University of Essex
Department of Mathematical Sciences

MA981: DISSERTATION

Brain Tumor Detection Using Machine Learning Algorithm

Dimpalben Vaghela
2204376

Supervisor: Guo Wenxing

August 24, 2023
Colchester

ABSTRACT

Nowadays, tumors rank as the second leading cause of cancer, posing a significant threat to a large number of patients. In the medical field, there is an urgent need for fast, automated, efficient, and reliable techniques to detect tumors, particularly brain tumors, as early detection plays a crucial role in treatment and patient outcomes. The ability to accurately detect tumors allows doctors to mitigate potential dangers and provide appropriate treatment, ultimately saving lives. Various image processing techniques are utilized in this application, enabling doctors to deliver precise and effective treatments, benefiting numerous tumor patients.

A tumor is essentially an abnormal mass of cells that grow uncontrollably. In the case of brain tumors, these cells consume essential nutrients that should nourish healthy brain tissue, leading to brain dysfunction. Currently, doctors manually locate the position and extent of brain tumors by examining MR images of the patient's brain. However, this approach can result in inaccurate tumor detection and is time-consuming. To address these challenges, deep learning architectures like Convolutional Neural Networks (CNN) , Support Vector machines (SVM), and specifically VGG 16 (Visual Geometry Group), InceptionV3, Resnet50, Trying with our own model with less number of total parameters Transfer Learning are employed for brain tumor detection. By utilizing these advanced neural network models, the system can predict whether a brain tumor is present in an image with high performance. If a tumor is detected, the model returns a positive response; otherwise, it provides a negative response. The implementation of these deep learning techniques offers a promising solution for swift and accurate brain tumor detection, allowing for timely interventions and improved patient outcomes. The use of CNN and Transfer Learning enhances the model's ability to make precise predictions and contributes to more efficient and reliable tumor detection in medical practice.

ACKNOWLEDGMENTS

I am deeply grateful to present this thesis, "Brain Tumor Detection using Machine Learning Algorithms," which is the result of extensive research.

First and foremost, I would like to express my heartfelt gratitude to my supervisor, Guo Wenxing. Without his invaluable guidance and mentorship, this research work would not have been possible. It has been an honor and privilege to work under his supervision. His expertise and continuous availability for monitoring and assisting me throughout this journey have been instrumental in shaping the outcomes of this research on brain tumor detection.

I extend my special appreciation to Supervisor Guo Wenxing for guiding me in the field of brain tumor detection research and offering constant support during the completion of my dissertation work. His encouragement and insights have been invaluable to the successful completion of this project.

Finally, I would like to acknowledge the University of Essex for providing a conducive environment and essential resources, including faculty guidance, that have been vital in realizing this research endeavor. Your assistance and encouragement have played a significant role in shaping this thesis.

I sincerely appreciate your contributions to our adventure.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 APPLICATION

Brain tumors are indeed a severe and complex health condition that affects both children and adults. Accurate detection and classification of brain tumors are crucial for providing appropriate care and planning treatment strategies. In recent years, advancements in technology, particularly in the fields of Machine Learning (ML) and Artificial Intelligence (AI), have shown great potential for improving the accuracy and efficiency of brain tumor diagnosis.

Magnetic Resonance Imaging (MRI) is widely regarded as the most reliable imaging technique for detecting brain tumors. MRI scans generate a large amount of image data, which can be challenging for radiologists to analyze manually due to the complexity and variability of brain tumors. This is where automated classification methods based on ML and AI, specifically utilizing Deep Learning algorithms, have demonstrated their superiority in accuracy and efficiency compared to manual categorization.

Deep Learning algorithms, such as Convolutional Neural Networks (CNNs) and Support Vector machines (SVMs), have shown remarkable capabilities in image analysis tasks, including medical imaging. CNNs, in particular, have been widely used for image classification and segmentation tasks. By leveraging the hierarchical features learned from vast amounts of labeled data, CNNs can automatically extract relevant patterns and features from MRI scans, enabling accurate tumor detection and classification.

Transfer Learning (TL) is another technique that can greatly enhance the performance

of deep learning models, especially in scenarios where the available labeled data is limited. TL allows pre-trained models, initially trained on large datasets, to be fine-tuned on smaller, domain-specific datasets such as brain MRI scans. By transferring the knowledge learned from the larger dataset, TL enables the model to achieve better generalization and accuracy on the specific task of brain tumour classification.

The effective machine learning technique Support Vector Machines (SVMs) is used to detect brain tumors. SVMs are capable of accurately classifying images of tumors and non-tumors from brain MRI scans. SVMs locate the best hyperplane to divide the two classes using the extracted characteristics from the scans. SVMs can differentiate between tumor characteristics and generalize to new data through training and optimization. SVMs are a useful tool for machine learning-based brain tumor identification because of their interpretability and adaptive ability.

A machine learning approach that can be used to find brain tumors is called the Random Forest method. The deep learning algorithms such as SVM, CNNs, and the TL system successfully classify brain MRI scans as cancerous or non-tumorous by using an ensemble of decision trees. To get precise and reliable forecasts, it combines the predictions of various trees. The management of high-dimensional data and the capture of complicated relationships in brain scans are two areas where this technology excels. Understanding the qualities that contribute to tumor diagnosis is made easier with the help of the Transfer Learning algorithm, which offers insightful information about feature importance. With dependable and understandable results, the Random Forest approach is a strong tool for machine learning-based brain tumor identification.

By proposing a system that integrates Deep Learning algorithms like CNN and SVM, doctors worldwide can benefit from improved brain tumor detection and classification. Such a system could automate the analysis of MRI scans, assisting radiologists in accurately identifying tumors and distinguishing between different types of brain tumors, including benign and malignant ones. This can lead to earlier diagnosis and more precise treatment planning, which ultimately contribute to extending patient lives and improving outcomes in the fight against brain tumors.

1.2 BRAIN TUMOR DETECTION SYSTEM

The brain is the most important and significant organ among the numerous that make up the human body. Brain tumours are one of the frequent causes of brain dysfunction. A tumour is nothing more than an accumulation of cells that are growing out of control. Brain failure occurs as a result of the growth of brain tumour cells, which eventually consume all the nutrition intended for healthy cells and tissues. Currently, clinicians manually examine the patient's MR images of the brain to determine the patient's brain tumor's location and size. This takes a lot of time and leads to an inaccuracy in the tumor's detection. Brain cancer is a serious condition that claims many lives every year. In order to diagnose brain tumours early on, a detection and classification method is available. The most difficult jobs in clinical diagnosis are those involving cancer categorization. This project focuses on such a system that analyses MRI pictures of various patients to identify tumour blocks and classify the type of tumour using a convolution neural network algorithm. The detection of brain tumours in the MRI images of cancer patients uses a variety of image processing techniques, including picture segmentation, image enhancement, and feature extraction.

The four stages of image pre-processing, image segmentation, feature extraction, and classification are involved in the detection of brain tumours utilising image processing techniques.

The performance of detecting and categorising brain tumours in MRI images is improved by using image processing and neural network approaches.

1.3 OVERVIEW OF BRAIN AND BRAIN TUMOR

The brain is the main component of the human neurological system. It is situated in the human head, and the skull protects it. All of the body's components are controlled by the human brain. It is a specific sort of organ that enables people to adapt to and endure all environmental conditions. Humans can behave and communicate their thoughts and feelings thanks to their brains. In this section, we discuss how the brain is organised to comprehend the most fundamental concepts [10].

Primary brain tumours (benign tumours) and secondary brain tumours (malignant tumours) are the two basic categories into which brain tumours are divided.Gliomas are a

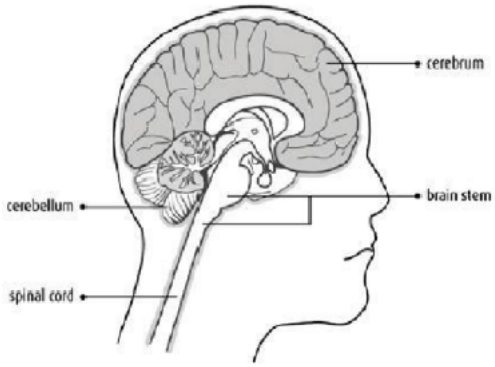

Figure 1.1: Basic Structure of human brain [11]

form of brain tumour that is a benign tumour and grows slowly in the brain. It comes from astrocytes, which are brain cells that are not neurons. Primary tumours are often less aggressive, but they put a lot of strain on the brain, which causes it to malfunction [12]. Secondary tumours are more aggressive and spread into adjacent tissues more quickly. Secondary brain tumours develop from other body parts. These tumours are caused by metastatic cancer cells that have travelled to various body parts, such as the brain and lungs. A secondary brain tumour is extremely dangerous. Lung, kidney, bladder, and other cancers are the primary causes of secondary brain tumours [13].

1.4 MAGNETIC RESONANCE IMAGING (MRI)

In 1969, Raymond v. Damadian created the initial magnetic picture. The most advanced technology, the first human body MRI, was created in 1977. We can see the inside features of the brain thanks to MRI, and by doing so, we can see the various kinds of tissues in the human body. When compared to other medical imaging methods like X-ray and computer tomography, MRI images are of higher quality.[14]. MRI is a useful tool for identifying brain tumours in humans. For mapping tumor-induced change, various MRI images, such as T1 weighted, T2 weighted, and FLAIR (Fluid attenuated inversion recovery) weighted images, are available.

T1 and T2 weighted MRI sequences are the most popular. Bright FAT makes up the sole type of tissue in T1 weighted, while Bright FAT and Water make up both of the two categories of tissue in T2. Repetition time (TR) is low when T1 weighting is used, whereas TE and TR are long when T2 weighting is used. The pulse sequence parameter known as

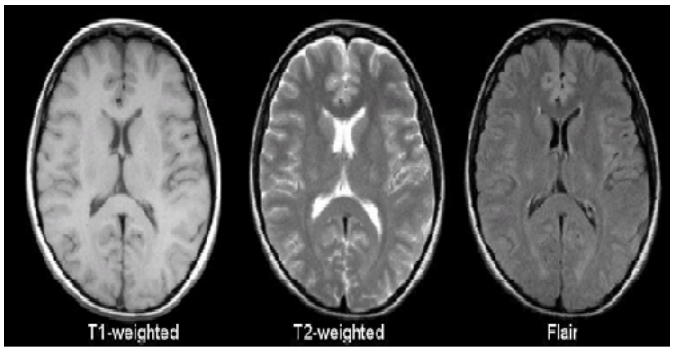

Figure 1.2: T1, T2 and Flair image [15]

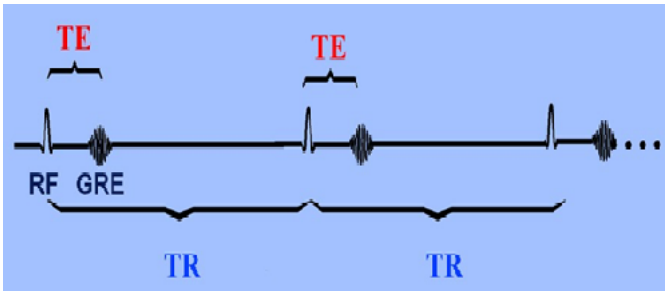

Figure 1.3: Graph of TE and TR [16]

TE and TR stands for repetition time and time to echo, respectively, and is measured in milliseconds (ms)[15]. The echo time is depicted in the image as the interval between the centre of the RF pulse and the centre of the echo, and TR is the interval between the TE repeating sequence of pulse and echo.

The third frequently used FLAIR sequence. The Flair sequence resembles a T2-weighted image almost exactly. The length of the TE and TR times is the only distinction. The table displays their approximate TR and TE times.

	TR (msec)	TE (msec)
T1-Weighted (short TR and TE)	500	14
T2-Weighted (long TR and TE)	4000	90
Flair (very long TR and TE	9000	114

Figure 1.4: Table of TR and TE time [15]

1.5 OBJECTIVE

- Develop efficient software to aid doctors in identifying brain tumors and their causes.
- Reduce patient waiting time for diagnosis and treatment.
- Facilitate early detection of brain tumors, enabling timely intervention and treatment.
- Provide patients with timely and appropriate medical consultations based on accurate tumor detection results.

These objectives highlight the importance of leveraging machine learning technology to improve brain tumor detection, benefit both medical practitioners and patients, and ultimately contribute to better healthcare outcomes. Good luck with your research and implementation!

1.6 THESIS ORGANIZATION

Chapter I: Introduction: problem statement, aim of study, and Research objective. Chapter II: Literature review, in which several previous studies are presented and explained in detail form. Chapter III: Methodology and CNN Algorithm, And the Proposed Method. Chapter IV: Details of the implementation of the experiment and the consequences gained for all planned situations and comparison of results. Chapter V: Conclusion and Future Work

CHAPTER 2

LITERATURE SURVEY

2.1 INTRODUCTION

Since the 1920s, numerous epidemiological studies looking into the occurrence of open-angle brain tumours, as described in [9], have been carried out on various continents. The terms "brain tumour" and "ocular hypertension" were frequently used interchangeably in older research, which violated a number of methodological requirements such failing to specify the diagnostic criteria for the disease [10]. The Welsh Brain Survey was one of the earliest studies to be praised for having been properly planned and carried out [11]. Only those aged 40 to 75 were included in the study, which investigated 92% of the town's eligible population.

According to the mechanical disease theory in [13], which is summarised in [14], when intraocular pressure exceeds physiological levels, the pressure gradient across the lamina cribrosa increases, causing pressure-induced backward bowing of the lamina cribrosa, misalignment of the pores within the connective tissue complex, and subsequent deformation and mechanical stress to the brain ganglion cells, their axons, and supporting cells. Both directly and indirectly, the mechanical stress brought on by this increased pressure damages brain ganglion cells. It has been demonstrated that this stress obstructs the vascular nerve's anterograde and retrograde axoplasmic flow. According to [15], the capacity of brain ganglion cells to endure and thrive can be viewed as a dynamic equilibrium between several stimuli supporting their survival and other aspects encouraging apoptosis and consequent extinction.

The move additionally, these changes would promote apoptosis in brain ganglion cells. The aberrant deposition of several extracellular matrix proteins, including collagen, elastin, and basement membrane material, can also be brought on by elevated interracial pressure (IOP), and this might alter the bio-mechanics. According to [17], some of the alterations that lead to brain tumour vessels neuropathy are caused by the possibly biochemical environment of the vessels nerve.

The genetics of brain tumour has been studied extensively using a binary or qualitative trait approach. There are various issues with this strategy. The first is the absence of a common definition for the extremely diverse illness that is concealed by the lone term "brain tumour." the several epidemiological studies that have been published that have different definitions, as explained in [18]. This is made worse by the many ways that the visual field, vascular disc, and intraocular pressure (IOP) are assessed, as well as the diagnostic standards used to identify brain vessel neuropathy, visual field abnormalities, and "normal" intraocular pressure. A literature review was done to more properly investigate this.

The genetics of brain tumour has been studied extensively using a binary or qualitative trait approach. There are various issues with this strategy. The first is the absence of a common definition for the extremely diverse illness that is concealed by the lone term "brain tumour." the several epidemiological studies that have been published that have different definitions, as explained in [18]. This is made worse by the many ways that the visual field, vascular disc, and intraocular pressure (IOP) are assessed, as well as the diagnostic standards used to identify brain vessel neuropathy, visual field abnormalities, and "normal" intraocular pressure. A literature review was done to more properly investigate this.

By displaying the prevalence of open-angle brain tumour in Rotterdam, we were able to highlight the significance of having a common definition for this condition.could differ by a factor of 12 (from 0.1 to 1.2%) based on how an open-angle brain example is handled.Different descriptions of the sickness were used to define the tumour.other significant population-based studies, as mentioned in [19].

The increasing incidence of brain tumours is the second factor. According to [20], the incidence is only 1% to 4% among white Caucasian people over the age of 50. Although this prevalence is adequate to classify a brain tumour as a common disease, if association methods are to be employed, a significant amount of work still needs to be made in order to locate a sufficient number of cases for a useful genome-wide association research. A

persistent brain tumour also develops slowly over time.

As a result, it can be difficult to locate pedigrees that are large enough for linkage analysis, as stated in [20]. Instead than focusing on the disease itself, one way to get around some of these problems is to break it down into contributory factors in the form of (QT) of Quantitative Traits. Generally speaking, a quantitative characteristic, or QT one that has a continuous distribution, like blood pressure, as opposed to a distinct characteristic, like blood type. Three broad categories can be used to categorise QTs. Numerous clinically significant characteristics, including weight, blood pressure, axial length and intraocular pressure. These characteristics can have any number of values, on a continuous scale, measured.

Meristic characteristics, such as egg production and bristle number, are measured by counting and take the form of integers. Last but not least, threshold traits are qualities that, although they appear to be binary traits on the surface, actually indicate an underlying continuous liability caused by the interaction of hereditary and environmental factors, according to [22]. When a certain point is reached, the defining characteristic emerges. The pathophysiology of the majority of common illnesses, including brain tumours, is probably described by this paradigm.

In [25], Hein Tun Zaw employs naïve Bayesian classification to pinpoint the tumour spot that houses all of the malignant tissue that has spread across the body. This research made use of Naive Bayes classifier-based scoring algorithms, preprocessing, morphological operations, pixel extraction, maximum entropy thresholding, and brain MRI databases. This technique seeks to identify a tumour area on several MRI scans of the brain and forecast the presence of a tumour in the identified region. An notable benefit of this technology is that, in contrast to other techniques, it enables accurate tumour detection in many regions of the brain, including the central area (at the level of the eyes).The technique suggested in [26] aims to categorise images of brain tumours, and it has been evaluated on 50 MRI scans with a success rate of 81.25% for tumour images and 100% for non-tumor images, with an overall accuracy of 94%. employing a convolutional network, meningiomas, gliomas, and pituitary glands were divided into three subtypes. a vector machine (VM) and a neural network (CNN). the photos from the dataset. The file has been shrunk to decrease computation and salty noise has been added to make the dataset bigger and the model more reliable. The comparison of the performance was carried out in Python using the Tensorflow and Google Colab platforms.

Particle swarm optimisation (PSO) technique is used for feature selection in [27] by Arun Kumar et al., who import MRI images of brain tumours into an online database and create a machine learning model. The Support Vector Machine (SVM) Classifier is the following. utilised to categorise the kind of tumour that is now in the brain.

Adela Kermietal [28] suggested leveraging the brain's similarities in 3D MRI and conventional analysis to segment brain tumours. Noise is removed during pre-processing of the image. Methods for symmetry analysis are effective and unpredictable. This approach automates the tumour identification process. The tumor's volume, shape, and matching margins are all calculated using a deformable 3D model based on the corrected geodetic plane. For detecting and segmenting tumours, the calculation time is typically 5 minutes. The results showed a precision and sensitivity of 38.04% and 89.01%, respectively.

V. Anita and S. Murugavalli [29] presented a precise, algorithm-based review methodology. The suggested approach was used to extract the characteristics and filtering coefficients of the outcomes after applying the discrete wavelet transform, and a self-organizing neural network map was trained using a nearby ANN. There were two stages to the testing process.

In comparison to SVM-based classification approaches, this technique is more effective and produces superior results. The outcomes demonstrate that this study can be included into image classification and computer recognition applications for medical imaging. The sensitivity and accuracy of this investigation were both 100%. A data mining technique was put out by Parveen and Iitpalsingh [30] to categorise magnetic resonance pictures. Preprocessing, separation, feature extraction, and classification are the four processes in the classification process. To improve speed and accuracy, the first phase entails segmenting the skull and improving it. Fuzzy sets are employed during the segmentation stage. A grayscale array is used to extract photographic details from MRI scans. Implementing data volume management (DVM) for image categorization is the last step. The end findings of this study showed that classifying magnetic resonance images may be done with great accuracy and efficiency. 91.24% was found to be the accuracy.

To acquire a thorough structural map and calculate operational revenue, Daniele Ravi et al. suggest innovative processing and size reduction techniques [31]. This approach involves two steps. The organisation is then categorised after performing a stochastic neighbourhood operation with a semantic segmentation technique utilising a semantic textile forest from which the T distribution flows. In this investigation, accuracy and precision were 81.90%

and 80.91%, respectively [32].

A brand-new algorithm for modelling glioblastoma was revealed by Lamia Salemi et al. [32]. Rapid diffusion mapping based on global pixel data is used to generate the tumor's remote region. The new method evaluates cancer growth over time by utilising Rapid technology. This technique doesn't need a lot of training and has an optimised execution time of less than 0.5 seconds per frame. When compared to safe cells, glioblastoma's effectiveness displays a range of grey areas. The two regions of the brain image are separated using this information. Then, regions of glioblastomas are contrasted with a predicted model based on density levels. The suggested system eliminates tumours instantly and predicts growth with an accuracy of 89.74Four phases of MRI image processing were proposed by Mukambika and Uma [33]. Pre-processing, segmentation, pattern extraction, and pattern recognition are the steps in this procedure.

These are the preliminary processes, and the double threshold approach is used to extract the brain from the MRI. The first method for removing brain tumours from brain MRI images was created using a nonparametric active contour deformable model. K-means clustering is the second technique. Besides segmentation, a two-step decision-making process is offered. In the initial stage of feature extraction, discrete wavelet transforms, grayscale matching matrices, and a support vector machine (SVM) are used. Classification.

Multiple patients' T2-enhanced magnetic resonance imaging data for brain tumours. There are 24 pictures of dangerous tumours and 17 pictures of benign tumours. With accuracy of 94.12% and 82.35%, respectively, level-corrected SVM segmentation and mean K values identify pictures as benign, normal, or malignant brains. Segmentation does not perform as well as the established K-Average level.

K. Sudharani and others in the matter. Techniques like the histogram, resampling, K-NN algorithm, and distance matrix are additional to the suggested methods [34]. The histogram first displays the total number of pixels in an image. 629x839 is the image's reduced size for accurate geometric representation. Use the applicable ANN algorithm to categorise and identify brain tumours after modifying parameter K. The classification was done using the Manhattan distance criterion. 48 photos make up the dataset that was used to test the method. The average score for every image analysed is 95%.

Razel Ahmed et al. proposed approaches that use artificial neural networks (ANN) for image preprocessing, segmentation, feature extraction, SVM classification, and tumour stag-

ing [35]. The Temper based K-means and modified Fuzzy C-means (TKFCM) technique was utilised in this work. This hybrid algorithm combines fuzzy c-means, k-means, and segmentation.

Primary and secondary statistics are derived based on area features for feature extraction. The SVM then categorises MRI scans of the brain as malignant or healthy.The ANN classifier is used to categorise the stage of brain tumours. The proposed approach had a 97.44% classification success rate.

2.2 KINDS OF BRAIN TUMOR

Astrocytomas are tumours that develop from astrocytes, the star-shaped cells that make up the brain's supporting or "glue-like" tissue. Depending on how normal or aberrant the cells seem, these tumours are staged on a scale from I to IV. Astrocytomas can be classified as low-grade or high-grade. Low-grade astrocytomas typically develop locally and expand slowly. High-grade astrocytomas develop quickly and need a distinct approach to therapy [7].

Anaplastic astrocytomas are Stage III tumours, as well. Treatment for these uncommon tumours must be more vigorous than for benign pilocytic astrocytoma. Anaplastic astrocytomas frequently develop tentacle-like extensions that encroach on nearby tissue, making full surgical removal of them challenging [8]. Additionally, glial and neuronal cells can both be seen in gangliogliomas. Adults only extremely seldom develop these tumours. They can usually be treated with surgery alone or radiation therapy because they are often slow-growing (Stage I) tumours [9,10].

2.3 ARTIFICIAL INTELLIGENCE

In 1956, the phrase "artificial intelligence" first arose. Today, artificial intelligence is frequently utilised to enhance data volume, data processing complexity, and the advancement of data and information technologies. In the 1950s, early research on artificial intelligence concentrated on issues like symbolic approaches and problem-solving. The US Department of Defence developed an interest in these AI techniques in the 1960s and started training computers to mimicr fundamental human thought processes.

Defence Advanced Research Projects Agency (DARPA), for instance, created a roadmap

for the 1970s. Through information systems, decision support, and intelligent sensing systems human ability this study has created potential to supplement and improve the formal and automated thinking present in modern computing. Artificial intelligence is now being developed to provide numerous advantages in a variety of industries, including the military, business, and healthcare.[36]

Significant medical advancements have recently been made thanks to artificial intelligence technology. AI has the potential to replace human reasoning in some functional areas of medicine (like radiography) and will eventually assist clinicians in making well-informed therapeutic judgements. The successful application of artificial intelligence in the healthcare industry has been made possible in recent years thanks to the expanding availability of medical data and the quick uptake of big data analytics [37].

2.3.1 COMPUTER VISION

In the field of artificial intelligence known as computer vision, items that can be seen in the real world are deciphered and understood. Although it is regarded as a sub-branch of artificial intelligence and has been for a while, it is becoming more significant in particular industries.

The Summer Vision Project, published by Seymour Papert and Marvin Minsky, introduced computer vision as a method for characterising objects in 1966 [38]. A computer vision system suggested by Kunihiko Fukushima, which was given significant consideration in 1979 [39], was used to conduct the following study. Neocognitron served as inspiration for the development of Convolutional Neural Networks (CNN), which have lately found success in application [40]. With the dazzling triumph of AlexNet, which ESA created as part of the ImagetNet image recognition competition [41], successful deep learning results have brought computer vision to our attention.

There are three fundamental steps in computer vision. These three processes include picture capture, image processing, and image comprehension. Image segmentation, object identification, face recognition, edge recognition, pattern recognition, image classification, and feature processing are some examples of computer vision types. Applications for computer vision are widely employed in the real world. Examples include the usage of facial recognition on Facebook, target identification in the military, transportation, driverless cars in businesses like Tesla and Google, facial recognition technology in security applications,

and automatic payment processes in contemporary retail stores.

2.3.2 MACHINE LEARNING

ML is a branch of AI that focuses on creating products that comprehend data and improve accuracy during a time when no software development is taking place. A process in information knowledge is a series of operations used in numerical handling. In order to train algorithms and generate decisions and forecasts based on fresh information, machine learning (ML) utilises a lot of data. improved algorithms, data processing, precise prediction, and decision-making. Building a machine learning application (or model) involves four key processes. Typically, a scientist will carry out this task in collaboration with a model specialist. The distinction between machine learning algorithms, so-called "classic" or "traditional" algorithms, and non-machine learning AI algorithms may be better understood by using this straightforward example. Consider a person who has no familiarity with animals. This young toddler is unable to discriminate between a dog and a cat. Think of a youngster in this thought experiment as an artificial intelligence algorithm, and a computer scientist researching the differences between the two [42]. In traditional AI applications, rather than teaching the baby the distinction between a cat and a dog, we instead provide him a whole set of instructions that will allow him to make the distinction. These instructions ought to be written in such a way that this young child can follow them and, with some help, tell the difference between a cat and a dog. It is clear how challenging this is for the teacher and the infant alike.

However, as you may have seen, we don't need to do this to improve the baby's chances of success. Instead, if we show the infant numerous images of cats and dogs and continually point out which is which, the baby will eventually start to tell the two apart accurately. As the name implies, this is exactly how ML algorithms function. Unluckily, learning algorithms are not pre-installed on computers when they are created. For each problem, computer scientists create and modify the best machine learning algorithm. This reasoning is the foundation of machine learning.

Recently, ML applications have been used in various artificial intelligence applications. For the majority of artificial intelligence applications we see in science fiction films, machine learning is essential [43]. There is still one more step we need to take, even if we have largely avoided conceptual ambiguity by highlighting the differences between the notions of ML

and AI: Machine learning is a method field that, despite being a more limited idea than artificial intelligence, nonetheless covers a wide variety of algorithms. Because of this, the term "machine learning" still falls short of fully describing the popularisation goods we're discussing. Due to this, discussing deep learning, a branch of machine learning, can help us avoid conceptual misunderstandings on a broadcast-level and better comprehend why artificial intelligence has become so popular over the past 10 years. Based on the training data that is currently available, classification and regression determine which class the new data will be included in. In other words, the classes that will be produced as a result of the classification process are already known. What kinds of data characteristics are employed as training data for machine learning, furthermore? The method of data mining that is most frequently employed is classification and regression.

The primary distinction between classification and regression is whether the linked variable is projected to be categorical or continuous. Regression and classification have converged more recently in studies. Regression and classification strategies both employ comparable algorithms. For instance, a training data set can be constructed based on the blood types, ages, and genders of patients with a certain illness. It is also possible to identify if the new patient has this condition. A business can develop a training data set by investigating the effects of an advertisement on sales, as well as the impact of the commercial's timing, length, and visuals. The company will avoid wasting money on superfluous advertising by using this training data set to teach its executives how to promote a new product [44].

2.3.3 TRAINING AND TESTING DATA SELECTION

From the Kaggle website I collected data and performed training, which represents the data the machine learning prototype will utilise to address the problem. In various cases, the training data is "tagged" to identify the categories and elements. That the model should be able to recognise. Additional information is not labelled, and the prototype should eliminate these characteristics and categorise them according to its unique [45].

2.3.4 CHOOSING THE ALGORITHM TO RUN ON THE TRAINING DATASET

A sequence of numerical handling measures make up the algorithm that will be applied. Whether there are labels on the data or not, how many data points are in the training dataset, and what kind of issue needs to be solved all affect the method [45].

2.3.5 USING THE ALGORITHM IN TRAINING TO BUILD THE MODEL

An iterative procedure is used to train the algorithm. In order to create a more accurate result, this method comprises running the variables through the algorithm, comparing them with the results, and then rerunning the variables with modified weights and labels [45].

2.3.6 MODEL APPLICATION AND FURTHER DEVELOPMENT

The model will then be tested using the updated data and will eventually become more accurate and efficient. Depending on the issue to be handled, new data must be chosen.

2.3.7 SUPERVISED LEARNING

The system is trained using labelled data in supervised machine learning. Each data point is assigned a label that divides it into one or more categories, such as "X" or "Y". This data, also known as training data, helps the system organise itself. In order to forecast new categories of data or "test" data, the knowledge from the training outcomes is used [46].

2.3.8 ARTIFICIAL NEURAL NETWORK (ANN)

Artificial neural networks (ANN) are a branch of artificial intelligence technology. A computer system called ANN aims to fully realise the capabilities of the human brain on its own. Traditional programming techniques are frequently insufficient for these systems. When creating artificial neural networks, he drew inspiration from biological neural networks in humans and animals. Information is transmitted by neurons in the human brain using electrical signals that are sent from one cell to another.

The brain's memory can store the transmitted signal. In other words, the computer's memory, which is used to retain information, is modelled after the brain of a biological system. The goal is the same, but a different mechanism is employed. In a nutshell, ANN is a type of data processing technique for computers that is intended for one-off events [47].

ANNs, like CNNs, have gained popularity in recent years for a variety of image-related challenges. The numerous advancements in the sector are probably to blame for the current rise in CNNs. In the 2016 ImageNet competition, AlexNet bested all other approaches, having a top-5 error 10.8 percentage points lower than the runner-up [47]. Following this, CNN's new architecture took first place in the annual competition. Through several papers and competitions, CNN has proven to be the best at image-related tasks, particularly categorization. Thanks to ideas like the ReLU, batch normalisation, skip connections, and others, these models are getting deeper. Greater depth enables more sophisticated internal representations.

The development of GPUs has therefore been crucial in handling the growing workload of training ever-larger models. The classification of a health professional is important in medicine for activities like brain tumour diagnosis. The job may involve manually separating different portions of the image or looking for certain patterns in the image that represent symptoms. CNNs have been demonstrated to perform effectively on certain tasks. Compared to many other systems, including traditional ANNs, very few design choices or presumptions in CNNs are made by humans. The majority of human assumptions, which may be the most challenging aspect of creating a model in complex scenarios, are avoided because CNNs are their feature extractors.

The dataset will be used to build the feature extractors and classifier for the model. This indicates that, as stated in [47], it is crucial to make the dataset representative of the domain it should handle. It is essential to have big datasets with variation that span the entire domain. The medical industry frequently deals with enormous amounts of data, image evaluation can be challenging, and objectivity is desired. These are the elements that CNNs excel at. It should be highlighted that despite the abundance of data, getting permission to utilise it might be challenging. ANN can still do some jobs with less data.

2.4 IDENTIFICATION OF RESEARCH

Finding studies on using deep learning to locate brain tumours was a major emphasis of the review. The search also concentrated on documents utilising the largest publicly available dataset on brain tumours from the Kaggle repository in order to compare approaches. Due to this, two search strings were used: one to identify healthy versus brain tumour patients using more basic phrases, and the other to find pertinent research performed on the dataset.

Google Scholar was used as the search engine. This search engine is used to look up scientific papers across different databases. Finding documents that have cited a particular article is one of the functions of Google Scholar. Although there was no official procedure in place, a similar strategy was applied in the preliminary research. The project's initial goal was to investigate methods in the field in order to learn what works when identifying glaucoma. In addition to using Google Scholar, recommendations from teachers and other students were also taken into consideration. To create search strings, two tables with groups of terms were put together. Words that are semantically synonymous or have a similar meaning are included in each group. In summary, the author in [47] achieved excellent classification results for brain tumours using 2D convolution and a VGG architecture. The author offers a novel approach to 3D convolution that performs better than other established techniques in its field. Finally, the author discusses a method for CNN visualisation. Exciting elements that can be employed in the papers will be emphasised in all of these works. This covers techniques, architectural decisions, performance research, and tools.

Source	**Dataset**	**Techniques**	**Performance**
[58]	BRATS 2015 BRATS-MICCAI	87837	787
[59]	https://radiopaedia.org	K-means and FCM	ACC 56.4%
[60]	BRATS	Patch based k-means with FCM	SI 91%
[62]	collected by authors	Bi-secting	ACC 83.05%
[63]	BRATS	Force Clustering	-
[62]	BRATS 2017	Random	DSC 62.5%

Table 2.1: Summary table for show previous study work.

CHAPTER 3

METHODOLOGY

3.1 INTRODUCTION

Pre-processing, which I performed using the CNN method, is frequently used to lower image contrast, undesirable noise, and brightness. For the pre-processing stage, the image is converted to grayscale, black and white, histogram equalisation, and thresholding. Using MATLAB software and a custom-built GUI, thus The most straightforward approach involves data augmentation, which involves adding predetermined cases to the training dataset, using a Meadin filter in the Convolution layer and one hidden layer, as well as 16 input neurons. Hyperparameters are boundaries whose values determine the upsides of model boundaries that a learning calculation ends up learning for final decision making based on predetermined parameters for testing the samples. They also regulate the algorithm training in terms of learning experience. The prefix 'hyper' suggests that they are 'high level' boundaries that govern the educational experience and the model boundaries that result from it, so In this study, it is challenging to gather raw data. Working with raw data presents additional challenges because of the multiple repeating rows and erroneous statistics that are not representative of reality. As a result, in order to create a dataset that the researcher can use, raw data must be filtered and modified. Duplicate rows are similarly eliminated from our data. Synthetic values have been added to the sets in order to better reflect the results.

3.2 METHODOLOGY OVERVIEW

In order to detect brain tumour disease, features must be extracted from recurrent images. A flowchart would show the many stages, which would improve understanding of the steps. This strategy will fill a research hole where there is now one model for each chronic illness image. The goal is to develop, deploy, and maintain a large-scale expert system for brain tumour disease. Networks, particularly deep learning methods, could easily develop issues.ăThe benchmark dataset used in this study is more recent and has recently become available to academics. The Kaggle Dataset [48] will be used with a convolutional neural network on MATLAB R2019b. The size and number of layers of this convolutional neural network will be the same as those of the previous one. They will employ their threshold formula, but it will be altered by the addition of a parameter. The optimisation process is applied to this parameter. The activation function and optimisation method choices, for example, are particular to this work. The accuracy of feature selection parameters will also be predicted utilising deep learning in this study employing a feature selection mechanism. This study will enable a more precise assessment. This study will provide a more precise assessment of the convolutional neural network method's performance on smaller feature subsets and a more fair comparison of brain tumour disease detection and classification using a more advanced deep learning approach.

The goal of this thesis is to identify and categorise brain tumours using CNN to speed up the diagnosing process. It accomplishes this in a two-step, semi-supervised manner. The model initially trains what they refer to as the feature extractor of the model on 2D slices of the brain tumour scans, and in a subsequent step trains the classification part of the model using the previously learned features on whole brain tumour images. CNN is computationally taxing, but in contrast to active gadgets, they don't require physical contact and are more approachable. This research also proposes a way for comparison patterns to improve their robustness to the classification of provided comparison instances. Deep learning-based approaches focus on recording the natural user movements without needing users to wear or hold any sensor. Despite being computationally taxing, CNN is benign compared to active devices and doesn't necessitate physical contact. This study also proposes a way for comparison patterns to improve their robustness to the categorization of provided comparison instances. Deep learning-based approaches focus on observing the natural user movements

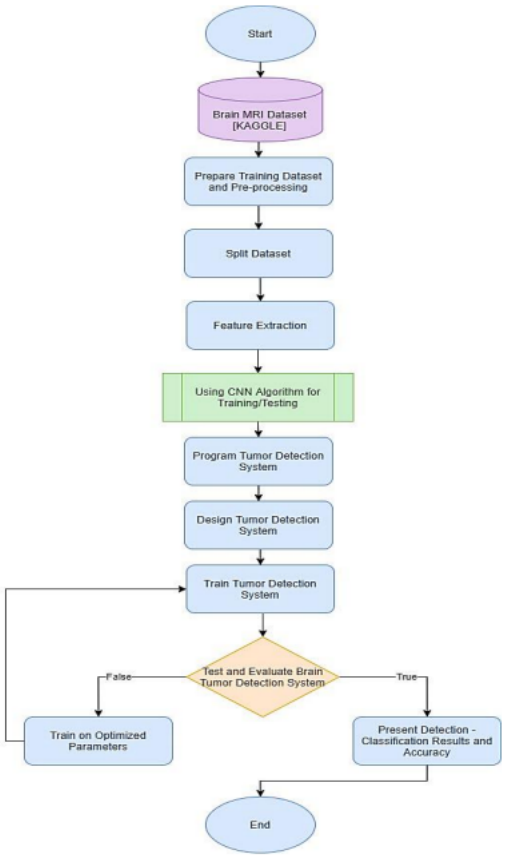

Figure 3.1: Flowchart shows an overall approach being followed

without needing users to wear or hold any sensor for series classification.

3.3 DATASET

Brain tumors present a complex and diverse challenge due to their varying sizes and locations within the brain. Understanding the nature of these tumors can be highly intricate. For accurate analysis, qualified neurosurgeons are required, making the process time-consuming and difficult, especially in regions with limited access to skilled medical professionals.

To address this issue, a cloud-based automated solution can provide a viable resolution. By leveraging cloud-based technology, data from MRI scans can be efficiently processed and analyzed without the need for on-site medical expertise. This approach can significantly enhance the accessibility and availability of tumor analysis, particularly in underserved regions.

For this project, a dataset containing MRI images has been utilized for training and test-

ing the automated brain tumor classification model. The dataset consists of four categories: glioma$_tumor, meningioma_tumor, no_tumor, and pituitary_tumor. Each category contains a varying number of f$

The training dataset comprises four directories with a substantial number of MRI images for each tumor category. Similarly, the testing dataset contains images for evaluation and validation purposes.

The utilization of this comprehensive dataset, available at the provided link, enables the development of a robust machine learning model for classifying brain tumors based on MRI images. By leveraging cloud-based capabilities and automated analysis, this project holds the potential to facilitate accurate and rapid brain tumor identification, ultimately contributing to improved diagnosis and treatment outcomes. Furthermore, the cloud-based solution can bridge the gap in medical expertise and resources in regions with limited access to skilled professionals, thereby improving healthcare accessibility and quality for patients affected by brain tumors.

Due to their different sizes and locations inside the brain, brain tumours pose a complicated and wide-ranging challenge. It can be very challenging to comprehend how these tumours function. Qualified neurosurgeons are necessary for correct analysis, which makes the process time-consuming and challenging, especially in areas with limited access to experienced medical personnel.

An automated cloud-based approach can offer a workable solution to this problem. Data from MRI scans can be effectively processed and analysed using cloud-based technologies without the requirement for on-site medical knowledge. This strategy can greatly improve tumour analysis's availability and accessibility, especially in underserved areas.

The automated brain tumour classification algorithm for this research has been trained and tested using a dataset of MRI images. The dataset includes.

