University of Sunderland

Faculty of Technology
Department of Computer Science

PROM02 – MSc Dissertation

MSc Data Science

Student Name: Altoglou Athanasios
Supervisor Name: Andrew Smith
Second Marker Name: Ashley Williamson

Brain Tumor Binary Classification from MRI images using Convolutional Neural Network and VGG16 with Transfer Learning

September 2022

Declaration

I declare the following:

(1) that the material contained in this dissertation is the end result of my own work and that due acknowledgement has been given in the bibliography and references to **ALL** sources be they printed, electronic or personal.

(2) the Word Count of this Dissertation is 13,000 words

(3) that unless this dissertation has been confirmed as confidential, I agree to an entire electronic copy or sections of the dissertation to being placed on the eLearning Portal, if deemed appropriate, to allow future students the opportunity to see examples of past dissertations. I understand that if displayed on the eLearning Portal it would be made available for no longer than five years and that students would be able to print off copies or download.

(4) I agree to my dissertation being submitted to a plagiarism detection service, where it will be stored in a database and compared against work submitted from this or any other Department or from other institutions using the service.

In the event of the service detecting a high degree of similarity between content within the service this will be reported back to my supervisor and second marker, who may decide to undertake further investigation that may ultimately lead to disciplinary actions, should instances of plagiarism be detected.

(5) I have read the University of Sunderland Policy Statement on Ethics in Research and I confirm that ethical issues have been considered, evaluated and appropriately addressed in this research.

SIGNED: ALTOGLOU ATHANASIOS

DATE: 04-September-2022

Abstract

Brain tumor is the abnormal mass in the cells form within the brain. Precise and accurate brain tumor detection in MRI images plays crucial role in clinical diagnosis and decision making for the treatment of the patients. Applying classical machine learning methods is a very challenging task for a non-expert because it is very time-consuming and prone to errors process. Recent advancement in deep learning has shown huge progress in healthcare, and convolutional neural networks displayed substantial results in image classification tasks. The purpose of this study is to apply deep learning techniques to classify whether a patient has brain tumor or not using two approaches. A custom CNN model is designed from the scratch and improved through experimentation as well as, a pre-trained model named VGG16 is used utilizing the art of transfer learning. To evaluate the performance of the proposed models, deep learning curves are interpreted, and standard evaluation metrics are measured. The final model created from the scratch can detect brain tumors from MRI images with 98% accuracy whereas the pre-trained model with 91% accuracy. The best model is compared with other studies which use popular state-of-the-art CNN and pre-trained models. Finally, a web application is created as a final data product which can be used by physicians or clinical experts to validate their initial screening and make better decision in brain tumor diagnosis.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

Cancer is one of the most life-threating diseases among children and adults, as it is the second leading cause of death worldwide, according to World Health Organization (WHO, 2022). The House of Commons in UK states that in 2021, 137,234 patients died from cancer in England while this number has increased by 8% since 2001 (Baker Curl, 2021). Although the early cancer diagnosis can be proved lifesaving, this is not always feasible. However, unlike cancer, tumors can be benign or malignant. Benign tumors vary from malignant in that benign can be considered as 'healthy' tumors which do not spread to other parts of the body and can be removed with surgery (Badža and Barjaktarović, 2020).

The human brain is a complex organ which is composed of around 100-billion nerve cells. It controls the whole nervous systems and therefore, any form of abnormality in the brain can put human life in danger (Kang et al., 2021). Tumors in the brain are the most serious ones among other abnormalities. A brain tumor is an unnatural and an abnormal growth of the cells in the brain that can be classified into two groups: benign (healthy) and malignant (cancerous) tumors. Benign brain tumors grow slowly, and they rarely attack neighbouring healthy cells while malignant brain tumors readily invade other brain cells as they progress in rapid rates and they are difficult to treat (Tandel *et al.*, 2019). Pituitary, gliomas and meningiomas are some of the primary brain tumors. According to the National Health System (NHS), brain tumors can be classified in four grades. Grade 1 and 2 brain tumors (healthy) include such pituitary and meningiomas tumors while grades 3 and 4 are malignant (cancerous) tumors such as gliomas (NHS, 2020).

Brain tumor is aggressive. In 2020, an estimate of 308,102 people were diagnosed with brain tumor worldwide while the 5-year survival rate is less than 40% for both men and women (CancerNet, 2022). Early detection of brain tumor is vital for patients which is often made by magnetic resonance imaging (MRI). Different types of medical imaging technologies like positron emission tomography (PET) and computed tomography (CT) exist in the medical field which are utilized to detect the internal parts of a human body. However, not only MRI is a non-invasive diagnostic technique but also provides beneficial information about the size, type, shape, and location of brain tumors in 2D and 3D formats (Kang et al., 2021). Figure 1 shows three different planes (coronal, axial, sagittal) of MRI

which provide more accurate and precise information about the characteristics of the brain tumors (Sriramakrishnan et al., 2019).

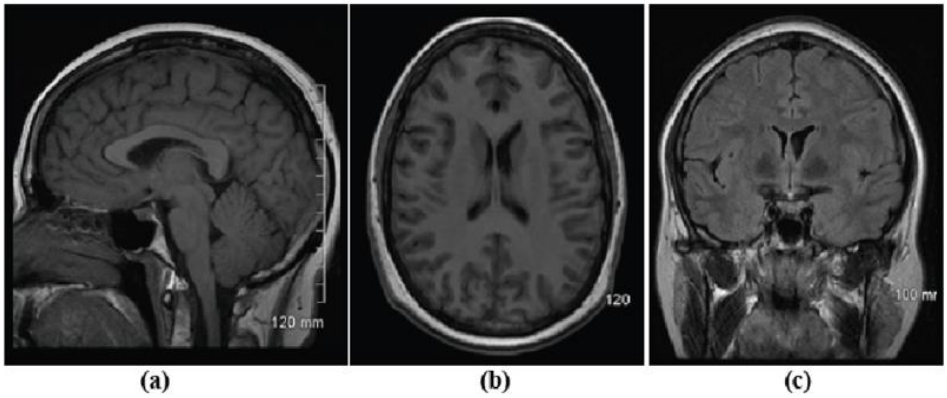

Figure 1. Brain MRI images in three different planes: a) coronal plane, b) axial plane c) sagittal plane

Appropriate planning and treatment as well as accurate diagnostic of brain tumors should be implemented to expand and improve the life of the patients. However, manual classification and examination of brain MRI images with similar appearances and structures can be susceptible to errors which depends on the experience and expertise of the radiologist or clinical doctor. On the other hand, traditional machine learning algorithms rely on several pre-processing steps, feature extraction and selection which is a demanding and time-consuming task for a non-expert to utilize them in their research (Swati et al., 2019). Deep learning and computer vision applications have seen huge advancements in the field of medicine and can be used effectively to diagnose tumor in brain MRI images. Deep learning models are different from the traditional machine learning methods because they can learn and extract automatically the most important patterns from a brain image, minimizing both the time and error of interpretation.

1.2 Aims and Objectives

1.2.1 Aims

The aim of this project is to accurately detect and classify brain tumor from MRI images using Convolutional Neural Networks and a pre-trained model named VGG16 with transfer learning. Also, a web application is created as a final data product which predicts whether a brain MRI image is affected by tumor or not. The app is expected to be installed

locally by the end-users and it can be used as a diagnostic tool by clinical doctors to make more accurate and faster decisions.

1.2.2 Objectives

At the end of the project, there are several topics which will be clarified and objectives to be met through the literature review and the practical work. These are as follows;

- The impact of Computer Vision and Deep Learning applications on the healthcare, especially in accurately detecting tumor in the brain.
- The main idea behind how convolutional neural networks work to 'understand' a brain image well-enough to determine whether a patient has brain tumor.
- Extract and load the dataset found on Kaggle and perform pre-processing steps to prepare the data for training.
- Build and train the deep learning models to accurately detect tumor in brain MRI images using TensorFlow framework and keras library.
- How to prevent the challenge of overfitting during the training phase of the models.
- Evaluate and compare the performance of the final Convolution Neural Network and the VGG16 model using various evaluation metrics such as the learning curves, accuracy, and confusion matrices choosing the best model to deploy in the web application.
- Develop the front-end of the website using HTML, CSS, and Bootstrap as well as the back-end using Flask framework and Python programming language.

1.3 Research Approach

The first research approach that will be used in this project for brain tumor prediction is to design and train a Convolutional Neural Network from the scratch, starting from a baseline model and try to improve it through experimentation. Secondly, a pre-trained model called VGG16 will be used, applying the technique of transfer learning.

1.4 Ethical, Social, Professional, Legal and Security Considerations

Artificial Intelligence and the development of autonomous algorithmic systems have the potential to improve the healthcare domain, improving the effectiveness of the medical treatments and addressing some of the most crucial tasks. However, with the rapid growth of these system in this field comes many more challenges (Gerke et al., 2020). Medical data are sensitive and contain personal information of the patients that should not be shared in any case. Therefore, it is vital that deep learning applications are implemented in a way that is legal, ethical, and secure. Legal and ethical issues raised by the misuse of these systems can have a negative impact on the health and the well-being of the individual as well as on the society as a whole.

The rest of this project is structured as follows: Chapter 2 presents a thorough literature review of related studies. Chapter 3 introduces the main idea behind Convolutional Neural Networks and the methodology followed to achieve the aim of this study. The experimental results, the performance evaluation of the proposed models, and a comparison of the best model with other methods are discussed in chapter 5. Chapter 6 provides an overview of the final data product along with an evaluation of the issues raised by automated systems in the medical field. Moreover, the project is evaluated, and a personal reflection is presented. Chapter 6 is the last section of this project and includes the conclusion and future works.

2 Literature Review

Brain tumor classification has already been studied by researchers who utilized traditional machine learning methods throughout the years. However, computer vision based on deep learning techniques has seen huge development in the medical field and has made a considerable impact on image analysis and classification (Mehmood et al., 2020). The first part of the literature review examines studies where researchers used machine learning algorithms for brain tumor classification and segmentation while the second part focuses on deep learning techniques for brain tumor detection and multi classification.

2.1 Traditional Machine Learning Approach

It is widely known that classification tasks using traditional machine learning techniques consists of many and important pre-processing steps, dimension reduction, principal component analysis and feature extraction and selection. Feature extraction is a crucial step in ML methods and evaluation metrics like accuracy, rely on extracted features. Feature extraction can be divided into two types: low-level (global) feature extraction such as first-order statistics (e.g., standard deviation, skewness and mean) and second-order statistics stemmed from wavelet transform, Gray Level Co-occurrence Matrix (GLMC) and Gabor (Swati et al., 2019).

Numerous studies have been conducted on brain tumor detection and classification using Wavelet Transform as the main feature extraction method which is combined with classic machine learning models. For example, Mathew and Anto (2017) performed various pre-processing steps such as skull and noise removal from the input brain image and applied Ostu's thresholding and morphological operations to detect the boundary of the tumor. Then, they utilized Discrete Wavelet Transform (DWT) decomposition to extract important features from the pre-processed image. The final features were used as inputs to the Support Vector Machine classifier which achieved 86% accuracy detecting normal or benign or malignant brain images. Arunachalam and Savarimuthu (2017) proposed a fully automatic system for brain tumor detection and segmentation. A publicly available dataset is used where the brain images were enhanced using the shift-invariant shearlet transform (SIST). The Nonsubsampled Contourlet transform (NCST) method was applied on these images to get the multi resolution images and then important features were extracted using discrete wavelet transform (DWT), grey level co-occurrence matrix (GLCM), and Gabor. The results

were used as inputs to feed the forward back propagation neural network in order to detect normal or abnormal brain images which achieved 99.8% classification accuracy. Also, the tumor region was segmented using k-means clustering algorithm.

In contrast, in their comprehensive image analysis of brain tumor detection, Bahadure et al. (2017) performed a combination of Berkeley Wavelet Transformation (BWT) to effectively segment brain MR images and Support Vector Machine (SVM) to classify normal and abnormal brain tissues based on the relevant features extracted from the segmented area of the tumor. More specifically, they pre-processed the input brain image to enhance the signal-to-noise ratio and applied a skull stripping algorithm to remove the skull. Feature extraction increased the performance of the proposed technique which achieved 96.51% overall accuracy and outperformed Back Propagation and K-Nearest Neighbours (K-NN) classifier, demonstrating the effectiveness of this method.

Another compelling research was conducted by Parveen and Singh (2015). In their study, authors introduced a hybrid technique for brain tumor detection and prediction combining fuzzy c-means clustering and Support Vector Machine. The quality of input brain images was improved using contrast improvement and mid-range stretch as well as, skull stripping algorithm was applied to remove the skull. Images were segmented with fuzzy c-means clustering algorithm and important features were extracted using Gray Level Run Length Matrix (GLRLM). As a classification tool for tumor detection, authors performed SVM trying three kernel functions: linear, quadratic, and polynomial. Results shown that linear kernel performed the best, with 91.66% accuracy and 83.33% sensitivity. Shree and Kumar (2018) aimed to classify brain MRI images into two classes: normal and abnormal. Gray-level co-occurrence matrix (GLCM) was applied to extract statistical features and these features were used as inputs to feed the Probabilistic Neural Network (PNN) which composed of an input layer, hidden layer, pattern, and output layer. The proposed method obtained nearly 100% on the training data and 95% on the testing data.

Even though low-level features can effectively depict and describe the image, these features have limited power of representation. This is because that different brain tumors have similar shape, appearance, size, and texture. As shown in the *Figure 2*, pituitary tumors may have similar appearance and shape like meningiomas tumors.

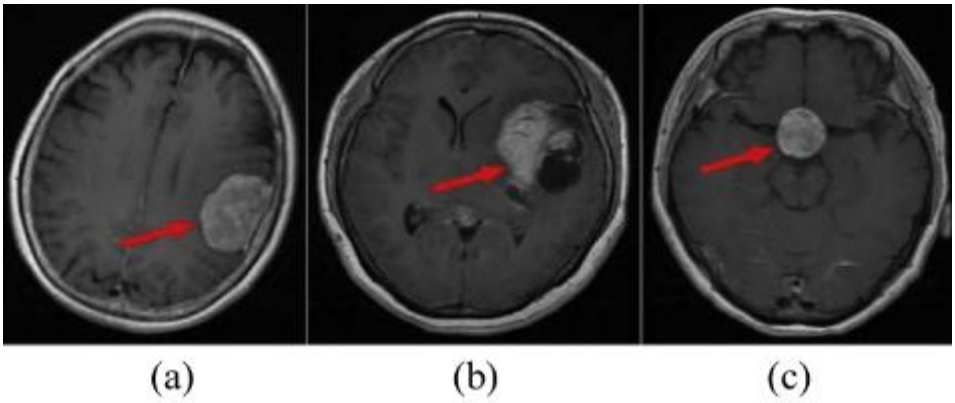

Figure 2. Tumors (a) and (b) are meningiomas with different shape, and (c) is a pituitary brain tumor with similar appearance to that as (a)

High-level (local) feature is the second type of feature extraction, and it includes methods like Bag-of-Words (BOW) and Fisher Vector (FV). Cheng et al. (2016) introduced and developed a novel content-based image retrieval system (CBIR) for retrieving three types of brain tumor in T1-weighted contrast-enhanced MRI images. Their research consists of three steps. The tumor region was augmented and then used to gather and integrate vital contextual information. Moreover, authors performed spatial division method to split the augmented tumor region into subregions while in each subregion, local features were extracted. Finally, a powerful fisher kernel was used to represent the local features of the subregions as vectors. Fisher Vector compared and achieved better performance than the bag-of-words representation using different vocabulary size.

In their comprehensive study of automatic brain detection, Cheng et al. (2015) proposed and tested three types of feature extraction: intensity histogram, grey level co-occurrence matrix (GLCM), and BOW features. Using image dilation, they augmented the tumor region which is selected as the region of interest (ROI), partitioning the augmented tumor area into ring format subregions. Results shown the effectiveness of the ROI which enhanced the accuracies in all three types of feature extraction and ring-shape partition gave higher accuracies in advance.

