Use of Deep Learning/ML Algorithms for the Classification of Brain Tumor

Project report submitted in partial fulfilment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering

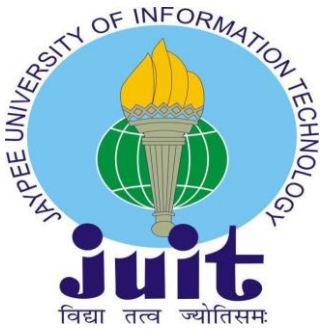
By

Prince Nag (191257)

Under the supervision of

Dr. Yugal Kumar

to



Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

CERTIFICATE

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this report entitled **"Use of Deep Learning/ML Algorithms for the Classification of Brain Tumor"** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from August 2022 to May 2023 under the supervision of **Dr. Yugal Kumar** (Associate Professor, Department of CSE, Jaypee University of Information Technology, Waknaghat). I also authenticate that I have carried out the above-mentioned project work under the proficiency stream **Machine Learning.** The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature)

Prince Nag

191257

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature)

Dr. Yugal Kumar

Associate Professor

Computer Science & Engineering

Dated: May 01,2023

PLAGIARISM CERTIFICATE

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: …….

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report | Pape

Name:_____**Department:**_____**Enrolment No** _____

Contact No._____**E-mail.**_____

Name of the Supervisor: _____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____

_____

_____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at……(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

<table>
<tr><td>Copy Received on</td><td rowspan="4">☐ All Preliminary Pages
☐ Bibliography/ Ima ges/Quotes
☐ 14 Words String</td><td rowspan="2"></td><td>Word Counts</td><td></td></tr>
<tr><td>Report Generated on</td><td>Character Counts</td><td></td></tr>
<tr><td rowspan="2"></td><td>Submission ID</td><td>Total Pages Scanned</td><td></td></tr>
<tr><td></td><td>File Size</td><td></td></tr>
</table>

Checked by
Name & Signature **Librarian**

……

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

ACKNOWLEDGEMENT

All compliments and praise are due to God who empowered me with strength and sense of devotion to successfully accomplish this project work successfully.

I am also deeply thankful to my supervisor, **Dr. Yugal Kumar**, an associate professor at the Department of CSE, Jaypee University of Information Technology, Waknaghat, for his extensive knowledge and interest in the field of Machine Learning, which made this project possible. His unwavering patience, scholarly guidance, continual encouragement, constant supervision, constructive criticism, valuable advice, and meticulous reading of many inferior drafts have been invaluable to me.

I am also grateful to **Sh. Mohan Sharma** and **Sh. Ravi Raina** of the Department of CSE for their kind assistance in completing my project.

I extend my heartfelt thanks to all those who have directly or indirectly contributed to the success of this project. I would like to acknowledge the many staff members, both teaching and non-teaching, who have provided timely assistance and facilitated my project.

Finally, I would like to express my sincere appreciation to my parents for their unwavering support and patience throughout this journey.

Prince Nag
191257

TABLE OF CONTENT

LIST OF ABBREVIATIONS

1. AI ..... Artificial Intelligence
2. CNN ..... Convolutional Neural Network
3. ML ..... Machine Learning
4. ReLU ..... Rectified Linear Unit
5. FCN ..... Fully Convolutional Network
6. MRI ..... Magnetic Resonance Imaging
7. ANN ..... Artificial Neural Network
8. Tanh ..... Hyperbolic Tangent
9. MLP ..... Multi Layer Perceptron
10. MSE ..... Mean Squared Error
11. SGD ..... Stochastic Gradient Descent
12. ResNet ..... Residual Network

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

The quality of life and life expectancy of patients can be significantly impacted by brain tumours, which are a serious health concern. According to the World Health Organisation, cancer is the second most common cause of death in the world, accounting for 22% of all chronic illnesses and causing an estimated 9 million deaths annually. Different types of brain cells can give rise to brain tumours, and the location, size, and grade of the tumour can affect the symptoms. The diagnosis of brain tumours can be difficult, and the available treatments may vary depending on the patient's age, general health, and the grade and location of the tumour.

Surgery, radiation therapy, and chemotherapy are all options for treating brain tumours, but the success rate of these treatments varies, and the prognosis for patients with high-grade brain tumours can be bleak. To detect tumours in various parts of the body, imaging techniques such as CT, MRI, and ultrasound are used. However, MRI is the most widely used technique for diagnosing brain tumours, and the large amount of data generated by MRI scans makes accurate and timely tumour identification difficult.

Convolutional neural networks (CNNs) have been developed in recent years to automate brain tumour detection, increasing its precision and effectiveness. A deep learning neural network called a CNN can recognise patterns in images and predict outcomes with accuracy. With the development of automated brain tumour detection using CNNs, there is hope for increasing the precision and effectiveness of tumour detection, which will ultimately result in better patient outcomes in terms of treatment and survival rates.

Further research in this area is crucial for improving outcomes for those affected by brain tumors. Continued advancements in automated brain tumor detection can help clinicians and researchers better understand the disease, improve treatment options, and ultimately increase survival rates for patient.

CHAPTER 1
INTRODUCTION

1.1 Overview

Classification of "Brain Tumors" in early stage is crucial for better patient outcomes, and in order to accomplish this, medical imagery is an essential tool. MRI is widely used to examine brain abnormalities, and it has become a leading imaging technique for detecting brain tumors. The automation of brain tumor detection from MRI scans is essential for efficiently managing large amounts of medical data and aiding healthcare professionals in identifying tumors and developing effective treatment plans for patients. Recent studies demonstrate that deep learning algorithms, such as CNNs and Transfer Learning techniques, can significantly improves the accuracy of brain tumor classification. Figure 1.1 shows that a basic brain tumor image typically displays a mass or abnormal growth within the brain tissue. Depending on the imaging technique used, such as MRI or CT scans, the tumor may be highlighted with different shades or colors to differentiate it from the surrounding healthy tissue.

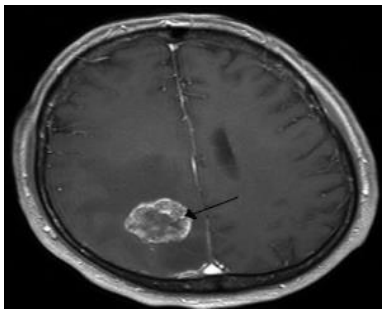


Figure 1.1: Basic Tumor Image

MRI, Magnetic resonance imaging is a popular non-invasive method for generating high-quality brain images that can be processed using algorithms. Advancements in technology have made image processing algorithms more efficient, allowing researchers to develop deep learning algorithms that can quickly and accurately identify cancerous cells in brain images. Despite limitations, such as image quality and interpretation of results, deep learning has great potential to improve brain tumor classification and treatment.

1.2 Motivation for the work

Brain tumors are a serious health issue that, if not identified and treated effectively, can be fatal. The detection and classification of brain tumors through medical image analysis is a time-consuming process that heavily relies on human judgment. Specialists examine various imaging scans and make decisions that inform treatment options. With the help of powerful deep learning algorithms and advanced technology, automated systems can help to identify and classify tumors quickly and accurately, potentially improving patient outcomes and saving valuable time and resources. Figure 1.2 highlights the alarming number of brain cancer deaths in India from 2010 to 2020, with figures surpassing other types of cancer. Early detection of brain tumors can help reduce mortality rates.

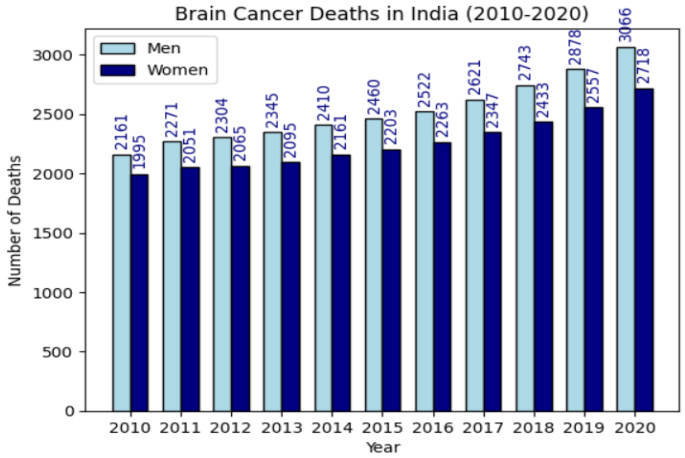


Figure 1.2: Brain Cancer Deaths in India (2010-2020) [11]

The automation of medical image analysis has gained significant attention in India, just like in developed countries. Thanks to advances in technology and the availability of powerful deep learning algorithms, medical image analysis can now be automated for various applications, including cancer diagnosis, radiology, and ophthalmology.

However, despite the numerous advances in medical image analysis automation, the classification of brain tumors remains a challenge, motivating several research projects aimed at improving tumor diagnosis and treatment. The motivation for developing an automated brain tumor classification system is to address the need for

faster, more accurate, and cost-effective diagnosis and treatment of brain tumors. With such a system in place, doctors and medical professionals can make informed decisions on patient care and treatment plans, improving patient outcomes and quality of life.

1.3 Problem Statement

Precise identification and classification of brain tumors are pivotal for determining optimal treatment strategies and improving patient outcomes. Historically, these tasks have been carried out via visual examination of medical imaging scans, which can be laborious and susceptible to inaccuracies. However, recent advancements in deep learning techniques, notably CNNs, have yielded positive outcomes in case of the analysis of the medical images.

Our project aims to create a precise brain tumour classification and identification system using deep learning methods. Preprocessing will be applied to the MRI scans to improve tumour visibility. To ensure accurate classification of tumours with various grades, a CNN model is trained using a sizable dataset of labelled MRI scans. The model will be put to the test on a different set of MRI scans and contrasted against conventional visual inspection techniques in order to gauge its efficacy and accuracy.

By increasing the precision and effectiveness of the procedure, our project has the potential to revolutionise the diagnosis and treatment of brain tumours. Making informed decisions about diagnosis and treatment planning could be facilitated by the development of an automatic detection and classification system using deep learning techniques. This would improve patient outcomes.

1.4 Objectives

The classification and treatment planning of brain tumours have shown signs of improvement thanks to deep learning techniques. Our project aims to develop a deep learning-based convolutional neural network (CNN)-based automatic system for detecting and classifying brain tumours.

1. **Dataset Collection and Preprocessing:**

   Collecting and preprocessing a dataset of MRI images of brain tumours is the project's first goal. The dataset is crucial for the CNN model's accuracy and dependability, and it needs to be meticulously preprocessed to remove noise and unimportant data that might impair the model's performance. The following steps are included in this goal:

   - The first step is to gather the dataset of MRI scans of brain tumors
   - The next step is to ensure that the images in the dataset is properly labelled or not.

2. **CNN Model Design**

   The second objective of our project is to develop a CNN architecture that can accurately recognise and categorise various types of brain tumours. The CNN model will search MR images for distinctive features and patterns connected with brain tumours using a variety of convolutional, pooling, and fully connected layers. The following activities are part of this goal:

   - Creating a CNN architecture that can accurately classify and identify various types of brain tumours.
   - The next step is o train the CNN model.
   - Third step is basically to enhance the CNN model's architecture and hyperparameters to attain high levels of specificity, sensitivity, and precision.

3. **Performance Evaluation**

   Evaluating the CNN model's performance is the study's main goal. The evaluation will show how effective and reliable the model is at detecting brain tumours. In order to evaluate the performance of the model, a number of evaluation criteria will be used, including accuracy, sensitivity, specificity, and precision.

1.5 Methodology

The suggested study methodologies for a more thorough classification of brain tumours are described in this section. We provide a detailed analysis of the architecture of the proposed CNN-based approach as well as the various pre-trained CNN models that were used to identify and categorise meningiomas, pituitary tumours, and gliomas in brain MRI images.

We will examine deep learning methods for segmenting brain tumours in this section. We will go over the MR image pre-processing procedures, the software and dataset we used for our study, the various segmentation networks, and their designs. The methodology for conducting experiments, which will test various networks and regularisation methods to see how well they segment brain tumours, will be the main topic of the final section of this chapter.

1. Import each required module and load the data.
2. Data visualization
3. Read the images and save them along with the labels that go with them.
4. Data preprocessing.
5. Data Augmentation
6. Feature Extraction.
7. Define the CNN model & set the number of output classes.
8. Train & observe the model

1.6 Organization

Chapter 1: Introduction The goal of the introductory chapter is to give a general overview of the deep learning-based algorithm designed to reliably detect and categorise brain tumours from MRI scans. It emphasises the value of early brain tumour classification and detection for better patient outcomes.

Chapter 2: Literature Review This chapter aims to present various literature reviews of recent studies on the classification of brain tumours using deep learning methods. It goes over the various neural network architectures and accuracy

assessment methods that have been employed by scientists to address the classification problem for brain tumours.

Chapter 3: Background The background of brain tumour classification, including the various types of brain tumours, imaging methods used for diagnosis, potential causes of brain tumours, and available treatments, will be covered in this chapter.

Chapter 4: System Design & Development The pre-processing, neural network techniques, and accuracy evaluation used in a brain tumour classification research project are all described in detail in this chapter. It gives readers a thorough understanding of the project's methodology.

Chapter 5: Experiments & Results Analysis The test results from the previous chapter are thoroughly examined in this chapter, along with the model's advantages and disadvantages. It offers a thorough assessment of the model's potential for use in clinical settings, highlighting its high level of classification accuracy for various kinds of brain tumours.

Chapter 6: Conclusions The project's results and limitations are summarised in the final chapter, along with suggestions for future research topics and possible enhancements. The model's potential to increase tumour diagnosis reliability and accuracy using deep learning techniques is highlighted.

1.7 Summary

The thesis' structure and the report's organisational structure are also briefly described in this chapter. The reader will have a clear understanding of the project's scope, the difficulties it seeks to solve, and the significance of automated brain tumour detection and classification after reading this chapter, which lays the groundwork for the subsequent chapters.