Four directories altogether make up our testing dataset: -

glioma tumor(100 files), meningioma tumor(115 files)

no tumor(105 files),pituitary tumor(74 files)

Four directories altogether make up our training dataset:-

glioma tumor(826 files), meningioma tumor(822 files)

no tumor(395 files), pituitary tumor(827 files)

This is the dataset link which I have used for training and testing:

https://www.kaggle.com/datasets/sartajbhuvaji/brain-tumor-classificationmri?resource=downlo

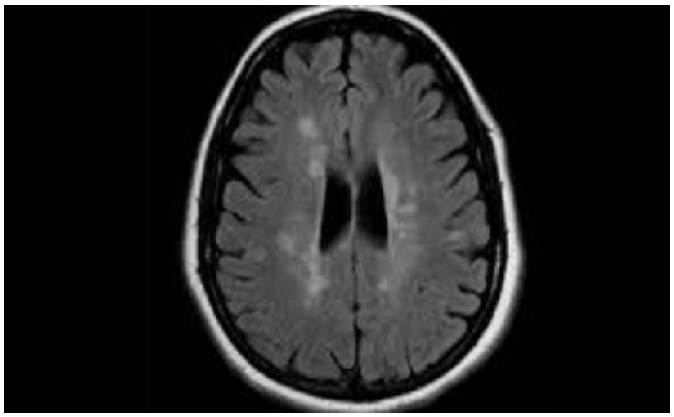

Figure 3.2: A sample of brain image being acquired from Kaggle

3.4 CONVOLUTIONAL NEURAL NETWORK (CNN)

The Convolutional neural network (CNN) has been operating because this closely resembles the amount of time needed for a brain input to come. This method should therefore rely on feedforward data processing. Cascade. When it was revealed that the exact recognition selective response only showed at 200ms after input, it was determined that additional recurrent processing was required in the brain to handle occlusion. Recurrent networks outperformed feedforward networks when trained to do comparison recognition of specific samples, according to deep learning researchers. The study mentioned above served as the basis for this thesis, which explores the usage of recurrent models to recognise provided similarities. Hyperparameters are boundaries whose values determine the upsides of model boundaries that a learning calculation ends up learning for final decision making based on predetermined parameters for testing the samples. They also regulate the algorithm training in terms of learning experience. The prefix 'hyper' suggests that they are 'high level' boundaries that govern the educational experience and the model boundaries that result from it.

The CNN is built on neural networks, and for imaging, the basic feedforward architecture was selected. Five additional models were built in addition to the initial network that was pre-trained on the Image network and used as a control. On the other hand, recurrent and feedforward connections would suggest a broader search space, but these do not seem to be particularly pertinent. That the recurrent layers provide too much search space, leading the models to overestimate their computational search regions and lose any utility they may have, is one explanation for this strange observation. Although reviewing the training

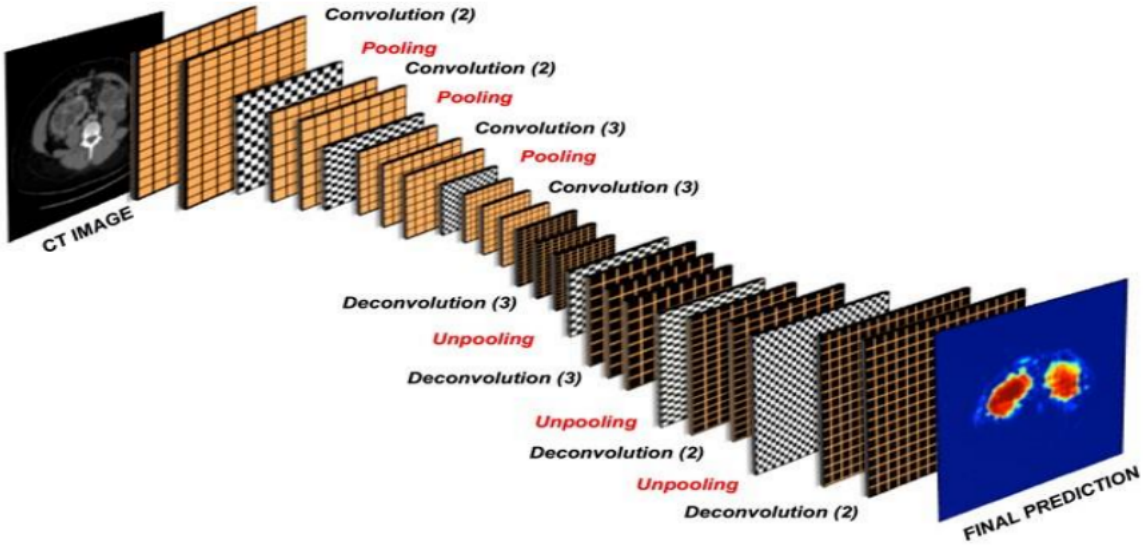

Figure 3.3: An architectural diagram of CNN[53]

evaluation can be helpful, the fact that fewer training losses are still linked to greater testing performance prevents any inferences from being made.

The CNN seeks to better comprehend how the training of the models varied and what effect this would have had on the observed result. Along with the performance of the image networks, for instance, visibility of about 90%, the training loss was also visible. This occlusion was chosen because it was in the area where the performance of the networks fluctuated the most. The researchers showed that not all image networks were equally capable of incorporating the additional input. It is clear that imaging networks give better information than feed forward networks. The goal of this thesis is to identify and categorise brain tumours using CNN to speed up the diagnosing process. It accomplishes this in a two-step, semi-supervised manner. The model initially trains what they refer to as the feature extractor of the model on 2D slices of the brain tumour scans, and in a subsequent step trains the classification part of the model using the previously learned features on whole brain tumour images.

The simplest approach is to employ data augmentation and increase the training dataset with the desired examples. Additionally, deep learning researchers have developed more sophisticated modular systems that explicitly incorporate numerous subsystems in charge of functions like segmentation, depth detection, and other complex picture representations. Another choice was to draw ideas from biology.

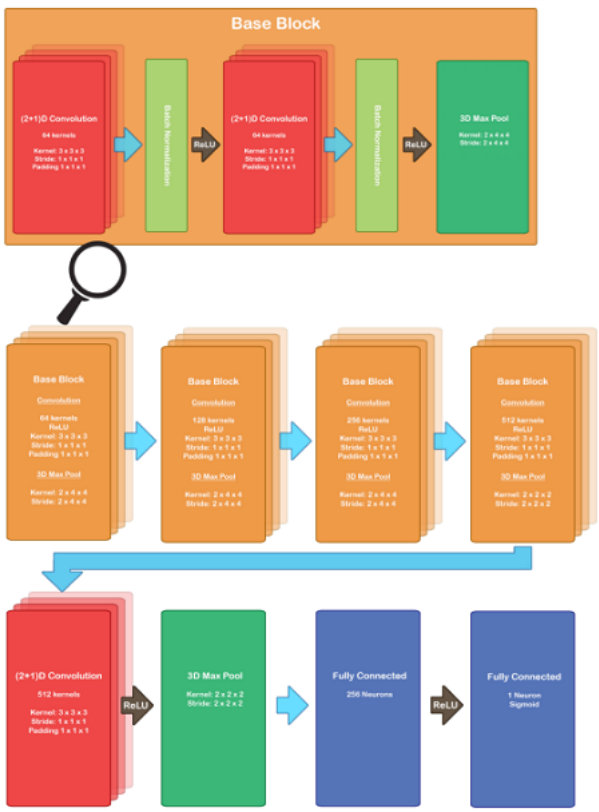

Figure 3.4: CNN with a description of layers in every singular block.

Twelve convolutional layers and three fully connected layers make up the proposed CNN network. In contrast to Image Net classification, the classification of a brain tumour in brain tumour sample images is very different. ImageNet requires the classification of common objects like animals, home goods, and so forth. Similar to this, photographs of brain tumour samples are used in medicine to identify and classify glaucoma symptoms. As a result, it is reasonable to assume that the deeper convolutional layers were replaced and the fully connected classification layers were trained from scratch. The fact that lower training losses were still connected to greater testing performance does not support the overfitting theory, although looking back at the training review can be helpful. Therefore, the price of these devices is high. Active devices are precise and easy to operate, but they can be cumbersome because they need to be physically touched by the user. Although computationally taxing, passive devices are easier to operate than active ones because they don't require direct physical interaction.

It is clear that image networks have a better propensity to fit the data than feedforward networks. The plot also shows that networks with two retrained layers are capable of achieving lower training losses than models with just one retrained layer.

3.4.1 DIMENSIONALITY OF IMAGES

the idea of convolution and how it's used in picture processing. However, some crucial information was omitted. For a convolutional neural network, a two-dimensional (2D) image with colour is not considered to be 2D. Red, green, and blue channels are typically present in colour imagery.

As a result, an image of this type actually has the following dimensions: channels, height, width, and three sizes. Therefore, a 3D kernel will be used. The convolution used is referred to as 2D convolution even though the image is 3D. This is so because the depth of the kernel and the depth of the picture are always equal when depth is seen as a channel. The seed can only move in two dimensionshorizontally and vertically.

The difference between convolution with channels and convolution with spatial depth causes the kernel to output one value at the centre of the seed, which causes the output image to be 2D.

3.4.2 CONVOLUTIONAL LAYER

Convolution-based layers are present in CNNs. In contrast to fully linked networks, which include layers of neurons, one layer has several kernels, each with its own set of trainable parameters. In a forward pass, each kernel in a layer is convolutioned before being given a bias, as seen in figure 5. One channel is produced in the output image by convolution using a single kernel. This is the situation because the kernel outputs a single channel with a single value while covering all tracks in the input image. The output channels are stacked depth-wise to produce a 2D image with a channel for each kernel if the input data is a 2D image with channels. Figure 3.5 displays both depth-wise and conventional convolution. The output of depth-wise convolution includes a value for each depth-wise piece that was observed, together with each piece's colour. Depth-wise convolution treats each depth-wise slice independently. The depth is treated as channels by regular convolution, which calculates with all channels simultaneously. The 3x3x3 kernel will produce one value as a result of this.

Any higher dimension can use the ideas presented here. For instance, 3D colour photographs for medical purposes are possible. This means that images with the same dimensions as beforechannels, depth, height, and widthare now 4D, with a channel dimension that

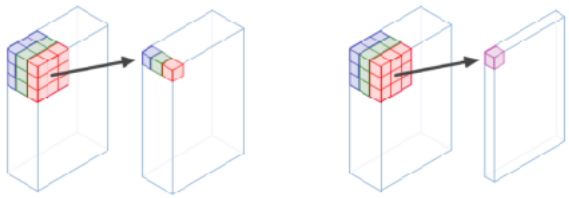

Figure 3.5: Depth-wise convolution vs. regular convolution with channels [54].

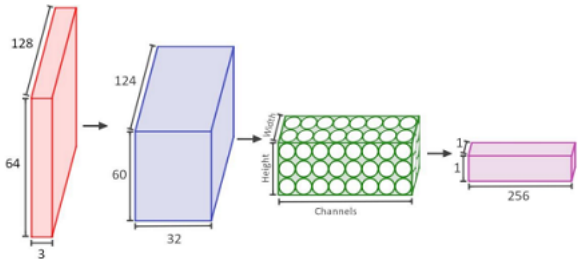

Figure 3.6: Layers of process

is the same as the channel dimension of the input volume. The three dimensions of depth, height, and width are where the kernel will migrate.

Figure 3.6 shows The 3 x 64 x 128 (channels height width) input image is represented by the red volume. The input image is processed using 32 kernels with the dimensions 3 x 5 x 5 and a stride of 1. Each kernel's output is transformed into a channel "slice" that is stacked depth-wise in the blue volume. The activations after the first layer can be seen as the blue volume. There was no padding used, hence the height and width were condensed. The green volume demonstrates how these forms might be thought of as 3D-stacked neurons. Only a small number of the blue volume's neurons are coupled to each neuron in the green volume.

3.4.3 DOWN SAMPLING AND IMAGE SAMPLING

Convolutional layers are not the only component of a convolutional neural network. Even though kernels have significantly decreased the number of parameters, it is frequently beneficial to further minimise the number of parameters. Downsampling is thus employed. There are numerous methods for downsampling. Simply by increasing the stride size beyond 1, we can shrink the output image. If padding is not used, a downsample occurs naturally, and techniques like dilation can shrink the final image. The kernel can be spread

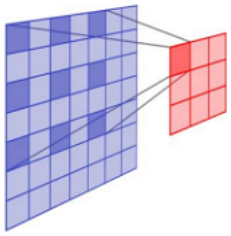

Figure 3.7: Illustration of dilation

by dilation, as seen in figure 3.7.

Twelve convolutions layers and three fully connected layers make up the proposed CNN network. In contrast to Image Net classification, the classification of a brain tumour in brain tumour sample images is very different. Image Net requires the classification of common objects like animals, home goods, and so forth. Similar to this, photographs of brain tumour samples are used in medicine to identify and classify glaucoma symptoms.

The gradient for all not local maximums values is set to 0 to enable back propagation across a max pool layer. The gradient will then only be calculated for the highest values M moving from the layer below the maximum pool to the layer above the maximum pool. By adding the gradient to chosen connections, it will appear as though the max pool layer does not exist. If kernels overlap, it is necessary to combine several error signals into a single unit.

3.4.4 BATCH NORMALIZATION

The entire training time is greatly reduced by normalising the convolutional layer's input [47]. The mean and variance of the input characteristics are typically adjusted to 0 and 1, respectively, to perform the normalisation. The network only has to handle numbers in one range for all input neurons due to the modified distribution of the input. By extending this idea to the network's hidden layers, batch normalisation goes one step further. Batch normalisation lowers covariant shift, which is one of its benefits. Consider how a neural network may interpret this as follows: To determine whether an image contains a brain tumour or not, you are building a network.

Brown brain tumours are the sole aspect of the issue that the border function that the network has learned about has seen. It lacks the knowledge to understand how the function should appear for more diverse types of brain tumours. Due to the input's lack of normalisation, the new the location of coloured brain tumours is too far away from the photographs,

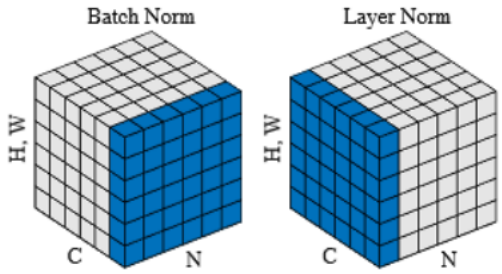

Figure 3.8: Difference between batch normalization and layer normalization

therefore the new brain tumour images are not generalizable by the network.

Batch normalisation performs normalisation and scaling over a mini-batch, which is a selection of input-target pairs sent through the network as a single batch in order to compute an average gradient. Only the mini-batch is taken into account when calculating the mean and variance, not the entire dataset. The network is slightly regularised as a result of this. Due to the mean and variance being slightly different between each mini-batch, the normalisation will be a little noisy. Smaller mini-batches have a stronger regularising effect since their mean and variance are closer to the mean and variance of the dataset in bigger mini-batches.

3.4.5 FEATURE EXTRACTION

Feature extraction is essential prior to the training and testing processes. This process comprises taking important details out of the image collection. This attribute includes contrast, correlation, energy, homogeneity, and entropy. The GLCM is a method for examining the texture of a picture by using the spatial relationship of pixels. In this study, feature extraction is crucial because it enables rapid differentiation between brain tumours and other illnesses. The distance between the matrices is typically stated in a variety of ways, with values typically falling between 0 and 45, 90, and 135. To increase accuracy, we can increase the number of features that are called. The entire training time is greatly reduced by normalising the convolutional layer's input [47]. The mean and variance of the input characteristics are typically adjusted to 0 and 1, respectively, to perform the normalisation. The network only has to handle numbers in one range for all input neurons due to the modified distribution of the input. By extending this idea to the network's hidden layers, batch normalisation goes one step further. Batch normalisation lowers covariant shift, which is one of its benefits. Consider how

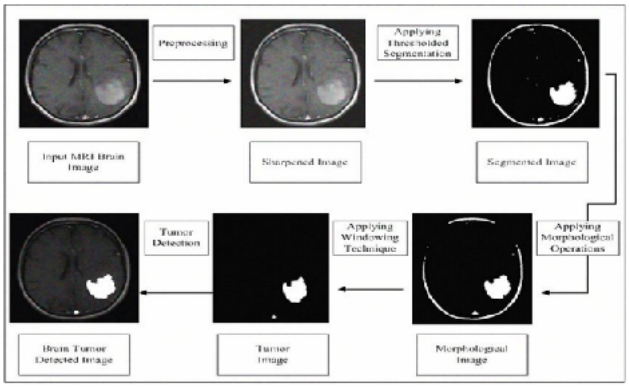

Figure 3.9: Brain image sample to acquire the interest of tumor area [55].

a neural network may interpret this as follows: To determine whether an image contains a brain tumour or not, you are building a network.

3.5 BRAIN IMAGE PRE-PROCESSING

The procedure of modifying pixels is referred to as "image processing". They are resizing pixels in an image to the proper dimensions. Image contrast, undesirable noise, and brightness are frequently reduced via data pre-processing. In order to prepare the image for preprocessing, it is converted to grayscale, black and white, histogram equalisation, and thresholding. The image was subjected to further image pre-processing procedures using a created GUI on MATLAB software. A neural network can be modified to perform the task of classifying brain tumours using the deep learning approach. On tasks related to image classification, which are on an abstract level the same domain, a model like CNN has demonstrated good performance. A deep convolutional network's earlier layers are supposed to find more basic features. Simple shapes like ed ges may be visible in the initial layers. Subsequent layers somewhat increase complexity by revealing geometric shapes, and with each subsequent layer, more intricate characteristics are displayed. Deep convolutional networks are akin to the most basic elements that a network finds. In order to reduce the number of parameters that need training, one should use the initial layers from a model that has already been trained.

However, it should be highlighted that there is no assurance that max-pooling won't exclude information that the CNN model would find beneficial. In addition to max pooling, there are various pooling methods. Generally speaking, they operate in the same manner

while employing other reduction functions, such as average. By simplifying the representation to just the most crucial elements, it is intended to promote spatial invariance. The CNN can become perplexed if the input has been altered or exposed to a lot of noise. By minimising the representation and keeping only the most crucial spatial relationships, max pooling aims to address this issue.

By simplifying the representation to just the most crucial elements, it is intended to promote spatial invariance. The CNN can become perplexed if the input has been altered or exposed to a lot of noise. By minimising the representation and keeping only the most crucial spatial relationships, max pooling aims to address this issue.

3.6 TRAINING AND TESTING

70% of the dataset was used for training, 20% for testing, and 10% for validation. Using the MATLAB programme, a convolutional neural network back propagation was produced. Each was set with a total of 60 epochs. Repeated retraining of the neural network can also increase the number of accuracy levels. Figure 3.10 depicts the CNN's training and testing with regard to model loss and 60-epoch training.

Parameter/Decision	**Choice**
Training Mini-Batch Size	1
Validation Mini-Batch Size	1
Learning Rate	0.005
Architecture	CNN
Training	70%
Testing and Validation	20% and 10%
No. of Epochs	60

Table 3.1: The description of parameters being used for training and testing an expert system.

The mini-batch size was constrained by the GPU's storage capacity, and as it turned out, it provided a respectable performance. Therefore, multi-GPU training, which would enable larger mini-batches, received less attention. Every time the training process discovered a higher validation accuracy, the model parameters were saved. Instead of early stopping, this strategy was applied. Because server standards provide for a total time limit on the

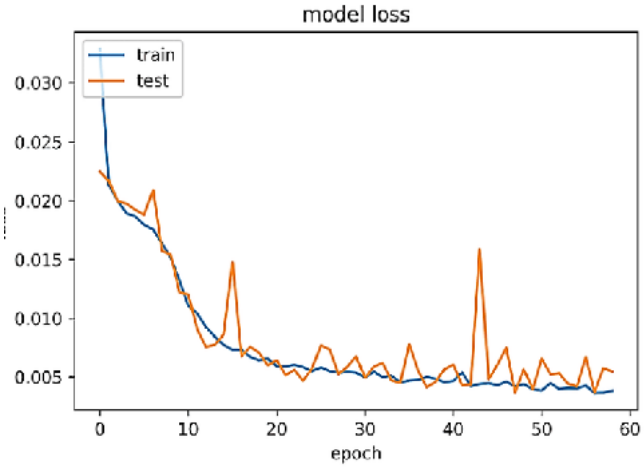

Figure 3.10: The training and testing parameters for CNN

GPU resource, storing models and applying time limitations was perfect for training several models sequentially on a single GPU. Three hours per model were allotted for the training period. After discovering that this is typically when a model reaches its highest validation loss and accuracy, the training time was chosen.

If a new design choice was tested and the network was occasionally manually monitored, a considerably more important time frame was defined. Every action was recorded, including training loss, to help assess the network's performance. Every ten steps, accuracy and validation loss were calculated. For the entire validation set, loss and accuracy were determined and averaged. By thresholding the network output at 0.5, an accuracy prediction was created.

3.7 SUMMARY

Convolutional neural networks are given more layers before being assembled into the CNN architecture. The usage of kernels of size 3 in this work is inspired by the well-known CNN model VGG16. ResNet and other models that have excelled at identifying and categorising medical disorders in ImageNet are the inspiration behind the usage of batch normalisation. The same rationales underlie the adoption of the max pooling for downsampling and the activation function ReLU. The number of convolutional layers was determined through trial and error and was compatible with the limitations of the resources at hand. A base block, or repeating pattern, was abstracted to the layer of the network. The validation accuracy for brain tumour detection and classification is significantly decreased when a block is reduced

to just one convolution. Additionally, the experimental arrangement lacked the capability for a network that was far bigger than CNN. Two CNN-based convolutional layers were present in each block. In the area of identifying and categorising brain tumour diseases, we observed similarities. The CNN appeared to be the best option for training when great complexity is desired but fewer parameters are needed because of the size of the dataset. However, the design can swiftly transition to 3D convolution.

CHAPTER 4

BACKGROUND STUDY

4.1 MEDICAL IMAGE

The visualisation of internal structures for use in disease surveillance, therapy, and diagnosis is known as medical imaging. Imaging techniques cover a variety of medical specialties, including radiology, nuclear medicine, optical imaging, and image-guided intervention.

4.2 IMPORTANCE OF MEDICAL IMAGE ANALYSIS

One of today's most cutting-edge technologies for processing photographs and films is digital image processing. Currently, digital image processing is most commonly used for x-rays. Prior to the invention of x-rays, it was extremely challenging for a doctor to inspect a patient's bones without first severing their skin and flesh in order to see if they were fractured or otherwise injured. Image processing can be used for both security and personal use, from every angle and lane.

Technology has advanced significantly since Roentgen's invention of the X-ray in 1895, giving rise to modern imaging methods like Computed Tomography (CT) and Magnetic Resonance Imaging (MRI). As time goes on, image technology will continue to evolve. However, in modern medical imaging, the emphasis is moving from picture creation and capture to image post-processing and data management. The necessity of making effective use of already-existing data motivates this. Recent advancements in imaging science have demonstrated the technology's ability to advance and change a variety of clinical medical practises.

4.3 TRADITIONAL MACHINE LEARNING CLASSIFIERS

A classification technique is used to evaluate our suggested model. As a result, we have completed half of the objective by developing a model that can detect or segment the tumour. Through model evaluation or examination of the suggested model, the model is justified. We developed a method that utilised six conventional machine learning algorithms. We have employed these numerous classifier types to support our model. To determine the accuracy of our proposed model's tumour identification, six conventional machine learning classifiers: K-Nearest Neighbour, Logistic Regression, Multi-layer Perceptron, Naive Bayes, Random Forest, and Support Vector Machine. Machine learning methods and deep learning models are employed to analyze brain images or data for brain tumor behavior. These models can learn patterns and characteristics specific to brain tumor activities, as they are trained on datasets containing both tumor and non-tumor images. The goal is to accurately detect and classify brain tumors using automated algorithms.

The process of creating a fraud detection model involves several key elements:

Data Import: The dataset is divided into two folders, testing and training, with each containing images in four subfolders. Data Preprocessing: Data preprocessing is essential to ensure the data is in a format suitable for machine learning algorithms. This may include data cleansing, feature engineering, outlier reduction, and data normalization to prepare the data for training the model effectively. Feature Extraction: Feature extraction involves identifying relevant qualities or characteristics that can be used to distinguish between honest and dishonest behaviors in fraud detection. Model Fitting: After data cleaning and preprocessing, the data is fitted to the machine learning model. Various algorithms can be used for model development, each providing different performance levels: Logistic Regression: Achieves an accuracy of 0.963. SVM (Support Vector Machine): Achieves an accuracy of 0.959. CNN (Convolutional Neural Network): Achieves an accuracy of 0.979. The use of different algorithms allows for the selection of the most suitable model for the specific fraud detection task. The CNN, in particular, has shown the highest accuracy in detecting fraudulent activities.

In conclusion, machine learning and deep learning techniques are powerful tools for brain tumor analysis and fraud detection. By leveraging these methods, accurate and automated classification of brain tumors and fraudulent behaviors can be achieved, leading to

improved medical diagnostics and enhanced security measures.

4.3.1 LOGISTIC REGRESSION

The logistic regression is a predictive analysis, just like all regression analyses. To describe data and explain the relationship between one dependent binary variable and one or more independent nominal, ordinal, interval, or ratio-level variables, we employ logistic regression. Like linear regression, logistic regression represents data using an equation. To anticipate an output value (y), input values (x) are blended linearly using weights or coefficient values (also known as the Greek capital letter Beta). The output value being modelled is a binary value (0 or 1) rather than a numeric number, which is a significant distinction from linear regression. The logistic regression equation is shown below:

where y is the anticipated result, b0 is the bias term (also known as the intercept term), and b1 is the coefficient for the single input value (x).

4.3.2 SUPPORT VECTOR MACHINE (SVM)

Support vector machines analyse data used for classification and regression analysis. They are supervised learning models with corresponding learning methods. A non-probabilistic binary linear classifier is created by an SVM training algorithm given a series of training samples, each of which has been labelled as belonging to one of two categories or the other [97]. An SVM model is a mapping of the examples as points in space with as much space between the examples of the various categories as possible. Then, based on which side of the gap they fall, new samples are projected into that same area and predicted to belong to a category. SVM may effectively conduct non-linear classification in addition to linear classification by implicitly mapping their inputs into high-dimensional feature spaces. This technique is known as the kernel trick.

4.3.3 CONVOLUTIONAL NEURAL NETWORK (CNN)

Kunihiko Fukushima developed the fundamental concept of Convolutional Neural Networks in the 1980s [88]. Convolutional Neural Networks (ConvNets or CNNs) are a type of neural network that have shown to be very successful in fields like image identification and classification. Convolutional neural networks, which are effective at classifying images,

dominate computer vision approaches. CNNs are a subclass of deep, feed-forward artificial neural networks (in which connections between nodes do not create cycles) and use a multi-layer perceptron variant that only needs a little amount of pre-processing. The explicit assumption that the inputs are images in ConvNet designs enables us to embed specific features into the architecture. These therefore greatly reduce the number of parameters in the network and improve the implementation efficiency of the forward function. Neurons that make up ConvNets have biases and weights that can be learned. Each neuron processes a few inputs, conducts a dot product, and may optionally do a non-linearity as a follow-up. From the unprocessed image pixels at one end to the class scores at the other, the entire network continues to represent a single differentiable score function. And the final layer has a loss function.

4.3.4 TRANSFER LEARNING

With previously trained neural networks, transfer learning is a sort of machine learning that is often used. Some transfer learning models, including as VGG16, ResNet-50, and Inception V3, are often used for picture categorization and detection. The most advantageous teaching strategies in terms of price and time efficiency are transfer learning techniques. Pre-trained models are used to transfer data and complete the work rather than starting from scratch, which takes more time and necessitates the use of GPU resources and massive image databases.

Deep convolutional neural network models may take days or even weeks to train on very large datasets. A way to short-cut this process is to re-use the model weights from pre-trained models that were developed for standard computer vision benchmark datasets, such as the ImageNet image recognition tasks. Top performing models can be downloaded and used directly, or integrated into a new model for your own computer vision problems.

In this post, you will discover how to use transfer learning when developing convolutional neural networks for computer vision applications.

After reading this post, you will know:

Transfer learning involves using models trained on one problem as a starting point on a related problem. Transfer learning is flexible, allowing the use of pre-trained models directly, as feature extraction preprocessing, and integrated into entirely new models. Keras provides convenient access to many top performing models on the ImageNet image recognition tasks

such as VGG, Inception, and ResNet. Transfer learning generally refers to a process where a model trained on one problem is used in some way on a second related problem.

In deep learning, transfer learning is a technique whereby a neural network model is first trained on a problem similar to the problem that is being solved. One or more layers from the trained model are then used in a new model trained on the problem of interest. Transfer learning has the benefit of decreasing the training time for a neural network model and can result in lower generalization error.

The weights in re-used layers may be used as the starting point for the training process and adapted in response to the new problem. This usage treats transfer learning as a type of weight initialization scheme. This may be useful when the first related problem has a lot more labeled data than the problem of interest and the similarity in the structure of the problem may be useful in both contexts. A range of high-performing models have been developed for image classification and demonstrated on the annual ImageNet Large Scale Visual Recognition Challenge, or ILSVRC.

This challenge, often referred to simply as ImageNet, given the source of the image used in the competition, has resulted in a number of innovations in the architecture and training of convolutional neural networks. In addition, many of the models used in the competitions have been released under a permissive license.

These models can be used as the basis for transfer learning in computer vision applications.

This is desirable for a number of reasons, not least:

- **Useful Learned Features**: The models have learned how to detect generic features from photographs, given that they were trained on more than 1,000,000 images for 1,000 categories.
- **State-of-the-Art Performance**: The models achieved state of the art performance and remain effective on the specific image recognition task for which they were developed.
- **Easily Accessible**: The model weights are provided as free downloadable files and many libraries provide convenient APIs to download and use the models directly.

The model weights can be downloaded and used in the same model architecture using a range of different deep learning libraries, including Keras.

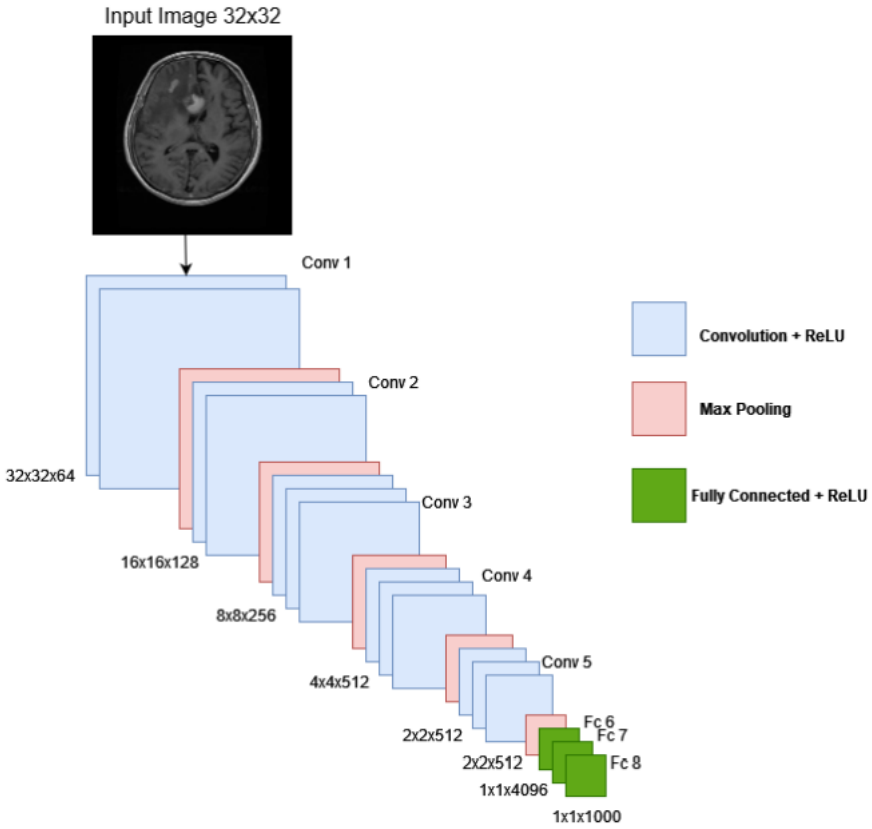

Figure 4.1: . Architecture of VGG16

4.3.5 VGG16

One of the VGG-NET-based networks is the VGG16 network structure, which Zisserman and Simonyan proposed in 2014 . Similar to AlexNet , VGG16 is a deeper network for recognising and categorising pictures. You may train VGG16 using the ImageNet database . The dataset can be stated more accurately when classifying and recognising the photos using VGG16 [66]. One advantage of VGG16 is that it performs better in challenging context identification tasks and when dealing with large amounts of data . The VGG16 network has a 3 by 3 receptive field and 16 convolutional layers. There are a total of 5 such layers (max-pooling layers), each measuring 2 x 2. Following the last max-pooling layer, three fully linked

4.3.6 Inception V3

Deep learning network model Inception v3 is mostly used for picture categorization and detection [7073]. Low computer configuration makes it challenging to train the Inception

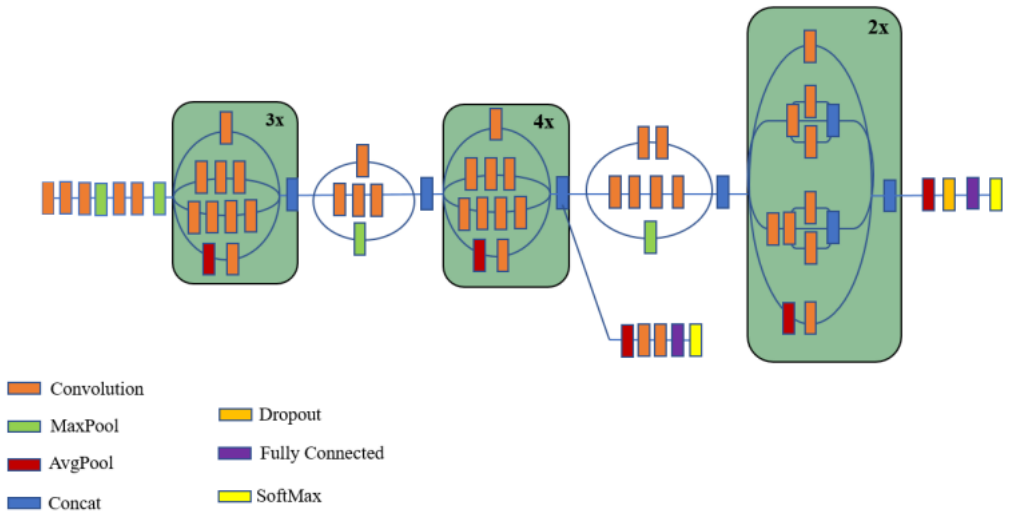

Figure 4.2: Architecture of Inception V3

V3 model; occasionally, it takes a few days [71,74].

Inception V3 is an improvement on Inception V1, which GoogleNet released in 2014 [75]. Inception V3 was introduced in 2015 with 42 layers and significantly lower error rates than its forerunners. Convolution, pooling, dropout, fully linked, and softmax are the stages of the Inception process [76, 77]. The architecture of Inception V3 is displayed in Figure 4.3.

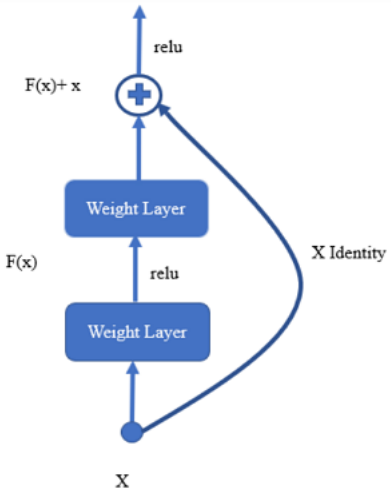

Figure 4.3: Residual block.

4.3.7 ResNet-50

ResNet-50 is the abbreviation for the residual network. One million samples from the ImageNet collection were used to train ResNet-50, a modification of the ResNet architecture with 50 deep layers. There are several average pooling convolutional units in the ResNet-50 architecture.

ResNet-50 is the abbreviation for the residual network. One million samples from the ImageNet collection were used to train ResNet-50, a modification of the ResNet architecture with 50 deep layers. There are several average pooling convolutional units in the ResNet-50 design.

CHAPTER 5

RESULTS AND DISCUSSION

5.1 INTRODUCTION

This chapter presents the findings from the approach described in Chapter 3's methodology section. We talked about how the data were analysed and what each experiment's findings revealed. The performance and accuracy of a CNN model based on deep learning for the detection and classification of brain tumours is discussed as the chapter's conclusion. CNN performs well based on knowledge and makes judgements in a situation when we just have the incomplete information needed for humans. Thankfully, we can readily meet this criterion by using the deep learning and neural network toolboxes. With 10-fold cross validation, the dataset's observations are all evenly distributed.

First and foremost,On the basis of the brain tumour MR image dataset, the VGG16, CNN, ResNet-50, and Inception V3 classification algorithms produced findings that are analysed in Table 3 and contrasted in Figure and Table shows how well the models performed in terms of accuracy, area under the curve (AUC), recall, and loss function outcomes. Based on the results in Table 1, it was determined that the CNN performed better than the other deep learning models after examining the techniques of the CNN, VGG16, ResNet-50, and Inception V3. The validation results for the CNN were 93.3% validation accuracy, 98.43% validation AUC, 91.1% validation recall, and 0.260 validation loss.

Figure 5.2 shows the graphs for the CNN, ResNet-50, Inception V3, and VGG16 vali-

Models	Accuracy (%)	AUC (%)	Recall (%)	Loss
CNN	93.30	98.43	91.13	0.25
ResNet-50	81.10	94.20	81.04	0.85
VGG16	71.60	89.60	70.03	1.18%
Inception V3	80.00	89.14	79.81	3.67%

Table 5.1: Deep learning models performance on brain tumor detection

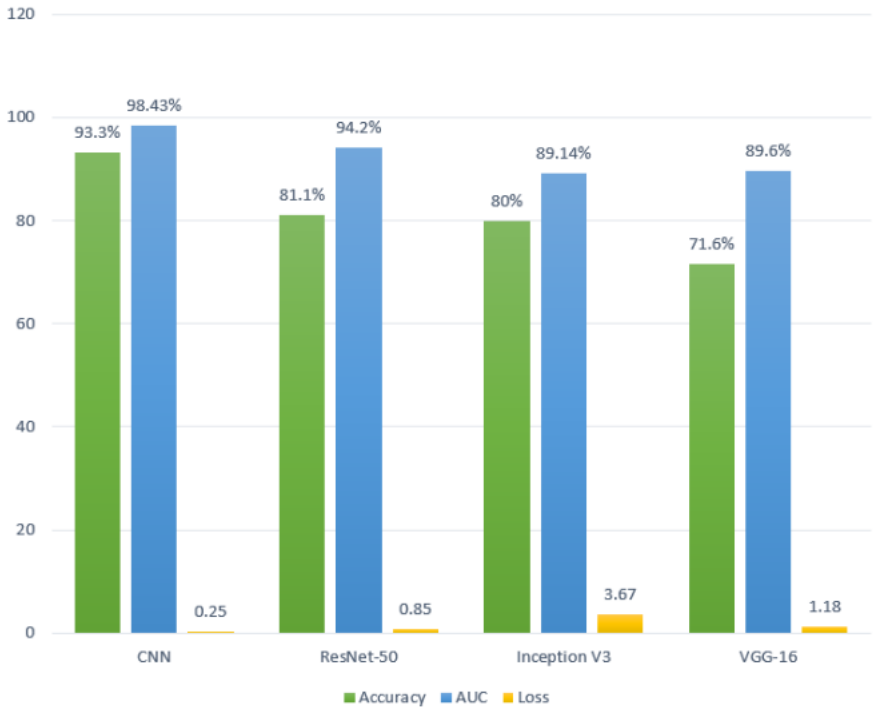

Figure 5.1: Performance analysis of the proposed model in terms of the accuracy, AUC, and loss

dation accuracy with relation to training accuracy. The orange lines show the validation accuracy, while the blue lines show the training accuracy. With a training accuracy rating of 90.50%, CNN here attained the greatest validation accuracy of 93.30%. The greatest training accuracy number for ResNet-50 was 98.43%, with a validation accuracy of 81.10%. The validation accuracy for Inception V3 was 80%, whereas the training accuracy was 91.79%. However, VGG16 has the lowest validation accuracy (71.60%) and training accuracy (79.20%) of any model. The Adam optimizer was used to choose the batch size of 18 and the epochs of 80 for the models' implementation.

The accuracy graph analysis revealed that the CNN outperformed the other models since there were no overfitting or underfitting issues, and the validation accuracy had a great output curve relative to the training accuracy.

