

**Automatic brain tumor detection and  
classification on MRI images using  
machine learning techniques.**

A Project report submitted in partial fulfillment of the requirements  
for the degree of B. Tech Electrical Engineering

*By*

**Subhranil Thakur (11701617026)**

**Supratik Ghosh (11701617024)**

**Ranjabati Roy (11701617047)**

**Anish Chakraborty (11701617074)**

*Under the supervision of*

**Prof.(Dr.) ALOK KOLE**

**DEPARTMENT OF ELECTRICAL ENGINEERING**



**RCC INSTITUTE OF INFORMATION  
TECHNOLOGY**

CANAL SOUTH ROAD, BELIAGHATA, KOLKATA - 700015, WEST BENGAL  
Maulana Abul Kalam Azad University of Technology (MAKAUT)  
©2021

**Department of Electrical Engineering**

**RCC INSTITUTE OF INFORMATION TECHNOLOGY**

GROUND FLOOR, NEWBUILDING, CANAL SOUTH ROAD, BELIAGHATA,

KOLKATA –700015, WEST BENGAL

PHONE: 033-2323-2463-154, FAX: 033-2323-4668

Email: [hodeercciit@rcciit.org.in](mailto:hodeercciit@rcciit.org.in), Website: <http://www.rcciit.org/academic/ee.aspx>



## **CERTIFICATE**

### **To whom it may concern**

This is to certify that the project work entitled **AUTOMATIC BRAIN TUMOR DETECTION AND CLASSIFICATION ON MRI IMAGES USING MACHINE LEARNING TECHNIQUES** is the bona fide work carried out by **Subhranil Thakur (11701617026)**, **Supratik Ghosh (11701617024)**, **Ranjabati Roy (11701617047)** and **Anish Chakraborty (11701617074)**, students of B.Tech in the Dept. of Electrical Engineering, RCC Institute of Information Technology (RCCIIT), Canal South Road, Beliaghata, Kolkata-700015, affiliated to Maulana Abul Kalam Azad University of Technology (MAKAUT), West Bengal, India, during the academic year **2020-21**, in partial fulfillment of the requirements for the degree of Bachelor of Technology in Electrical Engineering and that this project has not submitted previously for the award of any other degree, diploma and fellowship.

Signature of the Guide.

Name: Prof. (Dr.) Alok Kole

Designation: Professor

HoD, Dept. of EE  
RCC Institute of Information Technology  
Kolkata-700015

Signature of the HOD.

Name: Dr. Debasish Mondal

Designation: Associate Professor

Signature of the External Examiner.

Name:

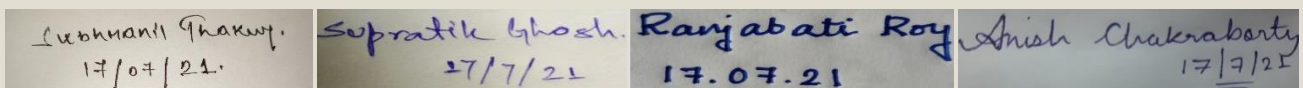
Designation:

## **ACKNOWLEDGEMENT**

It is our great fortune that we have got the opportunity to carry out this project work under the supervision of Prof. (Dr.) Alok Kole in the Department of Electrical Engineering, RCC Institute of Information Technology (RCCIIT), Canal South Road, Beliaghata, Kolkata- 700015, affiliated to Maulana Abul Kalam Azad University of Technology (MAKAUT), West Bengal, India. We express our sincere thanks and deepest sense of gratitude to our guide for his constant support, unparalleled guidance and limitless encouragement.

We wish to convey our gratitude to Prof. (Dr.) Debasish Mondal, HOD, Department of Electrical Engineering, and RCCIIT and to the authority of RCCIIT for providing all kinds of infrastructural facility towards the research work.

We would also like to convey our gratitude to all the faculty members and staffs of the Department of Electrical Engineering, RCCIIT for their whole hearted cooperation to make this work turn into reality.



The image shows four separate rectangular boxes, each containing a handwritten signature and a date. From left to right: 1. Signature: 'Sourabjit Ghosh', Date: '17/07/21'. 2. Signature: 'Supratik Ghosh', Date: '27/7/21'. 3. Signature: 'Ranjabati Roy', Date: '17.07.21'. 4. Signature: 'Anish Chakrabarty', Date: '17/7/21'.

**Full Signature of the Student(s)**

**Place: Kolkata**

**Date :- 07/07/2021**

# **CONTENTS**

S No.	TOPIC	PAGE No.
1.	ABSTRACT	5
2.	INTRODUCTION	6-7
3.	LITERATUREREVIEW	8-9
4.	WORKING THEORY OF OUR PROJECT	10-18
5.	IMPLEMENTATION METHODOLOGY	19-24
6.	FLOWCHART FOR DESIGN AND DEVELOPMENT OF PROPOSED PROJECT	25
7.	PYTHON PROGRAM FOR PROPOSED PROJECT	26-36
8.	PERFORMANCE ANALYSIS OF THE PROPOSED MODEL	37-43
9.	CONCLUSION	44
10.	FUTURE SCOPE	45
11.	BIBLIOGRAPHY	46-47
12.	REFERENCES	48

## ABSTRACT

Automated defect detection in medical imaging has become the emergent field in several medical diagnostic applications. Automated detection of tumor in MRI is very crucial as it provides information about abnormal tissues which is necessary for planning treatment. The conventional method for defect detection in magnetic resonance brain images is human inspection. This method is impractical due to large amount of data. Hence, trusted and automatic classification schemes are essential to prevent the death rate of human. So, automated tumor detection methods are developed as it would save radiologist time and obtain a tested accuracy. The MRI brain tumor detection is complicated task due to complexity and variance of tumors. In this project, it is proposed with machine learning algorithms to overcome the drawbacks of traditional classifiers where tumor is detected in brain MRI using machine learning algorithms. Machine learning and image classifier can be used to efficiently detect cancer cells in brain through MRI.

## **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 visualization 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, ResNet34 and Inception v3) using Convolutional Neural Network (CNN) architecture.

# LITERATURE-REVIEW

Krizhevsky et al. 2012 achieved state-of-the-art results in image classification based on transfer learning solutions upon training a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, he achieved top-1 and top-5 error rates of 37.5% and 17.0% which was considerably better than the previous state-of-the-art. He also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry. The neural network, which had 60 million parameters and 650,000 neurons, consisted of five convolutional layers, some of which were followed by max-pooling layers, and three fully-connected layers with a final 1000-way Softmax. To make training faster, he used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers he employed a recently-developed regularization method called -dropout that proved to be very effective.

Simonyan & Zisserman 2014 they investigated the effect of the convolutional network depth on its accuracy in the large-scale image recognition setting. These findings were the basis of their ImageNet Challenge 2014 submission, where their team secured the first and the second places in the localization and classification tracks respectively. Their main contribution was a thorough evaluation of networks of increasing depth using architecture with very small (3×3) convolution filters, which shows that a significant improvement on the prior-art configurations can be achieved by pushing the depth to 16–19 weight layers after training smaller versions of VGG with less weight layers.

Pan & Yang 2010's survey focused on categorizing and reviewing the current progress on transfer learning for classification, regression and clustering problems. In this survey, they discussed the relationship between transfer learning and other related machine learning techniques such as domain adaptation, multitask learning and sample selection bias, as well as co-variate shift. They also explored some potential future issues in transfer learning research. In this survey article, they reviewed several current trends of transfer learning.

Szegedy et al. 2015 proposed a deep convolutional neural network architecture codenamed

Inception, which was responsible for setting the new state of the art for classification and detection in the ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC14). The main hallmark of this architecture is the improved utilization of the computing resources inside the network. This was achieved by a carefully crafted design that allows for increasing the depth and width of the network while keeping the computational budget constant. His results seem to yield solid evidence that approximating the expected optimal sparse structure by readily available dense building blocks is a viable method for improving neural networks for computer vision.

