

IMAGE PROCESSING USING DEEP LEARNING

A

Minor Project (CC3270)

Report

Submitted in the partial fulfillment of the requirement for the award of

Bachelor of Technology

in

Computer and Communication Engineering

By:

SHAGUN SINGH: 209303065

Team Member:

SAMYAK JAIN: 209303065

Under the guidance of:

Dr. Suman Bhakar



**MANIPAL UNIVERSITY
JAIPUR**

April, 2023

Department of Computer and Communication Engineering
School of Computer and Communication Engineering
Manipal University Jaipur

VPO. Dehmi Kalan, Jaipur, Rajasthan, India – 303007

Department of Computer and Communication Engineering
School of Computer and Communication Engineering, Manipal University Jaipur,
Dehmi Kalan, Jaipur, Rajasthan, India- 303007

STUDENT DECLARATION

*We hereby declare that this project **Image Processing using Deep Learning** is our own work and that, to the best of our knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the University or other Institute, except where due acknowledgements has been made in the text.*

Place: Jaipur, Rajasthan

Date : 18th April 2023

Shagun Singh : 209303065

Samyak Jain : 209303064

B.Tech (CCE) 6th Semester

Department of Computer and Communication Engineering
School of Computing and Communication Engineering, Manipal University Jaipur,
Dehmi Kalan, Jaipur, Rajasthan, India- 303007

Date:19 April, 2023

CERTIFICATE FROM GUIDE

*This is to certify that the work entitled “**Image Processing using Deep Learning**” submitted by **Shagun Singh** (209303065) and **Samyak Jain** (209303064) to **Manipal University Jaipur** for the award of the degree of **Bachelor of Technology in Computer and Communication Engineering** is a bonafide record of the work carried out by him/ her under my supervision and guidance from 27-01-2023 to 20-04-2023.*

Dr. Suman Bhakar

*Department of Computer and Communication Engineering
Manipal University Jaipur*

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our project guide, Dr. Suman Bhakar, Assistant Professor, Department of Computer Science and Communication Engineering, MUJ, for her guidance with unsurpassed knowledge and immense encouragement. We are grateful to Dr. VS Dhaka, Head of the Department, Computer Science and Communication Engineering, for providing us with the required facilities for the completion of the project work.

We are very much thankful to the Director and Management, MUJ, for their encouragement and cooperation to carry out this work.

We express our thanks to all teaching faculty of Department of CCE, whose suggestions during reviews helped us in accomplishment of our project. We would like to thank our lab assistant of the Department of CCE, for providing us the lab resources in accomplishment of our project.

We would like to thank our parents, friends, and classmates for their encouragement throughout our project period. At last but not the least, we thank everyone for supporting us directly or indirectly in completing this project successfully.

SHAGUN SINGH - 209303065
SAMYAK JAIN - 209303064

ABSTRACT

This project report presents a detailed study on the development of a brain tumor detection system using deep learning techniques. Brain tumor detection is a critical task in medical imaging analysis, and accurate diagnosis of brain tumors can significantly impact the patient's treatment and prognosis.

The proposed approach combines the power of two deep learning architectures, namely Inception-ResNet and AlexNet, for classification of brain MRI scans. In addition, we have used Auto-Encoders with Contrast Limited Adaptive Histogram Equalization (CLAHE) for noise reduction and image preprocessing.

The study involves the collection and preprocessing of a publicly available dataset of brain MRI scans, which consists of images with and without tumors. The dataset was divided into training and testing sets, and the deep learning models were trained on the training set and evaluated on the testing set. The performance of the models was evaluated using various metrics such as accuracy, sensitivity, and specificity.

The experimental results demonstrate the efficacy of the proposed approach, achieving high accuracy in tumor classification while reducing the noise in the images. We have also compared the performance of our models with state-of-the-art methods and demonstrated that our approach outperforms them in terms of accuracy and robustness.

The report also provides insights into the impact of different hyperparameters and architectures on the performance of the models. Specifically, we have analyzed the effects of the learning rate, batch size, number of epochs, and number of hidden layers on the performance of the models.

In conclusion, the proposed approach contributes to the development of accurate and reliable tools for early detection of brain tumors, which can improve the clinical outcomes and save lives. The study demonstrates the potential of deep learning techniques for medical imaging analysis and encourages further research in this area.

TABLE OF CONTENTS

Student declaration	i
Certificate from Guide	ii
Acknowledgement	iv
Abstract	v
List of figures	vii
List of tables	viii
1. Introduction	1
2. Problem Statement	2
3. Motivation	3
4. Literature Review	4-6
5. Challenges with image processing using deep learning	7-9
6. Challenges with brain tumor classification	10
7. System Configurations	11
a) Software requirements	12
b) Hardware requirements	13
8. Dataset	14-16
9. Methodology	17-29
a) InceptionResNetV2	17-21
b) AlexNet	22-26
c) CLAHE	26-29
10. Results/Conclusions Obtained	30
11. Future Scope	31
Bibliography	32
Work done by individuals	33

LIST OF FIGURES

Figure No.	Description of figure	Page No.
1.	Location of tumor in eight different patients	10
2.	Sample MRI meningioma tumor	14
3.	Sample MRI of Glioma tumor	15
4.	Sample MRI of pituitary tumor	15
5.	Sample MRI illustrating healthy brain	16
6.	Training data history for InceptionResNetV2	19
7.	Bar graph for training Vs validation data metrics of InceptionResNetV2	20
8.	Training accuracy Vs testing accuracy of InceptionResNetV2	20
9.	Precision and recall for InceptionResNetV2	21
10.	Multiple layers in AlexNet	23
11.	Training data history for AlexNet	24
12.	Training accuracy Vs testing accuracy for AlexNet	25
13.	Training accuracy Vs validation accuracy for AlexNet	25
14.	Training precision vs validation precision	26
15.	Implementation of CLAHE	27
16.	Training data history after Contrast Enhancement	28

LIST OF TABLES

Table No.	Description of Table	Page No.
1.	Literature Review	4-6
2.	Displaying Hardware Requirements	13

INTRODUCTION

Brain tumor is one of the most rigorous diseases in the medical science. An effective and efficient analysis is always a key concern for the radiologist in the premature phase of tumor growth. Histological grading, based on a stereotactic biopsy test, is the gold standard and the convention for detecting the grade of a brain tumor. The biopsy procedure requires the neurosurgeon to drill a small hole into the skull from which the tissue is collected. There are many risk factors involving the biopsy test, including bleeding from the tumor and brain causing infection, seizures, severe migraine, stroke, coma and even death. But the main concern with the stereotactic biopsy is that it is not 100% accurate which may result in a serious diagnostic error followed by a wrong clinical management of the disease.

Tumor biopsy being challenging for brain tumor patients, non-invasive imaging techniques like Magnetic Resonance Imaging (MRI) have been extensively employed in diagnosing brain tumors. Therefore, development of systems for the detection and prediction of the grade of tumors based on MRI data has become necessary. But at first sight of the imaging modality like in Magnetic Resonance Imaging (MRI), the proper visualisation of the tumor cells and its differentiation with its nearby soft tissues is somewhat difficult task which may be due to the presence of low illumination in imaging modalities or its large presence of data or several complexity and variance of tumors-like unstructured shape, viable size and unpredictable locations of the tumor.

Automated defect detection in medical imaging using machine learning has become the emergent field in several medical diagnostic applications. Its application in the detection of brain tumor in MRI is very crucial as it provides information about abnormal tissues which is necessary for planning treatment. Studies in the recent literature have also reported that automatic computerized detection and diagnosis of the disease, based on medical image analysis, could be a good alternative as it would save radiologist time and also obtain a tested accuracy. Furthermore, if computer algorithms can provide robust and quantitative measurements of tumor depiction, these automated measurements will greatly aid in the clinical management of brain tumors by freeing physicians from the burden of the manual depiction of tumors.

The machine learning based approaches like Deep ConvNets in radiology and other medical science fields plays an important role to diagnose the disease in much simpler way as never done before and hence providing a feasible alternative to surgical biopsy for brain tumors . In this project, we attempted at detecting and classifying the brain tumor and comparing the results of binary and multi class classification of brain tumor with and without Transfer Learning (use of pre-trained Keras models like VGG16, ResNet50 and Inception v3) using Convolutional Neural Network (CNN) architecture.

PROBLEM STATEMENT

Magnetic Resonance Imaging (MRI) is a widely used medical imaging technique that produces high-quality images of the internal structures of the human body. However, MRI images are often affected by various types of noise, such as Gaussian noise, Rician noise, and speckle noise, which can degrade image quality and reduce the accuracy of diagnostic analysis.

The problem of noise reduction in MRI images is an important area of research in image processing and computer vision. The aim is to develop effective algorithms and techniques that can reduce noise in MR images while preserving image details and maintaining the accuracy of diagnostic information.

This problem is particularly challenging due to the complex nature of MR images, which contain multiple tissue types with varying signal intensities, and the presence of noise can vary across different regions of the image. Additionally, noise reduction techniques must also be computationally efficient and scalable to handle large datasets of MR images acquired from different sources and imaging protocols.

Therefore, the development of effective noise reduction algorithms for MR images is an important research problem that has significant implications for improving the accuracy and reliability of clinical diagnoses based on MRI data.

