically preserves the desirable properties of CNN and ViT and is efficient on small datasets. The model

showed an initial accuracy of 97%. We then proceeded by conducting a sequence of studies on the model to enhance its accuracy by utilizing different techniques based on literature. We showed that applying augmentation, normalization, an increase in batch size, and exponential decaying learning rate can drastically enhance the performance. The model achieved maximum testing accuracy of 99.16%, outperforming the state-of-the-art results.

For the segmentation task, we employed the U-Net architecture and studied with different skip-connection combinations to optimize the architecture for glioblastoma segmentation. We proposed the elimination of the first skip-connection, which resulted in the highest average dice score of 0.7304 compared to the original U-Net score of 0.7084 and other skip-connection variations.

Overall, our study showed that combining pre-training, CNN, and ViTs in one model, as well as optimizing the skip-connection configuration for segmentation, can significantly enhance the accuracy of brain tumor diagnosis. The proposed system has the potential to assist clinicians in making more accurate and efficient diagnoses, leading to better patient outcomes.

7.2 Direction for Future Research

In this section future directions for each task will be presented.

7.2.1 Further Studies

In the first task, it is clear that the model has shown high performance. In future direction, the performance might even be further improved by adjusting hyper parameters of the same model or using a different model. One scenario which is worth future investigation and has not been studied in this thesis is examining the same setting on ResNet-50. Specifically, the

settings which we used for CoAtNet and enabled us to achieve 99.16%. Those are the two augmentations, normalization, exponentially decaying learning rate, and large batch-size. The objective of this future study to observe whether ResNet-50 has better performance compared to CoAtNet. However, even if ResNet-50 appeared to show higher performance, the result which showed ResNet-50 to be convergent compared to CoAtNet when no pre-training is used indicate that CoAtNet could have the possibility the better generalize on unseen datasets that could have a different diversity and distribution compared to the dataset we used for training.

For the second task, the conducted study can be further validated by comparing the proposed elimination of first skip-connection on other variants of U-Net. As is known and discussed in the literature review section, the U-Net has massively evolved over time. Nevertheless, the evolution and variants of U-Net were all based on its original baseline. In this future study, we suggest comparing the most prominent U-Net variants while using the original baseline and while using the proposed configuration of this thesis. Such an study might be computationally very expensive, nevertheless, it will be close to a solid proof the proposed configuration.

7.2.2 Semi-supervised Learning

Another future direction which we propose for task 1 is the use of semi-supervised learning. Currently, semi-supervised learning is becoming a rather hot topic in the field of computer vision, especially medical imaging. The reason why is that in semi-supervised learning the challenge that is being targeted is the lack of enough labeled datasets. In medical imaging, not only obtaining data is difficult but also labeling it. If some researcher for example managed to collect 1 million patient samples of MRI, and

supposing that each sample takes a second to be labeled, it will take possibly weeks to label this dataset. In semi-supervised learning, clustering techniques are used to group data with similar distribution which makes it possible to train a model to correctly classify them even if they are not labelled. However, it is important that part of the dataset, even if only 10% of the dataset, is labelled. Also, semi-supervised learning has been rarely investigated in brain tumor classification especially on the dataset that we used on Cheng. Recently, more research in computer vision is focused towards employing semi-supervised learning techniques. Mainly, the idea is adopted from the major success of semi-supervised learning in NLP.

7.2.3 Dataset collection

The biggest challenge in medical imaging is finding datasets. Even the available ones tend to be very small relative to what is needed for the practical success of medical imaging with dataset hungry deep learning algorithms. Data collection can be a major contribution to the field. However, it requires to be done by a large group of researchers and requires massive effort and funding for its success.

7.3 Answer to Research Questions

In this section, the research questions previously proposed will be answered.

1. **Which type of model/method yields the highest performance in brain tumor classification: pre-trained, non-pre-trained?**
   Highest performance for brain tumor classification comes with pre-training.

2. **Which models perform best, custom models or benchmark models?**

Fine-tuned benchmark models have higher performance compared to custom created. However, if transfer learning is not used, then custom models are better.