He et al., 2015b introduced the ResNet, which utilizes skip connections and batch normalization. He presented a residual learning framework to ease the training of networks that are substantially deeper than those used previously. He explicitly reformulated the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. He provided comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth. On the ImageNet dataset he evaluated residual nets with a depth of up to 152 layers—8×deeper than VGG nets but still having lower complexity. An ensemble of these residual nets achieves 3.57% error on the ImageNet test set. This result won the 1st place on the ILSVRC 2015 classification task. He also presented analysis on CIFAR-10 with 100 and 1000 layers.

Ref. [22] reports the accuracy achieved by seven standard classifiers, viz. i) Adaptive Neuro-Fuzzy Classifier (ANFC), ii) Naive Bayes (NB), iii) Logistic Regression (LR), iv) Multilayer Perceptron (MLP), v) Support Vector Machine (SVM), vi) Classification and Regression Tree (CART), and vii) k-nearest neighbors (k-NN). The accuracy reported in Ref. [17] is on the BRaTS 2015 dataset (a subset of BRaTS 2017 dataset) which consists of 200 HGG and 54 LGG cases. 56 three-dimensional quantitative MRI features extracted manually from each patient MRI and used for the classification.



# WORKING THEORY

## MACHINE LEARNING:

Machine Learning (ML) is a subset of AI which is programmed to think on its own, perform social interaction, learn new information from the provided data and adapt as well as improve with experience. Although training time via Deep Learning (DL) methods is more than Machine Learning methods, it is compensated by higher accuracy in the former case. Also, DL being automatic, large domain knowledge is not required for obtaining desired results unlike in ML.

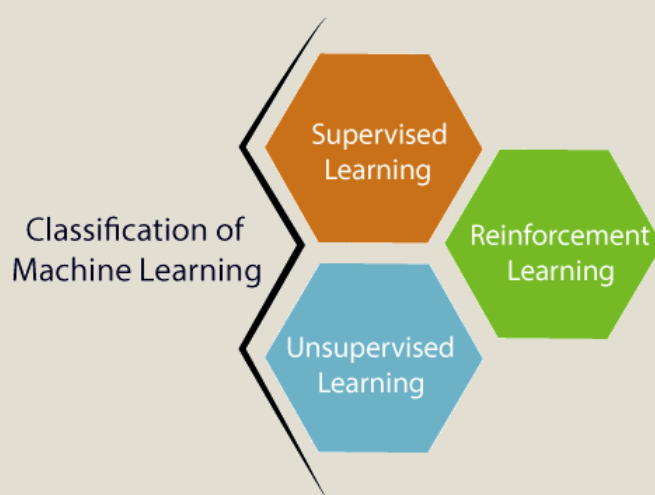


Fig 1: A diagram showing the classification of Machine learning

## BRAIN TUMOR:

In medical science, an anomalous and uncontrollable cell growth inside the brain is recognized as tumor. Human brain is the most receptive part of the body. It controls muscle movements and interpretation of sensory information like sight, sound, touch, taste, pain, etc.

The human brain consists of Grey Matter (GM), White Matter (WM) and Cerebrospinal Fluid (CSF) and on the basis of factors like quantification of tissues, location of abnormalities, malfunctions & pathologies and diagnostic radiology, a presence of tumor is identified. A tumor in the brain can affect such sensory information and muscle movements or even results in more dangerous situation which includes death. Depending upon the place of commencing, tumor can

be categorized into primary tumors and secondary tumors. If the tumor is originated inside the skull, then the tumor is known as primary brain tumor otherwise if the tumor's initiation place is Somewhere else in the body and moved towards the brain, then such tumors are called secondary tumors. Brain tumor can be of the following types-glioblastoma, sarcoma, metastatic bronchogenic carcinoma on the basis of axial plane. While some tumors such as meningioma can be easily segmented, others like gliomas and glioblastomas are much more difficult to localize. World Health Organization (WHO) categorized gliomas into - HGG/high grade glioma/glioblastoma/IV stage /malignant & LGG/low grade glioma/II and III stage /benign. Although most of the LGG tumors have slower growth rate compared to HGG and are responsive to treatment, there is a subgroup of LGG tumors which if not diagnosed earlier and left untreated could lead to GBM. In both cases a correct treatment planning (including surgery, radiotherapy, and chemotherapy separately or in combination) becomes necessary, considering that an early and proper detection of the tumor grade can lead to a good prognosis. Survival time for a GBM (Glioblastoma Multiform) or HGG patient is very low i.e. in the range of 12 to 15 months. Magnetic Resonance Imaging (MRI) has become the standard non-invasive technique for brain tumor diagnosis over the last few decades, due to its improved soft tissue contrast that does not use harmful radiations unlike other methods like CT(Computed Tomography), X-ray, PET (Position Emission Tomography) scans etc. The MRI image is basically a matrix of pixels having characteristic features.

Since glioblastomas are infiltrative tumors, their borders are often fuzzy and hard to distinguish from healthy tissues. As a solution, more than one MRI modality is often employed e.g. T1 (spin-lattice relaxation), T1-contrasted (T1c), T2 (spin-spin relaxation), proton density (PD) contrast imaging, diffusion MRI (dMRI), and fluid attenuation inversion recovery (FLAIR) pulse sequences. T1-weighted images with intravenous contrast highlight the most vascular regions of the tumor (T1c gives much more accuracy than T1.), called Enhancing tumor'(ET), along with the tumor core' (TC) that does not involve peritumoral edema. T2-weighted (T2W) and T2W-Fluid Attenuation Inversion Recovery (FLAIR) images are used to evaluate the tumor and peritumoral edema together defined as the whole tumor (WT). Gliomas and glioblastomas are difficult to distinguish in T1, T1c, T2 and PD. They are better identified in FLAIR modalities.

We have attempted to separate the brain tumor into following types-necrosis (1), edema (2), non- enhancing (malignant) (3) and enhancing (benign) (4) tumor. MRI images can be of three types on the basis of position from which they are taken which are Sagittal (side), Coronal (back) and Axial (top). We have used sagittal images in our project.

Process of brain tumor segmentation can be manual selection of ROI, Semi-automatic and fully-automatic. Popular machine learning algorithms for classification of brain tumor are Artificial Neural Network, Convolutional Neural Network, k-Nearest Neighbor (kNN), Decision Tree, Support Vector Machine (SVM), Naïve Bayes and Random Forest (RF). Here, we are using Convolutional Neural Network (CNN) for the detection and classification of the brain tumor.

### **BASIC OPERATION OF NEURAL NETWORKS:**

Neural Networks (NN) form the base of deep learning, a subfield of machine learning where the algorithms are inspired by the structure of the human brain. NN take in data, train themselves to recognize the patterns in this data and then predict the outputs for a new set of similar data. NN are made up of layers of neurons. These neurons are the core processing units of the network. First we have the input layer which receives the input; the output layer predicts our final output. In between, exist the hidden layers which perform most of the computations required by our network.