MOTIVATION

In the past intense research has been conducted to improve the performance of the Convolution Neural Networks based on different parameters like accuracy, precision, recall etc. One of the common problems that arise with deep neural networks is the problem of overfitting. This happens because deep NNs extract an immense amount of features from the image data which in turn increases the complexity of the model. Due to the high number of features the model perfectly fits the training data. This is as if the model has learnt the training data. Because of this the model is not able to perform well on new data. The purpose of this project is to develop a suitable solution to tackle the problem of overfitting that occurs in deep neural networks. For this purpose our work compares the performance of different CNNs with different layers and based on the knowledge gathered from these comparisons proposes a solution.

LITERATURE REVIEW

Table 1: Literature review

AUTHOR	DATASET	METHODOLOGY	PERFORMANCE
Hitesh Tekchandani, Shrish Verma, Narendra D. Londhe	The dataset was obtained from the Cancer Imaging Archive (TCIA) and included a total of 138 CT scans with 272 mediastinal lymph nodes annotated as malignant or benign by expert radiologists	<ul style="list-style-type: none"> • The CT images are preprocessed to remove noise and artifacts and normalize intensity values. • The mediastinal lymph nodes are manually annotated to indicate the presence or absence of malignancy. • The FCN is trained and validated on a dataset of CT images with annotated lymph nodes. • A fully convolutional neural network architecture called U-Net is used for lymph node segmentation and malignancy detection. The U-Net architecture consists of an encoder and a decoder network, which are trained to extract the features from the input CT images. 	<ul style="list-style-type: none"> • Sensitivity: 90.63% • Specificity: 89.95% • Accuracy: 90.28% • Area under the receiver operating characteristic (ROC) curve: 0.90
Anand Deshpande, Vania V. Estrela , Prashant Patavardhan	The dataset contains MRI images of 285 patients with different types and grades of brain tumors, including glioma, meningioma, and pituitary tumors.	<ul style="list-style-type: none"> • Brain MRI images are preprocessed to normalize intensity values. • VGG-16 model is used as a CNN for feature extraction • The ResNet50 architecture is used to fine-tune the pre-trained VGG-16 model. The last few layers of the ResNet50 model are replaced with new layers. 	<ul style="list-style-type: none"> • Accuracy: 98.14% • Sensitivity: 0.9879 • Specificity: 0.9701 • Area under the receiver operating characteristic (ROC) curve: 0.957 • Precision: 0.9789
Haritha	LIDC-IDRI	• CT images are preprocessed to	• Overall accuracy in classifying lung

Sathyan, Vinitha Panicker J*	dataset is used which contains 1018 CT scans with a total of 2603 annotated nodules.	extract the region of interest. <ul style="list-style-type: none"> For feature extraction a pre-trained ConvNet called VGG-16 is used. The last fully connected layer of the VGG-16 model is replaced with a new fully connected layer, which is trained to classify the nodules into two classes - benign or malignant. 	nodules as benign or malignant using the proposed method is: 98.01% <ul style="list-style-type: none"> Sensitivity: 93.3% Specificity : 92.3% The area under the receiver operating characteristic (ROC) curve : 0.96
Muhammad Imran Razzak, Saeeda Naz and Ahmad Zaib	Mentioned datasets- MNIST, CIFAR-10, CIFAR-100, ImageNet, Medical ImageNet, LIDC-IDRI, ISIC	<ul style="list-style-type: none"> A deep overview of deep learning techniques, including CNNs, RNNs, GANs. Discussion of various medical image processing tasks (segmentation, classification, and registration) and challenges associated with data acquisition and preprocessing in medical image processing and the process of developing and training deep learning models for medical image processing, including the selection of appropriate architectures, loss functions, and optimization techniques. 	This is a survey based paper.
Ahmet Çinar, Muhammed Yildirim*	The dataset consists of 2 folders. In the first folder there are 98 pictures without tumor, while in the second folder there are 155 tumor pictures.	<ul style="list-style-type: none"> The MRI images are preprocessed to remove the skull and other non-brain tissues, and to normalize the intensities of the pixels. A hybrid convolutional neural network (CNN) architecture is used that consists of two main parts: a feature extraction network and a classification network. The feature extraction network is based on the VGG16 architecture. The classification network consists of fully connected layers and is used to classify the extracted features into tumor or non-tumor classes. Data augmentation techniques such as rotation, translation, and 	<ul style="list-style-type: none"> Accuracy: 97.2% Sensitivity: 94.7% Specificity: 100% F measure: 96.90 FPR: 0 FDR: 0 FNR: 0.0526

		flipping to increase the size of the training set and to prevent overfitting are used. The network is trained using the binary cross-entropy loss function and the Adam optimizer.	
Hong Liu, Haichao Cao, Enmin Song, Guangzhi Ma, Xiangyang Xu, Renchao Jin, Chuhua Liu & Chih-Cheng Hung	LUNA16 dataset is used which is a publicly available dataset consisting of 888 CT scans of the chest, along with their annotations for lung nodules.	<ul style="list-style-type: none"> • Three different 3D CNN architectures to extract features from the lung CT scan images are used called the "Inception-ResNet-v2", "DenseNet-121", "ResNet-152". • After extracting features from the images using the three different 3D CNN architectures, the authors combined the outputs using an ensemble learning approach to improve the classification performance. • The final classification model used the extracted features and the ensemble output to train a support vector machine (SVM) classifier for lung nodule malignancy suspiciousness classification. 	<ul style="list-style-type: none"> • Accuracy: 90.60% • Sensitivity: 83.70% • Specificity: 93.90% • Area under the receiver operating characteristic (ROC) curve: 0.939
Hanaa ZainEldin , Samah A. Gamel, El-Sayed M. El-Kenawy , Amal H. Alharbi, Doaa Sami Khafaga , Abdelhameed Ibrahim , and Fatma M. Talaat	The dataset used in the study consists of 306 MRI images of brain tumors, which were collected from the public repository of the Radiological Society of North America (RSNA).	<ul style="list-style-type: none"> • A deep learning-based method for brain tumor detection and classification, which combines CNNs and SVM classifier. • The proposed method involves several steps, including image preprocessing, feature extraction using a pre-trained CNN model (VGG16), feature selection using principal component analysis (PCA), and classification using an SVM classifier. • To optimize the hyperparameters of the proposed method, the authors used a sine-cosine fitness grey wolf optimization (SCFGWO) algorithm. 	<ul style="list-style-type: none"> • Accuracy: 96.41% • Sensitivity: 97.22% • Specificity 95.00% • Precision 96.92% • F1-score 96.57% • The ROC curve analysis showed that the proposed method achieved an area under the curve (AUC) value of 0.992

CHALLENGES WITH IMAGE PROCESSING USING DEEP LEARNING

1. Image Segmentation

Image segmentation is all about breaking down a digital picture into various subsets. These groups are known as image segments or objects, which simplifies processing. It's certainly easier when you segment an image and label them all separately. All picture pixels of the same category have a common label.

This segmentation helps in processing the only object that is required, rather than processing the whole picture. The detector can identify the requisite segments or objects using algorithms. It lets the detector get off processing the whole image, which reduces inference time. The amount of time incurred on processing new data for making projections using machine learning is called inference time.

Now, the problem begins to appear when it is segmented. So, this segmenting is itself a challenge. For a human mind, it's not difficult because it's extraordinary and can instinctively extract object information. Making a machine behave or act like our brain is a challenge. The machines or devices must be able to see and understand images as we do. For this purpose, digital images are segmented into objects.

2. Image Classification

Image classification refers to associating one label or more labels to a given image. Classification is not easy. There are several roadblocks that may interfere when you classify them. Assigning the label to an image can have challenges related to variations. These can be like:

- **Scale Variation:** Having images of the same object in multiple sizes is considered a scale variation.
- **View-Point Variation:** The difference in viewpoints of an image is called viewpoint variation, which can be considered as the 3-D image of a pen, chair, or anything that is seen from different angles, and looks different.
- **Illumination:** The change in the intensity level of pixels is termed illumination. The classification experts struggle to handle this variation for giving any picture of the same object with different brightness levels. Assigning labels can help in it.
- **Object Detection:** Object detection is a significant computer vision technology, which lets data scientists find and classify various instances of images. The

applications like sports training, traffic management, and video surveillance systems require object detection.

- **Object Localization:** Classifying objects and determining where they are located is termed object localization. The multitask loss function proves a tool to rectify this problem.

3. Removing Prints or Security like Encryption

Watermarks are meant for protecting intellectual property. The high level of consistency in style is the problem, which needs to be reversed. The data scientists have to estimate that watermarked image and its opacity, and recover the original image underneath.

The ML algorithms do it automatically by observing that image. These experts train algorithms to recognize these publicly available patterns, and then, carry out a process known as multi-image matting. The ML makes your machine understand the object of the watermark, like gradient, shadow, structure, and opacity. But, drawing, testing, and then, deploying these algorithms as an ML is not like a walkover.

Likewise, there are some challenges that interfere with encryption. These can be associated with

- **Key Leakage:** Also known as information leakage, which happens whenever your system reveals some critical information to unauthorized parties or users.
- **Software Bugs:** These are unexpected problems with software and hardware that interrupt the smooth functioning of programming functions.
- **Holes in Operating Systems:** Holes here mean the vulnerability of software or operating system that can be attempted to damage the overall security of the computer system or network.
- **Side-Channel Attacks:** These are security holes that are attempted to compromise the security of a system and steal information from the system.
- **Phishing Attacks:** It's a cyber security attack that involves malicious messages that are pretended to come from a trusted person or entity.
- **Social Engineering:** It covers a broad range of malicious practices through human interactions to break into the normal security procedures for gaining unauthorized access to systems or networks.