However, these studies carry with them some limitations in the feature extraction phase of classic machine learning approach. This phase concentrates on either high or low-level feature extraction and it essentially depends on handcrafted features. For instance, Cheng et al. (2015) research relies on handcrafted feature extraction such as the partition of

the tumor region in a ring-form shape which is a demanding task as it requires sufficient prior knowledge and information. Therefore, according to the literature review, it is not a simple task to utilize traditional machine learning methods, especially for a non-expert, as it is susceptible to human errors.

2.2 Deep Learning Approach

In recent years, Computer Vision (CV) and Deep Learning have drawn a lot of attention in the field of healthcare. Deep Learning is a subfield of Artificial Intelligence (AI) and it is defined by deep stacks of computation. This depth enables deep learning models such as neural networks to find complex and hidden patterns in challenging datasets. Computer Vision is also a subfield of AI which allows computers the ability to extract high-level understanding from visual inputs such as videos and digital images. Therefore, a deep neural network can be trained and learn how to 'understand' a brain MRI image well-enough to solve similar problems like the human visual system is able to solve. Convolutional Neural Networks, also known as convnets or CNNs, are the best and most effective neural networks in image classification and other computer vision tasks. CNN models have been used in many studies to detect and classify brain tumor. Some researchers have built and designed their own CNN models from the scratch while some others have implemented the art of transfer learning.

Baranwal et al. (2020) performed a multi-classification task to classify three types of brain tumor using Support Vector Machine and their own CNN model. They built and trained a CNN model with 5 convolutional layers, a dropout layer and a flatten layer with 1024 neurons which had a better overall performance than the linear and polynomial SVM classifier. Confusion matrices and Receiver Operating Characteristic (ROC) curve, a graphical representation where True Positive Rate (TPR) and False Positive Rate (FPR) are plotted against, was used to evaluate the models. The area under the curve (AUC) is given by the ROC which measures the performance of the models as a final evaluation method. The ROC curve of the CNN model had larger area under the curve compared to SVM in all three predicted classes which indicates the effectiveness of the proposed model.

In another study, Ucuzal et al. (2019) presented and developed a free web-based application as a final data product which can classify meningioma, pituitary, and glioma types of brain tumor. The MRI images were pre-processed with various techniques such as image rotation, rescaling, length, and width change etc., and the CNN model was built using

TensorFlow framework and keras library. Their study was very successful based on their experimental results and evaluation metrics, achieving 99% accuracy on the training data. Flask library was used to develop the software which can be employed by clinical experts and professionals as a support tool to diagnose brain tumor more accurately and faster. In a similar study, Ankireddy (2020) created a web software app where users can upload a brain MRI image and the app will be able to make a reliable diagnosis and segmentation for each patient. This time, Mask R Convolutional Neural Network was used to segment and detect the brain tumor which was trained for 20 epochs. Authors state that the use of computer vision techniques can decrease the human error in brain tumor detection while it can increase the rate of accuracy in predictions.

In the first part of their research, Singh et al. (2021) proposed and built a CNN model which can classify whether a brain MRI image is affected by tumor or not. Various pre-processing steps have been performed such as cropping the brain image as well as increasing the quality and size of it using a technique called data augmentation. The CNN model achieved 90% accuracy on the training data and 86% on testing data. However, the loss and validation curves of the model show that the proposed model overfits the training data. The training loss is starting to increase at the end of the training and there is a large gap between the loss curves which is also a sign of overfitting, as shown in Figure 3. Paul et al. (2017) designed a Convolutional Neural Network from the scratch which is composed of two convolutional layers and two flatten layers with 800 neurons each. The aim of their study was to accurately classify three types of brain tumor (glioma, meningioma, pituitary) using T1-weighted contrast enhanced MRI images. Applying five-fold cross validation, the CNN model performed slightly better that the previous study with 91.43% overall accuracy. However, judging from the loss curves again, the proposed model suffered from overfitting because of the small samples of the dataset.

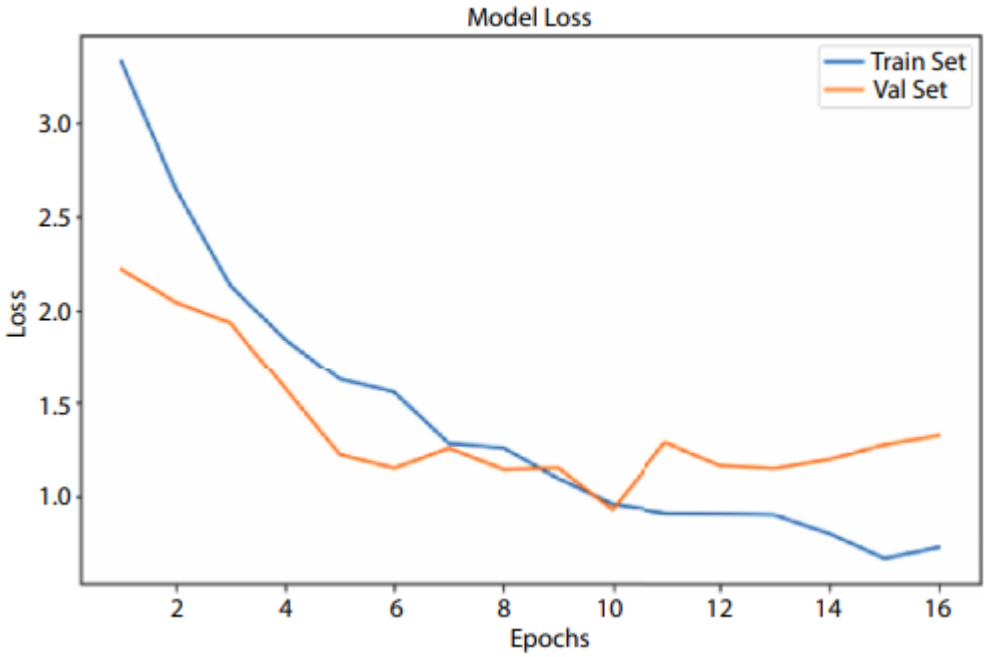

Figure 3. The proposed CNN model overfits the training data.

In their impressive research, Badža and Barjaktarović (2020) presented a novel CNN architecture for brain tumor multi-classification. The architecture of the proposed CNN model can be seen in Figure 4 and it consists of 22 layers which is simpler than a pre-trained model, meaning that it can be run in state-of-the-art personal computers, or it can be deployed on mobile platforms. The performance of the model was tested and evaluated applying four approaches: a combination of record-wise and subject-wise 10-fold cross-validation method and two databases. Using a record-wise 10-fold cross-validation in the augmented database gave the best results with 96.56% accuracy. Finally, results also show that the CNN model had a solid execution speed, and it can be used as supportive diagnostic tool from radiologists in medical field.

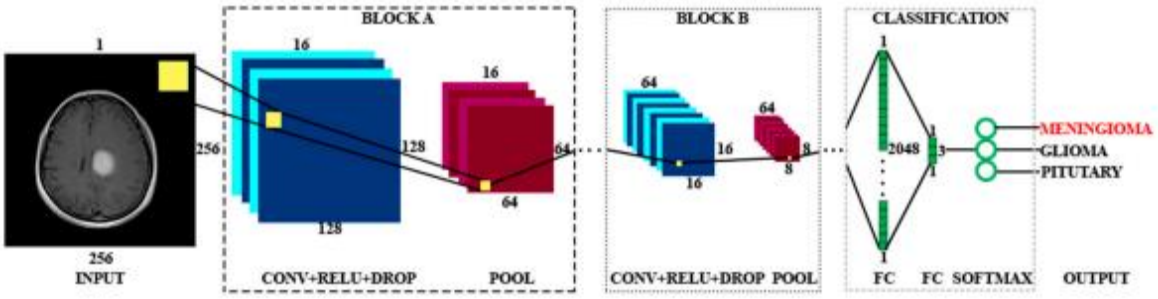

Figure 4. A novel architecture of a CNN model proposed by Badža and Barjaktarović

Irmak (2021) used a Convolutional Neural Network model to perform three classification tasks for the early brain tumor diagnosis. The first one was a binary classification problem where the CNN model can predict whether a brain MRI image is normal or abnormal. The CNN architecture can be seen in Figure 5 while it achieved 99.33% accuracy in brain tumor detection. The second study aimed to classify five types of tumors as normal, meningioma, glioma, metastatic, and pituitary with 92.66% accuracy. Finally, in classification task 3, the proposed CNN model can classify the glioma tumor into Grade 2, Grade 3, and Grade 5 stage. Grid search optimizer was used to tune all the hyperparameters. Experimental results and evaluation metrics such as accuracy, sensitivity, ROC and learning curves showed that the study was very successful because the author manage to prevent the challenge of overfitting in all three tasks. For example, it can be observed that in the Figure 6 the training loss curve is stabilized at the end of the training while there is no gap between them which demonstrates the effectiveness of this method.

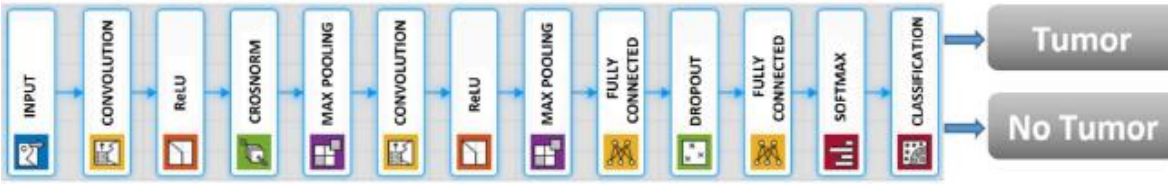

Figure 5. Architecture of the proposed model for the binary classification task.

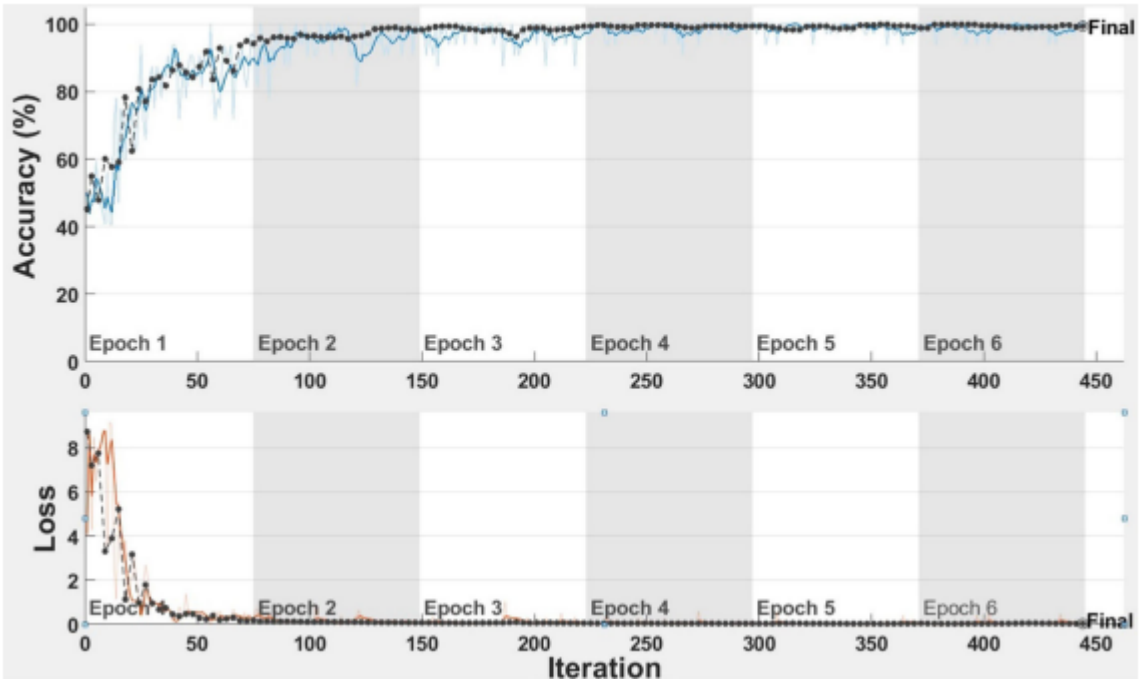

Figure 6. Loss and accuracy curves for the binary classification task of Irmak's study.

However, many studies still suffer from the problem of overfitting because of the size of the datasets or the way the models are trained. In order to achieve good performance in computer vision and image classification tasks, Convolutional Neural Networks should be trained on large datasets. Generally speaking, the size of the MRI images is usually small in the medical field, and this makes the deep CNN training very challenging and demanding. On the other hand, overfitting can be prevented and corrected by increasing the capacity of the model. The size of the network can be increased by making it wider (add more neurons to existed layers) while its complexity can be changed by making it deeper (add more layers). Training a deep network from the scratch is time consuming and demands a lot of patience.

Therefore, many researchers have used pre-trained models based on transfer learning as a substitute for training their own CNN models. Arora and Sharma (2021) performed exploratory data analysis (EDA) based on feature extraction and proposed three pre-trained models using transfer learning to detect brain tumor. VGG16, Inception V3, and ResNet were trained and evaluated in terms of training and validation accuracy. However, the main weakness of the study is the failure to address the problem of overfitting. In a glance, it can be observed that the models learn so much noise as well as a large gap is created between the training and validation loss which demonstrates a sign of overfitting. In another study, Çinar & Yildirim (2020) made use of a modified version of a pre-trained model called ResNet by removing the last five layers and adding eight new layers for brain tumor diagnosis. The improved CNN model produced successful results, achieving 97.2% accuracy while judging from the learning curves, authors overcame the challenge of overfitting. Also, the developed hybrid ResNet model outperformed other pre-trained models that were trained in this study, indicating the enhancement that is made in the classical architecture of the model.

Arbane et al. (2021) proposed three pre-trained CNN models based on transfer learning knows as Xception, MobileNet-V2, and ResNet-50 to detect and classify brain tumor from MRI images. In terms of accuracy and based on the learning curves, ResNet-50 had the poorest performance as the model overfitted the training data. MobileNet-V2's and Xception's learning curves showed the training and testing phase was successful while the former performed slightly better with 98.24% overall accuracy. In contrast, Swati et al.

(2019) suggested a different strategy for brain tumor classification. Instead of using a pre-trained model like other authors did, they proposed a block-wise VGG19 architecture based on transfer learning and fine tuning. With minimal pre-processing, an overall accuracy of 94.82% was obtained using five-fold cross-validation.

Finally, Deepak and Ameer (2019) designed a fully automatic system and applied GoogLeNet, a deep pre-trained neural network, to classify three types of brain tumor with minimal pre-processing using 3064 MRI images from 233 patients. ROC curve, accuracy, and other metrics were used to evaluate the proposed CNN model which performed very well with 98% accuracy. The experimental results demonstrate that transfer learning is a useful and effective technique especially when the sample of the images is small in a healthcare scenario.

Undoubtedly, based on the published studies, it can be observed that computer vision and deep learning techniques have become extremely popular and highly recommended for the clinical experts in medical image analysis. Not only do they reduce the semantic gap between the human error and the magnetic resonance imaging machine, but they also decrease the time of interpretation of brain MRI images. Moreover, it can be noticed that deep learning models obtained significantly better accuracies in brain tumor analysis compared the manual detection with traditional machine learning methods as they are faster and more accurate. Besides that, deep learning requires minimal image pre-processing. However, some studies have certain limitations which should definitely be considered in brain tumor segmentation and classification. A major drawback is that they did not deal with the problem of overfitting which was caused by either the limited amount of data or the way that CNN models were trained.

3 Materials and Practical Research Methodology

The purpose of this study is to accurately detect and classify brain tumor from MRI images using two research approaches. The first one is to create and train a CNN model from the scratch and improve its performance through experimentation while the second approach is to use a pre-trained model called VGG16 using the technique of transfer learning. In this section, the dataset used is discussed as well as the motivation and fundamental ideas behind how CNNs work in an image classification task is examined. A dataset is created, and the enhanced models are introduced along with the pre-trained model used with transfer learning. Finally, metrics used to measure the performance of the models are briefly mentioned.