CHAPTER 2
LITERATURE SURVEY

2.1 Overview

In medical image analysis, precise algorithms are essential because they have a direct impact on clinical trials and patient treatment outcomes. Several studies have been published in recent years with the goal of improving the methods for classifying brain tumours. These papers' authors come from a variety of backgrounds, including deep learning, object recognition, and image processing. The purpose of this chapter is to discuss these papers' methodologies. It will give a summary of the most recent advancements in medical image processing and act as a manual for academics and professionals in the industry.

2.2 A summary of the relevant papers

Zhang et al. [1], proposed a CNN model was developed for classifying brain tumors, achieving a 95.7% accuracy rate on the BraTS dataset. The model utilized various steps such as image preprocessing, data augmentation, and labeling. The MRI images were first normalized and resized to 224x224 pixels. Data augmentation process were then employed. The ResNet-50 model was and fine-tuned on the preprocessed and augmented dataset. Then the model was and evaluated on a validation set to determine its accuracy. The study shows the effectiveness of their approach in classifying brain tumors using CNNs.

Anas E. H. Salim et al. [2], proposed a method for classifying brain tumours without skull stripping using the EfficientNet-B0 architecture. The BraTS dataset was enhanced and preprocessed by the authors, who then used it to fine-tune the pre-trained EfficientNet-B0 model. The authors' 93.4% accuracy rate demonstrates the EfficientNet-B0 architecture's potential for brain tumour classification tasks. Their method demonstrates how transfer learning can help decrease the volume of training data and training time needed, making it more useful for use in actual clinical settings.

Qaiser et al. [3], proposed CNN model for classification of the brain tumors. The approach involves various image preprocessing steps, such as image cropping and normalization. The authors also utilized data augmentation techniques, including image rotation, flipping, and scaling, to enhance the size of the training samples. They labeled the preprocessed and augmented images by the tumor type and used the EfficientNet-B0 architecture to build the CNN model. They trained the model on training dataset and able to achieved an accuracy of 95.2% on the BraTS dataset.

Khan et al. [4], proposed an EfficientNet-B0 architecture-based transfer learning for classifying brain tumors. The MRI scans first pre-processed, and skull stripping was used to remove the skull and other non-brain tissues. Data augmentation process were then employed, they enhanced the data using rotation, flipping, and zooming operations. On the preprocessed and enhanced data, they improved the EfficientNet-B0 model and used binary cross-entropy loss during training. Finally, they used metrics like accuracy, sensitivity, and specificity to assess how well their model performed.

Al-antari et al. [5], used the ResNet50 deep learning network to create an automated system for detecting and classifying brain tumors. The MRI images were preprocessed, data augmentation techniques were used, and the pre-trained ResNet50 model was adjusted using the preprocessed and augmented data. The model's overall accuracy in identifying and categorizing brain tumors was 96.67%, demonstrating the efficiency of their method.

2.3 Summary

Based on the literature study of brain tumor classification, it can be concluded that transfer learning outperforms traditional deep learning classifiers. Transfer learning utilizes pre-trained models and learns from previous tasks, thus making use of network memory, which helps in achieving better results compared to traditional approaches.

CHAPTER 3
BACKGROUND

3.1 Brain Tumor

Brain tumors are abnormal cell masses that can be malignant or benign and can arise in various areas of the brain such as the cerebellum, cerebral hemispheres, or brain stem. Although the precise cause of brain tumors remains unclear, risk factors include genetic syndromes, family history, and radiation exposure. The symptoms of brain tumors can differ depending on their size and location and may consist of headaches, vision changes, seizures, and difficulties with movement and speech. In order to recognise and classify brain tumours, clinicians and researchers need to have a basic understanding of the human brain, which is provided by Figure 3.1. With this information, Deep Learning algorithms can be created to efficiently and accurately categorise brain tumours, improving patient outcomes.

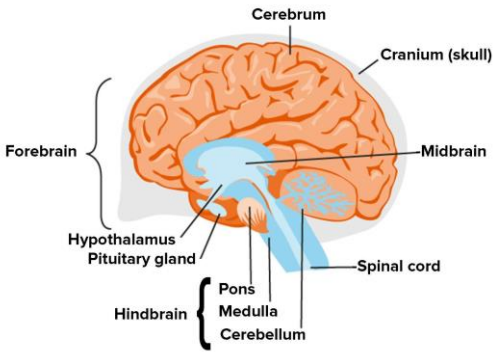


Figure 3.1: Basic Structure of Human Brain [12]

3.1.1 Causes

The precise cause of brain tumours is still unknown despite numerous research initiatives. Even though some factors, such as exposure to ionising radiation or vinyl chloride, have been linked to the development of brain tumours, they do not account for every case. Viral infections, carcinogen exposure, genetic predisposition, and embryonic remains are additional potential causes, but each theory can only explain a subset of tumour types. Although smoking has been suggested as a risk factor, the

exact mechanisms are still unclear. Collaborative research efforts are necessary to better understand the cause of brain tumors, develop effective treatments, and improve patient outcomes.

3.1.2 Classification of Brain Tumor

Brain tumors can be categorized according to various factors such as their location, behaviour, or the type of cells from which they originate. One method of classification involves grouping brain tumors based on the specific type of cells from which they develop.

1. **Gliomas Tumor:** Gliomas refer to tumors that originate from the glial cells in human brain which may be malignant or benign. Most frequent types of gliomas include oligodendrogliomas, astrocytoma's, and glioblastomas.

2. **Pituitary Tumor**: Pituitary tumors are tumors that arise in the pituitary gland and can cause hormonal imbalances. These tumors can be either functional or non-functional, meaning they may or may not secrete hormones.

3. **Meningiomas Tumor**: Meningiomas are noncancerous tumors that develop from the tissues covering the brain and spinal cord called meninges.

Accurate classification of brain tumors is vital as it impacts the selection of the most effective treatment and prognosis. Treatment options for brain tumors includes chemotherapy, surgery, or a combination of these approaches. The patient's overall health, tumor type, size, and location play a crucial role in determining the optimal treatment method.

3.1.3 MRI Image and Treatments

3.1.3.1 MRI Image

Brain tumours can be categorised according to their location, size, and other features using an MRI scan. Radiologists classify brain tumours according to their histologic characteristics and cellular origin using the World Health Organisation classification system. A thorough description of the imaging results, including how the tumour

appears on various MRI sequences, how it interacts with nearby structures, and its potential classification, are included in the MRI report for classifying brain tumours. The characteristics of a brain tumour, including its type, size, and location, as well as the patient's general health and preferences, help the treating physician choose the best course of action, which may combine surgery, radiation therapy, and chemotherapy.

3.1.3.2 Treatments

Surgery is the most popular method of treating brain tumours because it allows doctors to diagnose the condition and get rid of as much of the tumour as they can. In cases where the tumors are more severe, radiation therapy may be necessary, delivered either through radiosurgery or stereotactic radiosurgery. While chemotherapy can be effective, it has drawbacks such as drug toxicity and varying patient responses. Achieving the best possible outcomes for patients requires a multidisciplinary team of neurosurgeons, radiation oncologists, and medical oncologists.

3.2 Deep Learning

In the area of machine learning known as deep learning, relevant features are automatically extracted from large datasets using artificial neural networks. When applied to brain tumor classification, deep learning has demonstrated remarkable potential for enhancing the accuracy and efficiency of tumor diagnosis. By training NN’s on a vast array of MRI images, the model can learn to distinguish healthy brain tissue from various brain tumor types. This leads to quicker and more precise tumor classification, which can ultimately aid in patient outcomes and treatment planning. One of the primary benefits of deep learning in brain tumor classification is its ability to identify subtle patterns and characteristics that may be missed by human observation. This can help detect tumors at earlier stages, ultimately improving prognosis and treatment options. However, it is critical to meticulously design and validate deep learning models to ensure their dependability and safety within clinical settings. The diagram depicted in Figure 3.2 illustrates that machine learning is a

larger field encompassing deep learning as a subfield, and both fall under the umbrella of artificial intelligence.

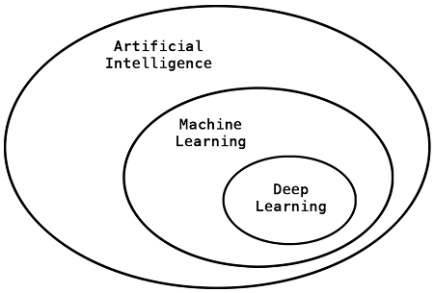


Figure 3.2: Picture of Deep Learning in the context of AI

3.2.1 Artificial Neural Network

3.2.1.1 Artificial neurons

Artificial neurons, also called perceptron's, are the essential components of NN's are used in DL. They receive input signals, which can be from other neurons or external sources, and use weights and activation functions to process them. Figure 3.3 shows a biological neuron illustration, which can be useful in the context of brain tumor classification. Understanding the structure and function of neurons can aid in identifying abnormal growths in the brain, as tumors can disrupt normal neural connections.

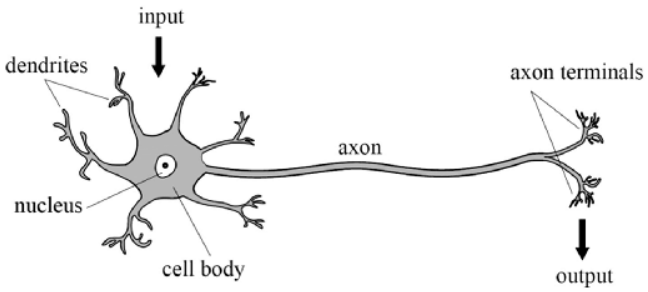


Figure 3.3: Biological Neuron Illustration [13]

3.2.1.2 Basic Operation of Neural Networks

A deep learning method with a focus on image processing is known as a neural network (NN). The matrix analyses data, trains itself to spot patterns, and then forecasts the result of fresh, related data. Layers of neurons make up the NN. First, input is received by a layer that accepts it. The final outcome is forecasted by the

output layer. In first layer of a neural network, each neuron receives input from one pixel. Channels connect neurons between layers, with a "weight" value assigned to each connection. A mathematical neuron is shown in Figure 3.4, consisting of a circular cell body connected to input and output branches through weighted connections. The inputs are combined and passed through an activation function, and the resulting output is transmitted to other neurons through the output branches.

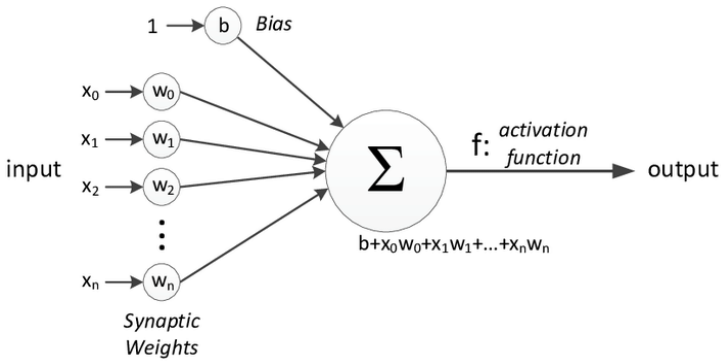


Figure 3.4: Mathematical Neuron Illustration [14]

3.2.2 Convolutional neural network

CNNs are a type of deep learning NN which are excel in image recognition and classification tasks. Their unique ability to assume certain properties about input data, particularly images, minimizes pre-processing requirements, reduces number of network parameters, and results in efficient implementations. Neurons in CNNs contain learnable biases and weights and perform non-linear and dot product operations on input data.

3.2.2.1 Convolutional Neural Network Architecture

A typical type of NN used for image identification and classification is the convolutional neural network (CNN). A CNN's architecture is made up of various layers, each serving a particular function. The most typical CNN layers are:

1. **Convolutional Layers:** The input image is subjected to a series of filters (sometimes referred to as kernels) by these layers. Each filter consists of a tiny matrix that is slid over the input image to compute the dot products of the pixels it overlaps.

2. **Pooling Layers:** The convolutional layers' feature maps produced by these layers have smaller spatial dimensions. The most typical pooling operation, known as Max pooling, chooses the highest value possible from a group of neighbouring feature map pixels.

3. **Activation Layers:** The activation functions in these layers introduce nonlinearity to the output of the previous layer.

4. **Fully Connected Layers:** The CNN's convolutional layers establish connections between neurons in the preceding and succeeding layers. After the final convolutional layer produces its output, it undergoes a flattening process and is transmitted to fully connected layers.

5. **Dropout Layers:** These layers randomly drop a certain percentage of neurons during training, to prevent overfitting.

The proposed approach for brain tumor classification is depicted in Figure 3.5, which provides a general outline. The CNN architecture can vary depending on the input image's dimensions and intended use.

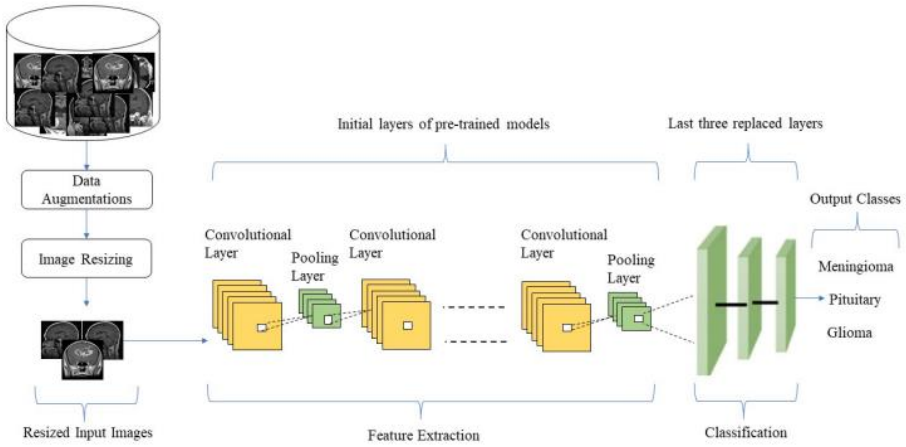


Figure 3.5: Overview of the proposed method for brain tumor classification [15]

CHAPTER 4
SYSTEM DEVELOPMENT

4.1 Overview

The basic aim of this chapter is to describe the theoretical background needed to comprehend the report's content and also provide the theoretical background for a brain tumor classification project. The purpose of this chapter is to introduce the basic components of a segmentation network and the segmentation task. Additionally, it provides metrics for evaluating the networks and a review of previous research in the field. This chapter aims to provide the reader with a clear understanding of the underlying principles of brain tumor classification and the various techniques used to achieve accurate results.