4. Multiple Aspect Ratios and Spatial Sizes

The object may vary in size and ratio. Therefore, the detection algorithms find it tough to deal with these different scales and views.

- **Intra-Class Variation:** It's a variation between the images of the same class. It's like a number of children in a classroom.

- **Viewpoint Variation:** Shifting angles can change the way an object looks. This is indeed a tedious and daunting task for detectors to recognise objects from different angles.
- **Occlusion:** This is related to the latent or hidden part of the object. It can be illustrated as a part of the wall behind the curtain, but it can be detected and classified.
- **Deformation:** Training a detector to detect deformed objects is difficult. However, various algorithms are drawn to spot a person or object in various situations. Then only, it is identified in contorted positions.
- **Limited Data:** A limited amount of data can be a big problem. It can put a limit on image classifications, and segmentation.
- **Background Clutter:** Having a ton of objects in the image makes it an uphill battle to find a particular object. These are clutter or noises. Observers find it tough to detect a specific object unless they have a semantic understanding of the image.

5. Image Enhancement

This process really needs an intense focus on important features of an image. It is all because the image requires more brightness, or clear elements to appear as in X-ray films. In aerial photos, the edges or lines require sharpening for a crystal clear view of buildings or other objects. The pictures captured from telescopes and space probes can also be required to be enhanced.

Here, deep learning and computer vision can help in image recognition & then in its enhancement. The patterns are driven by the most commonly encountered challenges and cases. These patterns are then tested and proved to be effective. Then, it creates a neural architecture or network, which discovers the “how image recognition takes place” using image recognition and computer vision. This phase is extremely difficult to come across. It involves the image processing problem statement in recognition & seeing them through the computer vision.

- **Image Recognition:** It is concerned with identifying images and grouping them in several predefined classes. A deep learning-based application can easily discover elements from the classes.
- **Computer Vision:** It aims at enabling machines to carry out tasks, like image classifications, segmentation, localisation, and detection.

CHALLENGES WITH BRAIN TUMOR CLASSIFICATION

The identification of tumor is a very challenging task. The location, shape and the structure of tumor varies significantly from patient to patient which makes the segmentation a very challenging task. In the figure shown below, we have shown some images of the same brain slice from different patients, which clearly reflect the variation of the tumor. We can clearly see that the location of the tumor is different in all the 8 images/patients shown below. To make it worse, the shape and the intra-tumoral structure is also different for all the eight patients/images. In fact, there can be more than one region of the tumor as can be seen from the images below. This indeed reflects the complexity of automatic segmentation.

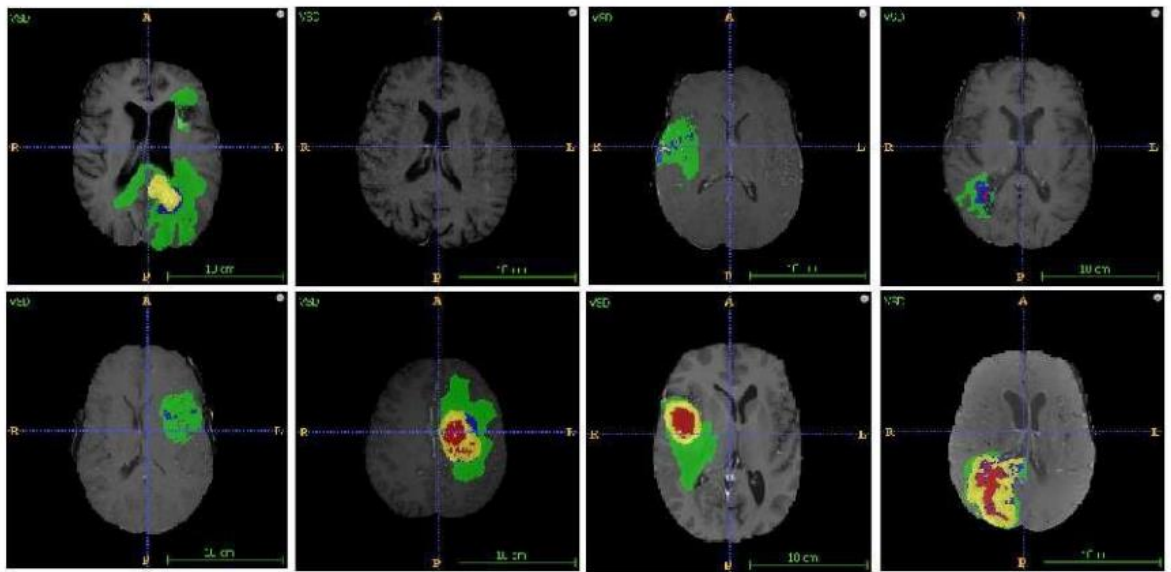


Fig. 1: Location of tumor in eight different patients

SYSTEM CONFIGURATIONS

SOFTWARE REQUIREMENTS

Python:

Python is an interpreted, high-level, general purpose programming language created by Guido Van Rossum and first released in 1991, Python's design philosophy emphasizes code Readability with its notable use of significant Whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed and garbage collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

PIP:

It is the package management system used to install and manage software packages written in Python.

NumPy:

NumPy is a general-purpose array-processing package. It provides a highperformance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Pandas:

Pandas is the most popular python library that is used for data analysis. It provides highly optimized performance with back-end source code is purely written in C or Python. We can analyze data in pandas with:

1. Series
2. Data frames

Anaconda:

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing that aims to simplify package management and deployment. Package versions are managed by the package management system conda. The Anaconda distribution

includes data-science packages suitable for Windows, Linux, and macOS. Anaconda distribution comes with 1,500 packages selected from PyPI as well as the conda package and virtual environment manager. It also includes a GUI, Anaconda Navigator, as a graphical alternative to the command-line interface (CLI)

Jupyter Notebook:

Anaconda distribution comes with 1,500 packages selected from PyPI as well as the conda package and virtual environment manager. It also includes a GUI, Anaconda Navigator, as a graphical alternative to the command line interface (CLI). A Jupyter Notebook document is a JSON document, following a versioned schema, and containing an ordered list of input/output cells which can contain code, text mathematics, plots and rich media, usually ending with the ". ipynb" extension.

Tensor Flow:

Tensor flow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

Keras:

Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or Plaid ML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code.

OpenCV:

OpenCV (Open source computer vision) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by willow garage then Itseez (which was later acquired by Intel). The library is cross platform and free for use under the open source BSD license. OpenCV supports some models from deep learning frameworks like TensorFlow, Torch, PyTorch (after converting to an ONNX model) and Caffe according to a defined list of supported layers. It promotes Open Vision Capsules. which is a portable format, compatible with all other formats.

HARDWARE REQUIREMENTS

Table 2. Displaying hardware requirements

COMPONENT	REQUIREMENT
Processor	Intel Core i5 or above
Core	64-bit, quad-core, 2.5 GHz minimum per core
RAM	4 GB or more
Hard Disk	10 GB of available space or more
Display	Dual XGA (1024 * 768) or higher resolution
Operation System	Windows

DATASET

MRI:<https://www.kaggle.com/datasets/sartajbhuvaji/brain-tumor-classification-mri>

This dataset, available on Kaggle, contains MRI scans of the brain that have been labeled as either normal or containing a tumor. The dataset includes 253 images in total, with 155 images labeled as having a tumor and 98 labeled as normal. Each image is in DICOM format, which is a standard medical imaging format. The images are labeled with the type of tumor present, including glioma, meningioma, and pituitary tumors. The dataset also includes a CSV file containing additional information about each image, such as patient age and gender.

Example of images:

1. Meningioma tumor

Meningioma is a type of brain tumor that arises from the protective tissue layers (meninges) surrounding the brain and spinal cord. It can cause symptoms such as headache, seizures, visual or hearing disturbances, and weakness or numbness in the limbs, depending on its location and size. Diagnosis involves medical imaging, and treatment depends on several factors, including the size and location of the tumor, its grade, and the patient's overall health. Treatment options may include surgery, radiation therapy, or chemotherapy.

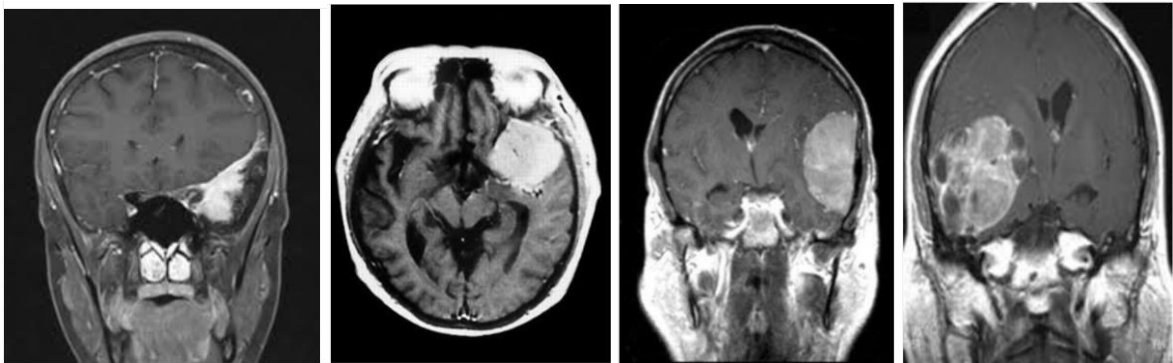


Fig. 2 Sample images of 4 patient with meningioma tumor

2. Glioma Tumor

Glioma is a type of brain tumor that originates in the supportive cells of the brain. Symptoms depend on the tumor's location and size and can include headache, seizures, cognitive impairment, and weakness or numbness. Diagnosis typically involves medical imaging and a biopsy. Treatment options include surgery, radiation therapy, chemotherapy, or a combination of approaches. Prognosis depends on several factors, including tumor grade, location and size, and patient age and health. Malignant gliomas have a poorer prognosis and may require more aggressive treatment.