3. **To what extent can the performance be improved by combining transformers, convolutional neural networks, and pre-training in one model for classifying brain tumors?**
   We have used CoAtNet which has shown that the model combines the advantages of both models and is able to train even on small datasets without the need for transfer learning to converge. With pre-training, the model is even able to perform much better.

4. **Which type of model/method yields the highest performance in brain tumor classification: pre-trained, non-pre-trained?**
   Highest performance for brain tumor classification comes with pre-training

5. **Which models perform best, custom models or benchmark models?**
   Fine-tuned benchmark models have higher performance compared to custom created. However, if transfer learning is not used, then custom models are better.

6. **To what extent can the performance be improved by combining transformers, convolutional neural networks, and pre-training in one model for classifying brain tumors?**
   We have used CoAtNet which has shown that the model combines the advantages of both models and is able to train even on small

datasets without the need for transfer learning to converge. With pre-training, the model is even able to perform much better.

7.4 Conclusion

In conclusion, this thesis aimed at designing a system to diagnose brain tumors. The thesis consisted of two tasks. The first task involved the classification of three different types of brain tumors: Glioma, Meningioma, Pituitary tumors. The classification model of task 1 was Coat Net, it was successfully in classifying the brain tumors up to an accuracy of 99.16% with the aid of augmentations, normalization, learning rate decay, and expansion of batch size. In the second task, the U-Net model was used and improved by eliminating the first skip-connection. The proposed improvement has shown itself to perform better than the original U-Net configuration with an average dice score of 0.7304 compared to 0.7084. For Enhanced tumor segmentation, we specifically found that eliminating the 4^{th} layer while keeping the rest achieved the highest score.

References

[1] J. Cheng, "brain tumor dataset," figshare, 22 4 2017. [Online]. Available: https://figshare.com/articles/dataset/brain_tumor_dataset/1512427.

[2] Sudhakar Tummala 1,* , Seifedine Kadry, Syed Ahmad Chan Bukhari, Hafiz Tayyab Rauf, "Classification of Brain Tumor from Magnetic Resonance Imaging Using Vision Transformers Ensembling," *Current Oncology,* vol. 29, pp. 7498-7511, 2022.

[3] Sartaj, "Brain Tumor MRI Dataset," Kaggle, 2021. [Online]. Available: https://www.kaggle.com/datasets/masoudnickparvar/brain-tumor-mri-dataset.

[4] Sartaj, "Brain Tumor Classificatio MRI," Kaggle, 2020. [Online]. Available: https://www.kaggle.com/datasets/sartajbhuvaji/brain-tumor-classification-mri.

[5] "Br35H :: Brain Tumor Detection 2020," Kaggle, 2020. [Online]. Available: https://www.kaggle.com/datasets/ahmedhamada0/brain-tumor-detection?select=no.

[6] Antonelli, Michela, Annika Reinke, Spyridon Bakas, Keyvan Farahani, Annette Kopp-Schneider, Bennett A. Landman, Geert Litjens et al. , "The medical segmentation decathlon," *Nature communications,* vol. 13, no. 1, pp. 1-13, 2022.

[7] Bjoern H. Menze*, Andras Jakab, Stefan Bauer, Jayashree Kalpathy-Cramer, Keyvan Farahani, Justin Kirby et al., "The Multimodal Brain Tumor Image segemntation benchmark BRATS," *IEEE transection on medical imaging,* vol. 34, no. 10, 2015.

[8] S. Bakas, H. Akbari, A. Sotiras, M. Bilello, M. Rozycki, J.S. Kirby, et al., "Advancing The Cancer Genome Atlas glioma MRI collections with expert segmentation labels and radiomic features," *Nature Scientific Data,* vol. 4, 2017.

[9] S. Bakas, M. Reyes, A. Jakab, S. Bauer, M. Rempfler, A. Crimi, "Identifying the Best Machine Learning Algorithms for Brain Tumor Segmentation, Progression Assessment, and Overall Survival Prediction in the BRATS Challenge," *arXiv preprint arXiv:1811.02629 (,* 2018.

