A Novel EfficientNet-based Approach for Brain Tumor Detection

Abel Gonfa Guta

A Thesis Submitted to

The department of Computer Science and Engineering

School of Electrical Engineering and Computing

Presented in Partial Fulfillment of the Requirement for the Degree of Master's in Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

May, 2024

Adama, Ethiopia

A Novel EfficientNet-based Approach for Brain Tumor Detection

Abel Gonfa Guta

Advisor: Worku Jifara (PhD)

A Thesis Submitted to

The department of Computer Science and Engineering

School of Electrical Engineering and Computing

Presented in Partial Fulfillment of the Requirement for the Degree of Master's in Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

May, 2024

Adama, Ethiopia

Approval

I, the advisor of the thesis entitled "A Novel EfficientNet-based Approach for Brain Tumor Detection" and developed by Abel Gonfa Guta, hereby certify that the recommendation and suggestions made by the board of examiners are appropriately incorporated into the final version of the thesis.

Worku Jifara (PhD.) _____ _____

Major Advisor Signature Date

We, the undersigned, members of the Board of Examiners of the thesis by Abel Gonfa Guta have read and evaluated the thesis entitled "A Novel EfficientNet-based Approach for Brain Tumor Detection" and examined the candidate during open defense. This is, therefore, to certify that the thesis is accepted for partial fulfillment of the requirement of the degree of Master of Science in Computer Science and Engineering.

Chairperson	Signature	Date
Internal Examiner	Signature	Date
External Examiner	Signature	Date

Finally, approval and acceptance of the thesis is contingent upon submission of its final copy to the Office of Postgraduate Studies (OPGS) through the Department Graduate Council (DGC) and School Graduate Committee (SGC).

Department Head	Signature	Date
School Dean	Signature	Date
Office of Postgraduate Studies, Dean	Signature	Date

Declaration

I hereby declare that this Master Thesis entitled “A Novel EfficientNet-based Approach for Brain Tumor Detection” is my original work. That is, it has not been submitted for the award of any academic degree, diploma or certificate in any other university. All sources of materials that are used for this thesis have been duly acknowledged through citation

Abel Gonfa Guta	Signature	Date

Recommendation

I, the advisor of this thesis, hereby certify that I have read the revised version of the thesis entitled “A Novel EfficientNet-based Approach for Brain Tumor Detection” prepared under my guidance by Abel Gonfa Guta submitted in partial fulfillment of the requirements for the degree of Maters of Science in Computer Science and Engineering. Therefore, I recommend the submission of revised version of the thesis to the department following the applicable procedures.

_____	_____	_____
Major Advisor	Signature	Date

ACKNOWLEDGEMENT

I wish to extend my sincerest thanks to all individuals who played a part in the successful completion of this research endeavor.

Most importantly, Thank you, God, for the blessings you showered on me and my family.

I am especially deeply grateful to my Advisor, Worku Jifara (PhD), for his most valuable guidance and support throughout this entire journey. His expertise and mentorship have been greatly instrumental in shaping this research.

It would be such an avarice if I wouldn't be grateful to Dr. Getnet for this research work. Though I met him a few days, the support and kindness were extraordinary. Thankyou Dr.

I also want to extend my appreciation to my colleagues and friends, who helped, fed back, and provided morale support in all phases of the project. Your insight and encouragement really stood motivating.

In addition, I would like to thank the participants for the countless hours of their time and the valuable views they freely gave; without them, this study could not have happened.

I really do appreciate the support from my family members, unending love, understanding, and patience throughout; it was supportive towards my aim.

This research was impossible without the input of all of them, whom I have mentioned above and I am grateful.

CONTENTS

ACKNOWLEDGEMENT i
CONTENTS ii
LIST OF TABLES iv
FIGURE v
ACRONYMS vi
ABSTRACT vii
CHAPTER 1 INTRODUCTION 1
1.1. Background of the study 1
1.1.1 Brain Tumor 1
1.1.2 Deep Learning 4
1.2 Statement of the problem 4
1.3 Research Questions 5
1.4 Objectives 5
1.4.1 General objective 5
1.4.2 Specific objective 5
1.5 Significance of the Study 6
1.6 Scope and limitation 6
1.6.1 Scope 6
1.6.2 Limitation 6
CHAPTER 2 LITERATURE REVIEW 7
2.1. Brain tumor and deep learning 7
2.2. Related Work 9
CHAPTER 3 METHODOLOGY 13
3.1. Method and Dataset Sampling 13
3.2. Tools and Techniques 14
3.3. Evaluation 15
CHAPTER 4 PROPOSED APPROACH 18
4.1. Dataset and its Processing 18
4.1.1 Dataset Processing 22
4.2. Baseline model: EfficientNet V-2 24
4.2.1. Neural architecture search 25

4.2.2. Fused MBConv 27
4.3. Experiment Configuration 28
4.3.1. Hyperparameters 28
4.3.2. Loss Function 31
4.3.3. Validation Strategy 32
4.4. Pruning 32
CHAPTER 5 Results and Discussions 36
5.1. Experiments 36
5.2. Mode Evaluation and Comparison 44
5.3. Research question discussion 46
CONCLUSIONS AND RECOMMENDATIONS 47
Conclusion 47
Recommendation 48
REFERENCES 49

LIST OF TABLES

Table 2-1 Summary of related works alongside their gap 12
Table 3-1 Classification report overview 17
Table 5-1 Result comparison among different models 45

FIGURE

Figure 1-1 Brain Tumor 3
Figure 1-2 Architecture of Convolutional Neural Network 4
Figure 3-1 EfficientNetV2-S architecture 14
Figure 3-2 Confusion Matrix 15
Figure 4-1 Sample images from Br35H :: Brain Tumor Detection 2020 18
Figure 4-2 Dataset Distribution of Br35H: Brain Tumor Detection 2020 19
Figure 4-3 Sample images from Brain Tumor Classification (MRI) 19
Figure 4-4 Dataset distribution of the Brain Tumor Classification (MRI) dataset 20
Figure 4-5 Code snippet: dataset splitting to train and validation 20
Figure 4-6 Code snippet: Dataset splitting validation and testing 21
Figure 4-7 Dataset distribution after splitting 21
Figure 4-8 Code snippet: Prefetching dataset 22
Figure 4-9 Sample of Augmented images 23
Figure 4-10 accuracy comparison between various deep learning models 25
Figure 4-11 Reinforcement learning method for NAS 26
Figure 4-12 MBConv and Fused-MBConv 28
Figure 4-13 learning rate comparison 29
Figure 4-14 Weight pruning 33
Figure 4-15 Neuron Pruning 33
Figure 4-16 Filter Pruning 34
Figure 4-17 Channel pruning 34
Figure 5-1 Transfer learning code snippet 36
Figure 5-2 Custom Head for the existing model 37
Figure 5-3 Model summary of the proposed architecture 37
Figure 5-4 Accuracy and loss graph: EfficientNet v2s trained with a custom head layer 38
Figure 5-5 Unfreezing layers for training code snippet 39
Figure 5-6 Accuracy and loss graph: EfficientNet v2s trained with custom head layer and fine-tuned the last 413 layers 39
Figure 5-7 Dataset distribution after merging datasets 41
Figure 5-8 Code Snippets: Pruning 42
Figure 5-9 Training and validation accuracy: EfficientNet v2s after pruning 43
Figure 5-10 Training and validation loss: EfficientNet v2s after pruning 43
Figure 5-11 Evaluation of the proposed model 44

ACRONYMS

AUC	Area Under Curve
GradCAM	Gradient-Weighted Class Activation Mapping
CNN	Convolutional Neural Network
CNS	Central Nervous System
CT	Computerized Tomography
DBN	Deep Belief Network
DL	Deep Learning
DNN	Deep Neural Network
DRL	Deep Reinforcement Learning
FC	Fully Connected
FPR	False Positive Rate
GPU	Graphics Processing Unit
HDD	Hard Disk Drive
IBM	International Business Machines
LIME	Local Interpretable Model-Agnostic Explanation
LSTM	Long Short-Term Memory
ML	Machine Learning
MRI	Magnetic Resonance Image
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
SHAP	Shapley Additive Explanation
SSD	Solid State Drive
TPR	True Positive Ratio
VGG	Visual Geometry Group
WHO	World Health Organization
XAI	Explainable Artificial Intelligence

ABSTRACT

Accurate and early detection of brain tumors is important to improve patient outcomes, but this has intrinsically been somewhat subjective and contemporarily time-consuming and expensive. We will outline an approach in this groundbreaker that empowered deep learning and the efficiency of EfficientNetV2 in revolutionizing brain tumor detection. Our approach outperforms existing methods on conventional brain tumor datasets, achieving more than 99.6% accuracy in the classification of tumors into clinically relevant subtypes. Only with an EfficientNetV2 as the backbone, combined with carefully designed data augmentation techniques, pruning techniques, and transfer learning strategies, the model can realize high accuracy with drastically fewer parameters and significantly less computation compared to other deep learning approaches applied for brain tumor detection. This makes our approach highly scalable and feasible for translation to resource-constrained clinical environments. We have also introduced a pruning technique, which makes it possible to reveal the insights into model compression and parameter efficiency. Thus, our work has huge implications for the potential that deep learning, and more so EfficientNetV2 architecture, holds in terms of changing the landscape of brain tumor detection to fast, accurate, and cheap diagnosis toward better patient outcomes and saving lives.

CHAPTER 1

INTRODUCTION

Brain tumors are abnormal growths of tissue in or around the brain and sometimes could be either cancerous or non-cancerous (Brain Tumor: Introduction | Cancer.Net, n.d.). Naturally, they can affect any part of the brain if they increase in size to press on nearby nerves, blood vessels, and tissue by changing the brain's functioning, even one's health. They can be classified as primary or metastatic (*Brain Tumor Overview - Harvard Health*, n.d.). Primary brain tumors grow in the brain, while metastatic brain tumors develop elsewhere in the body and spread to the brain. According to Brain Tumor Overview - Harvard Health, several researches have been made to categorical and detect this devastating disease at an early stage, including deep learning detection and classification models. This chapter covers the study's overall context, problem statement, research questions, aims, significance, and scope.

1.1. Background of the study

1.1.1 Brain Tumor

Brain tumors (see Figure 1-1) will yield two broad categories in general: malignant and benign. The former are characterized by rapid growth with infiltration, accompanied by destruction of the surrounding healthy brain tissue. Such tumors are malignant in condition due to the potential of recurring or spreading to other parts of the brain or, in very rare cases, to other organs in the body. Glioblastoma is defined as a form of astrocytoma and described as the most common type of malignant brain tumor, representing the fast-growing characteristic. They are known to grow quite fast and spread to other tissue in the brain, making it a very hard thing to treat. Other examples include medulloblastomas, mainly in children, and metastatic brain tumors that have their origins from primary cancers elsewhere in the body and secondarily invade the brain. In general, malignant brain tumors do have a worse prognosis in contrast to noncancerous brain tumors. Tumor location, size, and grade are variable factors which may have powerful implications for prognosis consideration and the treatment options of a patient.

On the contrary, benign tumors are non-cancerous cell growths and will not diffuse to invade other parts of the brain or body. These type of nature tumors grow very slowly, and it is less likely for them to invade the tissue of surrounding normal brain than for malignant tumors. Commonly occurring non-malignant tumors of the brain include Craniopharyngiomas, Pituitary adenomas, Acoustic neuromas, and Meningiomas. Although not as aggressive, non-malignant tumors can still cause a lot of health problems when their size increases and they exert pressure on the tissue of the brain that is lying around them. In that case, observation would also become part of the treatment option along with surgery and radiation therapy or both.

While knowledge of the different brain tumors forms a basis for diagnosis and treatment, equally important is the prognosis of these tumors. Cancer prognosis refers to an estimation or evaluation of the possible course of the disease, possible recovery from the disease, or possible cycle of survival. It's, therefore, a prognosis of the probable result of the disease one has or the likelihood of recovery or its re-occurrence. Prognosis is determined based upon such factors as the type and stage of cancer, age, and general condition of the patient, and the response to treatment. Cancer survival statistics are results based on studies of large groups of people who have the same type and stage of cancer. Such statistics depict how likely it is that certain treatment will produce a miracle or whether the case is likely to be responsive or hard to treat.

The latest World Health Organization (WHO) worldwide facts on brain tumors reveal the significant impact of this disease on individuals and society. According to the 2019 Global Burden of Disease Study, there were 347,992 global cases of brain and Central Nervous System (CNS) cancers, which showed a significant increase (94.35%) from the period between 1990 to 2019 (Fan et al., 2022).

Although search results show that Brain tumors in Ethiopia are not among the most common types of cancer, representing only a small percentage of identified pediatric cancers in the country, specifically 3% (Memirie et al., 2018), this could be due to the fact that the incidence rate of primary brain tumors in Ethiopia is not well-documented because of limited cancer research resources (Admasu et al., 2022). The 2018 Global Cancer Observatory estimates indicate that brain and nervous system cancers were the 17th cause of cancer morbidity and the 15th in mortality in Ethiopia (Meseret Assefa, n.d.).

Brain tumors' symptoms depend on their size, location, and growth rate. Common ones include headache, fits, nausea and vomiting, problems in vision/hearing, speech trouble/understanding language, and alterations in moods/character . According to (*Brain Tumor - Symptoms and Causes - Mayo Clinic*, n.d.) brain tumors are poorly understood; still, doctors have isolated some factors that may increase the risk. It also gives risk factors, which involve radiation exposure; a family history with regard to tumors of the brain; as well as various inherited syndromes that can increase the person's chance of developing tumors within the organ under discussion (*Brain Tumor Overview - Harvard Health*, n.d.).

A brain tumor may be detected by the gross lesion picture with physical examination, imaging studies by magnetic resonance imaging or Computerized Tomography, CT scan, and a biopsy for histologic tissue examination of a tumor sample (*Brain Tumor - Symptoms and Causes - Mayo Clinic*, n.d.). Medical doctors may also conduct a neurological examination to track brain functionality and detect an individual's brain tumor location (*Brain Tumors: Overview of Types, Diagnosis, Treatment Options | Cincinnati, OH Mayfield Brain & Spine*, n.d.). Treatments of brain tumors, however, rely on the type, size, and position of a tumor, in conjunction with one's age and one's overall health. Treatment may come in surgery, radiation, chemotherapy, immunotherapy, targeted therapy, etc., according to the advice of a doctor; other factors again may be combined (*Brain Tumor: Introduction | Cancer.Net*, n.d.).