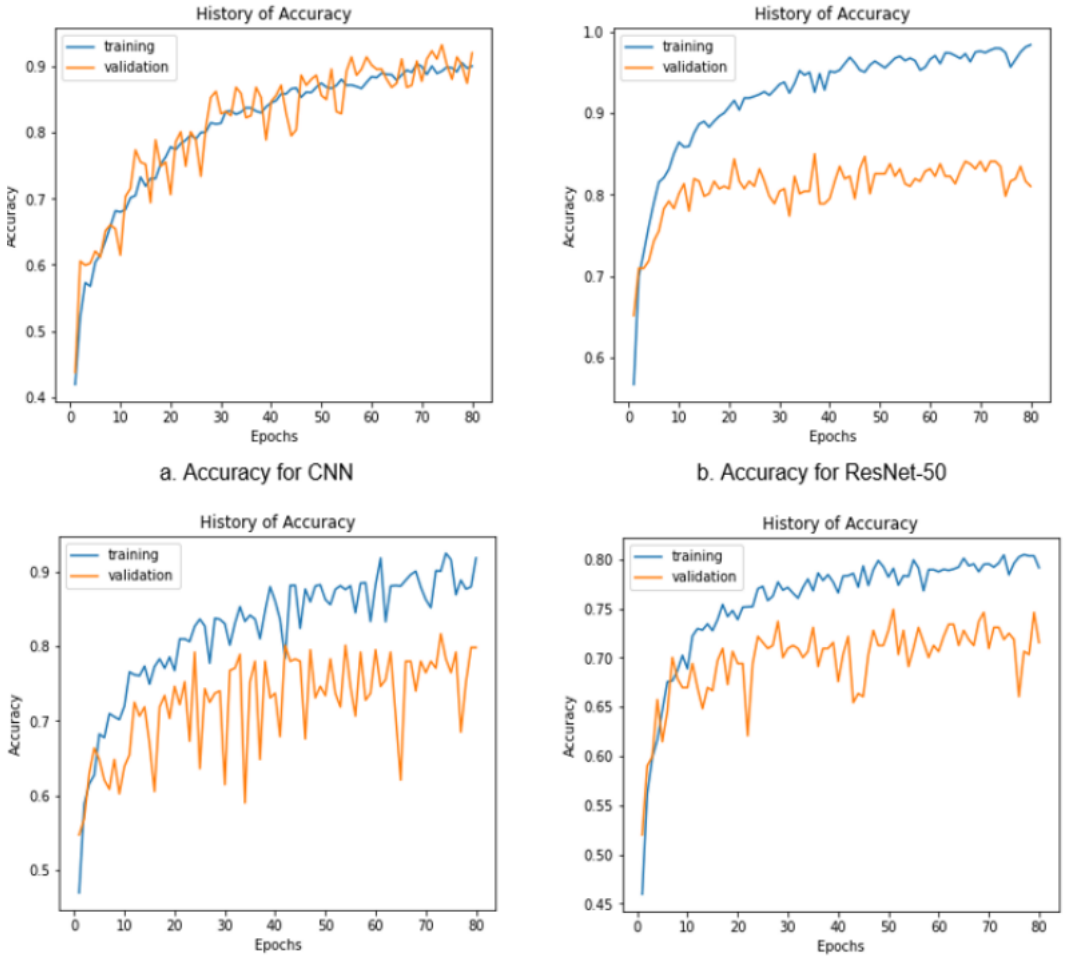

Figure 5.2: Accuracy Graphs for the CNN, ResNet-50, Inception V3, and VGG16.

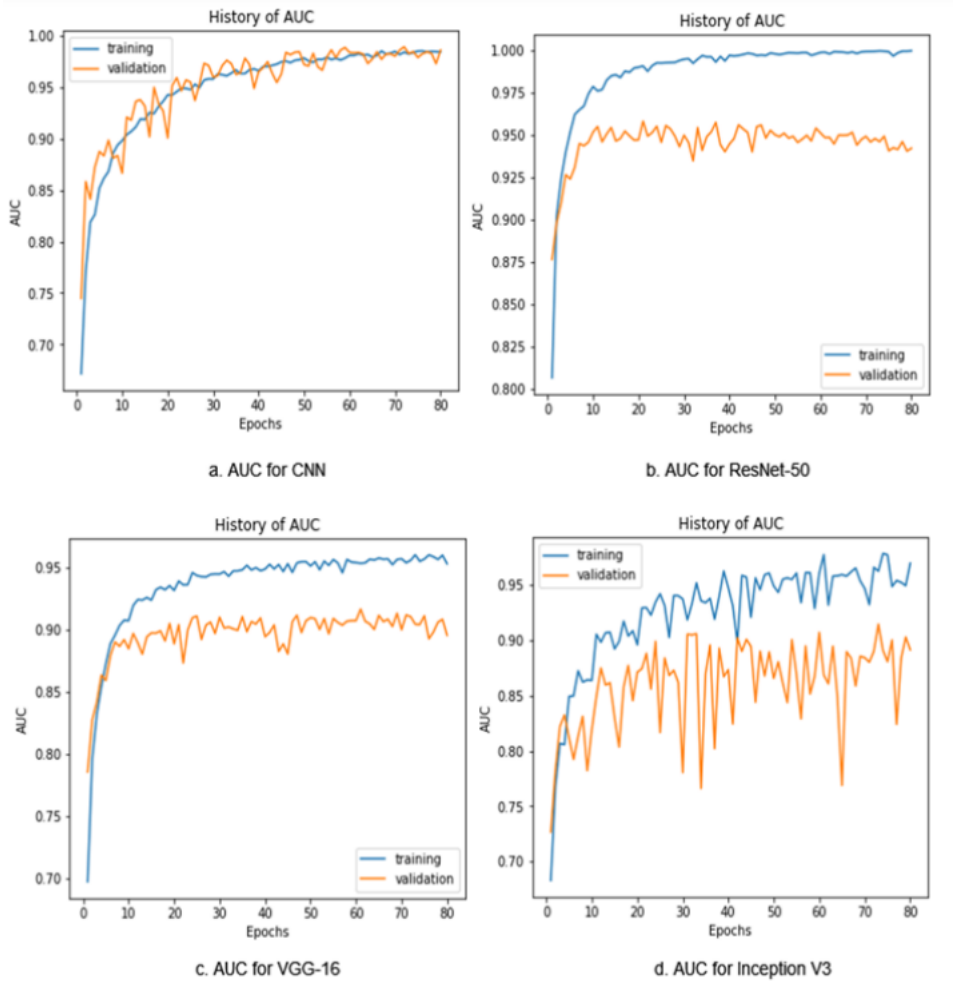

Figure 5.3: AUC graph for the CNN, ResNet-50, Inception V3, and VGG16.

Figure 5.3 presents the graphs for the CNN, ResNet-50, VGG16, and Inception V3 validation AUC in relation to the training AUC. The training AUC is shown by the blue lines, while the validation AUC is represented by the orange lines. With a training accuracy rating of 90.50%, CNN here attained the greatest validation accuracy of 93.30%. The model's performance and capacity to distinguish between classes are evaluated by the AUC. The performance of the model improves as the AUC value increases. With a training AUC value of 98.40%, the CNN here earned the highest validation AUC of 98.43%.

The maximum training AUC value for ResNet-50 was 99.95%, with a validation AUC of 94.20%. The least accurate training system, Inception V3, had a validation AUC of 89.14%. The least training accuracy was 95.32% and the validation AUC was 89.60% for VGG16.

The validation loss relative to the training loss graphs for the CNN, ResNet-50, VGG16, and Inception V3 are shown in Figure 11 in the appropriate manner. The training loss is shown by the blue lines, while the validation loss is shown by the orange lines. Loss is a

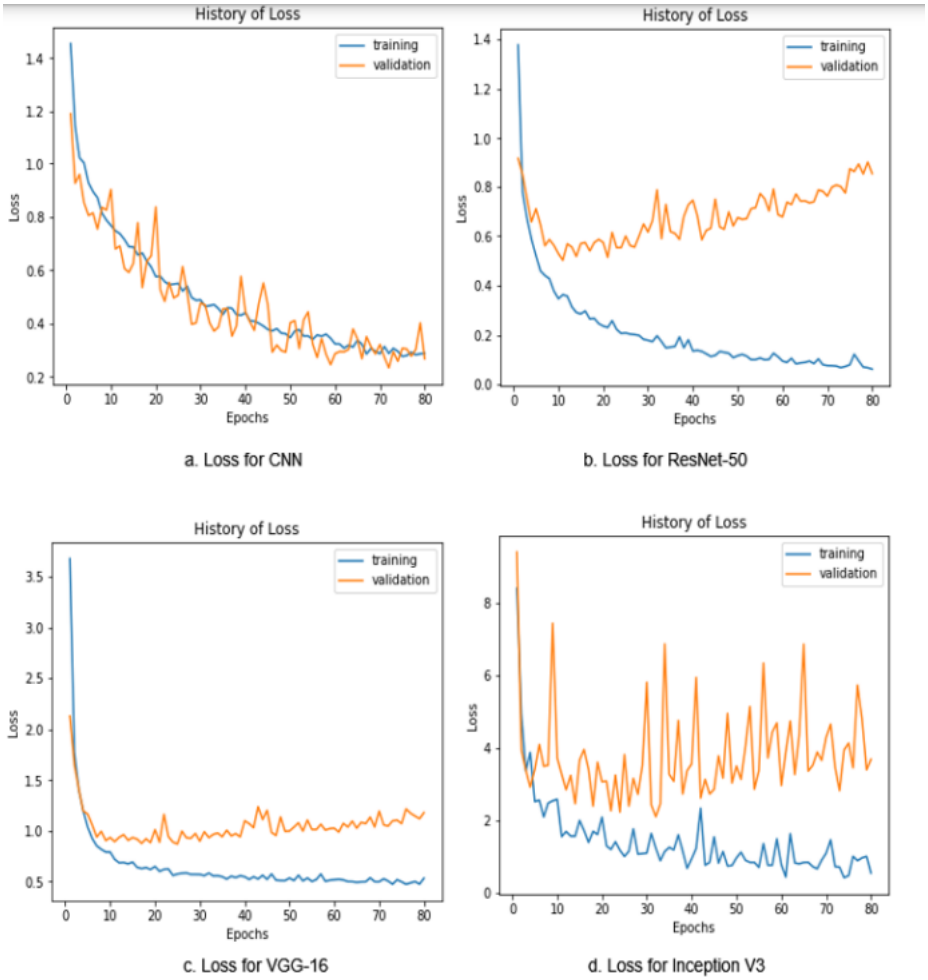

Figure 5.4: Loss graph for the CNN, ResNet-50, Inception V3, and VGG16.

punishment for making a wrong prediction. The loss, on the other hand, is a quantification of the error in the model's forecast at each epoch. The model's prediction is accurate if the loss is zero; if not, the loss is higher. We utilised the categorical cross-entropy loss function to determine the loss experienced throughout the detecting procedure. A loss function that is frequently employed in multi-class classification assignments is the categorical cross-entropy. In this case, the CNN had a training loss value of 0.289 and the lowest validation loss of 0.250. With a very low training loss value of 0.063, ResNet-50 obtained a validation loss of 0.853. With a validation loss of 3.67 and a training loss of 0.535, Inception V3 had the highest performance. VGG16, on the other hand, recorded validation losses of 1.18 and training losses of 0.533.

Figure 5.4 illustrates a comparison of the models' accuracy, AUC, and loss. The accuracy, AUC, and loss were taken into consideration while evaluating the models' performance. Figure 5 shows that the CNN outperformed the other models in terms of validation accuracy

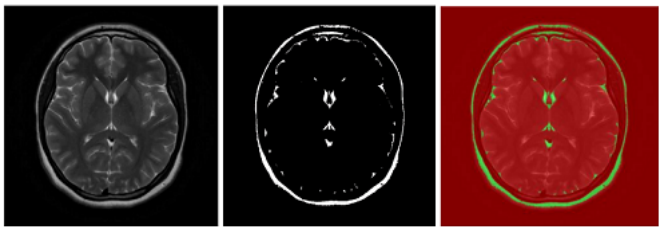

Figure 5.5: Test illustration 1

and AUC, with scores of 93.30% and 98.43%, respectively. The CNN got the lowest validation loss of 0.25, except from that. The lowest AUC score (89.14%) and largest loss (3.67) were both attained by Inception V3. Additionally, VGG16 had the highest loss value of 1.18 and the lowest accuracy score of 71.60%. ResNet-50 also had a reduced loss value and the second-highest accuracy and AUC score. The proposed CNN was deemed the best model for detecting multiple brain tumours using MR images after examining the overall scores and performance.

there are no lighting limitations because the algorithms are based on reflections from moving time. Second, compared to deep learning methods, ultrasonic wave processing of reviews reduces computational complexity. Ultrasonic waves are one technique for categorization based on deep learning to estimate the range of the moving time and the received signal strength. There are several approaches to estimate range using time of flight. The received signal strength can be used to identify the series occurrence. Following that, range estimation can be used to categorise the type of the executed series. To train the convolutional neural network to reconstruct inputs from a latent space representation, only illnesses and brain tumours were used. As a result, the convolutional neural network performs more accurately when given a healthy disease sample as input since it is simpler to identify the disease. When there is a sufficient amount of brain tumour in a disease sample image, a convolutional neural network requires more processing time. Nevertheless, it can be put through a thorough processing pipeline to find abnormal locations.

The input image of the brain, the segmented brain image, and the classification of any brain tumours inside the area of interest (ROI) are shown in Figures 4.11, 4.12, and 4.13.

The "best" model to store for training was chosen based on the accuracy of detecting positive tumours. For more information on how the network assesses data, precision and recall were chosen. The final choice for the total performance metric is AUC, which functions similarly to accuracy but also takes the degree of prediction certainty into account. A brain

Figure 5.6: Test illustration 2

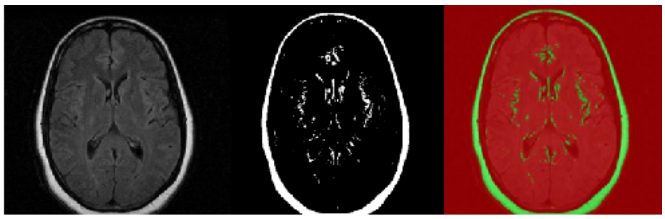

Figure 5.7: Test illustration 3

tumour will be correctly identified 98% of the time by the convolutional neural network, according to its precision score of 0.98. Because it advises using the network with caution, this is interesting. There is a 2% possibility that CNN will diagnose the erroneous condition if used blindly.

The hospital may incur costs as a result of using resources that are not beneficial to the patient. Furthermore, recollection shows that 98% of the individuals with brain tumours will be picked up by CNN. This gap can be further closed with cooperation and critical CNN use. By providing insight into CNN's decision-making process, the visualisation bridges the gap between the model and the health practitioner. The input picture of the brain, the segmented image of the brain, and the classification of the brain tumour inside the region of interest (ROI) are shown in Figures 4.14, 4.15, and 4.16, respectively.

The findings show that when trained on the Kaggle dataset, deep convolution outperforms simple convolution on all criteria. With a 0.01 difference in the mean recall, recall is the most significant difference among all the metrics examined. The deep convolution neu-

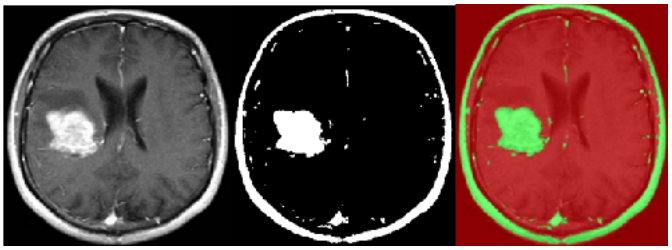

Figure 5.8: Test illustration 4

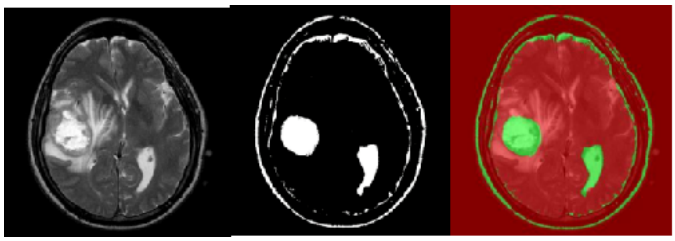

Figure 5.9: Test illustration 5

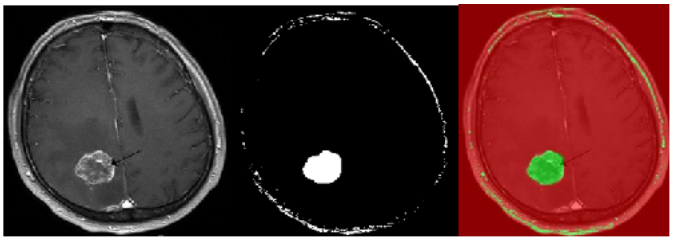

Figure 5.10: Test illustration 6

ral network performs noticeably better at identifying and categorising brain tumour disease, as seen by their AUC values. Additionally, the CNN model was trained using a randomly initialised train/test split in the ratio of around 70/20 and 10 independently for validation for the Kaggle dataset. This was achieved by combining all cases into a single array, rearrange them, and then divide the array in two, with each split containing the same number of issues as the original array. Then, the shuffled indices were retained with the model to ensure that no training cases would be used in the evaluation procedure.

Metric	**CNN**	Artificial **Neural Network**
Accuracy	0.97917 ś 0.025039	0.91667 ś 0.028464
Precision	0.98361 ś 0.019689	0.98554 ś 0.024118
Recall	0.98541 ś0.026435	0.88958 ś 0.055941
AUC	0.99765 ś0.0036011	0.98993 ś 0.0063338

Table 5.2: CNN accuracy metrics results are shown in fractions

The framework's brain tumour illness detection and classification matrices required to be developed using information from CNN. Already, maximum pooling identifies the highest contributing values. Therefore, from a discriminative site, one just needs to return through the max pool layer to locate the next discriminative position. The values are normalised by batch normalisation, thus the greatest contributing values remain the same. The discrimi-

Sample Image	Brain tumor Risk
IMG1.jpg	0.28218
IMG2.jpg	0.34816
IMG3.jpg	0.027847
IMG4.jpg	0.28081
IMG5.jpg	0.28086
IMG6.jpg	0.77814
IMG7.jpg	0.23869
IMG8.jpg	0.35365
IMG9.jpg	0.28825
IMG10.jpg	0.24779
IMG11.jpg	0.018874
IMG12.jpg	0.22376
IMG13.jpg	0.23201
IMG14.jpg	0.3038
IMG15.jpg	0.24816
IMG16.jpg	0.24682
IMG17.jpg	0.37032
IMG18.jpg	0.39568
IMG19.jpg	0.24506
IMG20.jpg	0.6983

Table 5.3: The Brain tumor images list with risks

nating sites in this instance were transferred to the following layer.

5.1.1 DISCUSSION

As demonstrated in Table 6, convolutional neural networks (CNN) perform well when used to process imagery data with a large number of instances. a basic comparison of our suggested architecture and a number of current systems. We have reviewed current methods that can reasonably identify and categorise brain cancers in other studies that have been published. One could argue that if CNN were ever implemented in practise, it might be more effective in returning the output likelihood of a brain tumour by comparing this to the AUC value. This would add to our knowledge of uncertainty. This teaches us to be sceptical of the findings. Although CNN performs well, visualisations showed symptoms of fragility. Positively, the approaches utilised to produce this finding demonstrate the value of deep learning and visualisation techniques.

Health professionals' work can be made easier and less uncertain by a system that visualises the output likelihood, making it a more practical tool to employ in practise.

CNN demonstrates that deep learning is a practical method for identifying and classifying brain tumour disease, as do many related works.

- Successful use of brain vascular coherence tomography (BVCT) scans to test the deep convolutional neural network.
- A CNN architecture taught on a single GPU without any pre-training is capable of state-of-the-art performance.
- Using CNN fixations to display symptoms associated with a medical classification.
- CNN is implemented and used to obtain more accurate results while creating an expert system.

It was beneficial to introduce and utilise a fresh dataset from the open-source Kaggle repository for visualising outcomes.

5.2 CONCLUSION

A major factor in lowering death rates globally can be early diagnosis of brain tumours. Correctly identifying brain tumours remains very difficult because of the tumor's shape, fluctuating size, and structure. The MR image classification has a significant impact on the clinical diagnostic and treatment choices made for patients with brain tumours. The tumour segmentation method and using MR images to identify brain tumours early are promising. Before the tumour location can be clearly identified and categorised, there is still much work to be done.

In our work, we employed a range of MRI brain tumour pictures for the objective of early brain tumour identification. Classification and detection are significantly impacted by deep learning models as well. With the help of several MR scans, we presented a CNN model for the early diagnosis of brain tumours and saw encouraging results. To guarantee the effectiveness of the ML models throughout the evaluation process, we used a number of indicators. We considered a few additional ML models in addition to the proposed model when evaluating our results. Regarding the research's shortcomings, the training process took a lengthy time due to the CNN's multiple layers and the computer's subpar GPU.

It would require more time to train if the dataset was huge, such as 1,000 photos. We decreased the training time after making improvements to our GPU system. Future research can be done to more accurately diagnose brain tumours by utilising the unique patient data acquired from any source.

As observed from the results, the three models, SVC, Logistic Regression, and CNN, achieved accuracy values of 0.963, 0.959, and 0.979, respectively. Notably, the Convolutional Neural Network (CNN) model outperformed the other two models, demonstrating its superior accuracy in precisely predicting the classification of brain tumors.

With a test accuracy of 0.979, the CNN model has proven to be highly effective in detecting brain tumors. This high accuracy indicates that the CNN model has successfully learned intricate patterns and features that are characteristic of brain tumor behavior from the training dataset. As a result, the CNN model is capable of making precise predictions, distinguishing between tumor and non-tumor regions in brain images.

The findings of this study highlight the potential of the CNN model as a valuable tool for medical practitioners in the field of brain tumor diagnosis and treatment planning. The accurate and automated classification of brain tumors offered by the CNN model can significantly enhance the efficiency and reliability of brain tumor detection. Medical professionals can benefit from the CNN model's capabilities, aiding them in making well-informed decisions and formulating appropriate treatment plans for brain tumor patients.

By leveraging the power of deep learning with the CNN model, the medical field can improve patient care and optimize resources, leading to more timely interventions and better patient outcomes. The CNN model's performance demonstrates its potential to revolutionize brain tumor diagnostics and contribute to advancements in medical imaging analysis.

5.3 FUTURE RECOMMENDATIONS

The following areas will be the most important to improve in the future and require the most attention:

- The development of CNN as it adds more layers to become DCNN.
- Data improvement and visualisation enhancement.

- Make progress by building an expert system that is more diverse and can predict more diseases.

Depth is one thing that could help CNN perform better. A deeper network can be evaluated with additional computer power. With a deeper network, a convolutional neural network might perform better. Additionally, skip connections can be added to maintain performance as the network goes deeper. Larger mini-batches could potentially enhance training if given additional computing power. When DCNN makes the right prediction but is based on odd focus locations, adding some pre-processing, like noise reduction, may help.

One other recommendation is to use a semi-supervised approach where only a small portion of the dataset needs to be labelled. Without the need for medical professionals to manually identify new instances in the dataset, this strategy enables more training.

In the future, a method that could do this would offer crucial insight into a model's logic. The localization map in CNN Fixations' architecture is one area that might be improved. One could try to further modify the 3D Gaussian blur's sigma parameter. This parameter determines how far the "blobs" of interest for detecting and classifying a brain tumour or other disease extend over the map. To represent the density of points, another approach or 2D Gaussian blur could be used for each slice.

A larger dataset with more variety can better prepare CNN for application in the real world. Giving extra data to a deep learning algorithm is one of the best methods to enhance it. The Kaggle dataset might someday replace existing standards in the industry. Once permissions have been given, the dataset could become much bigger. Benchmarking improves comparisons between methods, which is good for the industry. We've also talked about testing on real pros. One of the greatest ways to gauge how well an automated system might perform in practise is to compare it to human performance. A dataset with statistics on human performance and detailed instructions could serve as a useful standard for the industry.

CHAPTER 6

Appendix

Brain Tumor Detection Using Machine Learning SourceCode:

https://colab.research.google.com/drive/1kaLlOOtvGZ24KwuW8Alxmeg6gpRQGhRz?usp=sharing

6.1 INTRODUCTION

Nowadays, tumors rank as the second leading cause of cancer, posing a significant threat to a large number of patients. In the medical field, there is an urgent need for fast, automated, efficient, and reliable techniques to detect tumors, particularly brain tumors, as early detection plays a crucial role in treatment and patient outcomes. The ability to accurately detect tumors allows doctors to mitigate potential dangers and provide appropriate treatment, ultimately saving lives. Various image processing techniques are utilized in this application, enabling doctors to deliver precise and effective treatments, benefiting numerous tumor patients.

One of the more severe diseases that affects both children and adults is a brain tumour. 85 to 90 % of all primary Central Nervous System (CNS) tumours are brain tumours. About 11,700 patients are given a brain tumour diagnosis each year. For those who have a malignant brain or CNS tumour, the 5-year survival rate is roughly 34% for males and 36% for women. Benign, malignant, pituitary, and other types of brain tumours are all categorised. To extend patient lives, appropriate care, careful planning, and precise diagnostics must be used. Magnetic Resonance Imaging (MRI) is the most reliable method for finding brain tumours.

The scans provide an enormous amount of picture data. The radiologist examines these pictures. Because of the complexity of brain tumours and their characteristics, a manual examination can be prone to inaccuracy. Automated classification methods based on Machine Learning (ML) and Artificial Intelligence (AI) have consistently outperformed manual categorization in terms of accuracy. So it would be beneficial for doctors all over the world to propose a system that does detection and classification utilising Deep Learning Algorithms employing ConvolutionNeural Network (CNN), Support Vector Machine(SVM), and TransferLearning (TL).

6.2 DATASET

Brain tumours are intricate. The sizes and locations of the brain tumor(s) are highly aberrant. Because of this, it is quite difficult to comprehend the tumor's nature completely. For MRI analysis, a qualified neurosurgeon is also necessary. It can be very difficult and time-

consuming to compile findings from an MRI in poor nations since there are frequently not enough skilled medical professionals and people who are knowledgeable about tumours. So, this issue can be resolved by a cloud-based automated solution. Data from an MRI are in our folder. The photographs are already divided into folders for training and testing.Every folder contains four or more subfolders. These folders contain MRI images of the appropriate tumour types.

Four directories altogether make up our testing dataset: -
- glioma_tumor(100 files) - meningioma_tumor(115 files) - no_tumor(105 files) - pituitary_tumor(74 files)

Four directories altogether make up our training dataset: - - glioma_tumor(826 files) - meningioma_tumor(822 files) - no_tumor(395 files) - pituitary_tumor(827 files)

This is the dataset link which I have used for training and testing.

https://www.kaggle.com/datasets/sartajbhuvaji/brain-tumor-classification-mri?resource=download

6.3 PRELIMINARY ANALYSIS

I used CNN to conduct image classification on the brain tumour dataset in this jupyter notebook. A neural network trained on this short dataset won't actually provide us with a good result. As a result, I'm going to train the model using the Transfer Learning approach to provide incredibly accurate results.

6.4 METHODS

Machine learning methods or deep learning models are utilised to find brain tumour behaviour in a set of brain images or data. The model can learn patterns and characteristics that are typical of brain tumour activities since it is often trained on a dataset that includes both tumour and no tumour images. Creating a fraud detection model involves a number of different elements, such as: - - **Data Import:** The data is divided into two folders, testing and training. Both contain photos in 4 subfolders. - **Data preprocessing:** It is important to make sure that the data is in a format that machine learning algorithms can use after it has been collected. This may involve the use of data cleansing, feature engineering, outlier

reduction, and data normalisation. - **Feature Extraction:** Feature extraction is the process of identifying pertinent qualities or characteristics that can be utilised to distinguish between morally upright and dishonest behaviours. - **Model fitting:** Following data cleaning, the data have been fitted to the model. When developing a machine learning model, the following algorithms can be used: - - Logistic Regression: 0.963 - SVM: 0.959 - CNN: 0.979

6.5 CONCLUSION

Early detection of brain tumors can play a significant role in preventing higher mortality rates globally. Due to the tumor's form, changing size, and structure, the correct detection of brain tumors is still highly challenging. Clinical diagnosis and therapy decisionmaking for brain tumor patients are greatly influenced by the classification of MR images. Early brain tumor identification using MR images and the tumor segmentation method appear promising. Nevertheless, there is still a long way to go before the tumor location can be precisely recognized and categorized. For the purposes of early brain tumor detection in our study, we used a variety of MRI brain tumor images. Deep learning models also have a significant impact on classification and detection. We proposed a CNN model for the early detection of brain tumors, where we obtained promising result using a large amount of MR images. We employed a variety of indicators to ensure the efficiency of the ML models during the evaluation process. In addition to the proposed model, we also took into account a few other ML models to assess our outcomes. Regarding the limitations of our research, as the CNN had several layers and the computer did not have a good GPU, the training process took a long time. If the dataset is large, such as having a thousand images, it would take more time to train. After improving our GPU system, we minimized the training time. Future work can be performed to better correctly identify brain cancers by using individual patient information gathered from any source.

6.6 CODE

```
[ ]: import warnings
     warnings.filterwarnings('ignore')
```

```
[ ]:
```

```
#Importing required libraries.
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

import tensorflow as tf
from tensorflow.keras.layers import Conv2D,Input, ZeroPadding2D,␣
↪BatchNormalization, Activation, MaxPool2D, Flatten,Dense,Dropout
from tensorflow.keras.models import Model,load_model
from tensorflow.keras.callbacks import TensorBoard, ModelCheckpoint
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
from sklearn.utils import shuffle
import imutils
from os import listdir
import numpy as np
import time
import matplotlib.pyplot as plt
import cv2
import os

from google.colab import drive
drive.mount("/content/gdrive")
```

```
Mounted at /content/gdrive
```

Data Preprocessing

Cropping brain from an image using contours

[ ]:

```
#Loading the Dataset
import os

path = os.listdir('/content/gdrive/MyDrive/MA981-7-FY/Dataset/
 ↪Training/')
classes = {'no_tumor':0, 'pituitary_tumor':1}
```

[ ]:
```
dataset_path = '/content/gdrive/MyDrive/MA981-7-FY/Dataset/
 ↪Training/'

folders = os.listdir(dataset_path)
print(folders)
```

['pituitary_tumor', 'glioma_tumor', 'no_tumor', 'meningioma_tumor']

[ ]:
```
subdirs = os.listdir(dataset_path)[:2]
for subdir in subdirs:
    print(f"{subdir} contains {len(os.listdir(dataset_path+'/
 ↪'+subdir))} images")
```

pituitary_tumor contains 827 images
glioma_tumor contains 826 images

[ ]:
```
def load_images(folder):

    imgs = []
    target = 0
    labels = []
    for i in os.listdir(folder):
        subdir = os.path.join(folder, i)
        for j in os.listdir(subdir):
            img_dir = os.path.join(subdir,j)
            try:
```

```
                img = cv2.imread(img_dir)
                img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
                img = cv2.resize(img, (128,128))
                imgs.append(img)
                labels.append(target)
            except:
                continue
        target += 1

    imgs = np.array(imgs)
    labels = np.array(labels)

    return imgs, labels
```

[ ]:
```
data, labels = load_images(dataset_path)
data.shape, labels.shape
```

[breakable, size=fbox, boxrule=.5pt, pad at break*=1mm, opacityfill=0]
```
((2870, 128, 128), (2870,))
```

[ ]:
```
def plot_images(start, end):
    plt.figure(figsize=(22,8))
    for i in range(10):
        axs = plt.subplot(2,5, i+1)
        idx = np.random.randint(start, end)
        plt.imshow(data[idx], cmap='gray')
        plt.axis('on')
        axs.set_xticklabels([])
        axs.set_yticklabels([])
        plt.subplots_adjust(wspace=None, hspace=None)
```

[ ]:
```
import cv2
X = []   # For images
```

```
Y = []  # For classes
for cls in classes:
    pth = '/content/gdrive/MyDrive/MA981-7-FY/Dataset/Training/
 ↪'+cls
    for j in os.listdir(pth):
        img = cv2.imread(pth+'/'+j, 0)
        img = cv2.resize(img, (200,200))
        X.append(img)
        Y.append(classes[cls])
```

[ ]:
```
X = np.array(X)
Y = np.array(Y)
X_updated = X.reshape(len(X), -1)
```

[ ]:
```
np.unique(Y)
```

[breakable, size=fbox, boxrule=.5pt, pad at break*=1mm, opacityfill=0]
```
array([0, 1])
```

[ ]:
```
pd.Series(Y).value_counts()
```

[breakable, size=fbox, boxrule=.5pt, pad at break*=1mm, opacityfill=0]
```
1    827
0    395
dtype: int64
```

[ ]:
```
X.shape, X_updated.shape
```

[breakable, size=fbox, boxrule=.5pt, pad at break*=1mm, opacityfill=0]
```
((1222, 200, 200), (1222, 40000))
```

6.7 Visualizing the data

```
[ ]: def load_data(dir_list,image_size):
       X=[]
       Y=[]
       image_width,image_height=image_size

       for directory in listdir(dir_list):
         for filename in listdir(dir_list+"/"+directory):
           #image=cv2.imread(dir_list+"/"+directory+"/"+filename)
           image=crop_brain_contours(dir_list+"/"+directory+"/
     ↪"+filename,False)
           image=cv2.
     ↪resize(image,(image_width,image_height),interpolation=cv2.
     ↪INTER_CUBIC)
           image=image/255
           X.append(image)
           if directory=="yes":
             Y.append([1])
           else:
             Y.append([0])
       X,Y=shuffle(X,Y)
       X=np.array(X)
       Y=np.array(Y)
       print(f"Number of examples : {len(X)}")
       print(f"X shape is : {X.shape}")
       print(f"Y shape is : {Y.shape}")

       return X,Y
```

```
[ ]:
    plt.imshow(X[0], cmap='gray')
```

[breakable, size=fbox, boxrule=.5pt, pad at break*=1mm, opacityfill=0]

```
<matplotlib.image.AxesImage at 0x7dfdebaa9510>
```

max size=0.90.9output$_1$7$_1$.*png*

6.8 Preparing the data

```
[ ]: X_updated = X.reshape(len(X), -1)
     X_updated.shape
```

[breakable, size=fbox, boxrule=.5pt, pad at break*=1mm, opacityfill=0]

```
(1222, 40000)
```

6.9 Spliting the Data

```
[ ]: xtrain, xtest, ytrain, ytest = train_test_split(X_updated, Y,␣
      ↪random_state=10,
                                                           test_size=.20)
```

```
[ ]: xtrain.shape, xtest.shape
```

[breakable, size=fbox, boxrule=.5pt, pad at break*=1mm, opacityfill=0]

```
((977, 40000), (245, 40000))
```

6.10 Feature Scaling

```
[ ]: print(xtrain.max(), xtrain.min())
     print(xtest.max(), xtest.min())
     xtrain = xtrain/255
     xtest = xtest/255
     print(xtrain.max(), xtrain.min())
     print(xtest.max(), xtest.min())
```

```
255 0
255 0
1.0 0.0
1.0 0.0
```

6.11 Feature Selection (using PCA)

[ ]:
```
from sklearn.decomposition import PCA
```

[ ]:
```
print(xtrain.shape, xtest.shape)


pca = PCA(.98)
#pca_train = pca.fit_transform(xtrain)
#pca_test = pca.transform(xtest)
pca_train = xtrain
pca_test = xtest
```

```
(977, 40000) (245, 40000)
```

6.12 Training The Model

In this, I've used the methods below to train my model: - - Logistic Regression - SVMs, or support vector machines - Convolutional Neural Networks or CNN

6.13 Logistic Regression & Support Vector Machine Classification

[ ]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
```

[ ]:
```
lg = LogisticRegression(C=0.1)
lg.fit(xtrain, ytrain)
```

[breakable, size=fbox, boxrule=.5pt, pad at break*=1mm, opacityfill=0]
```
LogisticRegression(C=0.1)
```

[ ]:
```
sv = SVC()
sv.fit(xtrain, ytrain)
```

[breakable, size=fbox, boxrule=.5pt, pad at break*=1mm, opacityfill=0]
```
SVC()
```

6.14 Evaluation [Logistic Regression]

```
[ ]: print("Training Score:", lg.score(xtrain, ytrain))
     print("Testing Score:", lg.score(xtest, ytest))

     Training Score: 1.0
     Testing Score: 0.963265306122449
```

6.15 Evaluation [SVM]

```
[ ]: print("Training Score:", sv.score(xtrain, ytrain))
     print("Testing Score:", sv.score(xtest, ytest))

     Training Score: 0.9918116683725691
     Testing Score: 0.9673469387755103
```

6.16 Prediction

```
[ ]: pred = sv.predict(xtest)
```

```
[ ]:
     misclassified=np.where(ytest!=pred)
     misclassified

     [breakable, size=fbox, boxrule=.5pt, pad at break*=1mm, opacityfill=0]
     (array([ 41,  51,  52,  73,  76,  99, 112, 212]),)
```

```
[ ]:
     print("Total Misclassified Samples: ",len(misclassified[0]))
     print(pred[36],ytest[36])

     Total Misclassified Samples:  8
     1 1
```

Our model is able to perfectly detect 236 brain tumor out of 245

6.17 TEST MODEL

[ ]:
```
dec = {0:'No Tumor', 1:'Positive Tumor'}
```

[ ]:
```
plt.figure(figsize=(12,8))
p = os.listdir('/content/gdrive/MyDrive/MA981-7-FY/Dataset/Testing/
 ↪')
c=1
for i in os.listdir('/content/gdrive/MyDrive/MA981-7-FY/Dataset/
 ↪Testing/no_tumor/')[:9]:
    plt.subplot(3,3,c)

    img = cv2.imread('/content/gdrive/MyDrive/MA981-7-FY/Dataset/
 ↪Testing/no_tumor/'+i,0)
    img1 = cv2.resize(img, (200,200))
    img1 = img1.reshape(1,-1)/255
    p = sv.predict(img1)
    plt.title(dec[p[0]])
    plt.imshow(img, cmap='gray')
    plt.axis('off')
    c+=1
```

max size=0.90.9output$_4$4$_0$.*png*

[ ]:
```
plt.figure(figsize=(12,8))
p = os.listdir('/content/gdrive/MyDrive/MA981-7-FY/Dataset/
 ↪Training/')
c=1
for i in os.listdir('/content/gdrive/MyDrive/MA981-7-FY/Dataset/
 ↪Training/pituitary_tumor/')[:16]:
    plt.subplot(4,4,c)
```

```
    img = cv2.imread('/content/gdrive/MyDrive/MA981-7-FY/Dataset/
↪Training/pituitary_tumor/'+i,0)
    img1 = cv2.resize(img, (200,200))
    img1 = img1.reshape(1,-1)/255
    p = sv.predict(img1)
    plt.title(dec[p[0]])
    plt.imshow(img, cmap='gray')
    plt.axis('off')
    c+=1
```

max size=0.90.9output$_4$5$_0$.*png*

Cropping brain from image using contours

[ ]:

```
def crop_brain_contours( image ,plot=False):
  # load the image, convert it to grayscale, and blur it slightly
  image = cv2.imread(image)
  gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
  gray = cv2.GaussianBlur(gray, (5, 5), 0)
  # threshold the image, then perform a series of erosions +
  # dilations to remove any small regions of noise
  thresh = cv2.threshold(gray, 45, 255, cv2.THRESH_BINARY)[1]
  thresh = cv2.erode(thresh, None, iterations=2)
  thresh = cv2.dilate(thresh, None, iterations=2)
  # find contours in thresholded image, then grab the largest
  # one
  cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,cv2.
↪CHAIN_APPROX_SIMPLE)
  cnts = imutils.grab_contours(cnts)
  c = max(cnts, key=cv2.contourArea)
  # determine the most extreme points along the contour
  extLeft = tuple(c[c[:, :, 0].argmin()][0])
  extRight = tuple(c[c[:, :, 0].argmax()][0])
```