3.1 Dataset

A publicly available dataset is used in this research which is found in Kaggle (Hamada, 2020). This dataset consists of three folders. A folder called ‘no’ which contains 1500 brain MRI images that are not affected by tumor, a ‘yes’ folder with 1500 MRI images which are tumorous, and a ‘pred’ folder with 60 brain MRI images to make final predictions.

3.2 Convolution Neural Networks

Convolutional Neural Networks, also known as convnets or CNN, are the most commonly used neural networks in deep learning for image classification tasks. In mathematics, a convolution operation gives the convnet’s layers a distinctive structure which is very efficient at solving computer vision complicated problems. CNNs do three important things in order to make image classification practical: 1) Decrease the number of input neurons (nodes), 2) Allow small movements in where the pixels are in the image and 3) Take advantage of the correlations that exist in complex images.

A traditional CNN model used for image classification problems consists of two parts: a convolutional base which is used for feature extraction and a dense head which behaves as classifier and determines the class of the given image. A typical CNN architecture has five main layers: an input layer, convolutional, pooling, a fully connected

layer, and an output layer. Figure 7 shows a simple Convolution Neural Network architecture which is designed in a free diagram editor like all the diagrams in this study.

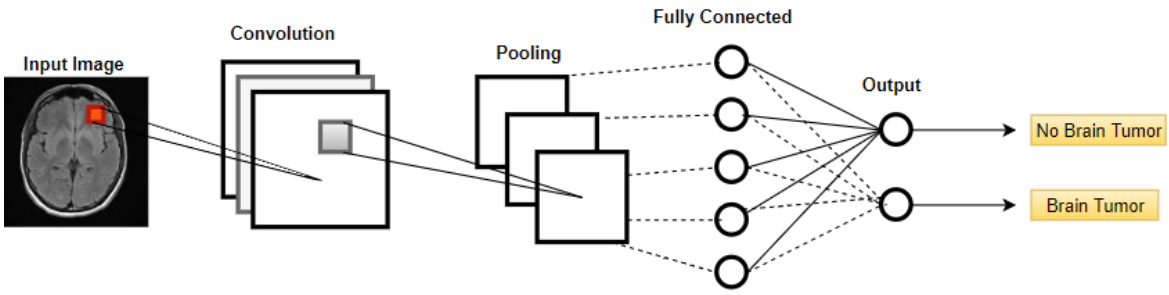

Figure 7. A simple CNN architecture.

The first thing a convnet does is apply a filter to the input image for a certain feature. A filter or kernel is represented as a small array that is usually 3 by 3 pixels while the intensity of each pixel in the kernel is decided by Backpropagation. In other words, pixel values are set randomly before training a CNN, and with Backpropagation these values ended up with more useful information. Therefore, the most important features are extracted and learnt by filtering the targeted image in a convolutional layer. Shifting the filter across the whole image, a feature map is created which is the result of the dot product between the input image and the filter (Brownlee, 2020). The feature map is used as an activation in the CNN and helps to find correlations there might be in the image. Typically, the feature map is run through a ReLU activation function which means that all positive values are the same as before while all the negative values are set to zero. Then, another filter is applied to the new feature map which select the maximum value and moves through the map in such a way that does not overlap itself known as max pooling. Max pooling layer decreases the dimensionality of the most important features and then the pooled layer is converted into a column of neurons which are used as inputs into a normal, everyday neural network. Finally, this neural network has a ReLU activation function and two output nodes, one for the images with brain tumor and one with normal brain images. Table 1 summarizes the components of a Convolutional Neural Network as well as their basic operations.

Layer type	Basic operation of each layer
Input layer	Takes in the input image and image shape is represented by a three-dimensional matrix for further layers.
Convolution layer	Extracts and learns the most important patterns from the input image.
Pooling layer	Reduce the dimension of the feature map which contains the learned image features.
Fully connected layer	This layer involves nodes, weights, and biases and further refines the patterns learned from the convnet layer.
Output layer	Display the predicted output in the shape of target classes.

Table 1. A summary of each layer used in a typical CNN architecture.

3.3 Creating the Dataset

To begin with, Python programming language is used to write code and build the models. Moreover, Google Colab, a free software tool which allows users to write python code through the browser, is selected because it provides access to Graphics Processing Units (GPUs) free of charge. Fast GPU is highly recommended because accelerates the computing time of the models' training, especially when the personal computer is outdated. The dataset is imported directly from Kaggle into the Google Colab using the command API of the dataset and is unzipped to create the dataset (see Table 3 for code in the Appendix A).

As it was mentioned before, there are 3000 brain MRI images which half of them are healthy and the remaining ones are tumorous. The shape of the image array is in the form of (width, height, Colour Channels). The height and width values are between 0 and 255 because this is the possible scale for red, green, and blue values while the colour channel value is always 3. In order to create and train the CNN models, a custom dataset is created which contains all these images and their associated label. However, there are some pre-processing steps that should be taken into account. OpenCV library is used to load the images from the specific file where the images are located and then using the PIL library, which provides the ability to manipulate and process the image data, the image is converted

to array. It should be noted that the MRI images captured from different location and therefore many of them have different size. Hence, the images are re-sized to 64 by 64 pixels, so all the images have the same size. Finally, they are converted into a numpay array and appended in the dataset with their associated label. The class with value equals to 0 goes for not tumorous images while the class with value equals to 1 goes for tumorous images since this is a binary classification task (see Table 4 for code in the Appendix A).

The dataset is split into training and testing subsets utilising the sklearn library. The first subset consists of 80% of the total images and is used to train the proposed models while the remaining 20% is used to evaluate their performance on images that have never seen before. Figure 8 shows four random images with their associated class from the training dataset, using the matplotlib library (see Table 5 for code in the Appendix A).

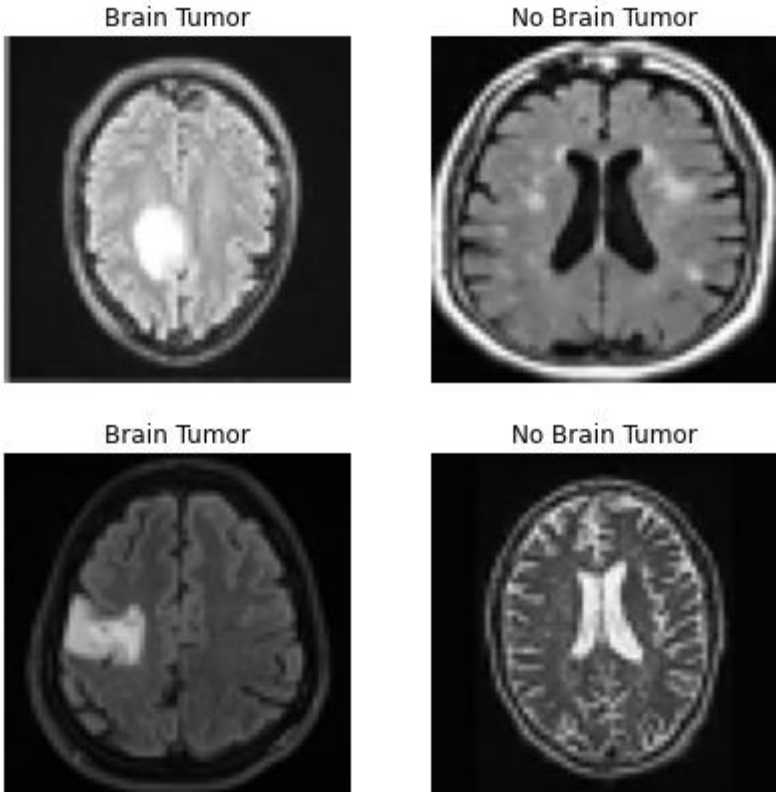

Figure 8. Plot of random brain images with their class names from the training dataset.

It's widely known that many machine learning models, including Neural Networks tend to perform better when the input values are between 0 and 1. Therefore, to scale or normalize the data is one of the most important and common pre-processing steps

when working with images. Scaling the pixel values can be achieved by dividing the image arrays by 255.

3.4 CNN Model

TensorFlow framework and keras library are used to build the CNN models. TensorFlow is an open-source software which allows users to build, train, and deploy machine learning models. Keras library is run on top of the TensorFlow which is a high-level wrapper for deep learning written in Python language. In this approach, different CNN architectures are examined in order to build the best model for brain tumor detection. It is always a good idea to start with a simple CNN model to have a baseline result and try to improve it through experimentation by increasing its complexity.

3.4.1 Create a baseline CNN model

The purpose is to create a CNN model that can generalize well to new unseen data but can also account for variance and patterns in known data. This is a definition of a good model and a good fit lives between an underfit and overfit model (Brownlee, 2019). Overfitting is a common and a very challenging phenomenon when training a neural network. An overfitted model is a model that learns and memorizes the training dataset very well and hence, the model loses its ability to generalize data that has never seen before.

The baseline CNN model has a typical architecture which consists of an input layer, a convolutional layer, a max pooling layer, a dense layer, and an output layer. The structure of the model it can be seen in Figure 7. The convolutional layer is 2D meaning that the inputs are 2 dimensional (width and height) while the number of filters is 32 and the kernel size is equal to 3. The rectifier function $(\max(0, x))$ or ReLU is used as an activation function and the max pooling layer has 2 by 2 pool size. To make the transition from a convolutional layer into a dense layer, a flatten layer is added to convert the multidimensional input into one-dimensional. The dense layer includes one node and the sigmoid is used as an activation function since the fully connected layer has one neuron as an output. In other case, if the output layer had two nodes the activation function should be the SoftMax.

After creating the model, the next step is to compile and the train the model. Three main parameters are used to compile the model: the loss function, the optimizer, and the evaluation metric. During training, the loss function will be used as a guide by the model to find the best values for its weights. Since this is a binary classification problem, the

'binary_crossentropy' loss function is applied. The optimizer is the Adam with all the defaults settings which minimize the loss by adjusting the weights while accuracy is the evaluation metric. Finally, the model is fitted and trained on the normalized training dataset for 10 epochs and 16 batches.

However, the learning curves show that this simple CNN model does not perform very well and overfits the training data. Therefore, the goal is to build another model and try to reduce overfitting. As it was mentioned in the literature review, there are many ways to prevent the challenge of overfitting. General speaking, if the neural network appears to overfit the data, one way to prevent it, is to the increase the capacity of the model. This can be achieved either by making it wider (add more units) or deeper (add more layers). Moreover, the number of the convolutional layers or the convolutional filters can also be increased to address the problem of overfitting.

3.4.2 Improved CNN Model

To enhance the base model, the complexity is increased by adding more convolutional layers and an another fully connected layer. Specifically, 2 more convolutional layers are used in the Convolutional Neural Network. The hyperparameters are exactly the same except in the last convolutional layer which has 64 filters instead of 32. Furthermore, a dense layer with 100 neurons and a ReLU activation function is added up before the output layer. The structure of this model can be seen in Figure 9. The training and validation loss curves show that the model has been improved a lot and the performance is enhanced after making the model more complex compared to the base model. However, even though the model is improved, it looks like that it may overfit the training data.

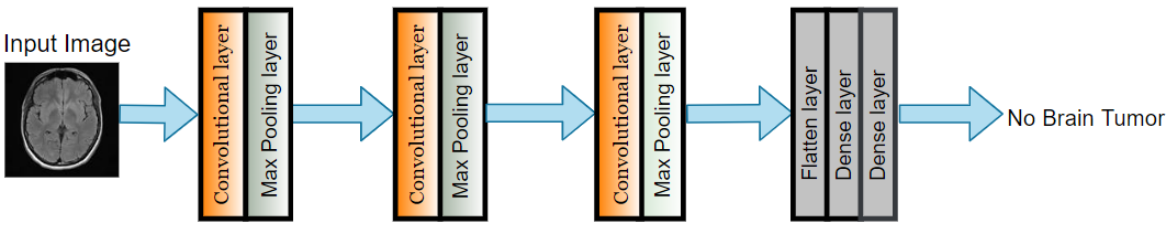

Figure 9. Structure of the enhanced model.

3.4.3 Final CNN model

In the deep learning world, there are a bunch of different layers apart from the dense layers. Some of them have weighted connections between them while some others can do some sort of transformations or pre-processing. One of those special layers is the dropout layer which can help correct overfitting by randomly dropping neurons and connections during model's training phase (Çinar and Yildirim, 2020). A typical dropout architecture can be seen in Figure 10.

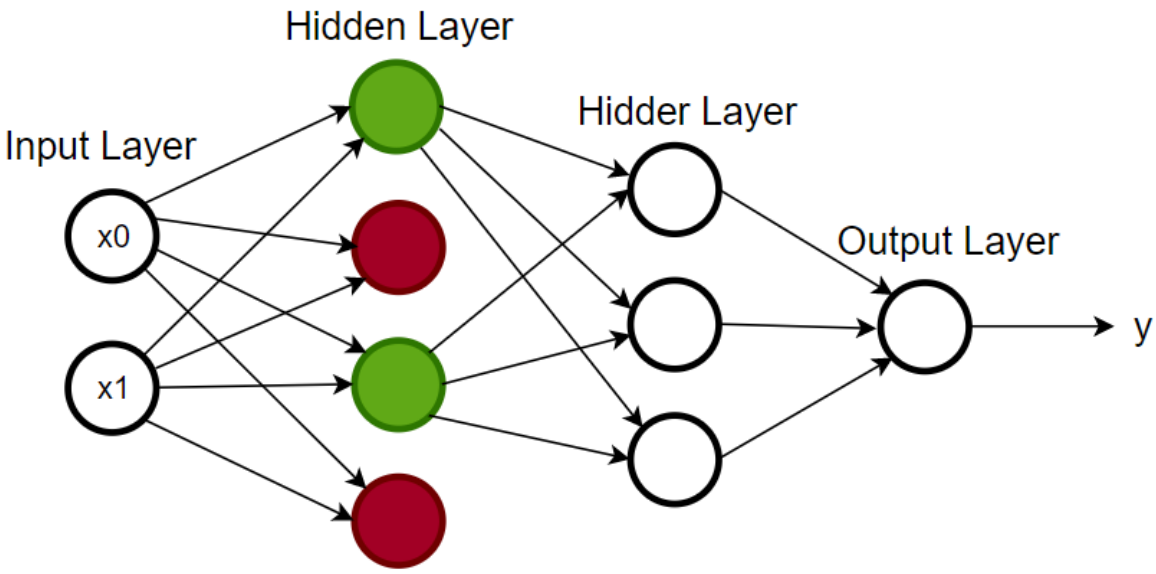

Figure 10. A 50% dropout layer in the first hidden layer.

With that being said, the final model has the same structure as the previous CNN model along with a dropout layer. The dense layer has now 64 neurons, followed by 50% dropout layer. Moreover, a kernel initializer parameter is added in two convolutional layers while the architecture of the proposed model can be seen in Figure 11. The model is also compiled with the same hyperparameters like the previous models and trained for 10 epochs with 16 batches as well. Judging from the learning curves which will be analysed in the next chapter, the final model does not overfit the training data, demonstrating the power of the dropout layer (see Table 6 for code in the Appendix A).

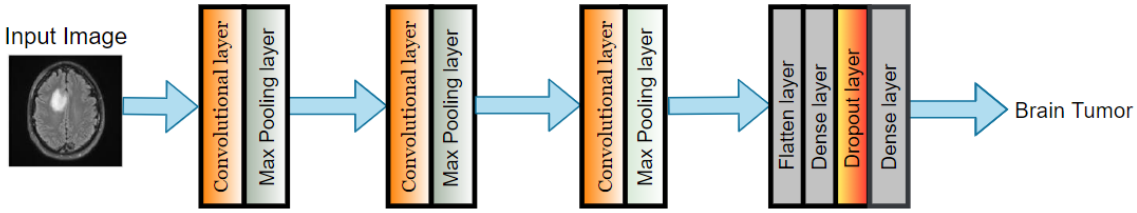

Figure 11. CNN architecture of the final model.