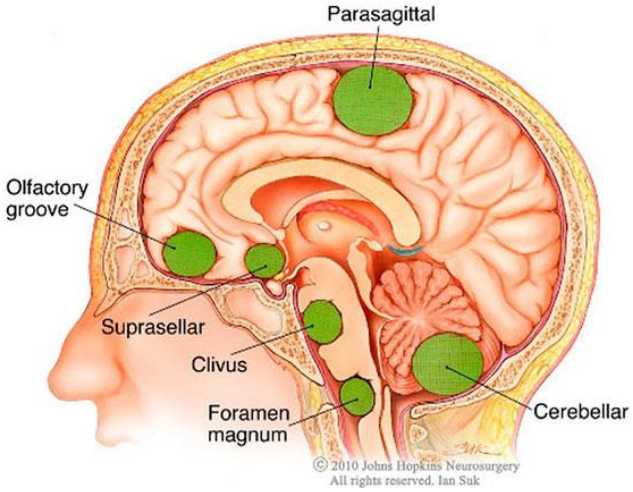

Figure 1-1 Brain Tumor

Image source (*The Most Common Brain Tumor: 5 Things You Should Know | Johns Hopkins Medicine*, n.d.)

1.1.2 Deep Learning

Deep learning finds applications in several fields, including health and pharma sectors, since it belongs to Artificial Intelligence and Machine Learning. Cancer prognosis is involved in the prediction of the outcome of cancer patients and their obtainable chances for survival. This broad area of this study has models including Convolutional Neural Network, Deep Neural Network, Deep Belief Network, and Long and Short-Term Network—Recurrent Neural Network. With these models being noted, CNN has scored better results in image, audio, and video processing so far.

Convolutional neural networks stand out from other neural networks due to their exceptional performance when dealing with images. These networks consist of three primary types of layers: the Convolutional layer, the Pooling layer, and Fully-connected (FC) layer (see Figure 1-2). The Convolutional layer serves as the initial layer in a convolutional network. While additional convolutional layers or pooling layers can follow the convolutional layer, the fully-connected layer acts as the final layer. As each layer is added, the complexity of the CNN increases, enabling it to identify larger portions of the image. The earlier layers primarily focus on simple features like colors and edges. As the image data progresses through the CNN's layers, it gradually recognizes more significant elements or shapes of the object until it ultimately identifies the intended object. Figure 1-2 is an example of CNN architecture (*What Are Convolutional Neural Networks? | IBM*, n.d.).

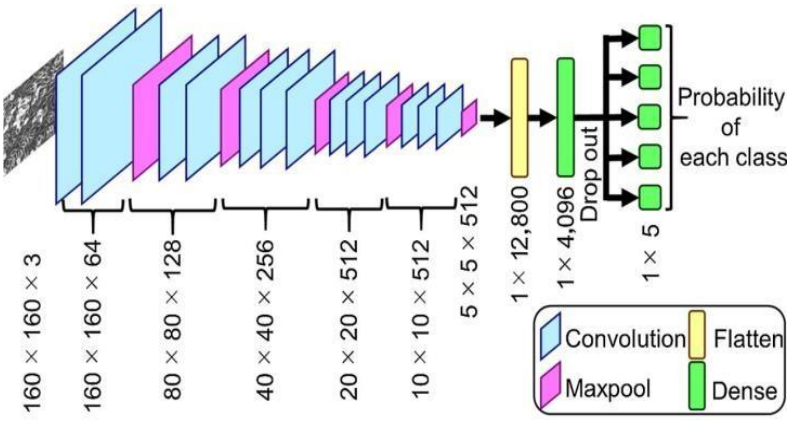

Figure 1-2 Architecture of Convolutional Neural Network

Image source: (*The Algorithms — Entheos A.I. Art*, n.d.)

1.2 Statement of the problem

Magnetic resonance imaging (MRI) brain tumor detection is a difficult and error-prone manual process.(Abdusalomov et al., 2023) To lower these errors by automation techniques,

researchers have been trying to deploy deep learning models to detect and classify brain tumors. However, these models suffer from several significant challenges. Firstly, the manual creation of baseline models is a time-consuming and labor-intensive process, making it difficult to establish a solid foundation for further research. Secondly, even with automated baseline model searches the vast search space of hyperparameters and architectures can lead to an overwhelming number of possibilities, making it challenging to identify the optimal configuration. Additionally, the high training time required to propose baseline models can be a major bottleneck in the research process. Furthermore, the inefficient scaling between model variants, which often involves manual tuning of hyperparameters, can result in suboptimal performance. Moreover, the large number of floating-point operations (FLOPs) in modern deep-learning models can lead to significant computational costs and memory requirements. Lastly, the slow training times for high-resolution images and the slow processing of MBConvs in early layers can hinder the development of efficient and effective models. These challenges collectively hinder the advancement of brain tumor detection tasks. Therefore, it is essential to address these problems and develop a more efficient and scalable model.

1.3 Research Questions

RQ1: Which baseline mode is optimal for the brain tumor detection task?

RQ2: Can a deep learning model performance to classify brain tumors be improved with a smaller number of parameters and minor computational cost?

RQ3: How does data augmentation impact the performance of the proposed model?

RQ4: How does the proposed approach compare to other major deep learning models in previous studies?

1.4 Objectives

1.4.1 General objective

The general objective of this study is to classify Brain Tumor from MRI medical images using the EfficientNet deep convolutional network.

1.4.2 Specific objective

the specific objectives are:

- To retrieve medical images from online public datasets.

- To identify the challenges and gaps in related studies
- To select the optimal baseline model for brain tumor detection
- To Evaluate the proposed efficient net deep CNN model with various detection and classification metrics.
- To compare the performance of the model with the previous state-of-the-art deep learning-based brain tumor detection.
- To measure the impact of data augmentation on the dataset

1.5 Significance of the Study

This research holds significant importance in the field of medical imaging and healthcare. These include but not limited to: improving the accuracy, efficiency, and standardization of brain tumor detection in medical imaging. It has the potential to positively impact patient care, aid radiologists and clinicians in their decision-making process, and contribute to advancements in the field of medical image analysis

1.6 Scope and limitation

1.6.1 Scope

This research focuses on developing an improved Brain Tumor detection model using EfficientNet. A comparative study has been performed on different architectures of CNN and Machine Learning (ML) to evaluate their effectiveness. The dataset which is used in this research is MRI. The improvement of the Brain tumor detection model has been achieved through CNN architecture called EfficientNet which enables multidimensional scaling.

1.6.2 Limitation

This research did not encompass the following due to resource (time, tool, and finance) limitations.

- Medical imaging techniques: the research uses medical images but does not intend to address the gap in the imaging techniques or devices.
- Brain Tumor classification: Brain tumors could be classified based on cell of origin. This study does not aim to classify the source of the disease.
- Due to computational cost, this study used carefully curated healthy and unhealthy 2D MRI images of a brain as dataset, and 3D images are not part of the study.

CHAPTER 2

LITERATURE REVIEW

2.1. Brain tumor and deep learning

For early brain tumor identification, the CNN model is used in(Mahmud et al., 2023). The research compared the suggested architecture to other models, including ResNet-50, VGG16, Inception V3, and ML models, and explored other models in detail. Theproposed model showed promise in this comparison. They discovered that the CNN model had an accuracy of 93.3%, an AUC of 98.43%, a recall of 91.19%, and a loss of 0.25 using a dataset of 3264 MR images. The study's vast dataset made training time and computation expensive, and the authors neglected to take into account specific patient information that would have improved the outcome.

(Gaur et al., 2022) proposed an explanation-driven DL model using a convolutional neural network (CNN), local interpretable model-agnostic explanation (LIME), and Shapley additive explanation (SHAP) for the prediction of discrete subtypes of brain tumors using an MRI image dataset to close the significant gap in explanation, interpretability, and high accuracy for DL models. The model added Gaussian noise to photos of lower quality in terms of noise and metal artifacts and utilized a dual-input CNN technique to overcome the classification difficulty. Comparing their CNN training results to other cutting-edge techniques, they found 94.64% accuracy. This study's limitations include the fact that no augmentation was done to prevent the overall performance reduction caused by feature removal. It was possible to utilize and impose on XAI classification algorithms with higher accuracy and better optimizer. The study could be repeated and used with other XAI algorithms,such as GradCAM, for better clinical problems. Additionally, algorithms to mimic natural occurrences could be applied to heterogeneous datasets for medical imaging modalities, electronic health record engines, multi-omics studies, and real-time monitoring, similar to the most recent advances in computing capacity, neuroimaging technologies, and digital phenotyping tools.

A hybrid deep learning model called Convolutional Neural Network-Long Short-Term Memory (CNN-LSTM) was proposed in the publication (Hassaballah et al., n.d.) for

categorizing and predicting brain cancers through Magnetic Resonance Images (MRI). On a dataset of MRI brain images, they conducted experiments. To extract the important featuresfrom photos, they first applied Convolutional Neural Network (CNN) after having their datapreprocessed. With a 99.1% significant classification accuracy, 98.8% precision, 98.9% recall, and 99.0% F1 measure, the proposed model accurately predicts the brain tumor. Theperformance of their approach to the problem of multi-class MR brain tumor images is not studied, although the results indicated that the proposed model is the best for identifying MR brain images. Different datasets such as Brast2022 and T-weighted which could have enhanced the performance of the proposed model are not also experimented with.

Two deep-learning models were suggested in the research (Khan et al., 2022)to identify and categorize brain cancers. The authors used two openly accessible datasets that each contain 152 and 3064 MRI pictures. On the first dataset, the authors used a convolution neural network (CNN) with 23 layers. On the second dataset, they applied transfer learning and combined the VGG16 architecture with the reflection of the "23 layers CNN" design that was proposed. Their experimental results show that their models reach up to 97.8% and 100% classification accuracy for their utilized datasets, respectively. While their proposed models produced promising classification results, there were still several problems that could have been resolved. For instance, the use of zero-shot, few-shot, and deep reinforcement learning (DRL) techniques could have aided in the demand for exact annotations and high-quality photos as well as the lack of a big annotated image dataset. The work is not validated on genuine clinical studies, which is another flaw in this study. The usage of additional layers or alternative regularization methods to work with a tiny image collection using the CNN model may have been explored by the authors.

Finding and segmenting brain cancers from 2D and 3D brain MRIs was the main goal of the study (Sailunaz et al., 2023). For this, a fully automated system with a web application user interface that uses deep neural networks like CNN, U-Net, and U-Net++ to detect and segment brain tumors with greater than 90% accuracy and Dice scores is described. To enable medical experts to add more detailed comments on the detection and segmentation findings that can be used to train the model for better future predictions and segmentations, the online application included a feedback entry option. There are some limitations on the sorts of input images for each action in this program. Additionally, the system is limited to afew well-known medical image-based DL models, such as the CNN, U-Net, and U-net++

models. The study excludes further analysis of the MRIs for the detection and segmentation of distinct tumor tissues, computation of different tumor characteristics, and tumor/cancer severity prediction.

Among several deep learning models applied to disease detection, EfficientNets are the best performing so far. In a paper titled "Cardamom Plant Disease Detection Approach Using EfficientNetV2" (Sunil et al., 2022), Authors use the EfficentNetsV2 model to detect Cardamom Plant Disease and U^2-Net to remove unwanted background image which has scored 98.26% detection accuracy. Although the model in the study scored the highest accuracy in the detection of the disease, the severity of the disease is not measured in the study.

To overcome the problem of extracting useful information from an image and removing complex background, authors (Ye et al., 2022) proposed algorithm which is based on dual attention mechanisms and EfficientNet. The algorithm combines the channel attention mechanism and spatial attention mechanism to strengthen the key feature information and suppress the interference feature information, such as background noise. The experiments showed that the algorithm can deliver recognition accuracy of up to 99.56%. the study clearly showed that layers adopted to the model are redundant which results in slow convolution operation, much more parameters, and training time.

2.2. Related Work

On (Zoph & Le, 2016) authors propose A method called Neural architecture search aiming at generating machine deep learning-based baseline architecture. In the paper, They provided a briefing stating that a recurrent network was utilized to create the model descriptions of neural networks. Additionally, the RNN was trained using reinforcement learning to optimize the expected accuracy of the generated architectures on a validation set. Their method was indeed successful in designing architecture that has surpassed human-invented architectures in terms of accuracy. Even though this method overcame the manual (try and error) procedure of designing baseline architecture, it required a huge amount of computing power as well as time due to the broad search space.

To overcome the limitations observed in the NAS paper, authors (Zoph et al., 2017) proposed a way to search configuration for blocks rather than individual layers. The research aimed at searching for blocks with multiple convolutional layers and pooling layers using the RL controller and just repeated these blocks N times to create the scalable NASNet architecture

which has reduced the time spent in search space. In addition, the paper introduced a novel regularization technique named ScheduledDropPath, which greatly enhances generalization in the NASNet model. Despite the advantages it has introduced, the research had limitations in considering different platforms.

(Tan et al., 2018) introduced latency as a reward signal to the controller in addition to the previous sole signal of accuracy. The authors of the research chose 7 blocks, and one layer of a block was sampled and repeated for each block. This finding was a breakthrough since it creates not only baseline architectures with high accuracy but also fast training time or low latency which would allow the model to run on mobile or edge devices. But still, this method was not optimal since consideration of optimal scaling the baseline mode was in demand by the study area.

In the paper EfficientNet (Tan & Le, 2019), there were two major concepts. One is that the rewarding signal in MnasNet was changed from latency to FLOPs aiming to find models with lower power consumption and computational requirements. This shift in the rewarding signal from latency to FLOPs in MnasNet was a strategic decision to prioritize computational efficiency and reduce power consumption in the model design process. The second major finding in this paper was the compound scaling method. The paper introduced a new compound coefficient formula to scale the baseline network which has been designed in RL (FLOP and accuracy as a reward signal) and attain seven different variants of the model. Developers may choose between each variant based on their resources. Despite the findings the paper had limitations in the following cases, first, when large image resolution was used to train the models (B6 or B7 models), the training was slow. Second, In the early layers of the network architecture, depth-wise convolutional layers (MBConv) were slow because they couldn't fully leverage modern accelerators. Lastly, Equal scaling was applied to the height, width, and image resolution to create the variant.

In (Dosovitskiy et al., 2020) the renowned transformers architectures that have been popular and have been used in NLP were made in use for image classification tasks. The new approach aims at reducing computational resources while training the model and indeed was successful. In spite of computing individual pixels, ViT divided the image to create patches which then linearly transformed into a vector using a learnable linear projection and processed the vector by a standard Transformer encoder as used in NLP. While these results

were encouraging, ViT has faced limitations in applying the method to Image detection and segmentation tasks.