```
    extTop = tuple(c[c[:, :, 1].argmin()][0])
    extBot = tuple(c[c[:, :, 1].argmax()][0])
    new_image = image[extTop[1]:extBot[1], extLeft[0]:extRight[0]]

    if plot:
      plt.figure()
      plt.subplot(1,2,1)
      plt.imshow(image)
      plt.
   ↪tick_params(axis="both",which="both",top=False,bottom=False,left=False,ri
      plt.title("Original_image")
      plt.show()
      plt.figure()
      plt.subplot(1,2,2)
      plt.imshow(new_image)
      plt.
   ↪tick_params(axis="both",which="both",top=False,bottom=False,left=False,ri
      plt.title("Cropped_image")
      plt.show()

    return image
```

“‘ # This is formatted as code Example of a image that is cropped

[ ]:
```
example_image="/content/gdrive/MyDrive/MA981-7-FY/Dataset/
 ↪augmented_data/no/aug_1 no._0_1061..jpg"
example_new_image=crop_brain_contours(example_image,True)
```

max size=0.90.9output$_{4}9_{0}.png$

max size=0.90.9output$_{4}9_{1}.png$

Loading Data

1.Reading and cropping. 2.Resizing to give input to neural network. 3.Applying normalization. 3.Appending images to X and labels to Y. 4.Shuffling X and Y.

[ ]:
```
def load_data(dir_list,image_size):
  X=[]
  Y=[]
  image_width,image_height=image_size

  for directory in listdir(dir_list):
    for filename in listdir(dir_list+"/"+directory):
      #image=cv2.imread(dir_list+"/"+directory+"/"+filename)
      image=crop_brain_contours(dir_list+"/"+directory+"/
↪"+filename,False)
      image=cv2.
↪resize(image,(image_width,image_height),interpolation=cv2.
↪INTER_CUBIC)
      image=image/255
      X.append(image)
      if directory=="yes":
        Y.append([1])
      else:
        Y.append([0])
  X,Y=shuffle(X,Y)
  X=np.array(X)
  Y=np.array(Y)
  print(f"Number of examples : {len(X)}")
  print(f"X shape is : {X.shape}")
  print(f"Y shape is : {Y.shape}")

  return X,Y
```

Storing augumented data in new directory named augumented_data

[ ]:

```
image_width,image_height=(240,240)
augumented_data_path="/content/gdrive/MyDrive/MA981-7-FY/Dataset/
 ↪augmented_data"
X,Y=load_data(augumented_data_path,(image_width,image_height))
```

```
Number of examples : 2065
X shape is : (2065, 240, 240, 3)
Y shape is : (2065, 1)
```

[ ]:
```
def plot_images(X,Y,n=50):
  # n is number of examples to plot
  for labels in [0,1]:
    images = X[np.argwhere(Y == labels)]
    n_images = images[:n]
    columns_n=10
    rows_n=int(n/columns_n)
    plt.figure(figsize=(20,10))
    i=1 # current plot
    for image in n_images:
        plt.subplot(rows_n, columns_n, i)
        plt.imshow(image[0])
        # remove ticks
        plt.tick_params(axis='both', which='both', top=False,␣
 ↪bottom=False, left=False, right=False,labelbottom=False,␣
 ↪labeltop=False, labelleft=False, labelright=False)
        i += 1
    label_to_str = lambda label: "Yes" if label == 1 else "No"
    plt.suptitle(f"Brain Tumor: {label_to_str(labels)}")
    plt.show()
```

[ ]:
```
plot_images(X,Y)
```

max size=0.90.9output$_5$5$_0$.*png*

max size=0.90.9output$_5$5$_1$.*png*

Splitting data into training, cross-validation and test set

[ ]:
```
def split_data(X, Y, test_size=0.2):

    """
    Splits data into training, development and test sets.
    Arguments:
        X: A numpy array with shape = (#_examples, image_width,␣
↪image_height, #_channels)
        Y: A numpy array with shape = (#_examples, 1)
    Returns:
        X_train: A numpy array with shape = (#_train_examples,␣
↪image_width, image_height, #_channels)
        y_train: A numpy array with shape = (#_train_examples, 1)
        X_val: A numpy array with shape = (#_val_examples,␣
↪image_width, image_height, #_channels)
        y_val: A numpy array with shape = (#_val_examples, 1)
        X_test: A numpy array with shape = (#_test_examples,␣
↪image_width, image_height, #_channels)
        y_test: A numpy array with shape = (#_test_examples, 1)
    """

    X_train, X_test_val, Y_train, Y_test_val = train_test_split(X,␣
↪Y, test_size=test_size)
    X_test, X_val, Y_test, Y_val = train_test_split(X_test_val,␣
↪Y_test_val, test_size=0.5)

    return X_train, Y_train, X_val, Y_val, X_test, Y_test
```

Let's use the following way to split:

1.70% of the data for training. 2.15% of the data for validation. 3.15% of the data for

testing.

[ ]:
```
X_train, Y_train, X_val, Y_val, X_test, Y_test = split_data(X, Y,␣
↪test_size=0.3)
```

[ ]:
```
print ("number of training examples = " + str(X_train.shape[0]))
print ("number of development examples = " + str(X_val.shape[0]))
print ("number of test examples = " + str(X_test.shape[0]))
print ("X_train shape: " + str(X_train.shape))
print ("Y_train shape: " + str(Y_train.shape))
print ("X_val (dev) shape: " + str(X_val.shape))
print ("Y_val (dev) shape: " + str(Y_val.shape))
print ("X_test shape: " + str(X_test.shape))
print ("Y_test shape: " + str(Y_test.shape))
```

```
number of training examples = 1445
number of development examples = 310
number of test examples = 310
X_train shape: (1445, 240, 240, 3)
Y_train shape: (1445, 1)
X_val (dev) shape: (310, 240, 240, 3)
Y_val (dev) shape: (310, 1)
X_test shape: (310, 240, 240, 3)
Y_test shape: (310, 1)
```

[ ]:
```
#nicely formatted time string
def  hms_string(sec_elapsed):
  h=int(sec_elapsed/(60*60))
  m=int((sec_elapsed%(60*60))/60)
  s=sec_elapsed%60
  return f"{h}:{m}:{round(s,1)}"
```

[ ]:

```
    def compute_f1_score(y_true, prob):
        # convert the vector of probabilities to a target vector
        y_pred = np.where(prob > 0.5, 1, 0)

        score = f1_score(y_true, y_pred)

        return score
```

```
[ ]:
    def data_percentage(y):

        m=len(y)
        n_positive = np.sum(y)
        n_negative = m - n_positive

        pos_prec = (n_positive* 100.0)/ m
        neg_prec = (n_negative* 100.0)/ m

        print(f"Number of examples: {m}")
        print(f"Percentage of positive examples: {pos_prec}%, number␣
     ↪of pos examples: {n_positive}")
        print(f"Percentage of negative examples: {neg_prec}%, number␣
     ↪of neg examples: {n_negative}")
```

```
[ ]:
    def plot_metrics(history):

        train_loss = history['loss']
        val_loss = history['val_loss']
        train_acc = history['accuracy']
        val_acc = history['val_accuracy']

        # Loss
        plt.figure()
```

```
    plt.plot(train_loss, label='Training Loss')
    plt.plot(val_loss, label='Validation Loss')
    plt.title('Loss')
    plt.legend()
    plt.show()

    # Accuracy
    plt.figure()
    plt.plot(train_acc, label='Training Accuracy')
    plt.plot(val_acc, label='Validation Accuracy')
    plt.title('Accuracy')
    plt.legend()
    plt.show()
```

6.18 DEEP TRANSFER LEARNING ON MEDICAL IMAGES

Research and work carried out by Brizzi Anouk and Oxisoglou Thomas as part of a research project supervised by Mr. Ronan Sicre for the end of the first year of the Master's program at the Faculty of Aix-Marseille, IAAA option.

Its aim is to test the transfer learning method on medical image classification. To do this, we used a brain tumor dataset, available here : https://www.kaggle.com/datasets/sartajbhuvaji/brain-tumor-classification-mri

This document presents the results of our best model.

Using VGG16(without fully connected layers) with parameters already trainned on Imagenet.

[ ]:
```
vgg16=tf.keras.applications.VGG16(include_top=False,␣
↪weights='imagenet', input_shape=(240,240,3),pooling=max)

Downloading data from https://storage.googleapis.com/tensorflow/
↪keras-
```

```
applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.
 ↪h5
58889256/58889256 [==============================] - 4s 0us/step
```

Adding some trainnable layers to the VGG16 model

[ ]:
```
VGG16=tf.keras.Sequential()
VGG16.add(vgg16)
VGG16.add(Dropout(0.3))
VGG16.add(Flatten())
VGG16.add(Dropout(0.5))
VGG16.add(Dense(1, activation='sigmoid'))
VGG16.layers[0].trainable = False
VGG16.compile(optimizer='adam', loss='binary_crossentropy',␣
 ↪metrics=['accuracy'])
```

[ ]:
```
VGG16.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 7, 7, 512)         14714688

 dropout (Dropout)           (None, 7, 7, 512)         0

 flatten (Flatten)           (None, 25088)             0

 dropout_1 (Dropout)         (None, 25088)             0

 dense (Dense)               (None, 1)                 25089

=================================================================
Total params: 14,739,777
```

```
Trainable params: 25,089
Non-trainable params: 14,714,688
```

Saving VGG models

[ ]:
```
checkpoint_path = "/content/gdrive/MyDrive/MA981-7-FY/Dataset/
↪VGGmodels/{epoch:03d}-{val_accuracy:.2f}"#"models/{epoch:
↪03d}-{val_loss:.2f}.h5"#"training_1/cp.ckpt"
checkpoint =␣
↪ModelCheckpoint(filepath=checkpoint_path,monitor="val_accuracy",␣
↪verbose=1, save_best_only=True, mode='auto',save_freq="epoch")
```

[ ]:
```
import time
start_time = time.time()

VGG16.fit(x=X_train, y=Y_train, batch_size=32, epochs=125,␣
↪validation_data=(X_val, Y_val),callbacks=[checkpoint])

end_time = time.time()
execution_time = (end_time - start_time)
print(f"Elapsed time: {hms_string(execution_time)}")
```

```
Epoch 1/125
46/46 [==============================] - ETA: 0s - loss: 0.5698 -␣
↪accuracy:
0.7218
Epoch 1: val_accuracy improved from -inf to 0.84194, saving model␣
↪to
/content/gdrive/MyDrive/MA981-7-FY/Dataset/VGGmodels/001-0.84

WARNING:absl:Found untraced functions such as␣
↪_jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
```

```
_jit_compiled_convolution_op, _jit_compiled_convolution_op while␣
 ↪saving (showing
5 of 13). These functions will not be directly callable after␣
 ↪loading.

46/46 [==============================] - 34s 414ms/step - loss: 0.
 ↪5698 -
accuracy: 0.7218 - val_loss: 0.3572 - val_accuracy: 0.8419
Epoch 2/125
45/46 [============================>.] - ETA: 0s - loss: 0.3684 -␣
 ↪accuracy:
0.8396
Epoch 2: val_accuracy improved from 0.84194 to 0.87742, saving␣
 ↪model to
/content/gdrive/MyDrive/MA981-7-FY/Dataset/VGGmodels/002-0.88

WARNING:absl:Found untraced functions such as␣
 ↪_jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while␣
 ↪saving (showing
5 of 13). These functions will not be directly callable after␣
 ↪loading.

46/46 [==============================] - 12s 259ms/step - loss: 0.
 ↪3675 -
accuracy: 0.8401 - val_loss: 0.2873 - val_accuracy: 0.8774
Epoch 3/125
45/46 [============================>.] - ETA: 0s - loss: 0.2648 -␣
 ↪accuracy:
0.8882
Epoch 3: val_accuracy improved from 0.87742 to 0.89355, saving␣
 ↪model to
/content/gdrive/MyDrive/MA981-7-FY/Dataset/VGGmodels/003-0.89
```

```
WARNING:absl:Found untraced functions such as␣
 ↪_jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while␣
 ↪saving (showing
5 of 13). These functions will not be directly callable after␣
 ↪loading.

46/46 [==============================] - 14s 314ms/step - loss: 0.
 ↪2644 -
accuracy: 0.8886 - val_loss: 0.2670 - val_accuracy: 0.8935
Epoch 4/125
45/46 [============================>.] - ETA: 0s - loss: 0.2177 -␣
 ↪accuracy:
0.9194
Epoch 4: val_accuracy improved from 0.89355 to 0.91613, saving␣
 ↪model to
/content/gdrive/MyDrive/MA981-7-FY/Dataset/VGGmodels/004-0.92

WARNING:absl:Found untraced functions such as␣
 ↪_jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while␣
 ↪saving (showing
5 of 13). These functions will not be directly callable after␣
 ↪loading.

46/46 [==============================] - 14s 308ms/step - loss: 0.
 ↪2173 -
accuracy: 0.9197 - val_loss: 0.2247 - val_accuracy: 0.9161
Epoch 5/125
45/46 [============================>.] - ETA: 0s - loss: 0.1835 -␣
 ↪accuracy:
```

```
0.9368
Epoch 5: val_accuracy improved from 0.91613 to 0.91935, saving␣
 ↪model to
/content/gdrive/MyDrive/MA981-7-FY/Dataset/VGGmodels/005-0.92

WARNING:absl:Found untraced functions such as␣
 ↪_jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while␣
 ↪saving (showing
5 of 13). These functions will not be directly callable after␣
 ↪loading.

46/46 [==============================] - 12s 255ms/step - loss: 0.
 ↪1831 -
accuracy: 0.9370 - val_loss: 0.2274 - val_accuracy: 0.9194
Epoch 6/125
45/46 [============================>.] - ETA: 0s - loss: 0.1589 -␣
 ↪accuracy:
0.9417
Epoch 6: val_accuracy improved from 0.91935 to 0.92258, saving␣
 ↪model to
/content/gdrive/MyDrive/MA981-7-FY/Dataset/VGGmodels/006-0.92

WARNING:absl:Found untraced functions such as␣
 ↪_jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while␣
 ↪saving (showing
5 of 13). These functions will not be directly callable after␣
 ↪loading.

46/46 [==============================] - 15s 327ms/step - loss: 0.
 ↪1592 -
```

```
accuracy: 0.9412 - val_loss: 0.2125 - val_accuracy: 0.9226
Epoch 7/125
45/46 [============================>.] - ETA: 0s - loss: 0.1522 -␣
 ↪accuracy:
0.9465
Epoch 7: val_accuracy improved from 0.92258 to 0.92903, saving␣
 ↪model to
/content/gdrive/MyDrive/MA981-7-FY/Dataset/VGGmodels/007-0.93

WARNING:absl:Found untraced functions such as␣
 ↪_jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while␣
 ↪saving (showing
5 of 13). These functions will not be directly callable after␣
 ↪loading.

46/46 [==============================] - 12s 268ms/step - loss: 0.
 ↪1521 -
accuracy: 0.9467 - val_loss: 0.1947 - val_accuracy: 0.9290
Epoch 8/125
45/46 [============================>.] - ETA: 0s - loss: 0.1348 -␣
 ↪accuracy:
0.9535
Epoch 8: val_accuracy did not improve from 0.92903
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪1345 -
accuracy: 0.9536 - val_loss: 0.1887 - val_accuracy: 0.9194
Epoch 9/125
45/46 [============================>.] - ETA: 0s - loss: 0.1209 -␣
 ↪accuracy:
0.9625
Epoch 9: val_accuracy did not improve from 0.92903
```

```
46/46 [==============================] - 9s 205ms/step - loss: 0.
↪1207 -
accuracy: 0.9626 - val_loss: 0.2604 - val_accuracy: 0.8677
Epoch 10/125
45/46 [============================>.] - ETA: 0s - loss: 0.1154 -␣
↪accuracy:
0.9611
Epoch 10: val_accuracy improved from 0.92903 to 0.93871, saving␣
↪model to
/content/gdrive/MyDrive/MA981-7-FY/Dataset/VGGmodels/010-0.94

WARNING:absl:Found untraced functions such as␣
↪_jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while␣
↪saving (showing
5 of 13). These functions will not be directly callable after␣
↪loading.

46/46 [==============================] - 12s 268ms/step - loss: 0.
↪1152 -
accuracy: 0.9612 - val_loss: 0.1702 - val_accuracy: 0.9387
Epoch 11/125
45/46 [============================>.] - ETA: 0s - loss: 0.0987 -␣
↪accuracy:
0.9722
Epoch 11: val_accuracy did not improve from 0.93871
46/46 [==============================] - 9s 205ms/step - loss: 0.
↪0984 -
accuracy: 0.9723 - val_loss: 0.1664 - val_accuracy: 0.9387
Epoch 12/125
45/46 [============================>.] - ETA: 0s - loss: 0.0998 -␣
↪accuracy:
```

```
0.9660
Epoch 12: val_accuracy did not improve from 0.93871
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0998 -
accuracy: 0.9661 - val_loss: 0.1640 - val_accuracy: 0.9387
Epoch 13/125
45/46 [============================>.] - ETA: 0s - loss: 0.0842 -␣
 ↪accuracy:
0.9799
Epoch 13: val_accuracy improved from 0.93871 to 0.94194, saving␣
 ↪model to
/content/gdrive/MyDrive/MA981-7-FY/Dataset/VGGmodels/013-0.94

WARNING:absl:Found untraced functions such as␣
 ↪_jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while␣
 ↪saving (showing
5 of 13). These functions will not be directly callable after␣
 ↪loading.

46/46 [==============================] - 12s 271ms/step - loss: 0.
 ↪0842 -
accuracy: 0.9799 - val_loss: 0.1605 - val_accuracy: 0.9419
Epoch 14/125
45/46 [============================>.] - ETA: 0s - loss: 0.0870 -␣
 ↪accuracy:
0.9743
Epoch 14: val_accuracy did not improve from 0.94194
46/46 [==============================] - 9s 201ms/step - loss: 0.
 ↪0868 -
accuracy: 0.9744 - val_loss: 0.1605 - val_accuracy: 0.9387
Epoch 15/125
```

```
45/46 [============================>.] - ETA: 0s - loss: 0.0778 -␣
 ↪accuracy:
0.9785
Epoch 15: val_accuracy did not improve from 0.94194
46/46 [==============================] - 9s 201ms/step - loss: 0.
 ↪0775 -
accuracy: 0.9785 - val_loss: 0.1513 - val_accuracy: 0.9419
Epoch 16/125
45/46 [============================>.] - ETA: 0s - loss: 0.0691 -␣
 ↪accuracy:
0.9840
Epoch 16: val_accuracy did not improve from 0.94194
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0690 -
accuracy: 0.9841 - val_loss: 0.1704 - val_accuracy: 0.9258
Epoch 17/125
45/46 [============================>.] - ETA: 0s - loss: 0.0597 -␣
 ↪accuracy:
0.9847
Epoch 17: val_accuracy did not improve from 0.94194
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0595 -
accuracy: 0.9848 - val_loss: 0.1551 - val_accuracy: 0.9419
Epoch 18/125
45/46 [============================>.] - ETA: 0s - loss: 0.0677 -␣
 ↪accuracy:
0.9778
Epoch 18: val_accuracy improved from 0.94194 to 0.94516, saving␣
 ↪model to
/content/gdrive/MyDrive/MA981-7-FY/Dataset/VGGmodels/018-0.95
```

```
WARNING:absl:Found untraced functions such as␣
 ↪_jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while␣
 ↪saving (showing
5 of 13). These functions will not be directly callable after␣
 ↪loading.

46/46 [==============================] - 12s 263ms/step - loss: 0.
 ↪0675 -
accuracy: 0.9779 - val_loss: 0.1468 - val_accuracy: 0.9452
Epoch 19/125
45/46 [============================>.] - ETA: 0s - loss: 0.0613 -␣
 ↪accuracy:
0.9840
Epoch 19: val_accuracy did not improve from 0.94516
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0612 -
accuracy: 0.9841 - val_loss: 0.1536 - val_accuracy: 0.9419
Epoch 20/125
45/46 [============================>.] - ETA: 0s - loss: 0.0502 -␣
 ↪accuracy:
0.9875
Epoch 20: val_accuracy improved from 0.94516 to 0.94839, saving␣
 ↪model to
/content/gdrive/MyDrive/MA981-7-FY/Dataset/VGGmodels/020-0.95

WARNING:absl:Found untraced functions such as␣
 ↪_jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while␣
 ↪saving (showing
```

```
5 of 13). These functions will not be directly callable after␣
 ↪loading.

46/46 [==============================] - 12s 271ms/step - loss: 0.
 ↪0501 -
accuracy: 0.9875 - val_loss: 0.1447 - val_accuracy: 0.9484
Epoch 21/125
45/46 [============================>.] - ETA: 0s - loss: 0.0597 -␣
 ↪accuracy:
0.9833
Epoch 21: val_accuracy did not improve from 0.94839
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0596 -
accuracy: 0.9834 - val_loss: 0.1544 - val_accuracy: 0.9387
Epoch 22/125
45/46 [============================>.] - ETA: 0s - loss: 0.0578 -␣
 ↪accuracy:
0.9806
Epoch 22: val_accuracy did not improve from 0.94839
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0576 -
accuracy: 0.9806 - val_loss: 0.1518 - val_accuracy: 0.9355
Epoch 23/125
45/46 [============================>.] - ETA: 0s - loss: 0.0597 -␣
 ↪accuracy:
0.9840
Epoch 23: val_accuracy did not improve from 0.94839
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0599 -
accuracy: 0.9841 - val_loss: 0.1493 - val_accuracy: 0.9419
Epoch 24/125
```

```
45/46 [============================>.] - ETA: 0s - loss: 0.0533 -␣
 ↪accuracy:
0.9847
Epoch 24: val_accuracy did not improve from 0.94839
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0537 -
accuracy: 0.9848 - val_loss: 0.1465 - val_accuracy: 0.9387
Epoch 25/125
45/46 [============================>.] - ETA: 0s - loss: 0.0578 -␣
 ↪accuracy:
0.9819
Epoch 25: val_accuracy did not improve from 0.94839
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0576 -
accuracy: 0.9820 - val_loss: 0.1571 - val_accuracy: 0.9387
Epoch 26/125
45/46 [============================>.] - ETA: 0s - loss: 0.0504 -␣
 ↪accuracy:
0.9868
Epoch 26: val_accuracy did not improve from 0.94839
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0504 -
accuracy: 0.9869 - val_loss: 0.1363 - val_accuracy: 0.9484
Epoch 27/125
45/46 [============================>.] - ETA: 0s - loss: 0.0449 -␣
 ↪accuracy:
0.9875
Epoch 27: val_accuracy improved from 0.94839 to 0.95161, saving␣
 ↪model to
/content/gdrive/MyDrive/MA981-7-FY/Dataset/VGGmodels/027-0.95
```

```
WARNING:absl:Found untraced functions such as␣
 ↪_jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while␣
 ↪saving (showing
5 of 13). These functions will not be directly callable after␣
 ↪loading.

46/46 [==============================] - 12s 262ms/step - loss: 0.
 ↪0448 -
accuracy: 0.9875 - val_loss: 0.1318 - val_accuracy: 0.9516
Epoch 28/125
45/46 [============================>.] - ETA: 0s - loss: 0.0494 -␣
 ↪accuracy:
0.9882
Epoch 28: val_accuracy did not improve from 0.95161
46/46 [==============================] - 9s 201ms/step - loss: 0.
 ↪0495 -
accuracy: 0.9882 - val_loss: 0.1399 - val_accuracy: 0.9484
Epoch 29/125
45/46 [============================>.] - ETA: 0s - loss: 0.0559 -␣
 ↪accuracy:
0.9792
Epoch 29: val_accuracy did not improve from 0.95161
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0559 -
accuracy: 0.9792 - val_loss: 0.1310 - val_accuracy: 0.9484
Epoch 30/125
45/46 [============================>.] - ETA: 0s - loss: 0.0429 -␣
 ↪accuracy:
0.9875
Epoch 30: val_accuracy did not improve from 0.95161
```

```
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0430 -
accuracy: 0.9875 - val_loss: 0.1955 - val_accuracy: 0.9194
Epoch 31/125
45/46 [============================>.] - ETA: 0s - loss: 0.0478 -␣
 ↪accuracy:
0.9812
Epoch 31: val_accuracy did not improve from 0.95161
46/46 [==============================] - 9s 204ms/step - loss: 0.
 ↪0478 -
accuracy: 0.9813 - val_loss: 0.1408 - val_accuracy: 0.9484
Epoch 32/125
45/46 [============================>.] - ETA: 0s - loss: 0.0370 -␣
 ↪accuracy:
0.9896
Epoch 32: val_accuracy did not improve from 0.95161
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0371 -
accuracy: 0.9896 - val_loss: 0.1420 - val_accuracy: 0.9484
Epoch 33/125
45/46 [============================>.] - ETA: 0s - loss: 0.0430 -␣
 ↪accuracy:
0.9861
Epoch 33: val_accuracy did not improve from 0.95161
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0429 -
accuracy: 0.9862 - val_loss: 0.1653 - val_accuracy: 0.9226
Epoch 34/125
45/46 [============================>.] - ETA: 0s - loss: 0.0442 -␣
 ↪accuracy:
0.9875
```

```
Epoch 34: val_accuracy did not improve from 0.95161
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0441 -
accuracy: 0.9875 - val_loss: 0.1474 - val_accuracy: 0.9452
Epoch 35/125
45/46 [============================>.] - ETA: 0s - loss: 0.0294 -␣
 ↪accuracy:
0.9951
Epoch 35: val_accuracy did not improve from 0.95161
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0294 -
accuracy: 0.9952 - val_loss: 0.1342 - val_accuracy: 0.9484
Epoch 36/125
45/46 [============================>.] - ETA: 0s - loss: 0.0356 -␣
 ↪accuracy:
0.9903
Epoch 36: val_accuracy did not improve from 0.95161
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0355 -
accuracy: 0.9903 - val_loss: 0.1506 - val_accuracy: 0.9419
Epoch 37/125
45/46 [============================>.] - ETA: 0s - loss: 0.0316 -␣
 ↪accuracy:
0.9917
Epoch 37: val_accuracy did not improve from 0.95161
46/46 [==============================] - 9s 201ms/step - loss: 0.
 ↪0315 -
accuracy: 0.9917 - val_loss: 0.1316 - val_accuracy: 0.9484
Epoch 38/125
45/46 [============================>.] - ETA: 0s - loss: 0.0250 -␣
 ↪accuracy:
```

```
0.9958
Epoch 38: val_accuracy did not improve from 0.95161
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0250 -
accuracy: 0.9958 - val_loss: 0.1295 - val_accuracy: 0.9516
Epoch 39/125
45/46 [============================>.] - ETA: 0s - loss: 0.0283 -␣
 ↪accuracy:
0.9924
Epoch 39: val_accuracy did not improve from 0.95161
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0282 -
accuracy: 0.9924 - val_loss: 0.1254 - val_accuracy: 0.9484
Epoch 40/125
45/46 [============================>.] - ETA: 0s - loss: 0.0300 -␣
 ↪accuracy:
0.9931
Epoch 40: val_accuracy did not improve from 0.95161
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0304 -
accuracy: 0.9931 - val_loss: 0.1289 - val_accuracy: 0.9484
Epoch 41/125
45/46 [============================>.] - ETA: 0s - loss: 0.0436 -␣
 ↪accuracy:
0.9861
Epoch 41: val_accuracy did not improve from 0.95161
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0442 -
accuracy: 0.9855 - val_loss: 0.1520 - val_accuracy: 0.9355
Epoch 42/125
45/46 [============================>.] - ETA: 0s - loss: 0.0285 -␣
 ↪accuracy:
```

```
0.9924
Epoch 42: val_accuracy did not improve from 0.95161
46/46 [==============================] - 9s 204ms/step - loss: 0.
 ↪0286 -
accuracy: 0.9924 - val_loss: 0.1560 - val_accuracy: 0.9355
Epoch 43/125
45/46 [============================>.] - ETA: 0s - loss: 0.0394 -␣
 ↪accuracy:
0.9882
Epoch 43: val_accuracy improved from 0.95161 to 0.95806, saving␣
 ↪model to
/content/gdrive/MyDrive/MA981-7-FY/Dataset/VGGmodels/043-0.96

WARNING:absl:Found untraced functions such as␣
 ↪_jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while␣
 ↪saving (showing
5 of 13). These functions will not be directly callable after␣
 ↪loading.

46/46 [==============================] - 12s 272ms/step - loss: 0.
 ↪0392 -
accuracy: 0.9882 - val_loss: 0.1262 - val_accuracy: 0.9581
Epoch 44/125
45/46 [============================>.] - ETA: 0s - loss: 0.0341 -␣
 ↪accuracy:
0.9903
Epoch 44: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 201ms/step - loss: 0.
 ↪0341 -
accuracy: 0.9903 - val_loss: 0.1181 - val_accuracy: 0.9484
Epoch 45/125
```

```
45/46 [============================>.] - ETA: 0s - loss: 0.0286 -␣
 ↪accuracy:
0.9903
Epoch 45: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0287 -
accuracy: 0.9903 - val_loss: 0.1237 - val_accuracy: 0.9516
Epoch 46/125
45/46 [============================>.] - ETA: 0s - loss: 0.0324 -␣
 ↪accuracy:
0.9889
Epoch 46: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0323 -
accuracy: 0.9889 - val_loss: 0.1372 - val_accuracy: 0.9516
Epoch 47/125
45/46 [============================>.] - ETA: 0s - loss: 0.0257 -␣
 ↪accuracy:
0.9937
Epoch 47: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0256 -
accuracy: 0.9938 - val_loss: 0.1411 - val_accuracy: 0.9452
Epoch 48/125
45/46 [============================>.] - ETA: 0s - loss: 0.0199 -␣
 ↪accuracy:
0.9958
Epoch 48: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 204ms/step - loss: 0.
 ↪0199 -
accuracy: 0.9958 - val_loss: 0.1475 - val_accuracy: 0.9452
```

```
Epoch 49/125
45/46 [============================>.] - ETA: 0s - loss: 0.0302 -␣
 ↪accuracy:
0.9917
Epoch 49: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0302 -
accuracy: 0.9917 - val_loss: 0.1387 - val_accuracy: 0.9516
Epoch 50/125
45/46 [============================>.] - ETA: 0s - loss: 0.0245 -␣
 ↪accuracy:
0.9965
Epoch 50: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0244 -
accuracy: 0.9965 - val_loss: 0.1457 - val_accuracy: 0.9452
Epoch 51/125
45/46 [============================>.] - ETA: 0s - loss: 0.0273 -␣
 ↪accuracy:
0.9910
Epoch 51: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0273 -
accuracy: 0.9910 - val_loss: 0.2118 - val_accuracy: 0.9194
Epoch 52/125
45/46 [============================>.] - ETA: 0s - loss: 0.0250 -␣
 ↪accuracy:
0.9924
Epoch 52: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 204ms/step - loss: 0.
 ↪0249 -
```

```
accuracy: 0.9924 - val_loss: 0.1284 - val_accuracy: 0.9516
Epoch 53/125
45/46 [============================>.] - ETA: 0s - loss: 0.0220 -␣
 ↪accuracy:
0.9965
Epoch 53: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0220 -
accuracy: 0.9965 - val_loss: 0.1389 - val_accuracy: 0.9516
Epoch 54/125
45/46 [============================>.] - ETA: 0s - loss: 0.0269 -␣
 ↪accuracy:
0.9931
Epoch 54: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0269 -
accuracy: 0.9931 - val_loss: 0.1431 - val_accuracy: 0.9516
Epoch 55/125
45/46 [============================>.] - ETA: 0s - loss: 0.0209 -␣
 ↪accuracy:
0.9937
Epoch 55: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0208 -
accuracy: 0.9938 - val_loss: 0.1474 - val_accuracy: 0.9516
Epoch 56/125
45/46 [============================>.] - ETA: 0s - loss: 0.0255 -␣
 ↪accuracy:
0.9924
Epoch 56: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0255 -
```

```
accuracy: 0.9924 - val_loss: 0.1291 - val_accuracy: 0.9516
Epoch 57/125
45/46 [============================>.] - ETA: 0s - loss: 0.0215 -␣
 ↪accuracy:
0.9944
Epoch 57: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0216 -
accuracy: 0.9945 - val_loss: 0.1436 - val_accuracy: 0.9484
Epoch 58/125
45/46 [============================>.] - ETA: 0s - loss: 0.0312 -␣
 ↪accuracy:
0.9903
Epoch 58: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0311 -
accuracy: 0.9903 - val_loss: 0.1326 - val_accuracy: 0.9581
Epoch 59/125
45/46 [============================>.] - ETA: 0s - loss: 0.0253 -␣
 ↪accuracy:
0.9924
Epoch 59: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0253 -
accuracy: 0.9924 - val_loss: 0.1295 - val_accuracy: 0.9548
Epoch 60/125
45/46 [============================>.] - ETA: 0s - loss: 0.0237 -␣
 ↪accuracy:
0.9917
Epoch 60: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0236 -
```

```
accuracy: 0.9917 - val_loss: 0.1258 - val_accuracy: 0.9516
Epoch 61/125
45/46 [============================>.] - ETA: 0s - loss: 0.0223 -␣
 ↪accuracy:
0.9917
Epoch 61: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0223 -
accuracy: 0.9917 - val_loss: 0.1283 - val_accuracy: 0.9484
Epoch 62/125
45/46 [============================>.] - ETA: 0s - loss: 0.0230 -␣
 ↪accuracy:
0.9944
Epoch 62: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0230 -
accuracy: 0.9945 - val_loss: 0.1329 - val_accuracy: 0.9484
Epoch 63/125
45/46 [============================>.] - ETA: 0s - loss: 0.0244 -␣
 ↪accuracy:
0.9931
Epoch 63: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0245 -
accuracy: 0.9931 - val_loss: 0.1437 - val_accuracy: 0.9419
Epoch 64/125
45/46 [============================>.] - ETA: 0s - loss: 0.0226 -␣
 ↪accuracy:
0.9924
Epoch 64: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0226 -
```

```
accuracy: 0.9924 - val_loss: 0.1462 - val_accuracy: 0.9484
Epoch 65/125
45/46 [============================>.] - ETA: 0s - loss: 0.0242 -␣
 ↪accuracy:
0.9910
Epoch 65: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0241 -
accuracy: 0.9910 - val_loss: 0.1535 - val_accuracy: 0.9419
Epoch 66/125
45/46 [============================>.] - ETA: 0s - loss: 0.0208 -␣
 ↪accuracy:
0.9917
Epoch 66: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0207 -
accuracy: 0.9917 - val_loss: 0.1647 - val_accuracy: 0.9484
Epoch 67/125
45/46 [============================>.] - ETA: 0s - loss: 0.0185 -␣
 ↪accuracy:
0.9958
Epoch 67: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0185 -
accuracy: 0.9958 - val_loss: 0.1752 - val_accuracy: 0.9452
Epoch 68/125
45/46 [============================>.] - ETA: 0s - loss: 0.0279 -␣
 ↪accuracy:
0.9917
Epoch 68: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0278 -
```

```
accuracy: 0.9917 - val_loss: 0.1502 - val_accuracy: 0.9484
Epoch 69/125
45/46 [============================>.] - ETA: 0s - loss: 0.0181 -␣
 ↪accuracy:
0.9965
Epoch 69: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0180 -
accuracy: 0.9965 - val_loss: 0.1525 - val_accuracy: 0.9484
Epoch 70/125
45/46 [============================>.] - ETA: 0s - loss: 0.0214 -␣
 ↪accuracy:
0.9937
Epoch 70: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0216 -
accuracy: 0.9938 - val_loss: 0.1529 - val_accuracy: 0.9484
Epoch 71/125
45/46 [============================>.] - ETA: 0s - loss: 0.0292 -␣
 ↪accuracy:
0.9910
Epoch 71: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0291 -
accuracy: 0.9910 - val_loss: 0.1496 - val_accuracy: 0.9419
Epoch 72/125
45/46 [============================>.] - ETA: 0s - loss: 0.0237 -␣
 ↪accuracy:
0.9924
Epoch 72: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0236 -
```

```
accuracy: 0.9924 - val_loss: 0.1580 - val_accuracy: 0.9387
Epoch 73/125
45/46 [============================>.] - ETA: 0s - loss: 0.0257 -␣
 ↪accuracy:
0.9931
Epoch 73: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0256 -
accuracy: 0.9931 - val_loss: 0.1495 - val_accuracy: 0.9452
Epoch 74/125
45/46 [============================>.] - ETA: 0s - loss: 0.0176 -␣
 ↪accuracy:
0.9937
Epoch 74: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0177 -
accuracy: 0.9938 - val_loss: 0.1429 - val_accuracy: 0.9452
Epoch 75/125
45/46 [============================>.] - ETA: 0s - loss: 0.0235 -␣
 ↪accuracy:
0.9910
Epoch 75: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0235 -
accuracy: 0.9910 - val_loss: 0.1971 - val_accuracy: 0.9258
Epoch 76/125
45/46 [============================>.] - ETA: 0s - loss: 0.0240 -␣
 ↪accuracy:
0.9896
Epoch 76: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0239 -
```

```
accuracy: 0.9896 - val_loss: 0.1478 - val_accuracy: 0.9516
Epoch 77/125
45/46 [============================>.] - ETA: 0s - loss: 0.0217 -␣
 ↪accuracy:
0.9937
Epoch 77: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0225 -
accuracy: 0.9931 - val_loss: 0.1555 - val_accuracy: 0.9484
Epoch 78/125
45/46 [============================>.] - ETA: 0s - loss: 0.0244 -␣
 ↪accuracy:
0.9910
Epoch 78: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0244 -
accuracy: 0.9910 - val_loss: 0.2041 - val_accuracy: 0.9258
Epoch 79/125
45/46 [============================>.] - ETA: 0s - loss: 0.0248 -␣
 ↪accuracy:
0.9937
Epoch 79: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0247 -
accuracy: 0.9938 - val_loss: 0.1396 - val_accuracy: 0.9516
Epoch 80/125
45/46 [============================>.] - ETA: 0s - loss: 0.0358 -␣
 ↪accuracy:
0.9889
Epoch 80: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0377 -
```

```
accuracy: 0.9875 - val_loss: 0.1648 - val_accuracy: 0.9452
Epoch 81/125
45/46 [============================>.] - ETA: 0s - loss: 0.0544 -␣
 ↪accuracy:
0.9799
Epoch 81: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 201ms/step - loss: 0.
 ↪0552 -
accuracy: 0.9792 - val_loss: 0.1480 - val_accuracy: 0.9387
Epoch 82/125
45/46 [============================>.] - ETA: 0s - loss: 0.0394 -␣
 ↪accuracy:
0.9847
Epoch 82: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0393 -
accuracy: 0.9848 - val_loss: 0.1723 - val_accuracy: 0.9323
Epoch 83/125
45/46 [============================>.] - ETA: 0s - loss: 0.0266 -␣
 ↪accuracy:
0.9910
Epoch 83: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0265 -
accuracy: 0.9910 - val_loss: 0.1668 - val_accuracy: 0.9452
Epoch 84/125
45/46 [============================>.] - ETA: 0s - loss: 0.0157 -␣
 ↪accuracy:
0.9958
Epoch 84: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0161 -
```

```
accuracy: 0.9958 - val_loss: 0.1513 - val_accuracy: 0.9419
Epoch 85/125
45/46 [============================>.] - ETA: 0s - loss: 0.0256 -␣
 ↪accuracy:
0.9903
Epoch 85: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0255 -
accuracy: 0.9903 - val_loss: 0.1531 - val_accuracy: 0.9387
Epoch 86/125
45/46 [============================>.] - ETA: 0s - loss: 0.0185 -␣
 ↪accuracy:
0.9931
Epoch 86: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0185 -
accuracy: 0.9931 - val_loss: 0.1438 - val_accuracy: 0.9548
Epoch 87/125
45/46 [============================>.] - ETA: 0s - loss: 0.0219 -␣
 ↪accuracy:
0.9910
Epoch 87: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0219 -
accuracy: 0.9910 - val_loss: 0.1473 - val_accuracy: 0.9516
Epoch 88/125
45/46 [============================>.] - ETA: 0s - loss: 0.0195 -␣
 ↪accuracy:
0.9924
Epoch 88: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0195 -
```

```
accuracy: 0.9924 - val_loss: 0.3051 - val_accuracy: 0.8968
Epoch 89/125
45/46 [============================>.] - ETA: 0s - loss: 0.0217 -␣
 ↪accuracy:
0.9944
Epoch 89: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0220 -
accuracy: 0.9945 - val_loss: 0.1386 - val_accuracy: 0.9548
Epoch 90/125
45/46 [============================>.] - ETA: 0s - loss: 0.0294 -␣
 ↪accuracy:
0.9910
Epoch 90: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0294 -
accuracy: 0.9910 - val_loss: 0.1662 - val_accuracy: 0.9484
Epoch 91/125
45/46 [============================>.] - ETA: 0s - loss: 0.0212 -␣
 ↪accuracy:
0.9937
Epoch 91: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0212 -
accuracy: 0.9938 - val_loss: 0.1603 - val_accuracy: 0.9484
Epoch 92/125
45/46 [============================>.] - ETA: 0s - loss: 0.0261 -␣
 ↪accuracy:
0.9931
Epoch 92: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0261 -
```

```
accuracy: 0.9931 - val_loss: 0.1561 - val_accuracy: 0.9452
Epoch 93/125
45/46 [============================>.] - ETA: 0s - loss: 0.0189 -␣
 ↪accuracy:
0.9931
Epoch 93: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0188 -
accuracy: 0.9931 - val_loss: 0.1635 - val_accuracy: 0.9452
Epoch 94/125
45/46 [============================>.] - ETA: 0s - loss: 0.0183 -␣
 ↪accuracy:
0.9924
Epoch 94: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0184 -
accuracy: 0.9924 - val_loss: 0.2323 - val_accuracy: 0.9226
Epoch 95/125
45/46 [============================>.] - ETA: 0s - loss: 0.0303 -␣
 ↪accuracy:
0.9889
Epoch 95: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0304 -
accuracy: 0.9889 - val_loss: 0.2100 - val_accuracy: 0.9290
Epoch 96/125
45/46 [============================>.] - ETA: 0s - loss: 0.0349 -␣
 ↪accuracy:
0.9875
Epoch 96: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 201ms/step - loss: 0.
 ↪0349 -
```