3.5 VGG16 with Transfer Learning

The second approach in this study is to apply a pre-trained deep learning model via transfer learning. Transfer learning technique became very popular in the field of deep learning, and it is used as a substitute of training CNN models from the scratch. In practice, people rarely train an entire Convolutional Neural Network from the scratch as it is very time consuming, trying to improve the model with different configurations such as by adding more special or hidden layers, adjusting the number of neurons in each layer, or changing the learning rate. Hence, many people utilize the transfer learning technique. When a Neural Network learns patterns (weights) in some sort of dataset, those patterns can be used for another task. Therefore, the general idea behind transfer learning is to take the learning patterns and knowledge of a deep learning model and use them for a different but related problem (Deepak and Ameer, 2019). Transfer of knowledge provides a lot of benefits in deep learning. First, it is broadly known that neural networks require a massive amount of data to perform in an accurate way, especially in the supervise learning where a lot of labelled data are needed. Since a model can use the patterns learnt from another model which is already trained, great results can be achieved using fewer custom data (Seldon, 2021). In healthcare for example, the size of the MRI samples is limited while transfer learning can be proved a useful and effective method in such a scenario. Moreover, deep learning models are designed to solve challenging and complex tasks, thus the training phase can be time consuming. Not only does transfer learning saves time and resources put into during training but also the whole training process is enhanced and becomes more efficient. Figure 12 shows the art of transfer learning.

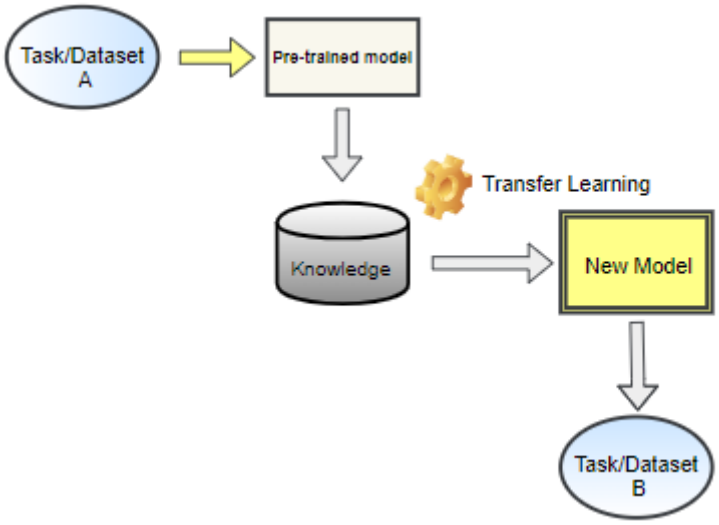

Figure 12. Basic operation of transfer learning.

TensorFlow and keras provide a bunch of pre-trained models each one with different architecture which are easily accessible. To detect and classify brain tumor in MRI images, a very deep pre-trained model named VGG16 is used with transfer learning. VGG16 is considered to be one of the most excellent and effective type of a Convolution Neural Network in image classification and object detection tasks as it was used to win an Image Large Scale Visual Recognition challenge in 2014. The architecture of a VGG16 model is displayed in Figure 13 and consists of thirteen convolutional layers, five max pooling layers, and three Dense layers (Thakur, 2019). Number 16 in VGG16 indicates the 16 learnable parameters which have weights. In summary, there are a total of 21 layers. The convolution layers have a kernel size 3 by 3, ‘same’ padding with stride 1. The stride parameter refers to the distance which the sliding window moves over the input image at each step. The max pooling layers have 2 by 2 pool size of stride 2 while these values remain the same throughout the whole architecture. VGG16 is a pretty large and very deep network with approximately 138 million parameters.

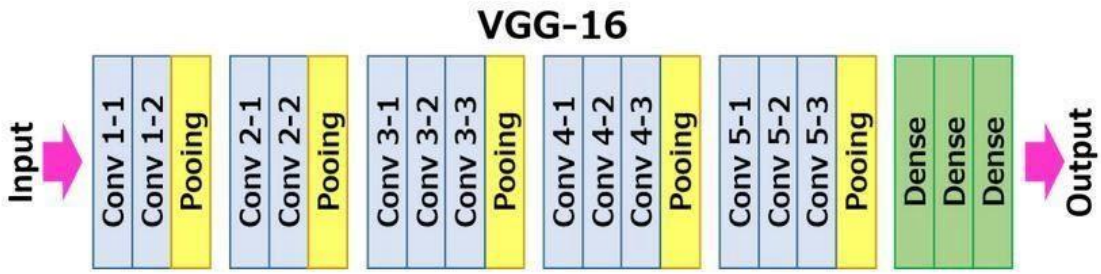

Figure 13. Typical structure of a VGG16 model.

The main idea behind the pre-trained model is to use it as a baseline and then add some more layers. To define the pre-trained model using transfer learning, the weights that have been learnt from a popular database called ImageNet, which consists of million of images of different objects, are used as the foundation of the model.

However, because of the fact that the VGG16 model has many learnable parameters, the final model takes a long time to train them, and it will not accurately predict the desired outcome. A common technique used in transfer learning is to freeze some or all of the pre-trained model's layers. In their paper, Brock et al. (2017) state that the training speed can be increased when the bottom layers are frozen proposing different methods of freezing them. Freezing the layers of the model means that the weights do not update during the training phase.

Therefore, all the learnt weights in the bottom layers of the VGG16 model are frozen so they are untrainable. In a computer vision model, the lower a layer is, the closer it is to the input layer. Additionally, the top layers of the pre-trained model are replaced by a Global Average Pooling layer, two fully connected layers with 1024 neurons, one more dense layer with 512 neurons and an output layer with one neuron. The output neuron is one because the sigmoid function is used as an activation (Figure 14). These layers are trained so the pre-trained model adjusts its weights to predict whether the given brain MRI image is affected by tumor or not.

The loss function, the optimizer, and the evaluation metric are used to compile the final model. Since this is a binary classification problem, the 'binary_crossentropy' loss function is applied. Adam is selected as the optimizer algorithm while accuracy is the evaluation metric. Finally, the model is also fitted and trained on the normalized training

dataset for 10 epochs and 16 batches like the previous CNN models, achieving a good overall performance (see Table 8 for code in Appendix A).

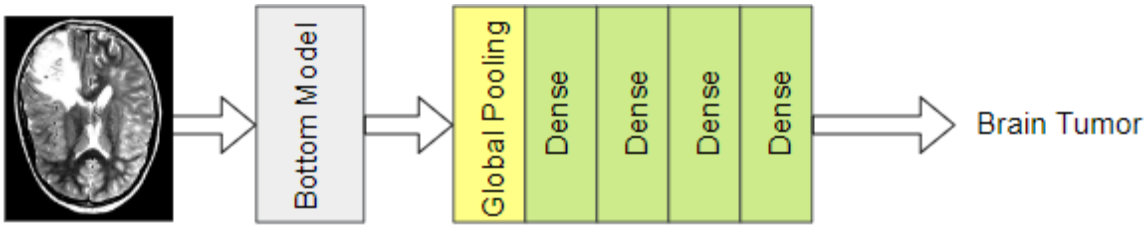

Figure 14 Structure of the VGG16 with added layers.

3.6 Performance Evaluation

It is essential that the performance of the deep learning models should be evaluated in image classification tasks to support the results achieved in the study. If not, the study would be academically and literarily weak, and remain incomplete. There are many metrics to measure the performance of the deep learning models in computer vision studies that have been utilized for a long time and have become standard evaluation metrics in various tasks. Some of them are accuracy, sensitivity or recall, specificity, precision, and f1-score. For a classification task, accuracy is the total number of correctly classified classes as a percentage of the whole. Sensitivity (recall) is the true positive rate which shows how good is the model at classifying normal brain images while, specificity describes the true negative rate which shows how good is the classifier at predicting tumor in the brain. Precision is the positive predictive rate, and f1-score describes the harmonic mean between of precision and recall. Eq. (1) illustrates the corresponding formula each of these evaluation metrics. FP, FN, TN, TP are false positive, false negative, true negative, and true positive, respectively.

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

$$Specificity = \frac{TN}{TN + FP}$$

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

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

$$F1\ score = 2^{*}\ \frac{Recall * Precision}{Recall + Precision}$$

(1)

Along with these metrics, plotting the learning curves of the proposed models is vital to evaluate their learning performance over time or experience. A learning curve shows how the training and validation loss change over time in each epoch during training. The training loss is the error of the deep learning model on the training dataset which demonstrates how good the model fits on them whereas the validation loss is a metric to evaluate its performance on the validation dataset. Learning curves are broadly used as a tool to diagnose potential errors with learning, such as an overfit or underfit model. Therefore, visualizing the training and validation loss in a graph is very important to evaluate the deep learning models. In the next chapter, the learning curves are reviewed as well as the results of the study are analysed in depth in order to find the best model and deploy it in the web application.

4 Analysis of Results

To begin with, in order to fairly evaluate the performance of the proposed models, they have been all trained with the same number of epochs and batches. Moreover, it is extremely important to review and analyse the learning curves as a starting point for the models' assessment to examine whether the problem of overfitting is prevented or not. Otherwise, high training and validation accuracies have no meaning if the model overfits the training data. Therefore, an overfitted model with high accuracy is not valid because it will perform poorly on unseen brain MRI images, and it cannot be used in a real-world application.

4.1 Results of the CNN models

The performance of the proposed models is evaluated interpreting their learning curves. The baseline model has only one convolution layer and it learns 31.649 parameters. The learnable parameters are the patterns that the model can learn from the data and intuitively, the more parameters a model has, the better is able to learn important features. However, this is not always the case. Sometimes, even though a convolution neural network learns less parameters, this can be proved more helpful to identify features in a brain MRI image. Figure 15 breakdowns the learning curves of the baseline model. Y axis shows the values of the training and validation loss over time while the x axis displays the number of epochs. It can be observed that the model is overfitting the training data for two main reasons. The validation loss starts to increase at the end of the training as well as the validation accuracy decreases which reveals a sign of overfitting. Hence, even though the

model has high accuracy, it does not perform very well because it is learning the weights in the training dataset very well and loses its ability to generalise on unseen data.

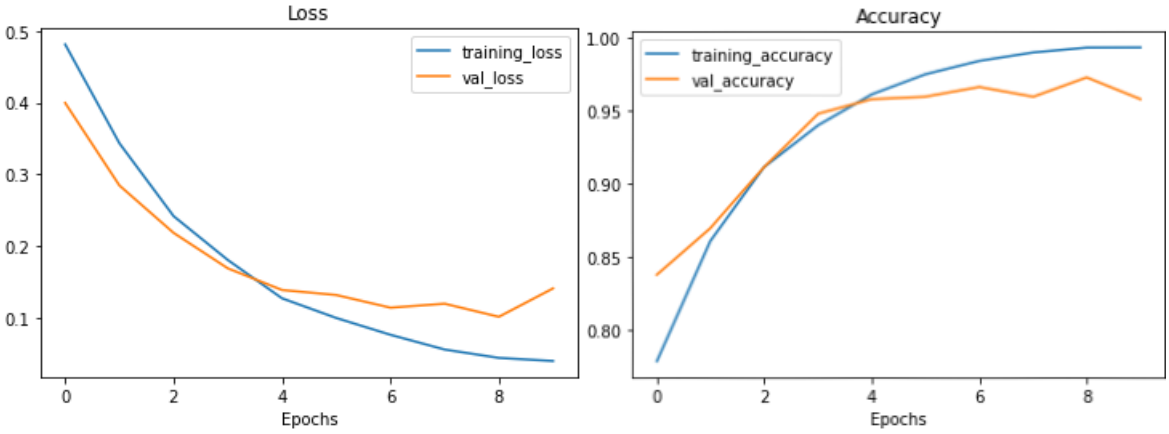

Figure 15. Learning curves of the baseline model.

The optimal position for the learning curves is to follow each other while the training curve is slightly above the validation curve. Furthermore, a large gap between the validation and training curve shows that the model is likely to overfit the data. So, the goal is to diminish this gap as much as possible.

The second CNN model has been improved a lot by adding two more convolution layers and two maximum pooling layers. The first two convolutional layers are mainly used to learn and extract features like edges and colours whereas the job of the third convolutional layer is to learn more complex features such as brain tumor borders to accurately detect them (Irmak, 2021). Examining the accuracy plot, it can be seen that the accuracy curves of the enhanced model follow each other, as shown in Figure 16. However, there is still a big gap between the loss curves which indicates that the model is probably overfitting the training data. Sometimes higher complexity doesn't necessarily mean that the CNN model will perform better.

Figure 16. Leaning curves of the enhanced model.

To diminish the gap between the loss curves and prevent the problem of overfitting, a final model is created which has almost the same architecture as the second model. The main difference is that a dropout layer is added between the dense layers. The performance of the final model improves almost immediately compared to the second model by adding only one dropout layer, demonstrating the power of this special layer. Figure 17 displays the learning curves of the final model. It can be observed that both of the loss and accuracy plots get a lot closer to each other, heading to the right direction. The training loss is decreasing smoothly, and it almost stabilised at the end of the training. In addition, the gap between the loss curves is reduced. Therefore, the final CNN model is the best among the other two, and it will be compared with the VGG16 pre-trained model at the end of this chapter.

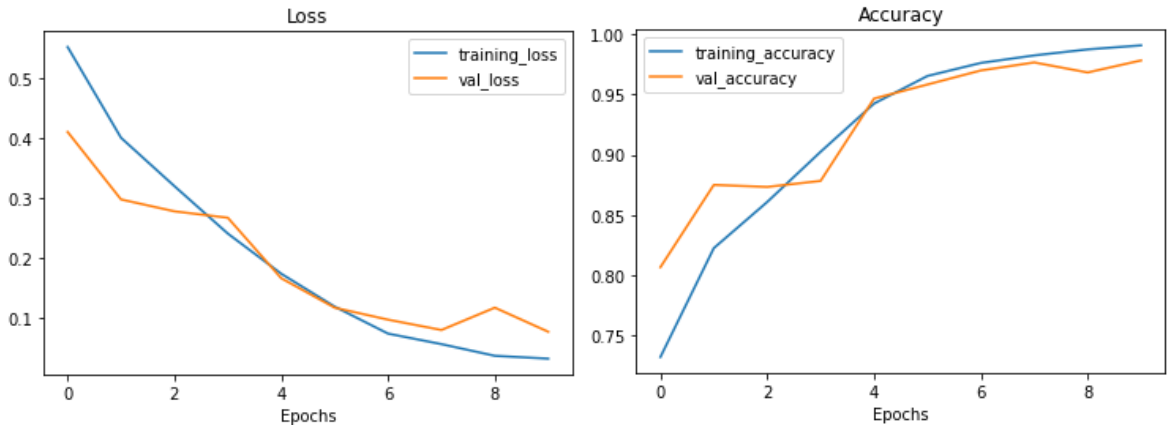

Figure 17. Learning curves of the best model.

4.1.1 Evaluation Metrics and real-time predictions of the final model

In this section, after the process of the binary classification task has been carried out, the performance of the best model should be also measured by various evaluation metrics and tested on brain MRI images that has never seen before. Using the sklearn library, a classification report is displayed in Figure 18 to investigate the evaluation metrics of the model. The model performed very well achieving 98% overall accuracy in brain tumor classification. In terms of other evaluation metrics such as recall and precision, the results obtained are satisfactory, demonstrating the effectiveness of the final model (see Table 7 for code in the Appendix A).

	precision	recall	f1-score	support
0	0.99	0.97	0.98	343
1	0.96	0.99	0.98	257
accuracy			0.98	600
macro avg	0.98	0.98	0.98	600
weighted avg	0.98	0.98	0.98	600

Figure 18. Classification report of the final model.