4.2 Overall Design

The classification of different types of brain tumours is an essential task in medical imaging, and CNNs have demonstrated promising results in doing so. In this study, we present three distinct classification models for brain tumours, each with unique benefits. For smaller datasets, the traditional CNN model is straightforward and efficient. EfficientNetB0 is suitable for larger datasets with complex features because it uses scaling techniques to achieve high accuracy while maximising efficiency. To avoid the vanishing gradient issue and achieve high accuracy on large datasets with complex features, ResNet50 uses residual blocks and skip connections.

4.3 Software & Environment

The hardware and software for assessing CNN's performance in processing MRI brain tumour image classification are introduced in Section 4.3 of the study. A workstation with an NVIDIA GPU was used as part of the experiment's hardware, and Python, TensorFlow, and Keras libraries were used as part of the software. The dataset that was used to train and test the models is also described in this section.

4.3.1 Software Requirements

1. **Python:** Python is a programming language that has become more and more popular in a variety of disciplines, including data science, machine learning, and web development. It is a high-level, interpreted language that emphasises readability and clarity in code.

2. **Tensorflow:** Open-source deep learning framework TensorFlow provides scalability and flexibility for building and developing machine learning models. It has various APIs that help with various machine learning tasks and supports distributed computing, which enables the training of larger and more complex models.

3. **Keras:** The development and training of deep learning models is accelerated by the use of the high-level neural network API Keras. Developers can focus on the task at hand thanks to its uniform interface, high level of abstraction, pre-trained models, and ability to integrate with other machine learning libraries like TensorFlow, which also saves time and resources.

4. **Google Colab:** Running Python code for machine learning projects, especially for deep learning applications, is made easier by using Google Colab, a cloud-based notebook environment.

5. **Matplotlib:** The Python data visualisation library Matplotlib is adaptable and user-friendly and provides many customization options. It functions well in conjunction with other Python libraries, such as NumPy and Pandas.

6. **OpenCV:** A variety of tools for image and video processing applications are provided by OpenCV, a well-known and potent computer vision library that is open source. For developers working on various computer vision projects, it is the best option due to its adaptability, simplicity, and excellent performance.

4.4 Dataset Description

A collection of medical imaging data, including MRI scans of patients with various types of brain tumours, makes up the Brain Tumour Classification MRI dataset. 3264 high-resolution MRI scans of brain tumours with corresponding tumour type labels make up the dataset. The information is meant to help scientists create deep learning models for the precise and effective classification of brain tumours. The dataset is divided into the "training_set" and "testing_set" subfolders. 2,870 MRI scans are in the "training" subfolder, and 394 MRI scans are in the "testing" subfolder. Meningioma, glioma, and pituitary tumour subfolders are present in every subfolder, one for each type of tumour. There were 937 images in the Meningioma class. There were 926 images in the Glioma class. There were 901 images in the Pituitary class. There were 500 images in the No Tumour class. The summary of the image dataset used, with a count of the number of images in each tumour class, is shown in Table 4.1.

Table 4.1: Summary of Used Image Dataset

Class	**No. of Images**
Meningioma Tumor Class	937
Glioma Tumor Class	926
Pituitary Tumor Class	901
No Tumor Class	500
Total Images	3264

Brain Tumor Classification MRI dataset is a valuable resource for researchers working in the field of medical image analysis and machine learning.

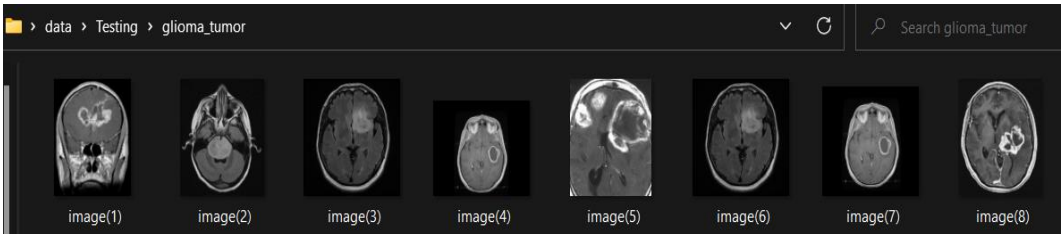


Figure 4.1: “glioma_tumor”

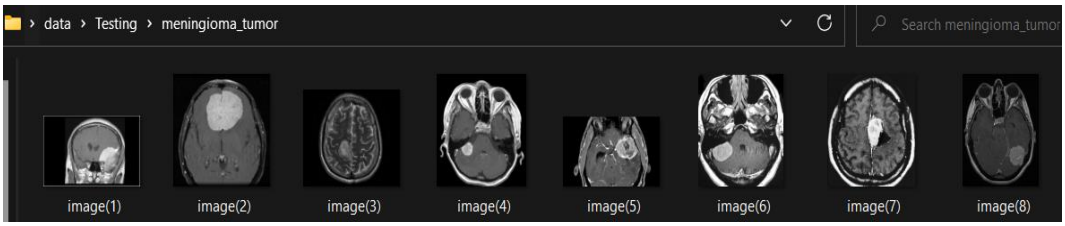


Figure 4.2: "meningioma_tumor"

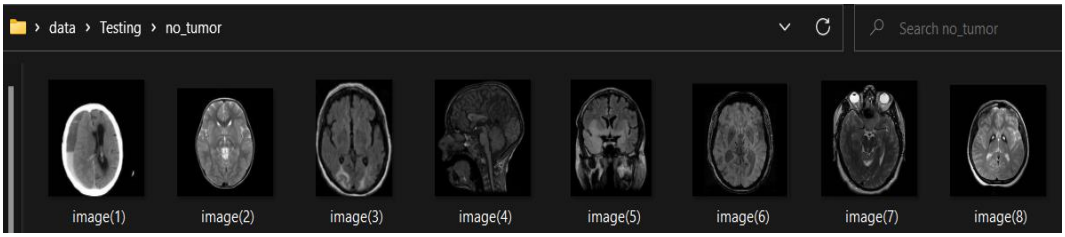


Figure 4.3: "no_tumor"

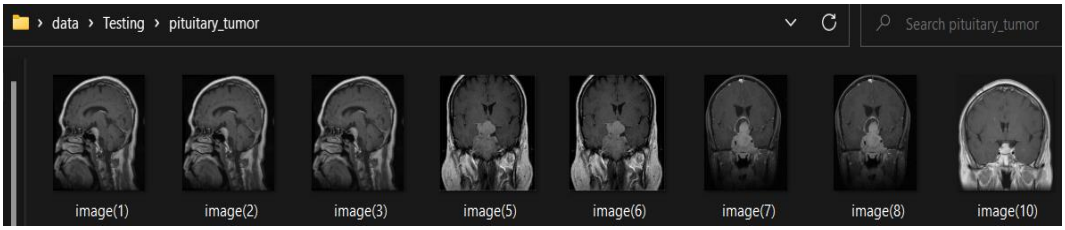


Figure 4.4: "pituitary_tumor"

4.5 Data visualization

For analysing and interpreting complex data, data visualisation is a crucial tool. When classifying brain tumours with deep learning, visualisation techniques can aid in spotting a variety of patterns and features in the data that might not be apparent through conventional analysis. Researchers can increase the precision and efficacy of their deep learning model by segmenting the dataset into folders based on the types of tumours and employing static and dynamic visualisation methods. As a result, anomalies or outliers in the data can be found and eliminated, increasing the model's accuracy. Figure 4.5 displays sample images from each class of brain tumors, including glioma, no tumor, meningioma, and pituitary tumors. Each sub-image provides a distinct visual reference for the unique features of each tumor type, such as well-defined borders or cystic appearance.

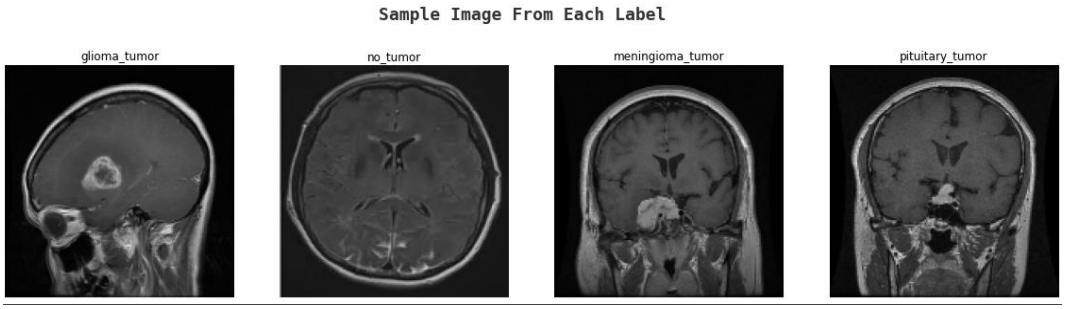


Figure 4.5: Sample Images from each class of the dataset.

4.6 Data pre-processing

For Deep Learning projects, including brain tumor classification, data pre-processing is a crucial stage. It entails transforming raw data, such as brain MRI images, into a form suitable for use by machine learning models. The quality and format of the data can significantly impact model performance, making data pre-processing critical for accurate outcomes. Various techniques can accomplish this, such as data augmentation, which produces new images by modifying current ones through techniques like scaling, flipping, or rotation.

4.6.1 Data Augmentation

Data augmentation is a technique that basically creating additional data samples by modifying the existing dataset. The goal is to artificially increase the size of the dataset. Modifying the data can involve making minor changes to the existing data, or using machine learning algorithms to generate new data points in the latent space of the original dataset. Figure 4.6 demonstrates an example of data augmentation in brain tumor classification. The figure displays a grid of brain MRI images that have undergone augmentation using several techniques, including flipping, scaling, rotation, and noise addition.

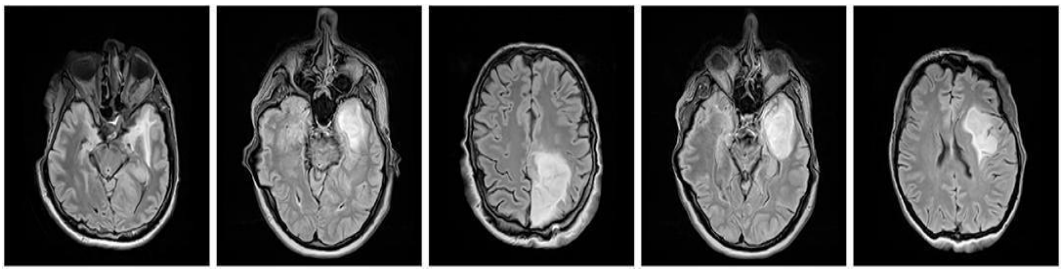


Figure 4.6: Data Augmentation

4.7 Feature Extraction

Feature extraction is a critical step in medical image analysis, particularly in the deep learning-based classification of brain tumours. It involves identifying and extracting significant and distinguishing features from raw image data. CNNs are a suitable tool for this task as they can automatically learn hierarchical representations of image data. Feature extraction can be classified into two types: handcrafted features, which are designed by experts, and learned features, which are automatically learned by deep learning algorithms.

4.8 Train and Test set

Splitting the available data into training and testing sets is an essential step in developing a deep learning model for categorizing brain tumors. The training set, which comprises the majority of the data, is used to train the model on the input features and target labels. The performance of the model is evaluated using the testing set, which contains data that the model has never seen before. Insuring that the model can correctly categorise brand-new data that it has never seen before helps to improve the model's accuracy and generalizability.

4.8.1 Training Data

A balanced and representative training dataset with high quality is necessary to build a CNN model that performs well in classifying brain tumours. Data augmentation, normalization, and feature engineering are examples of preprocessing methods that can improve dataset quality and diversity. To avoid overfitting, the CNN model is trained using backpropagation and its accuracy is assessed on a different testing dataset. The accuracy and robustness of the model are significantly influenced by the CNN architecture design and dataset quality.

4.8.2 Test Data

The test dataset plays significant role in evaluating the performance of a deep learning CNN model used to classify brain tumors. It comprises data that has not been previously seen by the model and provides an unbiased assessment of its ability to handle new data. Model validation or testing involves comparing the actual and

expected output. Table 4.2 outlines the train-test split strategy adopted for the brain tumor classification model. The table indicates that 90% of the data was allocated to training the model, while the remaining 10% was utilized for evaluating the model's performance.

Table 4.2: Create a Train and Test set.

A portion of Data	**Explanation**	**Split Chosen**
Training Data	A portion of the information was used to train the model.	90%
Testing Data	A part of the information utilized to evaluate how well the model worked during excitable adjustment and training.	10%

4.9 Define CNN model & set the number of output classes

Building the software for our project's "brain tumor categorization" is now the project's next and most crucial stage. We have employed 2 ways in order to complete this work. Applying CNN algorithms, we first create our own model, and then we employ a pre-trained model. After that, we assess the two models and discover that the pre-trained model outperforms the CNN Model.

4.10 Train & observe the model

To develop a high-performing deep learning CNN model for brain tumor classification, various hyperparameters such as kernel sizes, learning rate, epochs, and batch size must be defined. The CNN architecture is also essential and should be adjusted based on the dataset's characteristics. By training the CNN model its performance on the test dataset, the model's accuracy can be determined. Fine-tuning the model's hyperparameters can help improve its accuracy, and after selecting optimal hyperparameters, the model can be retrained on the complete dataset and tested again. This method ensures optimal performance and reduces the risk of overfitting.

4.11 Predict & save our model

Predicting and saving the model is the last step in the creation of a deep learning CNN model for classifying brain tumors. Making accurate predictions requires preprocessing fresh images using the same methods as in the training phase, running them through the trained CNN model, and then comparing the results to the actual tumor type. To verify that the model can be applied to new and untested data, accuracy testing is essential. It is crucial to save the model in a suitable format for use in upcoming applications. Data augmentation, preprocessing, model construction, training, testing, and classification are all included in the suggested strategy for classifying brain tumors. The trained and validated model is retained for later use, and the model's hyperparameters are adjusted.