[10] C. C. Aggarwal, Neural Networks and Deep Learning, Springer, 2018.

[11] Ian Goodfellow, Aaron Courville, Yoshua Bengio, Deep Learning, The MIT Press, 2015.

[12] B. Kainz, "Lecture Note in Equivariance," 11 12 2020. [Online]. Available: https://www.doc.ic.ac.uk/~bkainz/teaching/DL/notes/equivariance.pdf. [Accessed 8 1 2023].

[13] Zhang, Richard, "Making Convolutional Networks Shift-Invariant Again," in *International conference on machine learning PMLR 2019*, 2019.

[14] Azulay, Aharon, Yair Weiss, "Why do deep convolutional networks generalize so poorly to small images," arXiv preprint arXiv:1805.12177 , 2018.

[15] Engstrom, Logan, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, Aleksander Madry., "Exploring the Landscape of Spatial Robustness," in *International conference on machine learning*, 2019.

[16] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L., "Imagenet: A large-scale hierarchical image database," in *IEEE conference on computer vision and pattern recognition*, 2009.

[17] Dosovitskiy, Alexey, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv:2010.11929* , 2020.

[18] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 25 (NIPS 2012)*, 2012.

[19] V. Nair ,G. E. Hinton, " Rectified linear units improve restricted boltzmann machines," in *27th International Conference on Machine Learning*, 2010.

[20] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R.R. Salakhutdinov., "Improving Neural Networks by Preventing Co-adaptation of Feature Detectors," in *arXiv preprint arXiv:1207.0580*, 2012.

[21] Karen Simonyan, Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv preprint arXiv:1409.1556,* 2014.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[23] S. Ioffe, C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICLM* , 2015.

[24] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, " Deep residual learning for image recognition," *IEEE conference on computer vision and pattern recognition,* pp. 770-778, 2016.

[25] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin, "Attention is All you Need," *Advances in neural information processing systems,* vol. 30, 2017.

[26] Xiaolong Wang, Ross Girshick, Abhinav Gupta, Kaiming He, "Non-local neural networks," *CVPR,* 2018.

[27] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, sERGEY zAGORUYKO, "End-to-end object detection with transformer," *ECCV,* 2020.

[28] Liu, Ze, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo., "Swin transformer: Hierarchical vision transformer using shifted windows," in

Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021.

[29] Liu, Ze, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

[30] Sudre, Carole H., Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M. Jorge Cardoso., "Generalized dice overlap as a deep learning loss function for highly unbalanced segmentations," *DLMIA 2017, Springer International Publications,* vol. 3, pp. 240-248, 2017.

[31] Diederik P. Kingama, Jimmy Lei Ba, "ADAM, A Method for Stochastic Optimization," in *ICRL 2015*, 2015.

[32] Stephen Boyd, Lieven Vandenberghe, Convex Optimization, Cambrigde University Press, 2004.

[33] Zeiler, Matthew D, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701,* 2012.

[34] Geoffrey Hinton, Ni@sh Srivastava, Kevin Swersky, "Overview of mini-batch gradient descent," 2012. [Online]. Available: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf. [Accessed 8 2 2023].

[35] Duchi, J., Hazan, E., and Singer, Y., "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization,"

Journal of Machine Learning Research, vol. 12, pp. 2121-2159, 2011.

[36] Y. LeCun, C. Cortes, C. Burges, "The MNIST database of handwritten digits," 1998. [Online]. Available: http://yann.lecun.com/exdb/mnist/.

[37] Jun Cheng,Wei Huang,Shuangliang Cao,Ru Yang,Wei Yang,Zhaoqiang Yun,Zhijian Wang,Qianjin Feng, "Enhanced Performance of Brain Tumor Classification via Tumor Region Augmentation and Partition," *PLoS ONE* , vol. 10, no. 10, 2015.