Finally, a confusion matrix of the final model is displayed in Figure 19. In classification tasks, a confusion matrix is a table that describes the performance of the model at determining the positive and negative outcome. The testing dataset includes 600 brain MRI images and the proposed only misclassified 13 images which is very impressive. More specifically, the model predicts that 3 images are affected by brain tumor while they are normal and 10 images that are not tumorous but in reality, they have brain tumor. These results proof the ability of the final proposed Convolutional Neural Network to accurately detect brain tumor in MRI images.

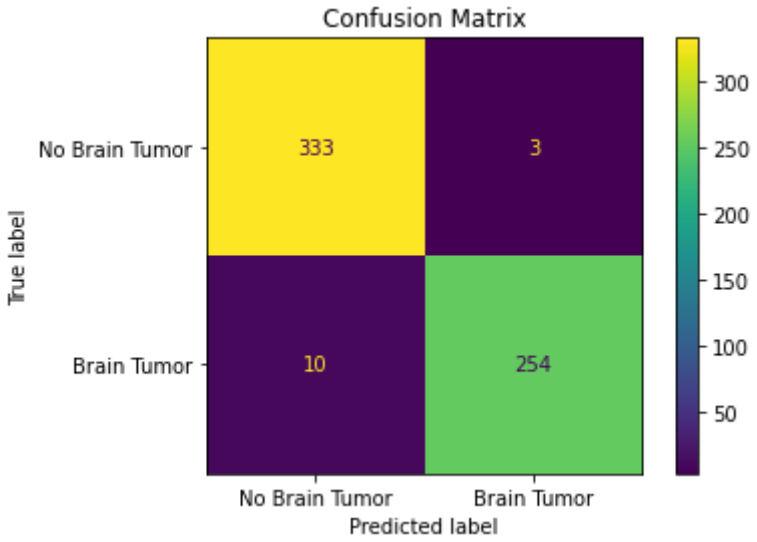

Figure 19. Confusion Matrix of the final model.

The final part is to test the performance of the model using the 'pred' folder. This folder contains brain MRI images different from the training and testing dataset and it is used to make predictions and examine how good the model performs on data that have never seen before. For the model to make predictions on unseen data, the brain images have to be in the same shape as the model has been trained on. Since the final model takes in images of shape (64,64,3), the prediction images have to be reshaped to use them with the model. However, after resizing the images, there is still a problem that should be taken into account. Although the brain image has now the right shape as the images that the model has been trained on, still a dimension is missing because the model was trained in batches. In reality, it is in the shape of (batch_size,64,64,3) and this problem can be fixed by adding an extra dimension to the image. Two random images are taken from the 'pred' folder, and the predictions can be seen in Figure 20.

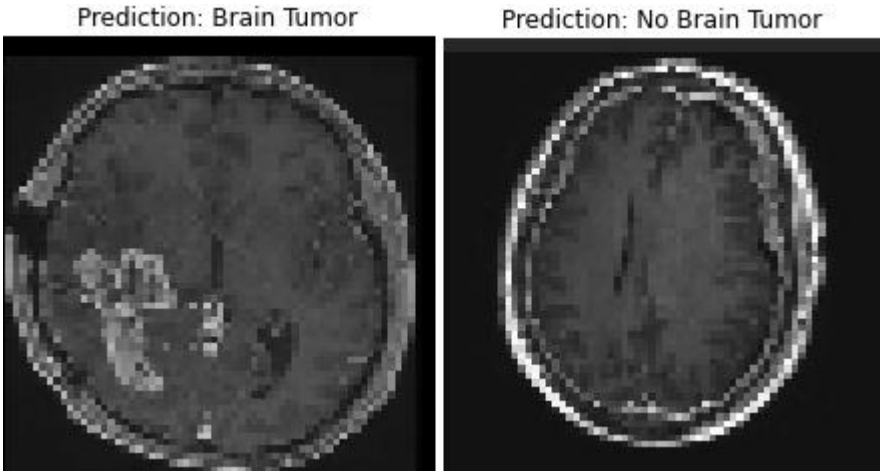

Figure 20. Prediction results of 2 test brain MRI images.

4.2 Results of the VGG16 model

The same performance evaluation process is repeated for the pre-trained model. The learning curves are examined before investigating other evaluation metrics to check whether the VGG16 model overfits the training data or not. As shown in Figure 21, the curves follow each other during training as well as the gap between the training and validation loss is very small. This actively demonstrates the fact that the pre-trained model fits good on the training data and can generalize on new brain images.

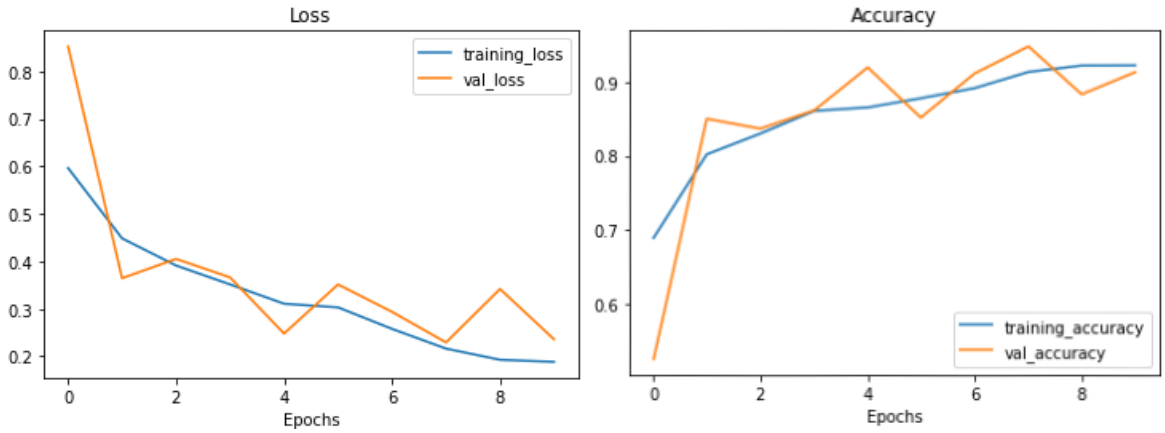

Figure 21. Loss and accuracy plots of the VGG16 model.

4.2.1 Evaluation Metrics and real-time predictions of the VGG16 model

A classification report and the confusion matrix are displayed again to measure and examine the performance of the VGG16 model in terms of the basic evaluation metrics. It can be observed that the pre-trained model obtained 91% accuracy for brain tumor detection while metrics like precision, recall, and f1-score achieved good overall results, as shown in Figure 22.

	precision	recall	f1-score	support
0	0.95	0.90	0.92	343
1	0.87	0.93	0.90	257
accuracy			0.91	600
macro avg	0.91	0.92	0.91	600
weighted avg	0.92	0.91	0.91	600

Figure 22. Classification report of the VGG16 model.

The confusion matrix in Figure 23, shows that the pre-trained model misclassified 52 brain images out of 600. It can be observed that it had a difficult experience to classify brain tumor as it predicts that 35 images were not tumorous, but they were affected by tumor. Also, the model predicts that 17 images are tumorous while they were normal (see Table 8 for code in the Appendix A).

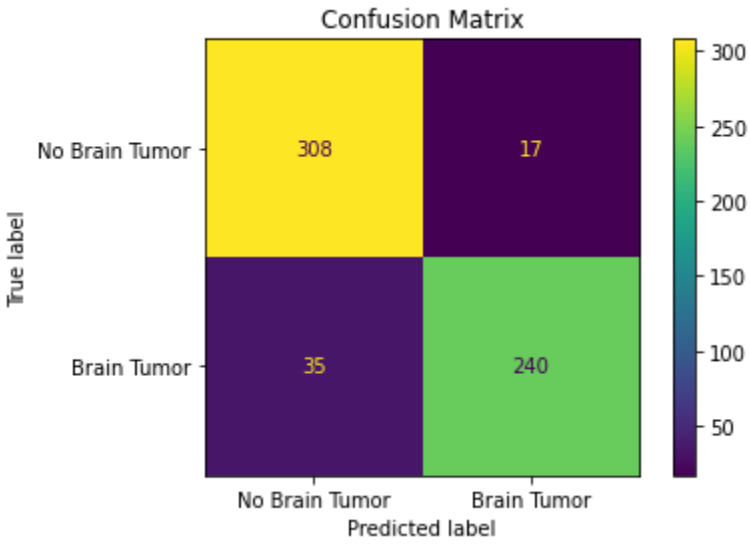

Figure 23. Confusion matrix of the VGG16 model.

Similarly, the pre-trained model is tested using brain images from the ‘pred’ folder. The size of the input image is re-sized to have the same size as the images that the model has been trained on and one extra dimension is added in the shape of the image. One random image is taken from the ‘pred’ folder, and the prediction can be seen in Figure 24.

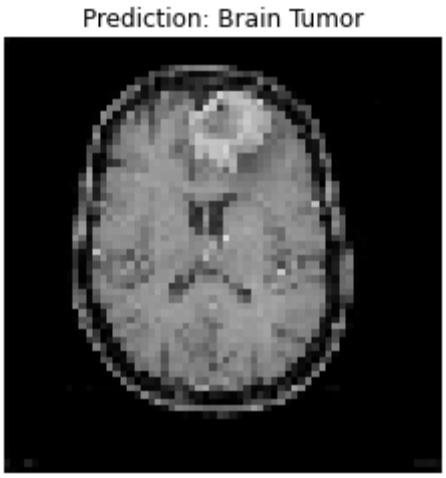

Figure 24. Prediction result of the VGG16 model.

4.3 Comparison of the final CNN and the VGG16 model

To summarize, both of the proposed models performed very well on data that have never seen before and most importantly, the problem of overfitting was prevented during the training phase. Figure 25 breakdowns the graphical comparison of the evaluation metrics between the final CNN model and the VGG16 model. It can be observed that even though that both models have high accuracies, the CNN model which is built from the scratch outperformed the pre-trained model. Furthermore, the loss and accuracy plots of the final model shows better performance in the training and testing phase as well as it is able to correctly classify and predict more brain MRI images that the VGG16 model. This is because that the CNN model is better at identifying and learning high level patterns from the given MRI brain image even though it learns way less trainable parameters. Finally, this model will be deployed in the data science product which is a web application that predicts whether a brain MRI image is affected by tumor or not.

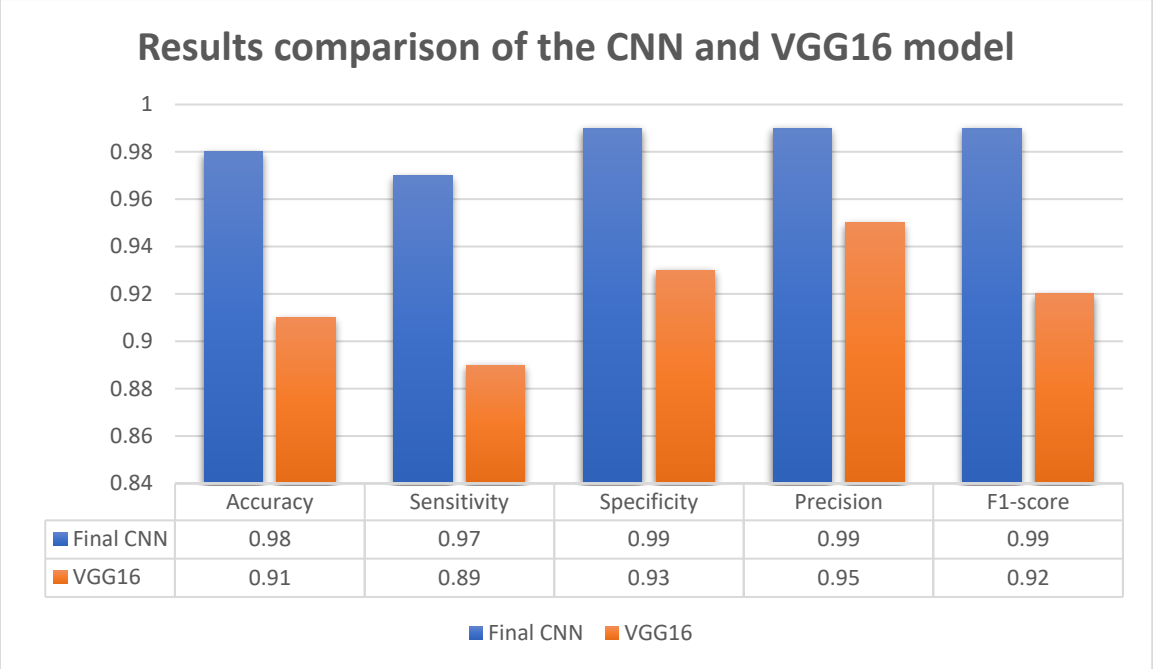

Figure 25. Graphical comparison of the proposed models.

4.4 Comparison of Results with Existing Literature

The results achieved by the best model are worth comparing to the results of other related studies that have been conducted in the same classification problem of brain tumor detection. For this purpose, the final convolution neural network model is compared to some popular state-of-the-art pre-trained model such as Inception v3 and ResNet in terms of accuracy. Table 2 shows that the proposed model outperformed other CNN and pre-trained models in brain tumor detection while the pre-trained ResNet model that obtained 97.2% accuracy is the closest model to the proposed model.

Authors	Classification Method	Accuracy
Mathew and Anto (2017)	Support Vector Machine	86%
Arora and Sharma, 2021	Inception V3	93.4%
Singh et al., 2021	CNN	90%

Çinar and Yildirim, 2020	ResNet with transfer learning	97.2%
Justin Paul et al., 2017	CNN	91.43%
Proposed model	CNN	98%

Table 2. Comparison of the proposed model with related studies.

5 Project Evaluation and Reflection

This chapter seeks to describe and provide useful information about the final data product of the project along with a critical evaluation about the social, legal, ethical, and professional issues raised by computer vision and AI in the field of healthcare. At the end of this chapter, the project is evaluated based on the quality of the research itself and how well each objective was met as well as, a personal reflection of the project is conducted.

5.1 Overview of Project

Once the final model is finalised and evaluated on unseen brain MRI images, it is then saved in hdf5 (h5) format which stands for Hierarchical Data Format in order to be deployed in the final data product. Coding part itself without a deliverable is meaningless and pointless because it cannot be used in the real-world. With that being said, as a final part of this project, a data product is created. A web application is designed using Python Flask REST API which can be used as a diagnostic tool for brain tumor detection. Flask is a micro web framework which helps users build web applications using Python programming language. Flask framework has two main key protocols which allow a client to obtain and provide information: the POST and GET methods. POST method is used to send HTML form data to a server while with the GET method a server sends data back to the user. Furthermore, an API which stands for Application Programming Interface, can serve as an interface for GET and POST requests.

The main idea behind the flask framework is that it wraps the final saved model as an API the get, and post requests are applied through routes. After posting information, the flask application will then feed it to the final model which is going to output its prediction. This process is used to develop the back-end of the website while this can be achieved with a few lines of code utilizing the flask library. In other words, a user can browse their files and upload a brain MRI image, then the image is passed to the final model and the flask application predicts whether the input brain image is tumorous or not, as displayed in Figure 26. It should be mentioned that a folder called ‘uploads’ is added and every time a user uploads an image, it will be saved in this folder. Also, a function named ‘secure_filename( )’ is applied to the filename of an uploaded file to secure it before storing it on the ‘uploads’ file. This step is extremely important because submitted files and data can be dangerous and

forged. The software used to create the web app and write Python code is the Visual Code Studio (see **Error! Reference source not found.** for code in Appendix A).