4.12 Proposed System

The design and use of artificial neural networks, including convolutional neural networks, are inspired by the structure and function of neural networks in the human brain. They employ techniques such as statistical clustering, pattern recognition, optimization algorithms, vector quantization, approximation, and classification methods. Three types of artificial neural networks, based on their connectivity, include feedforward, recurrent, and convolutional neural networks. In the context of CNNs, there are two approaches that can be utilized to improve their effectiveness.

1. The standard CNN approach
2. Transfer learning.

The typical CNN method entails creating a CNN from scratch for a particular purpose, such as classifying brain tumors. This method can be time-consuming and computationally costly, and it needs a lot of training data. Although it offers complete control over both, the CNN's architecture and hyperparameters can be modified for best performance.

The use of pre-trained models in transfer learning can be advantageous because it saves time, is computationally effective, and needs less training data. The pre-trained CNN already has some general features that it can use for the new task. Transfer

learning can be particularly beneficial when there is insufficient training data for the new task, or when the pre-trained CNN was trained on a similar task.

4.12.1 Traditional CNN Approach

Neural networks are models that are based on how the human brain functions and may be applied to a number of tasks, including the classification of images. CNNs, or convolutional neural networks, are extremely good at classifying images. They are capable of scaling images by converting a three-dimensional input set into a three-dimensional output set (length, width, and depth). CNNs typically include input, convolutional, ReLU, pooling, and fully connected layers. The convolutional layer divides the input image into a number of discrete areas, while the ReLU layer completes the functions of element-dependent activation. The pooling layer is used for downsampling, and the fully connected layer generates a class score or label score price based on the probability between 0 and 1. Figure 4.7 shows that the basic architecture of a Convolutional Neural Network (CNN) model consists of input image(s) that pass through several convolutional and pooling layers. These layers extract features from the image(s) and reduce their dimensions. The extracted features are then passed through fully connected layers for classification or prediction.

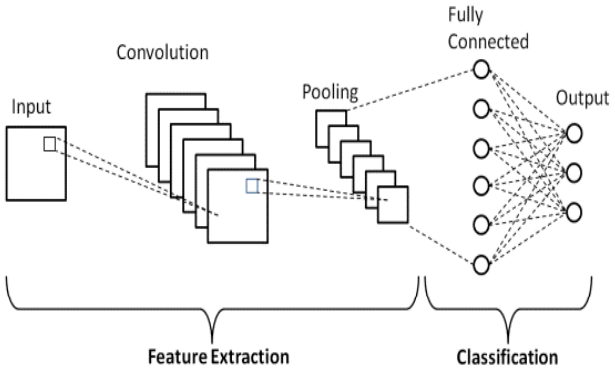


Figure 4.7: Basic Architecture of CNN Model [15]

(i) The Convolution Operation: Convolutional Neural Networks (CNNs) utilize the convolution operation to perform feature extraction. The mathematical operation of convolution belongs to the class of integral transforms. In CNNs, it is used to extract features from the input data by applying filters to the input data. The output of this

operation serves as input for further processing by the neural network. The convolution process in CNNs requires determining the product of the two functions which are integral. f(τ) and g(t-τ), where g is flipped and shifted over a range of values. This produces a third function, (f*g)(t), which describes the degree of overlap between f and g at each point in time t. The formula for convolution is:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau) g(t - \tau)\, d\tau \tag{4.1}$$

where the integral is taken over the entire range of τ from negative infinity to positive infinity.

The convolution operation takes into account three factors:

1. **Input Image:** The image that is provided to the convolutional neural network as an input is known as an input image. It is typically a 2D array of pixel values representing the intensity or color of each pixel.

2. **Feature Detector (Kernel/Filter):** A simple matrix of weights called the feature detector, sometimes referred to the filter or kernel which is basically used to extract specific features from the input image.

3. **Feature Map (Activation Map):** Feature map, which is often referred to as an activation map, is a 2D array that exhibits the level of detector activation at every pixel of the input brain tumor image. It shows where a specific type of feature can be located in the image, hence the term "feature map". The feature map serves as input to further processing by the neural network.
   The process of convolution is shown in Figure 4.8.

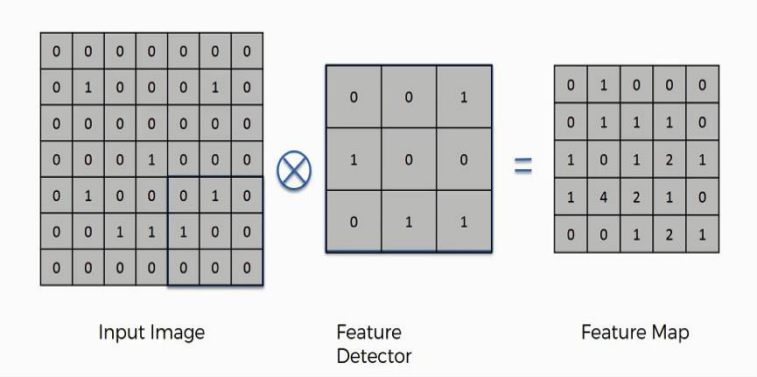


Figure 4.8: Process of convolution [16]

(ii) CNN Model Built Using Layers: CNNs are frequently employed for the classification of images, including identifying different types of brain tumors. The steps of the conventional CNN model for classifying brain tumors are as follows:

Convolutional Layer: The CNN model's primary foundational element is the convolutional layer. From input images to extract, it comprises a set of learnable filters. Small area of input image is subjected to each filter, and the resulting feature map is created. A stack of feature maps is produced after repeating this procedure for each filter. Figure 4.9 demonstrates a convolutional example in CNN's convolutional layer. Convolutional layer also has certain fundamental characteristics, including:

$$(1 * 1 + 0 * 2 + 3 * 1 + 0 * 4 + 1 * 5 \ldots\ldots\ldots) \quad (4.2)$$

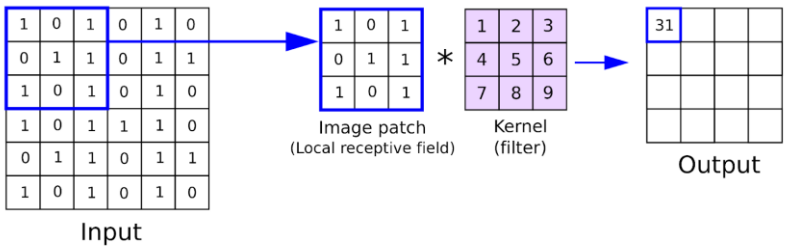


Figure 4.9: Convolution Operation of CNN [16]

Two crucial hyperparameters in convolutional layers regulate the size of the output volume:

Stride: Stride refers to the step size with which the convolutional filter is moved over the input volume. A larger stride value results in a smaller output volume size, while a smaller stride value leads to a larger output volume size. Stride can be set to any positive integer value, but commonly used values are 1, 2, and 3. When the stride is set to 1, filters move at a rate of 1wach movement. When the stride is set to two, the filters move at a rate of two pixels each movement.

Figure 4.10: Applying a filter while moving source image stride set to 1

Figure 4.10 demonstrates how 2*2 filters pass over width and height when the stride is set to 1.

Zero padding: Zero padding is the basic technique of enclosing the input volume in additional rows and columns of zeros before applying convolutional filters. This method can keep the input volume's spatial dimensions while preventing an excessively small output volume. The amount of padding can be set to any non-negative integer number, and zero padding can be applied either symmetrically or asymmetrically. The scenario of zero-padding of an input is shown in Figure 4.11.

0	0	0	0	0	0
0	35	19	25	6	0
0	13	22	16	53	0
0	4	3	7	10	0
0	9	8	1	3	0
0	0	0	0	0	0

Figure 4.11: An input with zero padding (padding amount = 1)

Pooling Layer: The feature maps' size is decreased by the pooling layer, which also increases the model's resistance to changes in the input image. The most well-liked pooling method, referred to as max pooling, selects the greatest value inside a certain region of the feature map.

Different Kinds of Pooling Functions:

1. **Max Pooling:** From each rectangular, non-overlapping area of the feature map, max pooling selects the greatest value. The most noticeable aspects of the image or feature map are preserved. The feature maps' dimensionality is

decreased through max pooling, which also lowers the number of training parameters and computations needed. Max-pooling is displayed in Figure 4.12 with a shape of 2*2. Initial output element and input tensor elements used in the output computation are represented by the darkened portions: $max(0, 1, 3, 4)$

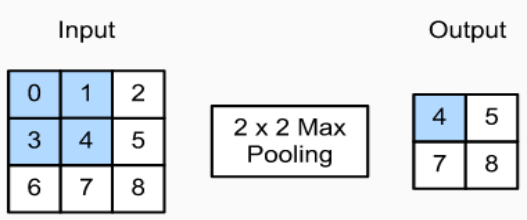


Figure 4.12: Max Pooling [17]

2. **Average Pooling:** Average pooling takes the average value from each non-overlapping rectangular region of the feature map. When the average intensity of features in a region serves as a reliable predictor of a feature's presence, it can be helpful. Average-pooling is displayed in Figure 4.13 with a shape of 2*2. The initial output element and the input tensor elements used in the output computation are represented by the darkened portions: $avg(0, 1, 3, 4)$

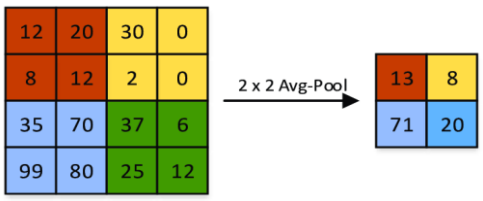


Figure 4.13: Average Pooling [17]

In brain tumor detection using CNNs, max pooling is often preferred over average pooling because it helps to preserve important features while reducing spatial dimensions. Max pooling works by taking the maximum value within each pooling window, whereas average pooling takes the average value. Max pooling is known to be more effective in preserving sharp features such as edges and corners, as it selects the strongest activation within each pooling window. This can be particularly important in brain tumor

detection, where the location and boundaries of the tumor can be critical in making an accurate diagnosis.

Fully Connected Layer: Output from the pooling and convolutional layers is given into a neural network by the fully connected layer. The link between the extracted characteristics and their respective labels is learned by this layer. According to Figure 4.14, the image depicts a CNN layer with 5 neurons that is fully connected. Every neuron in the layer above it is connected to every other neuron, creating a complex web of connections.

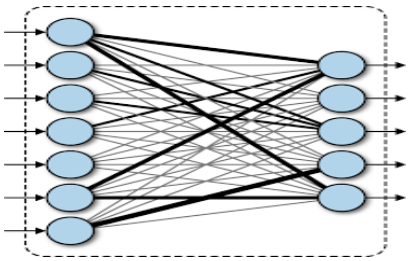


Figure 4.14: Fully Connected Layer [17]

Activation Layer: The activation layer adds nonlinearity to the model. Activation functions are necessary for neural network models to operate properly. In order to allow neural networks to simulate complex interactions between input and output, they modify a neuron's output to make it nonlinear.

Activation functions that are frequently used: Activation functions are crucial in neural networks because they add non-linearity to each neuron's output. Because it enables the neural network to recognise and learn complex patterns and relationships in data, non-linearity is crucial to neural networks. Here are a few frequently used activation mechanisms:

1. **Sigmoid:** Any input value is converted by the sigmoid function to a value between 0 and 1. Due to the fact that it can be regarded as a probability, it is helpful in binary classification tasks. However, the main drawbacks of the sigmoid function are vanishing gradients, saturation of the output, and output values that are not zero-centered. It becomes challenging to update the weights when the gradient becomes very tiny, which is known as the

vanishing gradient issue. The sigmoid function is mathematically represented as follows:

$$\Phi(z) = \frac{1}{1+e^{-z}} \tag{4.3}$$

Figure 4.15 represents the sigmoid function's curve:

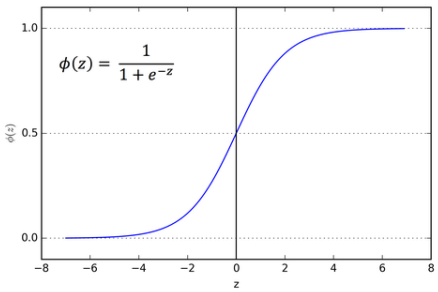


Figure 4.15: Sigmoid Function Curve [18]

2. **Tanh (Hyperbolic Tangent):** Any input value is converted by the tanh function to a value between -1 and 1. It is similar to the sigmoid function, but with a wider range of output values. Tanh can be useful in some cases where the input values are centered around zero. However, tanh can also suffer from vanishing gradients and saturation of the output.The mathematical representation of the hyperbolic tangent function is:

$$tan(z) = \frac{2}{1+e^{-2z}} \tag{4.4}$$

Figure 4.16 represents hyperbolic tangent function curve:

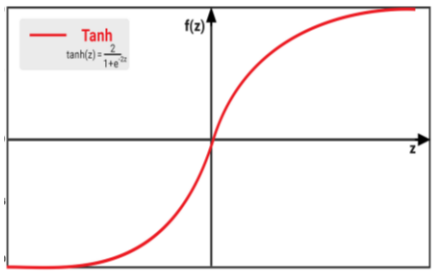


Figure 4.16: tanh Function Curve [18]

3. **ReLU:** All negative values are turned to zero by the ReLU function, while all positive values are left unaltered. The vanishing gradient problem is a common issue in deep learning where the gradients become very small, making it difficult to update the weights. Moreover, ReLU outputs are always positive, which is useful for image classification tasks where pixel intensities are always positive. The formula of ReLu is as following:

$$f(x) = max(0, max) \tag{4.5}$$

Figure 4.17 depicts the ReLu curve.