```
accuracy: 0.9875 - val_loss: 0.1626 - val_accuracy: 0.9484
Epoch 97/125
45/46 [============================>.] - ETA: 0s - loss: 0.0145 -␣
 ↪accuracy:
0.9972
Epoch 97: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0150 -
accuracy: 0.9965 - val_loss: 0.1621 - val_accuracy: 0.9484
Epoch 98/125
45/46 [============================>.] - ETA: 0s - loss: 0.0305 -␣
 ↪accuracy:
0.9826
Epoch 98: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0304 -
accuracy: 0.9827 - val_loss: 0.2919 - val_accuracy: 0.9065
Epoch 99/125
45/46 [============================>.] - ETA: 0s - loss: 0.0299 -␣
 ↪accuracy:
0.9882
Epoch 99: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0300 -
accuracy: 0.9882 - val_loss: 0.1432 - val_accuracy: 0.9484
Epoch 100/125
45/46 [============================>.] - ETA: 0s - loss: 0.0276 -␣
 ↪accuracy:
0.9896
Epoch 100: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0275 -
```

```
accuracy: 0.9896 - val_loss: 0.1561 - val_accuracy: 0.9419
Epoch 101/125
45/46 [============================>.] - ETA: 0s - loss: 0.0156 -␣
 ↪accuracy:
0.9931
Epoch 101: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0155 -
accuracy: 0.9931 - val_loss: 0.1514 - val_accuracy: 0.9419
Epoch 102/125
45/46 [============================>.] - ETA: 0s - loss: 0.0255 -␣
 ↪accuracy:
0.9924
Epoch 102: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0255 -
accuracy: 0.9924 - val_loss: 0.1371 - val_accuracy: 0.9452
Epoch 103/125
45/46 [============================>.] - ETA: 0s - loss: 0.0266 -␣
 ↪accuracy:
0.9931
Epoch 103: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0270 -
accuracy: 0.9924 - val_loss: 0.1401 - val_accuracy: 0.9548
Epoch 104/125
45/46 [============================>.] - ETA: 0s - loss: 0.0391 -␣
 ↪accuracy:
0.9889
Epoch 104: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0390 -
```

```
accuracy: 0.9889 - val_loss: 0.1362 - val_accuracy: 0.9548
Epoch 105/125
45/46 [============================>.] - ETA: 0s - loss: 0.0193 -␣
 ↪accuracy:
0.9937
Epoch 105: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0192 -
accuracy: 0.9938 - val_loss: 0.1376 - val_accuracy: 0.9516
Epoch 106/125
45/46 [============================>.] - ETA: 0s - loss: 0.0281 -␣
 ↪accuracy:
0.9903
Epoch 106: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0280 -
accuracy: 0.9903 - val_loss: 0.1795 - val_accuracy: 0.9355
Epoch 107/125
45/46 [============================>.] - ETA: 0s - loss: 0.0246 -␣
 ↪accuracy:
0.9937
Epoch 107: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0245 -
accuracy: 0.9938 - val_loss: 0.1641 - val_accuracy: 0.9484
Epoch 108/125
45/46 [============================>.] - ETA: 0s - loss: 0.0230 -␣
 ↪accuracy:
0.9917
Epoch 108: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0230 -
```

```
accuracy: 0.9917 - val_loss: 0.1528 - val_accuracy: 0.9484
Epoch 109/125
45/46 [============================>.] - ETA: 0s - loss: 0.0161 -␣
 ↪accuracy:
0.9944
Epoch 109: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0160 -
accuracy: 0.9945 - val_loss: 0.1730 - val_accuracy: 0.9452
Epoch 110/125
45/46 [============================>.] - ETA: 0s - loss: 0.0178 -␣
 ↪accuracy:
0.9931
Epoch 110: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0183 -
accuracy: 0.9924 - val_loss: 0.1696 - val_accuracy: 0.9452
Epoch 111/125
45/46 [============================>.] - ETA: 0s - loss: 0.0112 -␣
 ↪accuracy:
0.9972
Epoch 111: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0112 -
accuracy: 0.9972 - val_loss: 0.1849 - val_accuracy: 0.9355
Epoch 112/125
45/46 [============================>.] - ETA: 0s - loss: 0.0232 -␣
 ↪accuracy:
0.9917
Epoch 112: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0232 -
```

```
accuracy: 0.9917 - val_loss: 0.2456 - val_accuracy: 0.9161
Epoch 113/125
45/46 [============================>.] - ETA: 0s - loss: 0.0207 -␣
 ↪accuracy:
0.9924
Epoch 113: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0206 -
accuracy: 0.9924 - val_loss: 0.2248 - val_accuracy: 0.9355
Epoch 114/125
45/46 [============================>.] - ETA: 0s - loss: 0.0177 -␣
 ↪accuracy:
0.9924
Epoch 114: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 202ms/step - loss: 0.
 ↪0177 -
accuracy: 0.9924 - val_loss: 0.1759 - val_accuracy: 0.9419
Epoch 115/125
45/46 [============================>.] - ETA: 0s - loss: 0.0112 -␣
 ↪accuracy:
0.9972
Epoch 115: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0112 -
accuracy: 0.9972 - val_loss: 0.1450 - val_accuracy: 0.9516
Epoch 116/125
45/46 [============================>.] - ETA: 0s - loss: 0.0195 -␣
 ↪accuracy:
0.9944
Epoch 116: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0195 -
```

```
accuracy: 0.9945 - val_loss: 0.2949 - val_accuracy: 0.9161
Epoch 117/125
45/46 [============================>.] - ETA: 0s - loss: 0.0156 -␣
 ↪accuracy:
0.9944
Epoch 117: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0157 -
accuracy: 0.9945 - val_loss: 0.1502 - val_accuracy: 0.9484
Epoch 118/125
45/46 [============================>.] - ETA: 0s - loss: 0.0155 -␣
 ↪accuracy:
0.9965
Epoch 118: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0154 -
accuracy: 0.9965 - val_loss: 0.1967 - val_accuracy: 0.9323
Epoch 119/125
45/46 [============================>.] - ETA: 0s - loss: 0.0147 -␣
 ↪accuracy:
0.9951
Epoch 119: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0146 -
accuracy: 0.9952 - val_loss: 0.1854 - val_accuracy: 0.9355
Epoch 120/125
45/46 [============================>.] - ETA: 0s - loss: 0.0231 -␣
 ↪accuracy:
0.9903
Epoch 120: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0240 -
```

```
accuracy: 0.9896 - val_loss: 0.2079 - val_accuracy: 0.9387
Epoch 121/125
45/46 [============================>.] - ETA: 0s - loss: 0.0156 -␣
 ↪accuracy:
0.9944
Epoch 121: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0155 -
accuracy: 0.9945 - val_loss: 0.1518 - val_accuracy: 0.9516
Epoch 122/125
45/46 [============================>.] - ETA: 0s - loss: 0.0183 -␣
 ↪accuracy:
0.9924
Epoch 122: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0182 -
accuracy: 0.9924 - val_loss: 0.1749 - val_accuracy: 0.9484
Epoch 123/125
45/46 [============================>.] - ETA: 0s - loss: 0.0190 -␣
 ↪accuracy:
0.9937
Epoch 123: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0189 -
accuracy: 0.9938 - val_loss: 0.1865 - val_accuracy: 0.9452
Epoch 124/125
45/46 [============================>.] - ETA: 0s - loss: 0.0187 -␣
 ↪accuracy:
0.9937
Epoch 124: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0187 -
```

```
accuracy: 0.9938 - val_loss: 0.1707 - val_accuracy: 0.9516
Epoch 125/125
45/46 [============================>.] - ETA: 0s - loss: 0.0202 -␣
 ↪accuracy:
0.9944
Epoch 125: val_accuracy did not improve from 0.95806
46/46 [==============================] - 9s 203ms/step - loss: 0.
 ↪0202 -
accuracy: 0.9945 - val_loss: 0.2379 - val_accuracy: 0.9226
Elapsed time: 0:20:28.7
```

maximum validation accuracy is at 75th epoch

[ ]:
```
#rm -rf /content/gdrive/MyDrive/MA981-7-FY/Dataset/VGGmodels
```

[ ]:
```
history = VGG16.history.history
plot_metrics(history)
```

max size=0.90.9output$_7 6_0.png$

max size=0.90.9output$_7 6_1.png$

[ ]:
```
bestVGG16_model = load_model('/content/gdrive/MyDrive/MA981-7-FY/
 ↪Dataset/VGGmodels/075-0.99')
```

```
WARNING:tensorflow:SavedModel saved prior to TF 2.5 detected when␣
 ↪loading Keras
model. Please ensure that you are saving the model with model.
 ↪save() or
tf.keras.models.save_model(), *NOT* tf.saved_model.save(). To␣
 ↪confirm, there
should be a file named "keras_metadata.pb" in the SavedModel␣
 ↪directory.
```

Calculating accuracy of best VGG16 model on test set

[ ]:
```
loss, acc = bestVGG16_model.evaluate(x=X_test, y=Y_test)
```

```
10/10 [==============================] - 2s 143ms/step - loss: 0.
↪0205 -
accuracy: 0.9903
```

[ ]:
```
print (f"Test Loss = {loss}")
print (f"Test Accuracy = {acc}")
```

```
Test Loss = 0.020504727959632874
Test Accuracy = 0.9903225898742676
```

[ ]:
```
y_test_prob_v = bestVGG16_model.predict(X_test)
```

```
10/10 [==============================] - 1s 142ms/step
```

Calculating F1 score on test data

[ ]:
```
f1score_test_VGG = compute_f1_score(Y_test, y_test_prob_v)
print(f"F1 score: {f1score_test_VGG}")
```

```
F1 score: 0.9912023460410556
```

Result Time

[ ]:
```
data_percentage(Y) # whole data percentages
```

```
Number of examples: 2065
Percentage of positive examples: 52.54237288135593%, number of pos␣
↪examples:
1085
Percentage of negative examples: 47.45762711864407%, number of neg␣
↪examples: 980
```

[ ]:

```
print("Training Data:")
data_percentage(Y_train)
print("Validation Data:")
data_percentage(Y_val)
print("Testing Data:")
data_percentage(Y_test)
```

```
Training Data:
Number of examples: 1445
Percentage of positive examples: 51.4878892733564%, number of pos␣
 ↪examples: 744
Percentage of negative examples: 48.5121107266436%, number of neg␣
 ↪examples: 701
Validation Data:
Number of examples: 310
Percentage of positive examples: 54.83870967741935%, number of pos␣
 ↪examples: 170
Percentage of negative examples: 45.16129032258065%, number of neg␣
 ↪examples: 140
Testing Data:
Number of examples: 310
Percentage of positive examples: 55.16129032258065%, number of pos␣
 ↪examples: 171
Percentage of negative examples: 44.83870967741935%, number of neg␣
 ↪examples: 139
```

InceptionV3

```
[ ]:
INCV3=tf.keras.applications.InceptionV3(include_top=False,␣
 ↪weights='imagenet',  input_shape=(240,240,3),pooling=max)
```

```
[ ]:
incv3=tf.keras.Sequential()
incv3.add(INCV3)
incv3.add(Dropout(0.3))
```

```
incv3.add(Flatten())
incv3.add(Dropout(0.5))
incv3.add(Dense(1, activation='sigmoid'))
incv3.layers[0].trainable = False
incv3.compile(optimizer='adam', loss='binary_crossentropy',␣
↪metrics=['accuracy'])
```

[ ]:
```
incv3.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 inception_v3 (Functional)    (None, 6, 6, 2048)        21802784

 dropout_2 (Dropout)          (None, 6, 6, 2048)        0

 flatten_1 (Flatten)          (None, 73728)             0

 dropout_3 (Dropout)          (None, 73728)             0

 dense_1 (Dense)              (None, 1)                 73729

=================================================================
Total params: 21,876,513
Trainable params: 73,729
Non-trainable params: 21,802,784
_________________________________________________________________
```

[ ]:
```
checkpoint_path2 = "/content/gdrive/MyDrive/MA981-7-FY/Dataset/
↪inceptionv3/{epoch:03d}-{val_accuracy:.2f}"#"models/{epoch:
↪03d}-{val_loss:.2f}.h5"#"training_1/cp.ckpt"
```

```
checkpoint2 =␣
 ↪ModelCheckpoint(filepath=checkpoint_path2,monitor="val_accuracy",␣
 ↪verbose=1, save_best_only=True, mode='auto',save_freq="epoch")
```

[ ]:

```
start_time = time.time()

incv3.fit(x=X_train, y=Y_train, batch_size=32, epochs=100,␣
 ↪validation_data=(X_val, Y_val),callbacks=[checkpoint2])

end_time = time.time()
execution_time = (end_time - start_time)
print(f"Elapsed time: {hms_string(execution_time)}")
```

```
Epoch 1/100
46/46 [==============================] - ETA: 0s - loss: 0.9981 -␣
 ↪accuracy:
0.8083
Epoch 1: val_accuracy improved from -inf to 0.89355, saving model␣
 ↪to
/content/gdrive/MyDrive/MA981-7-FY/Dataset/inceptionv3/001-0.89

WARNING:absl:Found untraced functions such as␣
 ↪_jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while␣
 ↪saving (showing
5 of 94). These functions will not be directly callable after␣
 ↪loading.

46/46 [==============================] - 51s 987ms/step - loss: 0.
 ↪9981 -
accuracy: 0.8083 - val_loss: 0.3188 - val_accuracy: 0.8935
Epoch 2/100
```

```
45/46 [============================>.] - ETA: 0s - loss: 0.2842 -␣
 ↪accuracy:
0.9236
Epoch 2: val_accuracy did not improve from 0.89355
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪2832 -
accuracy: 0.9239 - val_loss: 0.4710 - val_accuracy: 0.8806
Epoch 3/100
45/46 [============================>.] - ETA: 0s - loss: 0.1308 -␣
 ↪accuracy:
0.9563
Epoch 3: val_accuracy improved from 0.89355 to 0.94839, saving␣
 ↪model to
/content/gdrive/MyDrive/MA981-7-FY/Dataset/inceptionv3/003-0.95

WARNING:absl:Found untraced functions such as␣
 ↪_jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while␣
 ↪saving (showing
5 of 94). These functions will not be directly callable after␣
 ↪loading.

46/46 [==============================] - 43s 944ms/step - loss: 0.
 ↪1306 -
accuracy: 0.9564 - val_loss: 0.1294 - val_accuracy: 0.9484
Epoch 4/100
45/46 [============================>.] - ETA: 0s - loss: 0.0670 -␣
 ↪accuracy:
0.9771
Epoch 4: val_accuracy improved from 0.94839 to 0.95484, saving␣
 ↪model to
/content/gdrive/MyDrive/MA981-7-FY/Dataset/inceptionv3/004-0.95
```

```
WARNING:absl:Found untraced functions such as␣
 ↪_jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while␣
 ↪saving (showing
5 of 94). These functions will not be directly callable after␣
 ↪loading.

46/46 [==============================] - 43s 954ms/step - loss: 0.
 ↪0668 -
accuracy: 0.9772 - val_loss: 0.1498 - val_accuracy: 0.9548
Epoch 5/100
45/46 [============================>.] - ETA: 0s - loss: 0.0715 -␣
 ↪accuracy:
0.9764
Epoch 5: val_accuracy did not improve from 0.95484
46/46 [==============================] - 5s 109ms/step - loss: 0.
 ↪0722 -
accuracy: 0.9758 - val_loss: 0.1598 - val_accuracy: 0.9516
Epoch 6/100
45/46 [============================>.] - ETA: 0s - loss: 0.0632 -␣
 ↪accuracy:
0.9778
Epoch 6: val_accuracy did not improve from 0.95484
46/46 [==============================] - 5s 110ms/step - loss: 0.
 ↪0630 -
accuracy: 0.9779 - val_loss: 0.1604 - val_accuracy: 0.9452
Epoch 7/100
45/46 [============================>.] - ETA: 0s - loss: 0.0289 -␣
 ↪accuracy:
0.9910
Epoch 7: val_accuracy did not improve from 0.95484
```

```
46/46 [==============================] - 5s 110ms/step - loss: 0.
 ↪0288 -
accuracy: 0.9910 - val_loss: 0.1438 - val_accuracy: 0.9516
Epoch 8/100
45/46 [============================>.] - ETA: 0s - loss: 0.0145 -␣
 ↪accuracy:
0.9924
Epoch 8: val_accuracy improved from 0.95484 to 0.96452, saving␣
 ↪model to
/content/gdrive/MyDrive/MA981-7-FY/Dataset/inceptionv3/008-0.96

WARNING:absl:Found untraced functions such as␣
 ↪_jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while␣
 ↪saving (showing
5 of 94). These functions will not be directly callable after␣
 ↪loading.

46/46 [==============================] - 42s 922ms/step - loss: 0.
 ↪0145 -
accuracy: 0.9924 - val_loss: 0.1347 - val_accuracy: 0.9645
Epoch 9/100
45/46 [============================>.] - ETA: 0s - loss: 0.0328 -␣
 ↪accuracy:
0.9875
Epoch 9: val_accuracy improved from 0.96452 to 0.97097, saving␣
 ↪model to
/content/gdrive/MyDrive/MA981-7-FY/Dataset/inceptionv3/009-0.97

WARNING:absl:Found untraced functions such as␣
 ↪_jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
```

```
_jit_compiled_convolution_op, _jit_compiled_convolution_op while␣
 ↪saving (showing
5 of 94). These functions will not be directly callable after␣
 ↪loading.

46/46 [==============================] - 43s 963ms/step - loss: 0.
 ↪0327 -
accuracy: 0.9875 - val_loss: 0.1210 - val_accuracy: 0.9710
Epoch 10/100
45/46 [============================>.] - ETA: 0s - loss: 0.0291 -␣
 ↪accuracy:
0.9882
Epoch 10: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0290 -
accuracy: 0.9882 - val_loss: 0.1556 - val_accuracy: 0.9581
Epoch 11/100
45/46 [============================>.] - ETA: 0s - loss: 0.0272 -␣
 ↪accuracy:
0.9931
Epoch 11: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0271 -
accuracy: 0.9931 - val_loss: 0.1126 - val_accuracy: 0.9613
Epoch 12/100
45/46 [============================>.] - ETA: 0s - loss: 0.0094 -␣
 ↪accuracy:
0.9972
Epoch 12: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 109ms/step - loss: 0.
 ↪0093 -
accuracy: 0.9972 - val_loss: 0.1012 - val_accuracy: 0.9645
```

```
Epoch 13/100
45/46 [============================>.] - ETA: 0s - loss: 0.0095 -␣
 ↪accuracy:
0.9951
Epoch 13: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 109ms/step - loss: 0.
 ↪0094 -
accuracy: 0.9952 - val_loss: 0.1472 - val_accuracy: 0.9581
Epoch 14/100
45/46 [============================>.] - ETA: 0s - loss: 0.0143 -␣
 ↪accuracy:
0.9958
Epoch 14: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 110ms/step - loss: 0.
 ↪0142 -
accuracy: 0.9958 - val_loss: 0.1854 - val_accuracy: 0.9516
Epoch 15/100
45/46 [============================>.] - ETA: 0s - loss: 0.0161 -␣
 ↪accuracy:
0.9951
Epoch 15: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 110ms/step - loss: 0.
 ↪0161 -
accuracy: 0.9952 - val_loss: 0.0758 - val_accuracy: 0.9677
Epoch 16/100
45/46 [============================>.] - ETA: 0s - loss: 0.0044 -␣
 ↪accuracy:
0.9993
Epoch 16: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 109ms/step - loss: 0.
 ↪0044 -
```

```
accuracy: 0.9993 - val_loss: 0.1170 - val_accuracy: 0.9677
Epoch 17/100
45/46 [============================>.] - ETA: 0s - loss: 0.0217 -␣
 ↪accuracy:
0.9931
Epoch 17: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 109ms/step - loss: 0.
 ↪0216 -
accuracy: 0.9931 - val_loss: 0.2104 - val_accuracy: 0.9548
Epoch 18/100
45/46 [============================>.] - ETA: 0s - loss: 0.0350 -␣
 ↪accuracy:
0.9896
Epoch 18: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 109ms/step - loss: 0.
 ↪0349 -
accuracy: 0.9896 - val_loss: 0.3068 - val_accuracy: 0.9226
Epoch 19/100
45/46 [============================>.] - ETA: 0s - loss: 0.0670 -␣
 ↪accuracy:
0.9778
Epoch 19: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0668 -
accuracy: 0.9779 - val_loss: 0.2086 - val_accuracy: 0.9484
Epoch 20/100
45/46 [============================>.] - ETA: 0s - loss: 0.0279 -␣
 ↪accuracy:
0.9903
Epoch 20: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 107ms/step - loss: 0.
 ↪0278 -
```

```
accuracy: 0.9903 - val_loss: 0.1989 - val_accuracy: 0.9548
Epoch 21/100
45/46 [============================>.] - ETA: 0s - loss: 0.0314 -␣
 ↪accuracy:
0.9882
Epoch 21: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 107ms/step - loss: 0.
 ↪0319 -
accuracy: 0.9875 - val_loss: 0.0928 - val_accuracy: 0.9645
Epoch 22/100
45/46 [============================>.] - ETA: 0s - loss: 0.0557 -␣
 ↪accuracy:
0.9868
Epoch 22: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 107ms/step - loss: 0.
 ↪0555 -
accuracy: 0.9869 - val_loss: 0.3801 - val_accuracy: 0.9258
Epoch 23/100
45/46 [============================>.] - ETA: 0s - loss: 0.0667 -␣
 ↪accuracy:
0.9854
Epoch 23: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 107ms/step - loss: 0.
 ↪0665 -
accuracy: 0.9855 - val_loss: 0.3897 - val_accuracy: 0.9387
Epoch 24/100
45/46 [============================>.] - ETA: 0s - loss: 0.0190 -␣
 ↪accuracy:
0.9937
Epoch 24: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 107ms/step - loss: 0.
 ↪0189 -
```

```
accuracy: 0.9938 - val_loss: 0.1840 - val_accuracy: 0.9581
Epoch 25/100
45/46 [============================>.] - ETA: 0s - loss: 0.0352 -␣
 ↪accuracy:
0.9917
Epoch 25: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0350 -
accuracy: 0.9917 - val_loss: 0.1487 - val_accuracy: 0.9645
Epoch 26/100
45/46 [============================>.] - ETA: 0s - loss: 0.0254 -␣
 ↪accuracy:
0.9903
Epoch 26: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 107ms/step - loss: 0.
 ↪0253 -
accuracy: 0.9903 - val_loss: 0.1503 - val_accuracy: 0.9645
Epoch 27/100
45/46 [============================>.] - ETA: 0s - loss: 0.0316 -␣
 ↪accuracy:
0.9951
Epoch 27: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 106ms/step - loss: 0.
 ↪0315 -
accuracy: 0.9952 - val_loss: 0.1609 - val_accuracy: 0.9645
Epoch 28/100
45/46 [============================>.] - ETA: 0s - loss: 0.0288 -␣
 ↪accuracy:
0.9924
Epoch 28: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 106ms/step - loss: 0.
 ↪0287 -
```

```
accuracy: 0.9924 - val_loss: 0.2122 - val_accuracy: 0.9645
Epoch 29/100
45/46 [============================>.] - ETA: 0s - loss: 0.0587 -␣
 ↪accuracy:
0.9854
Epoch 29: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 106ms/step - loss: 0.
 ↪0585 -
accuracy: 0.9855 - val_loss: 0.2142 - val_accuracy: 0.9677
Epoch 30/100
45/46 [============================>.] - ETA: 0s - loss: 0.0182 -␣
 ↪accuracy:
0.9931
Epoch 30: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0181 -
accuracy: 0.9931 - val_loss: 0.1858 - val_accuracy: 0.9677
Epoch 31/100
45/46 [============================>.] - ETA: 0s - loss: 0.0251 -␣
 ↪accuracy:
0.9937
Epoch 31: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 109ms/step - loss: 0.
 ↪0286 -
accuracy: 0.9931 - val_loss: 0.1960 - val_accuracy: 0.9677
Epoch 32/100
45/46 [============================>.] - ETA: 0s - loss: 0.0832 -␣
 ↪accuracy:
0.9806
Epoch 32: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0829 -
```

```
accuracy: 0.9806 - val_loss: 0.2134 - val_accuracy: 0.9548
Epoch 33/100
45/46 [============================>.] - ETA: 0s - loss: 0.0395 -␣
 ↪accuracy:
0.9910
Epoch 33: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0394 -
accuracy: 0.9910 - val_loss: 0.6002 - val_accuracy: 0.9226
Epoch 34/100
45/46 [============================>.] - ETA: 0s - loss: 0.0381 -␣
 ↪accuracy:
0.9903
Epoch 34: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0380 -
accuracy: 0.9903 - val_loss: 0.5104 - val_accuracy: 0.9290
Epoch 35/100
45/46 [============================>.] - ETA: 0s - loss: 0.0160 -␣
 ↪accuracy:
0.9958
Epoch 35: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0160 -
accuracy: 0.9958 - val_loss: 0.1634 - val_accuracy: 0.9677
Epoch 36/100
45/46 [============================>.] - ETA: 0s - loss: 0.0182 -␣
 ↪accuracy:
0.9937
Epoch 36: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0181 -
```

```
accuracy: 0.9938 - val_loss: 0.1912 - val_accuracy: 0.9581
Epoch 37/100
45/46 [============================>.] - ETA: 0s - loss: 0.0121 -␣
 ↪accuracy:
0.9965
Epoch 37: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0120 -
accuracy: 0.9965 - val_loss: 0.2660 - val_accuracy: 0.9581
Epoch 38/100
45/46 [============================>.] - ETA: 0s - loss: 0.0104 -␣
 ↪accuracy:
0.9965
Epoch 38: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0104 -
accuracy: 0.9965 - val_loss: 0.2295 - val_accuracy: 0.9677
Epoch 39/100
45/46 [============================>.] - ETA: 0s - loss: 0.0239 -␣
 ↪accuracy:
0.9944
Epoch 39: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 107ms/step - loss: 0.
 ↪0238 -
accuracy: 0.9945 - val_loss: 0.3704 - val_accuracy: 0.9516
Epoch 40/100
45/46 [============================>.] - ETA: 0s - loss: 0.0186 -␣
 ↪accuracy:
0.9937
Epoch 40: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 107ms/step - loss: 0.
 ↪0186 -
```

```
accuracy: 0.9938 - val_loss: 0.1911 - val_accuracy: 0.9677
Epoch 41/100
45/46 [============================>.] - ETA: 0s - loss: 0.0554 -␣
 ↪accuracy:
0.9896
Epoch 41: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0552 -
accuracy: 0.9896 - val_loss: 0.2311 - val_accuracy: 0.9645
Epoch 42/100
45/46 [============================>.] - ETA: 0s - loss: 0.0282 -␣
 ↪accuracy:
0.9931
Epoch 42: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 107ms/step - loss: 0.
 ↪0281 -
accuracy: 0.9931 - val_loss: 0.1895 - val_accuracy: 0.9645
Epoch 43/100
45/46 [============================>.] - ETA: 0s - loss: 0.0309 -␣
 ↪accuracy:
0.9937
Epoch 43: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 107ms/step - loss: 0.
 ↪0349 -
accuracy: 0.9931 - val_loss: 0.2243 - val_accuracy: 0.9645
Epoch 44/100
45/46 [============================>.] - ETA: 0s - loss: 0.0619 -␣
 ↪accuracy:
0.9896
Epoch 44: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0617 -
```

```
accuracy: 0.9896 - val_loss: 0.1502 - val_accuracy: 0.9645
Epoch 45/100
45/46 [============================>.] - ETA: 0s - loss: 0.0757 -␣
 ↪accuracy:
0.9833
Epoch 45: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0779 -
accuracy: 0.9827 - val_loss: 0.1678 - val_accuracy: 0.9710
Epoch 46/100
45/46 [============================>.] - ETA: 0s - loss: 0.2376 -␣
 ↪accuracy:
0.9701
Epoch 46: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪2368 -
accuracy: 0.9702 - val_loss: 0.4216 - val_accuracy: 0.9548
Epoch 47/100
45/46 [============================>.] - ETA: 0s - loss: 0.0602 -␣
 ↪accuracy:
0.9917
Epoch 47: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0600 -
accuracy: 0.9917 - val_loss: 0.2307 - val_accuracy: 0.9645
Epoch 48/100
45/46 [============================>.] - ETA: 0s - loss: 0.0304 -␣
 ↪accuracy:
0.9951
Epoch 48: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0303 -
```

```
accuracy: 0.9952 - val_loss: 0.1963 - val_accuracy: 0.9645
Epoch 49/100
45/46 [============================>.] - ETA: 0s - loss: 0.0315 -␣
 ↪accuracy:
0.9979
Epoch 49: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0314 -
accuracy: 0.9979 - val_loss: 0.2321 - val_accuracy: 0.9645
Epoch 50/100
45/46 [============================>.] - ETA: 0s - loss: 0.0071 -␣
 ↪accuracy:
0.9972
Epoch 50: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0071 -
accuracy: 0.9972 - val_loss: 0.2363 - val_accuracy: 0.9613
Epoch 51/100
45/46 [============================>.] - ETA: 0s - loss: 0.0201 -␣
 ↪accuracy:
0.9972
Epoch 51: val_accuracy did not improve from 0.97097
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0201 -
accuracy: 0.9972 - val_loss: 0.2395 - val_accuracy: 0.9613
Epoch 52/100
45/46 [============================>.] - ETA: 0s - loss: 0.0043 -␣
 ↪accuracy:
0.9986
Epoch 52: val_accuracy improved from 0.97097 to 0.97742, saving␣
 ↪model to
```

```
/content/gdrive/MyDrive/MA981-7-FY/Dataset/inceptionv3/052-0.98

WARNING:absl:Found untraced functions such as␣
 ↪_jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while␣
 ↪saving (showing
5 of 94). These functions will not be directly callable after␣
 ↪loading.

46/46 [==============================] - 41s 918ms/step - loss: 0.
 ↪0042 -
accuracy: 0.9986 - val_loss: 0.1118 - val_accuracy: 0.9774
Epoch 53/100
45/46 [============================>.] - ETA: 0s - loss: 0.0152 -␣
 ↪accuracy:
0.9993
Epoch 53: val_accuracy did not improve from 0.97742
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0152 -
accuracy: 0.9993 - val_loss: 0.1453 - val_accuracy: 0.9710
Epoch 54/100
45/46 [============================>.] - ETA: 0s - loss: 0.0111 -␣
 ↪accuracy:
0.9972
Epoch 54: val_accuracy did not improve from 0.97742
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0111 -
accuracy: 0.9972 - val_loss: 0.2853 - val_accuracy: 0.9613
Epoch 55/100
45/46 [============================>.] - ETA: 0s - loss: 0.0223 -␣
 ↪accuracy:
0.9972
```

```
Epoch 55: val_accuracy did not improve from 0.97742
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0223 -
accuracy: 0.9972 - val_loss: 0.1177 - val_accuracy: 0.9677
Epoch 56/100
45/46 [============================>.] - ETA: 0s - loss: 0.0256 -␣
 ↪accuracy:
0.9931
Epoch 56: val_accuracy did not improve from 0.97742
46/46 [==============================] - 5s 109ms/step - loss: 0.
 ↪0255 -
accuracy: 0.9931 - val_loss: 0.2175 - val_accuracy: 0.9677
Epoch 57/100
45/46 [============================>.] - ETA: 0s - loss: 0.0036 -␣
 ↪accuracy:
0.9986
Epoch 57: val_accuracy did not improve from 0.97742
46/46 [==============================] - 5s 110ms/step - loss: 0.
 ↪0036 -
accuracy: 0.9986 - val_loss: 0.2353 - val_accuracy: 0.9710
Epoch 58/100
45/46 [============================>.] - ETA: 0s - loss: 0.0192 -␣
 ↪accuracy:
0.9958
Epoch 58: val_accuracy did not improve from 0.97742
46/46 [==============================] - 5s 110ms/step - loss: 0.
 ↪0191 -
accuracy: 0.9958 - val_loss: 0.1579 - val_accuracy: 0.9774
Epoch 59/100
45/46 [============================>.] - ETA: 0s - loss: 0.0046 -␣
 ↪accuracy:
```

```
0.9972
Epoch 59: val_accuracy did not improve from 0.97742
46/46 [==============================] - 5s 110ms/step - loss: 0.
 ↪0046 -
accuracy: 0.9972 - val_loss: 0.1898 - val_accuracy: 0.9742
Epoch 60/100
45/46 [============================>.] - ETA: 0s - loss: 0.0193 -␣
 ↪accuracy:
0.9958
Epoch 60: val_accuracy improved from 0.97742 to 0.98065, saving␣
 ↪model to
/content/gdrive/MyDrive/MA981-7-FY/Dataset/inceptionv3/060-0.98

WARNING:absl:Found untraced functions such as␣
 ↪_jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while␣
 ↪saving (showing
5 of 94). These functions will not be directly callable after␣
 ↪loading.

46/46 [==============================] - 42s 932ms/step - loss: 0.
 ↪0193 -
accuracy: 0.9958 - val_loss: 0.1210 - val_accuracy: 0.9806
Epoch 61/100
45/46 [============================>.] - ETA: 0s - loss: 0.0047 -␣
 ↪accuracy:
0.9986
Epoch 61: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 107ms/step - loss: 0.
 ↪0047 -
accuracy: 0.9986 - val_loss: 0.1449 - val_accuracy: 0.9774
Epoch 62/100
```

```
45/46 [============================>.] - ETA: 0s - loss: 4.
 ↪3286e-04 - accuracy:
1.0000
Epoch 62: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 108ms/step - loss: 4.
 ↪3136e-04 -
accuracy: 1.0000 - val_loss: 0.2377 - val_accuracy: 0.9677
Epoch 63/100
45/46 [============================>.] - ETA: 0s - loss: 0.0098 -␣
 ↪accuracy:
0.9965
Epoch 63: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0098 -
accuracy: 0.9965 - val_loss: 0.1680 - val_accuracy: 0.9774
Epoch 64/100
45/46 [============================>.] - ETA: 0s - loss: 0.0051 -␣
 ↪accuracy:
0.9986
Epoch 64: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 109ms/step - loss: 0.
 ↪0050 -
accuracy: 0.9986 - val_loss: 0.1901 - val_accuracy: 0.9710
Epoch 65/100
45/46 [============================>.] - ETA: 0s - loss: 0.0177 -␣
 ↪accuracy:
0.9958
Epoch 65: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 109ms/step - loss: 0.
 ↪0176 -
accuracy: 0.9958 - val_loss: 0.1805 - val_accuracy: 0.9677
```

```
Epoch 66/100
45/46 [============================>.] - ETA: 0s - loss: 0.0147 -␣
 ↪accuracy:
0.9979
Epoch 66: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 109ms/step - loss: 0.
 ↪0146 -
accuracy: 0.9979 - val_loss: 0.4356 - val_accuracy: 0.9516
Epoch 67/100
45/46 [============================>.] - ETA: 0s - loss: 0.0133 -␣
 ↪accuracy:
0.9979
Epoch 67: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 110ms/step - loss: 0.
 ↪0133 -
accuracy: 0.9979 - val_loss: 0.0786 - val_accuracy: 0.9774
Epoch 68/100
45/46 [============================>.] - ETA: 0s - loss: 0.0069 -␣
 ↪accuracy:
0.9979
Epoch 68: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 109ms/step - loss: 0.
 ↪0068 -
accuracy: 0.9979 - val_loss: 0.1001 - val_accuracy: 0.9710
Epoch 69/100
45/46 [============================>.] - ETA: 0s - loss: 0.0273 -␣
 ↪accuracy:
0.9951
Epoch 69: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 109ms/step - loss: 0.
 ↪0272 -
```

```
accuracy: 0.9952 - val_loss: 0.3426 - val_accuracy: 0.9645
Epoch 70/100
45/46 [============================>.] - ETA: 0s - loss: 0.0074 -␣
 ↪accuracy:
0.9972
Epoch 70: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0074 -
accuracy: 0.9972 - val_loss: 0.3471 - val_accuracy: 0.9581
Epoch 71/100
45/46 [============================>.] - ETA: 0s - loss: 0.0411 -␣
 ↪accuracy:
0.9944
Epoch 71: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0409 -
accuracy: 0.9945 - val_loss: 0.2169 - val_accuracy: 0.9645
Epoch 72/100
45/46 [============================>.] - ETA: 0s - loss: 0.0213 -␣
 ↪accuracy:
0.9937
Epoch 72: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0212 -
accuracy: 0.9938 - val_loss: 0.3620 - val_accuracy: 0.9581
Epoch 73/100
45/46 [============================>.] - ETA: 0s - loss: 0.0639 -␣
 ↪accuracy:
0.9924
Epoch 73: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 107ms/step - loss: 0.
 ↪0637 -
```

```
accuracy: 0.9924 - val_loss: 0.2636 - val_accuracy: 0.9581
Epoch 74/100
45/46 [============================>.] - ETA: 0s - loss: 0.0148 -␣
 ↪accuracy:
0.9958
Epoch 74: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 107ms/step - loss: 0.
 ↪0147 -
accuracy: 0.9958 - val_loss: 0.2284 - val_accuracy: 0.9645
Epoch 75/100
45/46 [============================>.] - ETA: 0s - loss: 0.0296 -␣
 ↪accuracy:
0.9958
Epoch 75: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 107ms/step - loss: 0.
 ↪0295 -
accuracy: 0.9958 - val_loss: 0.2761 - val_accuracy: 0.9613
Epoch 76/100
45/46 [============================>.] - ETA: 0s - loss: 0.0036 -␣
 ↪accuracy:
0.9986
Epoch 76: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0036 -
accuracy: 0.9986 - val_loss: 0.2698 - val_accuracy: 0.9613
Epoch 77/100
45/46 [============================>.] - ETA: 0s - loss: 0.0155 -␣
 ↪accuracy:
0.9979
Epoch 77: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0155 -
```

```
accuracy: 0.9979 - val_loss: 0.4028 - val_accuracy: 0.9613
Epoch 78/100
45/46 [============================>.] - ETA: 0s - loss: 0.0073 -␣
 ↪accuracy:
0.9972
Epoch 78: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0073 -
accuracy: 0.9972 - val_loss: 0.2424 - val_accuracy: 0.9742
Epoch 79/100
45/46 [============================>.] - ETA: 0s - loss: 8.
 ↪6900e-04 - accuracy:
1.0000
Epoch 79: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 108ms/step - loss: 8.
 ↪6600e-04 -
accuracy: 1.0000 - val_loss: 0.2489 - val_accuracy: 0.9645
Epoch 80/100
45/46 [============================>.] - ETA: 0s - loss: 0.0186 -␣
 ↪accuracy:
0.9958
Epoch 80: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0186 -
accuracy: 0.9958 - val_loss: 0.3235 - val_accuracy: 0.9677
Epoch 81/100
45/46 [============================>.] - ETA: 0s - loss: 0.0138 -␣
 ↪accuracy:
0.9972
Epoch 81: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0138 -
```

```
accuracy: 0.9972 - val_loss: 0.3079 - val_accuracy: 0.9645
Epoch 82/100
45/46 [============================>.] - ETA: 0s - loss: 0.0223 -␣
 ↪accuracy:
0.9965
Epoch 82: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 109ms/step - loss: 0.
 ↪0222 -
accuracy: 0.9965 - val_loss: 1.0241 - val_accuracy: 0.9161
Epoch 83/100
45/46 [============================>.] - ETA: 0s - loss: 0.1464 -␣
 ↪accuracy:
0.9847
Epoch 83: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪1459 -
accuracy: 0.9848 - val_loss: 0.2224 - val_accuracy: 0.9774
Epoch 84/100
45/46 [============================>.] - ETA: 0s - loss: 0.0203 -␣
 ↪accuracy:
0.9972
Epoch 84: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0202 -
accuracy: 0.9972 - val_loss: 0.2478 - val_accuracy: 0.9710
Epoch 85/100
45/46 [============================>.] - ETA: 0s - loss: 0.0099 -␣
 ↪accuracy:
0.9965
Epoch 85: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0099 -
```