Our brain tumor images are composed of 128 by 128 pixels which make up for 16,384 pixels. Each pixel is fed as input to each neuron of the first layer. Neurons of one layer are connected to neurons of the next layer through channels. Each of these channels is assigned a numerical value known as weight'. The inputs are multiplied to the corresponding weight and their sum is sent as input to the neurons in the hidden layer. Each of these neurons is associated with a numerical value called the bias'which is then added to the input sum. This value is then passed through a threshold function called the activation function'. The result of the activation function determines if the particular neuron will get activated or not. An activated neuron transmits data to the neurons of the next layer over the channels. In this manner the data is propagated through the network this is called forward propagation'. In the output layer the neuron with the highest value fires and determines the output. The values are basically a probability. The predicted output is compared against the actual output to realize the error in prediction. The magnitude of the

Error gives an indication of the direction and magnitude of change to reduce the error. This information is then transferred backward through our network. This is known as back propagation'. Now based on this information the weights are adjusted. This cycle of forward propagation and back propagation is iteratively performed with multiple inputs. This process continues until our weights are assigned such that the network can predict the type of tumor correctly in most of the cases. This brings our training process to an end. NN may take hours or even months to train but time is a reasonable trade-off when compared to its scope Several experiments show that after pre-processing MRI images, neural network classification algorithm was the best more specifically CNN(Convolutional Neural Network) as compared to Support Vector Machine(SVM),Random Forest Field.

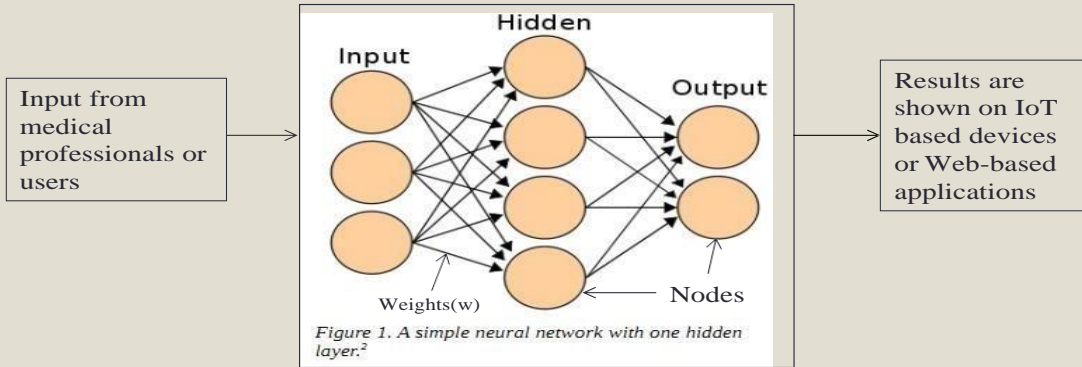
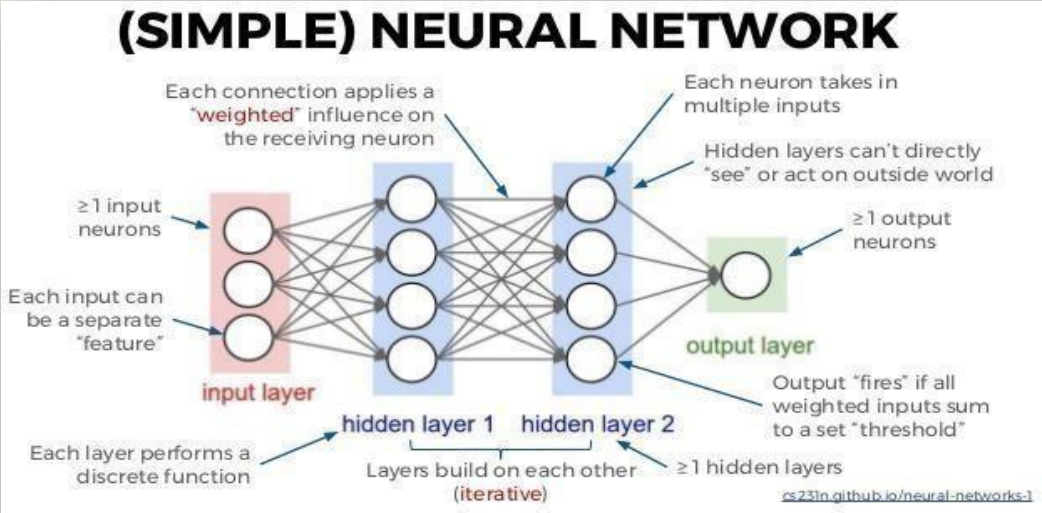


Fig 2: A multi-layer perceptron model of neural network

## **TRANSFER LEARNING:**

A major assumption in many machine learning and data mining algorithms is that the training and future data must be in the same feature space and have the same distribution. However, in many real-world applications, this assumption may not hold. For example, we sometimes have a classification task in one domain of interest, but we only have sufficient training data in another domain of interest, where the latter data may be in a different feature space or follow a different data distribution. In such cases, knowledge transfer, if done successfully, would greatly improve the performance of learning by avoiding much expensive data labelling efforts. In recent years, transfer learning has emerged as a new learning framework to address this problem.

Transfer learning allows neural networks using significantly less data. With transfer learning, we are in effect transferring the knowledge that a model has learned from a previous task, to our current one. The idea is that the two tasks are not totally disjoint, as such we can leverage whatever network parameters that model has learned through its extensive training, without having to do that training ourselves. Transfer learning has been consistently proven to boost model accuracy and reduce required training time, less data, less time, more accuracy. Transfer learning is classified to three different settings: inductive transfer learning, transductive transfer learning and unsupervised transfer learning. Most previous works focused on the settings. Furthermore, each of the approaches to transfer learning can be classified into four contexts based on -what to transfer in learning. They include the instance-transfer approach, the feature-representation-transfer approach, the parameter transfer approach and the relational- knowledge-transfer approach, respectively.

The smaller networks converged & were then used as initializations for the larger, deeper networks- This process is called pre-training. While making logical sense, pre-training is a very time consuming, tedious task, requiring an entire network to be trained before it can serve as an initialization for a deeper network.

## **ACTIVATION FUNCTION:**

Sigmoid function ranges from 0 to 1 and is used to predict probability as an output in case of binary classification while Softmax function is used for multi-class classification. tanh function ranges from -1 to 1 and is considered better than sigmoid in binary classification using feed forward algorithm. ReLU (Rectified Linear Unit) ranges from 0 to infinity and Leaky ReLU

(Better version of ReLU) ranges- from -infinity to +infinity. ReLU stands for Rectified Linear Unit for a non-linear operation. The output is  $f(x) = \max(0, x)$ . ReLU's purpose is to introduce non-linearity in our ConvNet. Since, the real world data would want our ConvNet to learn would be non-negative linear values. There are other nonlinear functions such as tanh or sigmoid that can also be used instead of ReLU. Most of the data scientists use ReLU since performance wise ReLU is better than the other two. Stride is the number of pixels that would move over the input matrix one at a time.

Sometimes filter does not fit perfectly fit the input image. We have two options: either pad the picture with zeros (zero-padding) so that it fits or drop the part of the image where the filter did not fit. This is called valid padding which keeps only valid part of the image.

### **CONVOLUTIONAL NEURAL NETWORK:**

Classifier models can be basically divided into two categories respectively which are generative models based on hand-crafted features and discriminative models based on traditional learning such as support vector machine (SVM), Random Forest (RF) and Convolutional Neural Network (CNN). One difficulty with methods based on hand-crafted features is that they often require the computation of a large number of features in order to be accurate when used with many traditional machine learning techniques. This can make them slow to compute and expensive memory-wise. More efficient techniques employ lower numbers of features, using dimensionality reduction like PCA (Principle Component Analysis) or feature selection methods, but the reduction in the number of features is often at the cost of reduced accuracy. Brain tumour segmentation employ discriminative models because unlike generative modelling approaches, these approaches exploit little prior knowledge on the brain's anatomy and instead rely mostly on the extraction of [a large number of] low level image features, directly modelling the relationship between these features and the label of a given vowel. In our project, we have used the Convolutional Neural Network architecture for Brain tumor Detection and Classification.

