**People's Democratic Republic of Algeria**

**Ministry of Higher Education and Scientific Research**

**University M'Hamed BOUGARA- Boumerdes**


**Institute of Electrical and Electronic Engineering**

**Department of Electronics**

Final Year Project Report Presented in Partial Fulfilment of

The requirement for the degree of

# Master

In **Electrical and electronic engineering**

Option: **Computer engineering**

Title:

## Brain Tumor Classification Using Deep Learning

Presented by:

BERRICHI Ryad

Supervisors:

Dr. NAMANE Rachid

Dr. BOUTELLAA Elhocine

Registration Number: 2022

# Dedication

This work is wholeheartedly dedicated to my parents, Salima and Ali, for their constant support and unconditional love. I am forever grateful for their efforts in providing me with moral, spiritual, and emotional support.

To my beloved sisters, Ryma and Maria, for being here for me and always supporting me throughout the process.

To all my friends and my family members

To my dear, Assia Bougrid, I am grateful for our paths crossing

To my grandmother, Allaoua Fatima, who did not get to see me becoming a grown man completing my thesis after she passed away due to cancer in 2012. In life, we loved you dearly, In death, we love you still. In our hearts you hold a place, no one else will ever fill.

To my grandfather, Berrichi Mohammed Rabah, who was a role model in my life as he taught me life values and how to be a real man. A heart that is made of pure gold has stopped beating, one of the most hardworking hands have been put to rest.

To all those who believe in me Thank you

# Acknowledgment

First and foremost, I would like to extend our deepest gratitude to God Almighty who provided me with his blessing and the opportunity to successfully conclude this project.

In the accomplishment of our final year project - Brain Tumor Classification Using Deep Learning -, I would like to express my deepest gratitude to my supervisors Dr. NAMANE Rachid and Dr. BOUTELLAA Elhocine for guiding me in further improving the project as well as the report.

A special thanks to my friend and colleague AFFOUN Nedjem Eddine for his valuable time spent providing insights and mentoring.

# Abstract

Brain tumors are a common type of cancer that affects brain tissue. They often cause symptoms such as headaches or seizures. They are usually diagnosed through brain scans such as magnetic resonance imaging (MRI). In recent years, computer scientists have developed algorithms that have shown promising results in automatically classifying these images into various types using deep learning models, which is a type of machine learning that uses artificial neural networks to recognize patterns in data.

Publicly available MRI scans (1500 cancerous and 1500 non-cancerous) are used to train deep learning models: VGG16, VGG19, ResNet50, and Xception. Each model is implemented using three approaches, namely: implementation from scratch, transfer learning, and fine-tuning. This comparative study aims to find the best approach for training models on small datasets. The obtained overall accuracies ranged from 88% to 99%.

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| ADAM | Adaptive Moment Estimation |
| ANN | Artificial Neural Networks |
| CNN | Convolutional Neural Networks |
| CT | Computed Tomography |
| DL | Deep Learning |
| DNN | Deep Neural Networks |
| GPU | Graphical Processing Unit |
| ML | Machine Learning |
| MRI | Magnetic Resonance Imaging |
| MRS | Magnetic Resonance Spectroscopy |
| PET | Position Emission Tomography |
| RELU | Rectified Linear Unit |
| SGD | Stochastic Gradient Descent |
| SPECT | Single-Photon Emission Computed Tomography |
| TN | True Negative |
| TP | True Positive |
| FN | False Negative |
| FP | False Positive |
| WHO | World's Health Organization |

# Introduction

A brain tumor is an abnormal growth of cells or mass in the brain leading to nervous system dysfunctions. It can be benign (non-cancerous) or malignant (cancerous), and it can be categorized into many types according to its size, location, origin, and growth rate. A brain tumor is a deadly disease having high mortality in all age groups. It is the third leading cause of cancer death in children and young adults [1].

Many medical imaging techniques are used to provide information about the location, size, shape, and type of brain tumor e.g. Position Emission Tomography (PET), Single-Photon Emission Computed Tomography (SPECT), Computed Tomography (CT), Magnetic Resonance Imaging (MRI), and Magnetic Resonance Spectroscopy (MRS). Among them, MRI imaging is the most popular and efficient technique due to the rich information it provides about the anatomy of human tissues, and due to its widespread availability and soft tissue contrast [2].

Machine learning and computer vision breakthroughs have paved the way for new inventions and algorithms development. It has demonstrated outstanding performance and can be used in a variety of areas, which captured the attention of researchers, particularly in the domain of anomaly detection and disease classification. Deep learning is a subfield of machine learning that has shown great and promising results in medical image classification.

The chance of recovery from brain tumors depends on the early detection of the brain tumor. To improve the survival rate, doctors and radiologists need to accurately detect the tumors. Deep learning has become a powerful tool for medical imaging analysis by contributing to the development of robust computer-aided diagnosis systems, which reduce the cost and time taken to detect brain tumors.

The objective of the proposed work is to participate in increasing the performance of early detection of brain tumors, help reduce cancer mortality by assisting radiologists in interpreting images at a much faster rate, and improve accuracy. This work aims to provide a comparative study of some of existing approaches and models for developing deep learning-based computer-aided systems for brain tumor classification. For this purpose, we used a dataset of MRI images obtained from the KAGGLE platform. It consists of MRI scans labeled cancerous and non-cancerous.

This report is structured as follows. The first chapter discusses the theoretical background including brain tumors, medical imaging techniques, deep learning theory, and related works. The second chapter introduces the dataset used in this work, the investigated models used for our image classification problem, and the used tools. The third chapter provides and discusses the obtained results. Finally, the report is ended with a general conclusion followed by future works and improvements.

# Chapter 1

# Theoretical Background and Related Works

## 1.1 Introduction

This chapter introduces brain tumors, medical imaging, and deep learning theory. First, It presents the anatomy of the brain and discusses brain tumors along with their symptoms, statistics, and types. Next, it provides an overview of medical imaging techniques used to diagnose brain tumors, where it focuses on the MRI technique used methodology. Additionally, it presents the theory behind deep learning, convolutional neural networks, and fine-tuning. Finally, it discusses works related to brain tumor classification using deep learning techniques.

## 1.2 Brain Tumor Overview

### 1.2.1 Brain Anatomy

The brain is a complex organ that supervises all the functions of the body [3]. It is made up of more than 100 billion nerves that communicate in trillions of connections called synapses [4]. The human brain is divided into different parts responsible for different nervous functions. It receives much information at one time through the five senses: sight, smell, touch, taste, and hearing. It assembles the received information and stores it in our memory. The brain controls thoughts, memory, speech, movement, and the function of organs within the human body [3]. The cerebrum, cerebellum, and brainstem are the three main parts that make up the brain, as depicted in Fig 1.1:

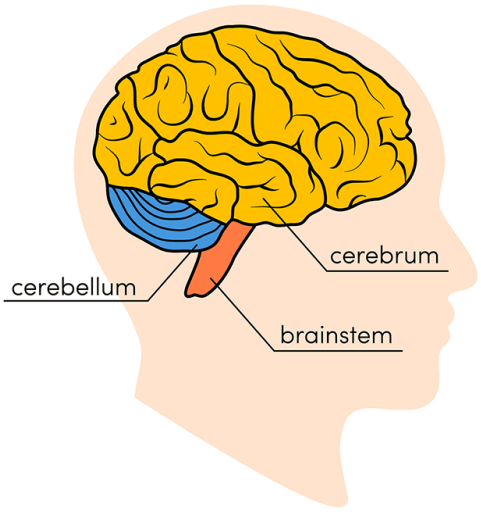


Figure 1.1: The three main parts of the brain [5].

**Cerebrum:** The cerebrum is the largest region of the brain, it controls movement and temperature. Speech, judgment, thinking, reasoning, problem-solving, emotions, and learning are all enabled by several parts of the cerebrum. Vision, hearing, touch, and other senses are covered by other functions. Each hemisphere (part of the cerebrum) of the brain is divided into four lobes: frontal, parietal, temporal, and occipital. As shown in Fig 1.2.

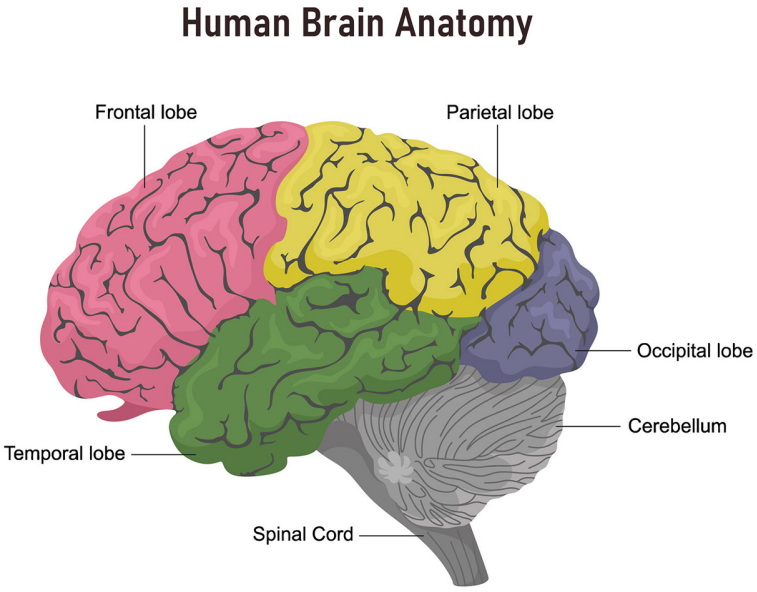


Figure 1.2: The four lobes of the cerebrum [5].

**Frontal lobe:** The frontal lobe is the brain's hugest lobe, located in the front of the head. It is responsible for personality traits, decision-making, and movement. Parts of the frontal lobe are frequently involved in recognizing smells. Broca's area is located in the frontal lobe and is related to speaking skills.

**The parietal lobe:** The parietal lobe, located in the middle of the brain, assists in the identification of objects and spatial relationships. It is also involved in pain and touch perception in the body. Wernicke's region, which helps the brain understand spoken language, is located in the parietal lobe.

**The occipital lobe:** The occipital lobe, located in the back of the brain, is responsible for vision.

**The temporal lobe:** The temporal lobes are located on the sides of the brain. They assist short-term memory, speaking, musical rhythm, and some degree of smell recognition.

**Cerebellum:** The cerebellum is a small region of the brain at the back of the head that is about the size of a tennis ball. Its job is to keep posture, balance, and equilibrium by coordinating voluntary muscle movements.

**Brainstem:** The cerebrum and cerebellum are connected to the spinal cord by the brainstem, which acts as a transmission center. It controls all automatic functions such as: breathing, heart rate, body temperature, wake and sleep cycles, digestion, sneezing, coughing, vomiting, and swallowing.

The brain and spinal cord are surrounded by three protective covering layers called meninges, as can be seen in Fig 1.3.

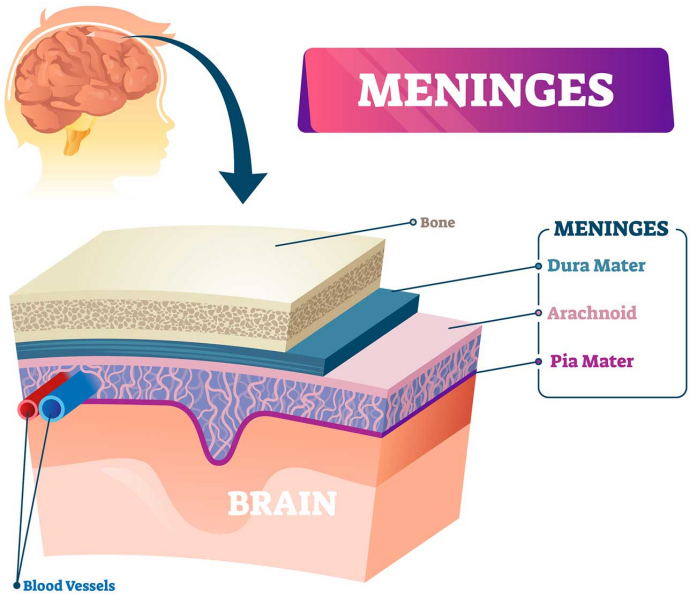


Figure 1.3: The meninges of the brain[5].

**The dura mater:** The dura mater, the outermost layer, is thick and robust. It contains two layers: the periosteal layer borders the inner dome of the skull (cranium), and the meningeal layer lies beneath it. Veins and arteries that feed blood to the brain can travel through the spaces between the layers.

**The arachnoid mater:** The arachnoid mater is a web-like layer of connective tissue that does not contain any nerves or blood arteries. The cerebrospinal fluid, or CSF, is found underneath the arachnoid mater. The entire central nervous system (brain and spinal cord) is cushioned by this fluid, which circulates these tissues to eliminate pollutants.

**The pia mater:** The pia mater is a thin membrane that surrounds and follows the contours of the brain's surface. Veins and arteries abound in the pia mater.

### 1.2.2 Brain Tumors

A brain tumor is a growth of cells or a mass in the brain that is abnormal [1]. Because the skull of the brain can not stretch out of shape, growth within such a limited space could result in nervous system anomalies and deficiencies. When tumors grow, the pressure inside the skull rises, causing brain damage and nervous system problems [6]. Examples of brain tumors are illustrated in Fig 1.4.

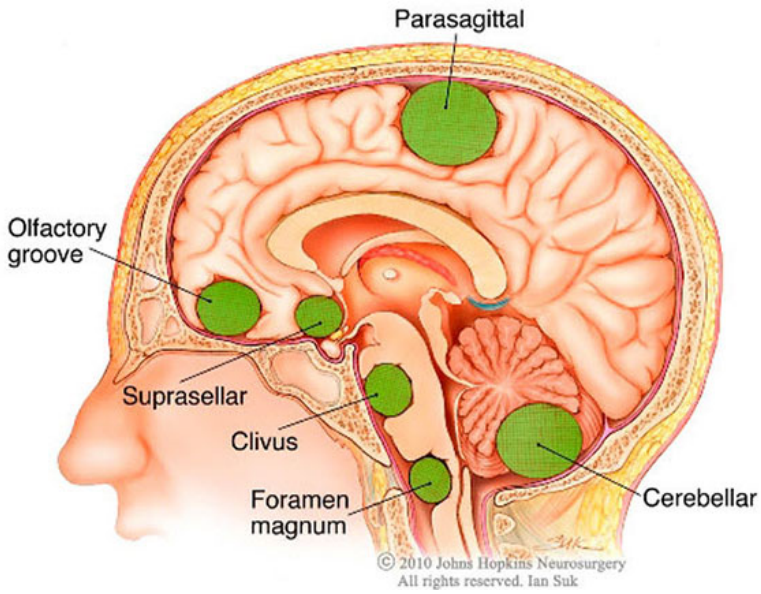


Figure 1.4: Examples of brain tumors [7].

### 1.2.3 Brain Tumor Statistics

Brain tumors have a high death rate across all age groups. In 2020, the American cancer society estimated the infection of 24,000 people and the death of 19,000. This year, they estimated the infection of 25,050 adults (14,170 men and 10,080 women) in the United States and the death of 18,280 adults (10,710 men and 7,570 women) from primary brain cancers and central nervous system (CNS) tumors. About 4170 children under the age of 15 will be diagnosed with a brain or CNS tumor this year in the United States. Worldwide, an estimated 251,329 people died from primary cancerous brain and CNS tumors in 2020 [8]. In Algeria, the "Global Cancer Observatory" estimated the infection of 1,777 people and the death of 1,478 in 2020[9]

### 1.2.4 Brain Tumor Symptoms

The troubles that a person may encounter are referred to as symptoms and signs. They indicate a disease or a condition and help in the medical diagnosis. Brain tumor symptoms can be general or specific. A general symptom is caused by the pressure of the tumor on the brain or spinal cord. Specific symptoms are caused when a specific part of the brain is not working well because of the tumor [10]. Symptoms of brain tumors depend on the location, size, and growth rate of the tumor. Severe headaches are common symptoms, which worsen with activity or in the early morning. Many people with brain tumors may also experience [1]:

- Fatigue.
- Nausea.
- Vomiting.
- Loss of sensation or movement in an arm or leg over time.
- Difficulties with balance, vision, speech and hearing.
- Increasing weakening or paralysis on one side of the body.
- Confusion in ordinary things.
- Inability to make decisions.
- Difficulty to execute simple orders.

### 1.2.5 Types of Brain Tumors

Many types of brain tumors exist, some are noncancerous (benign) and some are cancerous (malignant). According to their site of origination, brain tumors can be classified as primary (originate from inside the brain) and secondary (originate from inside the body and then travels to the brain).

Primary brain tumors grow in the brain or the surrounding tissues, they begin when the DNA of normal cells undergoes modifications (mutations). The mutations instruct the cells to reproduce and grow fast, even healthy cells would die. As a result, a tumor is developed from a mass of abnormal cells. Primary brain tumors come in a variety of types. The sort of cells involved gives each its name. Different types of brain tumors are mentioned below:

**Gliomas:** Astrocytomas, ependymomas, glioblastomas, oligoastrocytomas, and oligodendrogliomas are cancers that originate inside the brain or spinal cord.

**Meningiomas:** A meningioma is a type of tumor that grows inside the tissue surrounding both the brain and spinal cord (meninges). The majority of meningiomas are benign.

**Acoustic neuromas (schwannomas):** These are benign tumors that grow on the nerves that control balance and hearing in the inner ear and lead to the brain.

**Pituitary adenomas:** These tumors form in the pituitary gland, which is located near the base of the brain. These tumors can disrupt pituitary hormones, which can have far-reaching consequences throughout the body.

Secondary brain tumors grow after cancer spreads from another part of the body to the brain. Any cancer can spread to the brain. The following are examples of common types:

- Breast cancer.
- Colon cancer.
- Kidney cancer.
- Lung cancer.
- Melanoma.

How quickly a brain tumor grows can vary greatly. The growth rate as well as the shape and location of the brain tumor determine how it will affect the function of the nervous system. According to the World Health Organization (WHO), malignant brain tumors can be categorized into four grades[11], as demonstrated below:

**Grade 1:** The tumor grows slowly, and rarely spreads into nearby tissues. It may be possible to completely remove the tumor with surgery.

**Grade 2:** The tumor grows slowly but may spread to nearby tissues or recur.

**Grade 3:** The tumor grows quickly, it is likely to spread into nearby tissues, The tumor cells are abnormal. It has a high chance of transforming into a higher grade.

**Grade 4:** The tumor grows and spreads very quickly, very abnormal appearance, very short survival rate.

## 1.3 Medical Imaging Overview

Medical imaging is the technique and process of imaging the interior of the body for clinical examination and medical intervention, as well as visual representation of the function of particular organs or tissues. It is used by doctors and radiologists to inspect the human body to detect, diagnose, and monitor medical conditions and diseases. Many medical imaging techniques exist, where each type of technology provides specific pieces of information on the body part being investigated or treated, such as disease, injury, or the efficiency of medical treatment. The currently used are:

- Computed Tomography (CT).
- Positron Emission Tomography (PET).
- Single-photon Emission Computed Tomography (SPECT).
- Magnetic Resonance Imaging (MRI).
- Magnetic Resonance Spectroscopy (MRS).

### 1.3.1 Methodology of MRI

Magnetic Resonance Imaging (MRI) is the standard technique, the most popular, and efficient one [12]. According to the Statistics Portal, more than 36,000 MRI scanning machines were estimated to be in use in 2016 [2]. An example of an MRI machine can be seen in Fig 1.5 below.


Figure 1.5: MRI imaging machine [13].

Our work focuses on magnetic resonance imaging, which is a medical noninvasive 3D process that uses radio waves to generate images of the anatomy and physiological processes of the body. MRI scanners use high magnetic fields and computer-generated radio waves (radiofrequency energy) to create images, where the protons in fat and water molecules in the body

provide the majority of the signal in an MRI scan. The idea is to place the body inside a strong magnetic field which causes the water molecules to produce a faint signal, while a transmitter/receiver in the machine sends and receives radio waves, which are used to create cross-sectioned MRI images. An example of brain tumor MRI scans is shown in Fig 1.6.

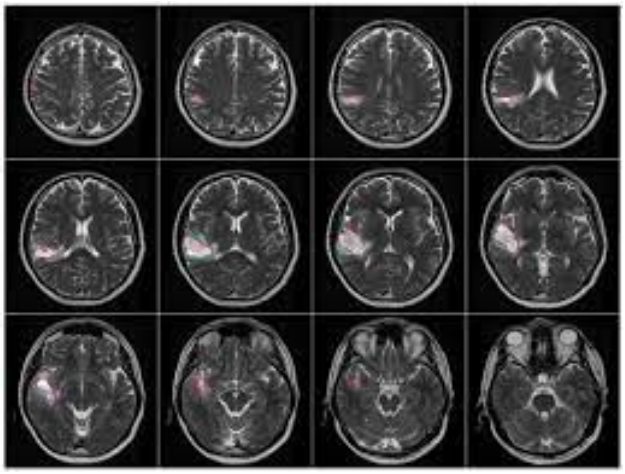

Figure 1.6: Brain tumor MRI scans [14].