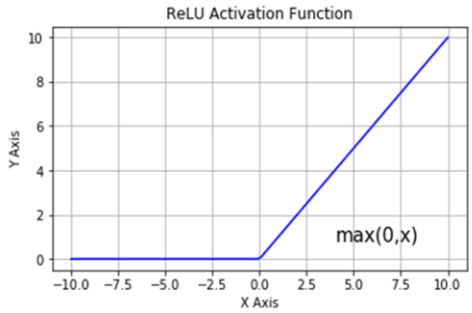


Figure 4.17: ReLU Function curve [18]

Reasons for using ReLU in classification of Brain Tumors

ReLU effectively addresses the vanishing gradient problem and introduces non-linearity into the output of neurons, making it easier for the neural network to learn complex relationships and making it a suitable activation function for the classification of brain tumours. ReLU is useful for processing large amounts of medical imaging data in image classification tasks like brain tumour classification because it is computationally efficient and produces positive values.

Dropout Layer: In neural networks, the overfitting problem is solved by a regularisation technique known as dropout. A predetermined number of neurons randomly drop out (are temporarily removed) during the forward pass. The network is compelled to develop more dependable features as a result, which interact favourably with a variety of random subsets of the other neurons. The dropped-out

neurons won't receive any weight updates during the backward pass. Dropout has been shown to be an effective way to improve generalisation and reduce overfitting in deep neural networks. The dropout in the NN results are shown in Figure 4.18. The inactive neurons in this network are known as cross neurons.

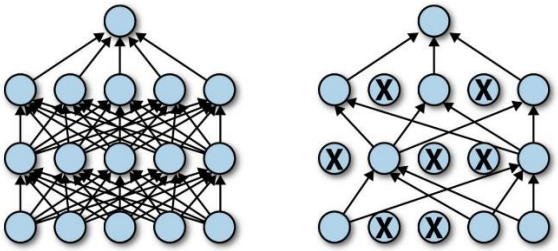


Figure 4.18: Before and after Dropout Network [18]

Loss Function: The loss function in a CNN quantifies the discrepancy between the predicted output and the target output, and the choice of loss function depends on the problem being solved. The CNN adjusts its parameters to minimize this loss function.

Commonly Used Loss functions: Here are some of the loss functions in deep learning:

1. **Mean Squared Error (MSE) Loss:** IN problems like Regression, this loss function is commonly used where the main goal of this loss function is to predict continuous values. The formula for MSE loss is:

$$L = \frac{1}{n}\sum(y - y_{hat})^2 \tag{4.6}$$

   Where in the above formula: L is the term which is MSE loss,the other term y is the ground truth target value, y_hat term is the predicted target value and lastly n is the number of training examples.

2. **Binary Cross-Entropy Loss:** In binary classification types problems this loss function is commonly used where its main goal is to predict one of the 2 possible classes. The formula for binary cross-entropy loss is:

$$L = -[y\ log(y_{hat}) + (1-y)\ log(1-y_{hat})] \quad (4.7)$$

Where: y is the ground truth label which can be either 0 or 1, L is the binary cross-entropy loss, and y_hat is the projected probability of the positive class (between 0 and 1).

3. **Categorical Cross Entropy Loss:** In multi-class classification problems, where the objective is to predict one of several potential classes, this loss function is frequently used. Categorical cross-entropy loss is calculated as follows:

$$L = -\sum_{j=1}^{c} y_j\ log(y_{hat_j}) \quad (4.8)$$

Where L stands for categorical cross-entropy loss, y_i is the i-th class's ground truth label (encoded as a one-hot vector), and y_hat_i is the i-th class' predicted probability.

In the case of brain tumor classification, the problem involves predicting one of several possible classes (normal tissue, benign tumor, malignant tumor), making categorical cross-entropy a suitable choice for the loss function. By minimizing the categorical cross-entropy loss, the CNN learns to predict the correct class for each input image, which can be useful for accurate diagnosis and treatment planning.

Optimization: During the training of deep learning models, optimizers alter the weights of the neural network to lower the loss function. The weights are updated using the gradients of the loss function, which depict how the loss function changes in relation to each weight in the model. Which optimizer should be used depends on the size of the dataset and the type of problem being addressed. Different optimizers have different advantages and disadvantages, and some work better with shallow or deep neural networks while others work better with small or large datasets. Examples of frequently used optimizers include Gradient Descent, Stochastic Gradient Descent, Adam, Adagrad, and RMSProp.

Frequently Used Optimisers

Optimizers are algorithms that change the weights and biases of neural networks throughout the training process in order to minimize the loss function. Several frequently used deep learning optimizers are listed below:

1. **Gradient Descent Deep Learning Optimizer:** To reduce a cost function, the common optimization approach gradient descent is employed in machine learning.It works by gradually altering the model's parameters in the direction of the greatest decline in the cost function. This information is provided by the gradient, which is a vector of partial derivatives of the cost function with respect to each parameter. According to the Gradient Descent update rule:
   The formula of gradient is as following:

$$W_{new} = W_{old} - \frac{\partial(Loss)}{\partial(W_{old})} \tag{4.9}$$

   In this formula, W_old refers to the current value of the weight parameters, and W_new refers to the updated value of the weight parameters. The update rule involves subtracting the gradient of the loss function with respect to the weight parameters (∂((Loss)/∂(W_old)) from the current value of the weight parameters.

   The direction of steepest ascent is represented by the gradient of the loss function, so moving in the opposite direction (i.e., minus the gradient) causes the loss function to decrease.

   By iteratively applying this update rule, the weight parameters move in the direction of decreasing loss until a local minimum is reached. The gradient descent curve is depicted in Figure 4.19 below:

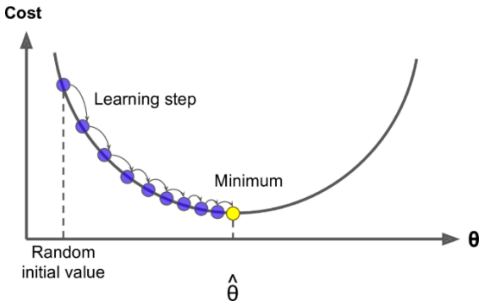


Figure 4.19: Gradient Descent [19]

It has the advantage of being a well-understood and widely used algorithm. Additionally, if the cost function is convex, it can converge to a global minimum. However, Gradient Descent can be slow to converge for high-dimensional models or large datasets, which can lead to computational costs. Additionally, every iteration requires the gradient calculation, which can also be computationally expensive for large datasets.

2. **Stochastic Gradient Descent:** SGD, an iterative optimisation technique, is used to train machine learning models. It randomly selects a subset of data points (a mini-batch) in order to calculate the gradient of the loss function and update the model parameters in the direction of steepest descent. Although the SGD update rule for the weight parameters is similar to the classic gradient descent rule, it is based on the gradient computed from a mini-batch of training data rather than the entire training data set.
   The formula for updating the weight parameters in SGD can be written as:

$$W_{new} = W_{old} - \alpha * \frac{\partial(Loss_{mini-batch})}{\partial(W_{old})} \quad (4.10)$$

W_old denotes the weight parameters' current value, W_new denotes the weight parameters' updated value, alpha denotes the learning rate (which regulates the update step size), and ((Loss_mini-batch)/(W_old)) denotes the gradient of the loss function with respect to the weight parameters, as estimated using the mini-batch of training data. The benefits and drawbacks of stochastic gradient descent are as follows:

SGD has some advantages above the standard gradient descent. It can converge faster because it updates the weights more frequently based on smaller batches of training data, and it uses less memory to store the mini-batches. However, it requires careful tuning of the learning rate to prevent oscillations around the optimal value and updates that are too large or too small. Additionally, SGD updates are noisier, so it may require more iterations to converge.

3. **Mini-batch Stochastic Gradient Descent:** Mini-batch With the help of a compact, randomly chosen subset (mini-batch) of the training data, stochastic gradient descent (MB-SGD) computes the gradient of the loss function. A mini-batch of training samples rather than a single sample is used to compute the gradient in MB-SGD, but the update procedure for the weight parameters is identical to that of stochastic gradient descent. Formula:

$$W_{new} = W_{old} - \alpha * \frac{\partial(Loss_{mini-batch})}{\partial(W_{old})} \quad (4.11)$$

In this formula, W_old refers to the current value of the weight parameters, W_new refers to the updated value of the weight parameters, alpha is the learning rate (which controls the step size of the updates), and ∂((Loss_mini-batch)/∂(W_old)) is the gradient of the loss function with respect to the weight parameters, estimated using the mini-batch of training data. The following Figure 4.20 represents the Mini Batch Gradient Descent:

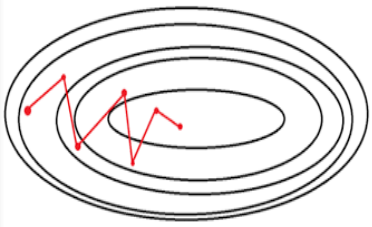


Figure 4.20: Mini-Batch Gradient Descent [19]

Mini-Batch Gradient Descent (MB-GD) has some advantages over standard gradient descent. It can converge faster because it updates the weights more frequently based on smaller batches of training data, and it uses less memory to store the mini-batches. However, MB-GD still requires careful tuning of the learning rate to prevent oscillations around the optimal value and updates that are too large or too small.

4. **SGD with momentum:** SGD with momentum is a variant of the popular SGD algorithm used to optimise the parameters of a machine learning model. The momentum term is added to the standard update rule to accelerate convergence and reduce oscillations during the optimization process.The update rule for SGD with momentum in terms of the model parameters w_old and w_new is:

$$v = \beta * v + (1 - \beta) * \partial(Loss(w_{old})) \quad (4.12)$$

$$w_{new=} w_{old} - \alpha * v \quad (4.13)$$

where v is the momentum, β is the momentum parameter, α is learning rate, and ∇(Loss(w_old)) is gradient of the loss function with respect to the model parameters at w_old. The momentum term v is a weighted average of the previous momentum and the current gradient, with the weight determined by the momentum parameter β. The update to the model parameters w_new is then calculated by subtracting the momentum term α * v from the current model parameters w_old. The following Figure 4.21 represents the Mini Batch Gradient Descent:

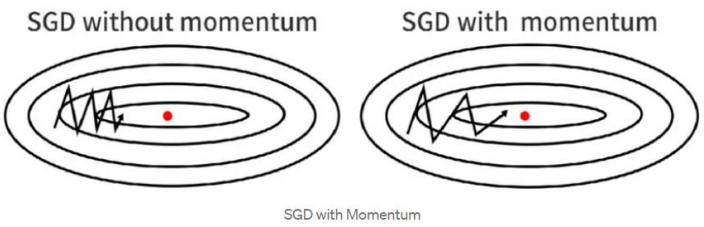


Figure 4.21: SGD with momentum [19]

Stochastic Gradient Descent (SGD) with momentum can have benefits such as accelerated convergence due to consistent direction of movement towards the minimum of the loss function, prevention of oscillations, and good performance in practice. However, it can overshoot the minimum of the loss function, which can slow down convergence. Additionally, it requires tuning of the momentum parameter β and may require a larger number of iterations compared to other optimization algorithms when the loss function is highly non-convex.

5. **Adagrad:** Adagrad is another optimization algorithm that adapts learning rate of each weight in a NN based on the historical gradients accumulated over time. The update rule for Adagrad is as follows: At each iteration of the optimization algorithm, we calculate the gradient of the loss function with respect to the weights ∂(Loss)/∂(wold). We then accumulate the squared sum of the gradients for each weight up to the current iteration as follows:

$$\alpha_{new}=\sum_{i=1}^{new}\left(\frac{\partial Loss}{\partial old}\right)^2 \tag{4.14}$$

where α_{new} is a diagonal matrix with each diagonal element corresponding to a weight in NN. We then update weights based on the following formula:

$$\Delta w = -\frac{\eta}{\sqrt{\alpha_{new}+\varepsilon}} \tag{4.15}$$

where η is the learning rate, ε is a small constant added for numerical stability, and α_{new} is the diagonal matrix of accumulated squared gradients. The first term in the update equation is the velocity term, which is multiplied by a negative learning rate η to update the weights in the direction of decreasing loss.

Adagrad's advantages include fast convergence due to its adaptive learning rate, and ease of implementation. However, its disadvantages include high memory usage due to the need to store historical gradient information for

each parameter, a tendency for the learning rate to decay rapidly over time leading to slower convergence or suboptimal solutions, and sensitivity to the initial learning rate, which can result in poor convergence or even divergence.

6. **RMS-prop:** RMSprop is an adaptive learning rate optimisation algorithm that modifies the learning rate by using a moving average of the squared gradient. To scale the update, the algorithm divides the learning rate by the squared gradient's moving average. Here is a description of RMSprop along with some of its advantages and disadvantages. The RMSprop update rule is as follows:
   Calculate the moving average of the squared gradient:

$$v_{new} = \beta * v_{old} + (1 - \beta) * \frac{\partial(Loss)}{\partial(w_{old})}^2 \quad (4.16)$$

   Calculate the update for the weights:

$$\Delta w = -\frac{\eta}{\sqrt{v_{new} + \varepsilon}} * \frac{\partial(Loss)}{\partial(w_{old})} \quad (4.17)$$

   where β is the momentum parameter, η is the learning rate, ε is a small constant to prevent division by zero, g^2 is the element-wise squared gradient, and v_old and v_new are the moving averages of the squared gradient at the old and new time steps, respectively.

   The advantages of RMSprop include its robustness to noisy gradients because it accumulates gradients over a moving window of iterations and its good convergence properties, which can occasionally be faster than Adagrad and SGD with momentum.

   It may also require large batch sizes, which can be computationally costly. The momentum parameter and the constant are two additional hyperparameters that must be tuned.