Convolutional neural network processes closely knitted data used for image classification, image processing, face detection etc. It is a specialized 3D structure with specialized NN analyzing RGB layers of an image .Unlike others, it analyses one image at a time Identifies and extracts important features and uses them to classify the image .Convolutional

Neural Networks (ConvNets) automatically learns mid-level and high-level representations or abstractions from the input training data. The main building block used to construct a CNN architecture is the convolutional layer. It also consists of several other layers, some of which are described as below:

- Input Layer-It takes in the raw pixel value of input image
- Convolutional Layer- It is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. It is a mathematical operation that takes two inputs such as image matrix and a filter or kernel to generate a feature map Convolution of an image with different filters can perform operations such as edge detection, blur and sharpen by applying filters.
- Activation Layer-It produces a single output based on the weighted sum of inputs
- Pooling Layer-Pooling layers section would reduce the number of parameters when the images are too large. Spatial pooling (also called subsampling or down sampling) reduces the dimensionality of each map but retains important information. Spatial pooling can be of different types:
  - Max Pooling – taking the largest element in the feature map
  - Average Pooling - taking the average of elements in the feature map
  - Sum Pooling – taking the sum of all elements in the feature map
- Fully Connected Layer-The layer we call as FC layer, we flattened our matrix into vector and feed it into a fully connected layer like a neural network. the feature map matrix will be converted as column vector (x1, x2, x3, ...). With the fully connected layers, we combined these features together to create a model. For classifying input image into various classes based on training set.
- Dropout Layer-It prevents nodes in a network from co-adapting to each other.

## ADVANTAGES-

1. It is considered as the best ml technique for image classification due to high accuracy.
2. Image pre-processing required is much less compared to other algorithms.
3. It is used over feed forward neural networks as it can be trained better in case of complex images to have higher accuracies.
4. It reduces images to a form which is easier to process without losing features which are critical for a good prediction by applying relevant filters and reusability of weights
5. It can automatically learn to perform any task just by going through the training data i.e. there no need for prior knowledge
6. There is no need for specialized hand-crafted image features like that in case of SVM, Random Forest etc.

## DISADVANTAGES-

1. It requires a large training data.
2. It requires appropriate model.
3. It is time consuming.
4. It is a tedious and exhaustive procedure.
5. While convolutional networks have already existed for a long time, their success was limited due to the size of the considered network.

**Solution**-Transfer Learning for inadequate data which will replace the last fully connected layer with pre-trained ConvNet with new fully connected layer.

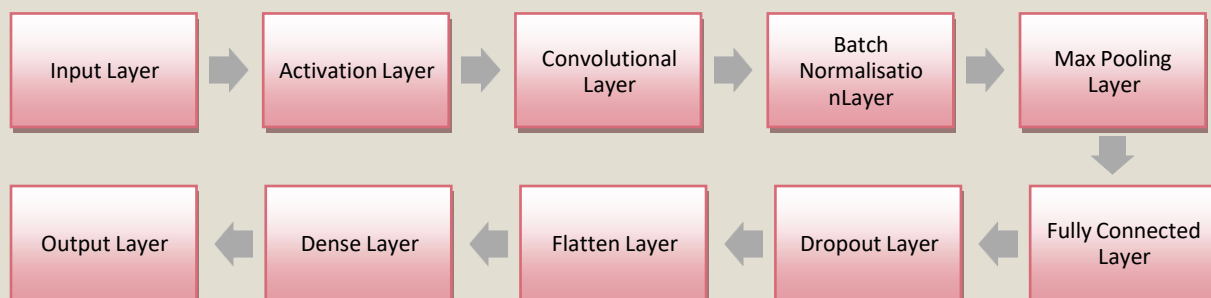


Fig 3: A diagram of a model trained from scratch using CNN architecture.



## EVALUATION METRICS:

- True Positive (TP) is the HGG class predicted in the presence of the LGG class of the glioma. True Negative (TN) is the LGG class predicted in the absence of the HGG class of glioma. False Positive (FP) is prediction of HGG class in the absence of LGG class. False Negative (FN) is prediction of LGG class in the absence of HGG class.
- Accuracy is the most intuitive performance measure. Accuracy is the amount of correctly Prediction made by the total number of predictions made. 
$$\text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN}$$
- Precision is defined as the number of true positives divided by the number of true Positives plus the number of false positives. 
$$\text{Precision} = \frac{TP}{TP+FP}$$
- Recall is also known as sensitivity. It is the fraction of the total amount of relative Relevant instances that were actually retrieved. 
$$\text{Recall} = \frac{TP}{TP+FN}$$
- F 1 Score is the weighted average or the harmonic mean of Precision and Recall taking Both metrics into account in the following equation: 
$$\text{F1 Score} = 2 \times \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$
.When we have an unbalanced dataset F 1 Score favored over accuracy because it takes both false positives and false negatives into account. F-measures are used to balance the

Ratio of false negatives using a weighting parameter (beta) it is given as 
$$F = P * \frac{R^{(1+\beta)^2}}{(P+R)\beta^2}$$

- Other performance metrics used are: sensitivity, specificity and error rate. Sensitivity represents the probability of predicting actual HGG class. Specificity value defines prediction of LGG class. They allow us to determine potential of over- or under-segmentations of the tumor sub-regions. The error rate (ERR) is the amount of predicted class that have been incorrectly classified by a decision model. The overall classification is also provided by the Area under the Curve (AUC) that represents better classification if the area under the curve is more. All of these performances metric is evaluated for FLAIR sequences.
- The DSC (dice similarity co-efficient) measures the overlap between the manual delineated brain tumour regions and the segmentation results of our fully automatic method that is. Mathematically, dice score/DSC is the number of false positives divided

by the number of positives added with the number of false positives. 
$$\text{DSC} = \frac{2TP}{FP+TP+FN}$$

and Dice loss = 
$$2 \frac{|X1 \cap Y1|}{|X1| + |Y1|}$$

## **IMPLEMENTATION METHODOLOGY:**

### **SOFTWARE REQUIREMENTS:**

Python 3 - We have used Python which is a statistical mathematical programming language like R instead of MATLAB due to the following reasons:

1. Python code is more compact and readable than MATLAB
2. The python data structure is superior to MATLAB
3. It is an open source and also provides more graphic packages and data sets

Keras (with Tensor Flow backend 2.3.0 version) - Keras is a neural network API consisting ofTensor Flow, CNTk, Theano etc.

Python packages like Numpy, Matplotlib, and Pandas for mathematical computation and plotting graphs, Simple ITK for reading the images which were in .mha format and Mahotas for feature extraction of GLCM

Kaggle was used to obtain the online dataset. <https://www.kaggle.com/navoneel/brain-mri-images-for-brain-tumor-detection/activity>

GitHub and Stackoverflow was used for reference in case of programming syntax errors.

OpenCV (Open Source Computer Vision) is a library of programming functions aimed at real time computer vision i.e. used for image processing and any operations relating to image like reading and writing images, modifying image quality, removing noise by using Gaussian Blur, performing binary thresholding on images, converting the original image consisting of pixel values into an array, changing the image from RGB to grayscale etc. It is free to use, simple to learn and supports C++, Java, C, Python. Its popular application lies in CamScanner or Instagram, GitHub or a web-based control repository.

Google Colaboratory (open-source Jupyter Notebook interface with high GPU facility) - Google Colab /Colaboratory is a free Jupyter notebook environment that requires no setup and runs entirely on cloud. With Colab, one can write and execute code, save and share analyses, access powerful computing resources, all for free from browser. Jupyter Notebook is a powerful way to iterate and write on your Python code for data analysis. Rather than writing and rewriting an entire code, one can write lines of code and run them at a time. It is built off of iPython which

is an interactive way of running Python code. It allows Jupyter notebook to support multiple languages as well as storing the code and writing own markdown.]