## 1.4 Deep Learning Theory

Deep Learning (DL) is a subfield of machine Learning (ML). It is inspired by the structure and workings of the human brain using Artificial Neural Networks (ANN) that imitates biological neural networks. It gives machines the ability to learn without being explicitly programmed [15]. Deep learning uses multi-layer artificial neural network architectures to extract progressively higher-level features from data, so it delivers state-of-the-art accuracy in tasks such as object detection and speech recognition. Many types of learning exist and can be presented in the following three classes:

**Supervised Learning:**

Supervised deep learning model learns and makes predictions using labeled data. It can be thought of as a teacher supervising the learning process, the correct answers are given.

**Unsupervised Learning:**

An unsupervised deep learning model learns and makes predictions using unlabelled data. It is a learning process where the model does not need to be supervised, the dataset is unlabelled.

**Semi-Supervised Learning:**

Semi-Supervised learning is the process where the dataset contains a very small amount

of labeled data and a very huge amount of unlabelled data. Where extracting important features from data is challenging and labeling samples is time-consuming for professionals.

### 1.4.1 Artificial Neural Network

Artificial Neural Networks (ANN), usually called Neural Networks (NN), are computing systems inspired by the biological neural networks that constitute the human brain. An ANN is a collection of connected units or nodes called artificial neurons, which imitate the biological neurons. In the brain, a neuron collects the input signal from its dendrite. The neuron sends out spikes of electrical activity through its axon, which can be split into thousands of branches. At the end of each branch, a synapse converts the activity from the axon into electrical effects that inhibit or excite activity on the contracted neuron. An example of a biological neural network is shown in Fig 1.7:

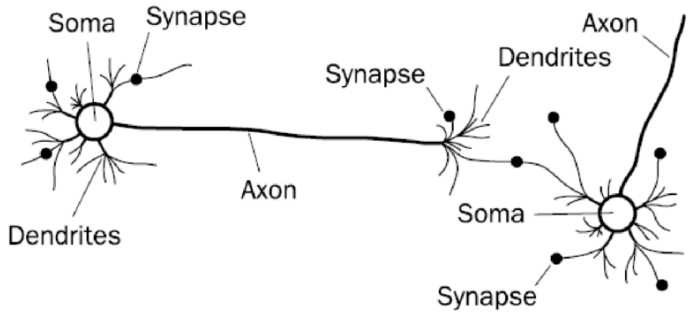


Figure 1.7: Example of a biological neural network [14].

ANNs are typically composed of interconnected units which serve as model neurons. They multiply each input by a weighting factor, then biasing the sum of the multiplier factor. They contain three types of layers: an input layer, an output layer, and hidden layers as demonstrated in Fig 1.8:

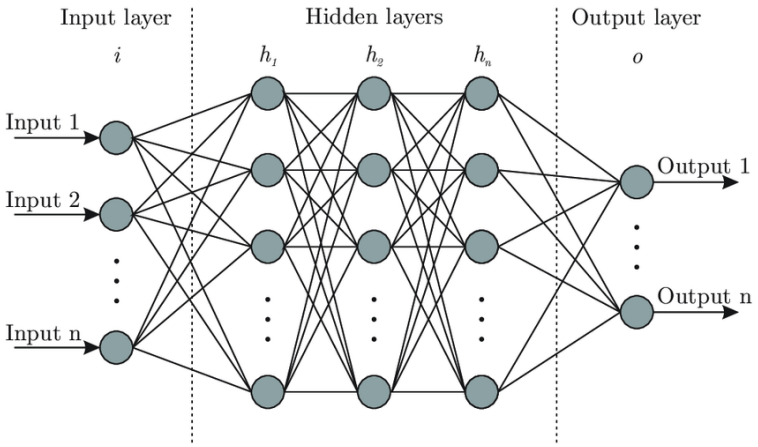


Figure 1.8: Artificial neural networks architecture [16].

Deep Neural Networks (DNN) are Artificial Neural Networks (ANN) with multiple layers between the input and output layers. There are different types of neural networks but they always consist of the same components: neurons, synapses, weights, biases, and functions

### 1.4.2 Convolutional Neural Networks

A Convolutional Neural Network (CNN or ConvNet) is one of the most popular types of deep neural networks [17]. It is a deep learning processing for data with a grid pattern, such as an image. The CNN model does not require a hand-crafted feature extraction or image segmentation. It obtains abstract features when input propagates toward deep layers, which are arranged in such a way to detect simpler patterns first and more complex patterns further. A typical convolutional neural network is demonstrated in Fig 1.9:

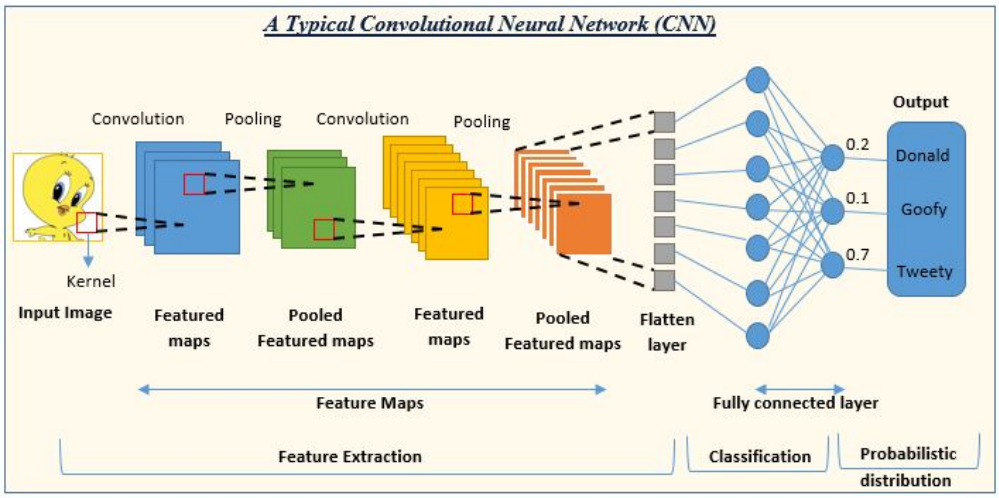


Figure 1.9: Convolutional neural network layers [18].

### 1.4.3 Types of CNN Architecture Layers

**Convolutional Layer:** The convolutional layer is a fundamental component of the CNN architecture that performs feature extraction. It carries the main portion of the network's computational load. The kernel slides across the height and width of the image performing a dot multiplication between the kernel and the portion of the receptive field. The layer outputs another image with different dimensions. An example of a convolutional layer and the convolution operation is shown in Fig 1.10:

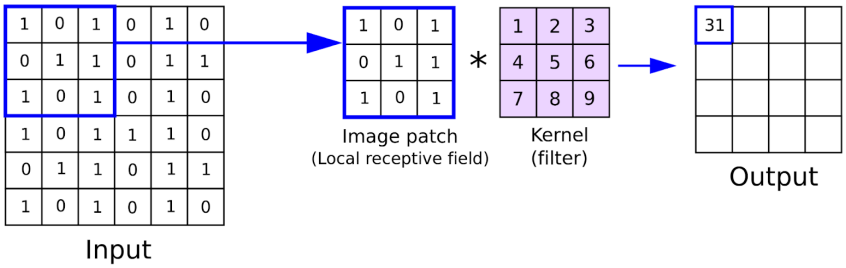


Figure 1.10: Convolutional layer operation [19].

**Kernel size:** The kernel, also called the convolution matrix, is a matrix that is slid over the image and multiplied with the input to enhance the output in the desired way.

**Stride:** The kernel's step size when sliding through the image is defined by the stride.

**Padding:** It is the term used in convolutional neural networks to describe how many pixels are added to an image when it is processed by the CNN kernel.

**Non-Linear Layer:** Convolution is a linear operation, while images are far from linear. Hence, the outputs of the convolution operation pass through a nonlinear activation function. There are several types of nonlinear functions, the popular ones being: sigmoid, hyperbolic tangent, and Rectified Linear Unit (RELU).

**Pooling Layer:** The pooling layer provides a typical down-sampling operation, which helps in reducing the spatial size of the feature map, it decreases the required amount of computation and weights. There are two types of pooling: average pooling and max-pooling, which can be shown in Fig 1.11 below:

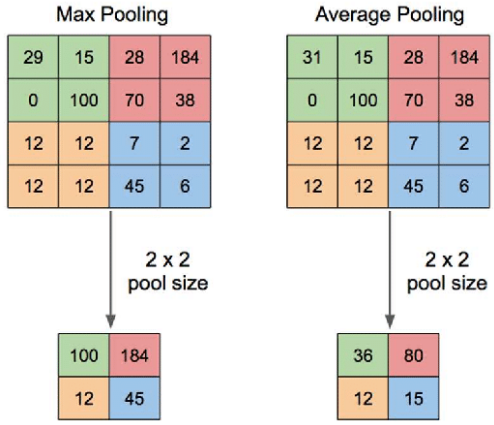


Figure 1.11: Max-pooling and Average-pooling layers [20].

**Flatten Layer:** Flattening layer converts the pooling output of 2-dimensional arrays into

a single long continuous linear vector before it is fed to the fully connected layer to classify the image.

**Fully Connected Layer:** The fully connected layer, also known as the dense layer, has several outputs equal to the number of classes. Once the features are extracted by the convolutional layer and down-sampled by the pooling layer are created, they are mapped by a subset of fully connected layers to the final outputs of the network, as the probabilities for each class in classification.

### 1.4.4 Training a Convolutional Neural Network

#### 1.4.4.1 Forward Propagation:

Forward Propagation, also known as forwarding pass, means in simple words moving in the forward direction from input to output layers and passing through the hidden layers. Given its input from the previous layer, each unit computes the weighted sum as follows:

$$y = f(\sum_{n=1}^{N} w_n \,.\, x_n + b) \tag{1.1}$$

Then the activation function is applied to the result:

**Activation Function:** The resultant weighted sum is used by the activation function as input and calculated activation outputs are fed as input to the next layer. The activation output varies between 0 and 1 or -1 and 1, it can be interpreted as yes or no. Since linear activation functions can not learn complex mappings, Non-linear are the most used, namely:

**Rectified Linear Unit:** Rectified Linear Unit (RELU) provides faster computation since it does not compute exponent operations and divisions. It will give the input directly if it is positive, otherwise, it will result in zero. It computes the function depicted in Fig 1.12:

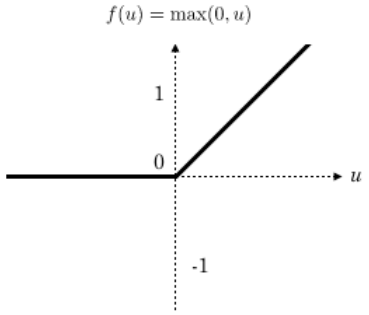


Figure 1.12: Rectified Linear Unit [21].

**Softmax:** The softmax activation function is used in a neural network to normalize the output of a network to a probability distribution over predicted output classes. It assigns to each class in a multi-class problem a value in the interval [0,1]. Those probabilities add up to 1. The below mathematical equation models the softmax activation function:

$$\delta(x_i) = \frac{e^{xj}}{\sum_i e^{xi}} \tag{1.2}$$

**Hyperbolic Tangent:** It stands for tangent hyperbolic function. The advantage of this mathematical approach is that negative input will be mapped strongly negative and the zero inputs will be mapped near zero in the hyperbolic tangent graph.

The forward pass workflow is depicted in Fig 1.13:

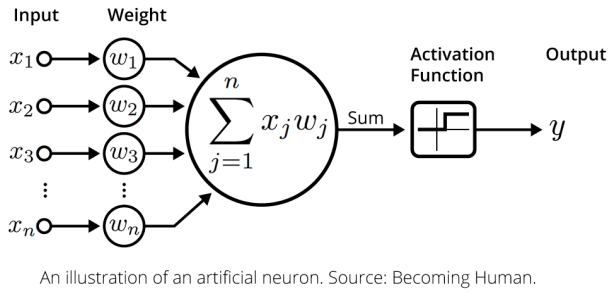


Figure 1.13: Illustration of weighted sum and activation function operations on a neuron [22].

The model performance on training data is calculated by the cost function.

**Cost Function:** It measures the compatibility between the output predictions and the given labels. The algorithm performance increases as the cost function decrease. It is the average of the loss function over all the training samples. Most used loss function for measuring performances of a classification model whose output is a probability between 0 and 1 is cross-entropy:

$$\pounds(y, \hat{y}) = -\sum_{j}^{M}\sum_{i}^{N}(y_{ji} \; . \; log\, \hat{y}_{ij}) \tag{1.3}$$

where M represents number of classes and N number of samples

#### 1.4.4.2 Optimizing a Neural Network

The optimization seeks the convergence of the loss function by continuously updating the parameters in each iteration. Different optimization algorithms are presented below:

**Gradient Descent:** Gradient descent is the most used optimization algorithm for linear regression and classification problems. It calculates the first-order derivative of a loss function to find the weights for which the loss function reaches a local minimum. There are mainly three types of gradient descent, namely:

**Batch Gradient Descent:** Batch gradient descent takes into consideration all the training data in every step. It calculates the average of the gradients of all the training examples and uses it to update the network parameters. It calculates:

$$\underbrace{\theta}_{\textit{the new parameter}} = \underbrace{\theta}_{\textit{the old parameter}} - \underbrace{\eta}_{\textit{the learning rate}} . \underbrace{\bigtriangledown \theta \imath(\theta)}_{\textit{the gradient}} \tag{1.4}$$

**Stochastic Gradient Descent:** Stochastic Gradient Descent (SGD) updates the model's parameters on each training example. SGD updates the model parameters more frequently and requires less time and less memory for convergence.

**Mini Batch Gradient Descent:** Mini Batch Gradient Descent merges the robustness of stochastic gradient descent and the efficiency of batch gradient descent. It splits the training dataset into mini-batches, which are used to calculate model cost functions and update model parameters.

**Adam Optimizer:** Adaptive Moment Estimation (ADAM) combines two gradient descent methodologies, namely:

**Momentum:** It calculates the exponentially weighted average which accelerates the gradient descent algorithm.

**Root Mean Squared Propagation (RMSP):** It takes the exponential moving average.

The ADAM optimizer is given by:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{1.5}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{1.6}$$

$$\hat{v}_t = max(\hat{v_{t-1}}, v_t) \tag{1.7}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \varepsilon} \tag{1.8}$$

**Batch Normalization:** Batch normalization re-centers and re-scales the layer's input. It is a type of supplement layer that normalizes the input values of the next layer which mitigates the risk of over-fitting and improves gradient flow through the network. It allows the use of higher learning rates without the risk of divergence.

$$z_{norm}^{(i)} = \frac{z^{(i)} - \mu}{\sqrt{\delta^2 + \varepsilon}} \tag{1.9}$$

$$\tilde{z}^{(i)} = \gamma . z_{norm}^{(i)} + \beta \tag{1.10}$$

Where, the mean and variance are given as follows:

$$\mu = \frac{1}{m} \sum_i z^i \tag{1.11}$$

$$\delta^2 = \frac{1}{m} \sum_i (z^i - \mu)^2 \tag{1.12}$$

#### 1.4.4.3 Backward Propagation

Backward propagation is the process of moving in the backward direction, from the output to the input layer. It is a computational approach for adjusting and correcting the weights to reach the minimized loss function using gradient descent.

The back propagation algorithm differentiates the loss function for each weight in the network, then the obtained gradient is fed to the gradient descent method.

An illustration of the back-propagation is demonstrated in Fig 1.14:

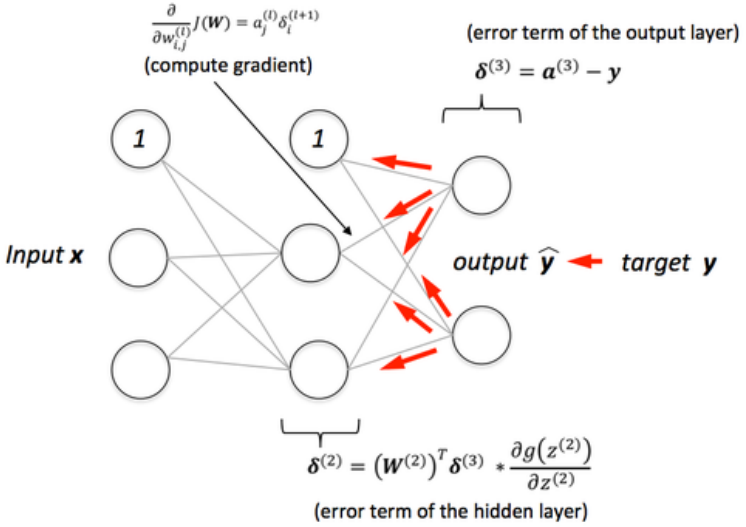


Figure 1.14: Illustration of back-propagation [23].

### 1.4.5 Bias-variance Trade-off

Deep learning models may experience bias and variance related prediction errors.

**Bias:** The difference between the average predictions of the model and the correct values it is trying to predict. High bias oversimplifies the model.

**Variance:** The variability of a model to a certain data point or a value. A high variance may result in over-fitting problems.

So, balancing the bias and variance is necessary to avoid issues such as:

#### 1.4.5.1 Over-fitting: high variance low bias

Over-fitting is a concept in deep learning which occurs when the model fits perfectly to the data used during training but can not be generalized to unseen data. This happens when the problem learns the noise in the training data, that does not apply to new data, which negatively impacts the ability of the model to generalize. Over-fitting can be reduced by regularization techniques, such as:

**L1/L2 Regularization:** L1 and L2 regularization techniques constrain a complex learning model, which reduces over-fitting. It applies a penalty to the cost function to push the weights toward zero. The main difference between L1 and L2 regularization techniques is that, unlike the L1 technique, the L2 regularization reduces the weights to zero.

$$L1_{Regularization} = \lambda \sum_{j=1}^{p} \|\beta_j\| \tag{1.13}$$

$$L2_{Regularization} = \lambda \sum_{j=1}^{p} (\beta_j)^2 \tag{1.14}$$

**Dropout:** Dropout is a regularization method that refers to randomly dropping out units in a neural network during a training phase. Dropping out refers to randomly not considering some neurons during a particular forward or backward pass.

**Early Stopping:** Early stop is a regularization method that stops the training phase and saves the model once the loss starts increasing. It can be implemented either by monitoring the loss graph or by setting an early stopping trigger.

**Hold Out:** Hold out technique is used by splitting the data-set into training and testing subsets. The most common split ratio is 80 percent for training and 20 percent for testing.

#### 1.4.5.2 Under-fitting: low variance high bias

Under-fitting is a problem in deep learning that occurs when the model is too simple that it is unable to underlay the relationship between the input and output data accurately. An under-fitted model performs poorly on both training and testing data. It can be reduced by some techniques, such as:

**Increasing duration of training:** Stopping the model too soon may result in an under-fitted model. However, as mentioned before, too late stopping may result in over-fitting. Hence, it is important to find the appropriate training time.

**Data augmentation:** It is a technique used to increase the amount of data by modifying the already existing dataset. Data augmentation is done by applying various image transformation techniques to the available dataset, such as flipping, rotating, re-scaling, and shifting. Fig 1.15 demonstrates the three cases of overfitting, underfitting, and Optimal-Fitting:

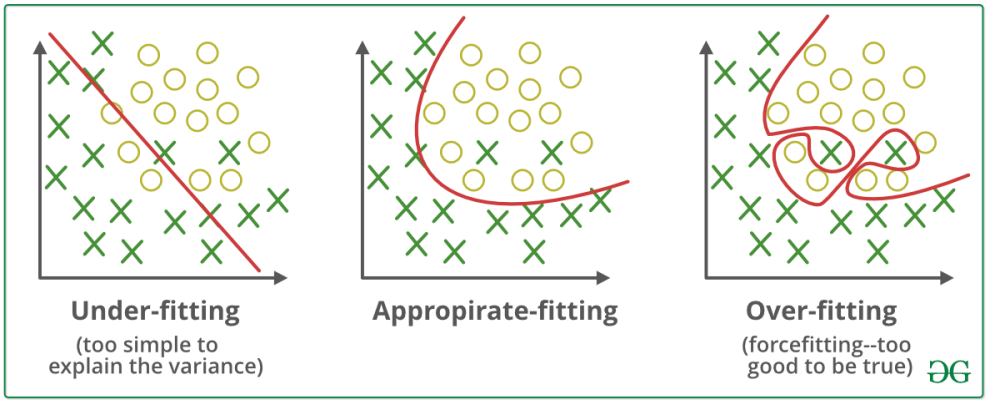


Figure 1.15: Overfitting underfitting and optimal-Fitting [24].