EfficientNet V-2, a CNN model that has shown a significant progression in accuracy, parameter efficiency, and training speed was introduced in (Tan & Le, 2021) .Authors of the paper introduced new operations and blocks like fused-MBconv that could use modern accelerators so as to increase training speed. The neural architecture search was done to jointly optimize accuracy, parameter efficiency, and training efficiency. Authors additionally use a scaling method that limits maximum image size to scale between the variants and a Progressive learning method to adaptively change the regularization technique along with increasing image size on training progress. This method is better in all ways than the previous related architecture but the method can be further improved with pruning, which additionally reduces the computations in the training by zeroing out nonimportant weights.

For the matter of easy tracking, Table 2-1 is presented below to show the most related works along side their gaps.

Table 2-1 Summery of related works alongside their gap

Title	Journal	Year	Method	Gap(s)
A Deep Analysis of Brain Tumor Detection from MR Images Using Deep Learning Networks	Algorithms	2023	CNN	Longer training time and high computational cost and
Explanation-Driven Deep Learning Model for Prediction of Brain Tumor Status Using MRI Image Data	Frontiers in Genetics	2022	CNN, LIME, SHAP	No augmentation and low accuracy
Ensemble deep learning for brain tumor detection	Frontiers in Computational Neuroscience	2022	(CNN-LSTM)	Classification problem and dataset variance is not investigated
Accurate brain tumor detection using deep convolutional neural network	Computational and Structural Biotechnology	2022	CNN & VGG 16	Couldn't deal with shortage of annotated images
Brain tumor detection and segmentation: Interactive framework with a visual interface and feedback facility for dynamically improved accuracy and trust	PLOS ONE	2023	CNN, U-Net and U-Net++	Restriction on image type for each operation
Cardamom Plant Disease Detection Approach Using EfficientNetV2	IEEE Xplore	2022	EfficientNetV2	the severity of the disease is not measured
An Improved EfficientNetV2 Model Based on Visual Attention	Computational Intelligence and Neuroscience	2022	EfficientNetV2 with a visual attention mechanism	slow convolution operation, more parameters, and training time

CHAPTER 3

METHODOLOGY

This section offers an extensive elucidation of the architecture and experimental setup of the proposed method. It encompasses the dataset utilized for this study, along with the data preprocessing techniques employed. Additionally, a concise explanation of the model training process is provided. To inquire about the understanding of the proposed system's development method, design the proposed solution, and describe the testing and implementation methods and tools used to achieve the proposed solution.

3.1. Method and Dataset Sampling

Our research followed a clear standard procedure for developing and evaluating brain tumor detection using deep learning techniques. We start by identifying the research area through literature review in which we end up by selecting deep learning as a thematic area in consideration of its vast application in recent years. Then identified the problem which is a brain tumor. Extensive literatures were reviewed with multiple entry points (i.e. authors, keywords, and journals). We have then identified the gaps and proposed a research question and objectives to measure the outcome of our research. Finally, the process looped through finding the right dataset type, size method, and maximum efficiency.

The study employed an EfficientNet-based approach for classifying brain tumors where the uniqueness of the model comes from a parameter reduction technique called pruning. The dataset (*Br35H :: Brain Tumor Detection 2020,* n.d.) and (*Brain Tumor Classification (MRI),* n.d.) used for training and testing our model was carefully curated from an open-source repository called Kaggle. The dataset comprised a diverse collection of brain MRI scans, including both healthy and tumor-affected cases.

To ensure a representative and unbiased sample, we have selected a dataset that maintained an appropriate balance between different tumor types, sizes, and locations within the brain. Additionally, we ensured that the dataset encompassed a wide range of demographic factors, such as age, gender, and ethnicity, to enhance the generalizability of our approach.

Stage	Operator	Stride	#Channels	#Layers
0	Conv3x3	2	24	1
1	Fused-MBConv1, k3x3	1	24	2
2	Fused-MBConv4, k3x3	2	48	4
3	Fused-MBConv4, k3x3	2	64	4
4	MBConv4, k3x3, SE0.25	2	128	6
5	MBConv6, k3x3, SE0.25	1	160	9
6	MBConv6, k3x3, SE0.25	2	256	15
7	Conv1x1 & Pooling & FC	-	1280	1

Figure 3-1 EfficientNetV2-S architecture

Image source (Tan & Le, 2021)

The baseline model used in this study is EfficientNetV2-S, the smaller but still efficient model variant of EfficientNetV2 (See section 4.2 for detailed discussion on EfficientNetV2-S). Figure 3-1 is an EfficientNetV2-S architecture. It shows used operators alongside their stride, number of channels, and number of layers.

3.2. Tools and Techniques

Our methodology was based on the power of deep learning and computer vision techniques in order to properly detect and classify a brain tumor.

The main tools and techniques applied in our work were EfficientNet V-2, a well-known deep learning baseline model, followed by a family of Convolutional Neural Networks used for extracting complex features from brain MRI scans. These models have done a great job in tasks such as image classification and detection. Applied transfer learning, considering that it used knowledge gained from pre-trained models on large-scale datasets. This approach would mean having to train the pre-trained model once and then fine-tune a lot on our dataset of brain tumors. This reduced the training time drastically and increased the performance of the model quite noticeably.

Due to the limited data samples, our training data needed augmentation. With this in mind, we employed a few augmentation techniques, such as rotation and flipping, to avoid overfitting and help improve the generalization capability in our model.

3.3. Evaluation

Deep learning models are really complex, and evaluation goes through a variety of thorough processes to be assured that they act as expected. Evaluation of deep learning models is carried out through different methods that are accompanied by strengths and weaknesses. One major method is the confusion matrix (see Figure 3-2), representing a table showing the number of true positives, true negatives, false positives, and false negatives a model produces.

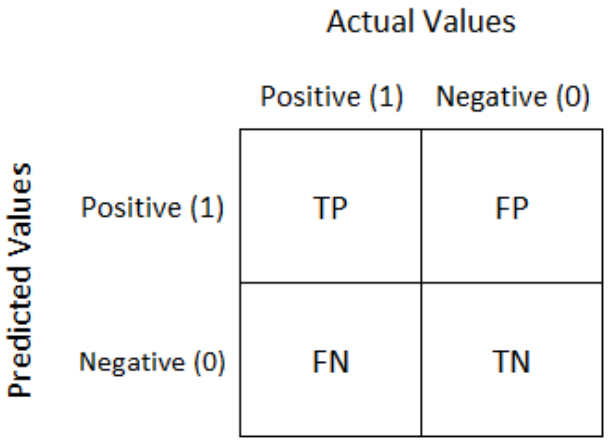

Figure 3-2 Confusion Matrix

From Figure 3-2, various metrics could also be retrieved. These include:

Accuracy: This is one of the major measures, and it quantifies the ratio of correctly classified instances against that of total instances in the test set. The accuracy can be calculated using the number of correct predictions divided by the total number of predictions (see Eq. (1)). This gives a good understanding and proves to be useful when errors are equally costly for both positive and negative classes. It nonetheless bears the caution that accuracy has to be understood with respect to an imbalanced dataset in which one class is grossly overrepresented as compared to the other.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad \ldots (1)$$

Precision: is an important metric that shows the correctness of the positive predictions made by a model. This is compared to the number of true positives divided by the total of true

positives and false positives (see Eq. (2)). This matters a lot in situations where the cost of a false positive is high, like medical diagnosis, where you might end up treating someone when really you shouldn't have.

$$Precision = \frac{TP}{TP + FP} \quad \ldots (2)$$

Recall: also called True positive rate, measures the proportion of actual positive instances that were identified correctly It calculates as shown in Equation (3). Importance of recall would be apparent in cases where false negatives come with a significant consequence, such as we saw in medical diagnosis. In that scenario, if a false negative occurred, it can sometimes prevent the detection of a treating the condition, which can cause harm as well as missed diagnosis.

$$Recall = \frac{TP}{TP + FN} \quad \ldots (3)$$

F1 Score: This takes both false positives and false negatives into account, It is the harmonic mean of Precision and Recall (see Eq. (4)). It provides a balanced view of both metrics since the class with a lower score should be encouraged. When you are interested in both precision and recall, you can use the F1 score; for instance, if you want to classify some images correctly as well as not miss a single recognition at all.

$$F1\ Score = 2 * \frac{\text{Precision} \times \text{Recall}}{Precision + Recall} \quad \ldots (4)$$

Classification Report

A classification report (see Table 3-1) is a comprehensive summary of the behavior of a certain model. It usually has details like the accuracy, precision, recall, and F1 score for each

class. This information gives a holistic view of the performance of the model and can be used to point out the possible room for improvements.

Table 3-1 Classification report overview

	Precision	**Recall**	**F1-Score**	**Support**
Class 0	X	X	X	X
Class 1	X	X	X	X
Accuracy			X	X
Macro Avg	X	X	X	X
Weighted Avg	X	X	X	X

Support: the number of actual occurrences of each class in the dataset. It is a measure of the number of instances in each class, which does not vary between models. The support is used to diagnose the performance evaluation process and provides a better understanding of the model's performance in each class

Macro Average: The per-class F1 scores are averaged without any weighting, resulting in an unweighted average (see Eq. (5)). This approach treats all classes equally, irrespective of their support values. Consequently, the macro average assigns equal significance to each class, regardless of its size. The macro average is computed by taking the arithmetic mean of all the per-class F1 scores.

$$Macro\ Average = \frac{F1_{class1} + F1_{class2}}{2} \quad \dots (5)$$

Weighted Average: it is calculated as the weighted average of the F1 scores for each class, taking into consideration the support values of every class and giving more weight to classes with higher support values (see Eq. (6)). It is obtained by multiplying the value of the support of each class by its F1 score and summing up the resulting products.

$$Weighted\ Average = \frac{F1_{class1} \times Support_{class1} + F1_{class2} \times Support_{class2}}{Total\ Support} \quad \dots (6)$$

CHAPTER 4

PROPOSED APPROACH

The task on brain tumor detection and classification in medical imaging might influence diagnosis, treatment, and prognosis of a patient. Conventional techniques usually applied for the classification of a brain tumor are performed manually through an examination of the medical scan. As such, they are time-consuming and subjective, with a lot of predisposition to human error. (Abdusalomov et al., 2023) Recently, deep learning has gained significant ground in medical image analysis, making it a promising area in the detection and classification of brain tumors that could turn out to be more accurate, efficient, and automated. The approach to brain tumor classification, powered by the EfficientNet V2 family of convolutional neural networks, is the content of this chapter.

4.1. Dataset and its Processing

For the experiments conducted in this research, two datasets were utilized: “Br35H :: Brain Tumor Detection 2020” and “Brain Tumor Classification (MRI) from Kaggle”. The “Br35H :: Brain Tumor Detection 2020” dataset has been used for all experiments alone except the last experiment used a combination of both datasets.

Br35H :: Brain Tumor Detection 2020

The Br35H :: Brain Tumor Detection 2020 dataset is an open-source collection of brain tumor images hosted on Kaggle. This dataset has 3,060 MRI scans. It is split into three folders: ‘yes’ for 1,500 brain tumor images, ‘no’ for 1,500 non-tumor images, and ‘pred’ for saving prediction results (see figure 4-2). Figure 4-1 shows sample images from Br35H:: Brain tumor detection 2020 dataset.

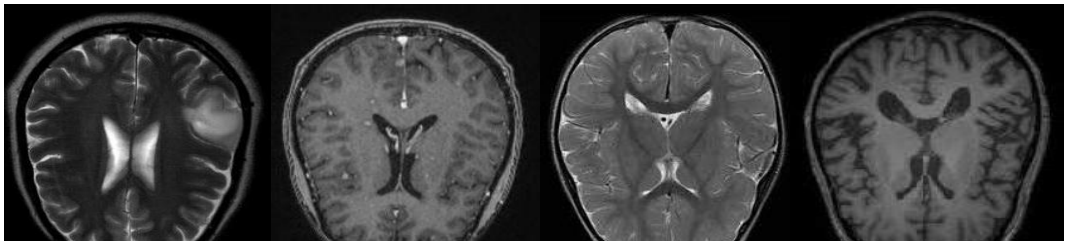

Figure 4-1 Sample images from Br35H :: Brain Tumor Detection 2020

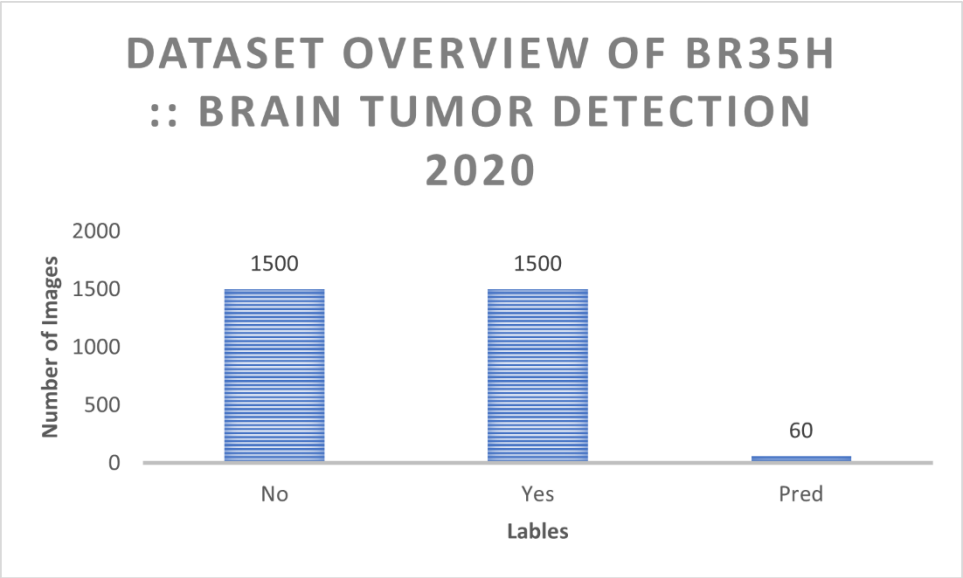

Figure 4-2 Dataset Distribution of Br35H: Brain Tumor Detection 2020

Brain Tumor Classification (MRI)

Brain Tumor Classification (MRI) is a public dataset hosted in Kaggle. This dataset contains MRI images of patients with tumors; these tumors are labeled with respect to their classification (see Figure 4-4 for dataset distribution).