### **HARDWARE REQUIREMENTS:**

Processor: Intel® Core™ i5-10<sup>th</sup> Gen CPU @ 5

GHz Installed memory (RAM): 8.00GB

System Type: 64-bit Operating System

### **IMAGE ACQUISITION:**

### **KAGGLE DATASET:**

The dataset contains 2 folders: yes and no which contains 253 Brain MRI Images. The folder yes contains 155 Brain MRI Images that are tumorous and the folder no contains 98 Brain MRI Images that are non-tumorous. You can find it [here](<https://www.kaggle.com/navoneel/brain-mri-images-for-brain-tumor-detection>).

1. 70% of the data for training.
2. 15% of the data for validation.
3. 15% of the data for testing..

```
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)
```

### **BRAIN TUMOR MRI IMAGES:**

A brain tumor is a mass or growth of abnormal cells in your brain.

Many different types of brain tumors exist. Some brain tumors are noncancerous (benign), and some brain tumors are cancerous (malignant). Brain tumors can begin in your brain (primary brain tumors), or cancer can begin in other parts of your body and spread to your brain as secondary (metastatic) brain tumors.

How quickly a brain tumor grows can vary greatly. The growth rate as well as the location of a brain tumor determines how it will affect the function of your nervous system.

Brain tumor treatment options depend on the type of brain tumor you have, as well as its size and location.

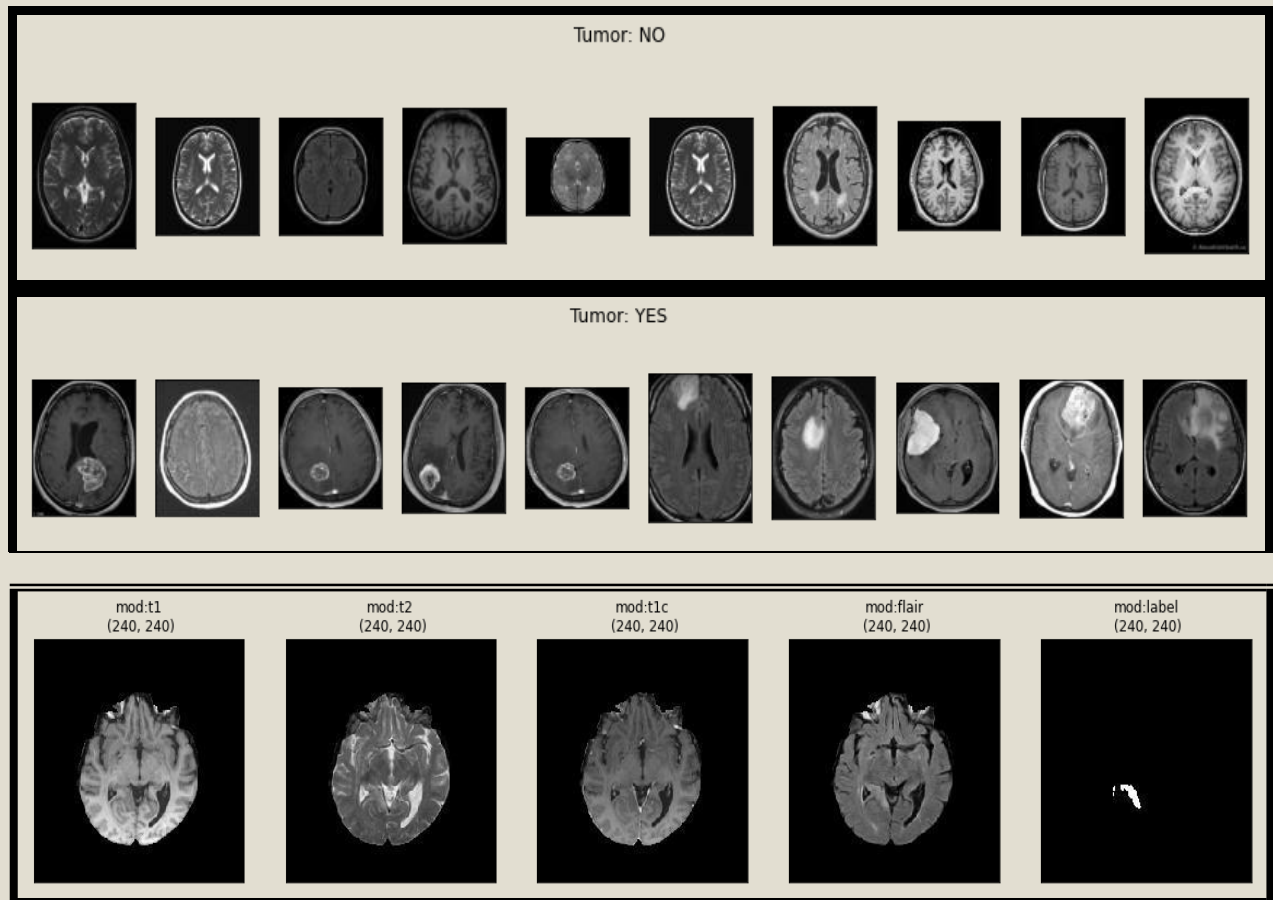


Fig 4: 1. Online Kaggle dataset (above two) 2. BRATS MICCAI dataset below)

### **BRATS MICCAI DATASET:**

The Multimodal Brain Tumor Segmentation (BRATS) MICCAI has always been focusing on the evaluation of state-of-the-art methods for the segmentation of brain tumors in magnetic resonance imaging (MRI) scans. Ample multi-institutional routine clinically-acquired multimodal MRI scans of glioblastoma (GBM) and lower grade glioma (LGG), with pathologically confirmed diagnosis and available OS, was provided as the training, validation and testing data for BRATS 2015 challenge. All BRATS multimodal scans are available as NIFTI files (.nii.gz) and these multimodal scans describe a) native (T1) and b) post-contrast T1-weighted (T1c), c) T2-weighted (T2), and d) T2 Fluid Attenuated Inversion Recovery (FLAIR) volumes, and were acquired with different clinical protocols and various scanners from multiple institutions. They described a mixture of pre- and post-operative scans and their ground truth labels have been annotated by the fusion of segmentation results from algorithms. All the imaging datasets have been segmented manually, by one to four raters, following the same annotation protocol, and their annotations were approved by experienced neuro-radiologists. Annotations comprise the whole tumor, the tumor core (including cystic areas), and the C- enhancing tumor core.

The dataset contains 2 folders for the purpose of training and testing. The 'train' folder contains 2 sub-folders of HGG and LGG cases-220 patients of HGG and 27 patients of LGG. The test folder contains brain images of 110 Patients with HGG and LGG cases combined. There are 5 different MRI image modalities for each patient which are T1, T2, T1C, FLAIR, and OT (Ground truth of tumor Segmentation). All these image files are stored in .mha format and are of the size of 240x240, resolution of (1 mm<sup>3</sup>) and skull-stripped. In the ground truth images, each voxel is labelled with zeros and non-zeros, corresponding to the normal pixel and parts of tumor cells, respectively.

### **DATA AUGMENTATION:**

Data augmentation consists of Grey Scaling(RGB/BW to ranges of grey),Reflection(vertical/horizontal flip),Gaussian Blur(reduces image noise),Histogram equalization(increases global contrast),Rotation(may not preserve image size),Translation(moving the image along x or y axis), linear transformation such as random rotation (0-10 degrees), horizontal and vertical shifts, and horizontal and vertical flips. Data Augmentation is done to teach the network desired invariance and robustness properties, when only few training samples are available.

### **IMAGE PRE-PROCESSING:**

Our pre-processing includes rescaling, noise removal to enhance the image, applying Binary Thresholding and morphological operations like erosion and dilation, contour forming (edge based methodology). In the first step of pre-processing, the memory space of the image is reduced by scaling the gray-level of the pixels in the range 0-255. We used Gaussian blur filter for noise removal as it is known to give better results than Median filter since the outline of brain is not segmented as tumor here.