On both training and validation sets, learning curves indicate the relationship between the training set and the selected evaluation metric (e.g., loss, accuracy, etc.). They can be a very valuable tool for evaluating model performance because they can inform if the model has a bias or variance problem. the relationship between bias, variance, and total error can be seen in the graphs in Fig 1.16:

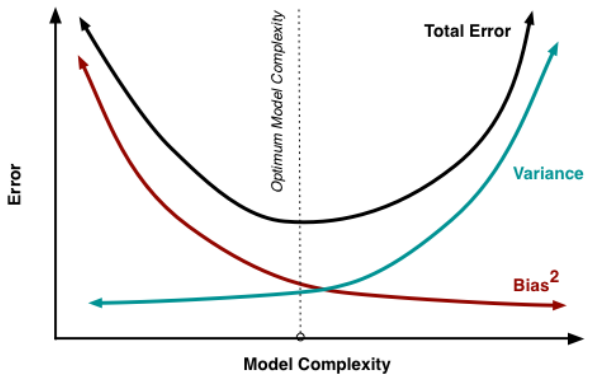


Figure 1.16: Bias-variance trade-off [25].

### 1.4.6 Steps of Deep Learning Classification Approach

The deep learning models follow conventionally standard predefined steps to classify input images. The description of steps and approaches is as follows:

#### 1.4.6.1 Image Preprocessing

Image preprocessing consists of removing unwanted data from the input images, which improves the quality of the input so that the results are more accurate. The input dataset images are split into training, validation, and testing sets. According to the model classes, each set is partitioned into labeled subsets. During this step, the input size and batch size are set.

#### 1.4.6.2 Data Augmentation

As mentioned before, data augmentation is the process of increasing the available data for training the model. In deep learning, data augmentation refers to approaches for increasing the amount of data by adding slightly changed copies of current data or creating new synthetic data from already existing data. One technique to augment the data is flipping the input images by random degrees.

#### 1.4.6.3 Segmentation

Image segmentation refers to splitting the digital image into multiple parts or subgroups called segments. It is an important step for further processing as feature extraction. The

main objective of image segmentation is to group sections of a picture that belong to the same object class together.

#### 1.4.6.4 Feature Extraction

Feature extraction is the process of capturing the important parts of the image. Convolutional neural networks do the feature extraction on their own. The most significant benefit of deep learning is that it eliminates the need to extract features from images manually. During the training phase, the network learns to extract features [26].

#### 1.4.6.5 Classification

The model learns to identify the category of new observations based on training data. It outputs the probability of an image belonging to each class and assigns it to the class with the highest probability.

### 1.4.7 Transfer Learning

Transfer learning is a machine learning research subject that focuses on storing and transferring knowledge learned while tackling one problem to a different but related one. It is a machine learning technique in which a model created for one task is utilized as the basis for a model on a different task. Transfer learning is the process of applying features acquired on one problem to a new related problem. Features from a model trained to recognize an object may be used to kick-start a model that learns to identify another object. Transfer learning is useful when the dataset contains insufficient data to train a model from scratch.

A pre-trained model is a saved network previously trained on a large dataset like IMAGENET typically on a large-scale image classification task.

The following workflow is the most common manifestation of transfer learning in deep learning: freeze layers from a previously trained model to prevent any of the knowledge they contain from being lost during subsequent training rounds. On top of the frozen layers, add some new trainable layers. They will learn how to convert previous features into predictions on a new dataset. Train the new layers on the new dataset.

### 1.4.8 Fine Tuning

Fine-tuning is a technique for putting transfer learning into practice. It is the process of taking a model that has already been trained for one task and then refining or adjusting it to

perform a second task comparable to the first one. Researchers can take advantage of what the model has previously learned by utilizing an artificial neural network that has already been created and trained rather than developing it from scratch.

Fine-tuning includes unfreezing the entire model (or a portion of it) and retraining it with a very low learning rate on the new data. Incrementally adapting the pre-trained features to the new data, has the potential to achieve significant improvement.

## 1.5 Related Work

Computer vision and machine learning have revolutionized the world in every way possible over the last couple of years [27]. It has demonstrated outstanding achievements in every sector, especially in the domain of anomaly detection. Its capability has also been applied and studied in detecting brain tumors utilizing MRI scans for effective prediction with outstanding results [28].

Kalaiselvi et *al.* [29] compared several CNN models for classifying brain MRI scans and chose the best CNN model for brain tumor detection. They built six CNN models in this study, that were trained on the BraTS2013 dataset and tested on the WBA dataset. Among the six models, the best model for tumor slice classification was discovered. They have attained about 96–99% of accuracy.

Çinar et *al.* [30] classified brain tumors based on MRI images in their research paper. The brain tumor MRI images utilized in this study were from the Kaggle platform "Brain MRI Images for Brain Tumor Detection" dataset. The classification process utilises CNN models, the base architecture is the Resnet50, whose last five layers have been removed and eight new layers have been added. They have attained about 97.2% of accuracy.

Sadad et *al.* [31] implemented segmentation using Unet architecture with ResNet50 as a backbone. The brain tumor MRI images utilized in this study were from the Figshare dataset. To improve the classification rate, the preprocessing and data augmentation techniques were applied. Different CNN models such as MobileNet V2, Inception V3, ResNet50, DenseNet201, and NASNet, were used for tumor classification and achieved accuracy of 91.8%, 92.8%, 92.9%, 93.1%, and 99.6% respectively.

Rehman Khan et *al.* [32] provided a deep learning approach to classify brain tumors using MRI data analysis through a finetuned VGG19 model. They categorized tumors into their appropriate classes. The data augmentation concept was used to expand available dataset for classifier training, which improves classification accuracy over 92%.

Makde et *al.* [33] presented a convolutional neural network framework in this paper. The framework is implemented using AlexNet and ZFNet architectures, and the system has been trained to detect tumors in lung nodules and the brain. For both architectures and datasets of lung CT and brain MRI images, classification accuracy is greater than 97%.

## 1.6 Summary

Over the last few years, computer vision and machine learning have revolutionized the world in every way possible, especially in the area of medical anomaly detection, including brain tumor classification. In this chapter, we went through the theoretical background needed for this project and its related works. We briefly discussed brain tumors, medical imaging, and deep learning theory. Then, we mentioned works related to our domain of interest.

# Chapter 2

# Design and Implementation of Deep Neural Networks for MRI-Based Brain Tumor Classification

## 2.1 Introduction

This chapter provides the details of our methodology in conducting the project. First, it discusses the dataset used for training, validation, and testing the models. After that, it presents an overview of used deep learning models, their history, and architectures. Finally, the chapter discusses the tools used for implementing the mentioned models.

## 2.2 Dataset

Deep learning algorithms require huge amounts of data for both training and validation. Fortunately, many brain tumor MRI-based datasets are available for research purposes. Our deep learning model must learn from input images how to successfully classify the unsees brain MRI scans to cancerous and non-cancerous. The dataset was acquired online from the KAGGLE platform. It belongs to Ahmed Hamada, a data scientist from Cairo, Egypt. It contains two classes, yes and no, each containing 1500 images. Since we are working on a supervised learning approach, each input image has a yes or no label.

## 2.3 Proposed Methodology

This section describes the methodology used in dealing with brain tumor classification using deep learning. First, we preprocessed the input images to ensure smooth learning and more accurate results. Then, we augmented our dataset to avoid any bias-variance related problems. Finally, we employed CNN, which is one of the most reliable deep learning algorithms.

### 2.3.1 Data Preprocessing

Data preprocessing is necessary before delivering the input images to the CNN models. As a first step, cleaning the input dataset is done by deleting improper pieces of information, which is images in inappropriate formats. Then, the dataset is split into 80% for training and 20% for testing data, where the training subset is split again to 80% for training and 20% for validation. Next, each subset is divided into "yes" and "no" classes. The distribution of samples after cleaning and splitting the dataset is demonstrated in Fig 2.1:

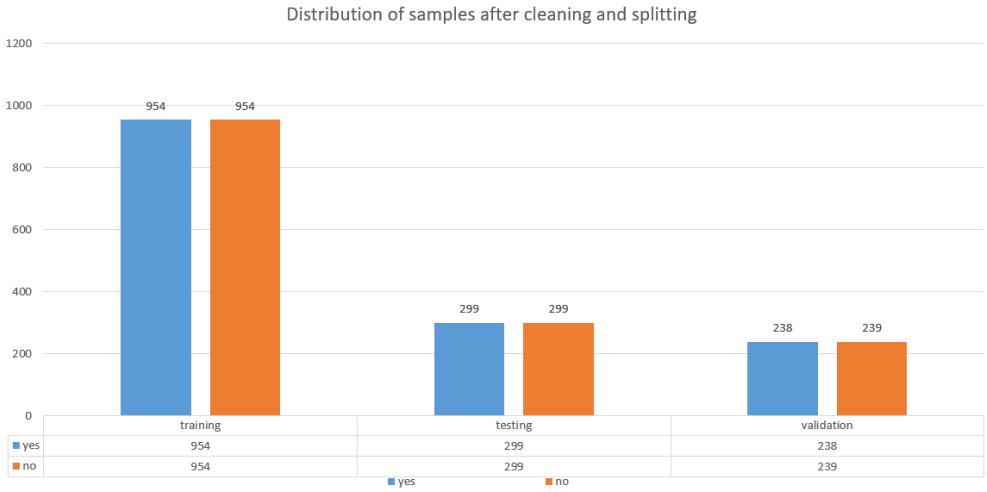


Figure 2.1: Distribution of samples after cleaning and splitting.

Finally, a label (0 or 1) is assigned to each image according to its class and the image size is fixed depending on the implemented model. Fig 2.2 illustrates an example of some images from the used dataset, each with its label:

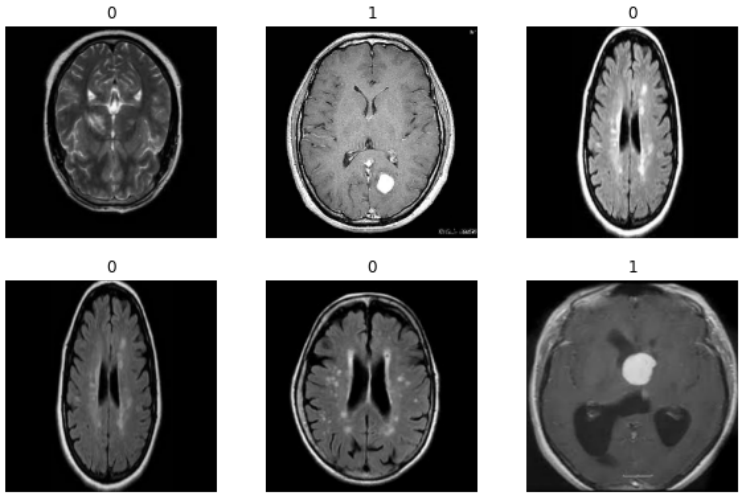


Figure 2.2: Example of labeled dataset images.

### 2.3.2 Data Augmentation

We artificially generated sample variety by applying random transformations to the training images, such as random horizontal flipping or slight random rotations. This allows the model to be exposed to a variety of training data while also slowing overfitting. An example of an augmented image from the dataset is illustrated in Fig 2.3:

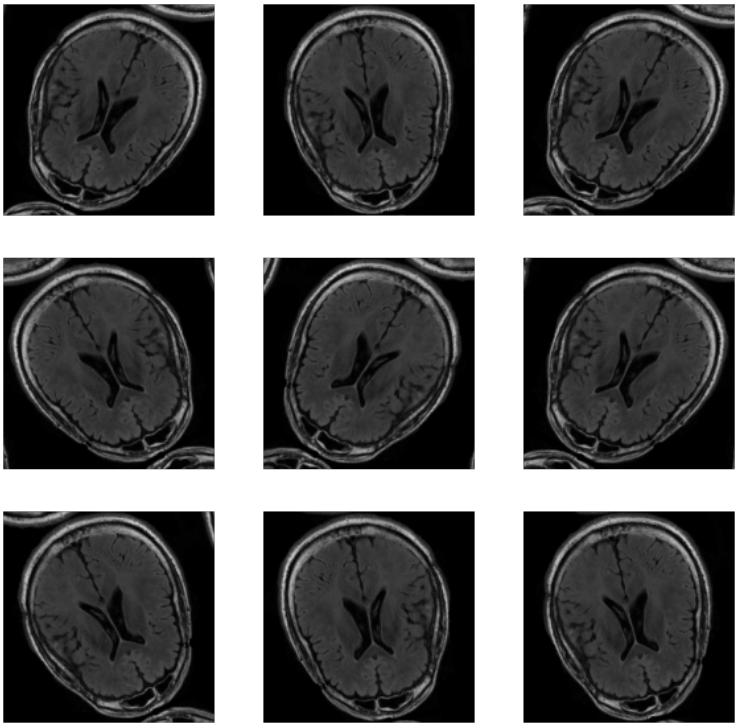

Figure 2.3: Example of augmented dataset image.

### 2.3.3 Hyperparameters

**Learning rate**:The learning rate of a network is the rate at which its parameters are updated. The learning process is slowed by a low learning rate, but it converges smoothly. A faster learning rate accelerates the learning process, although it may not converge.

**Batch size**: It refers to the number of training samples given to the neural network in one iteration before updating the parameters.

**Number of epochs**: The number of epochs is the total number of times the training dataset has been processed.

### 2.3.4 Neural Network Models Used for Image Classification

The architectures of models implemented in this report are discussed below:

#### 2.3.4.1 AlexNet

AlexNet is a convolutional neural network architecture designed by Alex Krizhevsky in collaboration with Ilya Sutskever and Geoffrey Hinton [34]. It was the ILSVRC 2012 challenge winner with a top 5 error. It is an 8-layer model, it consists of five convolutional layers with a combination of max pooling followed by 3 fully connected layers, they use RELU activation in each of these layers except the output layer(Softmax) [34].

The size of the input images is 227x227x3. The first convolution layer with 96 filters of size 11x11 with a stride of 4 is applied with a RELU activation function, This outputs a feature map with a size of 55x55x96. The output is passed through the first max-pooling layer, of size 3x3 and stride 2. Then, the resulting feature map of size 27x27x96 is passed through the second convolutional layer with 256 filters of size 5x5 each, stride 1, padding 2, and a RELU activation function. The output of size 27x27x256 is passed through the second max-pooling layer of size 3x3 and stride 2, the resulting feature map is of dimensions 13x13x256. The third convolution layer with 384 filters of size 3x3, stride 1, and padding 1 are applied with a RELU activation function. The output feature map is of shape 13x13x384. The fourth layer is identical to the third so the output remains of the same dimensions. The final convolution layer of 256 filters with size 3X3, stride 1, and padding 1, The activation function is RELU, resulting in a feature map of shape 13X13X256. The third max-pooling layer of size 3X3 and stride 2 results in a feature map of the shape 6X6X256.

After this, the first dropout layer of rate 0.5 is applied. Then the first fully connected layer with a RELU activation function generates the output of size 4096. Next comes another dropout layer with a dropout rate fixed at 0.5, followed by a second fully connected layer with 4096 neurons and RELU activation. Finally, the last fully connected layer or output layer with 1000 neurons. The activation function used at this layer is softmax. Fig 2.4 demonstrates the Alexnet architecture:

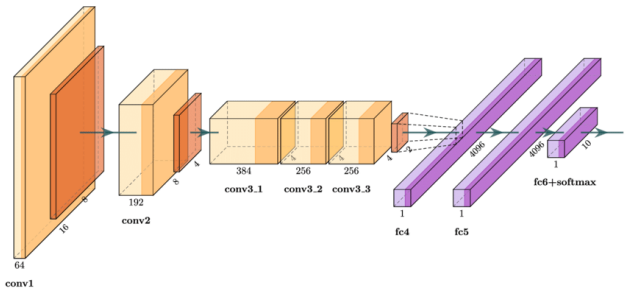


Figure 2.4: AlexNet architecture [35].

#### 2.3.4.2 VGG16

VGG16 is an acronym for Visual Geometric Group. It was designed by K.Symonian and A.Zisserman from Oxford University in the paper "Very Deep Convolutional Neural Network

for large scale image recognition" [36]. VGG16 is a 16 layers model that consists of 13 convolutional layers, 5 max-pooling layers, and 3 fully connected layers. It was the first runner-up of the ILSVRC 2014 challenge.

The size of input images is fixed to be 224x224x3. The first block consists of 2 convolutional layers, each with 64 filters of size 3x3, stride 1, and padding 1. Followed by a RELU activation function, resulting in an output of the same shape as the input images. The output is passed through a max-pooling layer of size 2x2 and stride 2. This results in an output of dimensions 112x112x64. The output is then passed through a second block similar to the first one, but with a kernel of size 3x3 each. The third block consists of 4 convolutional layers with 256 filters of size 3x3, stride 1, padding 1, and a RELU activation function for each. Then the output is passed through a max-pooling layer of size 2x2 and stride 2. The resulting output is of dimensions 28x28x256. The fourth and fifth blocks are identical, each block consists of 4 convolutional layers with 512 filters of size 3x3, stride 1, padding, 1 and a RELU activation function for each, and a max-pooling layer of size 2x2, and stride 2. The output of the fourth block is of shape 14x14x512. The output of the fifth layer is of shape 7x7x512.

The output of the convolutional layer passes through three fully connected layers with a flattening layer in-between. The first two fully connected layers have 4,096 neurons each, and the last fully connected layer is the output layer. It uses a sigmoid and has 1,000 neurons. Fig 2.5 illustrates the VGG16 architecture:

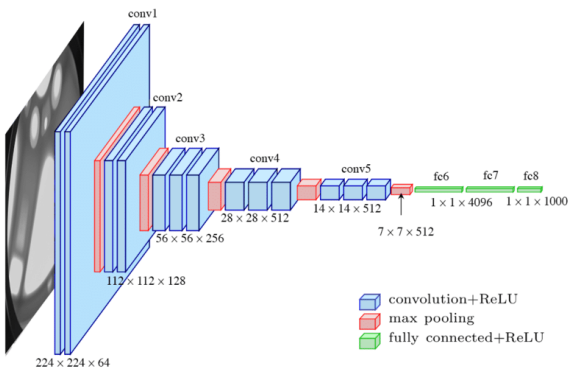


Figure 2.5: VGG16 architecture [37].

#### 2.3.4.3 VGG19

VGG19 is a variant of the VGG19 convolutional neural network model. It consists of 19 layers (16 convolution layers, 3 fully connected layers, 5 max-pooling layers, and 1 softMax layer).

The size of input images is fixed to be 224x224x3. The first block consists of 2 convolutional layers each with 64 filters of size 3x3, stride 1, and padding 1. Followed by a RELU activation function, it results in an output of the same shape as the input images. The output is passed

through a max-pooling layer of size 2x2 and stride 2. This results in an output of dimensions 112x112x64. The output is then passed through a second block similar to the first one, but with a size of 3x3 each. The third block consists of 3 convolutional layers with 256 filters of size 3x3, stride 1, padding 1, and a RELU activation function for each. Then the output is passed through a max-pooling layer of size 2x2, and stride 2. The resulting output is of dimensions 28x28x256. The fourth and fifth blocks are identical, each block consists of 3 convolutional layers with 512 filters of size 3x3, stride 1, padding 1, and a RELU activation function for each, and a max-pooling layer of size 2x2 and stride 2. The output of the fourth block is of shape 14x14x512. The output of the fifth layer is of shape 7x7x512.

The output of convolutional layers passes through three fully connected layers with a flattening layer in-between. The first two have fully connected layers 4,096 neurons each, and the last fully connected layer is the output layer. It uses a sigmoid activation function and has 1,000 neurons. Fig 2.6 demonstrates the VGG19 architecture:

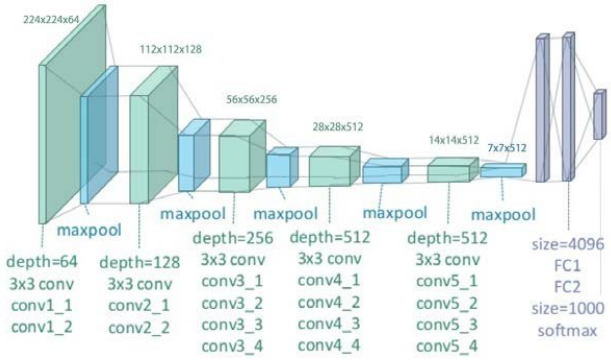


Figure 2.6: VGG19 architecture [38].

#### 2.3.4.4 ResNet50

ResNet is an acronym for Residual neural Network. ResNet was proposed in 2015 by Microsoft researchers Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun in a research paper entitled "Deep Residual Learning for Image Recognition" [39].

Involving more extra layers aids in the resolution of complex problems and increases the accuracy of the result. However, deeper networks may face the issue of degradation. In other words, as the number of layers increases, the accuracy levels may eventually get saturated or decline. This is due to the vanishing gradient problem:

**Vanishing gradient problem:** The gradients of the loss function approach zero when more layers with specific activation functions are added to the neural network, making the network difficult to train. This occurs because of the activation functions that convert a huge input space into a smaller one, limited between 0 and 1. Hence, a significant change in the input of the activation function results in a minor change in the outputs.

Residual neural networks proposed using residual blocks as a solution to the gradient vanishing:

Residual blocks are also called skip connections or shortcuts. They learn residual functions based on the layer's input. The skip connection adds the input value of the block to its output directly without passing through activation functions, as a result of which the block's overall derivative is increased.