It is organized at the top into four classes: glioma, meningioma, pituitary tumor, and no tumor. This multi-class classification task presents a more challenging problem compared to the binary classification offered by the Br35H :: Brain Tumor Detection 2020 dataset.

The availability of both binary and multi-class data allowed a full-scale assessment of the proposed EfficientNet-based approach. The Br35H :: Brain Tumor Detection 2020 dataset allowed for assessing the performance of the model in detecting the presence or absence of brain tumors, whereas the Brain Tumor Classification dataset for MRI was used for the evaluation of the model's differentiation among various kinds of brain tumors. Figure 4-3 shows sample images from Brain tumor classification (MRI) dataset.

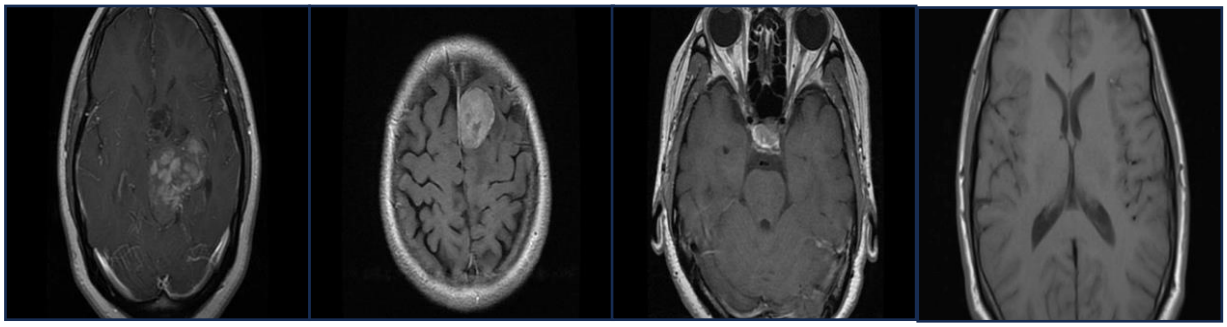

Figure 4-3 Sample images from Brain Tumor Classification (MRI)

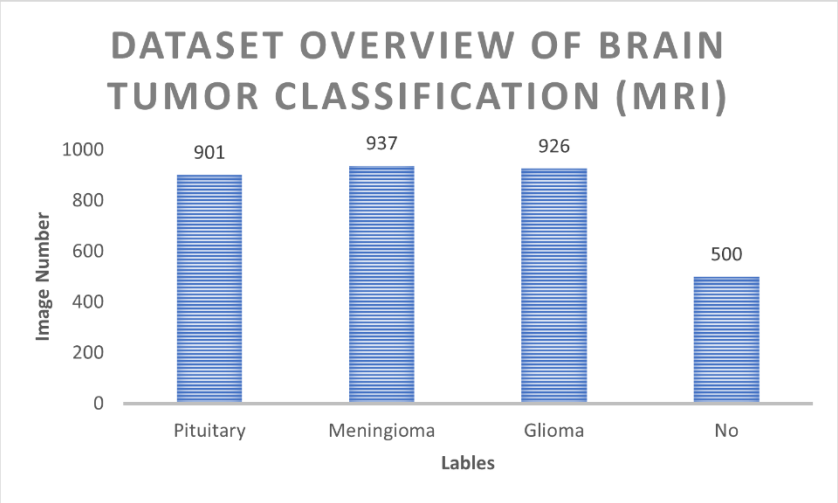

Figure 4-4 Dataset distribution of the Brain Tumor Classification (MRI) dataset

For most experiments in this research, the first dataset was used. The dataset was split into 80% testing and 20% validation. The "splitfolders" library enables the dataset to be split automatically. Figure 4-5 shows the code used to split the dataset into validation and training datasets and the output is stored in the folder named "outpu2". The code also confirms the generated subfolders in the given directory.

```
splitfolders.ratio('brain-tumor-detection', output="output2", seed=1337, ratio=(.8, 0.2))
```

```
Copying files: 3000 files [00:22, 135.98 files/s]
```

```
os.listdir('output2')
```

```
['train', 'val']
```

Figure 4-5 Code snippet: dataset splitting to train and validation

From the total of 3000 images, 2400 become testing and 600 become validation. Among validation datasets one-fifth (80) is assigned to testing and the remaining four-fifths remain as validation dataset. The code snippet below (see Figure 4-6) shows the code used for splitting the validation dataset into validation and testing.

```
val_batches = tf.data.experimental.cardinality(validation_dataset)
test_dataset = validation_dataset.take(val_batches // 5)
validation_dataset = validation_dataset.skip(val_batches // 5)
```

```
print('Number of validation batches: %d' % tf.data.experimental.cardinality(validation_dataset))
print('Number of test batches: %d' % tf.data.experimental.cardinality(test_dataset))
```

```
Number of validation batches: 16
Number of test batches: 3
```

Figure 4-6 Code snippet: Dataset splitting validation and testing

As shown in Figure 4-6 the validation data is split into two with a number of batches (32). Since the test dataset is split logically the change in labels could not be monitored through the folder.

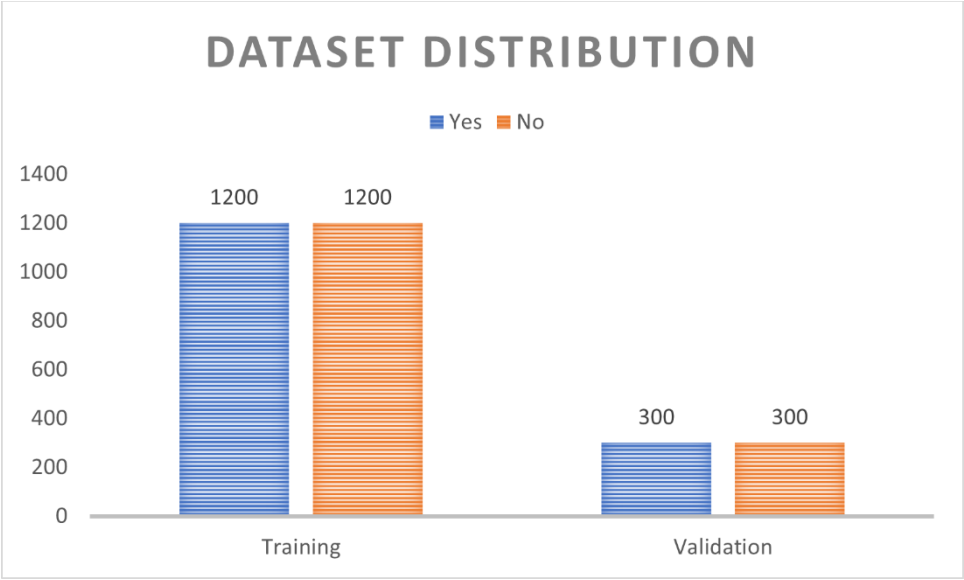

Figure 4-7 Dataset distribution after splitting

Figure 4-7 shows the dataset distribution where all the positive and negative instances are optimum and equal in both train and validation datasets. Having balanced a number of positive and negative instances has the following advantages.

Non-biased: In cases in which one class is incredibly underrepresented with respect to the other, models biased toward the majority class can result in very poor performance for the minority class. A balanced dataset helps mitigate this bias.

Better generalization: Due to the balanced dataset, the model is trained on an equal number of instances of both classes. This can enable it to learn the discriminating features more effectively and hence better generalize on unseen data.

Efficient convergence: Due to the use of class-imbalanced datasets, the model converges prematurely or gets stuck in a local optimum because the majority class dominates the loss function. Therefore, a balanced dataset improves the efficient convergence of the model for achieving better optimum.

Robust feature learning: Deep models have one of their strong points in the ability to learn relevant features by themselves from the data. For instance, if it is a balanced data set, the model could learn discriminative features from both classes, which would make the representations much more robust and accurate.

Trivial solutions should be avoided: Sometimes, especially in the case of extreme class imbalance in the model, it may learn some trivial solutions. For instance, it might constantly predict the majority class, resulting in poor performance on the minority class. Balanced datasets prevent such trivial solutions.

Interpretability: Training on a balanced dataset can provide more interpretability for learned representations and decision boundaries. class

4.1.1 Dataset Processing

Prefetch: while the current batch is being processed, the next batch is being prepared in the background (see Figure 4-8). This helps in reducing the time spent waiting for data during training. Prefetching data helps in overlapping data preprocessing and model training, reducing the overall training time. Moreover, the "tf.data.AUTOTUNE" function ensures that the system adapts to the available resources, maximizing performance without manual tuning while assigning buffer size. The training, validation, and testing datasets are prefetched in this model.

```
AUTOTUNE = tf.data.AUTOTUNE

train_dataset = train_dataset.prefetch(buffer_size=AUTOTUNE)
validation_dataset = validation_dataset.prefetch(buffer_size=AUTOTUNE)
test_dataset = test_dataset.prefetch(buffer_size=AUTOTUNE)
```

Figure 4-8 Code snippet: Prefetching dataset

Data Augmentation: It is a technique used to artificially increase the size of a dataset by applying random transformations to the existing data. Especially useful when the dataset is very small or the model is sensitive to some specific features or orientation. Data

augmentation should only be done after the dataset has been split. In general, techniques that will be used for data augmentation in this work are flipping and rotation.

The layer "RandomFlip('horizontal')" randomly flips the input data horizontally. This means if the input image is a face, then this layer may shift it to the left or right side. The argument 'horizontal' shows along which axis the flip occurs. Here, it occurred along the horizontal axes, the x-axis.

"RandomRotation(0.2)" This layer rotates the input data randomly by some angle. In this particular case, it is a uniform rotation anywhere between 0 and 0.2 radians, which is equivalent to approximately 11.5 degrees. Such may be further used in simulation effects of camera angle, object orientations, or any other types of rotation that exist in reality. Figure 4-9 shows a sample of augmented images from the dataset.

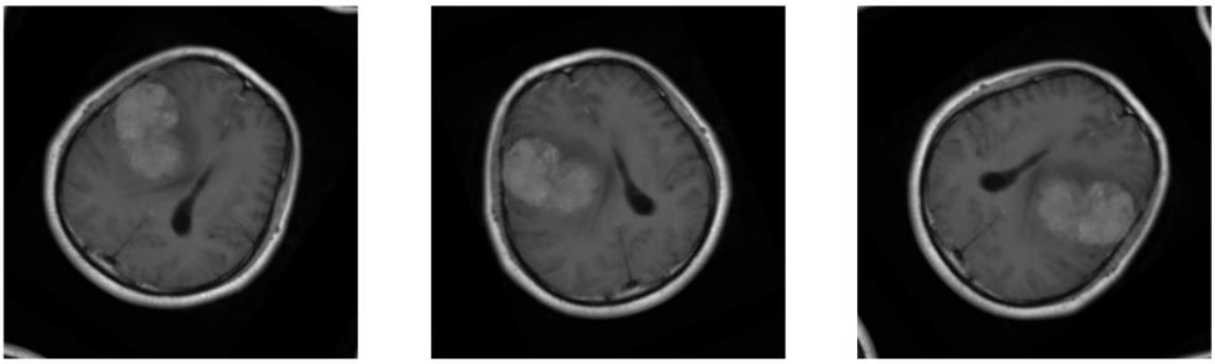

Figure 4-9 Sample of Augmented images

Rescaling: It is one of the methods for rescaling input data to a certain range. Several reasons exist as to why rescaling of input data is very important, which includes Numerical stability and better gradient decent that helps achieve faster convergence and better performance to handle a variety of activation functions. Now, as used in this project, the code for scaling will be explained.

rescale = tf.keras.layers.Rescaling(1./127.5, offset=-1)

It uses a scale factor of 1./127.5, hence will divide every pixel value by 127.5. It uses an offset of -1., meaning that 1 will be subtracted from each rescaled value. Provided the input values are in the normal range of 0 to 255, typical for pixel values in images, this rescaling will transform this to a range of approximately -1 to 1.

The maximum value, 255, will be rescaled to:

$$(255 * 0.00784) – 1 = 0.9992$$

The minimum value, 0, will be rescaled to:

$$(0 * 0.00784) – 1 = -1$$

Thus, every pixel of an image would lie between 1 and –1. Rescaling the input data within this window of –1 and 1 could permit the model to derive some of the mentioned advantages.

4.2. Baseline model: EfficientNet V-2

EfficientNetV2 was developed through a meticulous process that involved network architecture search and scaling to enhance training speed and parameter efficiency. As discussed in the literature review section, the journey of creating EfficientNetV2 began with the introduction of network architecture search (Zoph & Le, 2016), a technique that aims to optimize the structure of neural networks for specific tasks. This search process was combined with scaling to jointly improve training speed and parameter efficiency

EfficientNetV2 takes a step beyond EfficientNet in enhancing the speed of training and the efficiency of parameters. This network is created through a blend of scaling techniques, including width, depth, and resolution, along with neural architecture search. The primary objective is to maximize the efficiency of training speed and parameter utilization. Additionally, the search process incorporates novel convolutional blocks like Fused-MBConv. As a result, the authors successfully devised the EfficientNetV2 architecture, which outperforms both previous and contemporary state-of-the-art models in terms of speed and size, boasting up to 6.8 times reduction in size. Figure 4-10 is a comparison graph that illustrates the performance of various deep learning models against training time and image accuracy.

EfficientNetV2 changed 4 things that improved and made it better from EfficientNet (V1). The first one is Adding a combination of MBConv and Fused-MBConv blocks which enabled the model to fully utilize modern accelerators.

Secondly, while developing the baseline model through neural architecture search EfficientNetV2 used Accuracy, Parameter Efficiency, and Training Efficiency as a reward parameter. Third, the model used an intelligent scaling technique that limits maximum image size to reduce GPU time which leads to an increase in the training speed with minor accuracy

loss. The last improvement made by EfficientNet V-2 is that the model used progressive learning in which the regularization technique is also increased in accordance with the image size while training.