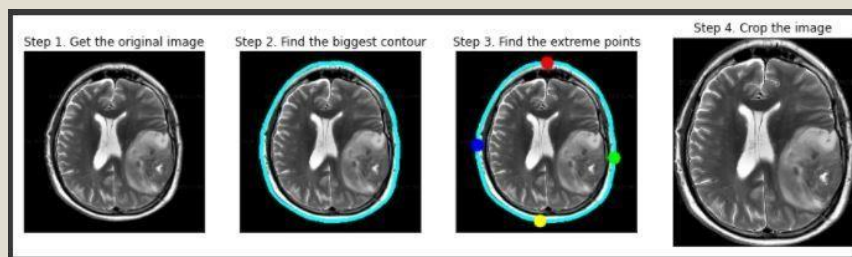


Fig 5. Data Preprocessing using OpenCV

## SEGMENTATION:

Brain tumor segmentation involves the process of separating the tumor tissues (Region of Interest - ROI) from normal brain tissues and solid brain tumor with the help of MRI images or other imaging modalities. Its mechanism is based on identifying similar type of subjects inside an image and forms a group of such by either finding the similarity measure between the objects and group the objects having most similarity or finding the dissimilarity measure among the objects and separate the most dissimilar objects in the space. Segmentation algorithms can be of two type which are bi-clusters (2 sub-parts) or multi-clustered (more than 2 sub-parts) algorithms. Segmentation can be done by using-Edge Detection, Region Growing, Watershed, Clustering via FCM, Spatial Clustering, Split and Merge Segmentation and Neural Network via MLP(ANN+DWT).

In order to identify the tumor region from the brain image, Binary Thresholding can be used (via Region Growing method), which converts a gray scale image to binary image based on the selected threshold values. The problems associated with such approach are that binary image results in loss of texture and the threshold value comes out be different for different images. Hence, we are looking for a more advanced segmentation algorithm, the watershed algorithm by using Otsu Banalization.

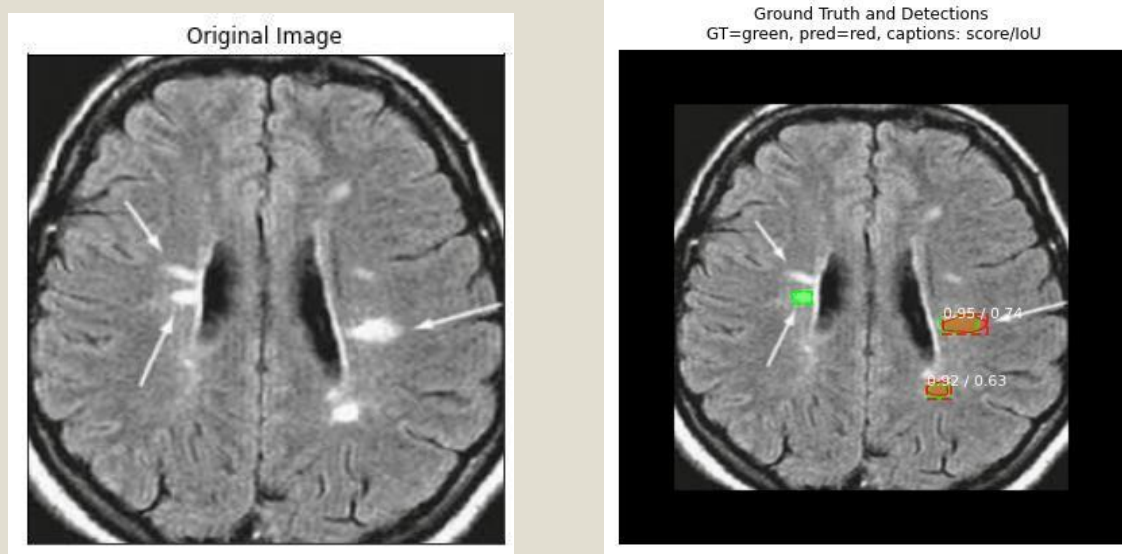


Fig 6: Tumor segmentation using Mask RCNN

## **FEATURE EXTRACTION:**

Feature Extraction is the mathematical statistical procedure that extracts the quantitative parameter of resolution changes/abnormalities that are not visible to the naked eye. Examples of such features are Entropy, RMS, Smoothness, Skewness, Symmetry, Kurtosis, Mean, Texture, Variance, Centroid, Central Tendency, IDM (Inverse Difference Moment), and Correlation, Energy, Homogeneity, Dissimilarity, Contrast, Shade, Prominence, Eccentricity, Perimeter, Area and many more.

Feature Extraction is identifying abnormalities. We need to extract some features from images as we need to do classification of the images using a classifier which needs these features to get trained on. We chose to extract GLCM (texture-based features). Gray Level Co-occurrence Matrix (GLCM) features are based on probability density function and frequency of occurrence of similar pixels. GLCM is a statistical method of examining texture that considers the spatial relationship of pixels.

## **MACHINE LEARNING TRAINING AND TESTING:**

Models for image classification with weights on ImageNet are Xception, VGG16, VGG19, ResNet, ResNet2, ResNet 34, Inception v2, Inception v3, MobileNet, MobileNet v2, DenseNet, AlexNet, GoogleNet, NasNet etc. For the implementation of Transfer Learning in our project, we have chosen VGG16, ResNet34 and Inception v3 as our samples.

After training the model, we need to validate and fine-tune the parameters and finally test the model on unknown samples where the data undergoes feature extraction on the basis of which the model can predict the class by matching corresponding labels. To achieve this, we can either split our dataset in the ratio of 60/20/20 or 70/20/10. We have used the former one.

For a given training dataset, back-propagation learning may proceed in one of the following two basic ways:

- Pattern/Sequential/Incremental mode where the whole sequence of forward and backward computation is performed resulting in weight adjustment for each pattern. It again starts from the first pattern till errors are minimized, within acceptable levels. It is done online,

Requires less local storage, faster method and is less likely to be trapped in local minima.

- Batch mode where the weight upgradation is done after all the N training sets or epochs are presented. After presentation of the full set, weights are upgraded and then again the whole batch/set is presented iteratively till the minimum acceptable error is arrived at by comparing the target and actual outputs. Training stops when a given number of epochs elapse or when the error reaches an acceptable level or when the error stops improving.

We have used this mode during our Machine Learning training by taking the value of N as 30.

In supervised network, the network learns by comparing the network output with the correct answer. The network receives feedback about the errors by matching the corresponding labels and weights in different layers and adjusts its weights to minimize the error. It is also known as learning through teacher or Reinforced Learning. In unsupervised network, there is no teacher i.e. labels are not provided along with the data to the network. Thus, the network does not get any feedback about the errors. The network itself discovers the interesting categories or features in the input data. In many situations, the learning goal is not known in terms of correct answers. The only available information is in the correlation of input data or signals. The unsupervised networks are expected to recognize the input patterns, classify these on the basis of correlations and produce output signals corresponding to input categories. It is a type of dynamic programming that trains algorithm using a system of reward and punishment. Agent learns without human interaction and examples and only by interacting with the environment. For our purpose, we have used supervised network or Reinforced Learning for training our model.