The ResNet50 architecture is 50 layers deep. It is divided into 5 stages, each stage contains a convolutional and an identity block:

**Identity Block** it has 3 convolutional layers. The input is passed directly through the shortcut, so it is used when the X and X-shortcut match up.

**Convolutional Block** It has 3 convolutional layers. The input is passed through a convolution layer in the shortcut, so it is used when the X and X-shortcut do not match up. Fig 2.7 demonstrates the ResNet50 architecture:

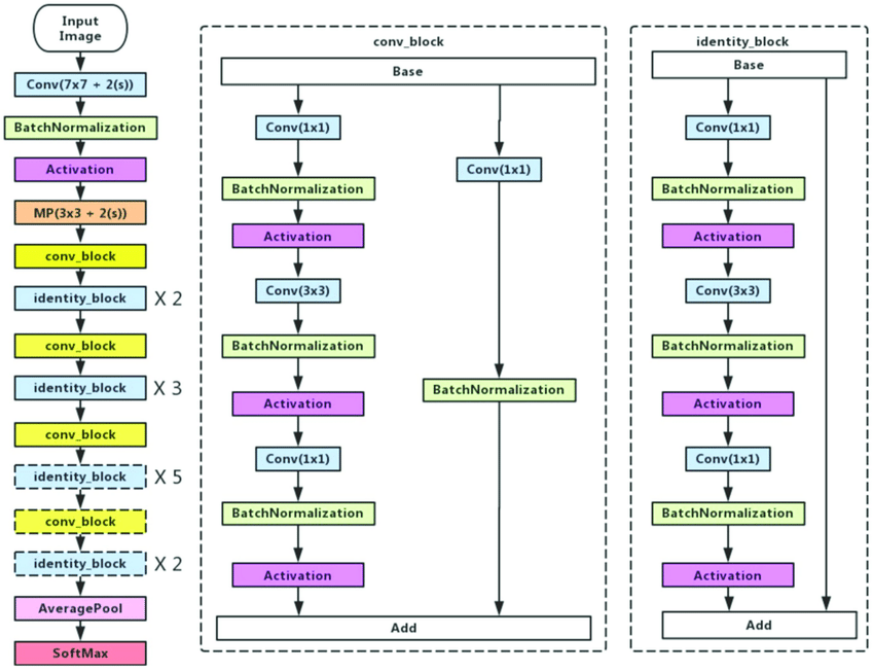


Figure 2.7: ResNet 50 architecture [40].

#### 2.3.4.5 XCEPTION

Xception is a convolutional neural network model. It was introduced by Francois Chollet in the research paper entitled "Xception: Deep Learning with Depthwise Separable Convolutions" [41]. It was the first runner-up in ILSVRC 2015 challenge. XCEPTION is 71 layers deep, it uses depthwise separable convolution.

The convolution operation is computationally expensive. Depth-wise separable convolutions have been introduced to reduce the cost of these computations. The computation is broken into two phases when using depth-wise separable convolutions, namely:

**Depth-wise convolution:** It is so-called because it deals with depth dimensions as well as spatial dimensions. It does not perform the convolution computation over all channels, but rather one at a time.

**Point-wise convolution:** It performs a typical convolution to change the dimensions.

The Xception architecture is based on the use of depth-wise separable convolution and residual blocks. It consists of 36 convolutional layers divided into 14 modules. Except for the first and last modules, all of the modules have linear residual connections surrounding them. Fig 2.8 shows the ResNet50 architecture:

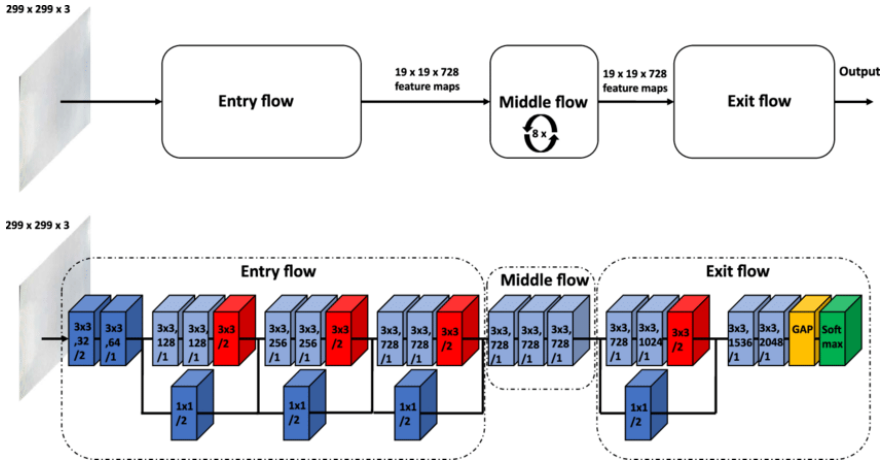


Figure 2.8: XCEPTION architecture [42].

### 2.3.5 Evaluation Experiments

In this project, we implemented each of the previously mentioned architectures using the three approaches explained below:

#### 2.3.5.1 Implementation from scratch

In this first approach, we re-built the architectures: Alexnet, VGG16, VGG19, Resnet50, and Xception from scratch using Keras.layers API's from Tensorflow and Keras libraries. We kept the same types and numbers of layers, strides, paddings, kernel sizes, and input sizes as in the original research papers. The objective is to test the ability of models that have been trained on small datasets to generalize and give accurate results on unseen data.

#### 2.3.5.2 Transfer Learning

In this second approach, we re-used model weights from previously trained models developed for large-scale computer vision problems, such as classifying 1,000,000 images for 1000 categories. We took layers of the pre-trained models and unfroze them using Keras.model API's from Tensorflow and Keras libraries. Then, we added new trainable layers, such as flattening and dense layers, on top of the frozen layers. Finally, we trained the model on our dataset.

#### 2.3.5.3 Fine Tuning

In this third approach, we re-used the same previously trained models as in approach two, except that this time we froze the model weights during the first 20 epochs. Then, we unfroze them during the following 80 epochs.

### 2.3.6 Tools

#### 2.3.6.1 Python

Python is an open-source interpreted high-level general-purpose programming language. It supports multiple programming paradigms, including structured, object-oriented, and functional programming. It has a philosophy that emphasizes code readability with the use of significant indentation. Python provides several modules and built-in packages. It has autonomous memory management and a dynamic typing system. It was first released in 1991 by Guido van Rossum at Centrum Wiskunde Informatica (CWI) [43].

#### 2.3.6.2 Tensorflow

TensorFlow is an open-source machine learning platform that runs from start to end. It has a large and flexible ecosystem of tools, libraries, and community resources that allow academics to advance the state-of-the-art in machine learning and developers to easily construct and deploy ML applications.

#### 2.3.6.3 Keras

Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library. It is the high-level API of TensorFlow 2: an approachable, highly-productive interface for solving machine learning

problems, with a focus on modern deep learning. It provides essential abstractions and building blocks for developing and shipping machine learning solutions with high iteration velocity. Keras empowers engineers and researchers to take full advantage of the scalability and cross-platform capabilities of TensorFlow 2: you can run Keras on TPU or large clusters of GPUs, and you can export your Keras models to run in the browser or on a mobile device.

#### 2.3.6.4 Google Colab

Google Colaboratory, or "Colab" for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser. It is especially well suited to machine learning, data analysis, and education. It allows speeding up the program by using the GPU and TPU.

## 2.4 Summary

In this project, we have classified images from a publicly available brain tumor MRI-based dataset using several deep learning approaches and models, benefiting from many deep learning development tools. In this chapter, we discussed the methodology of our project. First, we discussed our dataset and plotted the distribution of its samples. After that, we went through the used deep learning models and their architectures, namely: AlexNet, VGG16, VGG19, ResNet50, and Xception. Finally, the chapter discusses the tools used for implementing the mentioned models, such as Python, Tensorflow, Keras, and Google Cloab.

# Chapter 3

# Results and Discussion

## 3.1 Introduction

In this chapter, we will discuss the various performance metrics used to implement and evaluate deep learning models. Then, we will discuss the results obtained from the techniques we outlined earlier in the second chapter.

## 3.2 Performance Metrics

The main goal of developing a deep learning model is to create one that can generalize and classify unseen data. We use metrics to evaluate and visualize how well the model is performing. There are a wide variety of metrics that may be used to evaluate deep learning models, but we will focus on the four most commonly used, namely: accuracy, precision, recall, and F1-score.

### 3.2.1 Confusion Matrix

The confusion matrix is one of the measures for determining the model's correctness and accuracy. It is a square matrix used to visualize the performance of the model. Each entry in a confusion matrix represents the number of predictions generated by the model on the test set, correctly and wrongly classifying the classes. In our case, the following four parameters make up the confusion matrix:

**True positive**: The number of accurately classified tumor photos.

**True negative**: The number of accurately classified non-tumor photos.

**False positive**: The number of non-tumor photos that have been misclassified as tumors.

**False negative**: The number of tumor photos that have been misclassified as non-tumor images.

Fig 3.1 demonstrates a confusion matrix:

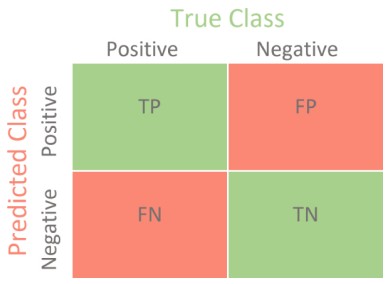


Figure 3.1: Binary classification confusion matrix [44].

### 3.2.2 Average Accuracy

The average accuracy, often known as the overall accuracy, is the percentage of total samples correctly classified by the classifier. It is calculated using the following formula:

$$average\,accuracy = \frac{1}{N}\sum_{i=1}^{N}\frac{TP+TN}{TP+TN+FP+FN} \tag{3.1}$$

### 3.2.3 Precision

The fraction of correct positive predictions among all positive predictions is known as the precision. The equation that represents it is:

$$precision = \frac{TP}{TP+FP} \tag{3.2}$$

### 3.2.4 Recall

This metric calculates the capacity of the model to make accurate predictions for each class. Its mathematical modeling is seen below:

$$recall = \frac{TP}{TP+FN} \tag{3.3}$$

In binary classification, recall of the positive class is "sensitivity" and recall of the negative class is "specificity".

### 3.2.5 F1-score

The F1-score is defined as the harmonic mean of precision and recall. It measures the model's performance and provides a better assessment of erroneously classified samples. It is represented by the following equation:

$$F1 - score = \frac{precision * recall}{precision + recall} \tag{3.4}$$

### 3.2.6 Reported Averages

**Macro Average:** Averaging the unweighted mean per label.

**Weighted Average** Averaging the support-weighted mean per label.

## 3.3 Results and Discussion

Overall, we implemented the previously explained CNN models using three approaches, namely: implementation from scratch, transfer learning (frozen weights), and fine-tuning (unfrozen weights). The details of the three experiments is presented in the following.

### 3.3.1 First Approach: Implementation from Scratch

#### 3.3.1.1 Alexnet

The Alexnet model has 87,655,170 total parameters (87,652,418 trainable and 2,752 non-trainable), using 2385 samples where it resulted in the learning curves of the accuracy and loss over 100 epochs, as shown in Fig 3.2 and Fig 3.3 respectively.

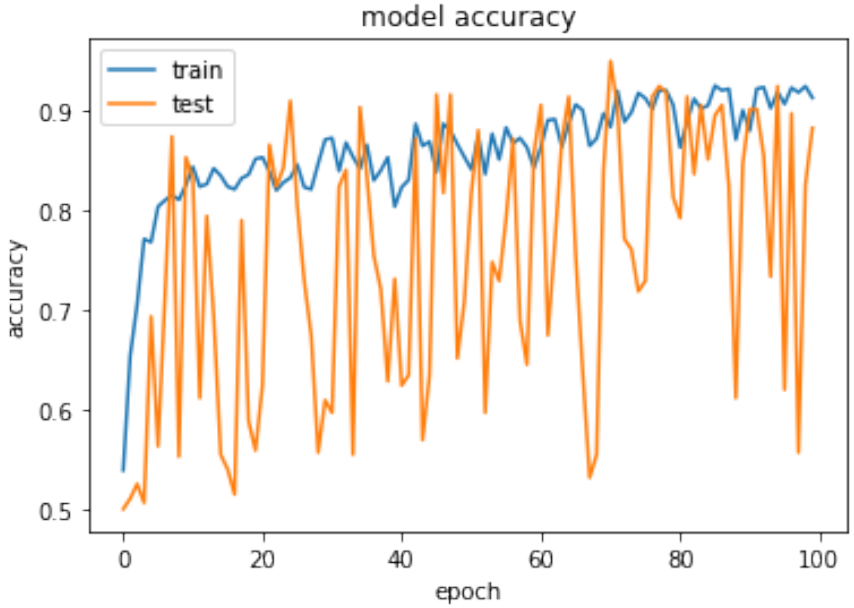


Figure 3.2: Training and validation accuracies curve of Alexnet model using first approach.

It can be seen from the accuracy curve that the model is doing well on training data but

less performing on the unseen validation data. We can also notice the fluctuation in the validation accuracy which means that the number of correct predictions is rising and falling irregularly.

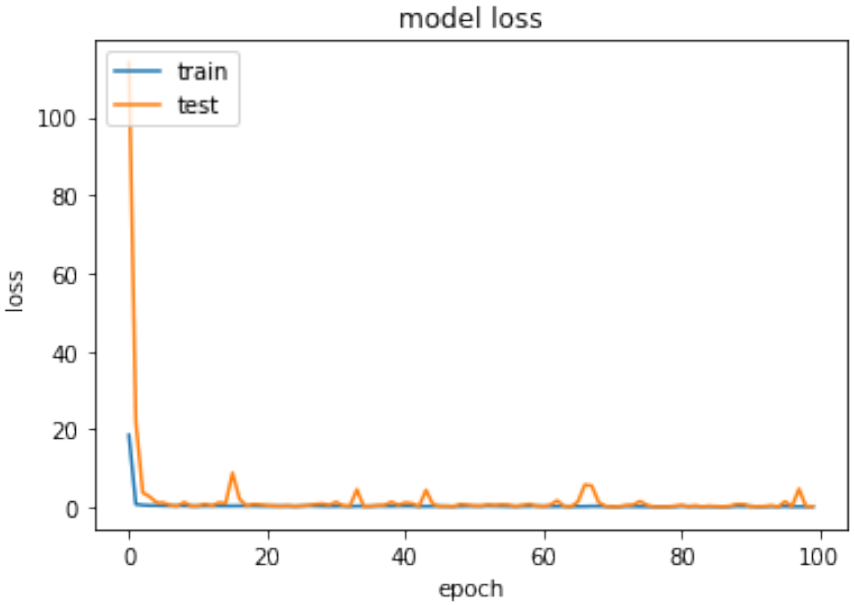


Figure 3.3: Loss curve of Alexnet model using first approach.

It can be seen from the loss curves that the validation loss is always greater than the training loss. This confirms that our model is poorly generalizing on unseen data.

The confusion matrix is plotted to verify and compare the misclassified labels and the correctly classified labels as can be seen in 3.4:

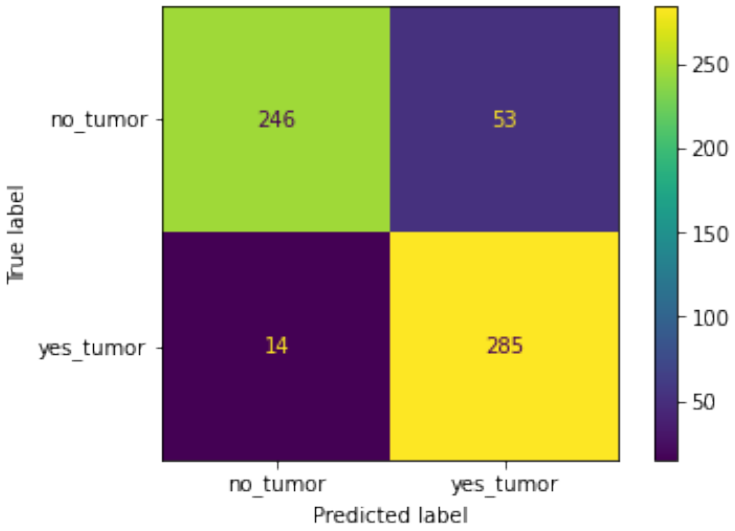


Figure 3.4: Confusion matrix of Alexnet model using first approach.

The resulted evaluations metrics are given in Table 3.1:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 0.95 | 0.82 | 0.88 | 299 |
| 1.0 | 0.84 | 0.95 | 0.89 | 299 |
| accuracy | | | 0.89 | 598 |
| macro avg | 0.89 | 0.89 | 0.89 | 598 |
| weighted avg | 0.89 | 0.89 | 0.89 | 598 |

Table 3.1: Classification report of AlexNet model using first approach

#### 3.3.1.2 VGG16

The VGG16 model has 21,220,674 total parameters (21,212,226 trainable and 8,448 non-trainable), using 2385 samples where it resulted in the learning curves of the accuracy and loss over 100 epochs, as shown in Fig 3.5 and Fig 3.6 respectively:

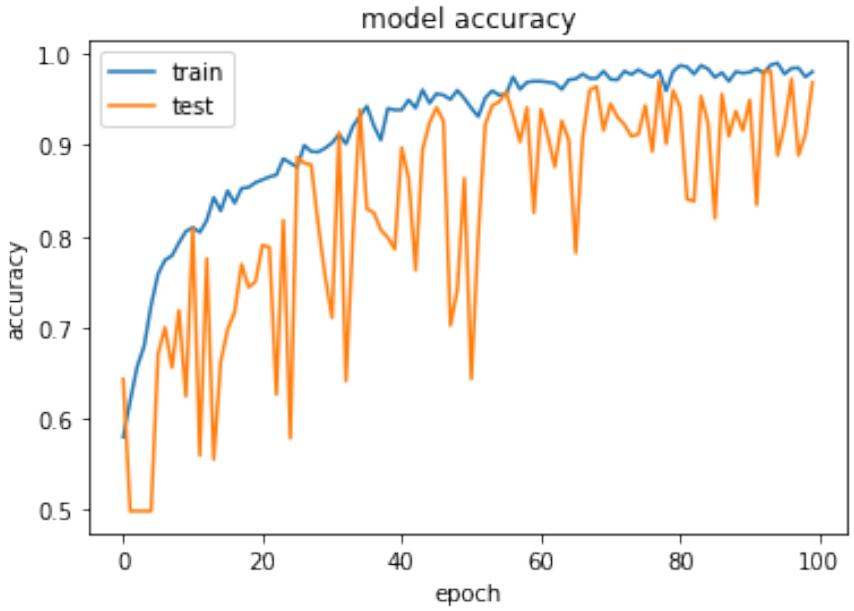


Figure 3.5: Tradining and validation accuracies curve of VGG16 model using first approach

The gap between the training and validation accuracies means that we are performing better in training than in validation, so the model is poorly generalizing on unseen data. However, the fluctuations in the validation data were reduced compared to AlexNet. The small-size convolution filters allow VGG16 to have a large number of weights, which leads to improved performance.

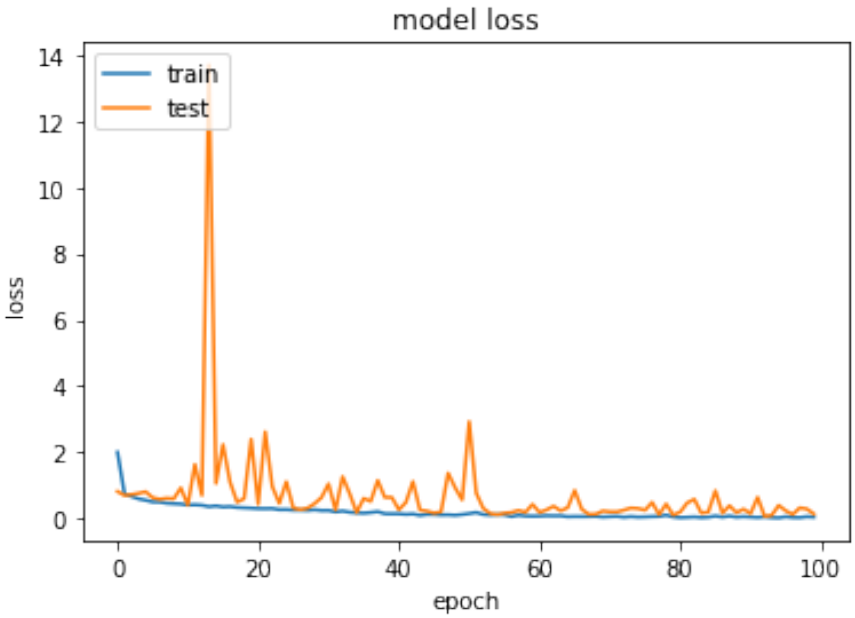


Figure 3.6: Loss curve of VGG16 model using first approach

VGG16 showed remarkable improvement in predicting correct labels making the validation loss closer to the training loss.