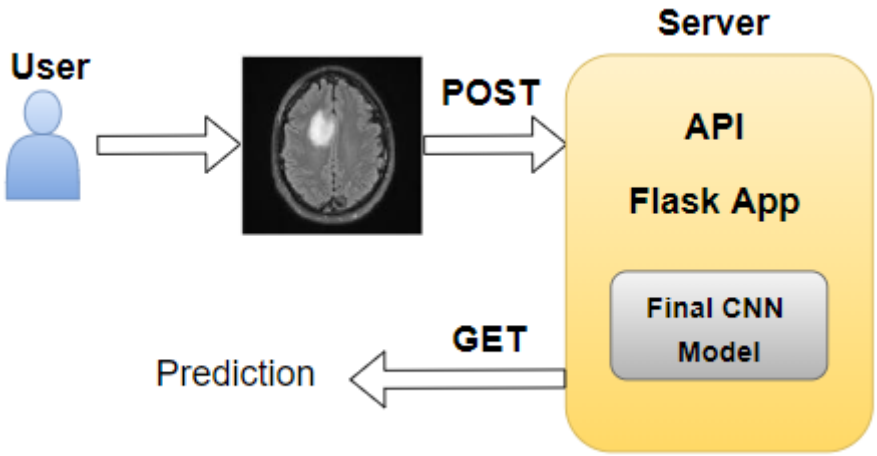

Figure 26. Flask operation behind the scenes.

The front-end development which refers on the visual representation of the website that the user interacts with, is created using HTML, CSS, and Bootstrap (see Table 9 for code in the Appendix). Figure 27 illustrates the front-end of the website. A user can choose a brain MRI image from their files and upload it in the application. Then, a button is appeared (Figure 28) where the user can click it and the app will make its prediction on brain tumor detection, as shown in Figure 29. As observable, the app makes correct predictions for the given brain MRI images which demonstrates the effectiveness of the proposed model as well as that the back-end is developed successfully. Furthermore, if the user desires to learn more about brain tumors, there is also a button which is a link to the Wikipedia site (see Table 10 for code in the Appendix A).

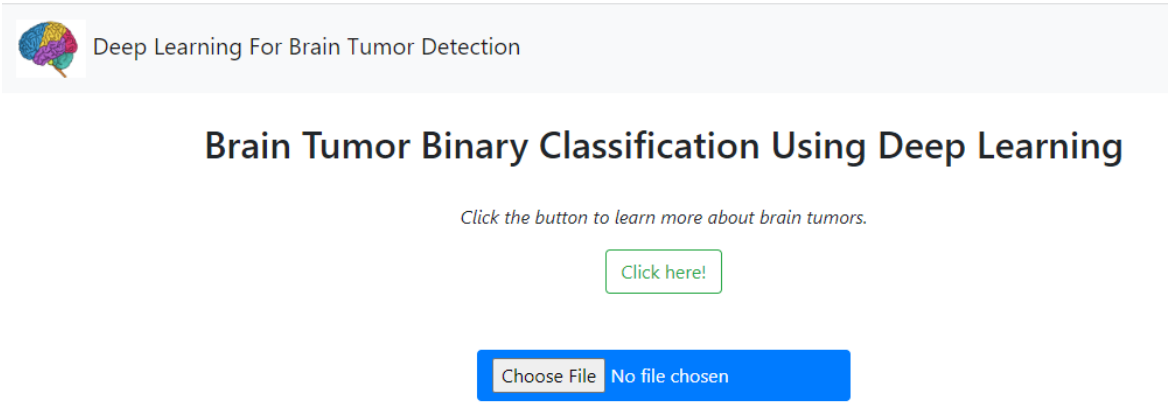

Figure 27. The front end of the website.

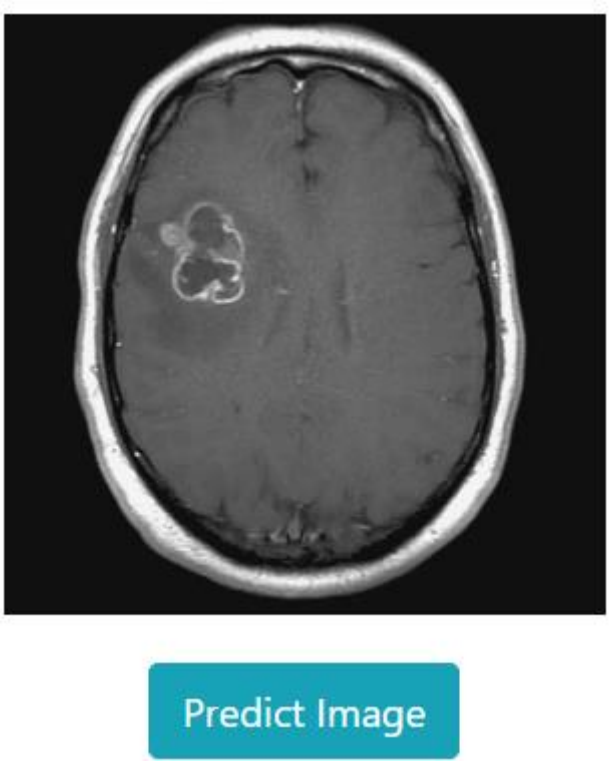

Figure 28. Predict button.

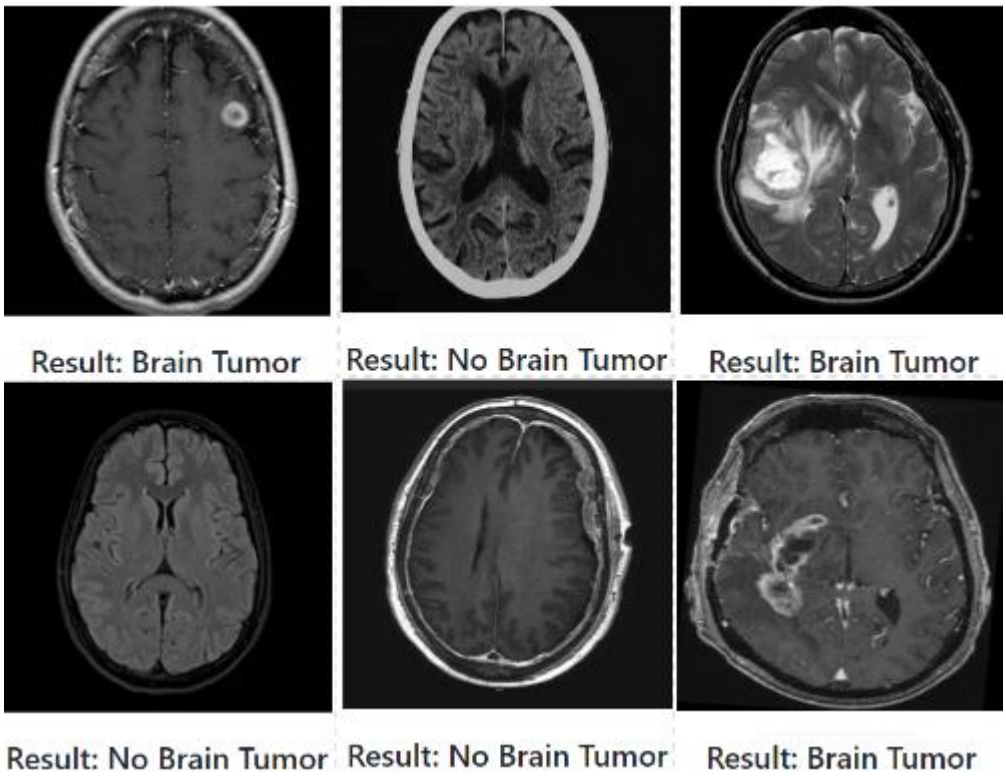

Figure 29. Predictions of six random brain images.

5.2 Evaluation of Methodology and Results

The Agile Development Lifecycle was implemented to design, develop, and evaluate the final data product and the project as a whole. Adopting the Agile methodology, the project is divided into small phases in order to make it easily manageable and successful. This approach consists of several stages that a project goes through from the planning into the launch phase, as shown in Figure 30 (Nehra, 2022). The first stage is the project planning which is the most crucial phase of project management. During this phase, an initial planning and research has been made and tasks have been scheduled according to the time frame of the project. Some activities such as building the CNN models required more time compared to others, hence appropriate planning and management makes it easier to create more complicated tasks and work around deadlines as well as make changes or adjustments if necessary. Also, the aim and objectives of the project are defined in this planning process as a specific and clearly defined aim can boost the performance of the project and increase its success rate.

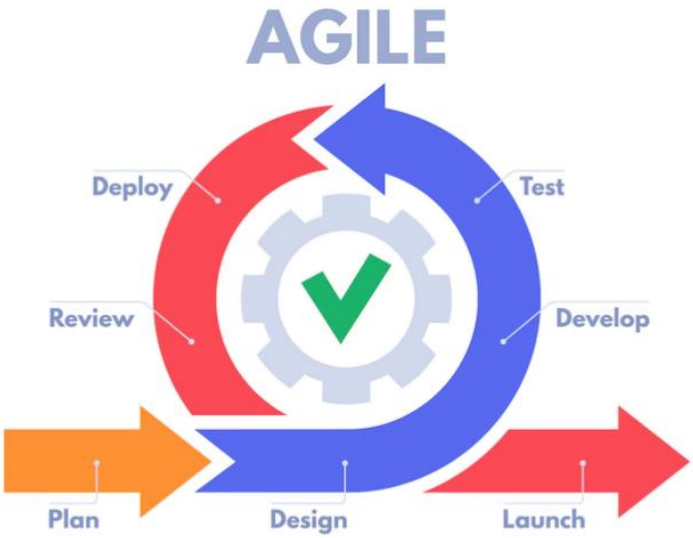

Figure 30. Agile Development Lifecycle.

The next phase of the Agile Methodology is the design, development, and evaluation of the final data product. Product design is extremely important before developing and launching any product, whether it is software or hardware-based product, because it interacts with the user. In this phase, the right resources and tools are selected to produce the code and deploy the final deep learning model as well as to develop the web application. The data product developed will follow the data science lifecycle, resulting in a web app, in an effective way to ensure that the end-product have all the requirements and functionality for a real-world medical scenario (Cady, 2017). Testing is an essential step in agile methodology because its regular occurrence. Testing process involves that the pipeline of code and the final application are completed without bugs and errors, and it runs in a way which is satisfactory for the users who test the web app.

The final phase of the Agile Lifecycle is to review and finally launch the product. Through the agile iteration, the product is again reviewed in a short period of time to make sure that it runs properly and then is launched to the real-world. The end user of the data science product who have no, or very limited knowledge of these technologies is expected to install the product locally in their personal computer and test it in an easy and user-friendly style. Therefore, the web application can be used as diagnostic tool for medical

professionals which allows them to post a brain MRI image and immediately receive results on brain tumor detection for each patient.

5.3 Evaluation of Ethical, Legal, Social, Security, and Professional issues

Over the past few years, Artificial Intelligence and especially Computer Vision are drawing much attention in the medical field, as medicine is becoming more and more data-driven discipline. Medical data are complicated because they can take different and complex forms such as patient medical exam results, data from images in many methods (MRI or CT), and data derived from healthcare testing machines. Therefore, the analysis required to examine, find patterns, and draw conclusion in healthcare goes beyond the classical statistics approaches while computer vision and deep learning applications show promising results in that field. These applications can used as diagnostic tools to support clinical experts to make better decisions about the treatment of the patients. For example, computer vision utilization can be used to accurate detect tumor in brain MRI images making the process less time consuming and boring. However, Vellido (2019) argues in his article that AI encounters a rapid procedure of commodification while this is related to many issues that reach out in society. Hence, societal issues may rise with the use of AI, and they should not be neglected especially in the medical domain.

The rapid advancement of AI systems has led to many and complicated ethical issues which must often be examined to ensure and guarantee that they are not used in a way that harm any individual. In many cases, Artificial Intelligence and deep learning applications can enhance the effectiveness of medical treatments but sometimes bias can creep in (Shaheen, 2021). This is mainly happened when the systems and algorithms are trained on limited or inadequate data which can lead to serious problems for patients. For example, during this study and based on the thorough literature review, a convolution neural network learns patterns from the given data and when the model is not trained properly and is overfitted, then it can absorb biases from that data. It is therefore very important that people who make and train AI systems to be aware of the bias risk and be able to minimize it at every phase of a data product development either by carefully deciding what dataset to use or which deep learning technology want to apply to train the systems (Gerke et al., 2020).

So, any deep learning model or algorithms trained by humans will be as fair, justice, and trustworthy as the data that are trained with.

Moreover, bias can also affect the decision-making of the clinical experts which has a detrimental effect on the patients and the society as a whole. AI systems are not always fair, and trustworthy resulting in healthcare issues of the patients. For example, an AI technology can prescribe the wrong treatment recommendation for an individual and the clinical expert can adopt this result which can harm the health of the patient (Shaheen, 2021). Hence, not only must professionals treat patients with care and expertise, but they also be aware of the systems drawbacks and limitations to make better decisions. Professional clinicians should be hold accountable and liable and they should not only rely on these technologies. AI systems are tools which should be under of the doctors who make the decision ultimately (Ammanath, 2022). When the well-being and health of an individual only depends on the decision of AI tools, the consequences could be very harmful for the patient, the clinical experts, and the society as a whole.

Needless to say, data protection, privacy and security are also an issue of AI in the medical field. Protecting patients' personal information is vital as medical data are increasingly exchanged between physicians and the care team or creating products based on these technologies. For instance, patient data can be shared outside of the hospital for extra investigation in order to gain some insights about their condition, carries the risk of data leakage and privacy. Healthcare data privacy cannot be maintained without security which should be adopted to build trust in AI systems. Some security safeguards that AI companies can implement is to routinely monitor and audit the data and the information systems as well as to determine and understand who is going to have access to the data and ensure that strict controls are provided (Malek & Pralika , 2022). Due to data protection and security issues, medical data are rarely publicly available by professionals making deep learning models' training more challenging. However, the dataset used in this study is publicly available from Kaggle. Moreover, without human interpretation and mediation the use of AI can raise challenges and issues in cybersecurity, making it vulnerable.

Legal concerns are raised by any use of AI and computer vision applications in the domain of medicine. A relevant example is the recent application of the General Data Protection Regulation (GDPR) which mandates that all decisions made by artificial intelligence and automated systems must be clearly explained (Vellido, 2019). Therefore,

all patients who receive treatment based on AI algorithmic systems must be provided with meaningful information behind this decision. The way that AI works for automated decision-making place these technologies in a legal spotlight. Finally, data governance should not be dimmed by the cutting-edge AI systems. Public trust and policies which ensure transparency, justice, and equality are required to build AI applications which can improve patients' health and health services.

5.4 Project Evaluation and Personal Reflection

5.4.1 Project Evaluation

Overall, convolutional neural networks are frequently employed in image classification tasks especially in the healthcare domain to diagnose medical diseases. Based on the literature review, different CNN architectures designed from the scratch and different pre-trained models using the technique of transfer learning shown good overall performance in brain tumor binary as well as in multi-class classification. However, depending on the image classification task, the structure, the complexity, and the parameters of the models should be changed accordingly. One of the difficulties encountered in this project is choosing the right architecture for the CNN model and train it properly to prevent the problem of overfitting. Sometimes, complex models with more trainable parameters do not achieve better results while adding only one dropout layer can significantly change the performance of the model. In brain tumor detection, the final model created by the scratch trained effectively showing to be a good-fitted model and achieved satisfactory accuracy of 98%. However, before measuring the performance of any deep learning model based on the standard evaluation metrics, the loss and accuracy plots should always be examined first to check if the model overfits or underfits the training data. Another key element obtained from the literature is that better results have been achieved by the models which have been trained using large datasets. Moreover, the final data product which is a web app for brain tumor detection can be used from clinical experts and radiologists as a second opinion tool for more accurate and fast results.

5.4.2 Personal Reflection

Personally speaking, valuable knowledge acquired throughout the whole journey of this project. To begin with, skills in the field of deep learning and computer vision are gained as different and many techniques were used to design and effectively train the custom Convolutional Neural Network models as well as how to use utilize the art of transfer learning to pre-train them. Thanks to this project, theoretical knowledge is also grasped about how CNNs can 'understand' a brain MRI image well-enough to extract the most important features and solve problems like the human visual system can do. Knowledge in TensorFlow framework and keras library is obtained in order to build the image classifier model. Many difficulties encountered during the training phase of the models like the problem of overfitting, which was addressed effectively, demonstrating the quality of the research.