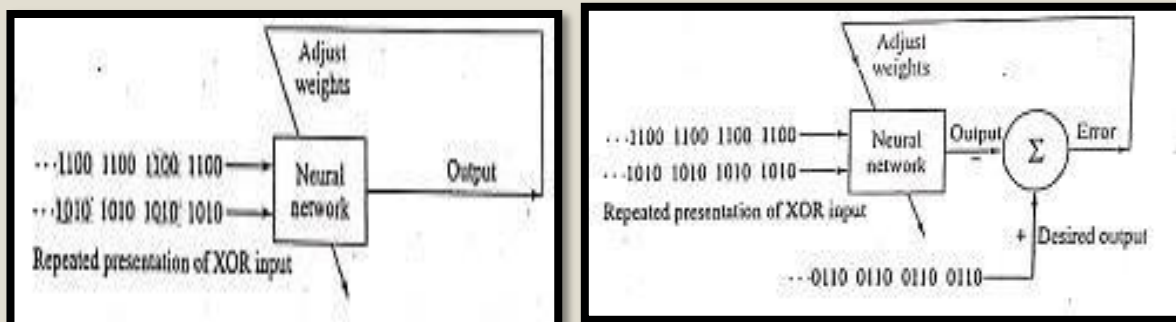
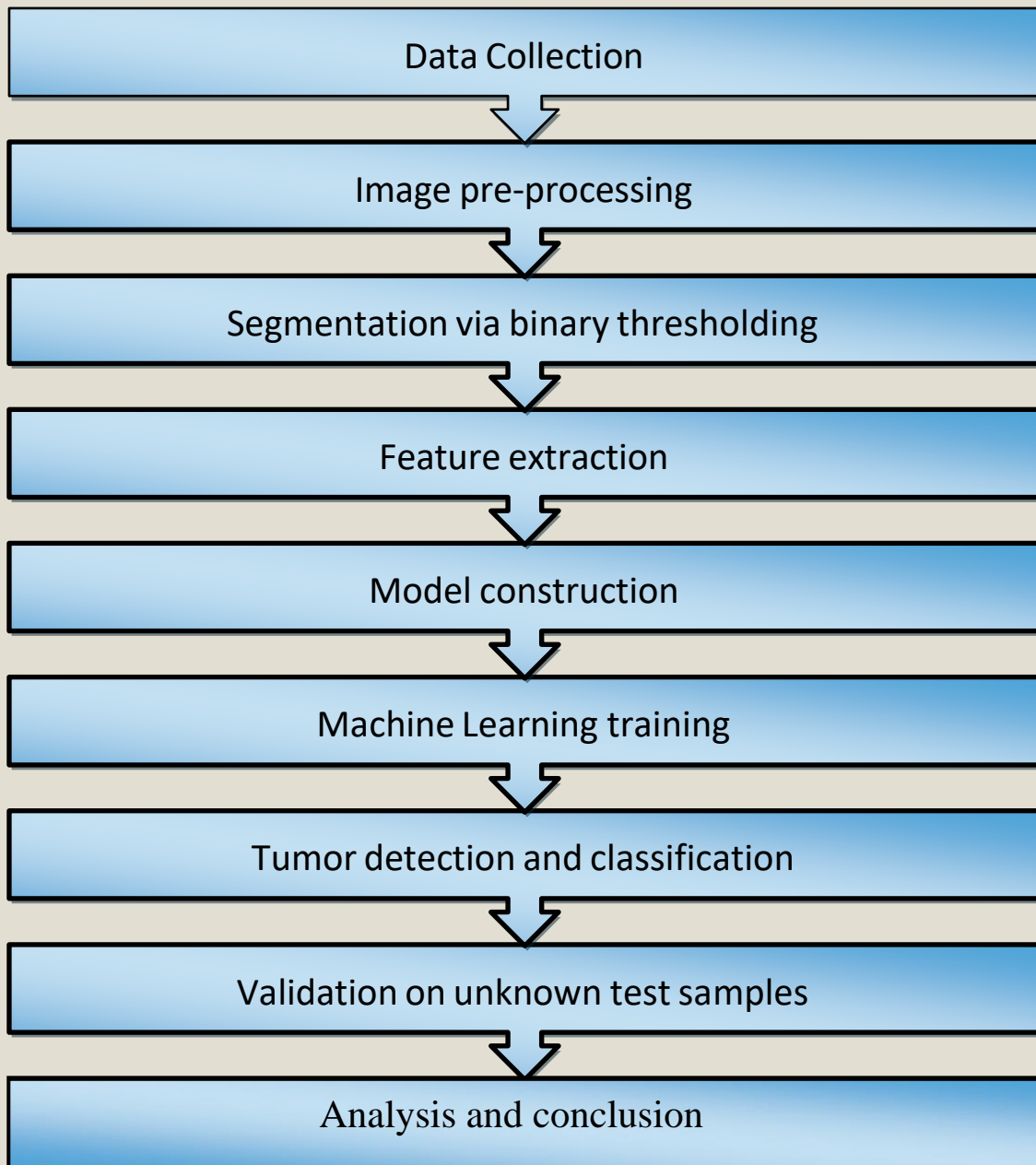


Fig 7: A diagram showing Unsupervised (left) and Supervised Learning Network (right)



## **FLOWCHART FOR DESIGN AND DEVELOPMENT OF PROPOSED PROJECT**



# PYTHON PROGRAM FOR THE PROPOSED PROJECT

## IMPORT NECESSARY PYTHON PACKAGES:

```
import tensorflow as tf
from tensorflow.keras.layers import Conv2D, Input, ZeroPadding2D, BatchNormali
zation, Activation, MaxPooling2D, Flatten, Dense
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 cv2
import imutils
import numpy as np
import matplotlib.pyplot as plt
import time
from os import listdir
```

## UPLOADING THE DATASET:

```
def load_data(dir_list, image_size):
    """
    Read images, resize and normalize them.
    Arguments:
        dir_list: list of strings representing file directories.
    Returns:
        X: A numpy array with shape = (#_examples, image_width, image_height,
#_channels)
        y: A numpy array with shape = (#_examples, 1)
    """

    # load all images in a directory
    X = []
    y = []
    image_width, image_height = image_size

    for directory in dir_list:
        for filename in listdir(directory):
            # load the image
            image = cv2.imread(directory + '\\\\' + filename)
            # crop the brain and ignore the unnecessary rest part of the image
            image = crop_brain_contour(image, plot=False)
            # resize image
```

```

        image = cv2.resize(image, dsize=(image_width, image_height), inter
polation=cv2.INTER_CUBIC)
        # normalize values
        image = image / 255.
        # convert image to numpy array and append it to X
        X.append(image)
        # append a value of 1 to the target array if the image
        # is in the folder named 'yes', otherwise append 0.
        if directory[-3:] == 'yes':
            y.append([1])
        else:
            y.append([0])

X = np.array(X)
y = np.array(y)

# Shuffle the data
X, y = shuffle(X, y)

print(f'Number of examples is: {len(X)}')
print(f'X shape is: {X.shape}')
print(f'y shape is: {y.shape}')

return X, y

```

## DATA AUGMENTATION:

```

import tensorflow as tf
from keras.preprocessing.image import ImageDataGenerator
import cv2
import imutils
import matplotlib.pyplot as plt
from os import listdir
import time

```

```

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 augment_data(file_dir, n_generated_samples, save_to_dir):
    """
    Arguments:
        file_dir: A string representing the directory where images that we want
to augment are found.