[38] Francisco Javier Díaz-Pernas, Mario Martínez-Zarzuela , Míriam Antón-Rodríguez, David González-Ortega, "A Deep Learning Approach for Brain Tumor Classification and Segmentation using a Multi-scale Convolutional Neural Network," *MDPI,* 2021.

[39] MilicaM. Badža, Marko Cˇ . Barjaktarovic´, "Classification of Brain Tumor from MRI Images Using c Convolutional Neural Network," *Apllied Sciences,* vol. 10, no. 1999, 15 March 2020.

[40] Xavier Glorot, Yoshua Bengio, "Understanding the Difficulty of Training Deep Forward Neural Networks," in *13th International Conference on Artificial Inellegince and Statistics*, 2010.

[41] HOSSAM H. SULTAN , NANCY M. SALEM , WALID AL-ATABANY, "Multi-Classifivatoin of Brain Tumor Images Using Deep Neural networks," *IEEE Access,* 2019.

[42] Zhiguan Huang; Xiaohao Du; Liangming Chen; Yuhe Li; Mei Liu; Yao Chou; Long Jin, "Convolutional Neural Network Based

on Complex Networks for Brain Tumor Image Classificaition with a Modified Activiaiton Function," *IEEE Access,* 2020.

[43] D. Hendrycks , K. Gimpel,, "Gaussian error linear units (GELUs)," *arXiv:1606.08415. [Online]. Available: http://arxiv.org/abs/1606.,* 2016.

[44] Geoffrey E. Hinton, Sara Sabour, Nicholas Frosst, "Dynamic Routing Between Capsules," *Advances in neural information processing systems,* vol. 30, 2017.

[45] Parnian Afshary, Arash Mohammadiy, and Konstantinos N. Plataniotis, "Brain Tumor Type Classification via Capsule Networks," in *IEEE International Conference on Image Processing*, 2018.

[46] Parnian Afshar , Arash Mohammadi , Konstantinos N. Plataniotis, "BayesCap: A Bayesian Approaach to Brain Tumor Classification Using Capsule Networks," *IEEE singal Processing Letters,* vol. 27, pp. 2024-2028, 2020.

[47] Rehman, A., Naz, S., Razzak, M. I., Akram, F., & Imran, M., "A Deep Learning-based Framework for Automatic Brain Tumor Classification usinf Transfer Learning," in *Ciruits, Systems, and Signal Processing*, 2020.

[48] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, "Going Deeper with Convolutions," *arXiv:1409.4842,* 2014.

[49] ZAR NAWAB KHAN SWATI 1,2, QINGHUA ZHAO3, MUHAMMAD KABIR 1, FARMAN ALI1, ZAKIR ALI1, SAEED AHMED1, AND JIANFENG LU, "Content-based Brain Tumlr Retrieval for MR Images Using Transfer Learning," *IEEE Access,* 2019.

[50] Özlem Polat, Cahfer Güngen, "Classification of brain tumors from MR images using deep transfer learning," *The Journal of Supercomputing,,* vol. 77, no. 7, pp. 7236-7353, 2021.

[51] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, "Generative Adversarial Networks," *arXiv:1406.2661,* 2014.

[52] Zhaoqing Pan, Weijie Yu, Xiaokai Yi, Asifullah Khan, Feng Yuan, Yuhui Zheng, "Recent Progress on Generative Adversarial Networks (GANs): A Survey," *IEEE Access,* 2019.

[53] Salome Kazeminiaa,Christoph Baurb, Arjan Kuijper, Bram Van Glinneken, Nassir Navad, Shadi Albarqouni, Anirban Mukhopadhyay, "GANs for Medical Image Analysis," *arXiv:1809.06222v3,* 2019.

[54] Bilal Ahmad, Jum Sum, Qi, You, Vasile Palade, Zhongjie Mao, "Brain Tumor Classification Using a Combination of Variational Autoencoders and Generative Adversarial Networks," *Biomedicines,* vol. 10, no. 2, p. 223, 2022.