The confusion matrix of the VGG16 model using the first approach can be seen in Fig 3.7:

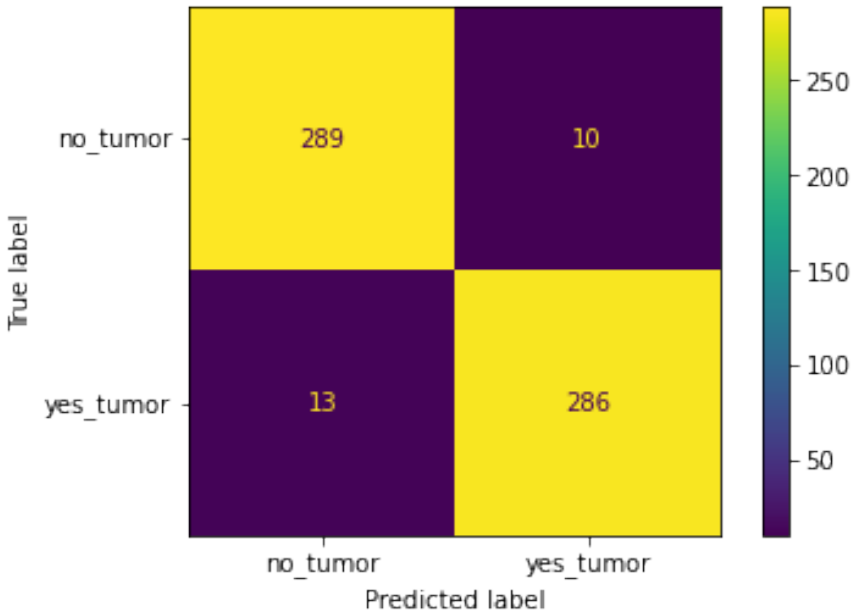


Figure 3.7: Confusion matrix of VGG16 model using first approach

The classification report of the VGG16 model using the first approach can be seen in Table 3.2:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 0.96 | 0.97 | 0.96 | 299 |
| 1.0 | 0.97 | 0.96 | 0.96 | 299 |
| accuracy | | | 0.96 | 598 |
| macro avg | 0.96 | 0.96 | 0.96 | 598 |
| weighted avg | 0.96 | 0.96 | 0.96 | 598 |

Table 3.2: Classification report of VGG16 model using first approach

#### 3.3.1.3 VGG19

The VGG19 model has 26,535,490 total parameters (26,524,482 trainable and 11,008 non-trainable), using 2385 samples where it resulted in the learning curves of the accuracy and loss over 100 epochs, as shown in Fig 3.8 and Fig 3.9 respectively:

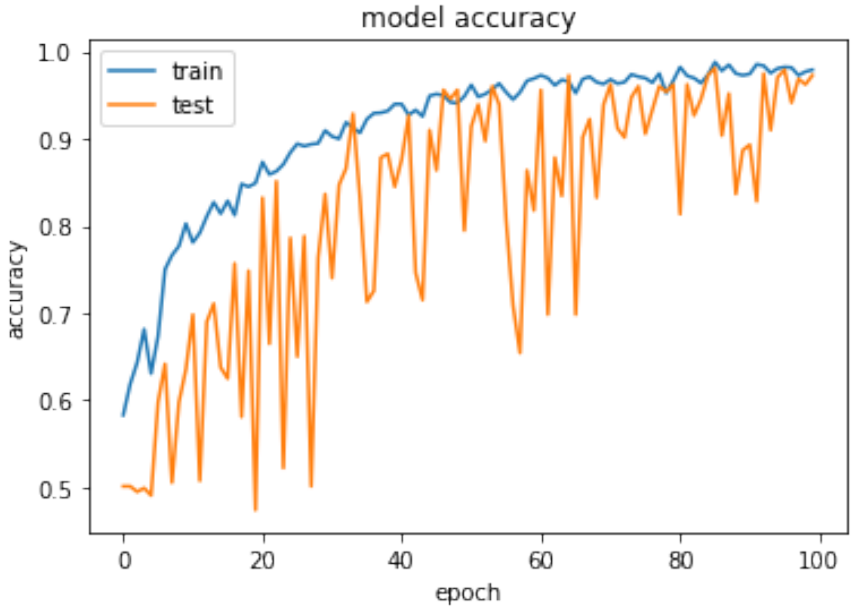


Figure 3.8: Accuracy curve of VGG19 model using first approach

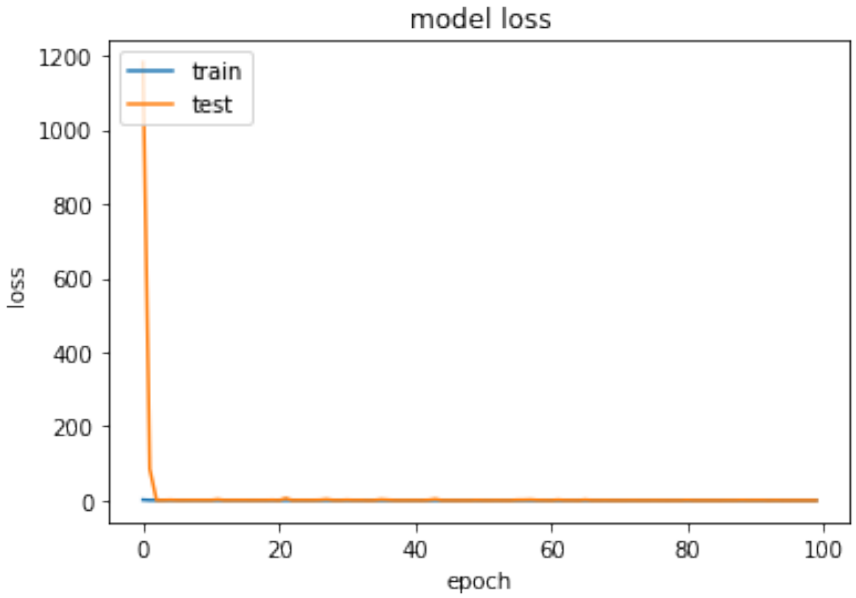


Figure 3.9: Loss curve of VGG19 model using first approach

The VGG19 model is a variant of VGG16, the only difference is that VGG19 has three additional convolutional layers. VGG19 behaved during this experiment exactly as VGG16 in terms of accuracy as well as loss.

The confusion matrix of the VGG19 model using the first approach can be seen in Fig 3.10:

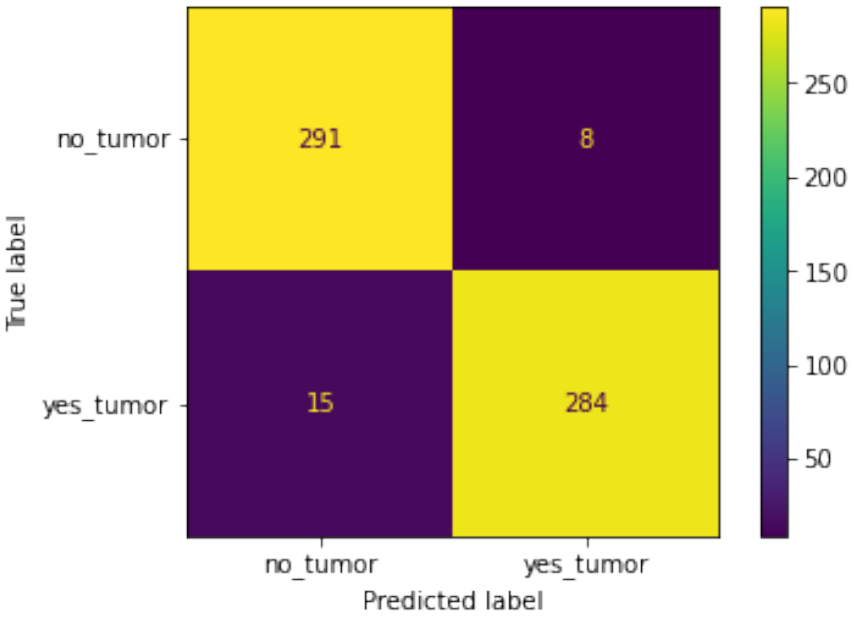


Figure 3.10: Confusion matrix of VGG19 model using first approach

The classification report of the VGG19 model using the first approach can be seen in Table 3.3 :

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 0.95 | 0.97 | 0.96 | 299 |
| 1.0 | 0.97 | 0.95 | 0.96 | 299 |
| accuracy | | | 0.96 | 598 |
| macro avg | 0.96 | 0.96 | 0.96 | 598 |
| weighted avg | 0.96 | 0.96 | 0.96 | 598 |

Table 3.3: Classification report of VGG19 model using first approach

#### 3.3.1.4 ResNet50

The ResNet50 model has 23,591,810 total parameters (23,538,690 trainable and 53,120 non-trainable), using 2385 samples where it resulted in the learning curves of the accuracy and loss over 100 epochs, as shown in Fig **??** and Fig 3.9 respectively:

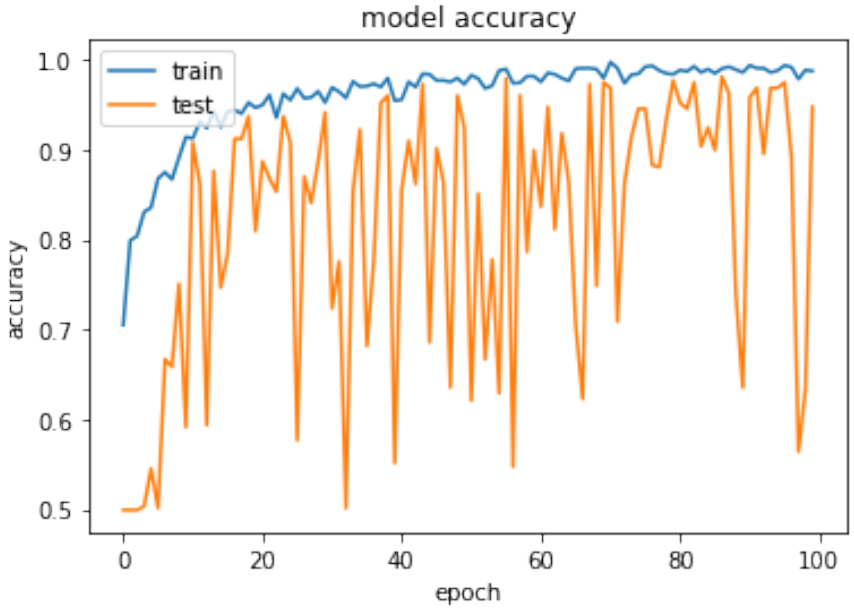


Figure 3.11: Accuracy curve of ResNet50 model using first approach

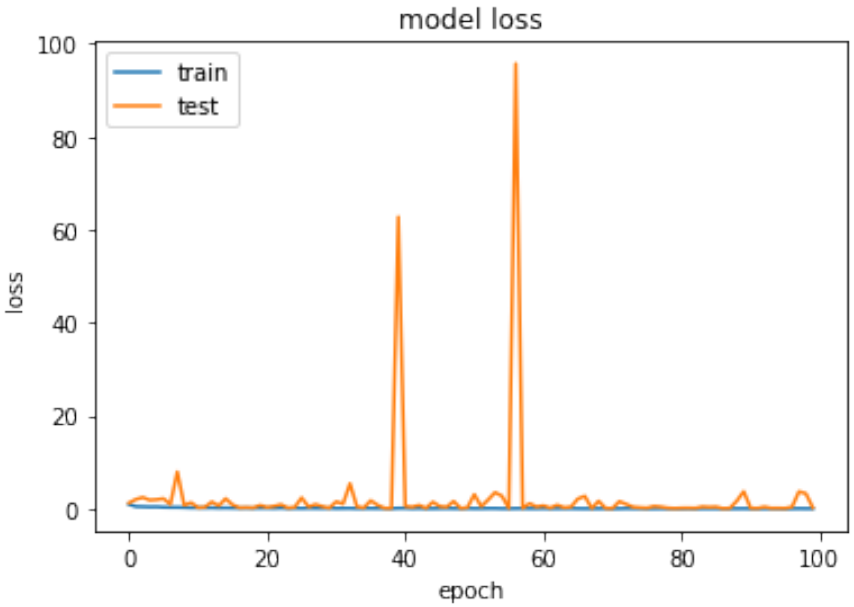


Figure 3.12: Loss curve of VGG19 model using first approach

The gap between validation and training accuracies is due to the inability to generalize on new unseen data. The spikes in the loss function are consequences of the mini-batch gradient descent in the ADAM optimizer because some batches are noisy or harder to learn than others.

The confusion matrix of the ResNet50 model using the first approach can be seen in Fig 3.13:

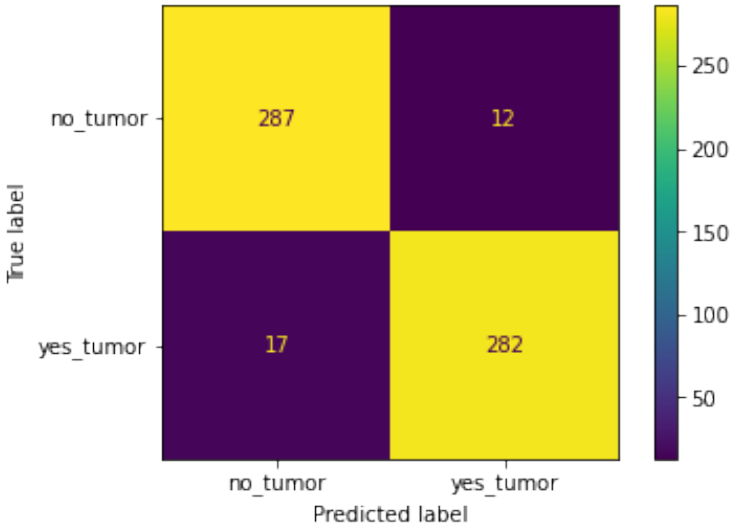


Figure 3.13: Confusion matrix of ResNet50 model using first approach

The classification report of ResNet50 model using first approach can be seen in Table 3.4:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 0.94 | 0.96 | 0.95 | 299 |
| 1.0 | 0.96 | 0.94 | 0.95 | 299 |
| accuracy | | | 0.95 | 598 |
| macro avg | 0.95 | 0.95 | 0.95 | 598 |
| weighted avg | 0.95 | 0.95 | 0.95 | 598 |

Table 3.4: Classification report of ResNet50 model using first approach

#### 3.3.1.5 Xception

The Xception model has 2,782,649 total parameters (2,773,913 trainable and 8,736 non-trainable), using 2385 samples where it resulted in the learning curves of the accuracy and loss over 100 epochs, as shown in Fig 3.14 and Fig 3.15 respectively:

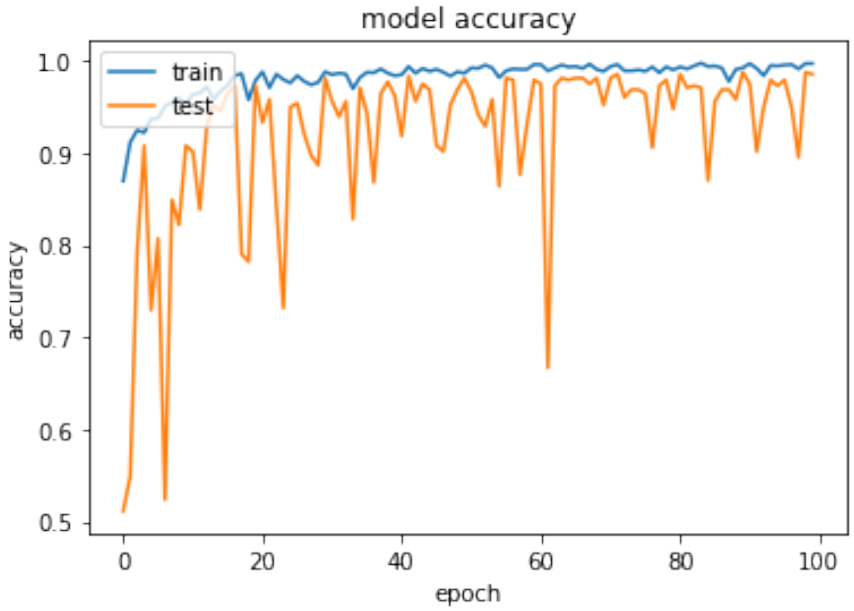


Figure 3.14: Accuracy curve of Xception model using first approach

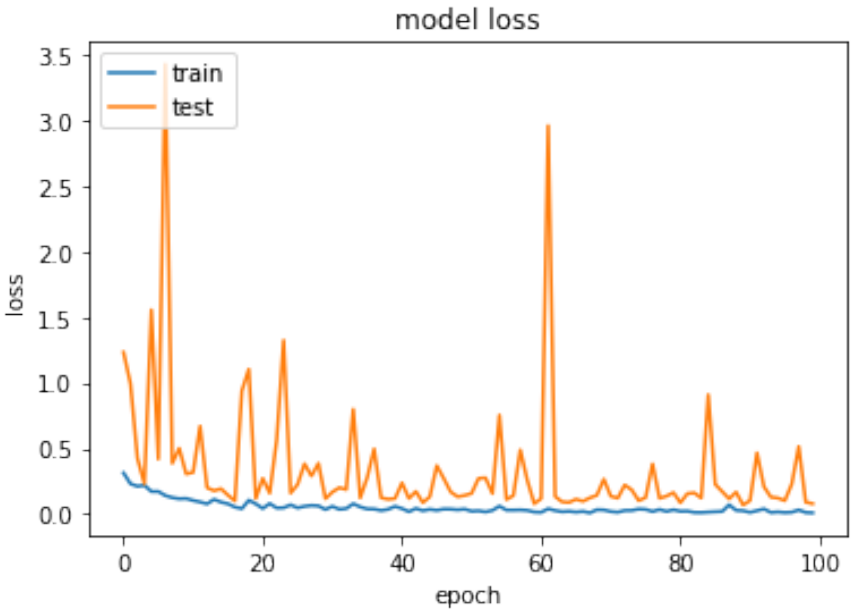


Figure 3.15: Loss curve of Xception model using first approach

The gap between training and validation accuracies was reduced significantly as compared to the other models as well as the range of fluctuations. This improvement is because the model is much deeper, it uses identity mapping to prevent gradient vanishing, and it uses depth-wise separable convolution which has fewer parameters than classic convolutional layers and thus is less prone to overfitting.

The confusion matrix of the Xception model using the first approach can be seen in Fig 3.16:

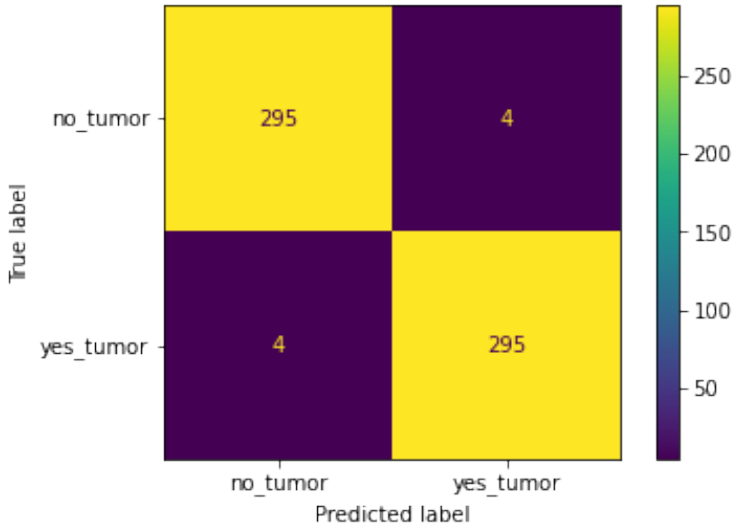


Figure 3.16: Confusion matrix of Xception model using first approach

The classification report of the Xception model using the first approach can be seen in Table 3.5:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 0.94 | 0.96 | 0.95 | 299 |
| 1.0 | 0.96 | 0.94 | 0.95 | 299 |
| accuracy | | | 0.95 | 598 |
| macro avg | 0.95 | 0.95 | 0.95 | 598 |
| weighted avg | 0.95 | 0.95 | 0.95 | 598 |

Table 3.5: Classification report of Xception model using first approach

### 3.3.2 Second Approach: Transfer Learning

#### 3.3.2.1 VGG16

The VGG16 model has 14,912,322 total parameters (197,634 trainable and 14,714,688 non-trainable), using 2385 samples where it resulted in the learning curves of the accuracy and loss over 100 epochs, as shown in Fig 3.17 and Fig 3.18 respectively:

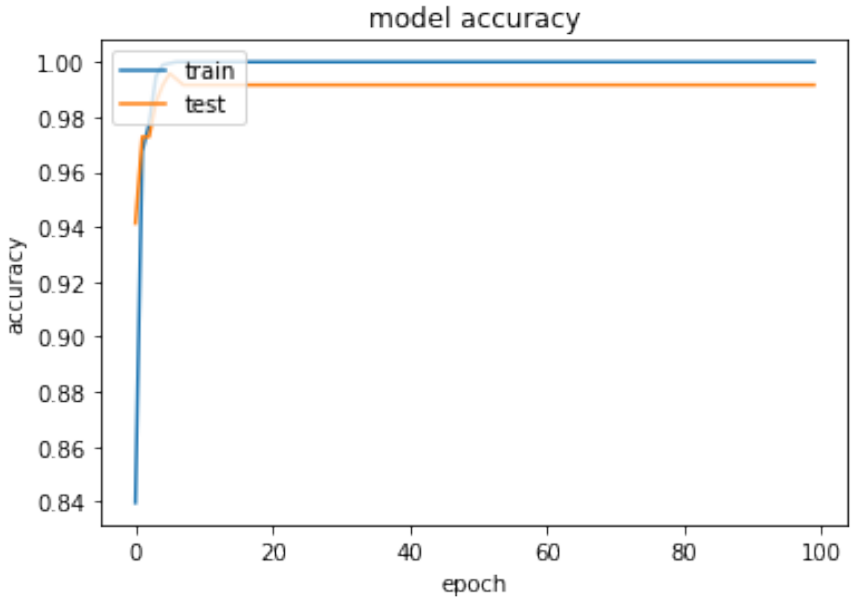


Figure 3.17: Accuracy curve of VGG16 model using second approach

The absence of fluctuations means that the model predictions are more accurate. We can also notice that the gap between training and validation accuracy is reduced, so our model generalizes better to unseen data.