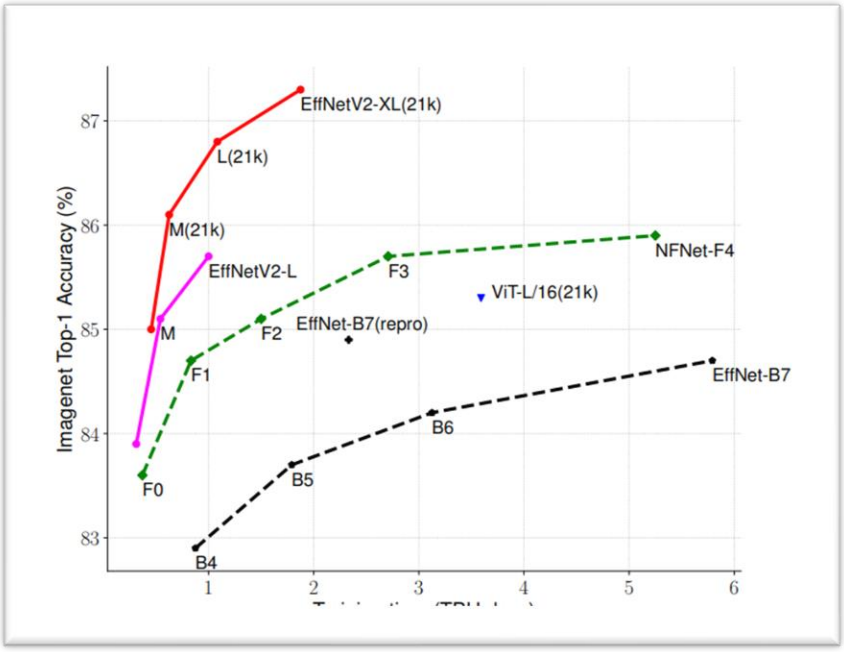

Figure 4-10 accuracy comparison between various deep learning models

Image source: (Tan & Le, 2021)

4.2.1. Neural architecture search

NAS is a new subfield of deep learning that focuses on the automation of neural network architecture design. In that respect, it eases the tedious and prone-to-error process of manual architecture engineering using automated methods to discover brand-new and high-performance neural architectures. NAS is important because it provides better efficiency and effectiveness on a wide array of tasks by machine learning models, from image recognition to speech recognition and machine translation.

Within such a short time, deep learning has reopened the playing field of artificial intelligence to huge improvements by machines at tasks involving sophisticated pattern recognition. Then, there are limits to manual design by human experts in scalability and innovation. Finally, NAS is the evolution of automated machine learning, as it seeks to further automate the process of architecture engineering itself.

In general, NAS can be organized by three major components: Search Space, Search Strategy, and Performance Estimation Strategy.

Search Space: The search space defines the realm of possible neural architectures that a NAS approach can explore. It encompasses the structural elements of neural networks, including layer types, operations, and connectivity patterns. By defining a suitable search space, NAS algorithms can efficiently navigate through a vast array of potential architectures.

Search Strategy: The search strategy dictates how the NAS algorithm explores the defined search space. It involves balancing the exploration of new architectures with the exploitation of promising ones to achieve optimal performance. This strategy is crucial in avoiding premature convergence to suboptimal solutions.

Performance Estimation Strategy: The objective of NAS is to identify architectures that exhibit high predictive performance on unseen data. The performance estimation strategy involves evaluating the effectiveness of different architectures. This process can be computationally expensive, leading to the development of methods that optimize performance estimation while reducing computational costs.

The NAS uses a common Reinforcement learning method (see Figure 4-11) to search for any possible architectures and hyperparameters from the search space which then proposes a child model. Lopes back and forth until it gets the maximum value based on the reward signal.

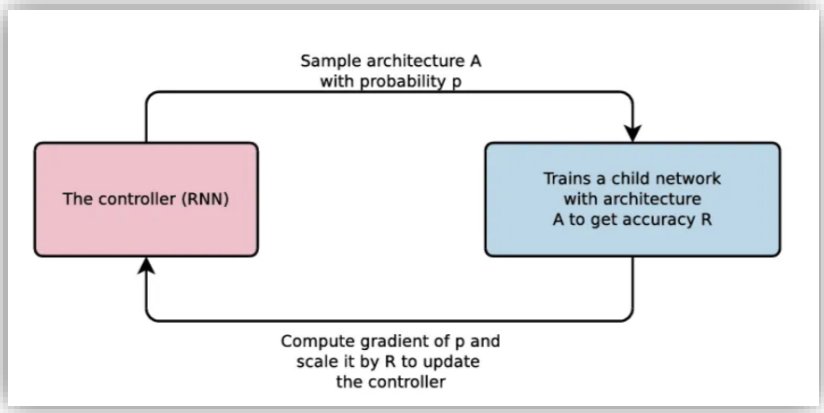

Figure 4-11 Reinforcement learning method for NAS

Image source: (Zoph & Le, 2016)

EfficientNetV2 models were searched from a search space enriched with new operations such as Fused-MBConv. The search space included different convolutional operation types like MBConv and Fused-MBConv. The search space was reduced by removing unnecessary options like pooling skip ops that were never used in the original EfficientNet, and reusing the same channel sizes from the EfficientNet backbone since they were already optimized in the original EfficientNet search. The enriched search space allowed searching for architectures that are faster to train and more parameter efficient compared to the original EfficientNet. The EfficientNetV2 architecture was developed using a combination of training-aware neural architecture search (NAS) and scaling to optimize training speed and parameter efficiency.

4.2.2. Fused MBConv

The EfficientNetV2 architecture introduced a new building block called Fused MBConv, which is a variation of the original MBConv (Mobile Inverted Bottleneck Convolution) block used in EfficientNet models (comparison of the blocks is presented in Figure 4-12). The Fused MBConv block aims to improve training speed and parameter efficiency compared to the original MBConv.

The original MBConv block consists of the following layers:

- A point-wise "expansion" convolution that maps inputs to the inner layer
- A depth-wise convolution layer, typically 3x3
- A point-wise "projection" convolution that maps the inner layer to the outputs

Batch normalization follows each layer, and the expansion and convolution layers have activation functions. The projection layer has no activation function, as the MobileNetV2 paper found that the nonlinearities decrease accuracy when used on the bottleneck layers.

The Fused MBConv block replaces the depth-wise conv3x3 and expansion conv1x1 in MBConv with a single regular conv3x3. This modification aims to better utilize mobile or server accelerators by reducing the number of operations and memory accesses.

The block plays an important role in improving training speed with a small overhead on parameters and FLOPs when applied in the early stages of the network It reduces memory access overhead compared to the original MBConv. However, if all blocks are replaced with Fused MBConv, it can significantly increase parameters and FLOPs while also slowing down the training. Finding the right combination of MBConv and Fused MBConv blocks is

a non-trivial task, which motivates the use of neural architecture search to automatically search for the best combination.

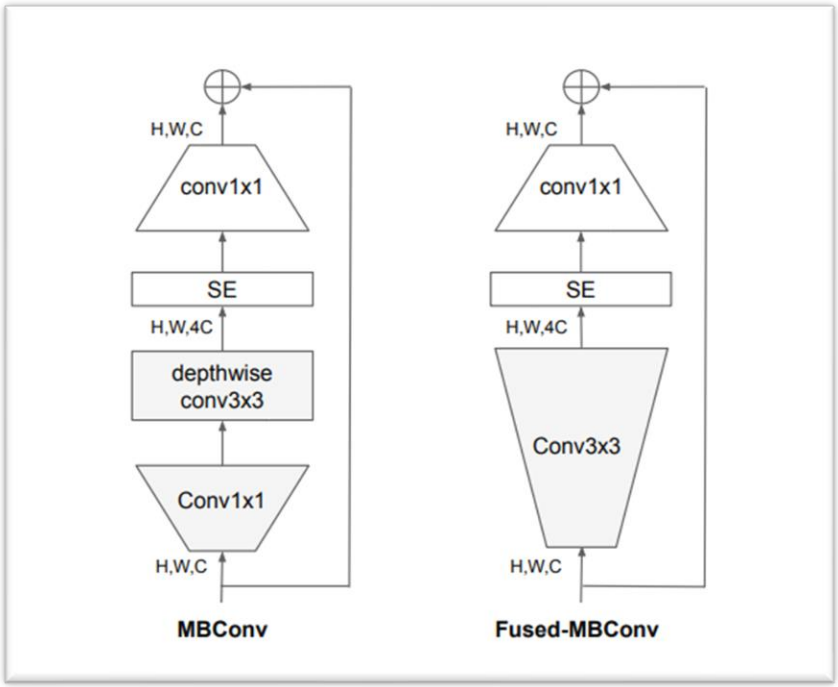

Figure 4-12 MBConv and Fused-MBConv

Image source: (Tan & Le, 2021)

4.3. Experiment Configuration

To ensure the reproducibility of the experiment, performance optimization, comparative analysis, and efficiency, the experimental configuration is recorded. This section discusses various settings such as hyperparameters, model architecture, optimization algorithms, data preprocessing steps, and other factors that influence the training process and the performance of the model.

4.3.1. Hyperparameters

These are Parameters that are set before the learning process begins.

Learning rate: It is one of the critical hyperparameters in the training process of deep learning models. As shown in Figure 4-13 the learning rate controls how big of steps the process is taking during optimization. This is to parameter that controls the model's learning speed. While this high learning rate could mean quick convergence, it also opens up the likelihood of overshooting with respect to the optimal solution and can cause instability. On

the other hand, a low learning rate may mean that convergence is very slow, and there is a good possibility of getting stuck in one of the local minima. As such, one should be very critical when choosing the learning rate to ensure a model performs at its best. The base learning rate was set to 0.0001 for experimental purposes in this study.

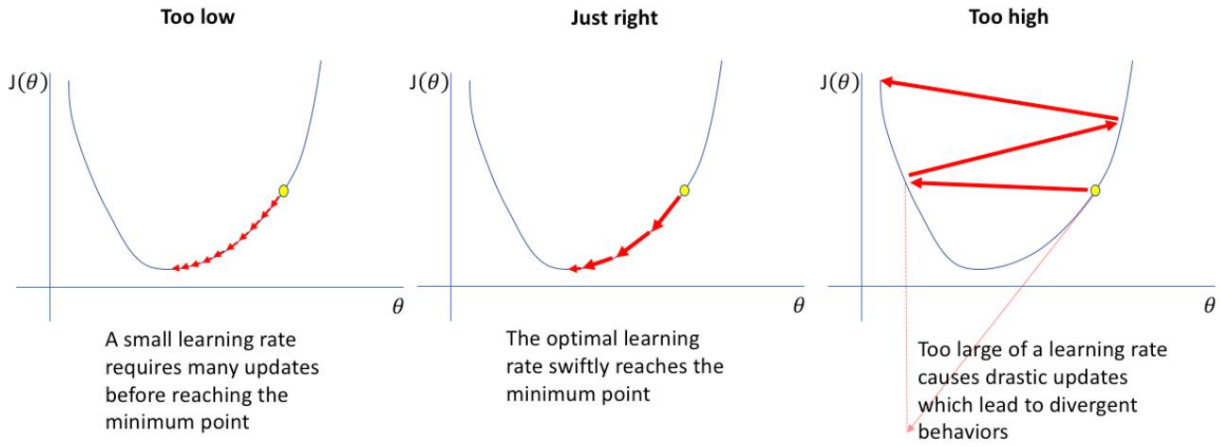

Figure 4-13 learning rate comparison

Image Source: (*Setting the Learning Rate of Your Neural Network.*, n.d.)

Batch size: The size of the batch indicates the quantity of data samples that are handled before adjusting the parameters of the model while it is being trained. Selecting the right batch size is crucial for effective training. Larger batch size can result in quicker training since more samples are processed simultaneously, but it might demand more memory and cause slower convergence. Conversely, a smaller batch size can offer more frequent updates to the model, which could potentially enhance generalization, although it might prolong training time due to the additional processing required for smaller batches. In light of this, the experiment utilized a batch size of 32.

Number of epochs: In deep learning, an epoch refers to the number of iterations the whole dataset propagates forward and backward to update the weights in a neural network. Too few training epochs may lead to underfitting. This comes as a result of the model missing out on the trends inherent in the data. On the other hand, too many epochs will result in overfitting: model simply memorizes the training data and performs terribly on new data. One should monitor the model's performance on a validation set and stop train when signs of overfitting appear. This work used variable epoch sizes against different trainings and fine-tuning of

any experiment. Most of the experiments were done with 10 epochs for training and 15 for fine-tuning.

Optimization Algorithm

Optimization algorithms focus on the minimization or maximization of a loss function, which mathematically defines the variance between predicted and actual outcomes. Thus, the ultimate goal of any optimization algorithm would be to obtain those model parameters that return the least loss or maximum accuracy. There are many optimization algorithms, such as Gradient Descent, Stochastic Gradient Descent, Momentum, Nesterov Accelerated Gradient, and Adam.

Among the many DL model optimizers, Adam's optimization is an obvious choice for deep-learning-model optimization. It allows for adaptive learning rates, robustness to noisy gradients, fast convergence, ease of implementation, a wide range of applicability across model architectures, state-of-the-art performance, and setting a learning rate for each of the parameters. Gradient magnitude helps to be efficient while training complex models with resilience against noisy gradients and requiring only very minimal hyperparameter tuning. Finally, optimization by Adam has returned the best possible result for our model.

$$\theta_{t+1} = \theta_t - \alpha . \frac{\nabla L(\theta_t)}{\sqrt{\epsilon + \mathrm{E}[\nabla 2 L(\theta_t)]}} \quad \ldots (7)$$

$$\alpha_t = \alpha_0 . \sqrt{\frac{V_t}{V_o}} \quad \ldots (8)$$

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot \nabla L(\theta_t) \quad \ldots (9)$$

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot \nabla L(\theta_t)^2 \quad \ldots (10)$$

Equation (7) is the formula used to update the model parameter θ at each $t+1$ iteration. While equation (8) demonstrates the adjustment of the learning rate α_t at each iteration based on the ratio of the current estimate of gradient variance (V_t) to the initial estimate (V_o). This allows Adam to adapt the learning rate for each parameter based on the history of gradients. the exponential moving average of the gradients, which is the first moment (mean) of the gradients is computed with Equation (9). The exponential moving average of the squared gradients, which is the second moment (uncentered variance) of the gradients is calculated with Equation (10).

4.3.2. Loss Function

Loss functions are very important in deep learning because they guide optimization through their provision of force in terms of the minimum disparity between the predicted and real output. The objective of the model will be to minimize the loss function that is going to return optimal parameters, resulting in the least loss. Various forms of loss functions include Mean Squared Error, Mean Absolute Error, Log Loss, and Cross-Entropy.

One of the most frequently applied and efficient loss functions in binary classification (see Eq. (11)), when the values of the target are from the set {0, 1}, is the binary cross-entropy loss. This particular loss function evaluates the mean disparity between the predicted and actual probability distributions for forecasting class 1. It is the favored loss function within the maximum likelihood inference framework and is frequently designated as the default loss function in Keras. The binary cross-entropy loss function is computed as the negative mean logarithm probability of the correct class, which is reduced during the training process.

$$L(y, \hat{Y}) = -[ylog(\hat{Y}) + (1-y)log(1-\hat{Y})] \quad \dots (11)$$

This loss function is particularly useful for binary classification tasks where the model needs to predict the probability of a class given the input features. It is also robust to noisy gradients and can handle imbalanced datasets effectively.