[55] Xie, Yuting, Fulvio Zaccagna, Leonardo Rundo, Claudia Testa, Raffaele Agati, Raffaele Lodi, David Neil Manners, and Caterina Tonon., "Convolutional neural network techniques for brain

tumor classification (from 2015 to 2022): Review, challenges, and future perspectives.," *Diagnostics,* vol. 12, no. 8, p. 1850, 2022.

[56] Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning,* vol. 15, pp. 1929-1958, 2014.

[57] Li, D.C.; Liu, C.W.; Hu, S.C., "A learning method for the class imbalance problem with medical data sets," *Computational Biology,* vol. 40, pp. 509-518, 2010.

[58] Mohan, G.; Subashini, M.M., "MRI based medical image analysis: Survey on brain tumor grade classification," *Biomedical singal processing,* vol. 39, pp. 139-161, 2018.

[59] Conner Shorten, Taghi M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data,* vol. 60, no. 6, 2019.

[60] Seyed-Mohsen MD, Alhussein F, Pascal F. , "DeepFool: a simple and accurate method to fool deep neural networks," *arXiv,* 2016.

[61] Dai, Zihang, Hanxiao Liu, Quoc V. Le, and Mingxing Ta, "Coatnet: Marrying convolution and attention for all data sizes," *Advances in Neural Information Processing Systems,* vol. 34, pp. 3965-3977, 2021.

[62] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," *the IEEE conference on computer vision and pattern recognition,* pp. 4510-4510, 2018.

[63] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yangi Zhou, Wei Li, Peter J Liu, "Exploring the limits of transfer learning with a unified text-t-test transformer," . *arXiv preprint arXiv:1910.10683,* 2019.

[64] SL Smith, PJ Kindermans, C Ying, QV Le , "Don't decay the learning rate, increase the batch size," *arXiv preprint arXiv:1711.00489,* 2017.

[65] Wortsman, M., Ilharco, G., Gadre, S.Y., Roelofs, R., Gontijo-Lopes, R., Morcos, A.S., Namkoong, H., Farhadi, A., Carmon, Y., Kornblith, S. and Schmidt, , "Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time.," in *International Conference on Machine Learning* , 2022.

[66] Maron, R.C., Hekler, A., Haggenmüller, S., von Kalle, C., Utikal, J.S., Müller, V., Gaiser, M., Meier, F., Hobelsberger, S., Gellrich, F.F. Sergon, M, "Model soups improve performance of dermoscopic skin cancer classifers," *European Journal of Cancer,* vol. 173, pp. 307-316, 2022.

[67] Ismael MR, Abdel-Qader I, "Brain tumor classification via statistical features and back-propagation neural network," in *IEEE International Conference on Electro/Information Technology*, 2018.

[68] Abiwinanda N, Hanif M, Hesaputra ST, Handayani A, Mengko TR, "Brain tumor classification in MRI using convolutional neural network," *Springer World Congr Med Phys Biomed Eng,* 2018.

[69] Afshar P, Plataniotis KN, Mohammadi A, "Capsule networks for brain tumor classifications based on MRI images and course tumor boundries," in *IEEE International Conference on Acoustics*..

[70] Pashaei A, Sajedi H, Jazayeri N, "Brain tumor classification via convolutional neural network and extreme learning machines," in *IEEE 8th International Conference on Computer and Knowledge Engineering* , Mashhad, Iran, 2018.

[71] Phaye SSR, Sikka A, Dhall A, Bathula DR, "Dense and diverse capsule networks: making the capsule learn better," 2018. [Online]. Available: http://arxiv.org/abs/abs/1805.04001arXiv:abs/1805.04001.

[72] Seetha J, Selvakumar Raja S, "Brain tumor classification using convolutional neural networks," *Biomed Pharmacol J,* vol. 11, no. 3, pp. 1457-1461, 2018.

[73] Avşar E, Salçın K, "Detection and classification of brain tumours from MRI images using faster R-CNN.," *Tehnički Glasnik,* vol. 13, no. 4, p. 337–342, 2019.