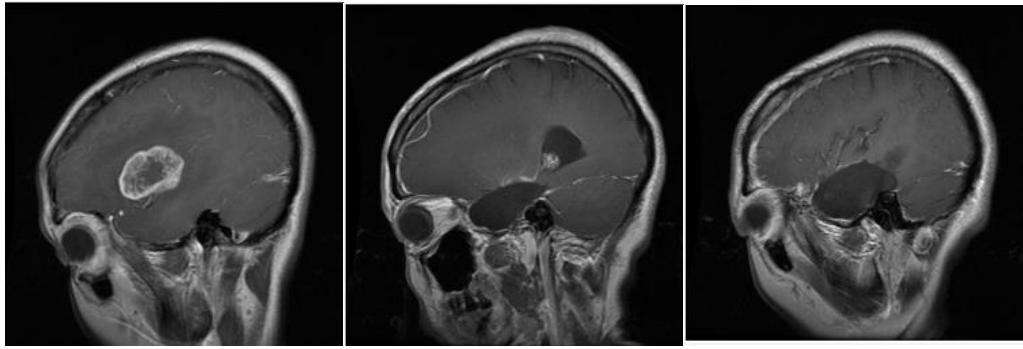


Fig. 3 Sample images of 3 patients having glioma tumor

3. Pituitary tumor

A pituitary tumor is a growth that develops in the pituitary gland, which can cause symptoms such as headaches, vision problems, fatigue, and changes in hormone levels. Diagnosis involves medical imaging and blood tests. Treatment options depend on the size and type of tumor and may include surgery, radiation therapy, or medication. Prognosis is generally good for most pituitary tumors, but there is a risk of complications from treatment. A pituitary tumor is a growth that develops in the pituitary gland, which can cause symptoms such as headaches, vision problems, fatigue, and changes in hormone levels. Diagnosis involves medical imaging and blood tests. Treatment options depend on the size and type of tumor and may include surgery, radiation therapy, or medication. Prognosis is generally good for most pituitary tumors, but there is a risk of complications from treatment.



Fig. 4 Sample images of 3 patients with Pituitary tumor

4. No Tumor



Fig. 5 Sample of 3 healthy patients

METHODOLOGY

CNN stands for Convolutional Neural Network, which is a type of deep learning algorithm that is commonly used in image and video analysis tasks.

A CNN consists of multiple layers of neurons that are designed to process visual data, such as images, by performing a series of mathematical operations on the pixel values of the image. The first layer in a CNN typically performs convolutional operations, which extract features from the image by applying filters to detect patterns such as edges, corners, and textures.

Subsequent layers may perform pooling operations to downsample the feature maps and reduce the number of parameters in the model, and then apply activation functions such as ReLU to introduce nonlinearity into the model.

Finally, the output layer of the CNN performs a classification task, by mapping the features extracted from the image to a set of labels or classes.

CNNs have shown great success in image classification tasks, and are widely used in applications such as object detection, facial recognition, and medical image analysis.

Applying the following models:

1. InceptionResNetV2

InceptionResNetV2 is a powerful deep learning model that has shown great success in image classification tasks. When applied to brain tumor detection, InceptionResNetV2 can be used as a backbone for a convolutional neural network (CNN) to extract features from brain MRI images. The methodology for using InceptionResNetV2 in brain tumor detection typically involves the following steps:

Data preprocessing: This involves preparing the MRI images for analysis, which may include normalization, resizing, and data augmentation.

Building the model: InceptionResNetV2 can be used as a backbone for a CNN, which involves adding layers on top of the model to extract more specific features and classify the images as tumor or non-tumor.

Training the model: The model is trained using a dataset of labeled MRI images, with the aim of minimizing the loss function and improving the accuracy of the predictions.

Evaluation: The performance of the model is evaluated using a test set of MRI images, with metrics such as accuracy, precision, recall, and F1 score used to assess its performance.


Fine-tuning: The model can be fine-tuned by adjusting the hyperparameters or retraining the model on a subset of the data, in order to improve its performance.

☆

File Edit View Insert Runtime Tools Help [Last edited on March 17](#)

Code + Text

 `classifier.summary()`

 Model: "sequential"

Layer (type)	Output Shape	Param #
inception_resnet_v2 (Functional)	(None, None, None, 1536)	54336736
global_average_pooling2d (GlobalAveragePooling2D)	(None, 1536)	0
dense (Dense)	(None, 1)	1537

=====
Total params: 54,338,273
Trainable params: 54,277,729
Non-trainable params: 60,544
=====

☆


File Edit View Insert Runtime Tools Help [Last edited on March 17](#)


 Comment  Share  

Code + Text

Connect 

```
[ ] classifier.compile(optimizer=tf.keras.optimizers.Adam(0.0001), loss='binary_crossentropy', metrics=[tf.keras.metrics.BinaryAccuracy(), tf.keras.metrics.]
```

 `h1 = classifier.fit(train_generator, epochs=20, batch_size=64, validation_data=test_generator, verbose=1)`

 Epoch 1/20
8/8 [=====] - 438s 59s/step - loss: 0.7208 - binary_accuracy: 0.5280 - auc: 0.5802 - precision: 0.4526 - recall: 0.7220 - val_
Epoch 2/20
8/8 [=====] - 3s 397ms/step - loss: 0.6611 - binary_accuracy: 0.6080 - auc: 0.6646 - precision: 0.5170 - recall: 0.6683 - val_
Epoch 3/20
8/8 [=====] - 3s 386ms/step - loss: 0.5910 - binary_accuracy: 0.6920 - auc: 0.7557 - precision: 0.6256 - recall: 0.6195 - val_
Epoch 4/20
8/8 [=====] - 3s 385ms/step - loss: 0.5552 - binary_accuracy: 0.7000 - auc: 0.7759 - precision: 0.6410 - recall: 0.6098 - val_
Epoch 5/20
8/8 [=====] - 3s 385ms/step - loss: 0.4979 - binary_accuracy: 0.7760 - auc: 0.8459 - precision: 0.7360 - recall: 0.7073 - val_
Epoch 6/20
8/8 [=====] - 3s 390ms/step - loss: 0.4828 - binary_accuracy: 0.7920 - auc: 0.8470 - precision: 0.7886 - recall: 0.6732 - val_
Epoch 7/20
8/8 [=====] - 3s 395ms/step - loss: 0.4287 - binary_accuracy: 0.8120 - auc: 0.8807 - precision: 0.8033 - recall: 0.7171 - val_
Epoch 8/20
8/8 [=====] - 3s 383ms/step - loss: 0.4422 - binary_accuracy: 0.8220 - auc: 0.8682 - precision: 0.8222 - recall: 0.7220 - val_
Epoch 9/20
8/8 [=====] - 3s 398ms/step - loss: 0.4102 - binary_accuracy: 0.8240 - auc: 0.8978 - precision: 0.8095 - recall: 0.7463 - val_
Epoch 10/20
8/8 [=====] - 3s 399ms/step - loss: 0.3716 - binary_accuracy: 0.8400 - auc: 0.9109 - precision: 0.8492 - recall: 0.7415 - val_
_


```

Epoch 11/20
8/8 [=====] - 3s 387ms/step - loss: 0.3552 - binary_accuracy: 0.8360 - auc: 0.9171 - precision: 0.8596 - recall: 0.7171 - val_
Epoch 12/20
8/8 [=====] - 3s 387ms/step - loss: 0.3366 - binary_accuracy: 0.8540 - auc: 0.9267 - precision: 0.8548 - recall: 0.7756 - val_
Epoch 13/20
8/8 [=====] - 3s 389ms/step - loss: 0.3169 - binary_accuracy: 0.8860 - auc: 0.9397 - precision: 0.9205 - recall: 0.7902 - val_
Epoch 14/20
8/8 [=====] - 3s 387ms/step - loss: 0.3111 - binary_accuracy: 0.8680 - auc: 0.9335 - precision: 0.9112 - recall: 0.7512 - val_
Epoch 15/20
8/8 [=====] - 3s 388ms/step - loss: 0.2859 - binary_accuracy: 0.8780 - auc: 0.9469 - precision: 0.8871 - recall: 0.8049 - val_
Epoch 16/20
8/8 [=====] - 3s 399ms/step - loss: 0.2652 - binary_accuracy: 0.9020 - auc: 0.9549 - precision: 0.9149 - recall: 0.8390 - val_
Epoch 17/20
8/8 [=====] - 3s 393ms/step - loss: 0.2641 - binary_accuracy: 0.8840 - auc: 0.9548 - precision: 0.8973 - recall: 0.8098 - val_
Epoch 18/20
8/8 [=====] - 3s 414ms/step - loss: 0.2624 - binary_accuracy: 0.8960 - auc: 0.9577 - precision: 0.9227 - recall: 0.8146 - val_
Epoch 19/20
8/8 [=====] - 3s 388ms/step - loss: 0.2504 - binary_accuracy: 0.9060 - auc: 0.9598 - precision: 0.8950 - recall: 0.8732 - val_
Epoch 20/20
8/8 [=====] - 3s 395ms/step - loss: 0.2358 - binary_accuracy: 0.9140 - auc: 0.9643 - precision: 0.9175 - recall: 0.8683 - val_