4.3.3. Validation Strategy

Validation metrics are very important in estimating the effectiveness of deep learning models. Out of these, Binary Accuracy, which is defined by Equation 1 shows how accurate the predictions are for binary classification tasks. The Binary Accuracy is calculated by the ratio of the sum of true positive and true negative predictions to the count of total predictions. Binary Accuracy is a simple, very powerful measure that gives an insight into a model's ability to correctly classify each instance into its class. This means much in binary classification tasks, in which it is necessary to predict the likelihood of a class given input features.

4.4. Pruning

There is no doubt that efficientnetv2 significantly reduces the number of parameters compared to the latest deep learning models in recent years. But this parameter reduction can even be improved in a technique called pruning. Pruning (*Pruning Comprehensive Guide | TensorFlow Model Optimization,* n.d.) is a technique used in deep learning to reduce the size of neural networks by removing unnecessary connections or weights. This process makes the model efficient, faster, less computationally intensive, and totally risk-free. It also serves in a very important way toward the optimization of deep-learning models for deployment on resource-constrained devices like mobile phones or even IoT devices. Reductions in model size can often be realized through pruning, which may later accelerate inference times by reducing memory needs and therefore save energy.

Several pruning techniques are available for implementation each with its own advantages and disadvantages. Following are list of pruning techniques.

Weight Pruning: This technique involves setting small weights in the neural network to zero or removing them entirely. Weight pruning can be done based on a threshold value, where weights below a certain threshold are pruned. Figure 4-14 shows a brief comparison of a sample architecture before and after pruning.

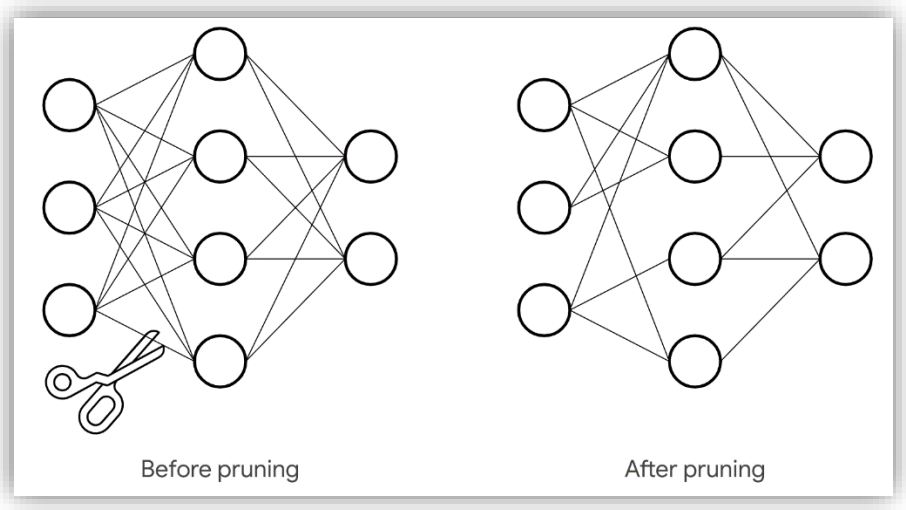

Figure 4-14 Weight pruning

Image source: (*TensorFlow Model Optimization Toolkit — Pruning API — The TensorFlow Blog*, n.d.)

Unit/Neuron Pruning: In unit pruning (see Figure 4-15), entire neurons or units in a layer are removed based on their importance. This technique helps in reducing the model size significantly.

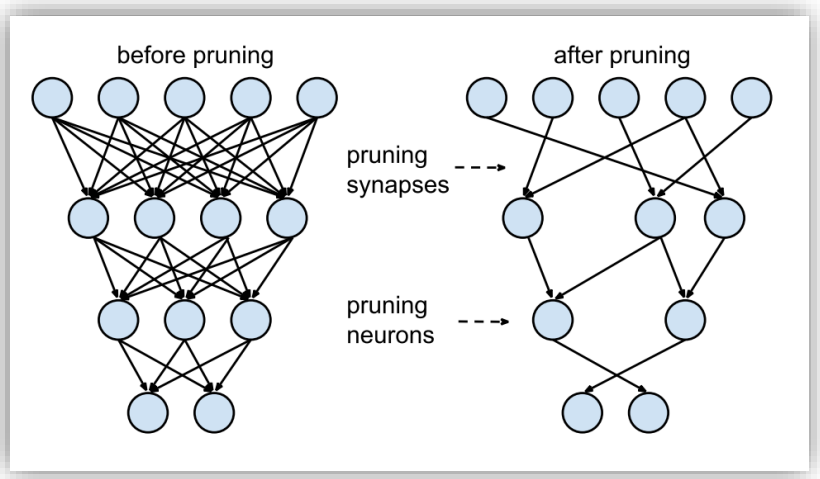

Figure 4-15 Neuron Pruning

Image source: (Han et al., 2015)

Filter Pruning: Filter pruning (see Figure 4-16) involves removing entire convolutional filters from the network. Filters with low importance are pruned based on criteria like the L1 or L2 norm of the filter weights.

Structured Pruning: Structured pruning (see Figure 4-17 involves removing entire channels, layers, or blocks of neurons instead of individual weights. This technique helps in maintaining the network's structure and can be more effective in certain architectures.

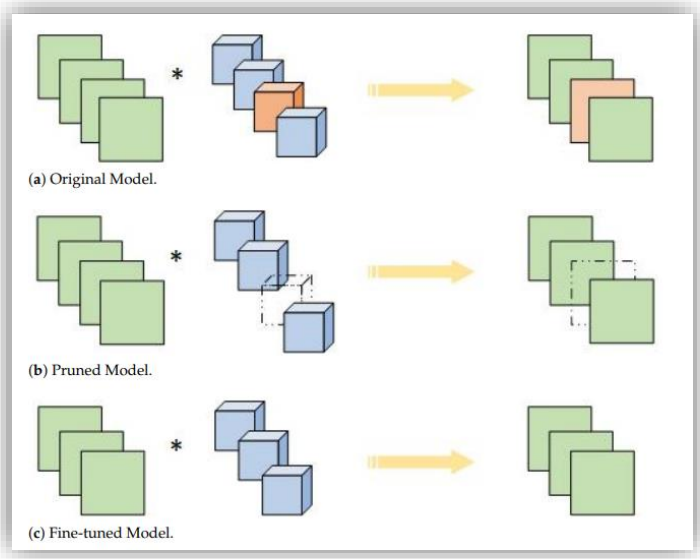

Figure 4-16 Filter Pruning

Image source: (Shao et al., 2021)

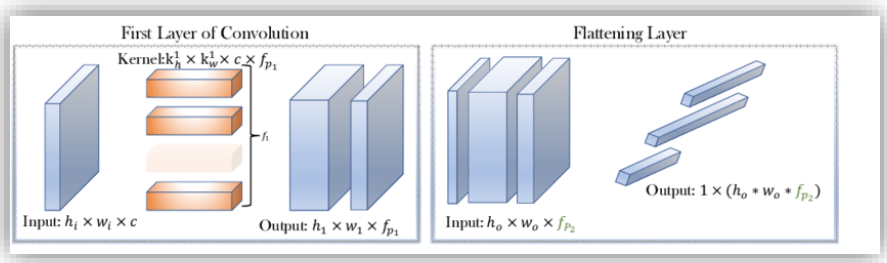

Figure 4-17 channel pruning

Image source: (Venkatesan et al., 2020)

While pruning the model, we may use a variety of algorithms to prune the model which may include Magnitude-based Pruning, L1/L2 Regularization, and Optimal Brain Damage/Lei Pruning. These pruning algorithms combined with pruning techniques have advantages for model compression, faster inference time, energy efficiency, and improved generalization by simplifying the model.

In this proposed method we have implemented the weight pruning as it is particularly advantageous over the rest. The primary benefit of weight pruning is its flexibility and fine-

grained control over the pruning process. By removing individual weights based on their importance, weight pruning can achieve significant model compression without drastically altering the network architecture. This allows for more precise tuning of the model size and performance trade-off, as weights with the least impact on the overall accuracy can be selectively removed. Additionally, weight pruning is generally easier to implement and can be applied to various neural network architectures without requiring major modifications.

CHAPTER 5

Results and Discussions

5.1. Experiments

The EfficientNet approach to classifying brain tumors, as proposed in this research, involves a series of consecutive experiments designed to test and refine the model's performance. These experiments are crucial in ensuring the model's efficacy and robustness in various scenarios. This section provides an in-depth analysis of the consecutive experiments conducted, highlighting the methodologies employed, results obtained, and insights gained.

Experiment 1

At the very early stage, a deep learning model with transfer learning (see Figure 5-1 for transfer learning code snippet) was developed using EfficientNet V-2S to classify brain MRI scans as either healthy or containing a tumor. The first experiment focused on selecting the most suitable EfficientNet variant for brain tumor detection. This involved training and evaluating multiple EfficientNet models, including EfficientNetB0 to EfficientNetB2, and EfficientNet V-2S on the provided dataset. Due to limited computational costs, the experiment couldn't be extended to the whole variants of EfficientNet. The results indicated that EfficientNet V-2S achieved 55.36% validation accuracy and the remaining models scored even below 55%. this is because the model is initially trained on an ImageNet dataset that has 10 classes.

```
IMG_SHAPE = IMG_SIZE + (3,)
base_model = tf.keras.applications.EfficientNetV2S(input_shape=IMG_SHAPE,
                                                   include_top=False,
                                                   weights='imagenet')
```

Figure 5-1 Transfer learning code snippet

Experiment 2

The results (Checkpoints are loaded) from experiment 1 were taken and trained the head (see Figure 5-2 for custom head layers) of the model with a brain tumor data set which has been preprocessed while keeping layers of the baseline model frozen.

"base_model.trainable = False"

```
inputs = tf.keras.Input(shape=(224, 224, 3))
x = data_augmentation(inputs)
#x = preprocess_input(x)
x = base_model(x, training=False)
x = global_average_layer(x)
x = tf.keras.layers.Dropout(0.2)(x)
outputs = prediction_layer(x)
model = tf.keras.Model(inputs, outputs)
```

Figure 5-2 Custom Head for the existing model

The number of trainable variables at this stage was "2" while the total number of trainable parameters in this model is "1281". A summary of the baseline model with a custom head layer in presented in Figure 5-3.

```
model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 224, 224, 3)]     0

 sequential (Sequential)     (None, 224, 224, 3)       0

 efficientnetv2-s (Function  (None, 7, 7, 1280)        20331360
 al)

 global_average_pooling2d (  (None, 1280)              0
 GlobalAveragePooling2D)

 dropout (Dropout)           (None, 1280)              0

 dense (Dense)               (None, 1)                 1281

=================================================================
Total params: 20332641 (77.56 MB)
Trainable params: 1281 (5.00 KB)
Non-trainable params: 20331360 (77.56 MB)
_________________________________________________________________
```

Figure 5-3 Model summary of the proposed architecture

With base learning rate = 0.0001 and 10 epochs the model managed to achieve 88.29 validation accuracy (see Fig 5-4 for accuracy and loss trend).

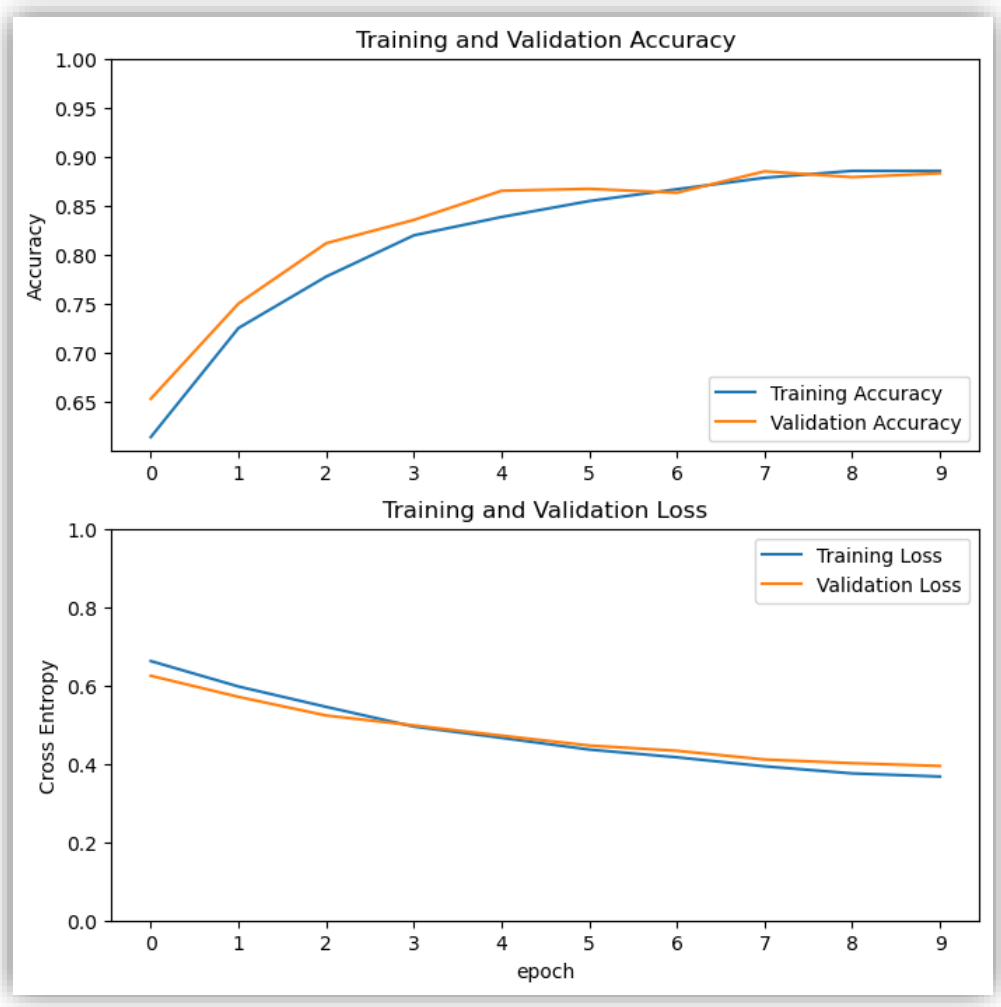

Figure 5-4 Accuracy and loss graph: EfficientNet v2s trained with a custom head layer

Experiment 3

Training the head layers stated in Figure 5-2 only with the brain tumor dataset, is not enough to get maximum efficiency so frozen layers after the 100th layer in the base model are set to trainable and finetuned the model for better performance.

```
base_model.trainable = True

print("Number of layers in the base model: ", len(base_model.layers))

# Fine-tune from this layer onwards
fine_tune_at = 100

# Freeze all the layers before the `fine_tune_at` layer
for layer in base_model.layers[:fine_tune_at]:
  layer.trainable = False
```

```
Number of layers in the base model:  513
```

Figure 5-5 Unfreezing layers for training code snippet

As shown in Figure 5-5, the number of trainable layers, variables, and parameters are significantly increased and the pre-trained weights and biases are finetuned by the new detection task.