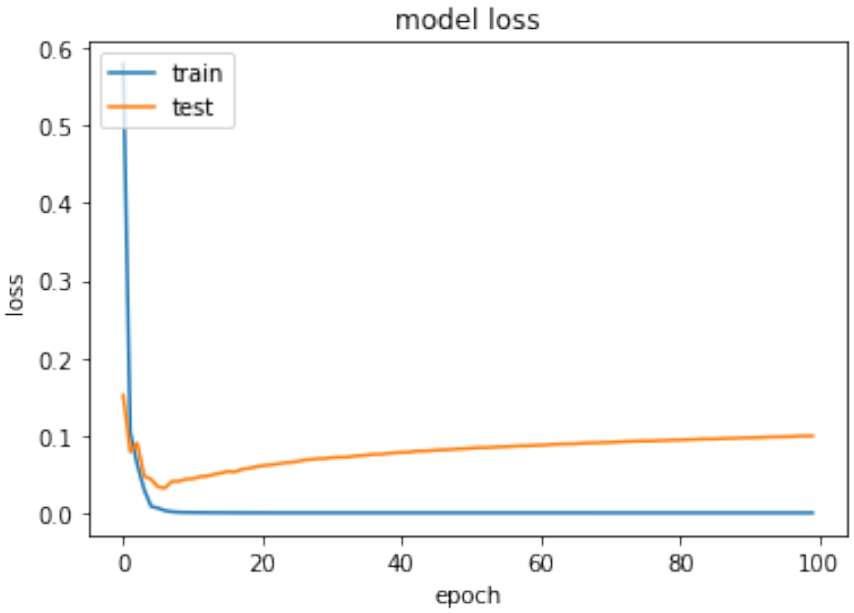


Figure 3.18: Loss curve of VGG16 model using second approach

The increase in the validation loss means that our model is suffering from vanishing gradient problem.

The confusion matrix of the VGG16 model using the second approach can be seen Fig 3.19:

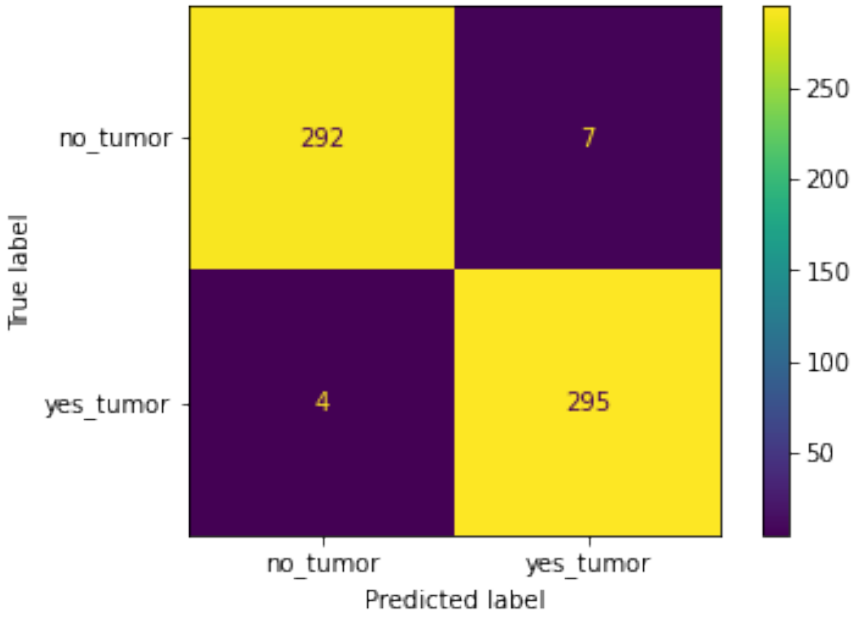


Figure 3.19: Confusion matrix of VGG16 model using second approach

The classification report of the VGG16 model using the second approach can be seen in Table 3.6:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 0.99 | 0.98 | 0.98 | 299 |
| 1.0 | 0.98 | 0.99 | 0.98 | 299 |
| accuracy | | | 0.98 | 598 |
| macro avg | 0.98 | 0.98 | 0.98 | 598 |
| weighted avg | 0.98 | 0.98 | 0.98 | 598 |

Table 3.6: Classification report of VGG16 model using second approach

#### 3.3.2.2 VGG19

The VGG19 model has 20,222,018 total parameters (197,634 trainable and 14,714,688 non-trainable), using 2385 samples where it resulted in the learning curves of the accuracy and loss over 100 epochs, as shown in Fig 3.20 and Fig 3.21 respectively:

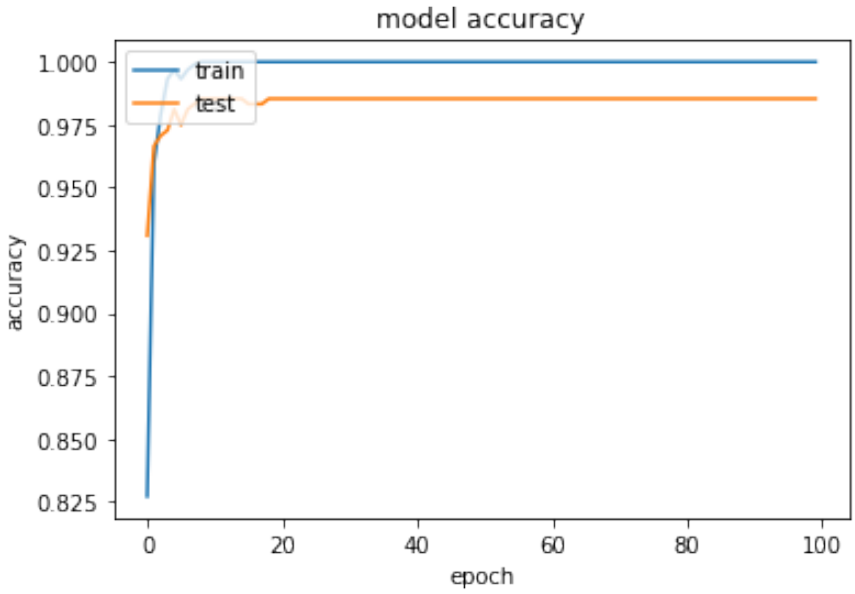


Figure 3.20: Accuracy curve of VGG19 model using second approach

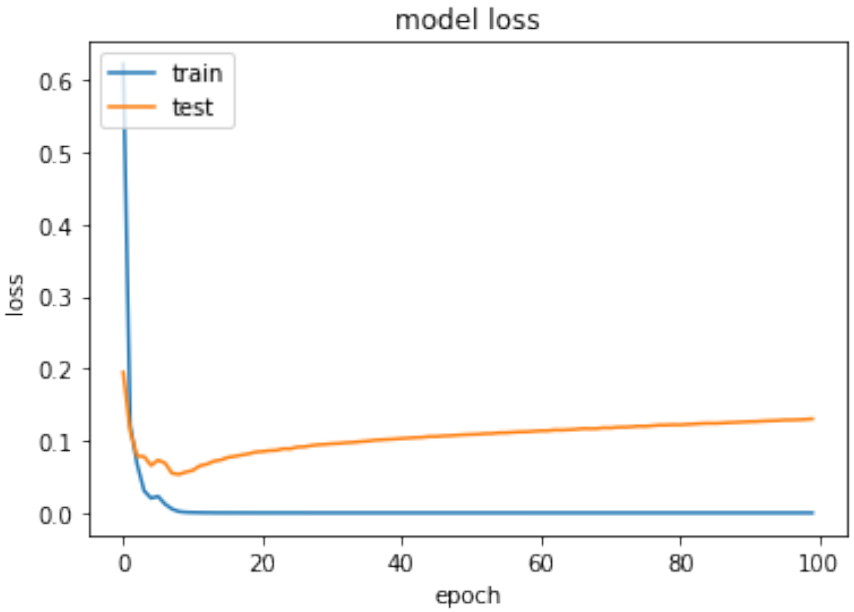


Figure 3.21: Loss curve of VGG19 model using second approach

As we said earlier, VGG19 is a variant of VGG16, so VGG19 behaved again during this experiment exactly as VGG16 in terms of accuracy as well as loss.

The confusion matrix of the VGG19 model using the second approach can be seen in Fig 3.22:

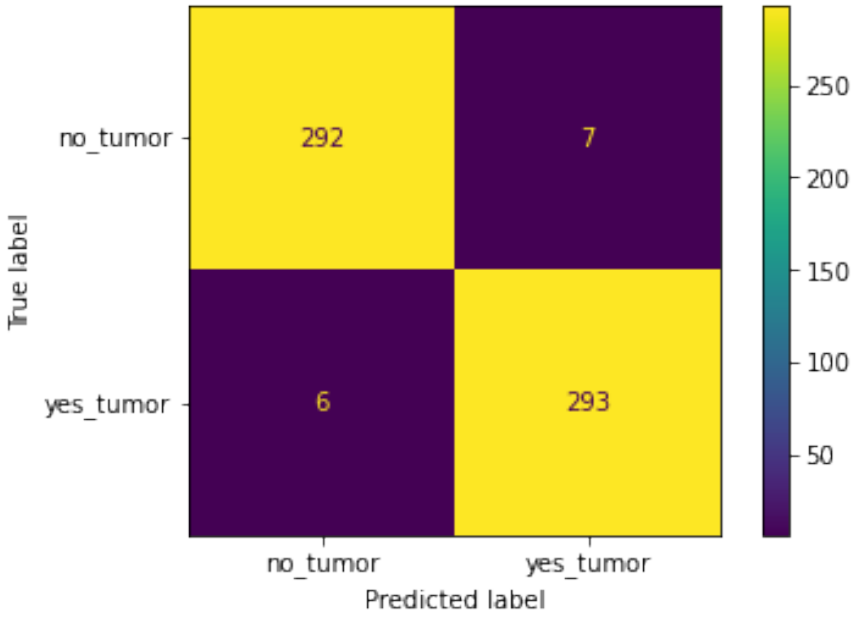


Figure 3.22: Confusion matrix of VGG19 model using second approach

The classification report of VGG19 model using second approach can be seen in Table 3.7:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 0.98 | 0.98 | 0.98 | 299 |
| 1.0 | 0.98 | 0.98 | 0.98 | 299 |
| accuracy | | | 0.98 | 598 |
| macro avg | 0.98 | 0.98 | 0.98 | 598 |
| weighted avg | 0.98 | 0.98 | 0.98 | 598 |

Table 3.7: Classification report of VGG19 model using second approach

#### 3.3.2.3 ResNet50

The ResNet50 model has 23,591,810 total parameters (4,098 trainable and 23,587,712 non-trainable), using 2385 samples where it resulted in the learning curves of the accuracy and loss over 100 epochs, as shown in Fig 3.23 and Fig 3.24 respectively:

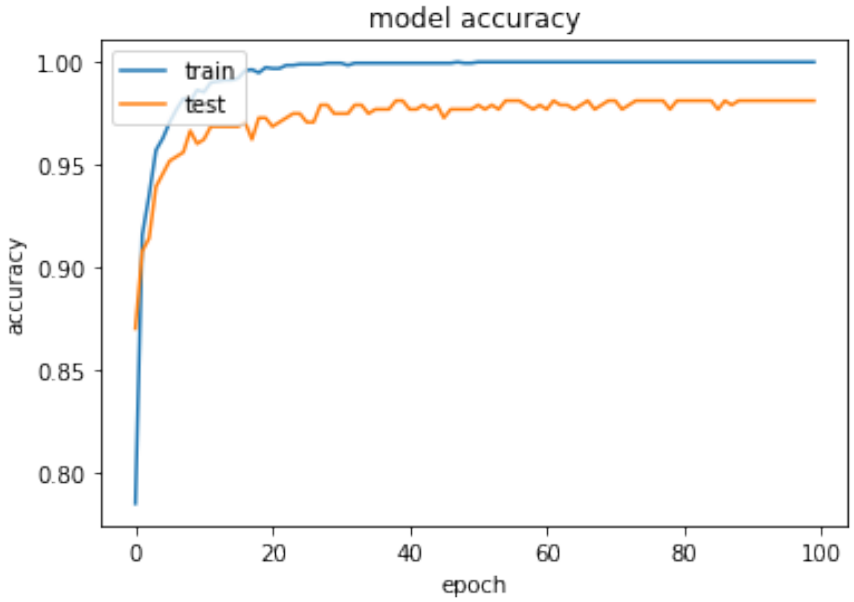


Figure 3.23: Accuracy curve of ResNet50 model using second approach

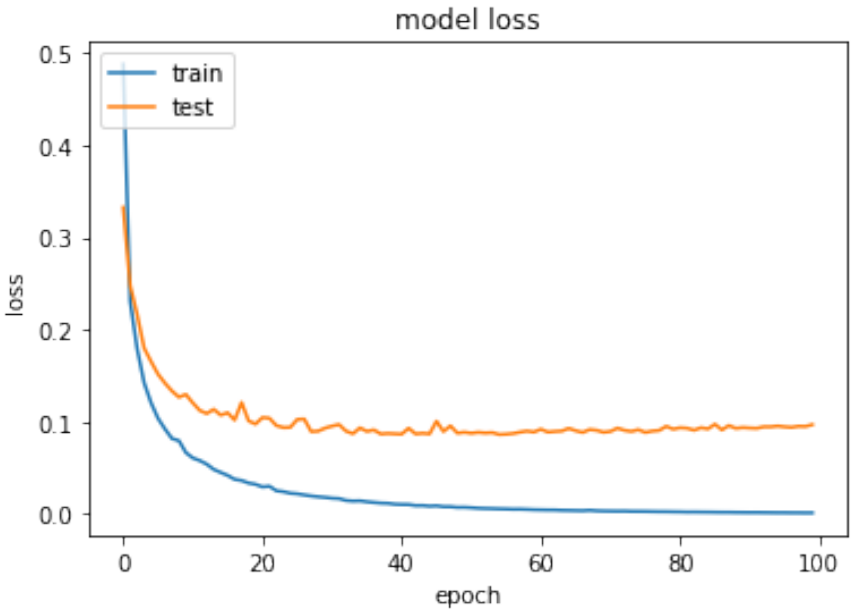


Figure 3.24: Loss curve of ResNet50 model using second approach

Unlike the previous two models, ResNet50 validation loss is not increasing. This is because it uses identity mappings to prevent vanishing problems.

The confusion matrix of the ResNet50 model using the second approach can be seen in Fig 3.25:

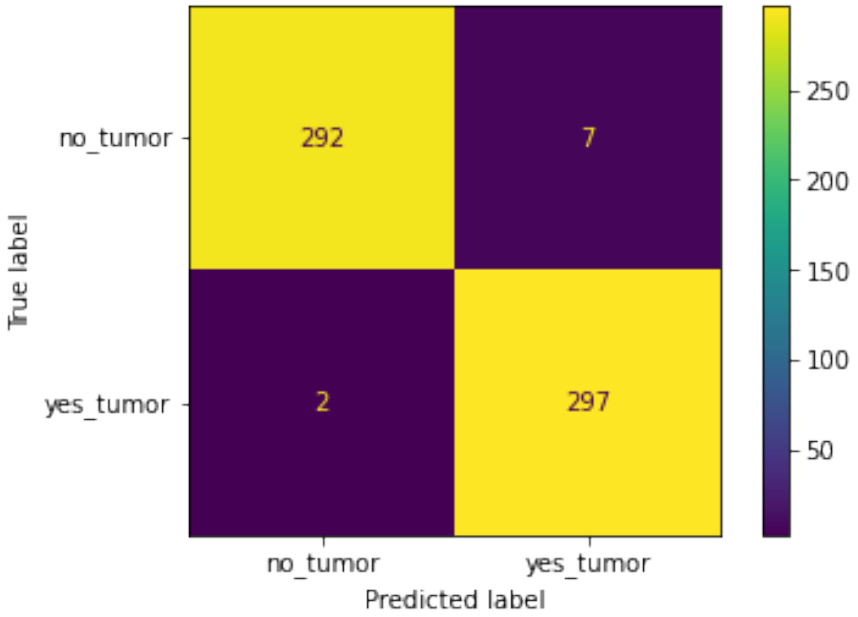


Figure 3.25: Confusion matrix of ResNet50 model using second approach

The classification report of the ResNet50 model using the second approach can be seen in Table 3.8:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 0.99 | 0.98 | 0.98 | 299 |
| 1.0 | 0.98 | 0.99 | 0.99 | 299 |
| accuracy | | | 0.98 | 598 |
| macro avg | 0.99 | 0.98 | 0.98 | 598 |
| weighted avg | 0.99 | 0.98 | 0.98 | 598 |

Table 3.8: Classification report of ResNet50 model using second approach

#### 3.3.2.4 Xception

The Xception model has 20,863,529 total parameters (2,049 trainable and 20,861,480 non-trainable), using 2385 samples where it resulted in the learning curves of the accuracy and loss over 100 epochs, as shown in Fig 3.26 and Fig 3.27 respectively:

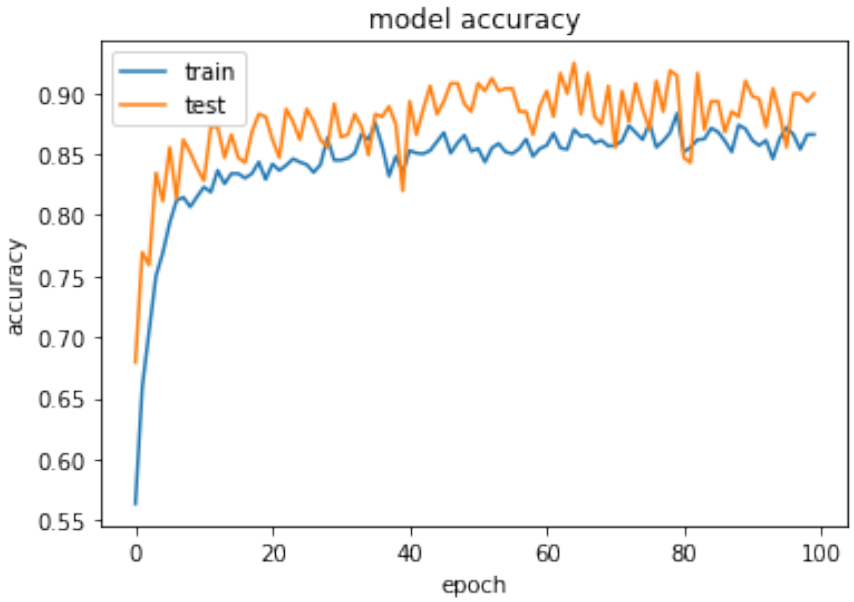


Figure 3.26: Accuracy curve of Xception model using second approach

The validation accuracy is higher than the training accuracy, it can also be seen that the accuracy is reduced as compared to the previous experiments, this is because Xception is a deeper and more complex model, so as the weights are frozen and can not be updated, it overfits.