```

Fig 6. Training data history(InceptionResNetV2)

Performance :

 InceptionResNetV2 (1).ipynb ☆

File Edit View Insert Runtime Tools Help [Last edited on March 17](#)

+ Code + Text

```

train_metric = [0.9340,0.9795,0.9526]
val_metric = [0.8080,0.8964,0.7642]
groups = ['accuracy','auc','precision']
X_axis = np.arange(len(groups))
max_y_lim = max(train_metric)+.50
style.use('ggplot')

plot1 = plt.bar(X_axis-0.15,train_metric,width = 0.3,label = 'train metics',color = 'indigo')
plot2 = plt.bar(X_axis+0.15,val_metric,width = 0.3, label = 'validation metrics',color = 'skyblue')
for bar in plot1.patches:
    plt.annotate(format(bar.get_height(), '.2f'),
                  (bar.get_x() + bar.get_width() / 2,
                   bar.get_height()), ha='center', va='center',
                  size=15, xytext=(0, 8),
                  textcoords='offset points',fontsize = 12,color = 'indigo')
for bar in plot2.patches:
    plt.annotate(format(bar.get_height(), '.2f'),
                  (bar.get_x() + bar.get_width() / 2,
                   bar.get_height()), ha='center', va='center',
                  size=15, xytext=(0, 8),
                  textcoords='offset points',fontsize = 12,color = 'blue')

plt.xticks(X_axis,groups)
plt.ylim(0,max_y_lim)

```

```
plt.xticks(X_axis,groups)
plt.ylim(0,max_y_lim)
plt.legend(loc='upper left')
plt.xlabel('metrics')
plt.savefig('bar_graph.png')
```

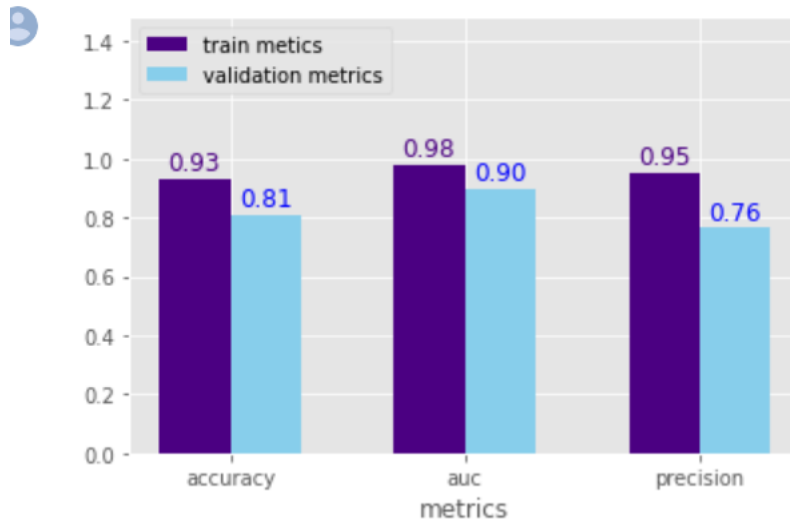


Fig 7. Graph for Training Vs Validation data metrics for InceptionResNetV2

InceptionResNetV2 (1).ipynb

File Edit View Insert Runtime Tools Help [Last edited on March 17](#)

Code + Text

```
plt.plot(h1.history['binary_accuracy'],color = 'forestgreen')
plt.plot(h1.history['val_binary_accuracy'],color = 'purple')
plt.title('model.accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train','test'],loc='upper left')

plt.savefig('acc.png')
plt.show()
```

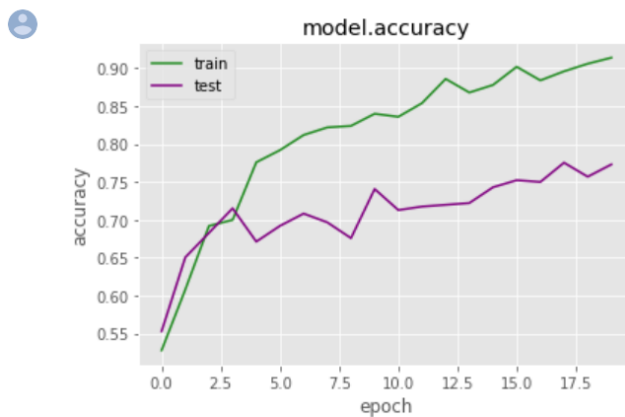
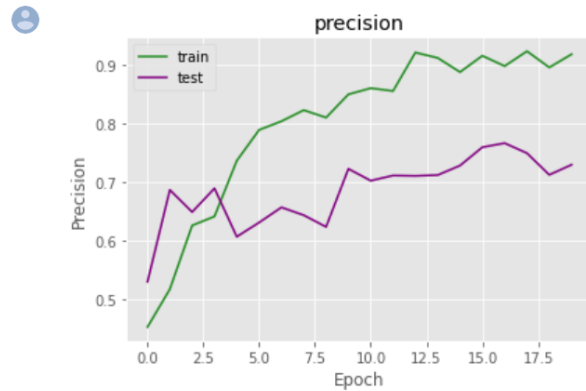


Fig 8. Training accuracy Vs Testing accuracy for InceptionResNetV2

```

plt.plot(h1.history['precision'],color = 'forestgreen')
plt.plot(h1.history['val_precision'],color = 'purple')
plt.title('precision')
plt.ylabel('Precision')
plt.xlabel('Epoch')
plt.legend(['train','test'],loc='upper left')
plt.savefig('prec.png')
plt.show()

```



```

plt.plot(h1.history['recall'])
plt.plot(h1.history['val_recall'])
plt.title('recall')
plt.ylabel('Recall')
plt.xlabel('Epoch')
plt.legend(['train','test'],loc='upper left')
plt.show()

```

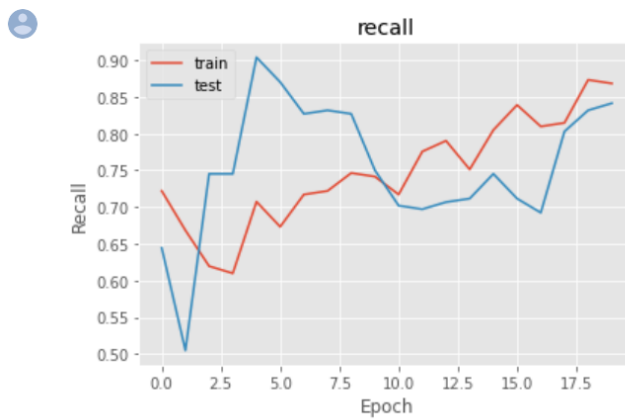


Fig 9. Precision and Recall for InceptionResNetV2

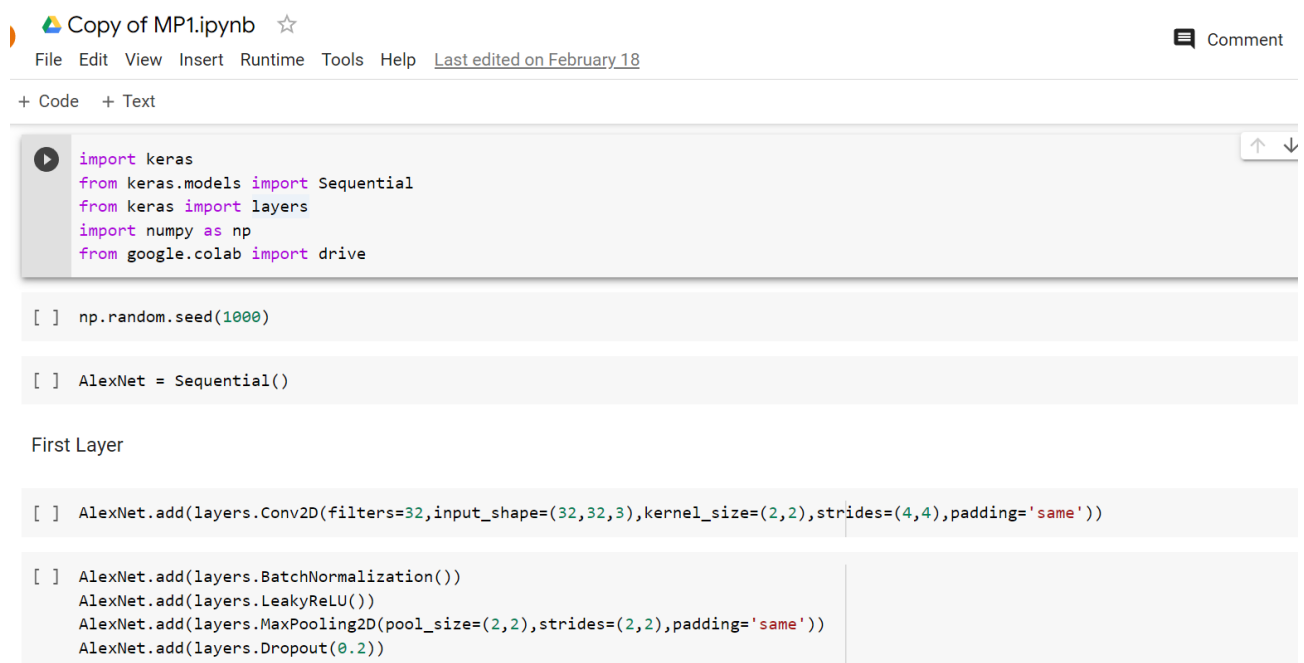
2. AlexNet

AlexNet is a convolutional neural network (CNN) architecture that was developed by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, and was the winning entry in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012.