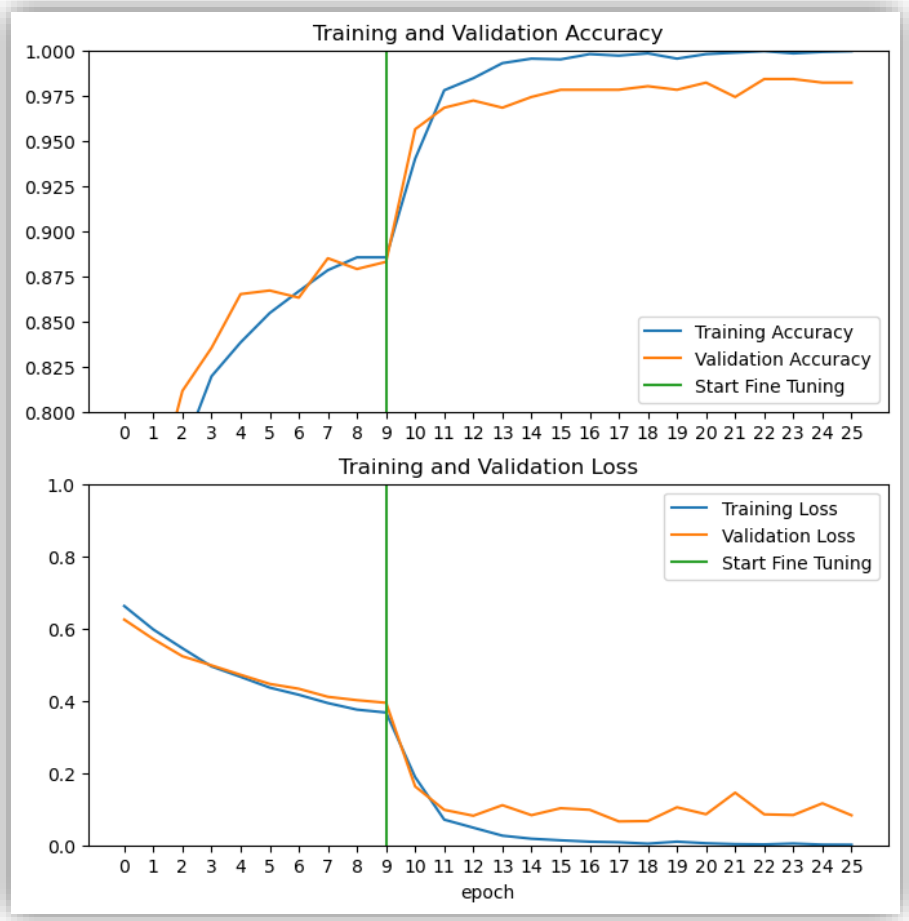

Figure 5-6 Accuracy and loss graph: EfficientNet v2s trained with custom head layer and fine-tuned the last 413 layers

The model was then trained for 15 epochs and achieved 98.95 validation accuracy. Figure 5-6 shows the progress of experiments 2 and 3 combined where the accuracy of an EfficientNet V2S model is radically increased after finetuning and the validation loss is decreased to the least value.

Experiment 4

This experiment is specifically related to the contribution of this research work. Figure 5-6 is not satisfactorily smooth which mainly implies 4 things these are:

Low Learning Rate: The learning rate might be too high. If the learning rate is set very high, then the model might be taking too large steps during optimization, causing it to overshoot and thus fluctuate a lot. Reducing the learning rate might help the model to converge more smoothly. But the initial learning rate is 0.000, lowering this value might have led the graph to converge at local minima.

Low Batch Size: A small batch size might be the cause for the fluctuations, as the estimates of the gradients become noisier with lower batch sizes. Increasing the batch size can lead to more accurate gradient estimates and hence smoother curves. 32 as batch number is considered a starting for training and is also optimal for computational cost. So we narrowed the possibilities of the cases presented below.

High Data Variance: Large fluctuations could indicate high variance in your training data. You could inspect the dataset to ensure there's a good variety of data, but not too much that could make it difficult for the model to learn underlying patterns. Although the dataset used in this paper is carefully curated, there might be too much similarity between the data instances so we added other MRI images from other sources and trained the model once again.

Overfitting: If your model is too complex, it might be overfitting to some extent, causing the mAP to fluctuate because it's not generalizing well to the validation data. You might want to experiment with simpler models or regularization techniques to check if overfitting is the issue. This case is ignored because the dataset and features to be extracted from the images are not that simple where the model doesn’t fit them.

To overcome the low variance and maintain a good balance between variance and bias we might follow a possible solution including merging two different datasets. Combining datasets from different sources, environments, or populations can lead to increased variance

due to differences in data distributions, noise levels, or other factors as it increases data heterogeneity of the data. This can result in overfitting if the model is too complex or if the datasets are not properly aligned

Before diving into how datasets are combined, one needs to understand what is bias-variance trade-off. Bias is the error coming from using a model that is too simple to explain how the real world works. While variance captures how much the prediction for a given datapoint would change if a different model were used. A high-bias model is too simple: it misses the underlying pattern of trends in training data, causing underfitting. On the other hand, a high-variance model is too complex: it fits too well against the training data but does not generalize very well to new, unseen data—overfitting.

Such merging can be a very powerful way to gain the strengths of both sets and, by doing so, improve model performance. It is, however, of great importance to strike a balance between the variance and the bias for it to enable good generalization. Some of the techniques to achieve this include data augmentation, regularization, model selection, and ensemble techniques. Such models, performing well with both the training and test data, could be built according to these techniques and the mentioned requirements of the problem.

Since this is an experimental procedure

To increase the data variance and bias balance both datasets mentioned above are used with the following dataset distribution (see Figure 5-7). The testing dataset is retrieved from the validation dataset which is 20% of the validation dataset.

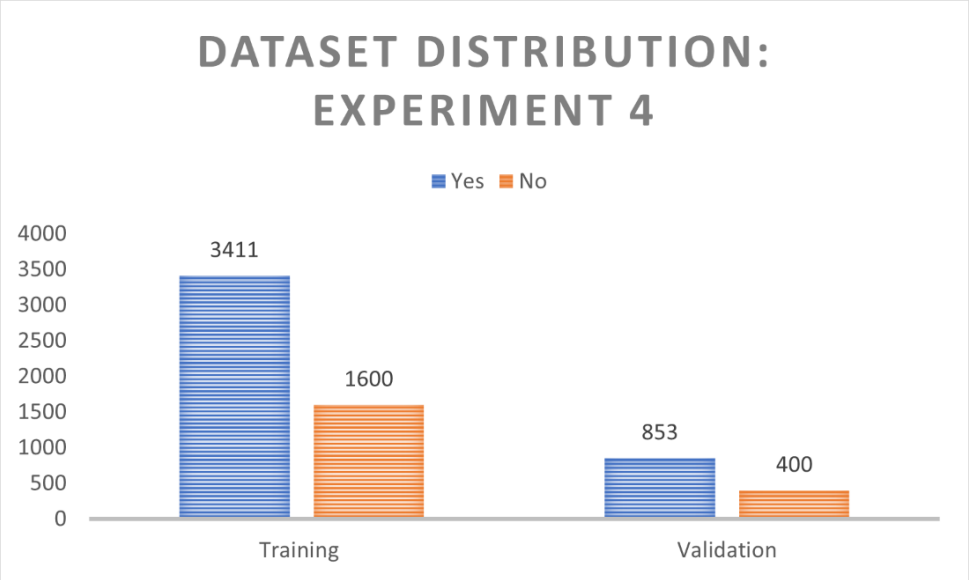

Figure 5-7 Dataset distribution after merging datasets

The model is re-trained with the same procedure but this time it is finetuned with a pruning technique to reduce parameter size and improve parameter efficiency.

```
model_for_pruning = tf.keras.models.clone_model(
    model,
    clone_function=apply_pruning_to_layer,
)
```

```
pruning_params = {
    'pruning_schedule': tfmot.sparsity.keras.PolynomialDecay(initial_sparsity=0.2,
        final_sparsity=0.8, begin_step=150, end_step=600),
    'block_size': (1, 1),
    'block_pooling_type': 'AVG'
}
```

```
model_for_pruning = tf.keras.models.clone_model(
    model,
    clone_function=apply_pruning_to_layer,
)
```

Figure 5-8 Code Snippets: Pruning

As shown in Figure 5-8, the layers in the baseline model are assigned to a new variable (cloned) but while doing this, the final layers with significantly more parameters are pruned to make the layers sparse. The pruning parameter decides the schedule which is the method that initially starts by sparsifying the layer parameter to 20% and increases to a maximum of 80%.

After this stage not only does the model graph get smothered but also the accuracy is increased to 99.6093 and the total model parameter is decreased from 20,332,641 to 20,331,360 (see Figure 5-9 and Figure 5-10 for accuracy and loss trend of the proposed model respectively).

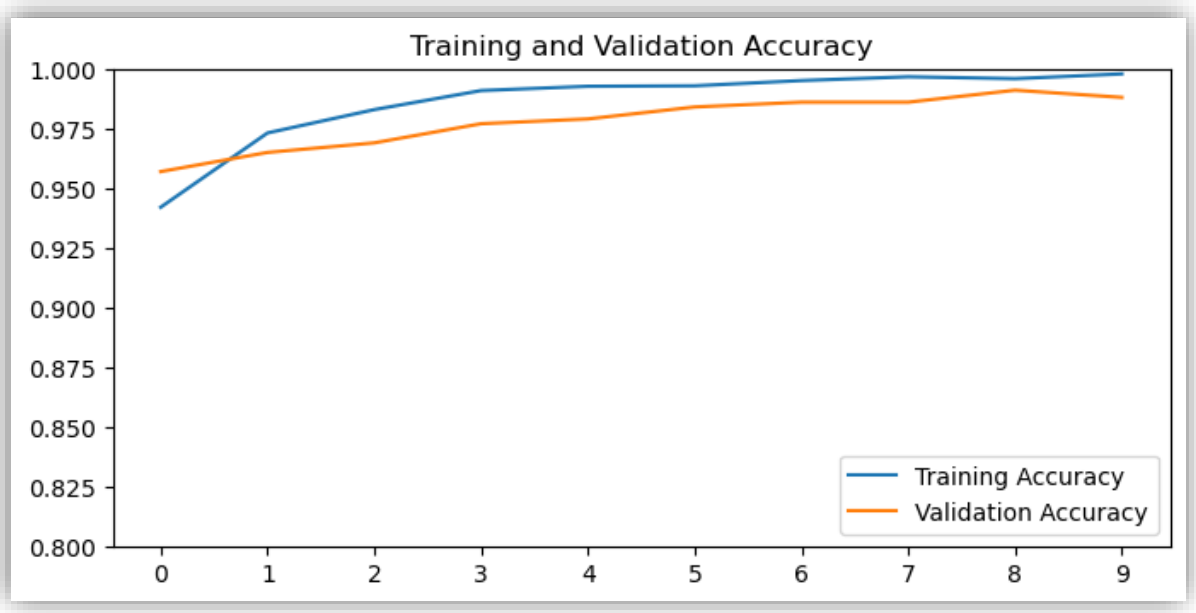

Figure 5-9 Training and validation accuracy: EfficientNet v2s after pruning

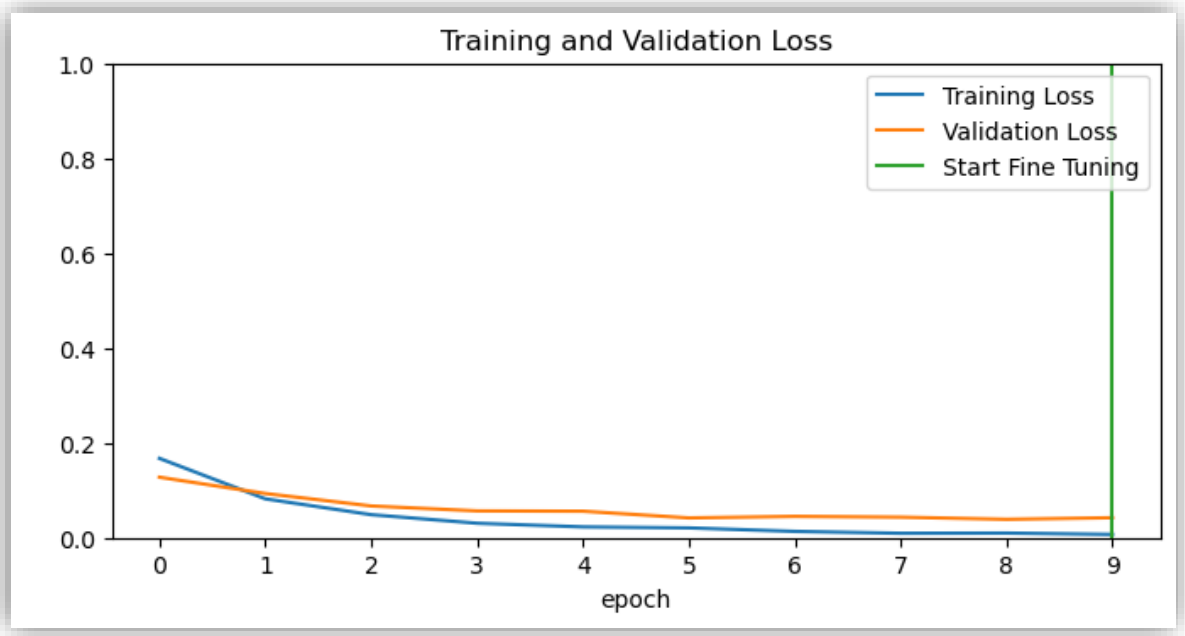

Figure 5-10 Training and validation loss: EfficientNet v2s after pruning

5.2. Mode Evaluation and Comparison

```
Precision: 0.9583333333333334
Recall: 1.0
F1 Score: 0.9787234042553191
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.89      0.94         9
           1       0.96      1.00      0.98        23

    accuracy                           0.97        32
   macro avg       0.98      0.94      0.96        32
weighted avg       0.97      0.97      0.97        32

Confusion Matrix:
 [[ 8  1]
 [ 0 23]]
Predictions:
 [1 1 0 1 0 0 0 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 0 0 1]
Labels:
 [1 1 0 1 0 0 0 1 1 1 1 1 1 1 1 1 0 1 1 0 0 1 1 1 1 1 1 1 0 0 1]
```

Figure 5-11 Evaluation of the proposed model

From Figure 5-11, it turned in an overall accuracy of 99.60%, a very high score, indicating that the model correctly classifies most of the samples. Another metric was the F1 score, which is a harmonic mean of precision and recall and was very high at 97.87%. This indicates a good balance between precision and recall of the model, making it a reliable classifier.