```
accuracy: 0.9965 - val_loss: 0.2607 - val_accuracy: 0.9613
Epoch 86/100
45/46 [============================>.] - ETA: 0s - loss: 0.0483 -␣
 ↪accuracy:
0.9924
Epoch 86: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 109ms/step - loss: 0.
 ↪0481 -
accuracy: 0.9924 - val_loss: 0.3353 - val_accuracy: 0.9645
Epoch 87/100
45/46 [============================>.] - ETA: 0s - loss: 0.0055 -␣
 ↪accuracy:
0.9993
Epoch 87: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 109ms/step - loss: 0.
 ↪0055 -
accuracy: 0.9993 - val_loss: 0.3105 - val_accuracy: 0.9677
Epoch 88/100
45/46 [============================>.] - ETA: 0s - loss: 0.0313 -␣
 ↪accuracy:
0.9965
Epoch 88: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 109ms/step - loss: 0.
 ↪0312 -
accuracy: 0.9965 - val_loss: 0.1593 - val_accuracy: 0.9742
Epoch 89/100
45/46 [============================>.] - ETA: 0s - loss: 0.0141 -␣
 ↪accuracy:
0.9972
Epoch 89: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0141 -
```

```
accuracy: 0.9972 - val_loss: 0.3330 - val_accuracy: 0.9710
Epoch 90/100
45/46 [============================>.] - ETA: 0s - loss: 0.0176 -␣
 ↪accuracy:
0.9986
Epoch 90: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0176 -
accuracy: 0.9986 - val_loss: 0.3704 - val_accuracy: 0.9677
Epoch 91/100
45/46 [============================>.] - ETA: 0s - loss: 0.0203 -␣
 ↪accuracy:
0.9958
Epoch 91: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0203 -
accuracy: 0.9958 - val_loss: 0.1441 - val_accuracy: 0.9806
Epoch 92/100
45/46 [============================>.] - ETA: 0s - loss: 0.0114 -␣
 ↪accuracy:
0.9986
Epoch 92: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0114 -
accuracy: 0.9986 - val_loss: 0.2770 - val_accuracy: 0.9742
Epoch 93/100
45/46 [============================>.] - ETA: 0s - loss: 0.0085 -␣
 ↪accuracy:
0.9986
Epoch 93: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0084 -
```

```
accuracy: 0.9986 - val_loss: 0.3379 - val_accuracy: 0.9677
Epoch 94/100
45/46 [============================>.] - ETA: 0s - loss: 0.0133 -␣
 ↪accuracy:
0.9972
Epoch 94: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0132 -
accuracy: 0.9972 - val_loss: 0.3376 - val_accuracy: 0.9677
Epoch 95/100
45/46 [============================>.] - ETA: 0s - loss: 0.0105 -␣
 ↪accuracy:
0.9979
Epoch 95: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 107ms/step - loss: 0.
 ↪0104 -
accuracy: 0.9979 - val_loss: 0.4553 - val_accuracy: 0.9613
Epoch 96/100
45/46 [============================>.] - ETA: 0s - loss: 0.0236 -␣
 ↪accuracy:
0.9951
Epoch 96: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0272 -
accuracy: 0.9945 - val_loss: 0.2219 - val_accuracy: 0.9774
Epoch 97/100
45/46 [============================>.] - ETA: 0s - loss: 0.1562 -␣
 ↪accuracy:
0.9819
Epoch 97: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪1557 -
```

```
accuracy: 0.9820 - val_loss: 0.0947 - val_accuracy: 0.9806
Epoch 98/100
45/46 [============================>.] - ETA: 0s - loss: 0.0382 -␣
 ↪accuracy:
0.9917
Epoch 98: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0380 -
accuracy: 0.9917 - val_loss: 0.1887 - val_accuracy: 0.9742
Epoch 99/100
45/46 [============================>.] - ETA: 0s - loss: 0.0222 -␣
 ↪accuracy:
0.9979
Epoch 99: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0221 -
accuracy: 0.9979 - val_loss: 0.1946 - val_accuracy: 0.9677
Epoch 100/100
45/46 [============================>.] - ETA: 0s - loss: 0.0029 -␣
 ↪accuracy:
0.9993
Epoch 100: val_accuracy did not improve from 0.98065
46/46 [==============================] - 5s 108ms/step - loss: 0.
 ↪0029 -
accuracy: 0.9993 - val_loss: 0.2565 - val_accuracy: 0.9710
Elapsed time: 0:12:48.9
```

[ ]:

```
history = incv3.history.history
plot_metrics(history)
```

max size=0.90.9output$_9 3_0$.*png*

max size=0.90.9output$_9 3_1$.png

Maximum validation accuracy at 47th epoch

[ ]:
```
bestincv3_model = load_model(filepath='/content/gdrive/MyDrive/
↪MA981-7-FY/Dataset/inceptionv3/047-0.98')

WARNING:tensorflow:SavedModel saved prior to TF 2.5 detected when␣
↪loading Keras
model. Please ensure that you are saving the model with model.
↪save() or
tf.keras.models.save_model(), *NOT* tf.saved_model.save(). To␣
↪confirm, there
should be a file named "keras_metadata.pb" in the SavedModel␣
↪directory.
```

Calculating accuracy of best inception V3 model on test set

[ ]:
```
loss, acc = bestincv3_model.evaluate(x=X_test, y=Y_test)

10/10 [==============================] - 3s 81ms/step - loss: 0.
↪1115 - accuracy:
0.9839
```

[ ]:
```
print (f"Test Loss = {loss}")
print (f"Test Accuracy = {acc}")

Test Loss = 0.11152724176645279
Test Accuracy = 0.9838709831237793
```

[ ]:
```
Y_test_prob_i = bestincv3_model.predict(X_test)

10/10 [==============================] - 2s 85ms/step
```

Calculating F1 score on test data

[ ]:

```
f1score_val = compute_f1_score(Y_test, Y_test_prob_i)
print(f"F1 score: {f1score_val}")
```

```
F1 score: 0.9853372434017595
```

Result time

[ ]:
```
print("Training Data:")
data_percentage(Y_train)
print("Validation Data:")
data_percentage(Y_val)
print("Testing Data:")
data_percentage(Y_test)
```

```
Training Data:
Number of examples: 1445
Percentage of positive examples: 51.4878892733564%, number of pos␣
 ↪examples: 744
Percentage of negative examples: 48.5121107266436%, number of neg␣
 ↪examples: 701
Validation Data:
Number of examples: 310
Percentage of positive examples: 54.83870967741935%, number of pos␣
 ↪examples: 170
Percentage of negative examples: 45.16129032258065%, number of neg␣
 ↪examples: 140
Testing Data:
Number of examples: 310
Percentage of positive examples: 55.16129032258065%, number of pos␣
 ↪examples: 171
Percentage of negative examples: 44.83870967741935%, number of neg␣
 ↪examples: 139
```

Resnet50

[ ]:

```
resnet50=tf.keras.applications.ResNet50(include_top=False,␣
 ↪weights='imagenet',input_shape=(240,240,3),pooling=max)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/
 ↪keras-
applications/resnet/
 ↪resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736 [==============================] - 5s 0us/step
```

[ ]:
```
RESNET50=tf.keras.Sequential()
RESNET50.add(resnet50)
RESNET50.add(Dropout(0.3))
RESNET50.add(Flatten())
RESNET50.add(Dropout(0.5))
RESNET50.add(Dense(1, activation='sigmoid'))
RESNET50.layers[0].trainable = False
RESNET50.compile(optimizer='adam', loss='binary_crossentropy',␣
 ↪metrics=['accuracy'])
```

[ ]:
```
RESNET50.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 resnet50 (Functional)        (None, 8, 8, 2048)        23587712

 dropout_4 (Dropout)          (None, 8, 8, 2048)        0

 flatten_2 (Flatten)          (None, 131072)            0

 dropout_5 (Dropout)          (None, 131072)            0
```

```
 dense_2 (Dense)              (None, 1)                 131073

=================================================================
Total params: 23,718,785
Trainable params: 131,073
Non-trainable params: 23,587,712
_________________________________________________________________
```

[ ]:
```
checkpoint_path3 = "/content/gdrive/MyDrive/MA981-7-FY/Dataset/
↪RESNET50/{epoch:03d}-{val_accuracy:.2f}"#"models/{epoch:
↪03d}-{val_loss:.2f}.h5"#"training_1/cp.ckpt"
checkpoint2 =␣
↪ModelCheckpoint(filepath=checkpoint_path3,monitor="val_accuracy",␣
↪verbose=1, save_best_only=True, mode='auto',save_freq="epoch")
```

[ ]:
```
start_time = time.time()

RESNET50.fit(x=X_train, y=Y_train, batch_size=32, epochs=100,␣
↪validation_data=(X_val, Y_val),callbacks=[checkpoint2])

end_time = time.time()
execution_time = (end_time - start_time)
print(f"Elapsed time: {hms_string(execution_time)}")

Epoch 1/100
46/46 [==============================] - ETA: 0s - loss: 1.6523 -␣
↪accuracy:
0.6166
Epoch 1: val_accuracy improved from -inf to 0.68065, saving model␣
↪to
/content/gdrive/MyDrive/MA981-7-FY/Dataset/RESNET50/001-0.68
```

```
WARNING:absl:Found untraced functions such as␣
 ↪_jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while␣
 ↪saving (showing
5 of 53). These functions will not be directly callable after␣
 ↪loading.

46/46 [==============================] - 38s 737ms/step - loss: 1.
 ↪6523 -
accuracy: 0.6166 - val_loss: 0.6547 - val_accuracy: 0.6806
Epoch 2/100
45/46 [============================>.] - ETA: 0s - loss: 0.8151 -␣
 ↪accuracy:
0.6861
Epoch 2: val_accuracy improved from 0.68065 to 0.79032, saving␣
 ↪model to
/content/gdrive/MyDrive/MA981-7-FY/Dataset/RESNET50/002-0.79

WARNING:absl:Found untraced functions such as␣
 ↪_jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while␣
 ↪saving (showing
5 of 53). These functions will not be directly callable after␣
 ↪loading.

46/46 [==============================] - 32s 706ms/step - loss: 0.
 ↪8175 -
accuracy: 0.6858 - val_loss: 0.4951 - val_accuracy: 0.7903
Epoch 3/100
45/46 [============================>.] - ETA: 0s - loss: 0.8799 -␣
 ↪accuracy:
```

```
0.6806
Epoch 3: val_accuracy did not improve from 0.79032
46/46 [==============================] - 6s 138ms/step - loss: 0.
 ↪8789 -
accuracy: 0.6810 - val_loss: 0.6087 - val_accuracy: 0.7516
Epoch 4/100
45/46 [============================>.] - ETA: 0s - loss: 0.7528 -␣
 ↪accuracy:
0.7146
Epoch 4: val_accuracy improved from 0.79032 to 0.80000, saving␣
 ↪model to
/content/gdrive/MyDrive/MA981-7-FY/Dataset/RESNET50/004-0.80

WARNING:absl:Found untraced functions such as␣
 ↪_jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while␣
 ↪saving (showing
5 of 53). These functions will not be directly callable after␣
 ↪loading.

46/46 [==============================] - 32s 710ms/step - loss: 0.
 ↪7529 -
accuracy: 0.7142 - val_loss: 0.4660 - val_accuracy: 0.8000
Epoch 5/100
45/46 [============================>.] - ETA: 0s - loss: 0.8146 -␣
 ↪accuracy:
0.7215
Epoch 5: val_accuracy did not improve from 0.80000
46/46 [==============================] - 6s 136ms/step - loss: 0.
 ↪8151 -
accuracy: 0.7218 - val_loss: 0.5880 - val_accuracy: 0.7548
Epoch 6/100
```

```
45/46 [============================>.] - ETA: 0s - loss: 0.7051 -␣
 ↪accuracy:
0.7535
Epoch 6: val_accuracy did not improve from 0.80000
46/46 [==============================] - 6s 137ms/step - loss: 0.
 ↪7055 -
accuracy: 0.7536 - val_loss: 0.7671 - val_accuracy: 0.7032
Epoch 7/100
45/46 [============================>.] - ETA: 0s - loss: 0.6996 -␣
 ↪accuracy:
0.7465
Epoch 7: val_accuracy improved from 0.80000 to 0.80968, saving␣
 ↪model to
/content/gdrive/MyDrive/MA981-7-FY/Dataset/RESNET50/007-0.81

WARNING:absl:Found untraced functions such as␣
 ↪_jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while␣
 ↪saving (showing
5 of 53). These functions will not be directly callable after␣
 ↪loading.

46/46 [==============================] - 33s 722ms/step - loss: 0.
 ↪7008 -
accuracy: 0.7460 - val_loss: 0.4618 - val_accuracy: 0.8097
Epoch 8/100
45/46 [============================>.] - ETA: 0s - loss: 0.6574 -␣
 ↪accuracy:
0.7576
Epoch 8: val_accuracy did not improve from 0.80968
46/46 [==============================] - 6s 137ms/step - loss: 0.
 ↪6553 -
```

```
accuracy: 0.7585 - val_loss: 0.7377 - val_accuracy: 0.7129
Epoch 9/100
45/46 [============================>.] - ETA: 0s - loss: 0.7263 -␣
 ↪accuracy:
0.7556
Epoch 9: val_accuracy did not improve from 0.80968
46/46 [==============================] - 6s 137ms/step - loss: 0.
 ↪7242 -
accuracy: 0.7564 - val_loss: 1.0751 - val_accuracy: 0.6548
Epoch 10/100
45/46 [============================>.] - ETA: 0s - loss: 0.6673 -␣
 ↪accuracy:
0.7785
Epoch 10: val_accuracy improved from 0.80968 to 0.82258, saving␣
 ↪model to
/content/gdrive/MyDrive/MA981-7-FY/Dataset/RESNET50/010-0.82

WARNING:absl:Found untraced functions such as␣
 ↪_jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while␣
 ↪saving (showing
5 of 53). These functions will not be directly callable after␣
 ↪loading.

46/46 [==============================] - 30s 670ms/step - loss: 0.
 ↪6669 -
accuracy: 0.7785 - val_loss: 0.4658 - val_accuracy: 0.8226
Epoch 11/100
45/46 [============================>.] - ETA: 0s - loss: 0.7755 -␣
 ↪accuracy:
0.7486
Epoch 11: val_accuracy did not improve from 0.82258
```

```
46/46 [==============================] - 6s 136ms/step - loss: 0.
 ↪7728 -
accuracy: 0.7495 - val_loss: 0.5717 - val_accuracy: 0.7581
Epoch 12/100
45/46 [============================>.] - ETA: 0s - loss: 0.8838 -␣
 ↪accuracy:
0.7493
Epoch 12: val_accuracy did not improve from 0.82258
46/46 [==============================] - 6s 137ms/step - loss: 0.
 ↪8808 -
accuracy: 0.7502 - val_loss: 0.5370 - val_accuracy: 0.8129
Epoch 13/100
45/46 [============================>.] - ETA: 0s - loss: 0.8993 -␣
 ↪accuracy:
0.7479
Epoch 13: val_accuracy did not improve from 0.82258
46/46 [==============================] - 6s 138ms/step - loss: 0.
 ↪9003 -
accuracy: 0.7481 - val_loss: 0.8419 - val_accuracy: 0.7387
Epoch 14/100
45/46 [============================>.] - ETA: 0s - loss: 0.7013 -␣
 ↪accuracy:
0.7812
Epoch 14: val_accuracy did not improve from 0.82258
46/46 [==============================] - 6s 139ms/step - loss: 0.
 ↪7025 -
accuracy: 0.7806 - val_loss: 0.6247 - val_accuracy: 0.7774
Epoch 15/100
45/46 [============================>.] - ETA: 0s - loss: 0.6716 -␣
 ↪accuracy:
0.8014
```

```
Epoch 15: val_accuracy did not improve from 0.82258
46/46 [==============================] - 6s 139ms/step - loss: 0.
 ↪6732 -
accuracy: 0.8007 - val_loss: 0.6520 - val_accuracy: 0.8065
Epoch 16/100
45/46 [============================>.] - ETA: 0s - loss: 0.7642 -␣
 ↪accuracy:
0.7924
Epoch 16: val_accuracy did not improve from 0.82258
46/46 [==============================] - 6s 138ms/step - loss: 0.
 ↪7680 -
accuracy: 0.7910 - val_loss: 0.9004 - val_accuracy: 0.7452
Epoch 17/100
45/46 [============================>.] - ETA: 0s - loss: 0.7503 -␣
 ↪accuracy:
0.7882
Epoch 17: val_accuracy did not improve from 0.82258
46/46 [==============================] - 6s 137ms/step - loss: 0.
 ↪7511 -
accuracy: 0.7875 - val_loss: 0.5061 - val_accuracy: 0.8065
Epoch 18/100
45/46 [============================>.] - ETA: 0s - loss: 0.6352 -␣
 ↪accuracy:
0.8146
Epoch 18: val_accuracy did not improve from 0.82258
46/46 [==============================] - 6s 137ms/step - loss: 0.
 ↪6373 -
accuracy: 0.8145 - val_loss: 0.6057 - val_accuracy: 0.8000
Epoch 19/100
45/46 [============================>.] - ETA: 0s - loss: 0.7422 -␣
 ↪accuracy:
```

```
0.7826
Epoch 19: val_accuracy did not improve from 0.82258
46/46 [==============================] - 6s 137ms/step - loss: 0.
↪7423 -
accuracy: 0.7820 - val_loss: 0.8349 - val_accuracy: 0.7581
Epoch 20/100
45/46 [============================>.] - ETA: 0s - loss: 0.7930 -␣
↪accuracy:
0.7896
Epoch 20: val_accuracy did not improve from 0.82258
46/46 [==============================] - 6s 136ms/step - loss: 0.
↪7908 -
accuracy: 0.7903 - val_loss: 0.6290 - val_accuracy: 0.7968
Epoch 21/100
45/46 [============================>.] - ETA: 0s - loss: 0.8247 -␣
↪accuracy:
0.7750
Epoch 21: val_accuracy did not improve from 0.82258
46/46 [==============================] - 6s 136ms/step - loss: 0.
↪8280 -
accuracy: 0.7744 - val_loss: 0.8257 - val_accuracy: 0.7677
Epoch 22/100
45/46 [============================>.] - ETA: 0s - loss: 0.7552 -␣
↪accuracy:
0.7965
Epoch 22: val_accuracy did not improve from 0.82258
46/46 [==============================] - 6s 136ms/step - loss: 0.
↪7563 -
accuracy: 0.7965 - val_loss: 0.7357 - val_accuracy: 0.7774
Epoch 23/100
45/46 [============================>.] - ETA: 0s - loss: 0.7337 -␣
↪accuracy:
```

```
0.7924
Epoch 23: val_accuracy did not improve from 0.82258
46/46 [==============================] - 6s 136ms/step - loss: 0.
 ↪7312 -
accuracy: 0.7931 - val_loss: 0.6174 - val_accuracy: 0.8065
Epoch 24/100
45/46 [============================>.] - ETA: 0s - loss: 0.6002 -␣
 ↪accuracy:
0.8264
Epoch 24: val_accuracy did not improve from 0.82258
46/46 [==============================] - 6s 136ms/step - loss: 0.
 ↪6009 -
accuracy: 0.8263 - val_loss: 0.8221 - val_accuracy: 0.7452
Epoch 25/100
45/46 [============================>.] - ETA: 0s - loss: 0.6117 -␣
 ↪accuracy:
0.8111
Epoch 25: val_accuracy did not improve from 0.82258
46/46 [==============================] - 6s 136ms/step - loss: 0.
 ↪6097 -
accuracy: 0.8118 - val_loss: 0.8115 - val_accuracy: 0.7581
Epoch 26/100
45/46 [============================>.] - ETA: 0s - loss: 0.6246 -␣
 ↪accuracy:
0.8104
Epoch 26: val_accuracy improved from 0.82258 to 0.84194, saving␣
 ↪model to
/content/gdrive/MyDrive/MA981-7-FY/Dataset/RESNET50/026-0.84

WARNING:absl:Found untraced functions such as␣
 ↪_jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
```

```
_jit_compiled_convolution_op, _jit_compiled_convolution_op while␣
 ↪saving (showing
5 of 53). These functions will not be directly callable after␣
 ↪loading.

46/46 [==============================] - 31s 686ms/step - loss: 0.
 ↪6225 -
accuracy: 0.8111 - val_loss: 0.5904 - val_accuracy: 0.8419
Epoch 27/100
45/46 [============================>.] - ETA: 0s - loss: 0.6291 -␣
 ↪accuracy:
0.8201
Epoch 27: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 136ms/step - loss: 0.
 ↪6279 -
accuracy: 0.8201 - val_loss: 0.9067 - val_accuracy: 0.7548
Epoch 28/100
45/46 [============================>.] - ETA: 0s - loss: 0.6182 -␣
 ↪accuracy:
0.8285
Epoch 28: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 137ms/step - loss: 0.
 ↪6215 -
accuracy: 0.8284 - val_loss: 0.8154 - val_accuracy: 0.7516
Epoch 29/100
45/46 [============================>.] - ETA: 0s - loss: 0.6680 -␣
 ↪accuracy:
0.8160
Epoch 29: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 137ms/step - loss: 0.
 ↪6699 -
accuracy: 0.8159 - val_loss: 0.8605 - val_accuracy: 0.7516
```

```
Epoch 30/100
45/46 [============================>.] - ETA: 0s - loss: 0.6418 -␣
 ↪accuracy:
0.8236
Epoch 30: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 138ms/step - loss: 0.
 ↪6396 -
accuracy: 0.8242 - val_loss: 0.7185 - val_accuracy: 0.8032
Epoch 31/100
45/46 [============================>.] - ETA: 0s - loss: 0.5740 -␣
 ↪accuracy:
0.8208
Epoch 31: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 139ms/step - loss: 0.
 ↪5720 -
accuracy: 0.8215 - val_loss: 0.9444 - val_accuracy: 0.7645
Epoch 32/100
45/46 [============================>.] - ETA: 0s - loss: 0.8174 -␣
 ↪accuracy:
0.7889
Epoch 32: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 139ms/step - loss: 0.
 ↪8175 -
accuracy: 0.7882 - val_loss: 0.7201 - val_accuracy: 0.8226
Epoch 33/100
45/46 [============================>.] - ETA: 0s - loss: 0.6924 -␣
 ↪accuracy:
0.8229
Epoch 33: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 137ms/step - loss: 0.
 ↪6905 -
```

```
accuracy: 0.8228 - val_loss: 0.6981 - val_accuracy: 0.8000
Epoch 34/100
45/46 [============================>.] - ETA: 0s - loss: 0.6410 -␣
 ↪accuracy:
0.8174
Epoch 34: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 137ms/step - loss: 0.
 ↪6406 -
accuracy: 0.8173 - val_loss: 0.7347 - val_accuracy: 0.8065
Epoch 35/100
45/46 [============================>.] - ETA: 0s - loss: 0.6987 -␣
 ↪accuracy:
0.8243
Epoch 35: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 136ms/step - loss: 0.
 ↪6963 -
accuracy: 0.8249 - val_loss: 1.0109 - val_accuracy: 0.7677
Epoch 36/100
45/46 [============================>.] - ETA: 0s - loss: 0.5854 -␣
 ↪accuracy:
0.8340
Epoch 36: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 136ms/step - loss: 0.
 ↪5863 -
accuracy: 0.8339 - val_loss: 0.6520 - val_accuracy: 0.8258
Epoch 37/100
45/46 [============================>.] - ETA: 0s - loss: 0.6317 -␣
 ↪accuracy:
0.8417
Epoch 37: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 136ms/step - loss: 0.
 ↪6298 -
```

```
accuracy: 0.8422 - val_loss: 0.8223 - val_accuracy: 0.7774
Epoch 38/100
45/46 [============================>.] - ETA: 0s - loss: 0.6784 -␣
 ↪accuracy:
0.8174
Epoch 38: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 135ms/step - loss: 0.
 ↪6761 -
accuracy: 0.8180 - val_loss: 1.4930 - val_accuracy: 0.7065
Epoch 39/100
45/46 [============================>.] - ETA: 0s - loss: 0.7745 -␣
 ↪accuracy:
0.8076
Epoch 39: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 136ms/step - loss: 0.
 ↪7774 -
accuracy: 0.8069 - val_loss: 0.6406 - val_accuracy: 0.8194
Epoch 40/100
45/46 [============================>.] - ETA: 0s - loss: 0.6954 -␣
 ↪accuracy:
0.8285
Epoch 40: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 136ms/step - loss: 0.
 ↪6938 -
accuracy: 0.8284 - val_loss: 1.5896 - val_accuracy: 0.6871
Epoch 41/100
45/46 [============================>.] - ETA: 0s - loss: 0.7188 -␣
 ↪accuracy:
0.8222
Epoch 41: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 136ms/step - loss: 0.
 ↪7164 -
```

```
accuracy: 0.8228 - val_loss: 0.6102 - val_accuracy: 0.8290
Epoch 42/100
45/46 [============================>.] - ETA: 0s - loss: 0.7675 -␣
 ↪accuracy:
0.8271
Epoch 42: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 136ms/step - loss: 0.
 ↪7748 -
accuracy: 0.8256 - val_loss: 0.6363 - val_accuracy: 0.8355
Epoch 43/100
45/46 [============================>.] - ETA: 0s - loss: 0.7780 -␣
 ↪accuracy:
0.8243
Epoch 43: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 137ms/step - loss: 0.
 ↪7786 -
accuracy: 0.8235 - val_loss: 0.7906 - val_accuracy: 0.8226
Epoch 44/100
45/46 [============================>.] - ETA: 0s - loss: 0.6472 -␣
 ↪accuracy:
0.8313
Epoch 44: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 137ms/step - loss: 0.
 ↪6462 -
accuracy: 0.8311 - val_loss: 0.7127 - val_accuracy: 0.8226
Epoch 45/100
45/46 [============================>.] - ETA: 0s - loss: 0.6290 -␣
 ↪accuracy:
0.8340
Epoch 45: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 137ms/step - loss: 0.
 ↪6268 -
```

```
accuracy: 0.8346 - val_loss: 1.0454 - val_accuracy: 0.7645
Epoch 46/100
45/46 [============================>.] - ETA: 0s - loss: 0.7945 -␣
 ↪accuracy:
0.8208
Epoch 46: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 137ms/step - loss: 0.
 ↪7929 -
accuracy: 0.8208 - val_loss: 1.3353 - val_accuracy: 0.7226
Epoch 47/100
45/46 [============================>.] - ETA: 0s - loss: 0.6597 -␣
 ↪accuracy:
0.8438
Epoch 47: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 137ms/step - loss: 0.
 ↪6664 -
accuracy: 0.8429 - val_loss: 0.7322 - val_accuracy: 0.8290
Epoch 48/100
45/46 [============================>.] - ETA: 0s - loss: 0.6326 -␣
 ↪accuracy:
0.8361
Epoch 48: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 137ms/step - loss: 0.
 ↪6305 -
accuracy: 0.8367 - val_loss: 0.6735 - val_accuracy: 0.8355
Epoch 49/100
45/46 [============================>.] - ETA: 0s - loss: 0.6513 -␣
 ↪accuracy:
0.8486
Epoch 49: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 136ms/step - loss: 0.
 ↪6490 -
```

```
accuracy: 0.8491 - val_loss: 0.8991 - val_accuracy: 0.8161
Epoch 50/100
45/46 [============================>.] - ETA: 0s - loss: 0.6451 -␣
 ↪accuracy:
0.8396
Epoch 50: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 136ms/step - loss: 0.
 ↪6429 -
accuracy: 0.8401 - val_loss: 1.0642 - val_accuracy: 0.7645
Epoch 51/100
45/46 [============================>.] - ETA: 0s - loss: 0.7802 -␣
 ↪accuracy:
0.8139
Epoch 51: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 136ms/step - loss: 0.
 ↪7801 -
accuracy: 0.8138 - val_loss: 0.7861 - val_accuracy: 0.7968
Epoch 52/100
45/46 [============================>.] - ETA: 0s - loss: 0.6160 -␣
 ↪accuracy:
0.8354
Epoch 52: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 136ms/step - loss: 0.
 ↪6168 -
accuracy: 0.8346 - val_loss: 0.7149 - val_accuracy: 0.8161
Epoch 53/100
45/46 [============================>.] - ETA: 0s - loss: 0.7479 -␣
 ↪accuracy:
0.8347
Epoch 53: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 136ms/step - loss: 0.
 ↪7454 -
```

```
accuracy: 0.8353 - val_loss: 1.2936 - val_accuracy: 0.7548
Epoch 54/100
45/46 [============================>.] - ETA: 0s - loss: 0.5644 -␣
 ↪accuracy:
0.8562
Epoch 54: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 136ms/step - loss: 0.
 ↪5641 -
accuracy: 0.8561 - val_loss: 0.6719 - val_accuracy: 0.8355
Epoch 55/100
45/46 [============================>.] - ETA: 0s - loss: 0.5277 -␣
 ↪accuracy:
0.8708
Epoch 55: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 137ms/step - loss: 0.
 ↪5262 -
accuracy: 0.8713 - val_loss: 0.8775 - val_accuracy: 0.7774
Epoch 56/100
45/46 [============================>.] - ETA: 0s - loss: 0.6446 -␣
 ↪accuracy:
0.8313
Epoch 56: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 136ms/step - loss: 0.
 ↪6426 -
accuracy: 0.8318 - val_loss: 0.7500 - val_accuracy: 0.8194
Epoch 57/100
45/46 [============================>.] - ETA: 0s - loss: 0.5912 -␣
 ↪accuracy:
0.8486
Epoch 57: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 137ms/step - loss: 0.
 ↪5986 -
```

```
accuracy: 0.8478 - val_loss: 0.8312 - val_accuracy: 0.8129
Epoch 58/100
45/46 [============================>.] - ETA: 0s - loss: 0.5106 -␣
 ↪accuracy:
0.8653
Epoch 58: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 137ms/step - loss: 0.
 ↪5125 -
accuracy: 0.8651 - val_loss: 0.7408 - val_accuracy: 0.8323
Epoch 59/100
45/46 [============================>.] - ETA: 0s - loss: 0.5349 -␣
 ↪accuracy:
0.8667
Epoch 59: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 137ms/step - loss: 0.
 ↪5332 -
accuracy: 0.8671 - val_loss: 0.7508 - val_accuracy: 0.8226
Epoch 60/100
45/46 [============================>.] - ETA: 0s - loss: 0.5206 -␣
 ↪accuracy:
0.8687
Epoch 60: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 136ms/step - loss: 0.
 ↪5224 -
accuracy: 0.8685 - val_loss: 0.9195 - val_accuracy: 0.8000
Epoch 61/100
45/46 [============================>.] - ETA: 0s - loss: 0.8368 -␣
 ↪accuracy:
0.8188
Epoch 61: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 136ms/step - loss: 0.
 ↪8357 -
```

```
accuracy: 0.8187 - val_loss: 0.9183 - val_accuracy: 0.7903
Epoch 62/100
45/46 [============================>.] - ETA: 0s - loss: 0.7791 -␣
 ↪accuracy:
0.8250
Epoch 62: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 137ms/step - loss: 0.
 ↪7848 -
accuracy: 0.8249 - val_loss: 0.9089 - val_accuracy: 0.7839
Epoch 63/100
45/46 [============================>.] - ETA: 0s - loss: 0.6714 -␣
 ↪accuracy:
0.8472
Epoch 63: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 137ms/step - loss: 0.
 ↪6698 -
accuracy: 0.8478 - val_loss: 1.0451 - val_accuracy: 0.7968
Epoch 64/100
45/46 [============================>.] - ETA: 0s - loss: 0.5270 -␣
 ↪accuracy:
0.8667
Epoch 64: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 136ms/step - loss: 0.
 ↪5316 -
accuracy: 0.8657 - val_loss: 0.7913 - val_accuracy: 0.8065
Epoch 65/100
45/46 [============================>.] - ETA: 0s - loss: 0.6756 -␣
 ↪accuracy:
0.8458
Epoch 65: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 136ms/step - loss: 0.
 ↪6771 -
```

```
accuracy: 0.8450 - val_loss: 1.3586 - val_accuracy: 0.7129
Epoch 66/100
45/46 [============================>.] - ETA: 0s - loss: 0.8111 -␣
 ↪accuracy:
0.8313
Epoch 66: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 136ms/step - loss: 0.
 ↪8136 -
accuracy: 0.8304 - val_loss: 0.9052 - val_accuracy: 0.7935
Epoch 67/100
45/46 [============================>.] - ETA: 0s - loss: 0.4918 -␣
 ↪accuracy:
0.8819
Epoch 67: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 136ms/step - loss: 0.
 ↪4991 -
accuracy: 0.8810 - val_loss: 0.9107 - val_accuracy: 0.8194
Epoch 68/100
45/46 [============================>.] - ETA: 0s - loss: 0.6177 -␣
 ↪accuracy:
0.8535
Epoch 68: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 137ms/step - loss: 0.
 ↪6210 -
accuracy: 0.8526 - val_loss: 0.8337 - val_accuracy: 0.8129
Epoch 69/100
45/46 [============================>.] - ETA: 0s - loss: 0.6059 -␣
 ↪accuracy:
0.8632
Epoch 69: val_accuracy did not improve from 0.84194
46/46 [==============================] - 6s 137ms/step - loss: 0.
 ↪6107 -
```

```
accuracy: 0.8623 - val_loss: 1.0429 - val_accuracy: 0.8290
Epoch 70/100
45/46 [============================>.] - ETA: 0s - loss: 0.5182 -␣
 ↪accuracy:
0.8618
Epoch 70: val_accuracy improved from 0.84194 to 0.84516, saving␣
 ↪model to
/content/gdrive/MyDrive/MA981-7-FY/Dataset/RESNET50/070-0.85

WARNING:absl:Found untraced functions such as␣
 ↪_jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while␣
 ↪saving (showing
5 of 53). These functions will not be directly callable after␣
 ↪loading.

46/46 [==============================] - 31s 677ms/step - loss: 0.
 ↪5164 -
accuracy: 0.8623 - val_loss: 0.7455 - val_accuracy: 0.8452
Epoch 71/100
45/46 [============================>.] - ETA: 0s - loss: 0.5613 -␣
 ↪accuracy:
0.8611
Epoch 71: val_accuracy did not improve from 0.84516
46/46 [==============================] - 6s 136ms/step - loss: 0.
 ↪5593 -
accuracy: 0.8616 - val_loss: 0.6622 - val_accuracy: 0.8290
Epoch 72/100
45/46 [============================>.] - ETA: 0s - loss: 0.5612 -␣
 ↪accuracy:
0.8562
Epoch 72: val_accuracy did not improve from 0.84516
```

```
46/46 [==============================] - 6s 136ms/step - loss: 0.
 ↪5603 -
accuracy: 0.8561 - val_loss: 1.0670 - val_accuracy: 0.7935
Epoch 73/100
45/46 [============================>.] - ETA: 0s - loss: 0.5990 -␣
 ↪accuracy:
0.8528
Epoch 73: val_accuracy did not improve from 0.84516
46/46 [==============================] - 6s 137ms/step - loss: 0.
 ↪6039 -
accuracy: 0.8519 - val_loss: 0.7515 - val_accuracy: 0.8419
Epoch 74/100
45/46 [============================>.] - ETA: 0s - loss: 0.6544 -␣
 ↪accuracy:
0.8528
Epoch 74: val_accuracy did not improve from 0.84516
46/46 [==============================] - 6s 138ms/step - loss: 0.
 ↪6529 -
accuracy: 0.8526 - val_loss: 0.7075 - val_accuracy: 0.8323
Epoch 75/100
45/46 [============================>.] - ETA: 0s - loss: 0.8060 -␣
 ↪accuracy:
0.8396
Epoch 75: val_accuracy did not improve from 0.84516
46/46 [==============================] - 6s 139ms/step - loss: 0.
 ↪8041 -
accuracy: 0.8394 - val_loss: 0.8277 - val_accuracy: 0.8194
Epoch 76/100
45/46 [============================>.] - ETA: 0s - loss: 0.7358 -␣
 ↪accuracy:
0.8382
```

```
Epoch 76: val_accuracy did not improve from 0.84516
46/46 [==============================] - 6s 139ms/step - loss: 0.
 ↪7350 -
accuracy: 0.8381 - val_loss: 1.0406 - val_accuracy: 0.7806
Epoch 77/100
45/46 [============================>.] - ETA: 0s - loss: 0.6900 -␣
 ↪accuracy:
0.8632
Epoch 77: val_accuracy did not improve from 0.84516
46/46 [==============================] - 6s 138ms/step - loss: 0.
 ↪6994 -
accuracy: 0.8616 - val_loss: 1.4604 - val_accuracy: 0.7290
Epoch 78/100
45/46 [============================>.] - ETA: 0s - loss: 0.7954 -␣
 ↪accuracy:
0.8354
Epoch 78: val_accuracy did not improve from 0.84516
46/46 [==============================] - 6s 137ms/step - loss: 0.
 ↪7927 -
accuracy: 0.8360 - val_loss: 0.9017 - val_accuracy: 0.8290
Epoch 79/100
45/46 [============================>.] - ETA: 0s - loss: 0.4998 -␣
 ↪accuracy:
0.8764
Epoch 79: val_accuracy did not improve from 0.84516
46/46 [==============================] - 6s 137ms/step - loss: 0.
 ↪4980 -
accuracy: 0.8768 - val_loss: 0.7964 - val_accuracy: 0.8355
Epoch 80/100
45/46 [============================>.] - ETA: 0s - loss: 0.5940 -␣
 ↪accuracy:
```

```
0.8562
Epoch 80: val_accuracy did not improve from 0.84516
46/46 [==============================] - 6s 137ms/step - loss: 0.
 ↪5923 -
accuracy: 0.8567 - val_loss: 0.7388 - val_accuracy: 0.8387
Epoch 81/100
45/46 [============================>.] - ETA: 0s - loss: 0.4982 -␣
 ↪accuracy:
0.8674
Epoch 81: val_accuracy did not improve from 0.84516
46/46 [==============================] - 6s 136ms/step - loss: 0.
 ↪5030 -
accuracy: 0.8664 - val_loss: 0.9598 - val_accuracy: 0.8065
Epoch 82/100
45/46 [============================>.] - ETA: 0s - loss: 0.4870 -␣
 ↪accuracy:
0.8799
Epoch 82: val_accuracy did not improve from 0.84516
46/46 [==============================] - 6s 136ms/step - loss: 0.
 ↪4853 -
accuracy: 0.8803 - val_loss: 0.7664 - val_accuracy: 0.8387
Epoch 83/100
45/46 [============================>.] - ETA: 0s - loss: 0.5602 -␣
 ↪accuracy:
0.8826
Epoch 83: val_accuracy did not improve from 0.84516
46/46 [==============================] - 6s 136ms/step - loss: 0.
 ↪5587 -
accuracy: 0.8830 - val_loss: 0.7574 - val_accuracy: 0.8387
Epoch 84/100
45/46 [============================>.] - ETA: 0s - loss: 0.6258 -␣
 ↪accuracy:
```

```
0.8507
Epoch 84: val_accuracy did not improve from 0.84516
46/46 [==============================] - 6s 135ms/step - loss: 0.
 ↪6237 -
accuracy: 0.8512 - val_loss: 1.0053 - val_accuracy: 0.7871
Epoch 85/100
45/46 [============================>.] - ETA: 0s - loss: 0.6852 -␣
 ↪accuracy:
0.8500
Epoch 85: val_accuracy did not improve from 0.84516
46/46 [==============================] - 6s 136ms/step - loss: 0.
 ↪6838 -
accuracy: 0.8498 - val_loss: 1.3291 - val_accuracy: 0.7968
Epoch 86/100
45/46 [============================>.] - ETA: 0s - loss: 0.6508 -␣
 ↪accuracy:
0.8625
Epoch 86: val_accuracy did not improve from 0.84516
46/46 [==============================] - 6s 136ms/step - loss: 0.
 ↪6557 -
accuracy: 0.8623 - val_loss: 0.9294 - val_accuracy: 0.8129
Epoch 87/100
45/46 [============================>.] - ETA: 0s - loss: 0.6063 -␣
 ↪accuracy:
0.8535
Epoch 87: val_accuracy did not improve from 0.84516
46/46 [==============================] - 6s 137ms/step - loss: 0.
 ↪6051 -
accuracy: 0.8533 - val_loss: 1.1114 - val_accuracy: 0.8194
Epoch 88/100
45/46 [============================>.] - ETA: 0s - loss: 0.5658 -␣
 ↪accuracy:
```