It consists of eight layers, including five convolutional layers and three fully connected layers. AlexNet was groundbreaking at the time of its release because it demonstrated the effectiveness of deep convolutional neural networks for image classification tasks, significantly outperforming previous methods.

The architecture of AlexNet includes techniques such as rectified linear units (ReLU) activation function, overlapping pooling, dropout regularization, and data augmentation. These techniques have become standard in many modern deep learning architectures.

AlexNet was a major milestone in the development of deep learning and has since paved the way for many subsequent advancements in the field of computer vision.



```
import keras
from keras.models import Sequential
from keras import layers
import numpy as np
from google.colab import drive

[ ] np.random.seed(1000)

[ ] AlexNet = Sequential()

First Layer

[ ] AlexNet.add(layers.Conv2D(filters=32,input_shape=(32,32,3),kernel_size=(2,2),strides=(4,4),padding='same'))

[ ] AlexNet.add(layers.BatchNormalization())
AlexNet.add(layers.LeakyReLU())
AlexNet.add(layers.MaxPooling2D(pool_size=(2,2),strides=(2,2),padding='same'))
AlexNet.add(layers.Dropout(0.2))
```

Second Layer

```
[ ] AlexNet.add(layers.Conv2D(filters=32, kernel_size=(2, 2), strides=(1,1), padding='same'))
    AlexNet.add(layers.BatchNormalization())
    AlexNet.add(layers.LeakyReLU())
    AlexNet.add(layers.MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='same'))
    AlexNet.add(layers.Dropout(0.2))
```

Third Layer

```
[ ] AlexNet.add(layers.Conv2D(filters=64, kernel_size=(2,2), strides=(1,1), padding='same'))
    AlexNet.add(layers.BatchNormalization())
    AlexNet.add(layers.LeakyReLU())
```

Fourth Layer

```
[ ] AlexNet.add(layers.Conv2D(filters=64, kernel_size=(2,2), strides=(1,1), padding='same'))
    AlexNet.add(layers.BatchNormalization())
    AlexNet.add(layers.LeakyReLU())
```

Fifth layer

```
[ ] AlexNet.add(layers.Conv2D(filters=64, kernel_size=(1,1), strides=(1,1), padding='same'))
    AlexNet.add(layers.BatchNormalization())
    AlexNet.add(layers.LeakyReLU())
```

```
▶ AlexNet.add(layers.Conv2D(filters=2, kernel_size=(1,1)))
```

```
[ ] AlexNet.add(layers.GlobalAveragePooling2D())
```

```
[ ] AlexNet.add(layers.Dense(units = 4, activation='softmax'))
```

```
[ ] AlexNet.summary()
```

Fig 10. Multiple layers in AlexNet

+ Code + Text

```

conv2d_3 (Conv2D)          (None, 2, 2, 64)          16448
batch_normalization_3 (Batc (None, 2, 2, 64)          256
hNormalization)

leaky_re_lu_3 (LeakyReLU)   (None, 2, 2, 64)          0

conv2d_4 (Conv2D)          (None, 2, 2, 64)          4160

batch_normalization_4 (Batc (None, 2, 2, 64)          256
hNormalization)

leaky_re_lu_4 (LeakyReLU)   (None, 2, 2, 64)          0

conv2d_5 (Conv2D)          (None, 2, 2, 2)           130

global_average_pooling2d (G (None, 2)                  0
lobalAveragePooling2D)

dense (Dense)              (None, 4)                  12

=====
Total params: 34,574
Trainable params: 34,062
Non-trainable params: 512

```

Copy of MP1.ipynb ☆

e Edit View Insert Runtime Tools Help Last edited on February 18

Comment

Share

⚙



ode + Text

Connect ▾

```

Epoch 1/20
31/31 [=====] - 7s 183ms/step - loss: 1.3051 - accuracy: 0.3866 - auc: 0.6410 - precision: 0.2683 - val_loss: 1.3879 - val_acc
Epoch 2/20
31/31 [=====] - 5s 158ms/step - loss: 1.2254 - accuracy: 0.5090 - auc: 0.7125 - precision: 0.6056 - val_loss: 1.3823 - val_acc
Epoch 3/20
31/31 [=====] - 5s 156ms/step - loss: 1.1284 - accuracy: 0.5498 - auc: 0.7674 - precision: 0.7278 - val_loss: 1.3949 - val_acc
Epoch 4/20
31/31 [=====] - 6s 193ms/step - loss: 1.0951 - accuracy: 0.5449 - auc: 0.7802 - precision: 0.7098 - val_loss: 1.4381 - val_acc
Epoch 5/20
31/31 [=====] - 5s 178ms/step - loss: 1.0038 - accuracy: 0.5954 - auc: 0.8253 - precision: 0.7293 - val_loss: 1.5671 - val_acc
Epoch 6/20
31/31 [=====] - 6s 176ms/step - loss: 0.9835 - accuracy: 0.5873 - auc: 0.8262 - precision: 0.6863 - val_loss: 1.5755 - val_acc
Epoch 7/20
31/31 [=====] - 6s 202ms/step - loss: 0.9473 - accuracy: 0.6150 - auc: 0.8433 - precision: 0.6957 - val_loss: 1.6436 - val_acc
Epoch 8/20
31/31 [=====] - 5s 155ms/step - loss: 0.9220 - accuracy: 0.6232 - auc: 0.8569 - precision: 0.7365 - val_loss: 1.8468 - val_acc
Epoch 9/20
31/31 [=====] - 5s 155ms/step - loss: 0.9323 - accuracy: 0.6101 - auc: 0.8502 - precision: 0.6916 - val_loss: 1.6827 - val_acc
Epoch 10/20
31/31 [=====] - 7s 221ms/step - loss: 0.8671 - accuracy: 0.6036 - auc: 0.8662 - precision: 0.7190 - val_loss: 1.7812 - val_acc
Epoch 11/20
31/31 [=====] - 5s 176ms/step - loss: 0.8492 - accuracy: 0.6117 - auc: 0.8727 - precision: 0.7099 - val_loss: 1.7256 - val_acc
Epoch 12/20
31/31 [=====] - 5s 175ms/step - loss: 0.8790 - accuracy: 0.6150 - auc: 0.8643 - precision: 0.7066 - val_loss: 1.6620 - val_acc
Epoch 13/20
31/31 [=====] - 5s 152ms/step - loss: 0.8427 - accuracy: 0.6052 - auc: 0.8718 - precision: 0.7544 - val_loss: 1.5444 - val_acc

Epoch 14/20
31/31 [=====] - 5s 152ms/step - loss: 0.8427 - accuracy: 0.6052 - auc: 0.8718 - precision: 0.7544 - val_loss: 1.5444 - val_acc
Epoch 14/20
31/31 [=====] - 6s 207ms/step - loss: 0.8542 - accuracy: 0.6117 - auc: 0.8697 - precision: 0.7190 - val_loss: 1.4383 - val_acc
Epoch 15/20
31/31 [=====] - 5s 152ms/step - loss: 0.7993 - accuracy: 0.6215 - auc: 0.8835 - precision: 0.7612 - val_loss: 1.2838 - val_acc
Epoch 16/20
31/31 [=====] - 5s 155ms/step - loss: 0.7649 - accuracy: 0.6525 - auc: 0.8942 - precision: 0.7312 - val_loss: 1.1730 - val_acc
Epoch 17/20
31/31 [=====] - 6s 183ms/step - loss: 0.7821 - accuracy: 0.6134 - auc: 0.8861 - precision: 0.7347 - val_loss: 1.3318 - val_acc
Epoch 18/20
31/31 [=====] - 6s 192ms/step - loss: 0.7840 - accuracy: 0.6395 - auc: 0.8872 - precision: 0.7244 - val_loss: 1.1401 - val_acc
Epoch 19/20
31/31 [=====] - 5s 158ms/step - loss: 0.7641 - accuracy: 0.6427 - auc: 0.8934 - precision: 0.7700 - val_loss: 1.2728 - val_acc
Epoch 20/20
31/31 [=====] - 6s 191ms/step - loss: 0.7697 - accuracy: 0.6542 - auc: 0.8961 - precision: 0.7428 - val_loss: 1.2686 - val_acc

```

Fig 11. Training data history (AlexNet)

PERFORMANCE:

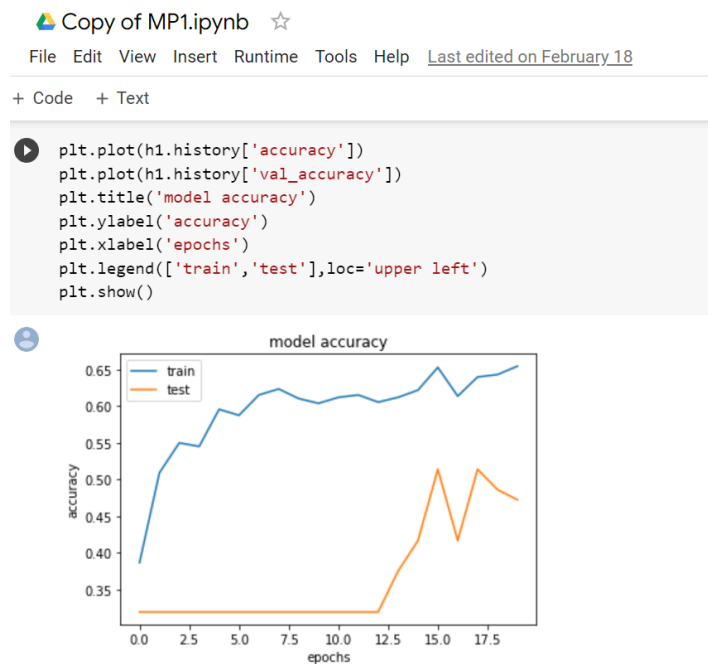


Fig 12. Training accuracy Vs Testing accuracy for AlexNet

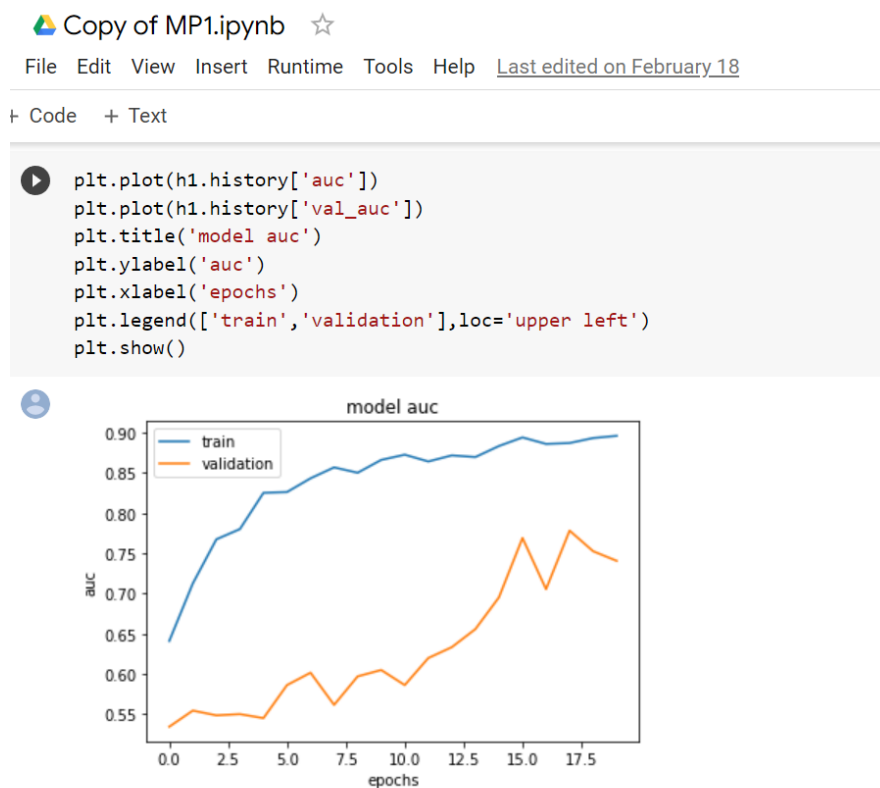


Fig 13. Training accuracy Vs Validation accuracy for AlexNet

Code + Text

```

plt.plot(h1.history['precision'])
plt.plot(h1.history['val_precision'])
plt.title('model precision')
plt.ylabel('precision')
plt.xlabel('epochs')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()

```

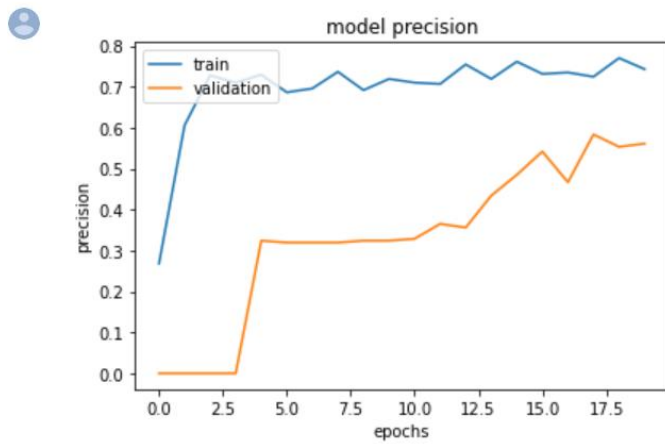


Fig 14. Training Precision Vs Validation Precision for AlexNet

3. CLAHE

CLAHE stands for Contrast Limited Adaptive Histogram Equalization. It is a computer image processing technique used to enhance the contrast of an image, especially when the image has poor contrast or when certain areas of the image are too bright or too dark.

Histogram equalization is a technique that redistributes the pixel values in an image so that the image has a more uniform distribution of intensities. However, it can sometimes produce unrealistic results, such as making the image appear too bright or too dark. CLAHE improves upon histogram equalization by limiting the contrast enhancement to a local region, rather than applying it globally across the entire image. This avoids over-enhancement in regions with high contrast, while still improving the contrast in regions with low contrast.

CLAHE works by dividing the image into small, overlapping regions, called tiles. A histogram equalization algorithm is then applied to each tile individually. The histogram equalization algorithm stretches the histogram of each tile to occupy the full intensity

range, while limiting the maximum slope of the histogram, which helps prevent over-enhancement. After equalization, the tiles are recombined to form the final image.

CLAHE is commonly used in medical imaging, such as X-rays, CT scans, and MRI, as well as in satellite imaging, astronomy, and other areas where contrast enhancement is necessary.

AFTER APPLYING CLAHE ON THE DATASET IMAGES

MP1.ipynb ☆

File Edit View Insert Runtime Tools Help Last saved at 12:02 PM

+ Code + Text

Applying CLAHE

▶ drive.mount('/content/drive')

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

[] TRAIN_DATA = '/content/drive/MyDrive/Minor_Project/dataset_mri/Training'

VAL = '/content/drive/MyDrive/Minor_Project/dataset_mri/Testing'

[] G_TRAIN = TRAIN_DATA+'/glioma_tumor'

N_TRAIN = TRAIN_DATA+'/no_tumor'

M_TRAIN = TRAIN_DATA+'/meningioma_tumor'

P_TRAIN = TRAIN_DATA+'/pituitary_tumor'

[] t_img = glob.glob(G_TRAIN+'/*')

[] clahe_img = []

[] import cv2

count = 0

import cv2

[] count = 0

clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))

for i in t_img:

img = cv2.imread(i,cv2.IMREAD_GRAYSCALE)

img_clahe = clahe.apply(img)

cv2.imwrite('/content/drive/MyDrive/Minor_Project/dataset_mri/CLAHE/clahe_g/g'+str(count)+'.jpg',img_clahe)

count+=1

[] m_img = glob.glob(M_TRAIN+'/*')

n_img = glob.glob(N_TRAIN+'/*')

p_img = glob.glob(P_TRAIN+'/*')

▶ import cv2

count = 0

clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))

for i in m_img:

img = cv2.imread(i,cv2.IMREAD_GRAYSCALE)

img_clahe = clahe.apply(img)

cv2.imwrite('/content/drive/MyDrive/Minor_Project/dataset_mri/CLAHE/clahe_m/m'+str(count)+'.jpg',img_clahe)

count+=1

```
[ ] import cv2
count = 0
clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
for i in n_img:
    img = cv2.imread(i,cv2.IMREAD_GRAYSCALE)
    img_clahe = clahe.apply(img)
    cv2.imwrite('/content/drive/MyDrive/Minor_Project/dataset_mri/CLAHE/clahe_n/n'+str(count)+'.jpg',img_clahe)
    count+=1
```

```
[ ] import cv2
count = 0
clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
for i in p_img:
    img = cv2.imread(i,cv2.IMREAD_GRAYSCALE)
    img_clahe = clahe.apply(img)
    cv2.imwrite('/content/drive/MyDrive/Minor_Project/dataset_mri/CLAHE/clahe_p/p'+str(count)+'.jpg',img_clahe)
    count+=1
```