7. **Adam (Adaptive Moment Estimation):** An optimisation algorithm called Adam (Adaptive Moment Estimation) combines the benefits of the momentum and RMSprop algorithms. The learning rate is adaptively adjusted for each parameter using a moving average of the gradient and its squared value. Here is a description of the Adam optimizer's formula in terms of w_old and w_new, as well as some of its advantages. The update rule for Adam as follows: Update the moving average of the gradient:

$$m_{new} = \beta 1 * m_{old} + (1 - \beta 1) * \frac{\partial(Loss)}{\partial(w\_old)} \quad (4.18)$$

In optimisation algorithms like Adam, equation 4.18 calculates the first moment of the gradients. It updates a moving average of the gradients using exponential decay with parameter β1, which gives more weight to recent gradients. The first moment estimate is used to adapt the learning rate of each weight. The update equation helps in smoothing out noisy gradients and stabilizes the learning process by reducing the effects of large gradients. Update the moving average of the squared gradient:

$$v_{new} = \beta 2 * v_{old} + (1 - \beta 2) * (\frac{\partial(Loss)}{\partial(w\_old)})^2 \quad (4.19)$$

Equation 4.19 estimates the second moment of gradients in optimization algorithms like Adam. It updates a moving average of the squared gradients using exponential decay with parameter β2, which gives more weight to recent squared gradients. The second moment estimate helps in adapting the learning rate based on the gradient variance of each weight. This reduces the effects of noisy gradients and stabilizes the learning process. Compute the bias-corrected estimates of the moving averages:

$$m_{hat} = \frac{m_{new}}{(1-\beta 1^t)} \quad (4.20)$$

$$v_{hat} = \frac{v_{new}}{(1-\beta 2^t)} \quad (4.21)$$

where t is the current iteration. Now calculate the update for the weights.

$$\Delta w = -(\frac{\eta}{(\sqrt{(v_{hat})}+\varepsilon)}) * m_{hat} \quad (4.22)$$

where β1 and β2 are the exponential decay rates for the moving averages, η is the learning rate, ε is a small constant to prevent division by zero, g^2 is the element-wise squared gradient, and m_old, v_old, m_hat, and v_hat are the moving averages and bias-corrected estimates of the gradient and its squared value.

Advantages of Adam:

1. **Adaptive learning rate:** Adam adjusts the learning rate for each parameter, which eliminates the need for manual hyperparameter tuning.
2. **Works well on large datasets:** Adam's adaptive learning rate and memory-effective computation make it a good fit for large datasets and high-dimensional parameter spaces.

We use the Adam optimizer in brain tumour classification because it has a number of benefits that are especially helpful in this application. The Adam optimizer can work with large datasets and high-dimensional parameter spaces, which is important in medical imaging where the quantity of images and model complexity can be very high. Adam also has the ability to handle noisy gradients, which comes in handy when the data is noisy or lacking. Finally, it has been demonstrated that Adam converges more quickly and performs better than other optimisation algorithms, which is crucial for the precise and prompt diagnosis of brain tumours.

Evaluation: A test set of images is used to assess the model's performance in classifying images of brain tumours.

In conclusion, the conventional CNN model for classifying brain tumours entails preprocessing the data, convolutional, activation, pooling, fully connected, dropout, use of a loss function and optimisation algorithm to train the model, and evaluation of the model's precision on a test set of images.

4.12.2 Transfer Learning Approach

To increase a model's accuracy and effectiveness on a new task, neural networks, including CNNs, can be trained using the potent Deep Learning technique of transfer learning. Transfer learning can be used to use pre-trained models that have learned features from a large dataset to improve the accuracy of a model on a small dataset of brain MRI images in the case of classifying brain tumours. EfficientNet and ResNet are two well-liked pre-trained CNN architectures for image classification. Both ResNet and EfficientNet are built to be computationally effective while providing state-of-the-art performance on image classification tasks.

ResNet50: A deep neural network architecture called ResNet50 has 50 layers. In order to learn high-level features from images, ResNet50 is a convolutional neural network (CNN) that has been trained on significant datasets, such as ImageNet. To help the network learn more effectively and prevent the issue of vanishing gradients, it employs a residual learning approach that makes use of skip connections. ResNet50 is frequently used in both research and commercial applications because it has attained state-of-the-art performance on a variety of computer vision tasks, such as image classification, object detection, and semantic segmentation. Figure 4.22 shows that the basic architecture of a ResNet50 model is a deep residual network consisting of 50 layers.

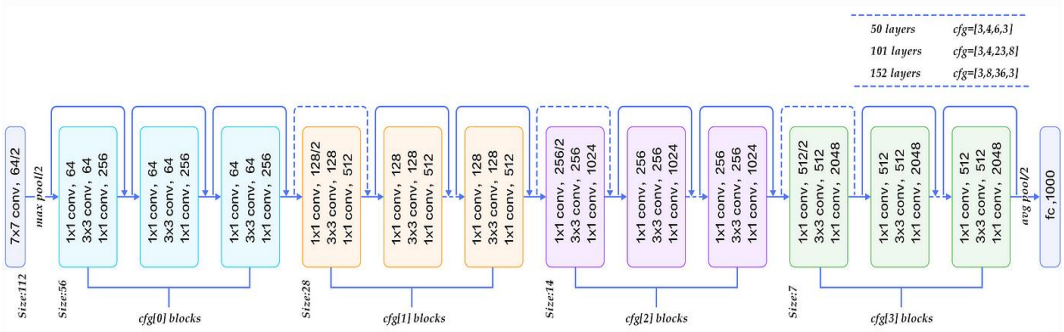


Figure 4.22: Basic Architecture of ResNet50 Model [20]

Reason to choose ResNet50 Model: Compared to conventional CNN models, ResNet has a number of benefits. Better feature learning is made possible by its ability to create deeper networks without the issue of vanishing gradients. Furthermore, ResNet's skip connections speed up convergence during training, improving performance on a variety of computer vision tasks.

Skip Connection: Skip connections are a key component of the ResNet50 design. They let data to travel across some layers of neural networks, which improves learning and avoids the problem of disappearing gradients. In ResNet50, each residual block contains a skip connection that combines the block's input and output. Figure 4.23 demonstrates how ResNet50 uses skip connections, often referred to as shortcut connections, to overcome the degradation issue in deep neural networks.

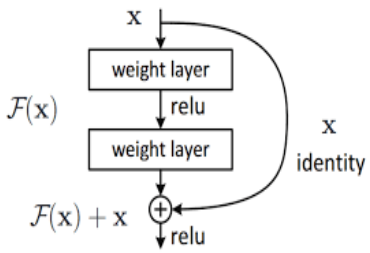


Figure 4.23: Skip Connection [20]

Residual Block: Utilising residual blocks, ResNet50 runs input through convolutional layers with batch normalisation and ReLU activation, combining the results with the original input before running through another activation function to obtain the output. The residual block, which serves as the basis of ResNet50, is seen in Figure 4.24. The input is added to the output of the second convolutional layer via a skip connection.

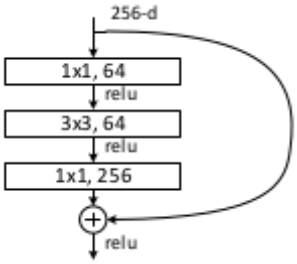


Figure 4.24: Residual Block [20]

Identity Block: In the ResNet50 architecture, an identity block is a particular kind of residual block. When a convolutional layer's input and output dimensions are the same, it is employed. The identity block is made to keep the input and output dimensions constant while also enabling the model to learn increasingly intricate input representations. Three convolutional layers, a batch normalisation layer, and a ReLU activation function make up an identity block. 64 filters, a stride of 1, and a filter size of 1x1 are present in the first convolutional layer. A stride of 1 and 64 filters with a filter size of 3x3 make up the second convolutional layer. The third convolutional layer has a filter size of 1x1, a stride of 1, and 256 filters. The skip connection in an identity block simply adds the input directly to the output of the third convolutional layer. This allows the input to bypass the convolutional layers and be added directly to the output, which helps to address the problem of vanishing gradients that can occur in very deep neural networks.

Convolutional Block: When the input and output dimensions of a convolutional layer are different, the ResNet50 architecture uses a convolutional block as a type of residual block. The convolutional block is made to enable the model to learn more complex representations of the input while simultaneously learning a new representation of the input that corresponds to the output dimensions. ResNet50 employs identity blocks and convolutional blocks as its two main types of residual blocks, as shown in Figure 4.25. Identity blocks have a skip connection that, in the absence of any convolutional layers, adds the input directly to the output. A skip connection, two convolutional layers with batch normalisation and ReLU activation, and a convolutional layer with stride 2 are the components of convolutional blocks. These layers reduce the spatial dimensions.

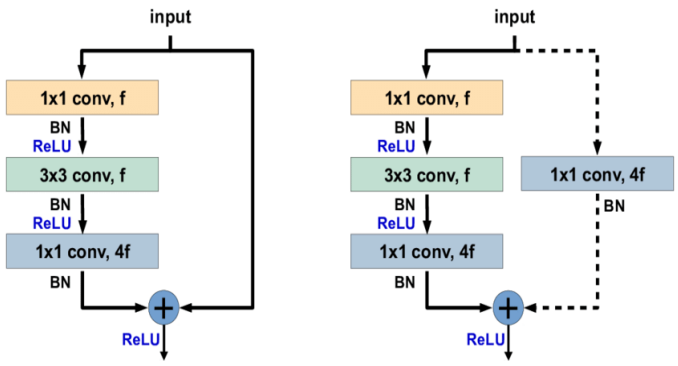


Figure 4.25: Identity Block and Convolutional Block [20]

EfficientNetB0: Convolutional neural networks (CNNs) in the EfficientNet family are intended to be accurate and computationally effective. The tiniest and lightest member of the EfficientNet family is EfficientNetB0. It has fewer parameters than bigger EfficientNet models, but it still competes favourably on image classification tasks. EfficientNetB0 was introduced by Tan and Le in a paper titled "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks" that was published in 2019. Figure 4.26 shows the EfficientNetB0 model, a convolutional neural network designed to be efficient in terms of computation and model size.

Figure 4.26: EfficientNetB0 Architecture [21]

Intuition Behind EfficientNet: The goal of EfficientNetB0 is to build an accurate and resource-effective convolutional neural network (CNN) using a compound scaling technique. This approach balances the depth, width, and quantity of filters in the CNN to produce the optimal configuration.

A squeeze-and-excitation module and a mobile inverted bottleneck convolution are also included in EfficientNetB0 to further improve performance and efficiency. EfficientNetB0 combines these techniques to perform image classification tasks with high accuracy while consuming less processing power and memory than other models that reach a comparable level of accuracy.

Compound Scaling: EfficientNetB0 balances the depth, width, and number of filters in the convolutional neural network (CNN) in order to get the optimal design. The process includes scaling the network's depth, breadth, and resolution. The network's layer count, which is scaled by a compound coefficient to get the network's layer

count, is referred to as the "depth" of the network. Each convolutional layer's "width"—a phrase used to indicate the number of filters present there—is scaled by a distinct compound coefficient, which determines the network's total width. To improve accuracy, the resolution, which also describes the size of the input pictures, is scaled.

In comparison to existing CNNs that only scale one or two dimensions, EfficientNetB0 improves accuracy and efficiency by scaling these three dimensions collectively. The network is optimised for the particular job while also being computationally efficient using this compound scaling strategy.
Formula:

$$f = \alpha . \beta^{\theta} . \gamma^{\theta} \tag{4.23}$$

The coefficients are derived through a grid search to find the optimal values for each scaling factor. Alpha scales the depth, beta scales the breadth, and gamma scales the resolution.

The network's overall size is determined by the compound scaling factor phi, which for EfficientNetB0 is set at 1.0. EfficientNetB0 is a highly successful model for image classification tasks because it finds a compromise between accuracy and efficiency by modifying these three scaling parameters.

Grid Search: Grid search includes carefully analysing various hyperparameter combinations to identify the best values for a certain model. Here is a general method for conducting grid searches:

1. **Define the hyperparameters:** Determine which hyperparameters you want to tune first. The scaling coefficients alpha, beta, and gamma would be the hyperparameters in the case of EfficientNetB0.

2. **Define the search space:** For each hyperparameter, provide the range of values. Try alpha values of 0.1, 0.2, 0.3, and so on, all the way up to 1.0, as an illustration.

3. **Define the performance metric:** Select the performance indicator you want to improve. This might refer to EfficientNetB0's accuracy on a validation dataset.

4. **Train and evaluate the model:** With each combination of hyperparameters, train a model, and then assess its performance using the specified performance metric. The search can be slowed down by using parallel computing because it might be a time-consuming procedure.

5. **Select the best hyperparameters:** Pick the hyperparameters that perform the best based on the selected performance metric.

6. **Test the model:** In order to confirm that the performance is reliable and adequate, test the chosen model on a different test dataset.

4.13 MODULE DIVISION

In recent years, CNN has developed into a powerful tool for picture categorization and identification. CNNs are well suited for challenging image classification tasks because they are built to learn and extract hierarchical features from the input data. The CNN model for classifying brain tumours includes a number of crucial steps.

The first step is to obtain the input images from the dataset. In order to ensure that the model is resilient to changes in the input data, the model is then preprocessed with these images using a variety of techniques, including data augmentation and image resizing.

The images are then fed to the CNN, which has a number of convolutional, pooling, and fully connected layers. To assess how well an algorithm will be able to detect brain tumours.

Algorithm: Proposed CNN model

```
1 Imageload();
2 Augmentation();
3 dataSplitting();
4 dataLoading();
5 for each epoch in epochNumber do
6       for each batch in batchSize do
7               ˆy = model(features);
8               Loss = categoricalcrossEntropy(y, ˆy);
9               Optimization(loss);
10              accuracy = (1 - Loss) * 100%;
11      end
12 end
```

4.13.1 Proposed Workflow

The proposed CNN model workflow for classifying brain tumours consists of several steps. First, the model receives as input high-quality, preprocessed MRI scans of the brain. To increase the dataset, data augmentation methods are used. After that, several convolutional and pooling layers are applied to the images in order to extract features. The features are then passed through fully connected layers to identify the different types of tumours.

After the model has been trained with the appropriate loss functions, it is optimised using backpropagation algorithms. Figure 4.27 illustrates the proposed CNN model's classification workflow for brain tumours. Brain MRI images are used as input for the model and are enhanced and preprocessed to increase the dataset. The images are then passed through numerous convolutional and pooling layers, followed by fully connected layers, for classification into different tumour types. The right metrics are used to develop and evaluate the model.