Furthermore, having no experience at all in web development, skills in HTML, CSS, Bootstrap, and Flask library are learnt. Flask framework was selected over Django to develop the web application because it is light-weight and more beginner friendly as it is required a few lines of python code to achieve the desired results. Moreover, understanding each component from the back-end technology is very important to tackle any errors and bags creating the app while flask framework makes it very helpful to achieve it.

Finally, a critical evaluation of the ethical, legal, security, and professional issues raised by computer vision and artificial intelligence applications has been made. Regarding this project, an ethical consideration that should be taken into account is the source of the dataset extracted from. The dataset is publicly available on Kaggle and therefore there is no concern about data privacy and protection. Moreover, when deep learning models are not trained effectively and the medical data are inadequate, bias can creep in which can lead to serious problems in the patients' health because it can affect the decision made by the clinical experts. At the end of the day, patient safety, privacy, and security should be prioritized by healthcare organizations, and anything may threaten these components should be evaluated in depth.

6 Conclusion and Recommendations

This chapter summarizes and evaluates the extent to which the objectives made in the introduction chapter have been met and justified through the literature review as well as the practical work of this dissertation. Also, a recommendation is included which focuses on the further research and practice that can be done as extent at the end of the project.

6.1 Conclusion

To summarize, the latest advances in the field of computer vision lead research and studies to evolve from traditional machine learning methods to the state-of-the-art deep learning techniques. Convolutional Neural Networks automatically extracts features and learn the most important patterns from a natural image in a way that is effective and far better than the classical way of feature extraction. This project presents a binary brain tumor classification for early diagnosis purposes using two approaches. With minimal pre-processing steps, a custom CNN model designed from the scratch, trained, and improved through experimentation. The best CNN model was compared with the VGG16 model which pre-trained using transfer leaning on a popular database called ImageNet. From the analysis of the experimental results performed on various brain MRI images, the final CNN model obtained a high accuracy of 98%, having better performance than the pre-trained model in terms of all the standard evaluation metrics. Based on the experimental results, the final proposed model not only outperformed studies with the traditional machine learning methods but also researches that utilize other pre-trained models and different CNNs' architectures. This study also presents a web application based on deep learning which can possibly contribute to healthcare as a tool utilised by clinical doctors and experts to assist them in brain tumor detection. To conclude, AI systems and computer vision application have many benefits and highly encouraged in the field of medicine as long as their main focus is on the health and well-being of the patients, protecting their rights and privacy through the prevention of AI misuse.

6.2 Recommendations

Although the custom CNN and the VGG16 models performed very well by achieving 98% and 91% overall accuracy, there is still scope for enhancement in terms of

increasing the accuracy and achieving better learning curves. This can be achieved by exploring and performing more pre-processing steps. For future work, various pre-processing steps will be applied such as cropping the input image or using a technique called data augmentation that increases the sample of data by rotating and flipping the images. These techniques can boost the performance of the image classifiers. Regarding the training of the models, different special layers can be explored and applied. For instance, instead of normalizing the data before training the models, a special layer called batch normalization, that scales the input data, can be used when building the model. Further research can also be made evolving the binary classification task into a multi-class classification task. This means that a model can be built to predict different types of tumors such as glioma, meningioma, and pituitary and therefore can be used in a practical application that can help doctors make better decisions in the medical field. Last but not least, image segmentation can also be performed in future research.

7 Reference List

Ammanath, B., 2022. How 'computer vision' could change healthcare, retail and more. [Online]
Available at: https://www.weforum.org/agenda/2022/03/how-computer-vision-change-healthcare/

Ankireddy, S. (2020) 'Assistive Diagnostic Tool for Brain Tumor Detection using Computer Vision', in 2020 IEEE MIT Undergraduate Research Technology Conference, URTC 2020. Institute of Electrical and Electronics Engineers Inc. Available at: https://doi.org/10.1109/URTC51696.2020.9668906.

Arbane, M., Benlamri, R., Brik, Y. and Djerioui, M. (2021) 'Transfer Learning for Automatic Brain Tumor Classification Using MRI Images', in 2020 2nd International Workshop on Human-Centric Smart Environments for Health and Well-Being, IHSH 2020. Institute of Electrical and Electronics Engineers Inc., pp. 210–214. Available at: https://doi.org/10.1109/IHSH51661.2021.9378739.

Arora, S. and Sharma, M. (2021) 'Deep Learning for Brain Tumor Classification from MRI Images', in Proceedings of the IEEE International Conference Image Information Processing. Institute of Electrical and Electronics Engineers Inc., pp. 409–412. Available at: https://doi.org/10.1109/ICIIP53038.2021.9702609.

Arunachalam, M. and Royappan Savarimuthu, S. (2017) 'An efficient and automatic glioblastoma brain tumor detection using shift-invariant shearlet transform and neural networks', International Journal of Imaging Systems and Technology, 27(3), pp. 216–226. Available at: https://doi.org/10.1002/ima.22227.

Badža, M.M. and Barjaktarović, M.C. (2020) 'Classification of brain tumors from mri images using a convolutional neural network', Applied Sciences (Switzerland), 10(6). Available at: https://doi.org/10.3390/app10061999.

Bahadure, N.B., Ray, A.K. and Thethi, H.P. (2017) 'Image Analysis for MRI Based Brain Tumor Detection and Feature Extraction Using Biologically Inspired BWT and SVM', International Journal of Biomedical Imaging, 2017. Available at: https://doi.org/10.1155/2017/9749108.

Baker Curl (2021) 'Cancer: summary of statistics (England)', House of Commons Library [Preprint].

Baranwal, S.K., Jaiswal, K., Vaibhav, K., Abhishek, K. and Srikantaswamy (2020) 'Performance analysis of Brain Tumour Image Classification using CNN and SVM', Proceedings of the Second International Conference on Inventive Research in Computing Applications (ICIRCA-2020) [Preprint].

Brock, A., Lim, T., Ritchie, J.M. and Weston, N. (2017) 'FreezeOut: Accelerate Training by Progressively Freezing Layers'. Available at: http://arxiv.org/abs/1706.04983.

Brownlee, J., 2019. Machine Learning Mastery. [Online]
Available at: https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/

Brownlee, J., 2020. Machine Learning Mastery. [Online]
Available at: https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/

Cady, F., 2017. The Data Science Handbook. NJ: John Wiley & Sons.

CancerNet, 2022. Cancer.Net. [Online]
Available at: https://www.cancer.net/cancer-types/brain-tumor/statistics#:~:text=It%20is%20estimated%20that%2018%2C280,and%20CNS%20tumors%20in%202020.

Cheng, J., Huang, W., Cao, S., Yang, R., Yang, W., Yun, Z., Wang, Z. and Feng, Q. (2015) 'Enhanced performance of brain tumor classification via tumor region augmentation and partition', PLoS ONE, 10(10). Available at: https://doi.org/10.1371/journal.pone.0140381.

Cheng, J., Yang, W., Huang, M., Huang, W., Jiang, J., Zhou, Y., Yang, R., Zhao, J., Feng, Y., Feng, Q. and Chen, W. (2016) 'Retrieval of Brain Tumors by Adaptive Spatial Pooling and Fisher Vector Representation', PLoS ONE, 11(6). Available at: https://doi.org/10.1371/journal.pone.0157112.

Çinar, A. and Yildirim, M. (2020) 'Detection of tumors on brain MRI images using the hybrid convolutional neural network architecture', Medical Hypotheses, 139. Available at: https://doi.org/10.1016/j.mehy.2020.109684.

Deepak, S. and Ameer, P.M. (2019) 'Brain tumor classification using deep CNN features via transfer learning', Computers in Biology and Medicine, 111. Available at: https://doi.org/10.1016/j.compbiomed.2019.103345.

Gerke, S., Minssen, T. and Cohen, G. (2020) 'Ethical and legal challenges of artificial intelligence-driven healthcare', in Artificial Intelligence in Healthcare. Elsevier, pp. 295–336. Available at: https://doi.org/10.1016/B978-0-12-818438-7.00012-5

Hamada, A., 2020. Kaggle. [Online]
Available at: https://www.kaggle.com/datasets/ahmedhamada0/brain-tumor-detection

Irmak, E. (2021) 'Multi-Classification of Brain Tumor MRI Images Using Deep Convolutional Neural Network with Fully Optimized Framework', Iranian Journal of Science and Technology - Transactions of Electrical Engineering, 45(3), pp. 1015–1036. Available at: https://doi.org/10.1007/s40998-021-00426-9.

Justin Paul, B., Fabbri, D. and Landman, B. (2017) 'Deep learning for brain tumor classification, medical imaging 2017 and Photonics', biomedical applications in molecular, structural, and functional imaging. International Society for Optics and Photonics [Preprint].

Kang, J., Ullah, Z. and Gwak, J. (2021) ‘Mri-based brain tumor classification using ensemble of deep features and machine learning classifiers’, Sensors, 21(6), pp. 1–21. Available at: https://doi.org/10.3390/s21062222.

Malek, L. A. & Pralika , J., 2022. Reuters. [Online]
Available at: https://www.reuters.com/legal/litigation/data-privacy-artificial-intelligence-health-care-2022-03-17/

Mathew, R. and Anto, B. (2017) ‘TUMOR DETECTION AND CLASSIFICATION OF MRI BRAIN IMAGE USING WAVELET TRANSFORM AND SVM’, in International Conference on Signal Processing and Communication.

Mehmood, A., Maqsood, M., Bashir, M. and Shuyuan, Y. (2020) ‘A deep siamese convolution neural network for multi-class classification of alzheimer disease’, Brain Sciences, 10(2). Available at: https://doi.org/10.3390/brainsci10020084

Nehra, M., 2022. Decipher Zone. [Online]
Available at: https://www.decipherzone.com/blog-detail/agile-development-lifecycle

NHS, 2020. [Online]
Available at: https://www.nhs.uk/conditions/brain-tumours/

Parveen and Singh, A. (2015) ‘Detection of brain tumor in MRI images, using combination of fuzzy c-means and SVM’, in 2nd International Conference on Signal Processing and Integrated Networks, SPIN 2015. Institute of Electrical and Electronics Engineers Inc., pp. 98–102. Available at: https://doi.org/10.1109/SPIN.2015.7095308.

Seldon, M., 2021. Seldon. [Online]
Available at: https://www.seldon.io/transfer-learning#:~:text=The%20main%20benefits%20of%20transfer,model%20will%20be%20pre%2Dtrained.

Singh, V., Sharma, S., Goel, S., Lamba, S. and Lamba, S. (2021) ‘Brain Tumor Prediction by Binary Classification Using VGG-16’, in Smart and Sustainable Intelligent Systems. Wiley, pp. 127–138. Available at: https://doi.org/10.1002/9781119752134.ch9.

Sriramakrishnan, P., Kalaiselvi, T., Thirumalaiselvi, M., Padmapriya, S.T. and Ramkumar, S. (2019) ‘A Role of Medical Imaging Techniques in Human Brain Tumor Treatment’, International Journal of Recent Technology and Engineering, 8(4S2), pp. 565–568. Available at: https://doi.org/10.35940/ijrte.d1105.1284s219.

Swati, Z.N.K., Zhao, Q., Kabir, M., Ali, F., Ali, Z., Ahmed, S. and Lu, J. (2019) ‘Brain tumor classification for MR images using transfer learning and fine-tuning’, Computerized Medical Imaging and Graphics, 75, pp. 34–46. Available at: https://doi.org/10.1016/j.compmedimag.2019.05.001.

Tandel, G.S., Biswas, M., Kakde, O.G., Tiwari, A., Suri, H.S., Turk, M., Laird, J.R., Asare, C.K., Ankrah, A.A., Khanna, N.N., Madhusudhan, B.K., Saba, L. and Suri, J.S. (2019) ‘A review on a deep learning perspective in brain cancer classification’, Cancers. MDPI AG. Available at: https://doi.org/10.3390/cancers11010111

Thakur, R., 2019. Towards Data Science. [Online]
Available at: https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c

Ucuzal, H., YAŞAR, Ş. and Çolak, C. (2019) 'Classification of brain tumor types by deep learning with convolutional neural network on magnetic resonance images using a developed web-based interface', 2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), pp. 1–5. Available at: https://doi.org/10.1109/ISMSIT.2019.8932761.

Varuna Shree, N. and Kumar, T.N.R. (2018) 'Identification and classification of brain tumor MRI images with feature extraction using DWT and probabilistic neural network', Brain Informatics, 5(1), pp. 23–30. Available at: https://doi.org/10.1007/s40708-017-0075-5.

Vellido, A. (2019) 'Societal Issues Concerning the Application of Artificial Intelligence in Medicine', Kidney Diseases, 5(1), pp. 11–17. Available at: https://doi.org/10.1159/000492428.

WHO, 2022. World Health Organization. [Online]
Available at: https://www.who.int/news-room/fact-sheets/detail/cancer
[Accessed 2022].

Yousef Shaheen, M. (2021) 'AI in Healthcare: medical and socio-economic benefits and challenges', SO [Preprint]. Available at: https://doi.org/10.14293/S2199-1006.1.SOR-.PPRQNI1.v1.

Appendix A. Program Code

```
# Upload the API key from my kaggle account
from google.colab import files

upload = files.upload()

for fn in upload.keys():
  print('User uploaded file "{name}" with length {length}
bytes'.format(
      name=fn, length=len(upload[fn])))

# move the kaggle.json into the folder where tha API expects to find
it
!mkdir -p ~/.kaggle/ && mv kaggle.json ~/.kaggle/ && chmod 600
~/.kaggle/kaggle.json

# download the dataset
!kaggle datasets download -d ahmedhamada0/brain-tumor-detection
import zipfile

# unzip the dataset and read it
zip_ref = zipfile.ZipFile("brain-tumor-detection.zip", "r")
zip_ref.extractall()
zip_ref.close()
```

Table 3. Extract and load the dataset from Kaggle.

```
# import necessary libraries
import os
import pathlib
import random
import cv2
from PIL import Image
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from keras.models import Sequential, load_model
from keras.layers import Conv2D, MaxPooling2D, ZeroPadding2D
from keras.layers import Flatten, Dense, Dropout, Activation,
GlobalAveragePooling2D
from tensorflow.keras.utils import plot_model
from keras.applications import vgg16
from keras.models import Model

# Walk through the 'no' directory and list number of files
for dirpath, dirnames, filenames in os.walk("no"):
  print(f"There are {len(dirnames)} directories and {len(filenames)}
images in '{dirpath}'.")
# also walk through in the 'yes' directory
```

```
for dirpath, dirnames, filenames in os.walk("yes"):
  print(f"There are {len(dirnames)} directories and {len(filenames)}
images in '{dirpath}'.")
# Another way to find out how many images are in a file
no_tumor_images = len(os.listdir("no"))
tumor_images = len(os.listdir("yes"))



# print the length of the images
print(f'Number of no tumorous brain images: {no_tumor_images}')
print(f'Number of tumorous brain images: {tumor_images}')
#view an image

def view_random_image(target_dir):
  # Setup target directory (we'll view images from here)
  target_folder = target_dir

  # Get a random image path
  random_image = random.sample(os.listdir(target_folder), 1)

  # Read in the image and plot it using matplotlib
  img = mpimg.imread(target_folder + "/" + random_image[0])

  # show the image
  plt.imshow(img)
  plt.axis("off");
  plt.title(target_dir)

  print(f"Image shape: {img.shape}") # print the shape of the image

  return img
# View a random image from the targeted folder ('no')
image1 = view_random_image(target_dir='no')
# View a random image from the 'yes' folder
image2 = view_random_image(target_dir='yes')
# view the  first image (actually just a big tensor/array)
image1
# view the image shape
image1.shape #returns (width,height,colour channel)
```