```
0.8687
Epoch 88: val_accuracy did not improve from 0.84516
46/46 [==============================] - 6s 137ms/step - loss: 0.
 ↪5638 -
accuracy: 0.8692 - val_loss: 1.4879 - val_accuracy: 0.7226
Epoch 89/100
45/46 [============================>.] - ETA: 0s - loss: 0.6050 -␣
 ↪accuracy:
0.8521
Epoch 89: val_accuracy did not improve from 0.84516
46/46 [==============================] - 6s 136ms/step - loss: 0.
 ↪6057 -
accuracy: 0.8512 - val_loss: 0.7859 - val_accuracy: 0.8194
Epoch 90/100
45/46 [============================>.] - ETA: 0s - loss: 0.6167 -␣
 ↪accuracy:
0.8590
Epoch 90: val_accuracy did not improve from 0.84516
46/46 [==============================] - 6s 135ms/step - loss: 0.
 ↪6147 -
accuracy: 0.8595 - val_loss: 0.9607 - val_accuracy: 0.7935
Epoch 91/100
45/46 [============================>.] - ETA: 0s - loss: 0.4896 -␣
 ↪accuracy:
0.8826
Epoch 91: val_accuracy did not improve from 0.84516
46/46 [==============================] - 6s 136ms/step - loss: 0.
 ↪4895 -
accuracy: 0.8824 - val_loss: 1.3310 - val_accuracy: 0.7452
Epoch 92/100
45/46 [============================>.] - ETA: 0s - loss: 0.5192 -␣
 ↪accuracy:
```

```
0.8819
Epoch 92: val_accuracy did not improve from 0.84516
46/46 [==============================] - 6s 137ms/step - loss: 0.
 ↪5176 -
accuracy: 0.8824 - val_loss: 0.7782 - val_accuracy: 0.8323
Epoch 93/100
45/46 [============================>.] - ETA: 0s - loss: 0.4859 -␣
 ↪accuracy:
0.8736
Epoch 93: val_accuracy did not improve from 0.84516
46/46 [==============================] - 6s 137ms/step - loss: 0.
 ↪4955 -
accuracy: 0.8727 - val_loss: 1.1382 - val_accuracy: 0.7806
Epoch 94/100
45/46 [============================>.] - ETA: 0s - loss: 0.5059 -␣
 ↪accuracy:
0.8771
Epoch 94: val_accuracy did not improve from 0.84516
46/46 [==============================] - 6s 136ms/step - loss: 0.
 ↪5083 -
accuracy: 0.8768 - val_loss: 0.8672 - val_accuracy: 0.8258
Epoch 95/100
45/46 [============================>.] - ETA: 0s - loss: 0.4599 -␣
 ↪accuracy:
0.8792
Epoch 95: val_accuracy did not improve from 0.84516
46/46 [==============================] - 6s 136ms/step - loss: 0.
 ↪4586 -
accuracy: 0.8796 - val_loss: 0.8856 - val_accuracy: 0.8323
Epoch 96/100
45/46 [============================>.] - ETA: 0s - loss: 0.5765 -␣
 ↪accuracy:
```

```
0.8785
Epoch 96: val_accuracy did not improve from 0.84516
46/46 [==============================] - 6s 137ms/step - loss: 0.
 ↪5797 -
accuracy: 0.8782 - val_loss: 0.8005 - val_accuracy: 0.8419
Epoch 97/100
45/46 [============================>.] - ETA: 0s - loss: 0.4496 -␣
 ↪accuracy:
0.8896
Epoch 97: val_accuracy did not improve from 0.84516
46/46 [==============================] - 6s 137ms/step - loss: 0.
 ↪4501 -
accuracy: 0.8893 - val_loss: 1.0579 - val_accuracy: 0.8129
Epoch 98/100
45/46 [============================>.] - ETA: 0s - loss: 0.7811 -␣
 ↪accuracy:
0.8500
Epoch 98: val_accuracy did not improve from 0.84516
46/46 [==============================] - 6s 136ms/step - loss: 0.
 ↪7807 -
accuracy: 0.8498 - val_loss: 0.7905 - val_accuracy: 0.8290
Epoch 99/100
45/46 [============================>.] - ETA: 0s - loss: 0.5455 -␣
 ↪accuracy:
0.8583
Epoch 99: val_accuracy did not improve from 0.84516
46/46 [==============================] - 6s 137ms/step - loss: 0.
 ↪5436 -
accuracy: 0.8588 - val_loss: 1.3822 - val_accuracy: 0.7452
Epoch 100/100
45/46 [============================>.] - ETA: 0s - loss: 0.9415 -␣
 ↪accuracy:
```

```
0.8229
Epoch 100: val_accuracy did not improve from 0.84516
46/46 [==============================] - 6s 137ms/step - loss: 0.
↪9410 -
accuracy: 0.8228 - val_loss: 1.2433 - val_accuracy: 0.7548
Elapsed time: 0:13:31.6
```

Maximum validation accuracy at 80th epoch

```
[ ]:
bestresnet50_model = load_model(filepath='/content/gdrive/MyDrive/
↪MA981-7-FY/Dataset/RESNET50/080-0.86')

WARNING:tensorflow:SavedModel saved prior to TF 2.5 detected when␣
↪loading Keras
model. Please ensure that you are saving the model with model.
↪save() or
tf.keras.models.save_model(), *NOT* tf.saved_model.save(). To␣
↪confirm, there
should be a file named "keras_metadata.pb" in the SavedModel␣
↪directory.
```

```
[ ]:
loss, acc = bestresnet50_model.evaluate(x=X_test, y=Y_test)

10/10 [==============================] - 2s 105ms/step - loss: 0.
↪2060 -
accuracy: 0.9613
```

```
[ ]:
print (f"Test Loss = {loss}")
print (f"Test Accuracy = {acc}")

Test Loss = 0.2059873640537262
Test Accuracy = 0.9612902998924255
```

```
[ ]:
Y_test_prob_r = bestresnet50_model.predict(X_test)
```

```
10/10 [==============================] - 2s 102ms/step
```

Calculating F1 score

[ ]:
```
f1score_val = compute_f1_score(Y_test, Y_test_prob_r)
print(f"F1 score: {f1score_val}")
```

```
F1 score: 0.9653179190751444
```

[ ]:
```
print("Training Data:")
data_percentage(Y_train)
print("Validation Data:")
data_percentage(Y_val)
print("Testing Data:")
data_percentage(Y_test)
```

```
Training Data:
Number of examples: 1445
Percentage of positive examples: 51.4878892733564%, number of pos␣
↪examples: 744
Percentage of negative examples: 48.5121107266436%, number of neg␣
↪examples: 701
Validation Data:
Number of examples: 310
Percentage of positive examples: 54.83870967741935%, number of pos␣
↪examples: 170
Percentage of negative examples: 45.16129032258065%, number of neg␣
↪examples: 140
Testing Data:
Number of examples: 310
Percentage of positive examples: 55.16129032258065%, number of pos␣
↪examples: 171
Percentage of negative examples: 44.83870967741935%, number of neg␣
↪examples: 139
```

Trying with our own model with less number of total parameters.

[ ]:
```
def model_built(input_shape):
  X_input = Input(input_shape)
  X = ZeroPadding2D((2, 2))(X_input)
  Conv2D(32,(7,7),strides=(1,1),name="conv0")(X)
  X=BatchNormalization(axis=3,name="bn0")(X)
  X=Activation("relu")(X)
  X=MaxPool2D((4,4),name="max_pool_0")(X)
  X=MaxPool2D((4,4),name="max_pool_1")(X)
  X=Flatten()(X)
  X=Dense(1,activation="sigmoid")(X)
 ␣
 ↪model=Model(inputs=X_input,outputs=X,name="tumor_detection_model")

  return model
```

[ ]:
```
IMG_shape=(image_width ,image_height ,3)
```

[ ]:
```
own_model=model_built(IMG_shape)
```

[ ]:
```
own_model.summary()
```

```
Model: "tumor_detection_model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_4 (InputLayer)        [(None, 240, 240, 3)]     0

 zero_padding2d (ZeroPadding  (None, 244, 244, 3)      0
 2D)

 bn0 (BatchNormalization)    (None, 244, 244, 3)       12
```

```
 activation_94 (Activation)   (None, 244, 244, 3)        0

 max_pool_0 (MaxPooling2D)    (None, 61, 61, 3)          0

 max_pool_1 (MaxPooling2D)    (None, 15, 15, 3)          0

 flatten_3 (Flatten)          (None, 675)                0

 dense_3 (Dense)              (None, 1)                  676

=================================================================
Total params: 688
Trainable params: 682
Non-trainable params: 6
_________________________________________________________________
```

[ ]:
```
own_model.compile(optimizer='adam', loss='binary_crossentropy',␣
↪metrics=['accuracy'])
```

[ ]:
```
checkpoint_path4 = "/content/gdrive/MyDrive/MA981-7-FY/Dataset/
↪try_with_own_model/{epoch:03d}-{val_accuracy:.2f}"
checkpoint4 =␣
↪ModelCheckpoint(filepath=checkpoint_path4,monitor="val_accuracy",␣
↪verbose=1, save_best_only=True, mode='auto',save_freq="epoch")
```

[ ]:
```
start_time = time.time()

own_model.fit(x=X_train, y=Y_train, batch_size=32, epochs=125,␣
↪validation_data=(X_val, Y_val),callbacks=[checkpoint4])

end_time = time.time()
execution_time = (end_time - start_time)
print(f"Elapsed time: {hms_string(execution_time)}")
```

```
Epoch 1/125
44/46 [============================>..] - ETA: 0s - loss: 0.6887 -␣
 ↪accuracy:
0.6342
Epoch 1: val_accuracy improved from -inf to 0.67742, saving model␣
 ↪to
/content/gdrive/MyDrive/MA981-7-FY/Dataset/try_with_own_model/
 ↪001-0.68
46/46 [==============================] - 6s 98ms/step - loss: 0.
 ↪6879 - accuracy:
0.6381 - val_loss: 0.6220 - val_accuracy: 0.6774
Epoch 2/125
45/46 [============================>.] - ETA: 0s - loss: 0.6226 -␣
 ↪accuracy:
0.6618
Epoch 2: val_accuracy improved from 0.67742 to 0.69032, saving␣
 ↪model to
/content/gdrive/MyDrive/MA981-7-FY/Dataset/try_with_own_model/
 ↪002-0.69
46/46 [==============================] - 2s 38ms/step - loss: 0.
 ↪6233 - accuracy:
0.6609 - val_loss: 0.6027 - val_accuracy: 0.6903
Epoch 3/125
45/46 [============================>.] - ETA: 0s - loss: 0.6003 -␣
 ↪accuracy:
0.6944
Epoch 3: val_accuracy improved from 0.69032 to 0.69355, saving␣
 ↪model to
/content/gdrive/MyDrive/MA981-7-FY/Dataset/try_with_own_model/
 ↪003-0.69
```

```
46/46 [==============================] - 2s 36ms/step - loss: 0.
↪6006 - accuracy:
0.6948 - val_loss: 0.6090 - val_accuracy: 0.6935
Epoch 4/125
45/46 [============================>.] - ETA: 0s - loss: 0.5872 -␣
↪accuracy:
0.6944
Epoch 4: val_accuracy improved from 0.69355 to 0.70000, saving␣
↪model to
/content/gdrive/MyDrive/MA981-7-FY/Dataset/try_with_own_model/
↪004-0.70
46/46 [==============================] - 2s 36ms/step - loss: 0.
↪5866 - accuracy:
0.6948 - val_loss: 0.6065 - val_accuracy: 0.7000
Epoch 5/125
45/46 [============================>.] - ETA: 0s - loss: 0.5886 -␣
↪accuracy:
0.6938
Epoch 5: val_accuracy did not improve from 0.70000
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪5884 - accuracy:
0.6934 - val_loss: 0.6037 - val_accuracy: 0.6774
Epoch 6/125
45/46 [============================>.] - ETA: 0s - loss: 0.5917 -␣
↪accuracy:
0.6965
Epoch 6: val_accuracy did not improve from 0.70000
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪5914 - accuracy:
0.6969 - val_loss: 0.5800 - val_accuracy: 0.6871
Epoch 7/125
```

```
45/46 [============================>.] - ETA: 0s - loss: 0.5647 -␣
 ↪accuracy:
0.7056
Epoch 7: val_accuracy did not improve from 0.70000
46/46 [==============================] - 1s 17ms/step - loss: 0.
 ↪5645 - accuracy:
0.7059 - val_loss: 0.6232 - val_accuracy: 0.6710
Epoch 8/125
45/46 [============================>.] - ETA: 0s - loss: 0.5612 -␣
 ↪accuracy:
0.7146
Epoch 8: val_accuracy improved from 0.70000 to 0.72258, saving␣
 ↪model to
/content/gdrive/MyDrive/MA981-7-FY/Dataset/try_with_own_model/
 ↪008-0.72
46/46 [==============================] - 2s 36ms/step - loss: 0.
 ↪5619 - accuracy:
0.7149 - val_loss: 0.5893 - val_accuracy: 0.7226
Epoch 9/125
45/46 [============================>.] - ETA: 0s - loss: 0.5497 -␣
 ↪accuracy:
0.7181
Epoch 9: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
 ↪5502 - accuracy:
0.7176 - val_loss: 0.5794 - val_accuracy: 0.7194
Epoch 10/125
45/46 [============================>.] - ETA: 0s - loss: 0.5529 -␣
 ↪accuracy:
0.7201
Epoch 10: val_accuracy did not improve from 0.72258
```

```
46/46 [==============================] - 1s 17ms/step - loss: 0.
 ↪5530 - accuracy:
0.7204 - val_loss: 0.5793 - val_accuracy: 0.6968
Epoch 11/125
45/46 [============================>.] - ETA: 0s - loss: 0.5466 -␣
 ↪accuracy:
0.7292
Epoch 11: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
 ↪5469 - accuracy:
0.7294 - val_loss: 0.6094 - val_accuracy: 0.7032
Epoch 12/125
45/46 [============================>.] - ETA: 0s - loss: 0.5458 -␣
 ↪accuracy:
0.7264
Epoch 12: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
 ↪5449 - accuracy:
0.7266 - val_loss: 0.5911 - val_accuracy: 0.7065
Epoch 13/125
45/46 [============================>.] - ETA: 0s - loss: 0.5453 -␣
 ↪accuracy:
0.7201
Epoch 13: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
 ↪5454 - accuracy:
0.7197 - val_loss: 0.5861 - val_accuracy: 0.7097
Epoch 14/125
45/46 [============================>.] - ETA: 0s - loss: 0.5410 -␣
 ↪accuracy:
0.7264
```

```
Epoch 14: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪5421 - accuracy:
0.7260 - val_loss: 0.6023 - val_accuracy: 0.7129
Epoch 15/125
45/46 [============================>.] - ETA: 0s - loss: 0.5426 -␣
↪accuracy:
0.7312
Epoch 15: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪5439 - accuracy:
0.7301 - val_loss: 0.5824 - val_accuracy: 0.7032
Epoch 16/125
45/46 [============================>.] - ETA: 0s - loss: 0.5380 -␣
↪accuracy:
0.7271
Epoch 16: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪5379 - accuracy:
0.7273 - val_loss: 0.5846 - val_accuracy: 0.6871
Epoch 17/125
45/46 [============================>.] - ETA: 0s - loss: 0.5321 -␣
↪accuracy:
0.7257
Epoch 17: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪5318 - accuracy:
0.7260 - val_loss: 0.5809 - val_accuracy: 0.6871
Epoch 18/125
45/46 [============================>.] - ETA: 0s - loss: 0.5260 -␣
↪accuracy:
```

```
0.7326
Epoch 18: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪5273 - accuracy:
0.7315 - val_loss: 0.5980 - val_accuracy: 0.6871
Epoch 19/125
45/46 [============================>.] - ETA: 0s - loss: 0.5409 -␣
↪accuracy:
0.7285
Epoch 19: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪5405 - accuracy:
0.7287 - val_loss: 0.5923 - val_accuracy: 0.7161
Epoch 20/125
45/46 [============================>.] - ETA: 0s - loss: 0.5536 -␣
↪accuracy:
0.7236
Epoch 20: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪5530 - accuracy:
0.7239 - val_loss: 0.5809 - val_accuracy: 0.6903
Epoch 21/125
45/46 [============================>.] - ETA: 0s - loss: 0.5709 -␣
↪accuracy:
0.7118
Epoch 21: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪5705 - accuracy:
0.7114 - val_loss: 0.6568 - val_accuracy: 0.6710
Epoch 22/125
44/46 [===========================>..] - ETA: 0s - loss: 0.5416 -␣
↪accuracy:
```

```
0.7273
Epoch 22: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪5404 - accuracy:
0.7273 - val_loss: 0.6031 - val_accuracy: 0.7097
Epoch 23/125
45/46 [============================>.] - ETA: 0s - loss: 0.5192 -␣
↪accuracy:
0.7347
Epoch 23: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪5198 - accuracy:
0.7343 - val_loss: 0.6198 - val_accuracy: 0.7000
Epoch 24/125
45/46 [============================>.] - ETA: 0s - loss: 0.5301 -␣
↪accuracy:
0.7340
Epoch 24: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪5303 - accuracy:
0.7336 - val_loss: 0.5867 - val_accuracy: 0.6742
Epoch 25/125
44/46 [===========================>..] - ETA: 0s - loss: 0.5245 -␣
↪accuracy:
0.7266
Epoch 25: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 18ms/step - loss: 0.
↪5224 - accuracy:
0.7260 - val_loss: 0.6183 - val_accuracy: 0.6968
Epoch 26/125
45/46 [============================>.] - ETA: 0s - loss: 0.5171 -␣
↪accuracy:
```

```
0.7424
Epoch 26: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 18ms/step - loss: 0.
↪5163 - accuracy:
0.7433 - val_loss: 0.5998 - val_accuracy: 0.7161
Epoch 27/125
45/46 [============================>.] - ETA: 0s - loss: 0.5194 -␣
↪accuracy:
0.7222
Epoch 27: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪5188 - accuracy:
0.7232 - val_loss: 0.5904 - val_accuracy: 0.7129
Epoch 28/125
45/46 [============================>.] - ETA: 0s - loss: 0.5149 -␣
↪accuracy:
0.7437
Epoch 28: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪5148 - accuracy:
0.7439 - val_loss: 0.5902 - val_accuracy: 0.6742
Epoch 29/125
45/46 [============================>.] - ETA: 0s - loss: 0.5108 -␣
↪accuracy:
0.7375
Epoch 29: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪5110 - accuracy:
0.7370 - val_loss: 0.5902 - val_accuracy: 0.7000
Epoch 30/125
45/46 [============================>.] - ETA: 0s - loss: 0.5313 -␣
↪accuracy:
```

```
0.7354
Epoch 30: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪5313 - accuracy:
0.7356 - val_loss: 0.6037 - val_accuracy: 0.6968
Epoch 31/125
45/46 [============================>.] - ETA: 0s - loss: 0.5132 -␣
↪accuracy:
0.7444
Epoch 31: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪5122 - accuracy:
0.7453 - val_loss: 0.5995 - val_accuracy: 0.6839
Epoch 32/125
44/46 [===========================>..] - ETA: 0s - loss: 0.5192 -␣
↪accuracy:
0.7301
Epoch 32: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪5187 - accuracy:
0.7301 - val_loss: 0.5966 - val_accuracy: 0.6806
Epoch 33/125
45/46 [============================>.] - ETA: 0s - loss: 0.5145 -␣
↪accuracy:
0.7375
Epoch 33: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪5133 - accuracy:
0.7384 - val_loss: 0.5896 - val_accuracy: 0.7000
Epoch 34/125
45/46 [============================>.] - ETA: 0s - loss: 0.5116 -␣
↪accuracy:
```

```
0.7368
Epoch 34: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪5110 - accuracy:
0.7370 - val_loss: 0.5914 - val_accuracy: 0.6968
Epoch 35/125
45/46 [============================>.] - ETA: 0s - loss: 0.5036 -␣
↪accuracy:
0.7458
Epoch 35: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪5028 - accuracy:
0.7467 - val_loss: 0.5941 - val_accuracy: 0.7032
Epoch 36/125
45/46 [============================>.] - ETA: 0s - loss: 0.5024 -␣
↪accuracy:
0.7389
Epoch 36: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪5025 - accuracy:
0.7398 - val_loss: 0.6075 - val_accuracy: 0.7161
Epoch 37/125
45/46 [============================>.] - ETA: 0s - loss: 0.5070 -␣
↪accuracy:
0.7479
Epoch 37: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪5081 - accuracy:
0.7467 - val_loss: 0.5986 - val_accuracy: 0.7129
Epoch 38/125
45/46 [============================>.] - ETA: 0s - loss: 0.5088 -␣
↪accuracy:
```

```
0.7417
Epoch 38: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪5095 - accuracy:
0.7419 - val_loss: 0.6214 - val_accuracy: 0.6903
Epoch 39/125
45/46 [============================>.] - ETA: 0s - loss: 0.5040 -␣
↪accuracy:
0.7625
Epoch 39: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪5036 - accuracy:
0.7626 - val_loss: 0.5924 - val_accuracy: 0.6774
Epoch 40/125
45/46 [============================>.] - ETA: 0s - loss: 0.5079 -␣
↪accuracy:
0.7465
Epoch 40: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪5074 - accuracy:
0.7467 - val_loss: 0.5933 - val_accuracy: 0.7032
Epoch 41/125
45/46 [============================>.] - ETA: 0s - loss: 0.5121 -␣
↪accuracy:
0.7465
Epoch 41: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪5116 - accuracy:
0.7467 - val_loss: 0.6433 - val_accuracy: 0.6871
Epoch 42/125
45/46 [============================>.] - ETA: 0s - loss: 0.5017 -␣
↪accuracy:
```

```
0.7493
Epoch 42: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪5016 - accuracy:
0.7495 - val_loss: 0.6309 - val_accuracy: 0.6806
Epoch 43/125
45/46 [============================>.] - ETA: 0s - loss: 0.5031 -␣
↪accuracy:
0.7493
Epoch 43: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪5025 - accuracy:
0.7495 - val_loss: 0.5932 - val_accuracy: 0.6677
Epoch 44/125
45/46 [============================>.] - ETA: 0s - loss: 0.4988 -␣
↪accuracy:
0.7437
Epoch 44: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4979 - accuracy:
0.7446 - val_loss: 0.6073 - val_accuracy: 0.6839
Epoch 45/125
45/46 [============================>.] - ETA: 0s - loss: 0.4994 -␣
↪accuracy:
0.7576
Epoch 45: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4999 - accuracy:
0.7571 - val_loss: 0.5934 - val_accuracy: 0.6839
Epoch 46/125
45/46 [============================>.] - ETA: 0s - loss: 0.4991 -␣
↪accuracy:
```

```
0.7444
Epoch 46: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4987 - accuracy:
0.7446 - val_loss: 0.6328 - val_accuracy: 0.6806
Epoch 47/125
45/46 [============================>.] - ETA: 0s - loss: 0.5151 -␣
↪accuracy:
0.7431
Epoch 47: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪5140 - accuracy:
0.7439 - val_loss: 0.6212 - val_accuracy: 0.6742
Epoch 48/125
45/46 [============================>.] - ETA: 0s - loss: 0.4952 -␣
↪accuracy:
0.7563
Epoch 48: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4948 - accuracy:
0.7564 - val_loss: 0.5996 - val_accuracy: 0.6968
Epoch 49/125
45/46 [============================>.] - ETA: 0s - loss: 0.4990 -␣
↪accuracy:
0.7563
Epoch 49: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4986 - accuracy:
0.7564 - val_loss: 0.5987 - val_accuracy: 0.6613
Epoch 50/125
45/46 [============================>.] - ETA: 0s - loss: 0.4983 -␣
↪accuracy:
```

```
0.7542
Epoch 50: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4990 - accuracy:
0.7529 - val_loss: 0.6003 - val_accuracy: 0.6806
Epoch 51/125
44/46 [===========================>..] - ETA: 0s - loss: 0.4997 -␣
↪accuracy:
0.7607
Epoch 51: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪5002 - accuracy:
0.7612 - val_loss: 0.6013 - val_accuracy: 0.6806
Epoch 52/125
45/46 [============================>.] - ETA: 0s - loss: 0.4944 -␣
↪accuracy:
0.7451
Epoch 52: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4946 - accuracy:
0.7446 - val_loss: 0.5988 - val_accuracy: 0.6806
Epoch 53/125
45/46 [============================>.] - ETA: 0s - loss: 0.4890 -␣
↪accuracy:
0.7542
Epoch 53: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4885 - accuracy:
0.7550 - val_loss: 0.6122 - val_accuracy: 0.6677
Epoch 54/125
45/46 [============================>.] - ETA: 0s - loss: 0.4965 -␣
↪accuracy:
```

```
0.7556
Epoch 54: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4967 - accuracy:
0.7557 - val_loss: 0.6015 - val_accuracy: 0.6774
Epoch 55/125
45/46 [============================>.] - ETA: 0s - loss: 0.4996 -␣
↪accuracy:
0.7500
Epoch 55: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4990 - accuracy:
0.7509 - val_loss: 0.5962 - val_accuracy: 0.6871
Epoch 56/125
45/46 [============================>.] - ETA: 0s - loss: 0.5023 -␣
↪accuracy:
0.7500
Epoch 56: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪5020 - accuracy:
0.7495 - val_loss: 0.5984 - val_accuracy: 0.6774
Epoch 57/125
45/46 [============================>.] - ETA: 0s - loss: 0.4970 -␣
↪accuracy:
0.7514
Epoch 57: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4967 - accuracy:
0.7516 - val_loss: 0.6272 - val_accuracy: 0.6742
Epoch 58/125
45/46 [============================>.] - ETA: 0s - loss: 0.4929 -␣
↪accuracy:
```

```
0.7549
Epoch 58: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4919 - accuracy:
0.7557 - val_loss: 0.5969 - val_accuracy: 0.6742
Epoch 59/125
45/46 [============================>.] - ETA: 0s - loss: 0.4974 -␣
↪accuracy:
0.7549
Epoch 59: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4963 - accuracy:
0.7557 - val_loss: 0.5968 - val_accuracy: 0.6839
Epoch 60/125
45/46 [============================>.] - ETA: 0s - loss: 0.4930 -␣
↪accuracy:
0.7556
Epoch 60: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4932 - accuracy:
0.7550 - val_loss: 0.6632 - val_accuracy: 0.6839
Epoch 61/125
44/46 [===========================>..] - ETA: 0s - loss: 0.5027 -␣
↪accuracy:
0.7344
Epoch 61: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪5023 - accuracy:
0.7363 - val_loss: 0.5991 - val_accuracy: 0.6742
Epoch 62/125
45/46 [============================>.] - ETA: 0s - loss: 0.4945 -␣
↪accuracy:
```

```
0.7688
Epoch 62: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
 ↪4935 - accuracy:
0.7696 - val_loss: 0.6150 - val_accuracy: 0.6677
Epoch 63/125
45/46 [============================>.] - ETA: 0s - loss: 0.4809 -␣
 ↪accuracy:
0.7590
Epoch 63: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
 ↪4806 - accuracy:
0.7592 - val_loss: 0.6006 - val_accuracy: 0.6581
Epoch 64/125
45/46 [============================>.] - ETA: 0s - loss: 0.4898 -␣
 ↪accuracy:
0.7583
Epoch 64: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
 ↪4894 - accuracy:
0.7585 - val_loss: 0.6103 - val_accuracy: 0.7097
Epoch 65/125
45/46 [============================>.] - ETA: 0s - loss: 0.4854 -␣
 ↪accuracy:
0.7583
Epoch 65: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
 ↪4861 - accuracy:
0.7564 - val_loss: 0.6008 - val_accuracy: 0.6742
Epoch 66/125
45/46 [============================>.] - ETA: 0s - loss: 0.4864 -␣
 ↪accuracy:
```

```
0.7535
Epoch 66: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4852 - accuracy:
0.7543 - val_loss: 0.6169 - val_accuracy: 0.6581
Epoch 67/125
45/46 [============================>.] - ETA: 0s - loss: 0.4785 -␣
↪accuracy:
0.7597
Epoch 67: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4793 - accuracy:
0.7592 - val_loss: 0.6032 - val_accuracy: 0.6839
Epoch 68/125
45/46 [============================>.] - ETA: 0s - loss: 0.4846 -␣
↪accuracy:
0.7625
Epoch 68: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4835 - accuracy:
0.7633 - val_loss: 0.6061 - val_accuracy: 0.6613
Epoch 69/125
45/46 [============================>.] - ETA: 0s - loss: 0.4823 -␣
↪accuracy:
0.7611
Epoch 69: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4838 - accuracy:
0.7599 - val_loss: 0.6215 - val_accuracy: 0.6645
Epoch 70/125
45/46 [============================>.] - ETA: 0s - loss: 0.4937 -␣
↪accuracy:
```

```
0.7632
Epoch 70: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4926 - accuracy:
0.7640 - val_loss: 0.6176 - val_accuracy: 0.6645
Epoch 71/125
45/46 [============================>.] - ETA: 0s - loss: 0.4797 -␣
↪accuracy:
0.7674
Epoch 71: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4784 - accuracy:
0.7682 - val_loss: 0.6071 - val_accuracy: 0.6452
Epoch 72/125
45/46 [============================>.] - ETA: 0s - loss: 0.4821 -␣
↪accuracy:
0.7611
Epoch 72: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4825 - accuracy:
0.7612 - val_loss: 0.6231 - val_accuracy: 0.6677
Epoch 73/125
44/46 [===========================>..] - ETA: 0s - loss: 0.5005 -␣
↪accuracy:
0.7521
Epoch 73: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4993 - accuracy:
0.7522 - val_loss: 0.6103 - val_accuracy: 0.7065
Epoch 74/125
44/46 [===========================>..] - ETA: 0s - loss: 0.4845 -␣
↪accuracy:
```

```
0.7578
Epoch 74: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4830 - accuracy:
0.7592 - val_loss: 0.6121 - val_accuracy: 0.6581
Epoch 75/125
45/46 [============================>.] - ETA: 0s - loss: 0.4818 -␣
↪accuracy:
0.7618
Epoch 75: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4819 - accuracy:
0.7612 - val_loss: 0.6092 - val_accuracy: 0.6613
Epoch 76/125
45/46 [============================>.] - ETA: 0s - loss: 0.4746 -␣
↪accuracy:
0.7688
Epoch 76: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4758 - accuracy:
0.7682 - val_loss: 0.6123 - val_accuracy: 0.6613
Epoch 77/125
44/46 [===========================>..] - ETA: 0s - loss: 0.4800 -␣
↪accuracy:
0.7635
Epoch 77: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4805 - accuracy:
0.7640 - val_loss: 0.6177 - val_accuracy: 0.6548
Epoch 78/125
45/46 [============================>.] - ETA: 0s - loss: 0.4828 -␣
↪accuracy:
```

```
0.7556
Epoch 78: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4826 - accuracy:
0.7557 - val_loss: 0.6115 - val_accuracy: 0.6516
Epoch 79/125
45/46 [============================>.] - ETA: 0s - loss: 0.4797 -␣
↪accuracy:
0.7632
Epoch 79: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4811 - accuracy:
0.7619 - val_loss: 0.6219 - val_accuracy: 0.7226
Epoch 80/125
45/46 [============================>.] - ETA: 0s - loss: 0.4864 -␣
↪accuracy:
0.7583
Epoch 80: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4867 - accuracy:
0.7585 - val_loss: 0.6454 - val_accuracy: 0.6677
Epoch 81/125
45/46 [============================>.] - ETA: 0s - loss: 0.4741 -␣
↪accuracy:
0.7750
Epoch 81: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4736 - accuracy:
0.7751 - val_loss: 0.6081 - val_accuracy: 0.6871
Epoch 82/125
45/46 [============================>.] - ETA: 0s - loss: 0.4723 -␣
↪accuracy:
```

```
0.7708
Epoch 82: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4745 - accuracy:
0.7696 - val_loss: 0.6151 - val_accuracy: 0.7065
Epoch 83/125
45/46 [============================>.] - ETA: 0s - loss: 0.4770 -␣
↪accuracy:
0.7625
Epoch 83: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4768 - accuracy:
0.7626 - val_loss: 0.6178 - val_accuracy: 0.6581
Epoch 84/125
45/46 [============================>.] - ETA: 0s - loss: 0.4774 -␣
↪accuracy:
0.7653
Epoch 84: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4782 - accuracy:
0.7647 - val_loss: 0.6130 - val_accuracy: 0.6613
Epoch 85/125
44/46 [===========================>..] - ETA: 0s - loss: 0.4801 -␣
↪accuracy:
0.7614
Epoch 85: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4756 - accuracy:
0.7640 - val_loss: 0.6130 - val_accuracy: 0.7032
Epoch 86/125
45/46 [============================>.] - ETA: 0s - loss: 0.4770 -␣
↪accuracy:
```

```
0.7708
Epoch 86: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4777 - accuracy:
0.7709 - val_loss: 0.6422 - val_accuracy: 0.6742
Epoch 87/125
45/46 [============================>.] - ETA: 0s - loss: 0.4808 -␣
↪accuracy:
0.7674
Epoch 87: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4815 - accuracy:
0.7675 - val_loss: 0.6108 - val_accuracy: 0.6677
Epoch 88/125
45/46 [============================>.] - ETA: 0s - loss: 0.4721 -␣
↪accuracy:
0.7653
Epoch 88: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4728 - accuracy:
0.7654 - val_loss: 0.6253 - val_accuracy: 0.6645
Epoch 89/125
45/46 [============================>.] - ETA: 0s - loss: 0.4788 -␣
↪accuracy:
0.7611
Epoch 89: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4785 - accuracy:
0.7612 - val_loss: 0.6048 - val_accuracy: 0.6839
Epoch 90/125
45/46 [============================>.] - ETA: 0s - loss: 0.4757 -␣
↪accuracy:
```

```
0.7660
Epoch 90: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4753 - accuracy:
0.7668 - val_loss: 0.6221 - val_accuracy: 0.7194
Epoch 91/125
45/46 [============================>.] - ETA: 0s - loss: 0.4762 -␣
↪accuracy:
0.7722
Epoch 91: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4771 - accuracy:
0.7716 - val_loss: 0.6175 - val_accuracy: 0.6613
Epoch 92/125
45/46 [============================>.] - ETA: 0s - loss: 0.4800 -␣
↪accuracy:
0.7750
Epoch 92: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4805 - accuracy:
0.7744 - val_loss: 0.6313 - val_accuracy: 0.6613
Epoch 93/125
45/46 [============================>.] - ETA: 0s - loss: 0.4731 -␣
↪accuracy:
0.7688
Epoch 93: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4752 - accuracy:
0.7675 - val_loss: 0.6095 - val_accuracy: 0.6968
Epoch 94/125
45/46 [============================>.] - ETA: 0s - loss: 0.4764 -␣
↪accuracy:
```

```
0.7646
Epoch 94: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4764 - accuracy:
0.7647 - val_loss: 0.6094 - val_accuracy: 0.6645
Epoch 95/125
45/46 [============================>.] - ETA: 0s - loss: 0.4666 -␣
↪accuracy:
0.7736
Epoch 95: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4669 - accuracy:
0.7737 - val_loss: 0.6066 - val_accuracy: 0.6806
Epoch 96/125
45/46 [============================>.] - ETA: 0s - loss: 0.4651 -␣
↪accuracy:
0.7743
Epoch 96: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4648 - accuracy:
0.7751 - val_loss: 0.6561 - val_accuracy: 0.6742
Epoch 97/125
45/46 [============================>.] - ETA: 0s - loss: 0.4706 -␣
↪accuracy:
0.7722
Epoch 97: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4713 - accuracy:
0.7716 - val_loss: 0.6125 - val_accuracy: 0.6677
Epoch 98/125
45/46 [============================>.] - ETA: 0s - loss: 0.4722 -␣
↪accuracy:
```

```
0.7660
Epoch 98: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4720 - accuracy:
0.7661 - val_loss: 0.6133 - val_accuracy: 0.6677
Epoch 99/125
45/46 [============================>.] - ETA: 0s - loss: 0.4662 -␣
↪accuracy:
0.7736
Epoch 99: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4654 - accuracy:
0.7744 - val_loss: 0.6315 - val_accuracy: 0.7194
Epoch 100/125
45/46 [============================>.] - ETA: 0s - loss: 0.4687 -␣
↪accuracy:
0.7701
Epoch 100: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4701 - accuracy:
0.7696 - val_loss: 0.6132 - val_accuracy: 0.6710
Epoch 101/125
45/46 [============================>.] - ETA: 0s - loss: 0.4680 -␣
↪accuracy:
0.7653
Epoch 101: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4696 - accuracy:
0.7640 - val_loss: 0.6061 - val_accuracy: 0.6871
Epoch 102/125
45/46 [============================>.] - ETA: 0s - loss: 0.4624 -␣
↪accuracy:
```

```
0.7750
Epoch 102: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4622 - accuracy:
0.7751 - val_loss: 0.6092 - val_accuracy: 0.6774
Epoch 103/125
45/46 [============================>.] - ETA: 0s - loss: 0.4650 -␣
↪accuracy:
0.7806
Epoch 103: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 16ms/step - loss: 0.
↪4657 - accuracy:
0.7799 - val_loss: 0.6185 - val_accuracy: 0.6645
Epoch 104/125
45/46 [============================>.] - ETA: 0s - loss: 0.4640 -␣
↪accuracy:
0.7715
Epoch 104: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4640 - accuracy:
0.7716 - val_loss: 0.6123 - val_accuracy: 0.6774
Epoch 105/125
45/46 [============================>.] - ETA: 0s - loss: 0.4685 -␣
↪accuracy:
0.7750
Epoch 105: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4693 - accuracy:
0.7751 - val_loss: 0.6156 - val_accuracy: 0.6645
Epoch 106/125
45/46 [============================>.] - ETA: 0s - loss: 0.4650 -␣
↪accuracy:
```

```
0.7771
Epoch 106: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4648 - accuracy:
0.7772 - val_loss: 0.6104 - val_accuracy: 0.6806
Epoch 107/125
45/46 [============================>.] - ETA: 0s - loss: 0.4628 -␣
↪accuracy:
0.7764
Epoch 107: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4626 - accuracy:
0.7765 - val_loss: 0.6311 - val_accuracy: 0.6613
Epoch 108/125
45/46 [============================>.] - ETA: 0s - loss: 0.4680 -␣
↪accuracy:
0.7701
Epoch 108: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4676 - accuracy:
0.7702 - val_loss: 0.6379 - val_accuracy: 0.6613
Epoch 109/125
45/46 [============================>.] - ETA: 0s - loss: 0.4659 -␣
↪accuracy:
0.7778
Epoch 109: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4646 - accuracy:
0.7785 - val_loss: 0.6188 - val_accuracy: 0.7097
Epoch 110/125
45/46 [============================>.] - ETA: 0s - loss: 0.4692 -␣
↪accuracy:
```

```
0.7764
Epoch 110: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4696 - accuracy:
0.7758 - val_loss: 0.7340 - val_accuracy: 0.6516
Epoch 111/125
45/46 [============================>.] - ETA: 0s - loss: 0.4948 -␣
↪accuracy:
0.7458
Epoch 111: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4949 - accuracy:
0.7460 - val_loss: 0.6146 - val_accuracy: 0.6774
Epoch 112/125
45/46 [============================>.] - ETA: 0s - loss: 0.4587 -␣
↪accuracy:
0.7833
Epoch 112: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4583 - accuracy:
0.7834 - val_loss: 0.6333 - val_accuracy: 0.7161
Epoch 113/125
45/46 [============================>.] - ETA: 0s - loss: 0.4731 -␣
↪accuracy:
0.7674
Epoch 113: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4723 - accuracy:
0.7682 - val_loss: 0.6133 - val_accuracy: 0.6903
Epoch 114/125
45/46 [============================>.] - ETA: 0s - loss: 0.4721 -␣
↪accuracy:
```

```
0.7701
Epoch 114: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4719 - accuracy:
0.7709 - val_loss: 0.6552 - val_accuracy: 0.6774
Epoch 115/125
45/46 [============================>.] - ETA: 0s - loss: 0.4672 -␣
↪accuracy:
0.7681
Epoch 115: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4681 - accuracy:
0.7675 - val_loss: 0.6259 - val_accuracy: 0.7065
Epoch 116/125
45/46 [============================>.] - ETA: 0s - loss: 0.4628 -␣
↪accuracy:
0.7799
Epoch 116: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4619 - accuracy:
0.7806 - val_loss: 0.6089 - val_accuracy: 0.6968
Epoch 117/125
45/46 [============================>.] - ETA: 0s - loss: 0.4603 -␣
↪accuracy:
0.7764
Epoch 117: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4595 - accuracy:
0.7772 - val_loss: 0.6116 - val_accuracy: 0.6935
Epoch 118/125
45/46 [============================>.] - ETA: 0s - loss: 0.4600 -␣
↪accuracy:
```