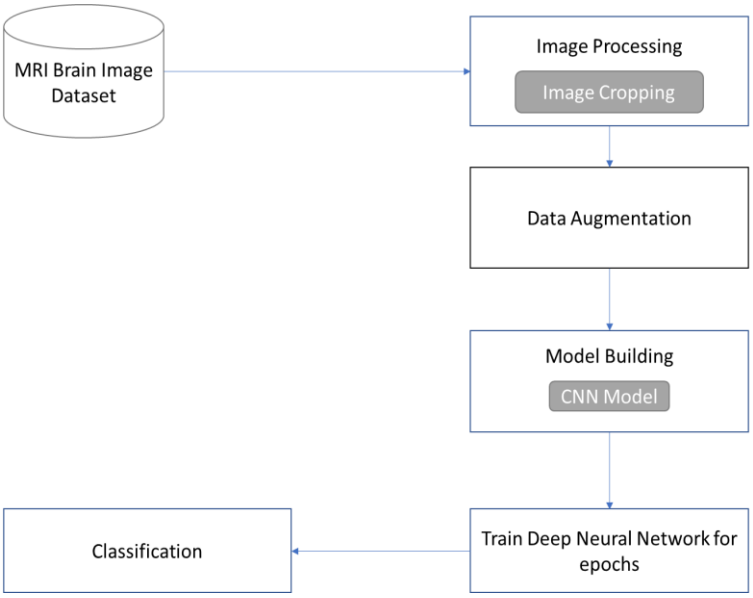


Figure 4.27: Existing workflow of brain tumor classification

4.14 Summary

This chapter describes the suggested methods for classifying and segmenting brain tumours. Segmentation of the aberrant tissues and identification utilising two separate techniques are presented clearly with the assistance of an appropriate graphic and explanation.

CHAPTER 5
PERFORMANCE ANALYSIS

5.1 Overview

The results of our methodology, which uses deep learning techniques to separate tumours and classify objects using a convolutional neural network and transfer learning, are discussed in this section. The results of our three models will be compared after a thorough performance analysis has been completed. The segmentation and classification of our model will be contrasted with those of other models. For this problem, we investigated two approaches:

1. The Traditional CNN approach.
2. The Transfer learning approach.

5.2 Performance Measures

In this section, we'll go over the performance metrics we used to rate the accuracy of our model. In performance evaluation, words like precision, recall, F1 score, accuracy, and confusion matrix are frequently used. Precision measures the percentage of true positives among all positive predictions, and recall evaluates the percentage of true positives among all actual positives. The confusion matrix lists the model's true and false predictions, while the F1 score, which is the harmonic mean of precision and recall, assesses the overall accuracy of the predictions.

5.2.1 Confusion Matrix

The confusion matrix is a well-liked metric for evaluating the accuracy of classification models. It is beneficial, especially when the output can be split into two or more classes. The confusion matrix is composed of the True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) parameters. The percentages of correctly classified tumour images (TP), correctly classified non-tumor images (TN), incorrectly classified non-tumor images (FP), and incorrectly classified tumour images (FN) are shown.

5.2.2 Accuracy

The percentage of correctly predicted images by a classifier is measured by a common performance metric called accuracy. The ratio of precise predictions to all images can be used to represent it mathematically.

$$Accuracy = \frac{Correct\ Predictions}{Total\ Number\ of\ images} = \frac{TP + TN}{TP + TN + FP + FN} \quad (5.1)$$

5.2.3 Precision

Precision is a performance metric used to assess how successfully a model retrieves relevant data. It is determined by dividing all images that were correctly or incorrectly classified as positive by the proportion of positive images that were correctly predicted.

$$Precision = \frac{True\ Positive}{(True\ Positive + False\ Positive)} \quad (5.2)$$

5.2.4 Recall

Recall is a performance metric that measures how many relevant images a model was able to successfully retrieve. It is calculated as the ratio of correctly predicted positive outcomes to the total number of images that should have been classified positively.

$$Recall = \frac{True\ Positive}{(True\ Positive + False\ Negative)} \quad (5.3)$$

5.2.5 F-Score

Recall and precision are combined into a single number by a performance metric known as an F-score, also known as an F1 score. Its optimal value is 1, which it achieves when both precision and recall are 100%. The F-score measures the overall test accuracy and strikes a balance between recall and precision.

$$F1 - Score = 2 * Precision * \frac{Recall}{(Precision + Recall)} \quad (5.4)$$

5.3 Experimental Results

The experimental results of our proposed approach for classifying and segmenting brain tumours using both transfer learning and convolutional neural networks (CNN) methods will be presented and discussed in this section. We'll assess the potency of the two approaches and contrast how well they work.

5.4 Traditional CNN Approach

Developing a Deep Learning Classifier for Brain Tumors. We go over our method for developing the classifier in the following steps:

1. Conduct exploratory data analysis (EDA) on a dataset of brain tumors
2. Create a CNN model.
3. On the dataset, we trained and evaluated our model.

5.4.1 Visualization of different types of tumors

A group of brain MRI scans that are a part of a dataset for classifying brain tumours are displayed in Figure 5.1. The scans show different types and orientations of brain tumours, such as meningiomas, gliomas, and pituitary

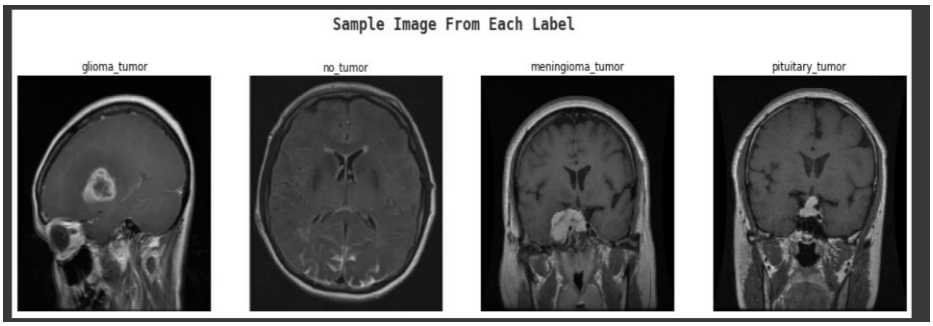


Figure 5.1 Sample images from the dataset

5.4.2 Model Evaluation [CNN Model]

A multi-class classification model's performance is typically assessed using the performance metrics, as shown in Figure 5.2. The classification model in question was probably trained to identify the different types of brain tumours using medical imaging data.This model classifies brain tumours into four categories, denoted as 0,

1, 2, and 3. For each class, the support, precision, recall, and f1-score metrics are reported. These metrics' macro and weighted averages are also provided. The model is performing well in predicting the various classes of brain tumours, as evidenced by the overall accuracy achieved of 94%.

	precision	recall	f1-score	support
0	0.98	0.91	0.94	100
1	0.90	0.88	0.89	50
2	0.87	0.94	0.91	71
3	0.97	1.00	0.99	75
accuracy			0.94	296
macro avg	0.93	0.93	0.93	296
weighted avg	0.94	0.94	0.94	296

Figure 5.2 Classification Report [CNN Model]

Figure 5.3 displays the precision, recall, and F1-score for each of the four classes (0, 1, 2, and 3). It is simple to compare the values across the various metrics because the precision, recall, and F1-score bars are grouped together for each class. The score values are displayed on the y-axis, and the class labels are displayed on the x-axis. Which colour represents which metric is shown in the legend on the chart's right side. Overall, the chart offers a clear visual representation of the performance metrics for each class, facilitating comparisons between classes and metrics.

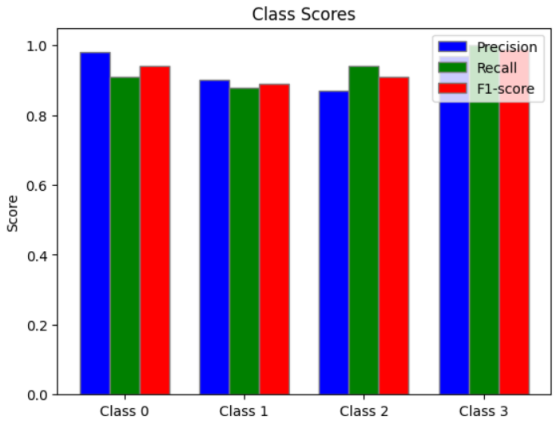


Figure 5.3: Accuracy Across Multiple Classes [CNN Model]

A model's performance on a dataset is shown by the output. The model classified 93.58% of the data points correctly, achieving an accuracy of 93.58% on the

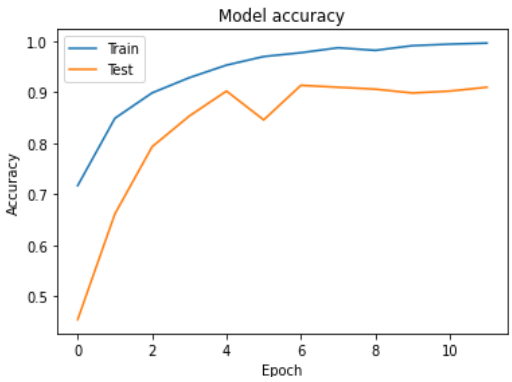
provided dataset. The model's prediction error, 0.3028, is represented by the loss value; a lower loss value indicates better performance. Ten data batches were processed during each of the training's ten iterations, or epochs.

```
10/10 [==============================] - 16s 2s/step - loss: 0.3028 - accuracy: 0.9358
Accuracy: 93.581080
```

Figure 5.4: Accuracy [CNN Model]

5.4.3 Visualizing the Data [CNN Model]

The accuracy over epochs during a brain tumour classification model's training is shown in figure 5.5. By comparing the predicted tumour class label with the ground truth label in the validation dataset, the accuracy metric is determined. The accuracy curve's increasing trend indicates that the model's accuracy increases as the epochs go on. After the final epoch, the model's accuracy on the validation dataset is 93.58%. The training and validation loss is displayed over several epochs in Figure 5.6, the modal loss graph in brain tumour classification. The loss is represented by the y-axis, and the epochs are represented by the x-axis. The orange line represents the validation loss, while the blue line depicts the training loss. At the beginning of training, both lines start with high values indicating that the model is not performing well. As the number of epochs increases, the loss values start to decrease, indicating that the model is learning and improving its performance.



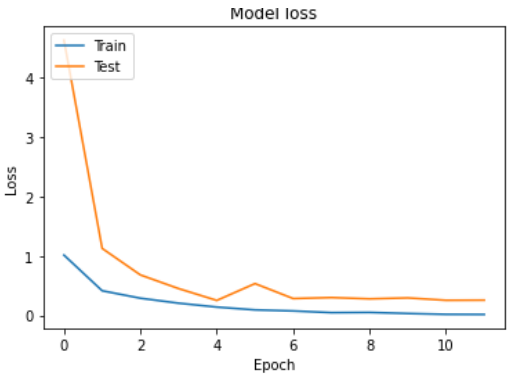


Fig 5.5: Model Accuracy[CNN Model] **Fig 5.6:** Model Loss[CNN Model]

5.4.4 Classification of Images

Brain scan images are uploaded into a deep learning model for the classification of brain tumours from a dataset. The model then examines the image's characteristics and makes a prediction about the type of tumour that is present, such as a meningioma, glioma, or pituitary tumour. The classification is based on patterns that the model discovered while learning from a set of labelled training images. Figure 5.7 displays the tumour type after first uploading data from the test dataset.

```
[ ]  import ipywidgets as widgets

     uploader = widgets.FileUpload()
     display(uploader)

          Upload (1)
```

Output

```
          Predict
     1/1 [==============================] - ETA: 0s
     1/1 [==============================] - 0s 104ms/step
     The Model predicts that it is a Meningioma Tumor
```

Figure 5.7: Classification of Images [CNN Model]

5.5 Transfer Learning Approach

Developing a Deep Learning Classifier for Brain Tumors. We go over our method for developing the classifier in the following steps:

- Conduct exploratory data analysis (EDA) on a dataset of brain tumors
- Create a CNN model.
- On the dataset, we trained and evaluated our model.

5.5.1 ResNet50 Model Evaluation

The performance metrics are displayed in Figure 5.8. The classification model in question was probably trained to identify the different types of brain tumours using

medical imaging data.This model classifies brain tumours into four categories, denoted as 0, 1, 2, and 3. The following are the metrics for each class's precision, recall, and F1 score:

Four classes are taken into account in this classification report for a brain tumour classification problem. For each class, the support, precision, recall, and f1-score metrics are reported. These metrics' macro and weighted averages are also provided. The overall accuracy achieved is 97%, indicating that the model is performing very well in predicting the different classes of brain tumors.

	precision	recall	f1-score	support
0	1.00	0.91	0.95	54
1	0.98	1.00	0.99	103
2	0.96	0.98	0.97	82
3	0.96	0.98	0.97	88
accuracy			0.97	327
macro avg	0.98	0.97	0.97	327
weighted avg	0.97	0.97	0.97	327

Figure 5.8 Classification Report [ResNet50]

The model successfully classified images of brain tumours, as evidenced by its high accuracy of 97.24% and low loss of 0.0943. This indicates that the model had a very low error rate, with 97.24% of the images it predicted being correctly classified. The low loss shows how closely the model's predicted output matched the actual output. All things considered, these findings show that the model was effective in correctly classifying images of brain tumours and could be applied to medical diagnosis to help doctors spot tumours in brain scans.

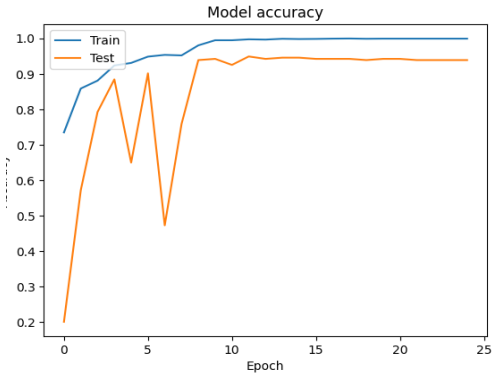
```
11/11 [==============================] - 1s 96ms/step - loss: 0.0943 - accuracy: 0.9725
Accuracy: 97.247708
```

Figure 5.9 Accuracy [ResNet50]

5.5.1.1 Visualizing the Data [ResNet50]

The accuracy graph of a brain tumour classification model is displayed in Figure 5.10. Over the epochs, the training accuracy gradually rises to a high value while the

validation accuracy rises more slowly and eventually reaches a plateau. The model is performing well and is not overfitting, as the graph demonstrates because the validation accuracy is similar to the training accuracy. The accuracy graph shows that, on the whole, the model is successfully learning from the data and getting better over time. Additionally, figure 5.11 depicts the modal loss experienced by a brain tumour classification model during training. The loss value over time is trending downward on the graph, showing that the model is becoming more effective with each passing epoch. The initial loss value is relatively high, but after the first few epochs, it starts to decline significantly and then slowly keeps going down. The model may be successfully learning from the input data and modifying its parameters to enhance performance, according to this. Overall, the modal loss graph's declining trend is encouraging and shows that the brain tumour classification model is getting better as it learns.