[74] Zhou Y, Li Z, Zhu H, Chen C, Gao M et al, "Holistic brain tumor screening and classification based on densenet and recurrent neural network glioma multiple sclerosis stroke and traumatic brain injuries," *Springer International Publishing, Brainlesion,* p. 208–217, 2019.

[75] Anaraki AK, Ayati M, Kazemi F , "Magnetic resonance imaging-based brain tumor grades classification and grading via

convolutional neural networks and genetic algorithms," *Biocyber Biomed Engineering,* vol. 39, no. 1, pp. 63-74, 2019.

[76] Gumaei A, Hassan MM, Hassan MR, Alelaiwi A, Fortino G, "A hybrid feature extraction method with regularized extreme learning machine for brain tumor classification.," *IEEE Access,* vol. 7, p. 36266–36273, 2019.

[77] Sultan HH, Salem NM, Al-Atabany W, "Multi-classification of brain tumor images using deep nerual network," *IEEE Access,* vol. 7, p. 69215–69225, 2019.

[78] Deepak, S.; Ameer, P.M., "Brain tumor classification using deep CNN features via transfer learning," *Computational Biology,* vol. 111, 2019.

[79] Kaplan K, Kaya Y, Kuncan M, Ertunç HM, "Brain tumor classification using modified local binary patterns LBP feature extraction methods," *Medical Hypothessis,* vol. 139, 2020.

[80] Ghaffari, Mina & Sowmya, Arcot & Oliver, Ruth. , "Automated Brain Tumor Segmentation Using Multimodal Brain Scans: A Survey Based on Models Submitted to the BraTS 2012-2018 Challenges," *IEEE Reviews in Biomedical Engineering,* vol. 10, no. 110, 2019.

[81] O. Ronneberger, P. Fischer, and B. Thomas,, "U-Net: Convolutional Networks for Biomedical Image Segmentation," *Medical Image Computing and Computer-Assisted Intervention (MICCAI),* vol. 9351, p. 234–241, 2015.

[82] Long, Jonathan, Evan Shelhamer, Trevor Darrell, "Fully convolutional networks for semantic segmentation," *IEEE conference on computer vision and pattern recognition,* pp. 3431-3440. , 2015.

[83] Çiçek, Özgün, Ahmed Abdulkadir, Soeren S. Lienkamp, Thomas Brox, and Olaf Ronneberger, "3D U-Net: learning dense volumetric segmentation from sparse annotation," *International conference on medical image computing and computer-assisted intervention,* pp. 424-432, 2016.

[84] Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473,* 2014.

[85] Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," in *arXiv preprint arXiv:1409.0473* , 2014.

[86] Zhang, Jianxin, Xiaogang Lv, Hengbo Zhang, and Bin Liu., "AResU-Net: Attention residual U-Net for brain tumor segmentation.," *Symmetry,* vol. 12, no. 5, p. 721., 2020.

[87] Jiang Zeyu acahsngxing Ding, Minfeng Liu, Dacheng Tao, "Two-stage cascaded U-Net: 1srt place solution to Brats challange 2019," *International MICCAI brain lesion workshop,* pp. 231-241, 2019.

[88] Talha Bukhari, Syed, and Hassan Mohy-ud-Din. , "E1D3 U-Net for Brain Tumor Segmentation: Submission to the RSNA-ASNR-

MICCAI BraTS 2021 Challenge," *arXiv e-prints (2021): arXiv-2110.*, 2021.

[89] Wang, Wenxuan, Chen Chen, Meng Ding, Hong Yu, Sen Zha, and Jiangyun Li. "Transbts: Multimodal brain tumor segmentation using transformer, "Transbts: Multimodal brain tumor segmentation using transformer," *International Conference on Medical Image Computing and Computer-Assisted Intervention,* pp. 109-119, 2021.

[90] Hatamizadeh, Ali, Yucheng Tang, Vishwesh Nath, Dong Yang, Andriy Myronenko, Bennett Landman, Holger R. Roth, and Daguang Xu, "Unetr: Transformers for 3d medical image segmentation," *IEEE/CVF Winter Conference on Applications of Computer Vision,* pp. 574-584, 2022.