```
0.7771
Epoch 118: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4611 - accuracy:
0.7765 - val_loss: 0.6194 - val_accuracy: 0.6742
Epoch 119/125
45/46 [============================>.] - ETA: 0s - loss: 0.4586 -␣
↪accuracy:
0.7757
Epoch 119: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4574 - accuracy:
0.7765 - val_loss: 0.6300 - val_accuracy: 0.6581
Epoch 120/125
45/46 [============================>.] - ETA: 0s - loss: 0.4605 -␣
↪accuracy:
0.7729
Epoch 120: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4597 - accuracy:
0.7737 - val_loss: 0.6150 - val_accuracy: 0.7032
Epoch 121/125
45/46 [============================>.] - ETA: 0s - loss: 0.4637 -␣
↪accuracy:
0.7771
Epoch 121: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4642 - accuracy:
0.7772 - val_loss: 0.6183 - val_accuracy: 0.7065
Epoch 122/125
45/46 [============================>.] - ETA: 0s - loss: 0.4715 -␣
↪accuracy:
```

```
0.7764
Epoch 122: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4709 - accuracy:
0.7765 - val_loss: 0.6216 - val_accuracy: 0.6774
Epoch 123/125
45/46 [============================>.] - ETA: 0s - loss: 0.4527 -␣
↪accuracy:
0.7799
Epoch 123: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4543 - accuracy:
0.7792 - val_loss: 0.6124 - val_accuracy: 0.6935
Epoch 124/125
45/46 [============================>.] - ETA: 0s - loss: 0.4652 -␣
↪accuracy:
0.7701
Epoch 124: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4661 - accuracy:
0.7689 - val_loss: 0.6124 - val_accuracy: 0.7032
Epoch 125/125
45/46 [============================>.] - ETA: 0s - loss: 0.4630 -␣
↪accuracy:
0.7722
Epoch 125: val_accuracy did not improve from 0.72258
46/46 [==============================] - 1s 17ms/step - loss: 0.
↪4642 - accuracy:
0.7709 - val_loss: 0.6087 - val_accuracy: 0.6903
Elapsed time: 0:1:48.9
```

[ ]:
```
history = own_model.history.history
```

[ ]:
```
plot_metrics(history)
```

max size=0.90.9output$_1$27$_0$.png

max size=0.90.9output$_1$27$_1$.png

model gives best validation accuracy on 43rd epoch

[ ]:
```
besttry_with_own_model_model = load_model(filepath='/content/
↪gdrive/MyDrive/MA981-7-FY/Dataset/try_with_own_model/043-0.72')
```

```
WARNING:tensorflow:SavedModel saved prior to TF 2.5 detected when␣
↪loading Keras
model. Please ensure that you are saving the model with model.
↪save() or
tf.keras.models.save_model(), *NOT* tf.saved_model.save(). To␣
↪confirm, there
should be a file named "keras_metadata.pb" in the SavedModel␣
↪directory.
```

[ ]:
```
loss, acc = besttry_with_own_model_model.evaluate(x=X_test,␣
↪y=Y_test)
```

```
10/10 [==============================] - 0s 12ms/step - loss: 0.
↪4931 - accuracy:
0.7839
```

[ ]:
```
print (f"Test Loss = {loss}")
print (f"Test Accuracy = {acc}")
```

```
Test Loss = 0.49313226342201233
Test Accuracy = 0.7838709950447083
```

[ ]:

```
Y_test_prob_o = besttry_with_own_model_model.predict(X_test)

10/10 [==============================] - 0s 9ms/step
```

calculating F1 score

[ ]:
```
f1score_val = compute_f1_score(Y_test, Y_test_prob_o)
print(f"F1 score: {f1score_val}")

F1 score: 0.8101983002832861
```

Result time:

[ ]:
```
print("Training Data:")
data_percentage(Y_train)
print("Validation Data:")
data_percentage(Y_val)
print("Testing Data:")
data_percentage(Y_test)

Training Data:
Number of examples: 1445
Percentage of positive examples: 51.4878892733564%, number of pos␣
↪examples: 744
Percentage of negative examples: 48.5121107266436%, number of neg␣
↪examples: 701
Validation Data:
Number of examples: 310
Percentage of positive examples: 54.83870967741935%, number of pos␣
↪examples: 170
Percentage of negative examples: 45.16129032258065%, number of neg␣
↪examples: 140
Testing Data:
Number of examples: 310
```

```
Percentage of positive examples: 55.16129032258065%, number of pos␣
 ↪examples: 171
Percentage of negative examples: 44.83870967741935%, number of neg␣
 ↪examples: 139
```

6.19 Using CNN

I used CNN to conduct image classification on the brain tumour dataset in this jupyter notebook. A neural network trained on this short dataset won't actually provide us with a good result. As a result, I'm going to train the model using the Transfer Learning approach to provide incredibly accurate results.

[ ]:
```
import os
import cv2
import numpy as np
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten,␣
 ↪Dense
from tensorflow.keras.utils import to_categorical
from keras.preprocessing.image import ImageDataGenerator

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import tensorflow as tf
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'

from keras.models import Sequential
from keras.layers import Dense, Flatten, Conv2D, MaxPooling2D
from keras.layers import Input, Activation, LeakyReLU, Dropout
from keras.losses import BinaryCrossentropy
```

```
from tensorflow.keras.optimizers import Adam



from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
# Load and preprocess the data
path = '/content/gdrive/MyDrive/MA981-7-FY/Dataset/Training/'
classes = {'no_tumor': 0, 'pituitary_tumor': 1}

X = []
Y = []
for cls in classes:
    pth = path + cls
    for j in os.listdir(pth):
        img = cv2.imread(pth + '/' + j, 0)
        img = cv2.resize(img, (200, 200))
        X.append(img)
        Y.append(classes[cls])

X = np.array(X)
Y = np.array(Y)

# Normalize the pixel values
X = X / 255.0

# Convert labels to one-hot encoding
Y = to_categorical(Y)

# Split the data into training and testing sets
xtrain, xtest, ytrain, ytest = train_test_split(X, Y,␣
↪random_state=10, test_size=0.20)
```

```
# Define the CNN model
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(200,␣
↪200, 1)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dense(2, activation='softmax'))

# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy',␣
↪metrics=['accuracy'])

# Train the model
model.fit(xtrain, ytrain, batch_size=32, epochs=10,␣
↪validation_data=(xtest, ytest))

# Evaluate the model
score = model.evaluate(xtest, ytest, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])

Epoch 1/10
31/31 [==============================] - 6s 66ms/step - loss: 0.
↪4475 - accuracy:
0.7922 - val_loss: 0.2860 - val_accuracy: 0.8571
Epoch 2/10
```

```
31/31 [==============================] - 1s 36ms/step - loss: 0.
↪1773 - accuracy:
0.9324 - val_loss: 0.0873 - val_accuracy: 0.9633
Epoch 3/10
31/31 [==============================] - 1s 36ms/step - loss: 0.
↪0711 - accuracy:
0.9795 - val_loss: 0.0457 - val_accuracy: 0.9796
Epoch 4/10
31/31 [==============================] - 1s 36ms/step - loss: 0.
↪0239 - accuracy:
0.9949 - val_loss: 0.0242 - val_accuracy: 0.9878
Epoch 5/10
31/31 [==============================] - 1s 37ms/step - loss: 0.
↪0099 - accuracy:
0.9949 - val_loss: 0.0812 - val_accuracy: 0.9714
Epoch 6/10
31/31 [==============================] - 1s 36ms/step - loss: 0.
↪0413 - accuracy:
0.9846 - val_loss: 0.0597 - val_accuracy: 0.9837
Epoch 7/10
31/31 [==============================] - 1s 36ms/step - loss: 0.
↪0315 - accuracy:
0.9908 - val_loss: 0.0322 - val_accuracy: 0.9837
Epoch 8/10
31/31 [==============================] - 1s 36ms/step - loss: 0.
↪0220 - accuracy:
0.9928 - val_loss: 0.0116 - val_accuracy: 0.9918
Epoch 9/10
31/31 [==============================] - 1s 36ms/step - loss: 0.
↪0202 - accuracy:
0.9898 - val_loss: 0.0296 - val_accuracy: 0.9837
```

```
Epoch 10/10
31/31 [==============================] - 1s 36ms/step - loss: 0.
↪0130 - accuracy:
0.9969 - val_loss: 0.1580 - val_accuracy: 0.9510
Test loss: 0.15804548561573029
Test accuracy: 0.9510204195976257
```

[ ]:
```
# Saving the model for future
model.save('cnn-model.h5')
```

[ ]:
```
trdata = ImageDataGenerator()
traindata = trdata.flow_from_directory(directory="/content/gdrive/
↪MyDrive/MA981-7-FY/Dataset/Training/",target_size=(224,224))
tsdata = ImageDataGenerator()
testdata = tsdata.flow_from_directory(directory="/content/gdrive/
↪MyDrive/MA981-7-FY/Dataset/Testing", target_size=(224,224))
```

```
Found 2870 images belonging to 4 classes.
Found 394 images belonging to 4 classes.
```

[ ]:
```
model = Sequential()
model.
↪add(Conv2D(input_shape=(224,224,3),filters=64,kernel_size=(3,3),padding="
↪activation="relu"))
model.add(Conv2D(filters=64,kernel_size=(3,3),padding="same",␣
↪activation="relu"))
model.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))
model.add(Conv2D(filters=128, kernel_size=(3,3), padding="same",␣
↪activation="relu"))
model.add(Conv2D(filters=128, kernel_size=(3,3), padding="same",␣
↪activation="relu"))
model.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))
```

```
model.add(Conv2D(filters=256, kernel_size=(3,3), padding="same",␣
 ↪activation="relu"))
model.add(Conv2D(filters=256, kernel_size=(3,3), padding="same",␣
 ↪activation="relu"))
model.add(Conv2D(filters=256, kernel_size=(3,3), padding="same",␣
 ↪activation="relu"))
model.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same",␣
 ↪activation="relu"))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same",␣
 ↪activation="relu"))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same",␣
 ↪activation="relu"))
model.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same",␣
 ↪activation="relu"))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same",␣
 ↪activation="relu"))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same",␣
 ↪activation="relu"))
model.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))
```

[ ]:
```
model.add(Flatten())
model.add(Dense(units=4096,activation="relu"))
model.add(Dense(units=4096,activation="relu"))
model.add(Dense(units=2, activation="softmax"))
```

[ ]:
```
from tensorflow.keras.optimizers import Adam

opt = Adam(lr=0.001)
# model.compile(optimizer=opt, loss=keras.losses.
 ↪categorical_crossentropy, metrics=['accuracy'])
```

```
model.compile(optimizer=opt, loss='categorical_crossentropy',␣
↪metrics=['accuracy'])
```

```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use␣
↪`learning_rate`
or use the legacy optimizer, e.g.,tf.keras.optimizers.legacy.Adam.
```

[ ]:
```
model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_97 (Conv2D)          (None, 224, 224, 64)      1792

 conv2d_98 (Conv2D)          (None, 224, 224, 64)      36928

 max_pooling2d_7 (MaxPooling  (None, 112, 112, 64)     0
 2D)

 conv2d_99 (Conv2D)          (None, 112, 112, 128)     73856

 conv2d_100 (Conv2D)         (None, 112, 112, 128)     147584

 max_pooling2d_8 (MaxPooling  (None, 56, 56, 128)      0
 2D)

 conv2d_101 (Conv2D)         (None, 56, 56, 256)       295168

 conv2d_102 (Conv2D)         (None, 56, 56, 256)       590080

 conv2d_103 (Conv2D)         (None, 56, 56, 256)       590080
```

```
 max_pooling2d_9 (MaxPooling  (None, 28, 28, 256)      0
 2D)

 conv2d_104 (Conv2D)         (None, 28, 28, 512)       1180160

 conv2d_105 (Conv2D)         (None, 28, 28, 512)       2359808

 conv2d_106 (Conv2D)         (None, 28, 28, 512)       2359808

 max_pooling2d_10 (MaxPoolin  (None, 14, 14, 512)      0
 g2D)

 conv2d_107 (Conv2D)         (None, 14, 14, 512)       2359808

 conv2d_108 (Conv2D)         (None, 14, 14, 512)       2359808

 conv2d_109 (Conv2D)         (None, 14, 14, 512)       2359808

 max_pooling2d_11 (MaxPoolin  (None, 7, 7, 512)        0
 g2D)

 flatten_5 (Flatten)         (None, 25088)             0

 dense_6 (Dense)             (None, 4096)              102764544

 dense_7 (Dense)             (None, 4096)              16781312

 dense_8 (Dense)             (None, 2)                 8194

=================================================================
Total params: 134,268,738
Trainable params: 134,268,738
```

```
Non-trainable params: 0
_________________________________________________________________
```

[ ]:

```
from tensorflow.keras.models import load_model

# Load the trained CNN model
model = load_model('cnn-model.h5')  # Loading the cnn model that␣
 ↪we have trained

# Directory path of the images to be tested
testing_path = '/content/gdrive/MyDrive/MA981-7-FY/Dataset/Testing/
 ↪no_tumor/'

# Load and preprocess the images for testing
X_test = []
for j in os.listdir(testing_path):
    img = cv2.imread(testing_path + j, 0)
    img = cv2.resize(img, (200, 200))
    X_test.append(img)

X_test = np.array(X_test)
X_test = X_test / 255.0  # Normalize pixel values

# Reshape the input images to match the input shape of the CNN␣
 ↪model
X_test = X_test.reshape(X_test.shape[0], 200, 200, 1)

# Make predictions using the trained CNN model
predictions = model.predict(X_test)

# Convert the predicted probabilities to class labels
predicted_classes = np.argmax(predictions, axis=1)
```

```
# Print the predicted class labels
misclassified_count = 0
for i, pred_cls in enumerate(predicted_classes):
    if pred_cls == 0:
        print(f'Image {i+1}: no_tumor')
    else:
        print(f'Image {i+1}: pituitary_tumor (misclassified)')
        misclassified_count += 1
print(f'Total misclassified images: {misclassified_count}')
```

```
4/4 [==============================] - 0s 12ms/step
Image 1: no_tumor
Image 2: no_tumor
Image 3: no_tumor
Image 4: no_tumor
Image 5: no_tumor
Image 6: no_tumor
Image 7: no_tumor
Image 8: no_tumor
Image 9: no_tumor
Image 10: no_tumor
Image 11: no_tumor
Image 12: no_tumor
Image 13: no_tumor
Image 14: no_tumor
Image 15: no_tumor
Image 16: no_tumor
Image 17: no_tumor
Image 18: no_tumor
Image 19: no_tumor
Image 20: no_tumor
Image 21: no_tumor
```

```
Image 22: no_tumor
Image 23: no_tumor
Image 24: no_tumor
Image 25: no_tumor
Image 26: no_tumor
Image 27: no_tumor
Image 28: no_tumor
Image 29: no_tumor
Image 30: no_tumor
Image 31: no_tumor
Image 32: no_tumor
Image 33: no_tumor
Image 34: no_tumor
Image 35: no_tumor
Image 36: no_tumor
Image 37: no_tumor
Image 38: no_tumor
Image 39: no_tumor
Image 40: no_tumor
Image 41: no_tumor
Image 42: no_tumor
Image 43: no_tumor
Image 44: no_tumor
Image 45: no_tumor
Image 46: no_tumor
Image 47: no_tumor
Image 48: no_tumor
Image 49: no_tumor
Image 50: no_tumor
Image 51: no_tumor
Image 52: no_tumor
Image 53: no_tumor
```

```
Image 54: no_tumor
Image 55: no_tumor
Image 56: no_tumor
Image 57: no_tumor
Image 58: no_tumor
Image 59: no_tumor
Image 60: no_tumor
Image 61: no_tumor
Image 62: no_tumor
Image 63: no_tumor
Image 64: no_tumor
Image 65: no_tumor
Image 66: no_tumor
Image 67: no_tumor
Image 68: no_tumor
Image 69: no_tumor
Image 70: no_tumor
Image 71: no_tumor
Image 72: no_tumor
Image 73: no_tumor
Image 74: no_tumor
Image 75: no_tumor
Image 76: no_tumor
Image 77: no_tumor
Image 78: no_tumor
Image 79: no_tumor
Image 80: no_tumor
Image 81: no_tumor
Image 82: no_tumor
Image 83: no_tumor
Image 84: no_tumor
Image 85: no_tumor
```

```
Image 86: no_tumor
Image 87: no_tumor
Image 88: no_tumor
Image 89: no_tumor
Image 90: no_tumor
Image 91: no_tumor
Image 92: no_tumor
Image 93: no_tumor
Image 94: no_tumor
Image 95: no_tumor
Image 96: no_tumor
Image 97: no_tumor
Image 98: no_tumor
Image 99: no_tumor
Image 100: no_tumor
Image 101: no_tumor
Image 102: no_tumor
Image 103: no_tumor
Image 104: no_tumor
Image 105: no_tumor
Total misclassified images: 0
```

[ ]:

```
import os

dataset_path = '/content/gdrive/MyDrive/MA981-7-FY/Dataset/
 ↪brain_tumor_dataset/'
folders = os.listdir(dataset_path)
print(folders)
```

```
['yes', 'no']
```

[ ]:

```
subdirs = os.listdir(dataset_path)[:2]
for subdir in subdirs:
    print(f"{subdir} contains {len(os.listdir(dataset_path+'/
 ↪'+subdir))} images")
```

```
yes contains 155 images
no contains 98 images
```

[ ]:

```
def load_images(folder):

    imgs = []
    target = 0
    labels = []
    for i in os.listdir(folder):
        subdir = os.path.join(folder, i)
        for j in os.listdir(subdir):
            img_dir = os.path.join(subdir,j)
            try:
                img = cv2.imread(img_dir)
                img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
                img = cv2.resize(img, (128,128))
                imgs.append(img)
                labels.append(target)
            except:
                continue
        target += 1

    imgs = np.array(imgs)
    labels = np.array(labels)

    return imgs, labels
```

[ ]:

```
data, labels = load_images(dataset_path)
data.shape, labels.shape
```

[breakable, size=fbox, boxrule=.5pt, pad at break*=1mm, opacityfill=0]
```
((253, 128, 128), (253,))
```

[ ]:
```
def plot_images(start, end):
    plt.figure(figsize=(22,8))
    for i in range(10):
        axs = plt.subplot(2,5, i+1)
        idx = np.random.randint(start, end)
        plt.imshow(data[idx], cmap='gray')
        plt.axis('on')
        axs.set_xticklabels([])
        axs.set_yticklabels([])
        plt.subplots_adjust(wspace=None, hspace=None)
```

[ ]:
```
plot_images(98,252)
```

max size=0.90.9output_151_0.png

[ ]:
```
plot_images(0,97)
```

max size=0.90.9output_152_0.png

[ ]:
```
norm_data = data / 255.
norm_data = np.expand_dims(norm_data, axis=3)
norm_data.shape, norm_data[0]
```

[breakable, size=fbox, boxrule=.5pt, pad at break*=1mm, opacityfill=0]
```
((253, 128, 128, 1),
 array([[[0.],
         [0.],
```

```
        [0.],
        ...,
        [0.],
        [0.],
        [0.]],

       [[0.],
        [0.],
        [0.],
        ...,
        [0.],
        [0.],
        [0.]],

       [[0.],
        [0.],
        [0.],
        ...,
        [0.],
        [0.],
        [0.]],

       ...,

       [[0.],
        [0.],
        [0.],
        ...,
        [0.],
        [0.],
        [0.]],
```

```
        [[0.],
         [0.],
         [0.],
         ...,
         [0.],
         [0.],
         [0.]],

        [[0.],
         [0.],
         [0.],
         ...,
         [0.],
         [0.],
         [0.]]]))
```

```
[ ]:
SEED = 40
```

```
[ ]:
tf.random.set_seed(SEED)

model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(filters=64,
                           kernel_size=3,
                           activation='relu',
                           input_shape=(128,128,1)),
    tf.keras.layers.Conv2D(32,3,activation='relu'),
    tf.keras.layers.MaxPool2D(pool_size=2,
                              padding='valid'),
    tf.keras.layers.Conv2D(32,3,activation='relu'),
    tf.keras.layers.Conv2D(16,3,activation='relu'),
    tf.keras.layers.MaxPool2D(2),
    tf.keras.layers.Flatten(),
```

```
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

[ ]:
```
model.compile(loss = tf.keras.losses.BinaryCrossentropy(),
              optimizer = tf.keras.optimizers.Adam(),
                metrics = ["accuracy"])
```

[ ]:
```
history = model.fit(norm_data, labels, epochs = 10,␣
↪validation_split = 0.20)
```

```
Epoch 1/10
7/7 [==============================] - 4s 186ms/step - loss: 0.
↪6345 - accuracy:
0.7673 - val_loss: 1.1528 - val_accuracy: 0.0000e+00
Epoch 2/10
7/7 [==============================] - 0s 33ms/step - loss: 0.5702␣
↪- accuracy:
0.7673 - val_loss: 0.8467 - val_accuracy: 0.0000e+00
Epoch 3/10
7/7 [==============================] - 0s 32ms/step - loss: 0.5295␣
↪- accuracy:
0.7673 - val_loss: 1.5149 - val_accuracy: 0.0000e+00
Epoch 4/10
7/7 [==============================] - 0s 33ms/step - loss: 0.4611␣
↪- accuracy:
0.7673 - val_loss: 1.2120 - val_accuracy: 0.0000e+00
Epoch 5/10
7/7 [==============================] - 0s 32ms/step - loss: 0.4473␣
↪- accuracy:
0.8069 - val_loss: 1.1501 - val_accuracy: 0.2745
Epoch 6/10
7/7 [==============================] - 0s 33ms/step - loss: 0.4220␣
↪- accuracy:
```

```
0.8218 - val_loss: 1.1471 - val_accuracy: 0.5294
Epoch 7/10
7/7 [==============================] - 0s 32ms/step - loss: 0.4110␣
 ↪- accuracy:
0.8119 - val_loss: 1.1678 - val_accuracy: 0.5294
Epoch 8/10
7/7 [==============================] - 0s 32ms/step - loss: 0.3831␣
 ↪- accuracy:
0.8366 - val_loss: 0.9619 - val_accuracy: 0.6078
Epoch 9/10
7/7 [==============================] - 0s 33ms/step - loss: 0.4250␣
 ↪- accuracy:
0.8267 - val_loss: 0.8413 - val_accuracy: 0.6078
Epoch 10/10
7/7 [==============================] - 0s 32ms/step - loss: 0.3517␣
 ↪- accuracy:
0.8366 - val_loss: 1.4964 - val_accuracy: 0.4706
```

[ ]:
```
result = model.evaluate(norm_data, labels, verbose=0)
print(f"Accuracy on Evaluation: {result[1]*100:.2f}%\nLoss:␣
 ↪{result[0]:.4f}")
```

```
Accuracy on Evaluation: 77.87%
Loss: 0.5947
```

[ ]:
```
np.random.seed(SEED)
indxs = np.random.randint(0, 252, 20)
```

[ ]:
```
y_pred_prob = model.predict(norm_data[indxs])
y_pred = np.array([1 if prob>0.5 else 0 for prob in y_pred_prob])

y_true = labels[indxs]
y_pred.shape, y_true.shape
```

```
1/1 [==============================] - 0s 390ms/step
```

[breakable, size=fbox, boxrule=.5pt, pad at break*=1mm, opacityfill=0]
```
((20,), (20,))
```

[ ]:
```
y_pred
```

[breakable, size=fbox, boxrule=.5pt, pad at break*=1mm, opacityfill=0]
```
array([0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1])
```

[ ]:
```
plt.figure(figsize = (8,8))
cm = confusion_matrix(y_true, y_pred)
sns.heatmap(cm, cmap = "Greens", annot = True, fmt = ".2g", cbar =␣
↪False)
plt.show()
```

max size=0.90.9output$_1$62$_0$.png

[ ]:
```
print(classification_report(y_true, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.73      1.00      0.85        11
           1       1.00      0.56      0.71         9

    accuracy                           0.80        20
   macro avg       0.87      0.78      0.78        20
weighted avg       0.85      0.80      0.79        20
```

[ ]:
```
model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_110 (Conv2D)         (None, 126, 126, 64)      640

 conv2d_111 (Conv2D)         (None, 124, 124, 32)      18464

 max_pooling2d_12 (MaxPoolin  (None, 62, 62, 32)       0
 g2D)

 conv2d_112 (Conv2D)         (None, 60, 60, 32)        9248

 conv2d_113 (Conv2D)         (None, 58, 58, 16)        4624

 max_pooling2d_13 (MaxPoolin  (None, 29, 29, 16)       0
 g2D)

 flatten_6 (Flatten)         (None, 13456)             0

 dense_9 (Dense)             (None, 1)                 13457

=================================================================
Total params: 46,433
Trainable params: 46,433
Non-trainable params: 0
_________________________________________________________________
```

```
histdf = pd.DataFrame(history.history)

plt.figure(figsize=(12,8))
plt.plot(histdf['accuracy'], label='Training Acc')
plt.plot(histdf['val_accuracy'], label='Validation Acc')
```

```
plt.plot(histdf['loss'], label='Loss')
plt.legend()
plt.show()
```

max size=0.90.9output$_1$65$_0$.png

So, out of 105 images, our model is giving 2 errors so clearly our practical error rate from this seems to be 2/105 *100 = 1.9 %

Result: -

From the above, we can see that we are getting accuracy of 0.963, 0.959 and 0.979 by svc, logistic regression, transfer learning and cnn respectively. However, the Convolutional Neural Network (CNN) model outperformed both with a test accuracy of 0.979, indicating its superior performance in accurately predicting brain tumor classification. These findings highlight the effectiveness of the CNN model in identifying brain tumors and its potential to be a valuable tool for medical professionals in improving diagnosis and treatment strategies for patients with brain tumors.

Bibliography

[1] GURUNATHAN, Akila; KRISHNAN, Batri. Detection and diagnosis of brain tumors using deep learning convolutional neural networks International Journal of Imaging Systems and Technology (2021).

[2] HS, Santhosh Kumar; KARIBASAPPA, K. An approach for brain tumour detection based on dual-tree complex Gabor wavelet transform and neural network using Hadoop big data analysis Multimedia Tools and Applications (2022).

[3] KIRAN, P.; PARAMES HACHARI, B. D. Resource Optimized Selective Image Encryption of Medical Images Using Multiple Chaotic Systems Microprocessors and Microsystems (2022).

[4] SHAN, Chengxiang; LI, Qiang; WANG, Ching-Hsin. Brain Tumor Segmentation using Automatic 3D Multi-channel Feature Selection Convolutional Neural Network Journal of Imaging Science and Technology (2022).

[5] ILHAN, Ahmet; SEKEROGLU, Boran; ABIYEV, Rahib Brain tumor segmentation in MRI images using nonparametric localization and enhancement methods with U-net International Journal of Computer Assisted Radiology and Surgery (2022).

[6] SALVI, Massimo, et al. The impact of pre-and post-image processing techniques on deep learning frameworks: A comprehensive review for digital pathology image analysis Computers in Biology and Medicine (2021).

[7] CHEN, Hanting, et al. Pre-trained image processing transformer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2021).

[8] SCHAPIRO, Denis, et al. MCMICRO A scalable, modular image-processing pipeline for multiplexed tissue imaging Nature methods (2022).

[9] XIONG, Yihui; ZUO, Renguang Robust feature extraction for geochemical anomaly recognition using a stacked convolutional denoising autoencoder Mathematical Geosciences (2022).

[10] PILLAI, Abhiram, et al. Breast Cancer Detection in Mammograms Using Deep Learning. In: Applied Information Processing Systems (2022).

[11] SMITH, Melvyn L.; SMITH, Lyndon N.; HANSEN, Mark F The quiet revolution in machine vision-a state-of-the-art survey paper, including historical review, perspectives, and future directions Computers in Industry (2021). 44

[12] QI, Ping, et al. Integrating functional data analysis with case-based reasoning for hypertension prognosis and diagnosis based on real-world electronic health records BMC Medical Informatics and Decision Making (2022).

[13] SAIBENE, Aurora; ASSALE, Michela; GILTRI, Marta. Expert systems: Definitions, advantages and issues in medical field applications. Expert Systems with Applications (2021).

[14] ALHARAN, Abbas FH; ALI, Nabeel Salih; ALGELAL, Zahraa M. AN ADAPTIVE NEURO-FUZZY INFERENCE SYSTEM AND PRINCIPAL COMPONENT ANALYSIS: A HYBRIDIZED METHOD FOR VIRAL HEPATITIS DIAGNOSIS SYSTEM Colloquium-journal.

[15] MCMONNIES, Charles W. Why the symptoms and objective signs of dry eye disease may not correlate Journal of Optometry (2021).

[16] DHARMAWAN, Dhimas Arief. Assessing fairness in performance evaluation of publicly available retinal blood vessel segmentation algorithms Journal of Medical Engineering Technology (2021).

[17] ABDULSAHIB, Aws A., et al. Comprehensive review of retinal blood vessel segmentation and classification techniques: intelligent solutions for green computing in medical images, current challenges, open issues, and knowledge gaps in fundus medical images Network Modeling Analysis in Health Informatics and Bioinformatic (2021).

[18] FIGUEROA, Anna G., et al. Levodopa positively affects neovascular age-related macular degeneration The American journal of medicine (2021).

[19] VIVEK, Pratyush, et al. CNN Models and Machine Learning Classifiers for Analysis of Goiter Disease. In: 2022 IEEE International Students Conference on Electrical, Electronics and Computer Science (SCEECS). IEEE (2022).

[20] LAHMOOD HAMEED, Fakhri; DAKKAK, Omar Brain Tumor Detection and Classification Using Convolutional Neural Network (CNN) International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA). IEEE (2022).

[21] REZAIE, Tayebeh, et al. Adult-onset primary open-angle glaucoma caused by mutations in optineurin (2002).

[22] HEINSCH, Milena, et al. Supporting friends and family of adults with a primary brain tumour: A systematic review Health Social Care in the Community (2022).

[23] HUANG, Zheng, et al. AMF-Net: An adaptive multisequence fusing neural network for multi-modality brain tumor diagnosis Biomedical Signal Processing and Control (2022). 45

[24] Ramdas Vankdothu, Mohd Abdul Hameed, Husnah Fatima, A Brain Tumor Identification and Classification Using Deep Learning based on CNN-LSTM Method Computers and Electrical Engineering (2022).

[25] S K Rajeev, M. Pallikonda Rajasekaran, G. Vishnuvarthanan, T. Arunprasath, A biologically-inspired hybrid deep learning approach for brain tumor classification from magnetic resonance imaging using improved gabor wavelet transform and Elmann-BiLSTM network Biomedical Signal Processing and Control, (2022).

[26] R. Sindhiya Devi, B. Perumal, M. Pallikonda Rajasekaran, A hybrid deep learning based brain tumor classification and segmentation by stationary wavelet packet transform and adaptive kernel fuzzy c means clustering Advances in Engineering Software, (2022).

[27] LEI, Xiaoliang, et al. Brain tumor segmentation in MR images using a sparse constrained level set algorithm Expert Systems with Applications, (2022).

[28] SHINY, K. V., et al. Study and Analysis of Various Automatic Brain Tumour Segmentation and Classification: A Challenging Overview Turkish Journal of Computer and Mathematics Education (TURCOMAT) (2022).

[29] UGALE, Vivek Dhruv; PAWAR, Swati S.; PAWAR, Sheetal. Brain Tumour Detection using Image Processing. In: 2022 IEEE 11th International Conference on Communication Systems and Network Technologies (CSNT). IEEE (2022)

[30] PURE, M. Tech Scholar Nayan; TIWARI, Ashish. MRI Image Segmentation and Classification Using KFCM and Convolution Neural Networks (2021).

[31] MZOUGHI, Hiba, et al. Towards a computer aided diagnosis (CAD) for brain MRI glioblastomas tumor exploration based on a deep convolutional neuronal networks (D-CNN) architectures Multimedia Tools and Applications (2021).

[32] Sudharani, K., Sarma, T. C., Rasad, K. S Intelligent Brain Tumor lesion classification and identification from MRI images using k-NN technique In 2015 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT) IEEE (2015).

[33] Ahmmed, R., Swakshar, A. S., Hossain, M. F., Rafiq, M. AClassification of tumors and it stages in brain MRI using support vector machine and artificial neural network. In 2017 International Conference on Electrical, Computer and Communication Engineering (ECCE) (2015).

[34] Machhale, K., Nandpuru, H. B., Kapur, V., Kosta, L MRI brain cancer classification using hybrid classifier (SVM-KNN) In 2015 International Conference on Industrial Instrumentation and Control (ICIC) (2015).

[35] Melanie Bauer, Celine Berger, Kathrin Gerlach, Eva Scheurer, Claudia Lenz, Post mortem evaluation of brain edema using quantitative MRI, Forensic Science International (2022). 46

[36] Shinsuke Koike, Saori C. Tanaka, Tomohisa Okada, Toshihiko Aso, et al. Brain/-MINDS beyond human brain MRI project: A protocol for multi-level harmonization across brain disorders throughout the lifespan, NeuroImage: Clinical (2022).

[37] Noman Haleem, Matteo Bustreo, Alessio Del Bue, A computer vision based online quality control system for textile yarns Computers in Industry, (2022).

[38] Suman Paneru, Idris Jeelani, Computer vision applications in construction: Current state, opportunities & challenges (2021)

[39] Prashant Saurabh Minz, Charanjiv Singh Saini, Comparison of computer vision system and colour spectrophotometer for colour measurement of mozzarella cheese, (2021).

[40] N. Rozendorn, G. Greenberg, O. Madgar, I. Gluck, M. Vered, E. Alon, A. Dobriyan , Preoperative MRI for oral tongue squamous cell carcinoma: timing and correlation to histopathology International Journal of Oral and Maxillofacial Surgery (2022)

[41] SCHÖLKOPF, Bernhard. Causality for machine learning. In: Probabilistic and Causal Inference: The Works of Judea Pearl (2022)

[42] MURPHY, Kevin P. Probabilistic machine learning: an introduction. MIT press (2022)

[43] PHOON, Kok-Kwang; ZHANG, Wengang. Future of machine learning in geotechnics. Georisk: Assessment and Management of Risk for Engineered Systems and Geohazards (2022)

[44] BHAT, Nayeem Ahmad; FAROOQ, Sheikh Umar. An Improved Method for Training Data Selection for Cross-Project Defect Prediction. Arabian Journal for Science and Engineering (2022)

[45] KOSTOPOULOS, Georgios; KOTSIANTIS, Sotiris Exploiting semi-supervised learning in the education field: A critical survey Advances in Machine Learning/Deep Learning-Based Technologies (2022)

[46] BHAGYA RAJ, G. V. S.; DASH, Kshirod K. Comprehensive study on applications of artificial neural network in food process modeling Critical Reviews in Food Science and Nutrition (2022)

[47] Available Online: **https://www.kaggle.com/navoneel/brain-mri-images-forbrain-tumor-detection**

[48] Acharya, U.R.; Sree, S.V.; Ribeiro, R.; Krishnamurthi, G.; Marinho, R.T.; Sanches, J.; Suri, J.S. Data mining framework for brain tumor disease classification using SVM: A hybrid feature extraction paradigm (2016)

[49] E. Dervisevic, S. Pavljasevic, A. Dervisevic, S.S. Kasumovic, Challenges In Early Brain tumor Detection using LTSM and RNN, (2016)

[50] R. Bock, J. Meier, L. G. Nyl, G. Michelson, Brain tumor risk index: automated brain tumor detection from color chronic images using Deep Belief Network DBN, Medical Image Analysis (2017)

[51] **https://eng.snu.ac.kr/online/engmed/06ebb0/95eb2a0/eab8b/ec9ca4ec/84b1eb/a19ceab5/90ec88**

[52] **https://machinethink.net/blog/googles-mobile-net-architecture-on-iphone/**

[53] Akram, M. U., Usman. Computer aided system for brain tumor detection and segmentation In International conference on Computer networks and information technology (2022)

[54] VEERAMUTHU, A., et al. MRI brain tumor image classification using a combined feature and image-based classifier Frontiers in Psychology, 2022

[55] Salçin, K Detection and classification of brain tumours from MRI images using faster R-CNN (2022)

[56] BRINDHA, P. Gokila, et al. Brain tumor detection from MRI images using deep learning techniques In: IOP Conference Series: Materials Science and Engineering (2021)

[57] Younis, A., Qiang, L., Nyatega, C. O., Adamu, M. J., Kawuwa, H. B. Brain Tumor Analysis Using Deep Learning and VGG-16 Ensembling Learning Approaches (2022)

[58] Dehkordi, A. A., Hashemi, M., Neshat, M., Mirjalili, S., Sadiq, A. S. Brain Tumor Detection and Classification Using a New Evolutionary Convolutional Neural Network (2022) 60. T. A. Jemimma and Y. J. V. Raj, "Brain Tumor Segmentation and Classification Using Deep Belief Network," Second International Conference on Intelligent Computing and Control Systems (ICICCS) (2018)

[59] Mawaddah, S., Mufid, M. R., Basofi, A., Fiyanto, A., Aditama, D., Nurlaila, N. Rhizome Image Classification Using Support Vector Machine. In International Conference on Applied Science and Technology on Social Science (2021) 48

[60] Amin, Javaria Sharif, Muhammad Raza, Mudassar Saba, Tanzila Sial, Rafiq Shad, Shafqat Brain tumor detection: a long short-term memory (LSTM)-based learning model Neural Computing and Applications. (2020)

[61] E. Noether. Invariante Variationsprobleme. *Nachr. d. König. Gesellsch. d. Wiss. zu Göttingen, Math-phys. Klasse, Seite 235-157*, 1918.

[62] A. M. Turing. Computing machinery and intelligence *Mind*, 59:433–460, 1950.

[63] Mohamed Arbane1, Rachid Benlamri2, Youcef Brik1 and Mohamed Djerioui1 1LASS laboratory, Facuty of technology, Mohamed Boudiaf University, Msila, Algeria 2Department of Software Engineering, Lakehead University, Ontario, Canada.

[64] Goodfellow, I.; Bengio, Y.; Courville, A. Deep Learning; M.I.T. Press: Cambridge, MA, USA, 2016. [Google Scholar]

[65] Kamnitsas, K.; Ledig, C.; Newcombe, V.F.J.; Simpson, J.P.; Kane, A.D.; Menon, D.K.; Rueckert, D.; Glocker, B. Efficient multiscale 3D CNN with fully connected C.R.F. for accurate brain lesion segmentation. Med. Image Anal.

[66] Kwon, D.; Akbari, H.; Da, X.; Gaonkar, B.; Davatzikos, C. Multimodal brain tumor image segmentation using GLISTR. In Proceedings of the BRATS-MICCAI(2014), Boston, MA, USA, 14 September 2014.

[67] Devi, R.L. Detection and Automated Classification of Brain Tumor Types in MRI Images using a Neural Network with Grid Search Optimization. In Proceedings of the 2021 Fifth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Palladam, India, 1113 November 2021; IEEE: Piscataway, NJ, USA.

[68] Pandian, A.A.; Balasubramanian, R. Fusion of contourlet transform and zernike moments using content based image retrieval for M.R.I. brain tumor images. Indian J.

[69] Sci. Technol. Litjens, G.; Kooi, T.; Bejnordi, B.E.; Setio, A.A.A.; Ciompi, F.; Ghafoorian, M.; Sánchez, C.I. A survey on deep learning in medical image analysis. Med Image Anal.

[70] Brainweb:SimulatedBrainDatabase=> **http://brainweb.bic.mni.mcgill.ca/cgi/brainweb1**.

[71] Obtainable Online: **www.cancer.ca/ /media/CCE 10/08/2015**

[72] J. C. Buckner, P. D. Brown, B. P. ONeill, F. B. Meyer , C. J. Wetmore,J. H Uhm, "Central nervous system tumors." In Mayo Clinic Proceedings,Vol. 82, No. 10, pp. 1271- 1286, October 2007.

[73] Deepa , Singh Akansha. (2016). - Review of Brain Tumor Detection from tomography. International Conference on Computing for Sustainable Global Development (INDIA-Com)

[74] R. A. Novellines, M. D. - Squire's fundamentals of radiology; Six Edition; UPR, 2004.

[75] preston, D. c. (2006). Magnetic Resonance Imaging (MRI) of the Brain and Spine from Basics. casemed.case.edu .

[76] Hendrik RE. (2005) Glossary of MR Terms from American College of Radiology .

[77] .A. Demirhan, M. Toru, and I. Guler, Segmentation of tumor and edema along with healthy tissues of brain using wavelets and neural networks, IEEE Journal of Biomedical and Health Informatics, vol. 19, no. 4, pp. 14511458, 2015.

[78] Nilesh Bhaskarrao Bahadure, A.K. (2017, March 6). Retrieved from **https://www.hindawi.com/journals/ijbi/2017/9749108/**.

[79] S. Mohsin, S. Sajjad, Z. Malik, and A. H. Abdullah, Efficient way of skull stripping in MRI to detect brain tumor by applying morphological operations, after detection of false background, International Journal of Information and Education Technology, vol. 2, no. 4, pp. 335337, 2012.

[80] Gavale, P. M., Aher, P. V., Wani, D. V. (2017, April 4). Retrieved from **https://www.irjet.net/archives/V4/i4/IRJET-V4I462.pdf**.