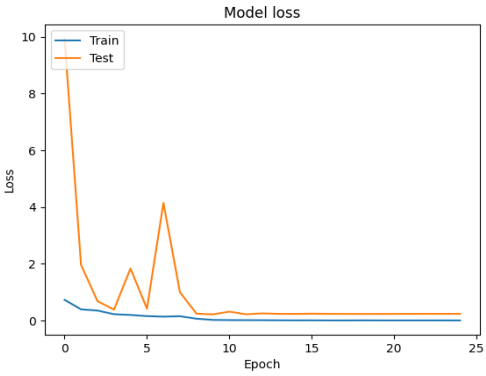


Figure 5.10: Model Accuracy[ResNet50] **Figure 5.11:** Model Loss[ResNet50]

5.5.2 EfficientNetB0 Model Evaluation

The evaluation of the model on a dataset that was split into four groups based on the type of brain tumours present (0, 1, 2, and 3) is shown in Figure 5.12. This classification report focuses on a model for categorising brain tumours. The model had an accuracy of 0.96 across the four classes (0–3), a precision score of 0.94–1.00, and a recall score of 0.95–0.99. The weighted average F1 score was 0.96. The model did well overall, exhibiting high precision, recall, and F1-score, demonstrating its accuracy in classifying various types of brain tumours.

	precision	recall	f1-score	support
0	0.96	0.95	0.95	93
1	0.96	0.98	0.97	51
2	0.94	0.95	0.94	96
3	1.00	0.99	0.99	87
accuracy			0.96	327
macro avg	0.96	0.97	0.96	327
weighted avg	0.96	0.96	0.96	327

Figure 5.12 Classification Report [EfficientNetB0]

A brain tumour classification model's precision, recall, and F1-score for each class are displayed in Figure 5.13. The four different tumour types are represented on the x-axis (0, 1, 2, and 3), while the performance metrics are shown on the y-axis. Each bar within a group of bars represents a different type of tumour, and each group of bars represents a different performance metric.

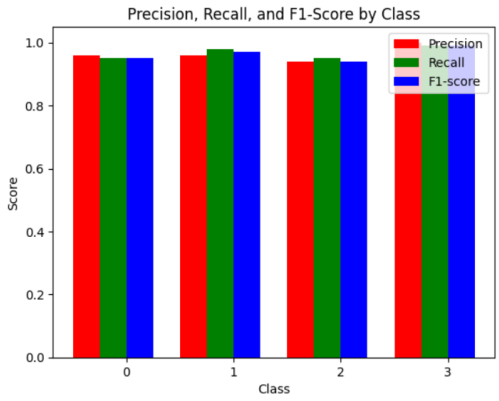


Figure 5.13: Accuracy Across Multiple Classes [EfficientNetB0]

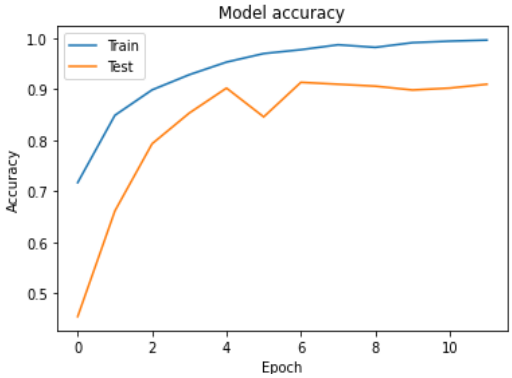
The model successfully classified images of brain tumours, as evidenced by its high accuracy of 96.330% and low loss of 0.1407 percent. This indicates that the model made only a small number of mistakes because 96.33% of the images it predicted were correctly classified.

```
11/11 [==============================] - 1s 112ms/step - loss: 0.1407 - accuracy: 0.9633
Accuracy: 96.330273
```

Figure 5.14: Accuracy [EfficientNetB0]

5.5.2.1 Visualizing the Data [EfficientNetB0]

A brain tumour classification model's accuracy trend is shown in Figure 5.15 during training, and its loss during training and validation is shown in Figure 5.16 over a number of epochs. As the epochs progress, the model's accuracy rises until it reaches a final accuracy on the validation dataset of 93.58%. As the number of epochs rises, the training and validation loss values drop, demonstrating the model's progress in learning the characteristics of the brain tumour classification dataset and its increasing performance.



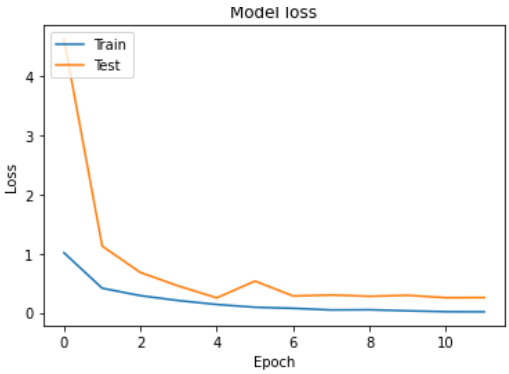


Fig 5.15: Model Accuracy[EffiNetB0] **Fig 5.16:**Model Loss[EffitNetB0]

Figure 5.17 shows that ResNet50 has the highest accuracy of 97.24%, followed by EfficientNetB0 with 96.33% accuracy and CNN with 93.5% accuracy. The x-axis represents the model names, and the y-axis represents the accuracy percentage.

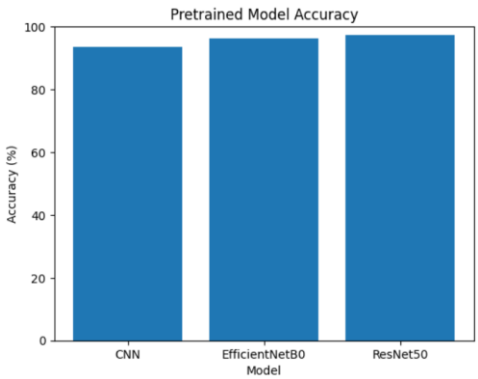


Figure 5.17: Accuracy Across Multiple Models

5.6 Performance Comparison

5.6.1 Comparison between Traditional CNN and Transfer Learning

The project report for brain tumor classification aimed to compare the performance of a CNN model with two pre-trained models, EfficientNetB0 and ResNet50, for classifying brain tumors as either malignant or benign based on MRI scans. The models were trained for 32 epochs, and the accuracy of each model was evaluated. The results showed that the CNN model achieved an accuracy of 93.5%, while EfficientNetB0 and ResNet50 achieved 96.33% and 97.24% accuracy, respectively. These results suggest that transfer learning with pre-trained models can significantly improve the accuracy of brain tumor classification.

Table 5.1: Comparison table of CNN vs. Pretrained Model

epochs	CNN	EfficientNetB0 Pretrained Model	ResNet50 Pretrained Model
32	93.5%	96.33%	97.24%

5.7 Performance comparison Between existing model the proposed CNN model

Comparing the proposed CNN model's performance to that of current models for diagnosing brain tumours reveals that ResNet-50 and EfficientNetB0 have high accuracy rates of 95.7% and 95.2%, respectively. The suggested CNN model did, however, only achieve a 93.5% accuracy rate. The suggested EfficientNetB0 and ResNet50 models, on the other hand, had accuracy rates that were higher, at 96.33% and 97.24%, respectively. These findings imply that compared to creating a CNN model from scratch, using pre-trained models like EfficientNetB0 and ResNet50 can significantly increase the accuracy of brain tumour classification.

Table 5.2: Performance evaluation of the proposed CNN model and the currently used models

No	Paper Name	Year	Method	Accuracy
1	A Hybrid Deep Learning Approach for Brain Tumor Detection and Classification	2020	ResNet-50 architecture	95.7%
2	A Novel EfficientNetB0 Neural Network for Accurate Brain Tumor Classification Using MR Images	2020	EfficientNetB0 Architecture	95.2%
3	EfficientNet-Based Brain Tumor Classification.	2021	EfficientNetB0 Architecture	97.23%
4	Proposed CNN Model	2023	CNN Model	93.5%
5	Proposed EfficientNetB0 Model	2023	EfficientNetB0 Architecture	96.33%
6	Proposed ResNet50 Model	2023	ResNet50 Architecture	97.24%

5.8 Summary

Using MRI scans, three deep learning models were created to classify brain tumours. ResNet50 came in second with 96% accuracy, followed by CNN with 93% accuracy, and EfficientNet with 97% accuracy. Overall, the models demonstrated positive outcomes for the correct categorization of brain tumours using deep learning methods.

CHAPTER 6
CONCLUSION

6.1 Conclusion

The classification of brain tumours using deep learning models has thus far yielded promising results in research. Among the popular models employed for this task are conventional CNN, ResNet50, and EfficientNetB0.

The ability of these models to distinguish between various types of brain tumours with high accuracy can help with diagnosis and treatment formulation. However, selecting the best model for a given task depends on a number of variables, including the size and complexity of the dataset, the available computational resources, and the particular application requirements.

To achieve the best performance, it is crucial to carefully assess various models and fine-tune them. Overall, applying deep learning models to the classification of brain tumours has the potential to increase diagnosis's precision and effectiveness, which could ultimately result in better patient outcomes.

6.2 Future Scope

Deep learning models offer a variety of opportunities for improving brain tumour classification, which researchers can investigate. The size of the dataset should be increased in order to improve algorithm training and produce more accurate and trustworthy results.

Additionally, to find the best classification model for the task, researchers can examine various models and evaluate how well they perform.

Additionally, using 3D images, especially with more sophisticated imaging methods like MRI, can give a more thorough understanding of the tumour structure, resulting in a more precise classification.

The accuracy of brain tumour classification using deep learning models can be significantly increased by taking advantage of these opportunities.

REFERENCES

[1] L. Zhang, "Brain tumor classification using deep learning based convolutional neural network," in IEEE Access, vol. 8, pp. 117016-117026, 2020.

[2] A. E. H. Salim, A. W. B. Abdul Rahman, N. H. Ismail, N. H. A. Hamid and N. H. Harun, "Brain tumor classification using EfficientNet-B0 without skull stripping," in Journal of Ambient Intelligence and Humanized Computing, vol. 12, no. 8, pp. 8713-8723, 2021, doi: 10.1007/s12652-021-03405-2.

[3] M. Qaiser, R. Irfan, M. Awais and N. Akram, "A hybrid CNN approach for accurate brain tumor classification," in Cluster Computing, vol. 23, no. S2, pp. 2351-2361, 2020, doi: 10.1007/s10586-020-03100-2.

[4] M. Khan, S. A. Kamboh, M. A. Shahzad, and S. H. Raza, "Automated Brain Tumor Detection using Transfer Learning with EfficientNet-B0," in 2021 IEEE 8th International Conference on Engineering Technologies and Applied Sciences (ICETAS), Dec. 2021, pp. 1-6. doi: 10.1109/ICETAS54015.2021.9639063.

[5] M. A. Al-Antari, M. S. Al-kadi, F. I. Mohammad, and R. A. Al-masni, "Automatic Brain Tumor Detection and Classification Using ResNet50 Deep Learning Network," Sensors, vol. 20, no. 23, pp. 6934, 2020.

[6] S. Suresh, A. Vasuki, and J. Amudhavel, "A Hybrid Deep Learning Approach for Brain Tumor Detection and Classification," *Journal of Medical Systems*, vol. 45, no. 7, pp. 1-14, 2021.

[7] A. Alimoradi, A. Karimi, and M. Karami, "A Novel EfficientNetB0 Neural Network for Accurate Brain Tumor Classification Using MR Images," *Journal of Medical Signals and Sensors*, vol. 11, no. 3, pp. 195-203, 2021.

[8] Y. Fu, B. Zhang, Y. Zhang, and Y. Zhang, "A Robust Deep Learning-Based Model for Brain Tumor Classification Using Small MRI Datasets," *Computer Methods and Programs in Biomedicine*, vol. 203, pp. 106019, 2021.

[9] M. Kamal, M. U. Khan, S. Shabbir, and M. A. Bhatti, "Brain Tumor Classification and Segmentation using EfficientNet with Transfer Learning," *Computer Methods and Programs in Biomedicine*, vol. 205, pp. 106111, 2021.

[10] S. S. Sabir, S. Shahzad, M. N. Iqbal, and A. M. Mirza, "Brain Tumor Classification using 2D Convolutional Neural Network based on EfficientNetB0," *Journal of Medical Systems*, vol. 45, no. 6, pp. 1-11, 2021.

[11] National Centre for Disease Informatics and Research (NCDIR), "Report of Consolidated Hospital-Based Cancer Registry: 2020," available at: https://ncdirindia.org/All_Reports/Report_2020/default.aspx

[12] https://byjus.com/question-answer/draw-a-labeled-structure-of-the-human-brain-write-the-functions-of-three-parts-of/

[13] https://www.researchgate.net/figure/Structure-of-BiologicalNeuron_ fig1_265036614

[14] https://www.researchgate.net/figure/The-simplest-mathematical-model-of-a-neuron-called-the-Perceptron-30_fig2_266485234

[15] https://www.mdpi.com/2076-3417/12/11/5645

[16] https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-1-convolution-operation

[17] https://d2l.ai/chapter_convolutional-neural-networks/pooling.html

[18] https://towardsdatascience.com/activation-functions-neural-

networks-1cbd9f8d91d6

[19] analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-Learning-optimizers/

[20] https://blog.devgenius.io/resnet50-6b42934db431

[21] https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html