Table 4. Inspect and visualize the data.

```
# create two empty lists to store the images and the labels for each
image
dataset = []
label = []

# create a path and then return the entries in the directory given by
the path
image_directory_no = 'no/'

# path for with no brain tumor
no_tumor_images = os.listdir(image_directory_no)
```

```
for i, image_name in enumerate(no_tumor_images):
    #split the image name and check if the type is jpg
    if(image_name.split('.')[1]=='jpg'):

        # load the image from the targeted directory
        image=cv2.imread(image_directory_no + image_name)

        # convert the PIL image into an array
        image=Image.fromarray(image,'RGB')

        # resize the image size so all the images have the same width
and height
        image=image.resize(((64,64)))

        # convert the image into numpy array and append it into our
dataset
        dataset.append(np.array(image))

        # append the associated labels of the image in the label list
and convert them in an array format
        label.append(0) #0 goes for no tumorous brain images

# check the length of the dataset
len(dataset)
# create a path and then return the entries in the directory given by
the path
image_directory_yes = 'yes/'

# path for tumorous brain images
tumor_images = os.listdir(image_directory_yes)

for i, image_name in enumerate(tumor_images):
    #split the image name and check if the type is jpg
    if(image_name.split('.')[1]=='jpg'):

        # load the image from the targeted directory
        image=cv2.imread(image_directory_yes + image_name)

        # convert the PIL image into an array
        image=Image.fromarray(image,'RGB')

        # resize the image size so all the images have the same width
and height
        image=image.resize(((64,64)))

        # convert the image into numpy array and append it into our
dataset
        dataset.append(np.array(image))

        # append the associated labels of the image in the label list
and convert them in an array format
        label.append(1) #1 goes for  tumorous brain images
#check the length of the dataset and the label variable
print(f'The dataset contains {len(dataset)} images')
print(f'Labels in each image: {len(label)}'
# convert the dataset and label into a numpy array
dataset = np.array(dataset)
```

```
label = np.array(label)
#check the type
type(dataset)
type(label)
#check the shape of the dataset
dataset.shape
# more visualizations
def image(data,i):
    #plot the i image
    plt.imshow(data[i])
    print(f'The shape is: {data[i].shape}')
    if label[1] == 0:
      plt.title('No tumor in the brain')
    else:
      plt.title('Tumor in the brain')
#view the image
image(dataset,1)
# split the dataset into training and testing subsets
from sklearn.model_selection import train_test_split
train_data, test_data, train_labels, test_labels =
train_test_split(dataset,label,test_size=0.2,

random_state=0,shuffle=True)
# Show the first training example
print(f"Training sample:\n{train_data[0]}\n")
print(f"Training label: {train_labels[0]}")
#check the shape of our data
train_data.shape, train_labels.shape, test_data.shape,
test_labels.shape
#plot a single example
plt.imshow(train_data[0]);
#check our samples label
train_labels[0]
# create a list of the class names
class_names = ['No Brain Tumor','Brain Tumor']
plot an example image and its label
plt.imshow(train_data[17])
plt.title(class_names[train_labels[17]]);
#plot multiple random images from our training data set
plt.figure(figsize=(7,7))
for i in range(4):
    ax = plt.subplot(2,2,i+1)
    rand_index = random.choice(range(len(train_data)))
    plt.imshow(train_data[rand_index])
    plt.title(class_names[train_labels[rand_index]])
    plt.axis(False)



# Normalize the data
train_data_norm = tf.keras.utils.normalize(train_data,axis=1)
test_data_norm = tf.keras.utils.normalize(test_data,axis=1)
# check a single example of normalized data
train_data_norm[1]
```

Table 5. Create the dataset and split it into training and testing subsets.

```
#create a simple CNN model
model_1 = keras.Sequential([
          layers.Conv2D(filters=32,
                        kernel_size=(3,3),
                        activation='relu',
                        input_shape=(64,64,3)),
          layers.MaxPooling2D(pool_size=(2,2)),
          layers.Flatten(),
          layers.Dense(units=1,activation='sigmoid')
])
#compile the model
model_1.compile(optimizer='adam',
                loss='binary_crossentropy',
                metrics=['accuracy'])
#fit the model
history_1 = model_1.fit(train_data_norm,train_labels,
                    epochs=10, batch_size=16,
                    verbose=1,
                    validation_data=(test_data_norm,test_labels))
#to inspect the performance of each model,a seperate plot is used for
training and validation data
def plot_loss_curves(history):
    loss = history.history['loss']
    val_loss = history.history['val_loss']

    accuracy = history.history['accuracy']
    val_accuracy = history.history['val_accuracy']

    epochs = range(len(history.history['loss']))

   # Plot loss
    plt.plot(epochs, loss, label='training_loss')
    plt.plot(epochs, val_loss, label='val_loss')
    plt.title('Loss')
    plt.xlabel('Epochs')
    plt.legend()

  # Plot accuracy
    plt.figure()
    plt.plot(epochs, accuracy, label='training_accuracy')
    plt.plot(epochs, val_accuracy, label='val_accuracy')
    plt.title('Accuracy')
    plt.xlabel('Epochs')
    plt.legend();
#plot the learning curves of the model_1
plot_loss_curves(history_1)



#create a model with higher complexity

model_2 = keras.Sequential([
            layers.Conv2D(filters = 32,
                          kernel_size = (3,3),
                          activation='relu',
                          input_shape=(64,64,3)),
            layers.MaxPooling2D(pool_size=(2,2)),
```

```
            layers.Conv2D(32,(3,3),activation='relu'),
            layers.MaxPooling2D(pool_size=(2,2)),

            layers.Conv2D(64,(3,3),activation='relu'),
            layers.MaxPooling2D(pool_size=(2,2)),

            layers.Flatten(),
            layers.Dense(100,activation='relu'),
            layers.Dense(1,activation='sigmoid')

])
#compile the model
model_2.compile(optimizer='adam',loss='binary_crossentropy',
              metrics=['accuracy'])
#fit the model
history_2 = model_2.fit(train_data_norm,train_labels,
                    epochs=10, batch_size=16,
                    verbose=1,
                    validation_data = (test_data_norm,test_labels))
#plot the learning curves
plot_loss_curves(history_2)


#create another model with a dropout layer
model=Sequential()

model.add(Conv2D(32, (3,3), input_shape=(64,64, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(32, (3,3), kernel_initializer='he_uniform'))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))


model.add(Conv2D(64, (3,3), kernel_initializer='he_uniform'))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Flatten())
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(1))
model.add(Activation('sigmoid'))
#compile the model
model.compile(loss='binary_crossentropy',optimizer='adam',
metrics=['accuracy'])
#fit the model
history = model.fit(train_data_norm,train_labels,
batch_size=16,
verbose=1, epochs=10,
validation_data=(test_data_norm, test_labels))

plot_loss_curves(history)
```

Table 6. Build the CNN models.

```
probs = model.predict(test_data_norm)
probs[:3]
test_labels[:10]
preds = []
for element in probs:
  if element > 0.5:
    preds.append(1)
  else:
    preds.append(0)
preds[:10]
from sklearn.metrics import confusion_matrix, classification_report

print(classification_report(test_labels,preds))
#a fuction which displays the confusion matrix
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

def confusion_matrix_calc(preds,test_labeles,num_classes):
    positions = np.arange(0,num_classes)
    classes = np.arange(0,num_classes)
    cm=confusion_matrix(preds, test_labels,labels=classes)
    disp=ConfusionMatrixDisplay(cm,display_labels=classes)
    classes_name = class_names
    plt.figure(figsize=(10,10))
    disp.plot(cmap = "viridis")
    plt.xticks(positions, classes_name)
    plt.yticks(positions, classes_name)
    plt.title('Confusion Matrix')
    return
confusion_matrix_calc(preds,test_labels,2)
# See the inputs and outputs of each layer from the best model
plot_model(model, show_shapes=True)
#save the model
model.save('final_model_brain_tumor.h5')

#print the classes that we are dealing with
print(class_names)
#load the final model
final_model = load_model('final_model_brain_tumor.h5')
#view an example image from the pred folder
folder = 'pred/'

#walk through inside the folder and select the first image
pred_folder_images = os.listdir(folder)
sample_img = pred_folder_images[0]

# Read in the image and plot it using matplotlib
img = mpimg.imread(folder + "/" + sample_img)

#plot the image
plt.imshow(img)
plt.axis(False)
plt.show()
```

```
#check the shape of the image
img.shape
def load_and_prep_image(target_filename,image_shape=64):

  #read an image from the pred folder
  image = tf.io.read_file(target_filename)

  #decode the read file into tensor and make sure that color channels
are 3
  image = tf.image.decode_image(image,channels=3)

  #resize the image
  image = tf.image.resize(image, size=[image_shape,image_shape])

  #scale the image
  image = image/255.
  return image
#pass the image into the function
brain_image = load_and_prep_image('pred/pred16.jpg')
brain_image
# Add an extra axis
print(f"Shape before new dimension: {brain_image.shape}")
brain_image = tf.expand_dims(brain_image, axis=0) # add an extra
dimension at axis 0
#steak = pred[tf.newaxis, ...] # alternative to the above, '...' is
short for 'every other dimension'
print(f"Shape after new dimension: {brain_image.shape}")
brain_image
#make a prediction
pred = model.predict(brain_image)
pred
#index the predicted class by rounding the prediction probability
pred_class = class_names[int(tf.round(pred)[0][0])]
pred_class
def pred_and_plot(model, filename, class_names):
  """
  Imports an image located at filename, makes a prediction on it with
  a trained model and plots the image with the predicted class as the
title.
  """
  # Import the target image and preprocess it
  img = load_and_prep_image(filename)

  # Make a prediction
  pred = model.predict(tf.expand_dims(img, axis=0))

  # Get the predicted class
  pred_class = class_names[int(tf.round(pred)[0][0])]

  # Plot the image and predicted class
  plt.imshow(img)
  plt.title(f"Prediction: {pred_class}")
  plt.axis(False);
pred_and_plot(model, 'pred/pred13.jpg', class_names)
pred_and_plot(model, 'pred/pred26.jpg' , class_names)
```

Table 7. Make predictions of the final CNN model.

```
#create the base model
img_rows, img_cols = 64,64

#load the pre-trained model using the imagenet database
vgg = vgg16.VGG16(weights = 'imagenet',
                  include_top = False,
                  input_shape = (img_rows, img_cols, 3))
#freeze the layers because they are already trained
for layer in vgg.layers:
    layer.trainable = False

#print the layers
for (i,layer) in enumerate(vgg.layers):
    print(str(i) + " "+ layer.__class__.__name__, layer.trainable)

#add some more layers to the vgg16 input model
def add_layers(bottom_model, num_classes):
      """creates the top or head of the model that will be
      placed ontop of the bottom layers"""

      top_model = bottom_model.output
      top_model = GlobalAveragePooling2D()(top_model)
      top_model = Dense(1024,activation='relu')(top_model)
      top_model = Dense(1024,activation='relu')(top_model)
      top_model = Dense(512,activation='relu')(top_model)
      top_model = Dense(num_classes,activation='sigmoid')(top_model)
      return top_model
num_classes = 1

#call the function and use as input model the pre-trained model vgg16
base_model = add_layers(vgg, num_classes)

model_vgg16 = Model(inputs = vgg.input, outputs = base_model)
#check the suammry of the model
model_vgg16.summary()
#compile the model
model_vgg16.compile(optimizer='adam',
                    loss='binary_crossentropy',
                    metrics=['accuracy'])
#fit the model
vgg16_history = model_vgg16.fit(train_data_norm,train_labels,
                                epochs=10,
                                batch_size=16,

validation_data=(test_data_norm,test_labels))
plot_loss_curves(vgg16_history)
probs_vgg16= model_vgg16.predict(test_data_norm)
probs[:3]
#create a pred list
preds_vgg16 = []
for element in probs_vgg16:
  if element > 0.5:
    preds_vgg16.append(1)
  else:
    preds_vgg16.append(0)
#check the first five predictions
preds_vgg16[:5]
```

```
#print the classification report
print(classification_report(test_labels,preds_vgg16))
#using the created function, the confusion matrix is displayed
confusion_matrix_calc(preds_vgg16,test_labels,2)
pred_and_plot(model_vgg16, 'pred/pred23.jpg', class_names)
```

Table 8. Build the VGG16 model and make predictions.

```
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Brain Tumor Classification Using Deep Learning</title>
    <link href="{{ url_for('static', filename='css/bootstrap.min.css')
}}" rel="stylesheet">
    <script src="{{ url_for('static', filename='js/jquery.min.js')
}}"></script>
    <script src="{{ url_for('static', filename='js/bootstrap.min.js')
}}"></script>
    <link href="{{ url_for('static', filename='css/test.css') }}"
rel="stylesheet">
    <script src="{{ url_for('static', filename='js/newjs.js') }}"
type="text/javascript"></script>
</head>

<body>
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
        <div class="container">
            <a class="navbar-brand" href="#">
                <img
src="https://images.twinkl.co.uk/tw1n/image/private/t_630/u/ux/screens
hot-2021-03-24-at-11.40.36_ver_1.png" alt="" width="60" height="50">
                Deep Learning For Brain Tumor Detection</a>
        </div>
    </nav>

    {% block content %}
    {% endblock %}

</body>


</html>

{% extends "import.html" %}
{% block content %}

<center>
    <br><h2>Brain Tumor Binary Classification Using Deep
Learning</h2><br>

    <p><i>Click the button to learn more about brain tumors.</i></p>
```

```
    <div style="text-align: center">
        <a href="https://en.wikipedia.org/wiki/Brain_tumor" class="btn
btn-outline-success" role="button">Click here!</a>
    </div><br><br>


    <form id="upload-file" method="post" enctype="multipart/form-
data">
         <input type="file" name="file" class="btn btn-primary"
id="imageUpload" accept=".png, .jpg, .jpeg">
    </form>


    <div class="image-section" style="display:none;">
         <img id="imagePreview"  class="img-responsive" src="#"
style="width:300px;height:300px;"/><br><br>
         <div>
            <button type="button" class="btn btn-info btn-lg "
id="btn-predict">Predict Image</button>
         </div>
    </div>


    <div class="loader" style="display:none;"></div>


    <h3 id="result">
        <span> </span>
    </h3>


</center><br><br>
{% endblock %}



h3{
    text-align: center;

}

form{
    text-align: center;
}

p{
    text-align: center;
    text-decoration: underline;
}


.responsive {
    width: 100%;
    height: auto;
  }

  .logo-image{
```

```
    width: 46px;
    height: 46px;
    border-radius: 50%;
    overflow: hidden;
    margin-top: -6px;
}
```

Table 9. Front-end code of the website.

```
import os
import tensorflow as tf
import numpy as np
from tensorflow.keras.preprocessing import image
from PIL import Image
import cv2
from keras.models import load_model
from flask import Flask, request, render_template
from werkzeug.utils import secure_filename


app = Flask(__name__)

#load the best model
model =load_model('final_model_brain_tumor.h5')


def get_classes(classes):
        if classes==0:
                return "No Brain Tumor"
        elif classes==1:
                return "Brain Tumor"

#pre-process the input image and make the prediction
def get_Result(img):
    image = cv2.imread(img)
    image = Image.fromarray(image, 'RGB')
    image = image.resize((64, 64))
    image = np.array(image)
    input_img = np.expand_dims(image, axis=0)
    result = model.predict(input_img)
    print(f'Model prediction: {result}')
    return result


@app.route('/', methods=['GET'])
def index():
    return render_template('index.html')

#user upload
@app.route('/predict', methods=['GET', 'POST'])
def upload():
    if request.method == 'POST':
```

```
        f = request.files['file']

        basepath = os.path.dirname(__file__)
        file_path = os.path.join(
            basepath, 'uploads', secure_filename(f.filename))
        f.save(file_path)
        value=get_Result(file_path)
        result=get_classes(value)
        return result
    return None


if __name__ == '__main__':
    app.run(host='localhost',port=5000,debug=True)
```

Table 10. Back-end code of the website.