[91] Hatamizadeh, Ali, Vishwesh Nath, Yucheng Tang, Dong Yang, Holger R. Roth, and Daguang Xu, "Swin unetr: Swin transformers for semantic segmentation of brain tumors in mri images," *MICCAI Brainlesion Workshop, Springer,* pp. 272-284, 2022.

[92] Jia, Qiran, Hai Shu., "Bitr-unet: a cnn-transformer combined network for mri brain tumor segmentation," *MICCAI Brainlesion Workshop,,* pp. 3-14, 2022.

[93] Woo, S., Park, J., Lee, J. Y., Kweon, I. S., "CBAM: Convolutional Block Attention Module," *Proceeding of the European conference on computer vision (ECCV),* pp. 3-19, 2018.

[94] Zhou, Z., Siddiquee, M. M. R., Tajbakhsh, N., Liang, J., "Unet++: Redesigning skip connections to exploit multiscale features in

image segmentation," *IEEE transections multiscale features in images segmentationn,* vol. 39, no. 6, pp. 1856-1867, 2019.

[95] Hou, Yichen., "Research on segmentation of MRI brain tumor image based on improved UNet3+," *International Conference on High Performance Computing and Communication (HPCCE 2021),* vol. 12162, pp. 242-247, 2022.

[96] Qin, Chuanbo, Yujie Wu, Wenbin Liao, Junying Zeng, Shufen Liang, Xiaozhi Zhang, "Improved U-Net3+ with stage residual for brain tumor segmentation," *BMC Medical IMaging 22,* vol. 22, no. 1, pp. 1-15, 2022.

[97] Myronenko, Andriy, "3D MRI brain tumor segmentation using autoencoder regularization," *International MICCAI Brainlesion Workshop,* pp. 311-320, 2018.

[98] Kamnitsas, Konstantinos, Wenjia Bai, Enzo Ferrante, Steven McDonagh, Matthew Sinclair, Nick Pawlowski, Martin Rajchl et al., "Ensembles of multiple models and architectures for robust brain tumour segmentation," *International MICCAI brainlesion workshop,* pp. 450-462, 2017.

[99] Kamnitsas, Konstantinos, Enzo Ferrante, Sarah Parisot, Christian Ledig, Aditya V. Nori, Antonio Criminisi, Daniel Rueckert, and Ben Glocker. , "Deep Medic for Brain Tumor Segmentation," *International workshop on brain lesion,* pp. 138-149, 2016.

[100] vIsensee, Fabian, Paul F. Jaeger, Simon AA Kohl, Jens Petersen, and Klaus H. Maier-Hein, "nnU-Net: a self-configuring method

for deep learning-based biomedical image segmentation," *Nature methods* , vol. 18, no. 2, pp. 203-211, 2021.

[101] Futrega, Michał, Alexandre Milesi, Michał Marcinkiewicz, and Pablo Ribalta, "Optimized U-Net for brain tumor segmentation," *International MICCAI Brain lesion Workshoop,* pp. 15-29, 2022.

[102] Wang, Haonan, Peng Cao, Jiaqi Wang, and Osmar R. Zaiane, "Uctransnet: rethinking the skip connections in u-net from a channel-wise perspective with transformer," *AAAI Conference on Artificial Intelligence,* vol. 36, no. 3, p. 244, 2022.

[103] Futrega, Michał, Alexandre Milesi, Michał Marcinkiewicz, Pablo Ribalta, "Optimized U-Net for brain tumor segmentation," *International MICCAI Brainlesion Workshop,* pp. 15-29, 2022.

[104] Tong He, Zhi Zhang, Hang Zhang, Zhongyu,e Zhang Junyuan, Xie Mu Li, "Bag of Tricks for Image Classification with Convolutional Neural Networks," *arXiv:1812.01187v2,* 2018.