This model precision of 95.83% implies that each time it predicts the positive class, it is correct 95.83% of the time. This is quite a very strong result, stating that it is possible for the model to minimize false positives to near nil, which is important in medical applications where false alarms may mean serious trouble.

A recall of 100% returns us to the fact that the model missed none of the positive samples. This is an excellent result in proving the ability of the model to capture all relevant cases—one very important aspect to ensure completeness in diagnosis.

The classification reports return the performance of the model for each class. This gave a precision of 100%, a recall of 89%, and an F1 score of 94% for class 0, and a precision of 96%, a recall of 100%, and an F1 score of 98% for class 1.

The high performance for both classes proves that the model will input the correct differentiation among various types or stages of brain tumors. Probably, it has misclassified a few from class 0 as class 1, since its recall for class 0 is slightly lower; otherwise, the class-wise metrics are very strong.

It gives a clear indication of how well the model is performing with respect to classification. There, it shows that the model correctly classified 9 samples as class 0 and 23 samples as class 1, with only 1 false negative, class 1 predicted as class 0. This corroborates further the high accuracy and low misclassification of the model.

One may wonder how the pruning effect increases the overall accuracy, and this is because by selectively removing redundant or less important parameters from the model, pruning optimizes the network's architecture, allowing it to focus on the most relevant features and patterns in the data.

The proposed method's classification performance is contrasted with current cutting-edge architectures. Specifically, MobileNetv3(Howard et al., 2019), DenseNet169(Huang et al., 2016), VGG19(Simonyan & Zisserman, 2014), ResNet152(He et al., 2015), and the baseline model EfficientNet V2-S(Tan & Le, 2021). The metrics used to compare among architectures are accuracy and loss. Table 5-1, shows a summary of the model performance of previous models and the proposed model. Our proposed approach stands out to be the first to achieve the highest classification accuracy and lowest validation loss.

Table 5-1 Result comparison among different models

Architecture	**Training Phase**		**Testing Phase**	
	Accuracy	**Loss**	**Accuracy**	**Loss**
MobileNetv3	99.75	0.0359	98.52	0.1272
DenseNet169	99.22	0.0241	97.53	0.958
VGG19	99.07	0.0451	95.62	0.1245
ResNet152	98.86	0.0603	96.92	0.1854
EfficientNet V2-S	90.08	0.2768	91.80	0.2527
EfficientNet V2-S with pruning	**99.78**	**0.0070**	**99.61**	**0.0089**

5.3. Research question discussion

Answer for RQ1: As shown in the above sections the EfficientNet-V2-S model outperformed various CNN models not only by achieving maximum accuracy but also low computational complexity. Being created by NAS, the EfficientNet V2-S Model overcomes most of the gaps stated in the problem statement section.

Answer for RQ2: Yes indeed! The performance of a deep learning model can be improved with minor computational costs and a small number of parameters. By using the pruning technique less important weights get trimmed to make the training speed faster and score better accuracy

Answer for RQ3: With random flipping and rotation with 2 radians, the number of artificial data instances has increased and so the generalizability of the model.

Answer for RQ4: As shown in experiment 4 our proposed model achieved 99.60 with 20,332,641 parameters which beats previous models done to classify brain tumors.

CONCLUSIONS AND RECOMMENDATIONS

Conclusion

This work hereby concludes with the successful demonstration of EfficientNet in classifying brain tumors. In that respect, the proposed model created only a synergistic effect of EfficientNet V-2 architectures with transfer learning, pruning, and fine-tuning techniques to achieve state-of-the-art accuracy in 99.60% of brain tumor detection tasks. Some of the key findings of the study are that first, the fine-tuned EfficientNet V2-S architecture on a brain tumor dataset outperformed these effective models of CNNs: EfficientNet, VGG16, ResNet50, InceptionV3 along all evaluation matrices accuracy, precision, recall, and F1-score. Second, transfer learning from pre-trained EfficientNet models on the ImageNet dataset instead of training from scratch improves the model in terms of minimizing training time and the requirement for large datasets. Third, the overall accuracy achieved by the proposed approach was 99.6%, with high precision and recall for each class of brain tumors (Healthy and Unhealthy), thus being robust and reliable. In conclusion, with regard to efficiency and fast inference time, this model is predisposed to real-time applications and further integration into clinical decision support systems.

Recommendation

Based on these findings, a Novel EfficientNet-based Approach for Brain Tumor Detection is hereby recommended for adoption. In this research we propose to extend the scope of the study in the future with the following ideas. Further testing would be needed on more extended and diverse datasets of brain tumors, not only for the generalization and robustness of the model but also for its integration into the currently existing medical imaging workflows, with an evaluation of the performance in a real-world clinical setting. Moreover, investigating the model for classifying other brain lesions or abnormalities beyond tumors could hugely enhance its applications. Stable collaboration between clinicians and researchers is still required for the refinement of the model for clinical application and its effective translation into medical practice. If all the recommendations related to the described EfficientNet approach were followed, it would unmistakably promote the field of brain tumor detection, eventually improving the quality of care and outcomes for the patients.

REFERENCES

Abdusalomov, A. B., Mukhiddinov, M., & Whangbo, T. K. (2023). Brain Tumor Detection Based on Deep Learning Approaches and Magnetic Resonance Imaging. *Cancers*, *15*(16). https://doi.org/10.3390/CANCERS15164172

Admasu, F. T., Demissie, B., Yitbarek, G. Y., Geto, Z., Tesfaw, A., Zewde, E. A., Tilahun, A., Walle, G., Bekele, T. T., Habte, M. L., Feyisa, T. O., Amare, T. J., Alebachew, W., Asnakew, S., Sisay, E., Tiruneh, M., Yemata, G. A., Aytenew, T. M., & Dejenie, T. A. (2022). Evaluation of total oxidative stress and antioxidant capacity of brain tumour patients attending referral hospitals in Addis Ababa, 2020: a comparative cross-sectional study. *Ecancermedicalscience*, *16*. https://doi.org/10.3332/ECANCER.2022.1391

Br35H :: Brain Tumor Detection 2020. (n.d.). Retrieved May 27, 2024, from https://www.kaggle.com/datasets/ahmedhamada0/brain-tumor-detection

Brain tumor - Symptoms and causes - Mayo Clinic. (n.d.). Retrieved August 21, 2023, from https://www.mayoclinic.org/diseases-conditions/brain-tumor/symptoms-causes/syc-20350084

Brain Tumor Classification (MRI). (n.d.). Retrieved May 27, 2024, from https://www.kaggle.com/datasets/sartajbhuvaji/brain-tumor-classification-mri

Brain Tumor: Introduction | Cancer.Net. (n.d.). Retrieved August 21, 2023, from https://www.cancer.net/cancer-types/brain-tumor/introduction

Brain Tumor Overview - Harvard Health. (n.d.). Retrieved August 21, 2023, from https://www.health.harvard.edu/a_to_z/brain-tumor-overview-a-to-z

Brain tumors: overview of types, diagnosis, treatment options | Cincinnati, OH Mayfield Brain & Spine. (n.d.). Retrieved August 21, 2023, from https://mayfieldclinic.com/pe-braintumor.htm

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2020). *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. http://arxiv.org/abs/2010.11929

Fan, Y., Zhang, X., Gao, C., Jiang, S., Wu, H., Liu, Z., & Dou, T. (2022). Burden and trends of brain and central nervous system cancer from 1990 to 2019 at the global, regional, and country levels. *Archives of Public Health*, *80*(1), 1–14. https://doi.org/10.1186/S13690-022-00965-5/FIGURES/6

Gaur, L., Bhandari, M., Razdan, T., Mallik, S., & Zhao, Z. (2022). Explanation-Driven Deep Learning Model for Prediction of Brain Tumour Status Using MRI Image Data. *Frontiers in Genetics*, *13*. https://doi.org/10.3389/fgene.2022.822666

Han, S., Pool, J., Tran, J., & Dally, W. J. (2015). Learning both Weights and Connections for Efficient Neural Networks. *Advances in Neural Information Processing Systems*, *2015-January*, 1135–1143. https://arxiv.org/abs/1506.02626v3

Hassaballah, M., Singh, S., Kumar, P., Ullah Khan, H., Abbas, S., & Sha, M. (n.d.). *Ensemble deep learning for brain tumor detection*.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016-December*, 770–778. https://doi.org/10.1109/CVPR.2016.90

Howard, A., Sandler, M., Chen, B., Wang, W., Chen, L. C., Tan, M., Chu, G., Vasudevan, V., Zhu, Y., Pang, R., Le, Q., & Adam, H. (2019). Searching for MobileNetV3. *Proceedings of the IEEE International Conference on Computer Vision*, *2019-October*, 1314–1324. https://doi.org/10.1109/ICCV.2019.00140

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2016). Densely Connected Convolutional Networks. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-January*, 2261–2269. https://doi.org/10.1109/CVPR.2017.243

Khan, M. S. I., Rahman, A., Debnath, T., Karim, M. R., Nasir, M. K., Band, S. S., Mosavi, A., & Dehzangi, I. (2022). Accurate brain tumor detection using deep convolutional neural network. *Computational and Structural Biotechnology Journal*, *20*, 4733–4745. https://doi.org/10.1016/j.csbj.2022.08.039

Mahmud, M. I., Mamun, M., & Abdelgawad, A. (2023). A Deep Analysis of Brain Tumor Detection from MR Images Using Deep Learning Networks. *Algorithms*, *16*(4). https://doi.org/10.3390/a16040176

Memirie, S. T., Habtemariam, M. K., Asefa, M., Deressa, B. T., Abayneh, G., Tsegaye, B., Abraha, M. W., Ababi, G., Jemal, A., Rebbeck, T. R., & Verguet, S. (2018). Estimates of Cancer Incidence in Ethiopia in 2015 Using Population-Based Registry Data. *Journal of Global Oncology*, *4*(4), 1–11. https://doi.org/10.1200/JGO.17.00175

Meseret Assefa. (n.d.). *HISTOPATHOLOGICAL PATTERN OF CENTRAL NERVOUS SYSTEM TUMORS: A 5- YEAR RETROSPECTIVE STUDY, AT A TERTIARY HOSPITAL IN ETHIOPIA*.

Pruning comprehensive guide | TensorFlow Model Optimization. (n.d.). Retrieved May 27, 2024, from https://www.tensorflow.org/model_optimization/guide/pruning/comprehensive_guide

Sailunaz, K., Bestepe, D., Alhajj, S., Özyer, T., Rokne, J., & Reda Alhajj. (2023). Brain tumor detection and segmentation: Interactive framework with a visual interface and feedback facility for dynamically improved accuracy and trust. *PLoS ONE*, *18*(4 April). https://doi.org/10.1371/journal.pone.0284418

Setting the learning rate of your neural network. (n.d.). Retrieved June 20, 2024, from https://www.jeremyjordan.me/nn-learning-rate/

Shao, L., Zuo, H., Zhang, J., Xu, Z., Yao, J., Wang, Z., & Li, H. (2021). Filter Pruning via Measuring Feature Map Information. *Sensors 2021, Vol. 21, Page 6601*, *21*(19), 6601. https://doi.org/10.3390/S21196601

Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. https://arxiv.org/abs/1409.1556v6

Sunil, C. K., Jaidhar, C. D., & Patil, N. (2022). Cardamom Plant Disease Detection Approach Using EfficientNetV2. *IEEE Access*, *10*, 789–804. https://doi.org/10.1109/ACCESS.2021.3138920

Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., & Le, Q. V. (2018). *MnasNet: Platform-Aware Neural Architecture Search for Mobile*. http://arxiv.org/abs/1807.11626

Tan, M., & Le, Q. V. (2019). *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. http://arxiv.org/abs/1905.11946

Tan, M., & Le, Q. V. (2021). *EfficientNetV2: Smaller Models and Faster Training*. http://arxiv.org/abs/2104.00298

TensorFlow Model Optimization Toolkit — Pruning API — The TensorFlow Blog. (n.d.). Retrieved June 20, 2024, from https://blog.tensorflow.org/2019/05/tf-model-optimization-toolkit-pruning-API.html

The Algorithms — Entheos A.I. Art. (n.d.). Retrieved June 20, 2024, from https://www.entheosart.com/the-algorithms/

The Most Common Brain Tumor: 5 Things You Should Know | Johns Hopkins Medicine. (n.d.). Retrieved June 20, 2024, from https://www.hopkinsmedicine.org/health/wellness-and-prevention/the-most-common-brain-tumor-5-things-you-should-know

Venkatesan, R., Swaminathan, G., Zhou, X., & Luo, A. (2020). *Out-of-the-box channel pruned networks*. http://arxiv.org/abs/2004.14584

What are Convolutional Neural Networks? | IBM. (n.d.). Retrieved August 21, 2023, from https://www.ibm.com/topics/convolutional-neural-networks

Ye, Y., Zhou, H., Yu, H., Hu, H., Zhang, G., Hu, J., & He, T. (2022). An Improved EfficientNetV2 Model Based on Visual Attention Mechanism: Application to Identification of Cassava Disease. *Computational Intelligence and Neuroscience*, *2022*. https://doi.org/10.1155/2022/1569911

Zoph, B., & Le, Q. V. (2016). *Neural Architecture Search with Reinforcement Learning*. http://arxiv.org/abs/1611.01578

Zoph, B., Vasudevan, V., Shlens, J., & Le, Q. V. (2017). *Learning Transferable Architectures for Scalable Image Recognition*. http://arxiv.org/abs/1707.07012