Fig 15. Implementation of CLAHE

PERFORMANCE

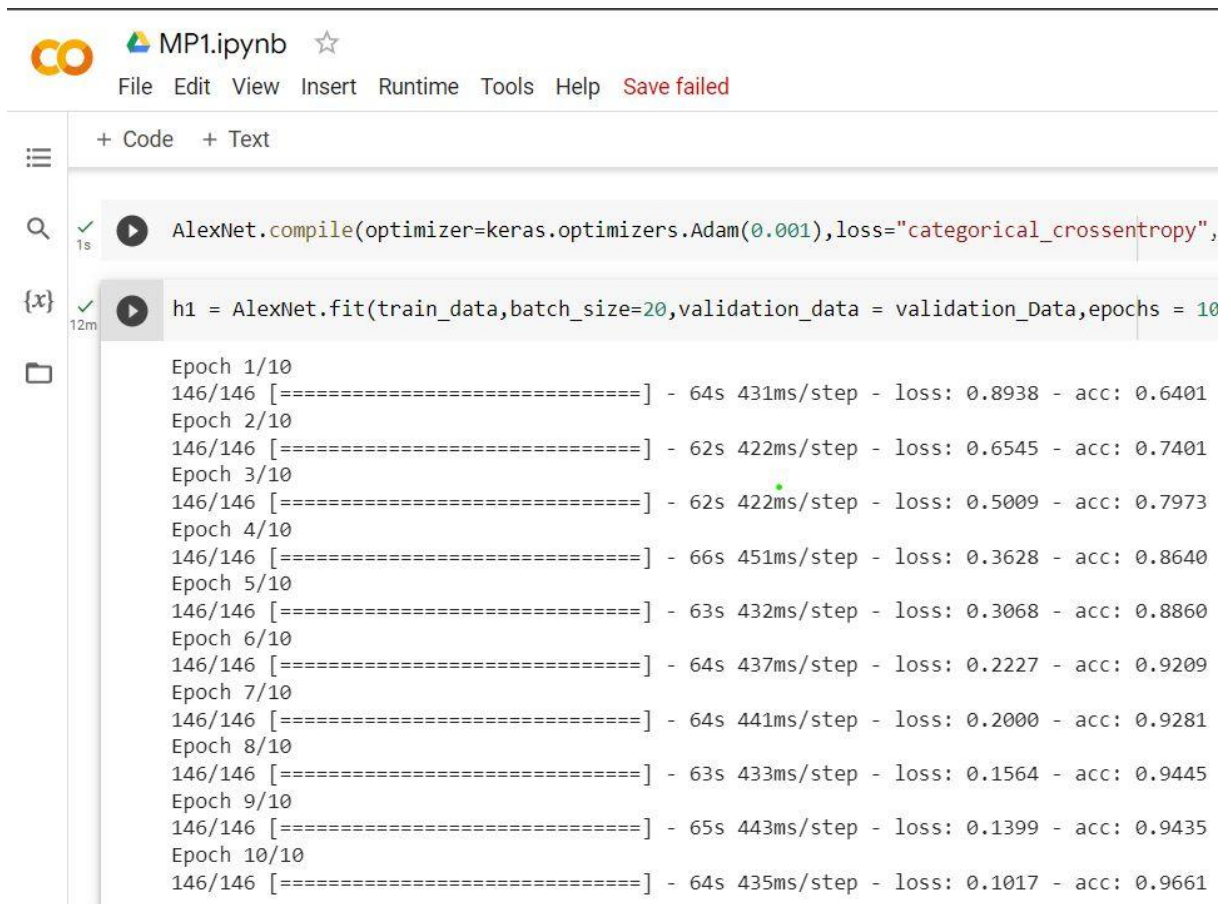
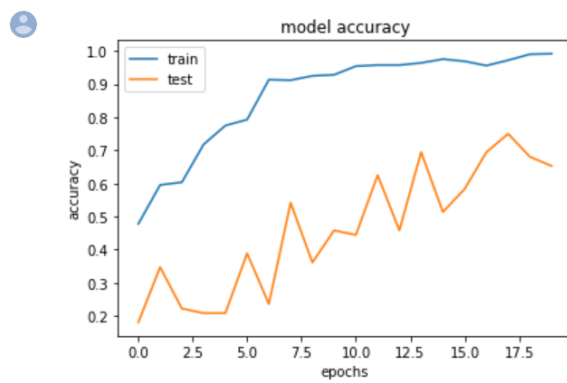


Fig 16. Training data history after Contrast enhancement

```
plt.plot(h1.history['acc'])
plt.plot(h1.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epochs')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



RESULTS/ CONCLUSION OBTAINED

In recent years, deep learning has emerged as a powerful tool for medical image analysis, particularly in the detection of brain tumors. In this study, we employed two state-of-the-art deep learning models, InceptionResNetV2 and EfficientNet, to detect brain tumors in MRI scans. However, the initial results of our study were not satisfactory.. We then implemented an AlexNet-like model, which improved the accuracy to 0.6542.

Despite the improved accuracy, we were still not satisfied with the results and looked for ways to further enhance the performance of our model. We decided to apply the CLAHE algorithm to the MRI dataset to improve the contrast and visibility of the brain images. CLAHE works by dividing the image into small, overlapping regions, called tiles, and applying a histogram equalization algorithm to each tile individually. The tiles are then recombined to form the final image, with enhanced contrast and improved visibility.

After applying the CLAHE algorithm to the MRI dataset, we retrained our AlexNet-like model and observed a significant improvement in the accuracy of our model, achieving an accuracy of 0.9672. This result is a remarkable improvement over the initial accuracy and demonstrates the effectiveness of the CLAHE algorithm in enhancing the performance of deep learning models in medical image analysis.

In conclusion, the results of our study highlight the potential of deep learning models in the detection of brain tumors and the importance of pre-processing techniques like CLAHE in improving the accuracy of these models. This finding has significant implications for the diagnosis and treatment of brain tumors, as early and accurate detection can lead to more effective treatment and improved patient outcomes.

FUTURE SCOPE

It is observed on examination that the proposed approach needs a vast training set for better accurate results; in the field of medical image processing, the gathering of medical data is a tedious job, and, in few cases, the datasets might not be available. In all such cases, the proposed algorithm must be robust enough for accurate recognition of tumor regions from MR Images. The proposed approach can be further improvised through in cooperating weakly trained algorithms that can identify the abnormalities with a minimum training data and also self-learning algorithms would aid in enhancing the accuracy of the algorithm and reduce the computational time.

BIBLIOGRAPHY

- [1] Anand Deshpande, Vania V. Estrela , Prashant Patavardhan, ‘The DCT-CNN-ResNet50 architecture to classify brain tumors with super-resolution, convolutional neural network, and the ResNet50’. Link: <https://www.sciencedirect.com/science/article/pii/S2772528621000133>
- [2] Haritha Sathyan, Vinitha Panicker J, ‘Lung Nodule Classification Using Deep ConvNets On CT Images’. Link: <https://ieeexplore.ieee.org/document/8494084>
- [3] Muhammad Imran Razzak, Saeeda Naz and Ahmad Zaib, ‘Deep Learning for Medical Image Processing: Overview, Challenges and the Future’. Link: https://link.springer.com/chapter/10.1007/978-3-319-65981-7_12
- [4] Ahmet Çinar, Muhammed Yildirim, ‘Detection of tumors on brain MRI images using the hybrid convolutional neural network architecture’. Link: <https://www.sciencedirect.com/science/article/abs/pii/S0306987720301717>
- [5] Hong Liu, Haichao Cao, Enmin Song, Guangzhi Ma, Xiangyang Xu, Renchao Jin, Chuhua Liu & Chih-Cheng Hung , ‘Multi-model Ensemble Learning Architecture Based on 3D CNN for Lung Nodule Malignancy Suspiciousness Classification’. Link: <https://link.springer.com/article/10.1007/s10278-020-00372-8>
- [6] Hanaa ZainEldin , Samah A. Gamel, El-Sayed M. El-Kenawy , Amal H. Alharbi, Doaa Sami Khafaga , *Abdelhameed Ibrahim* , and Fatma M. Talaat, ‘Brain Tumor Detection and Classification Using Deep Learning and Sine-Cosine Fitness Grey Wolf Optimization’. Link: https://www.researchgate.net/publication/366500150_Brain_Tumor_Detection_and_Classification_Using_Deep_Learning_and_Sine-Cosine_Fitness_Grey_Wolf_Optimization
- [7] Hitesh Tekchandani, Shrish Verma, Narendra D. Londhe, ‘Mediastinal lymph node malignancy detection in computed tomography images using fully convolutional network’. Link: <https://www.sciencedirect.com/science/article/abs/pii/S0208521618305588>
- [8] Shi, Z., He, L., Suzuki, K., Nakamura, T., & Itoh, H. (2009). ‘Survey on Neural Networks Used for Medical Image Processing. International Journal of computational science’. Link: <https://pubmed.ncbi.nlm.nih.gov/26740861/>
- [9] <https://towardsdatascience.com/alexnet-the-architecture-that-challenged-cnns-e406d5297951>
- [10] <https://www.analyticsvidhya.com/blog/2022/08/image-contrast-enhancement-using-clahe/>
- [11] https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/Dir

WORK DONE BY INDIVIDUALS

Here is the detailed breakdown of Shagun Singh's and Samyak Jain's work for the project on brain tumor detection using deep learning:

Shagun Singh (209303065)

- Extensive research: performed in-depth research on the topic of brain tumor detection using deep learning. This involved reading academic papers, reviewing previous research work, and understanding the various techniques and algorithms used in this field.
- Literature review: Based on the research work, prepared a comprehensive literature review for the project. This review included an overview of the current state-of-the-art methods and techniques for brain tumor detection using deep learning.
- InceptionResNetV2 model implementation: implemented the InceptionResNetV2 model for the project. This model is a deep convolutional neural network architecture that is commonly used for image classification tasks. Fine-tuning the pre-trained InceptionResNetV2 model was done by using the dataset of brain tumor images to achieve better performance.
- Project report and PPT: created the project report and presentation slides for the project. The report included details about the research work, methodology, implementation, and results obtained from the project.

Samyak Jain (209303064)

- Extensive research: performed extensive research on the topic of brain tumor detection using deep learning. This involved studying academic papers and reviewing previous research work related to the topic.
- AlexNet model implementation: implemented the AlexNet model for the project. This model is another deep convolutional neural network architecture that is commonly used for image classification tasks. Fine-tuning the pre-trained AlexNet model was done by using the dataset of brain tumor images to achieve better performance.
- CLAHE implementation: implemented a noise reduction model called Contrast Limited Adaptive Histogram Equalization (CLAHE) for the project. This model is used to enhance the contrast and details of an image by improving its brightness and contrast. Applied CLAHE to the brain tumor images to improve the quality of the images before feeding them into the neural network models.
- Project report and PPT: Samyak also contributed to the project report and presentation slides by providing details about his research work, methodology, implementation, and results obtained from the project.