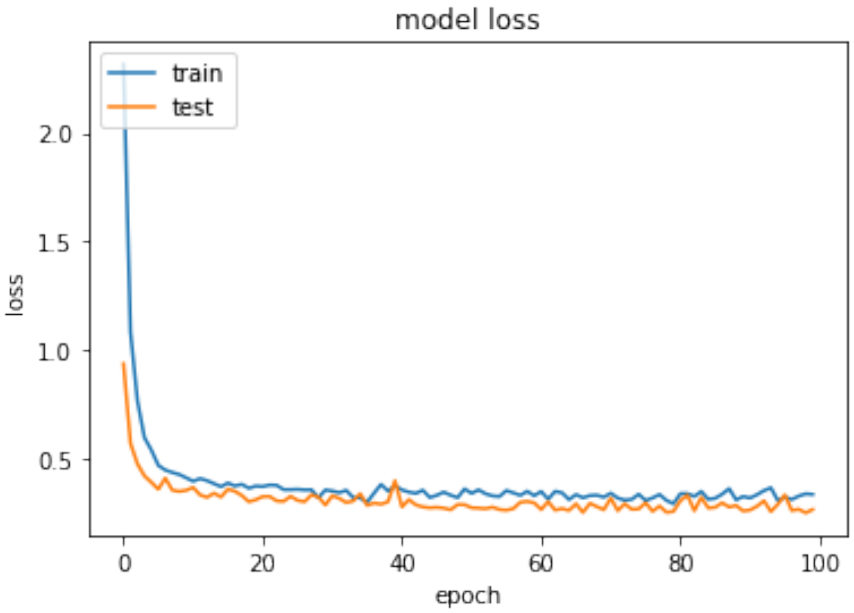


Figure 3.27: Loss curve of Xception model using second approach

The confusion matrix of the Xception model using the second approach can be seen in Fig 3.28:

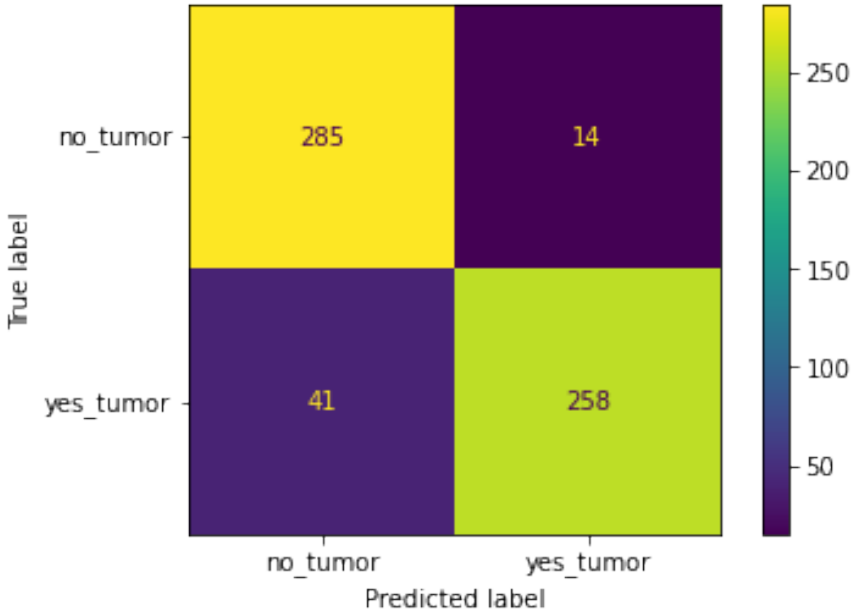


Figure 3.28: Confusion matrix of Xception model using second approach

The classification report of the Xception model using the second approach can be seen in Table 3.9:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 0.87 | 0.95 | 0.91 | 299 |
| 1.0 | 0.95 | 0.86 | 0.90 | 299 |
| accuracy | | | 0.91 | 598 |
| macro avg | 0.91 | 0.91 | 0.91 | 598 |
| weighted avg | 0.91 | 0.91 | 0.91 | 598 |

Table 3.9: Classification report of Xception model using second approach

### 3.3.3 Third Appraoch: Fine Tuning

#### 3.3.3.1 VGG16

The VGG16 model has 14,912,322 total parameters (14,912,322 trainable and 0 non-trainable), using 2385 samples where it resulted in the learning curves of the accuracy and loss over 100 epochs, as shown in Fig 3.29 and Fig 3.30 respectively:

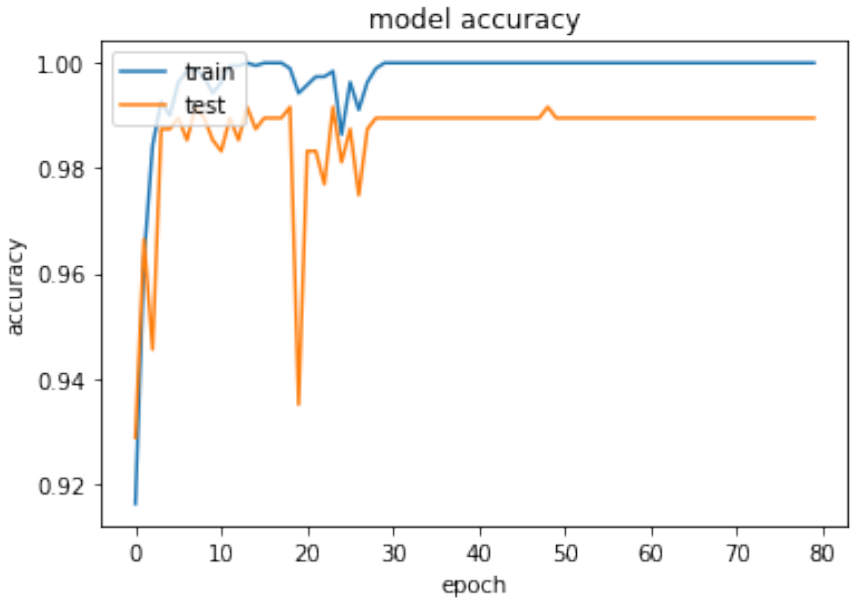


Figure 3.29: Accuracy curve of VGG16 model using third approach

The model performed well in both training and validation sets resulting in good close enough accuracies.

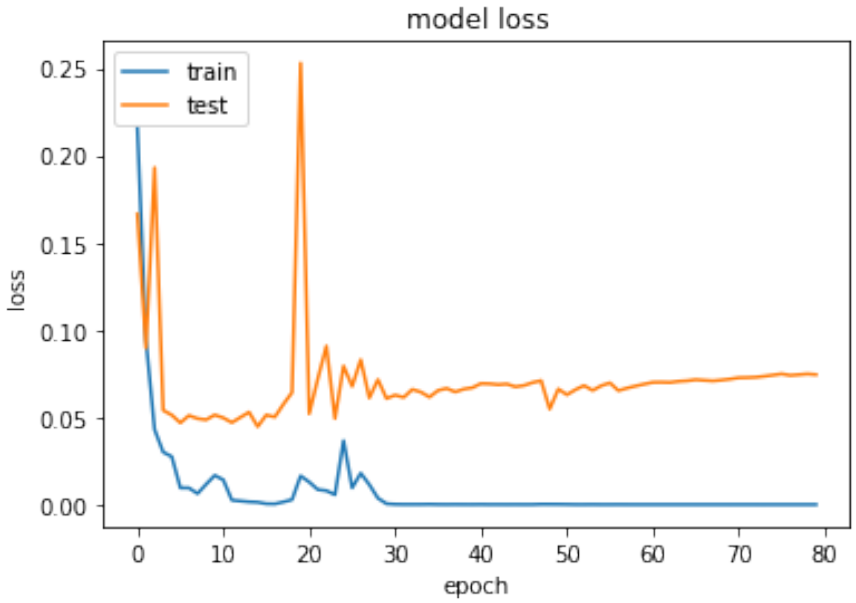


Figure 3.30: Loss curve of VGG16 model using third approach

The increase in the validation loss means that the model is still suffering from the vanishing gradient problem.

The confusion matrix of the VGG16 model using the third approach can be seen in Fig 3.31:

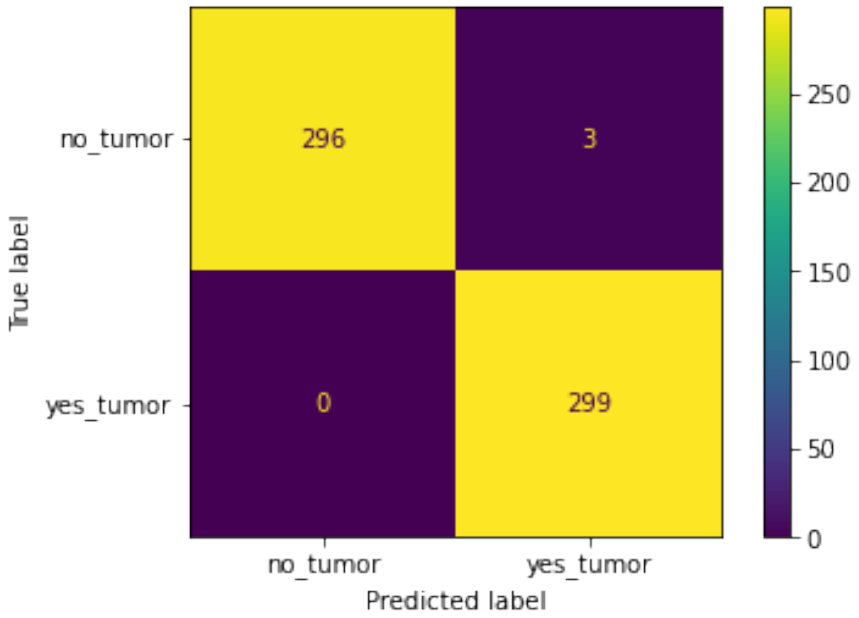


Figure 3.31: Confusion matrix of VGG16 model using third approach

The classification report of the VGG16 model using the third approach can be seen in Table

3.10:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 1.00 | 0.99 | 0.99 | 299 |
| 1.0 | 0.99 | 1.00 | 1.00 | 299 |
| accuracy | | | 0.99 | 598 |
| macro avg | 1.00 | 0.99 | 0.99 | 598 |
| weighted avg | 1.00 | 0.99 | 0.99 | 598 |

Table 3.10: Classification report of VGG16 model using third approach

#### 3.3.3.2 VGG19

The VGG19 model has 20,222,018 total parameters (20,222,018 trainable and 10 non-trainable), using 2385 samples where it resulted in the learning curves of the accuracy and loss over 100 epochs, as shown in Fig 3.32 and Fig 3.33 respectively:

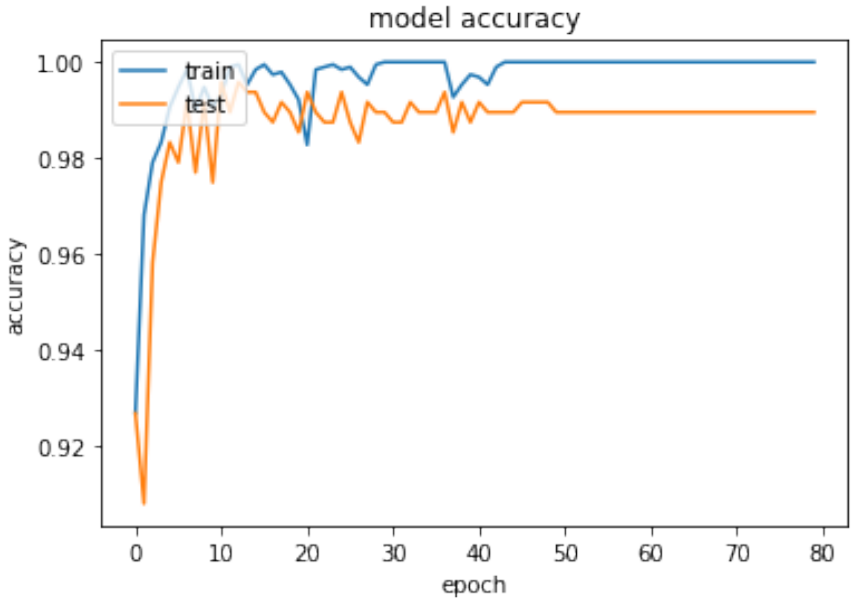


Figure 3.32: Accuracy curve of VGG19 model using third approach

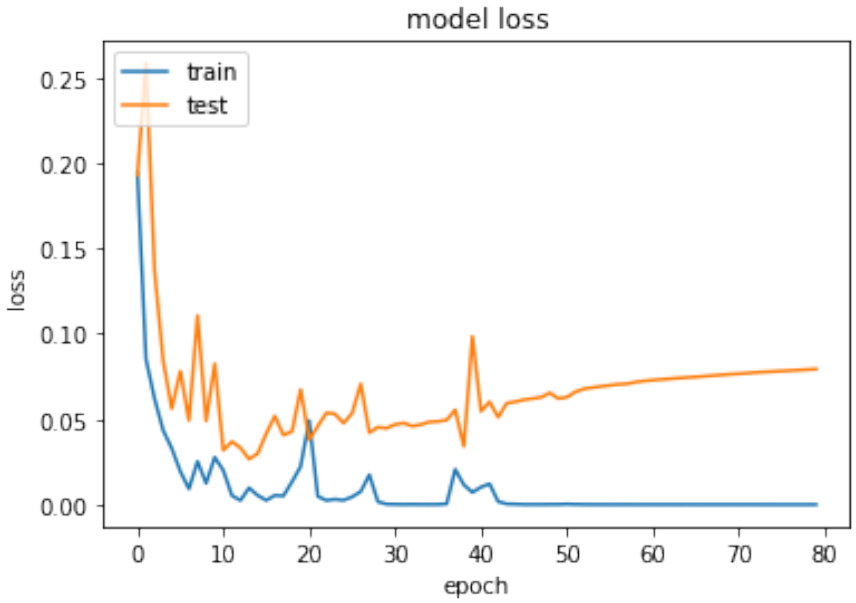


Figure 3.33: Loss curve of VGG19 model using third approach

As mentioned earlier, VGG19 is a variant of VGG16, so VGG19 behaved again during this experiment exactly as VGG16 in terms of accuracy as well as loss.

The confusion matrix of the VGG19 model using the third approach can be seen in Fig 3.34:

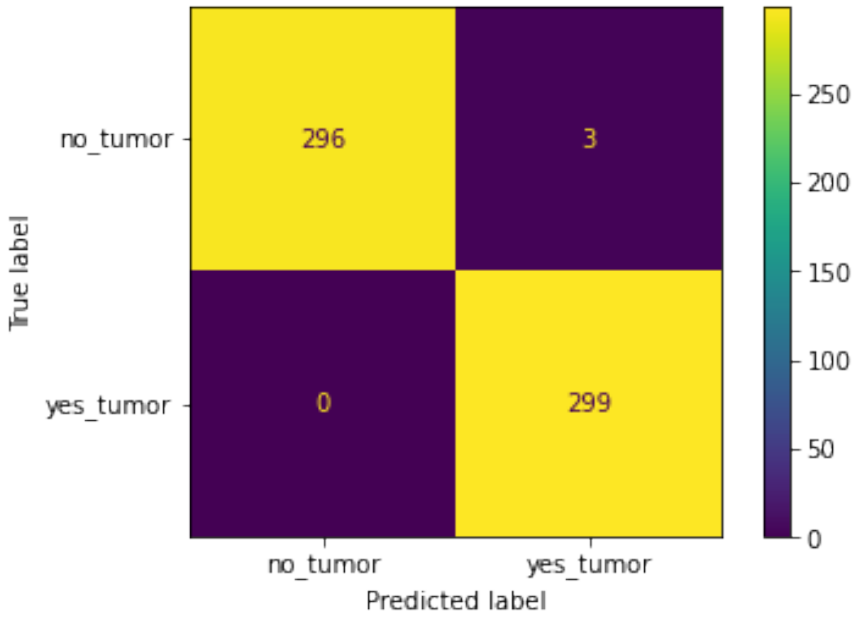


Figure 3.34: Confusion matrix of VGG19 model using third approach

The classification report of the VGG19 model using the third approach can be seen in Table

3.11:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 1.00 | 0.99 | 0.99 | 299 |
| 1.0 | 0.99 | 1.00 | 1.00 | 299 |
| accuracy | | | 0.99 | 598 |
| macro avg | 1.00 | 0.99 | 0.99 | 598 |
| weighted avg | 1.00 | 0.99 | 0.99 | 598 |

Table 3.11: Classification report of VGG19 model using third approach

#### 3.3.3.3 ResNet50

The ResNet50 model has 23,591,810 total parameters (23,538,690 trainable and 53,120 non-trainable), using 2385 samples where it resulted in the learning curves of the accuracy and loss over 100 epochs, as shown in Fig 3.35 and Fig 3.36 respectively:

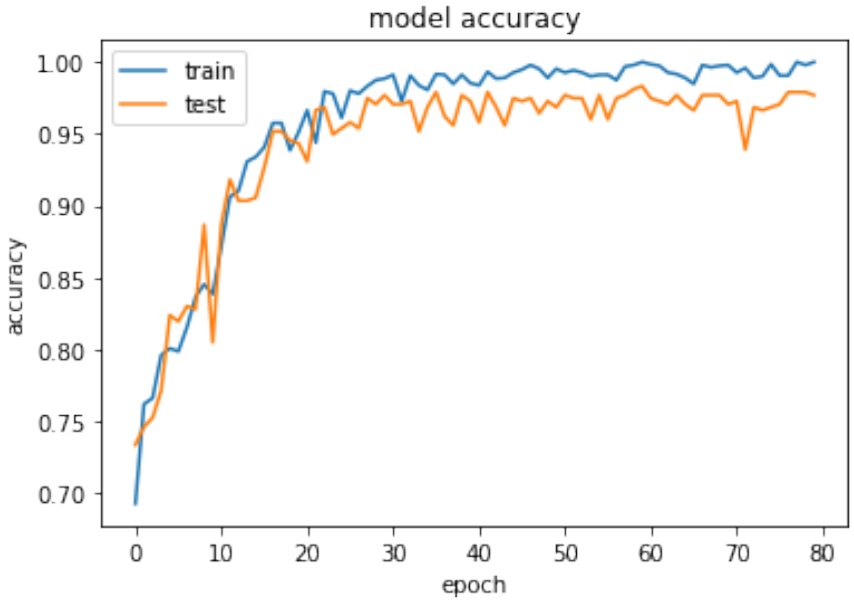


Figure 3.35: Accuracy curve of ResNet50 model using third approach

The model performed well in both training and validation sets resulting in good close enough accuracies.

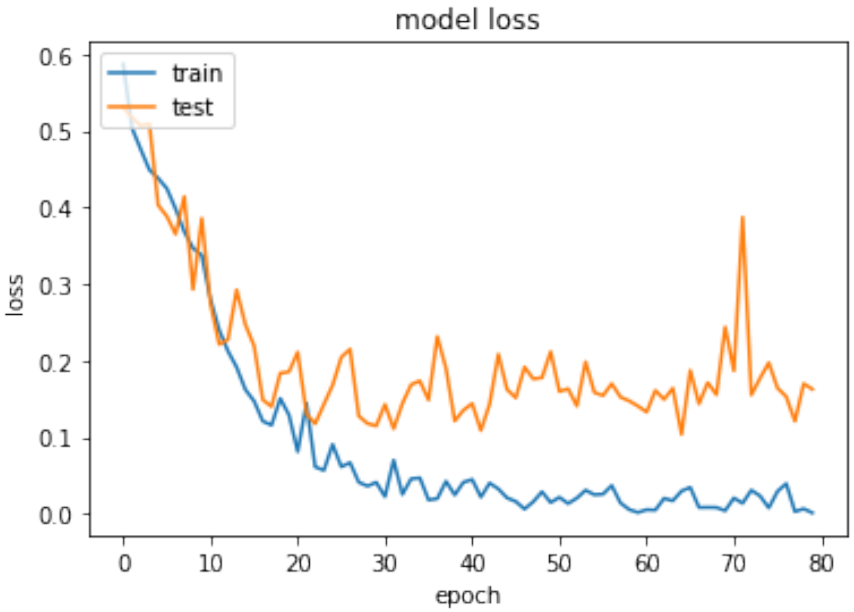


Figure 3.36: Loss curve of ResNet50 model using third approach

Again, ResNet50 has shown remarkable performance in preventing vanishing gradient problems, ResNet50 validation loss is not increasing. This is because it uses identity mappings.