Appendix (1)

In this appendix, we present the results of a side initial study we conducted which shows that using the expanded dataset gives better results compared to using only the original one. This study served as our basis for expanding the dataset we used. We only used CNN architecture which we presented in chapter number 4. We will compare both results of CNN in chapter 4 which was trained on the extended dataset, and CNN of this appendix which is trained on the original dataset without extension. We used the same settings as we did in chapter 4 (refer to setup section). In addition, for the original dataset, we split the training and testing as 82% and 18%, respectively.

Extended Dataset

CNN Layers	Accuracy	Precision	Recall	F1-Score
2	0.85	0.87	0.88	0.87
3	0.89	0.91	0.95	0.98
4	0.91	0.93	0.95	0.94
5	0.90	0.91	0.95	0.93
6	0.89	0.91	0.96	0.94

Original (Non-extended) Dataset

CNN Layers	Accuracy	Precision	Recall	F1-Score
2	0.47	0.00	0.00	0.00
3	0.47	0.00	0.00	0.00
4	0.57	0.85	0.84	0.85
5	047	0.00	0.00	0.00
6	047	0.00	0.00	0.00

Clearly, using the original dataset alone causes the model to have lower accuracies. In the case of 2, 3, 5 and 6 CNN layers it even causes the model to diverge as the accuracy is very low and the other metricizes are shown as zeros. This is due to the fact that the dataset is very small which makes it very likely to overfit. Whereas when the dataset is extended, the model converges to much better results. As such, this study proved to us that using the extended dataset gives better results.

الملخص

ادت التطورات الحديثة في تقنيات التعلم العميق إلى تحسين تشخيص أورام المخ بشكل كبير من خلال تمكين تطوير نماذج قوية. تهدف هذه الرسالة إلى المساهمة في تطوير أنظمة تشخيص أورام المخ من خلال استكشاف وتحسين تقنيات التعلم العميق الحديثة المختلفة، مع التركيز على مهمتين: التصنيف والتجزئة.

تتألف المهمة الأولى من سلسلة من ثلاث دراسات مكثفة، تتضمن تقييمًا شاملاً، وتحليل النتائج، والمقارنات. في الدراسة الأولى، تمت مقارنة أداء النماذج المعيارية المدربة مسبقًا وغير المدربة مسبقًا مثل الفي جي حي والريزنيت بنماذج السي ان ومحول الرؤية المصممة خصيصاً. تطبق الدراسة الثانية خورزمية نموذج الحساء لاختيار النماذج عالية الأداء من الدراسة الأولى لتحسين الدقة. في الدراسة الثالثة، تم استخدام نموذج الكوات نت مُدرَّب مسبقًا لتصنيف أورام المخ وتعزيزه باستخدام تقنيات مثل الزيادات، وتحجيم الدُفعات الكبيرة، والتطبيع، ومعدل التعلم المتدهور بشكل كبير. حقق نموذجنا أعلى دقة بلغت 99.16%، متجاوزًا أحدث النماذج، وتمت مقارنة نتائجنا بنتائج الأدبيات السابقة.

في المهمة الثانية، نقوم بتقييم أداء العديد من نماذج اليونت الحد الأدنى من تكوين اتصالات التخطي على تقسيم الورم الأرومي الدبقي، وهو نوع محدد من ورم الدماغ الخبيث. أظهرت النتائج أن حذف المسار الأول يؤدي إلى التكوين الأمثل لاتصال تخطي اليونت.

بشكل عام، تستخدم هذه الأطروحة نهجًا صارمًا لاستكشاف إمكانات تقنيات التعلم العميق لتشخيص ورم الدماغ، .والنتائج لها آثار مهمة لتحسين دقة تشخيص ورم الدماغ.

جامعة البحرين

كلية الهندسة

قسم الهندسة الكهربائية والالكترونية

تصنيف وتجزئة اورام الدماغ باستخدام تقنيات التعلم العميق

أطروحة مقدمة كجزء من متطلبات الحصول على درجة الماجستير في نظم الذكاء الاصطناعي

اعداد

جواهر عادل علي البنكي

اشراف

بروفيسور إبراهيم مطر

مايو 2023