The confusion matrix of the ResNet50 model using the third approach can be seen in Fig 3.37:

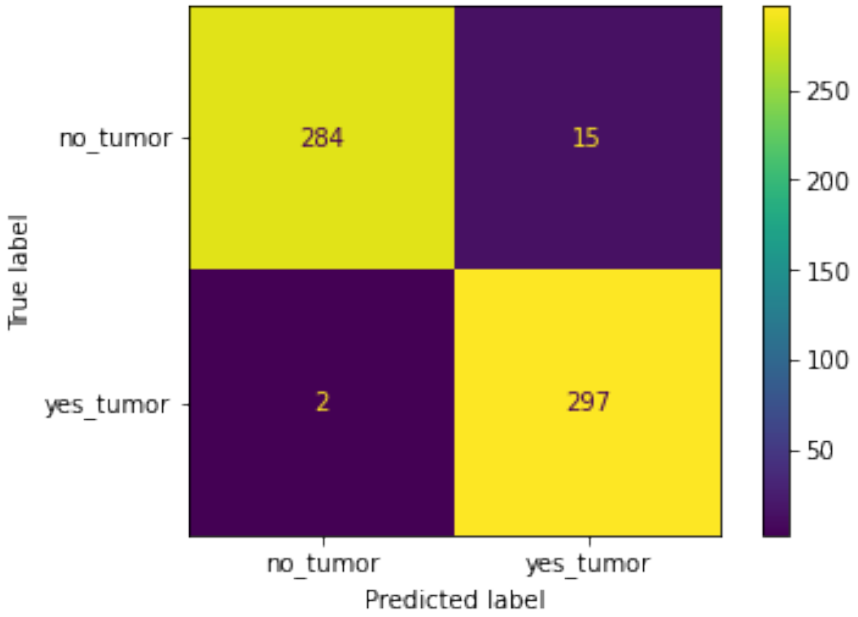


Figure 3.37: Confusion matrix of ResNet50 model using third approach

The classification report of the ResNet50 model using the third approach can be seen in Table 3.12:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 0.99 | 0.95 | 0.97 | 299 |
| 1.0 | 0.95 | 0.99 | 0.97 | 299 |
| accuracy | | | 0.97 | 598 |
| macro avg | 0.97 | 0.97 | 0.97 | 598 |
| weighted avg | 0.97 | 0.97 | 0.97 | 598 |

Table 3.12: Classification report of ResNet50 model using third approach

#### 3.3.3.4 Xception

The Xception model has 20,863,529 total parameters (20,809,001 trainable and 54,528 non-trainable), using 2385 samples where it resulted in the learning curves of the accuracy and loss over 100 epochs, as shown in Fig 3.38 and Fig 3.39 respectively:

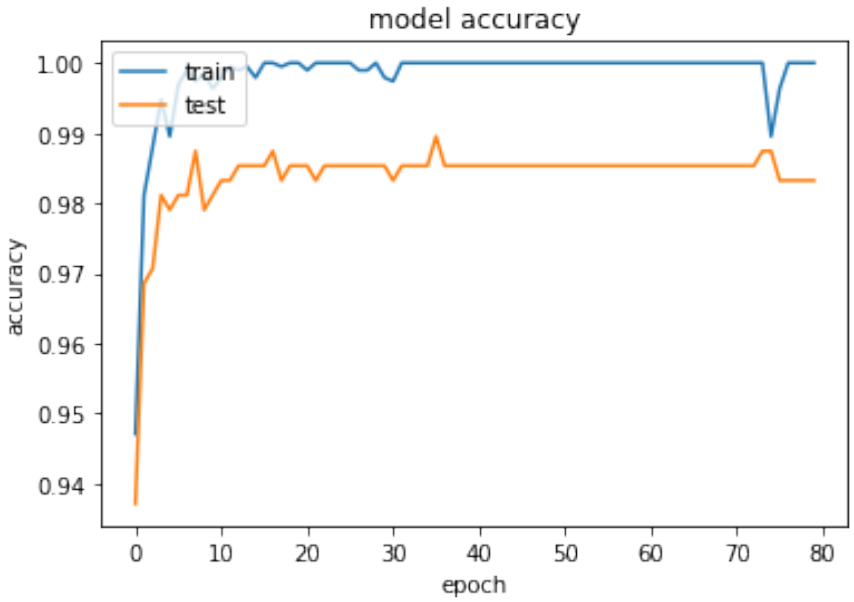


Figure 3.38: Accuracy curve of Xception model using third approach

The model performed well in both training and validation sets resulting in good close enough accuracies.

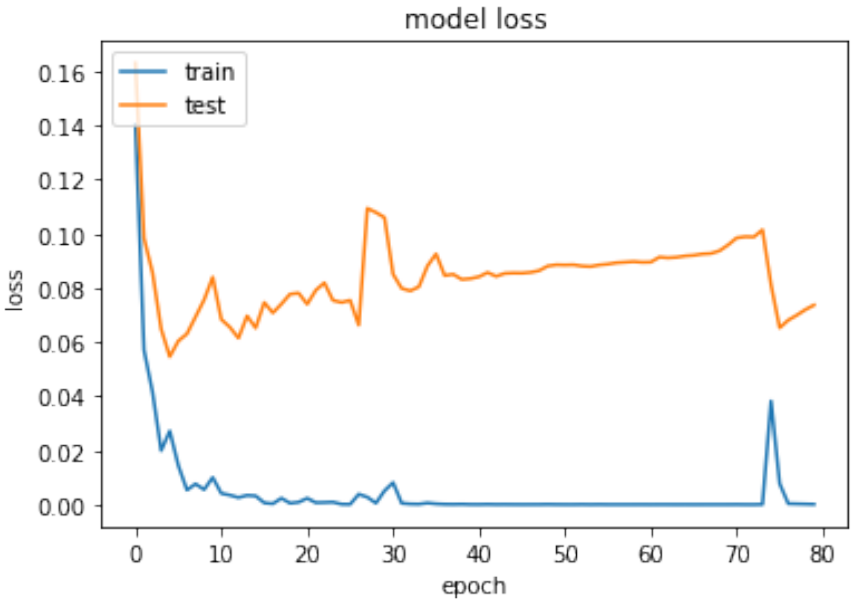


Figure 3.39: Loss curve of Xception model using third approach

Xception has shown remarkable performance in preventing vanishing gradient problems by using identity mappings.

The confusion matrix of the Xception model using the third approach can be seen in Fig 3.40:

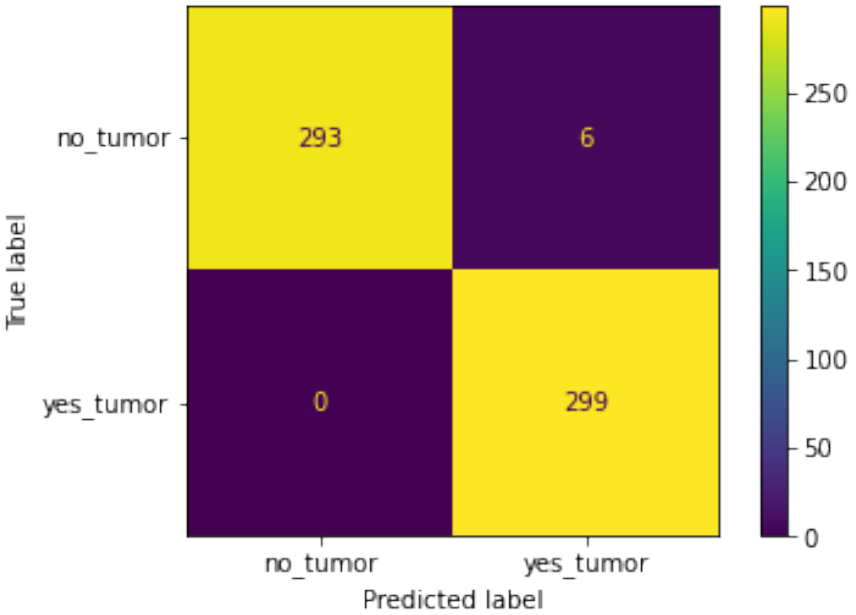


Figure 3.40: Confusion matrix of Xception model using third approach

The classification report of the Xception model using the third approach can be seen in Table 3.13:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 1.00 | 0.98 | 0.99 | 299 |
| 1.0 | 0.98 | 1.00 | 0.99 | 299 |
| accuracy | | | 0.99 | 598 |
| macro avg | 0.99 | 0.99 | 0.99 | 598 |
| weighted avg | 0.99 | 0.99 | 0.99 | 598 |

Table 3.13: Classification report of Xception model using third approach

As a summary, table 3.14 demonstrates the loss and accuracy for each implemented model:

| | 1st approach | 2nd approach | 3rd approach |
|---|---|---|---|
| Alexnet | 0.88795 | | |
| VGG16 | 0.96153 | 0.98160 | 0.99498 |
| VGG19 | 0.96153 | 0.97826 | 0.99498 |
| ResNet50 | 0.95150 | 0.98494 | 0.97157 |
| Xception | 0.98662 | 0.90802 | 0.98996 |

Table 3.14: Accuracy for each implemented model

### 3.3.4 Results Comparison with Recent Works

Many research papers were published in the field of brain tumor classification. Unfortunately, we have experienced troubles in finding peviously published works that have used our dataset. However, we compared our results to other notebooks on the KAGGLE platform. The comparaison is demonstrated in Table 3.15 below:

| Model | Model | Accuracy |
|---|---|---|
| First notebook [45] | EfficientNetB4 | 98.6% |
| Second notebook [46] | 4-layers CNN | 89% |
| Third notebook [47] | Pretrained ResNet34 | 89% |
| Fourth notebook [**notebookfourth**] | 3-layers CNN | 97% |
| Fifth notebook [48] | EfficientNetB2 | 99.6% |
| Fine-tuned VGG16 | | 99.5% |
| Fine-tuned VGG19 | | 99.5% |
| Xception implemented from scratch | | 99.6% |

Table 3.15: Results Comparison with Recent Works

### 3.3.5 Conclusions

The main objective of this project is to make a comparative study between the three approaches and between the CNN models themselves.

Training deep learning models implemented from scratch performed well in terms of evaluation metrics. Additionally, it provides more freedom to exploit the layers and modify them in case needed. However, deep learning models trained on small-scale datasets may suffer from the inability to generalize on unseen data and from overfitting even when data augmentation and regularization techniques are used.

Transfer learning with frozen weights is a great tool to take knowledge of a previously trained model (on a large-scale dataset) and apply it to a new problem. This second approach gave better results in general. However, deep and complex models may degrade in terms of accuracy because of the inability to adjust weights.

Fine-tuning gave the best results in terms of evaluation metrics with all the implemented CNN models. Models pre-trained on large-scale datasets perform well on small ones when weights are unfrozen. Additionally, the use of models pre-trained on large-scale datasets eliminates the overfitting problems and helps generalize better. The most significant disadvantage of utilizing pre-trained models is that we are forced to employ the same architecture, which may be so limiting.

VGG16 and VGG19 models have shown remarkable performance in extracting and classifying

the input MRI scans into cancerous and noncancerous. However, their both loss curves start increasing after a number of epochs, which means they suffer from the vanishing gradient problem.

ResNet50 has shown satisfactory results in terms of accuracy. Additionally, it resolved the vanishing gradient problem by introducing residual blocks.

Xception model gave the best results by using the first approach, and promising results in the third approach. It also resolved the vanishing gradient problem by using residual blocks. Introducing the depth-wise separable convolutions made the model faster and less computationally expensive.

## 3.4 Summary

In this chapter, we described the performance metrics used to implement and evaluate deep learning models. Then, we separately plotted and discussed for each experiment the learning curves, confusion matrices, and classification reports. Next, we draw a summary table to resume all the models' findings. Additionally, we interpreted and discussed the results for the three approaches as well as the implemented models. Finally,

## 3.5 Conclusion and Future Work

Computer vision and deep learning have shown remarkable performance in the field of biomedical imaging due to their promising results in the detection and classification of medical anomalies, especially cancerous diseases. Brain tumors are growths of abnormal cells in the brain. They can affect anyone at any age. Brain cancer is the leading cause of death from cancer in children and young adults [49]. There are several ways to detect brain tumors. This project focused on MRI-based brain tumor classification, which uses radio waves to create detailed images of the brain.

The proposed work -Brain Tumor Classification using Deep Learning- aims to compare three approaches to implementing deep learning models: implementation from scratch, transfer learning, and fine-tuning. In addition, we implemented four CNN models: VGG16, VGG19, ResNet50, and Xception. Each one using the three approaches, namely: implementation from scratch, transfer learning, and fine-tuning.

In the last chapter, the performance metrics chosen for evaluation were presented allowing the discussion of classification reports and confusion matrices after the analysis of learning curves of both training and validation sets. As a summary, a table including the accuracies and losses for each experiment was illustrated. Fine-tuning proved to be the most effective approach for training models with small datasets. Xception model has shown the best performance in training a small dataset from scratch with 98.66% accuracy, while VGG16 resulted in the best accuracy among fine-tuned models with more than 99%.

In the light of these findings, we believe that developing robust and automatic brain tumor classification approaches is greatly beneficial to healthcare experts as it decreases the time and cost to detect and diagnose to detect anomalies diseases, which is a life saver.

Many improvements can be done in our work. We propose to explore more advanced deep learning algorithms, such as R-CNN, Fast R-CNN, Faster R-CNN, and Capsule Networks. Additionally, we can introduce different noise reduction techniques for more accurate results. Finally, we can exploit many other image classification tasks in the field of biomedical imaging.

# Bibliography


[1] *Brain tumor*. https://www.mayoclinic.org/diseases-conditions/brain-tumor/symptoms-causes/syc-20350084. may,2022.

[2] *Survey of magnetic resonance imaging availability in West Africa*. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6295297/. may,2022.

[3] *Brain Anatomy and How the Brain Works*. https://www.hopkinsmedicine.org/health/conditions-and-diseases/anatomy-of-the-brain. may,2022.

[4] *MIND BRAIN 100 Trillion Connections: New Efforts Probe and Map the Brain's Detailed Architecture*. https://www.scientificamerican.com/article/100-trillion-connections/. may,2022.

[5] *Brain Anatomy and How the Brain Works*. https://www.hopkinsmedicine.org/health/conditions-and-diseases/anatomy-of-the-brain. may,2022.

[6] *Understanding Brain Tumors*. https://www.healthline.com/health/brain-tumor. may,2022.

[7] *The Most Common Brain Tumor: 5 Things You Should Know*. https://www.hopkinsmedicine.org/health/wellness-and-prevention/the-most-common-brain-tumor-5-things-you-should-know. may,2022.

[8] *Brain Tumor: Statistics*. https://www.cancer.net/. may,2022.

[9] "12-algeria-fact-sheets". In: *International Agency for Research On Cancer* (March,2021), p. 2.

[10] *Brain tumours*. https://www.nhs.uk/conditions/brain-tumours/. may,2022.

[11] *Brain Tumors*. https://www.aans.org/en/Patients/Neurosurgical-Conditions-and-Treatments/Brain-Tumors. may,2022.

[12] Gopal S Tandel et al. "A Review on a Deep Learning Perspective in Brain Cancer Classification". In: *Cancers* 11 (2019).

[13] *NOVA-MRI: Novel Applications in 19F Magnetic Resonance Imaging*. https://nova-mri.eu/EU_Projects/NOVA.nsf/xStart_Basic.xsp. may,2022.

[14] Ali M. Hasan et al. "Segmentation of Brain Tumors in MRI Images Using Three-Dimensional Active Contour without Edge". In: *Symmetry* 8.11 (2016). ISSN: 2073-8994. DOI: 10.3390/sym8110132. URL: https://www.mdpi.com/2073-8994/8/11/132.

[15] *Artificial Neural Network, Its inspiration and the Working Mechanism.* https://www.analyticsvidhya.com/blog/2021/04/artificial-neural-network-its-inspiration-and-the-working-mechanism/. may,2022.

[16] Facundo Bre, Juan Gimenez, and Víctor Fachinotti. "Prediction of wind pressure coefficients on building surfaces using Artificial Neural Networks". In: *Energy and Buildings* 158 (Nov. 2017). DOI: 10.1016/j.enbuild.2017.11.045.

[17] *Introduction to Convolutional Neural Networks (CNN).* https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/. may,2022.

[18] *Convolutional Neural Network: An Overview.* https://www.analyticsvidhya.com/blog/2022/01/convolutional-neural-network-an-overview/. may,2022.

[19] *What Is A Convolutional Layer?* https://analyticsindiamag.com/what-is-a-convolutional-layer/. may,2022.

[20] Muhamad Yani, S Irawan, and Casi Setianingsih. "Application of Transfer Learning Using Convolutional Neural Network Method for Early Detection of Terry's Nail". In: *Journal of Physics: Conference Series* 1201 (May 2019), p. 012052. DOI: 10.1088/1742-6596/1201/1/012052.

[21] Leo Pauly et al. "Deeper Networks for Pavement Crack Detection". In: July 2017. DOI: 10.22260/ISARC2017/0066.

[22] *Machine Learning Activation Function in Neural Network.* https://hamdi-ghorbel78.medium.com/machine-learning-activation-function-in-neural-network-12caac615964. may,2022.

[23] *ARTIFICIAL NEURAL NETWORK (ANN) 4 - BACKPROPAGATION OF ERRORS.* https://www.bogotobogo.com/python/scikit-learn/Artificial-Neural-Network-ANN-4-Backpropagation.php. may,2022.

[24] *ML — Underfitting and Overfitting.* https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/. may,2022.

[25] *ML Fundamentals: Bias Variance Trade-off.* https://analyticsindiamag.com/what-is-a-convolutional-layer/. may,2022.

[26] Pedro Costa. *Do we need any feature extraction of image to train Deep neural network for image classifications?* Aug. 2017.

[27] *How Artificial Intelligence Revolutionized Computer Vision: A Brief History.* https://www.motionmetrics.com/how-artificial-intelligence-revolutionized-computer-vision-a-brief-history/. may,2022.

[28] Maria Nazir, Sadia Shakil, and Khurram Khurshid. "Role of deep learning in brain tumor detection and classification (2015 to 2020): A review". In: *Computerized Medical Imaging and Graphics* 91 (2021), p. 101940.

[29] T Kalaiselvi et al. "Deriving tumor detection models using convolutional neural networks from MRI of human brain scans". In: *International Journal of Information Technology* 12.2 (2020), pp. 403–408.

[30] Ahmet Çinar and Muhammed Yildirim. "Detection of tumors on brain MRI images using the hybrid convolutional neural network architecture". In: *Medical hypotheses* 139 (2020), p. 109684.

[31] Tariq Sadad et al. "Brain tumor detection and multi-classification using advanced deep learning techniques". In: *Microscopy Research and Technique* 84.6 (2021), pp. 1296–1308.

[32] Amjad Rehman Khan et al. "Brain tumor segmentation using K-means clustering and deep learning with synthetic data augmentation for classification". In: *Microscopy Research and Technique* 84.7 (2021), pp. 1389–1399.

[33] Vipin Makde et al. "Deep neural network based classification of tumourous and non-tumorous medical images". In: *International Conference on Information and Communication Technology for Intelligent Systems*. Springer. 2017, pp. 199–206.

[34] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012).

[35] Nicola Strisciuglio, Manuel Lopez Antequera, and Nicolai Petkov. "Enhanced robustness of convolutional networks with a push–pull inhibition layer". In: *Neural Computing and Applications* 32 (Dec. 2020), pp. 1–15. DOI: 10.1007/s00521-020-04751-8.

[36] A Zisserman and K Simonyan. "Very Deep Convolutional Neural Network for Large-Scale Image Recognition". In: ICLR, 2015.

[37] Max Ferguson et al. "Automatic localization of casting defects with convolutional neural networks". In: Dec. 2017, pp. 1726–1735. DOI: 10.1109/BigData.2017.8258115.

[38] Yufeng Zheng, Clifford Yang, and Aleksey Merkulov. "Breast cancer screening using convolutional neural network and follow-up digital mammography". In: May 2018, p. 4. DOI: 10.1117/12.2304564.

[39] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[40] Qingge Ji et al. "Optimized Deep Convolutional Neural Networks for Identification of Macular Diseases from Optical Coherence Tomography Images". In: *Algorithms* 12 (Feb. 2019), p. 51. DOI: 10.3390/a12030051.

[41] François Chollet. "Xception: Deep learning with depthwise separable convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1251–1258.

[42] Erik Westphal and Hermann Seitz. "A Machine Learning Method for Defect Detection and Visualization in Selective Laser Sintering based on Convolutional Neural Networks". In: *Additive Manufacturing* 41 (Mar. 2021), p. 101965. DOI: 10.1016/j.addma.2021.101965.

[43] *Python (programming language)*. https://en.wikipedia.org/wiki/Python_(programming_language). may,2022.

[44] Arda Aras. *Explaining what learned models predict: In which cases can we trust machine learning models and when is caution required ?* Mar. 2020.

[45] *mustafaelhendy/finaal*. `https://www.kaggle.com/code/mustafaelhendy/finaal`. may,2022.

[46] *prakharpipersania/brain-mri-new*. `https://www.kaggle.com/code/prakharpipersania/brain-mri-new`. may,2022.

[47] *narayanamangilipally/brain-tumor-classification*. `https://www.kaggle.com/code/narayanamangilipally/brain-tumor-classification`. may,2022.

[48] *salikhussaini49/brain-tumor-prediction-97*. `https://www.kaggle.com/code/salikhussaini49/brain-tumor-prediction-97`. may,2022.

[49] *Cancer in Children and Adolescents*. `https://www.cancer.gov/types/childhood-cancers/child-adolescent-cancers-fact-sheet`. may,2022.

People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

University M'Hamed
BOUGARA - Boumerdes

Institute of Electrical and
Electronic Engineering

# Authorization for Final Year Project Defense

Academic year: 2021/2022

The undersigned supervisor: **Elhocine BOUTELLAA**
authorizes the student(s):

**Ryad BERRICHI** Option **Computer engineering**

to defend his /~~her~~ /~~their~~ final year Master program project entitled:

## Brain Tumor Classification Using Deep Learning

during the ☒ July ☐ ~~September session~~.

Date: **22/ 06 / 2022**

The Supervisor

The Department Head