```

```

    n_generated_samples: A string representing the number of generated sam
ples using the given image.
    save_to_dir: A string representing the directory in which the generate
d images will be saved.
    """

#from keras.preprocessing.image import ImageDataGenerator
#from os import listdir

data_gen = ImageDataGenerator(rotation_range=10,
                              width_shift_range=0.1,
                              height_shift_range=0.1,
                              shear_range=0.1,
                              brightness_range=(0.3, 1.0),
                              horizontal_flip=True,
                              vertical_flip=True,
                              fill_mode='nearest'
                              )

for filename in listdir(file_dir):
    # load the image
    image = cv2.imread(file_dir + '\\\\' + filename)
    # reshape the image
    image = image.reshape((1,)+image.shape)
    # prefix of the names for the generated sampels.
    save_prefix = 'aug_' + filename[:-4]
    # generate 'n_generated_samples' sample images
    i=0
    for batch in data_gen.flow(x=image, batch_size=1, save_to_dir=save_to_
dir,
                              save_prefix=save_prefix, save_forma
t='jpg'):
        i += 1
        if i > n_generated_samples:
            break

```

## SPLITTING THE DATASET INTO TRAIN, TEST AND VAL AND FEATURE EXTRACTION:

```
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, im  
age_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, imag  
e_height, #_channels)  
        y_test: A numpy array with shape = (#_test_examples, 1)  
    """  
  
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))
```

## IMAGE PRE-PROCESSING AND PERFORMING BINARY THRESHOLDING:

```
def augment_data(file_dir, n_generated_samples, save_to_dir):  
    """  
    Arguments:  
        file_dir: A string representing the directory where images that we wan  
t to augment are found.
```

```

    n_generated_samples: A string representing the number of generated sam
ples using the given image.
    save_to_dir: A string representing the directory in which the generate
d images will be saved.
    """

#from keras.preprocessing.image import ImageDataGenerator
#from os import listdir

data_gen = ImageDataGenerator(rotation_range=10,
                              width_shift_range=0.1,
                              height_shift_range=0.1,
                              shear_range=0.1,
                              brightness_range=(0.3, 1.0),
                              horizontal_flip=True,
                              vertical_flip=True,
                              fill_mode='nearest'
                              )

for filename in listdir(file_dir):
    # load the image
    image = cv2.imread(file_dir + '\\\\' + filename)
    # reshape the image
    image = image.reshape((1,)+image.shape)
    # prefix of the names for the generated sampels.
    save_prefix = 'aug_' + filename[:-4]
    # generate 'n_generated_samples' sample images
    i=0
    for batch in data_gen.flow(x=image, batch_size=1, save_to_dir=save_to_
dir,
                              save_prefix=save_prefix, save_forma
t='jpg'):
        i += 1
        if i > n_generated_samples:
            break

```

## MODEL CONSTRUCTION:

### 1. Load pre-trained models (with transfer learning)

```

def crop_brain_contour(image, plot=False):

    #import imutils
    #import cv2

```

```

#from matplotlib import pyplot as plt

# Convert the image to grayscale, and blur it slightly
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)

# Find the extreme points
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])

# crop new image out of the original image using the four extreme points (
left, right, top, bottom)
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, right=False,
                    labelbottom=False, labeltop=False, labelleft=False, la
belright=False)

    plt.title('Original Image')

    plt.subplot(1, 2, 2)
    plt.imshow(new_image)

    plt.tick_params(axis='both', which='both',

```

```

        top=False, bottom=False, left=False, right=False,
        labelbottom=False, labeltop=False, labelleft=False, la
belright=False)

    plt.title('Cropped Image')

    plt.show()

    return new_image

```

## 2. Build model from scratch (without transfer learning)

```

def build_model(input_shape):
    """
    Arugments:
        input_shape: A tuple representing the shape of the input of the mode
1. shape=(image_width, image_height, #_channels)
    Returns:
        model: A Model object.
    """
    # Define the input placeholder as a tensor with shape input_shape.
    X_input = Input(input_shape) # shape=(?, 240, 240, 3)

    # Zero-Padding: pads the border of X_input with zeroes
    X = ZeroPadding2D((2, 2))(X_input) # shape=(?, 244, 244, 3)

    # CONV -> BN -> RELU Block applied to X
    X = Conv2D(32, (7, 7), strides = (1, 1), name = 'conv0')(X)
    X = BatchNormalization(axis = 3, name = 'bn0')(X)
    X = Activation('relu')(X) # shape=(?, 238, 238, 32)

    # MAXPOOL
    X = MaxPooling2D((4, 4), name='max_pool0')(X) # shape=(?, 59, 59, 32)

    # MAXPOOL
    X = MaxPooling2D((4, 4), name='max_pool1')(X) # shape=(?, 14, 14, 32)

    # FLATTEN X
    X = Flatten()(X) # shape=(?, 6272)
    # FULLYCONNECTED
    X = Dense(1, activation='sigmoid', name='fc')(X) # shape=(?, 1)

    # Create model. This creates your Keras model instance, you'll use this
instance to train/test the model.
    model = Model(inputs = X_input, outputs = X, name='BrainDetectionModel')

    return model

```



## MACHINE LEARNING TRAINING:

```
start_time = time.time()

model.fit(x=X_train, y=y_train, batch_size=32, epochs=10, validation_data=(X_val, y_val), callbacks=[tensorboard, checkpoint])

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

model.fit(x=X_train, y=y_train, batch_size=32, epochs=3, validation_data=(X_val, y_val), callbacks=[tensorboard, checkpoint])

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

start_time = time.time()

model.fit(x=X_train, y=y_train, batch_size=32, epochs=3, validation_data=(X_val, y_val), callbacks=[tensorboard, checkpoint])

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

start_time = time.time()

model.fit(x=X_train, y=y_train, batch_size=32, epochs=3, validation_data=(X_val, y_val), callbacks=[tensorboard, checkpoint])

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

start_time = time.time()
```

```

model.fit(x=X_train, y=y_train, batch_size=32, epochs=5, validation_data=(X_val, y_val), callbacks=[tensorboard, checkpoint])

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

history = model.history.history

for key in history.keys():
    print(key)

```

## EVALUATION OF MODEL PERFORMANCE:

```

print('Train: %.3f, Test: %.3f'%(train_acc,test_acc))
accuracy=accuracy_score(y_test, predictions)
print('Accuracy: %f'% accuracy)
precision=precision_score(y_test, predictions)
print('Precision: %f'% precision)
recall=recall_score(y_test, predictions)
print('Recall: %f'% recall)
f1 = f1_score(y_test, predictions)
print('F1 score: %f'% f1)
kappa=cohen_kappa_score(y_test, predictions)
print('Cohens kappa: %f'% kappa)
auc=roc_auc_score(y_test, predictions)
print('ROC AUC: %f'%auc)
matrix=confusion_matrix(y_test, predictions)
print(matrix)

history_1= vgg16_history
history_2=inception_v3_history
history_3=resnet50_history
def ModelGraphTrainingSummary(history,N,model_name):
    print("Generating plots...")
    sys.stdout.flush()
    matplotlib.use("Agg")
    matplotlib.pyplot.style.use("ggplot")
    matplotlib.pyplot.figure()
    matplotlib.pyplot.plot(np.arange(0, N),history.history["loss"], label="train_loss")
    matplotlib.pyplot.plot(np.arange(0, N),history.history["val_loss"], label="val_loss")
    matplotlib.pyplot.title("Training Loss and Accuracy on Brain Tumor Classification")
    matplotlib.pyplot.xlabel("Epoch #")
    matplotlib.pyplot.ylabel("Loss/Accuracy of "+model_name)
    matplotlib.pyplot.legend(loc="lower left")
    matplotlib.pyplot.savefig("plot.png")

```

```

def ModelGraphTrainngSummaryAcc(history,N,model_name):
    print("Generating plots...")
    sys.stdout.flush()
    matplotlib.use("Agg")
    matplotlib.pyplot.style.use("ggplot")
    matplotlib.pyplot.figure()
    matplotlib.pyplot.plot(np.arange(0, N),history.history["accuracy"], label="train_accuracy")
    matplotlib.pyplot.plot(np.arange(0, N),history.history["val_accuracy"], label="val_accuracy")
    matplotlib.pyplot.title("Training Loss and Accuracy on Brain Tumor Classification")
    matplotlib.pyplot.xlabel("Epoch #")
    matplotlib.pyplot.ylabel("Accuracy of "+model_name)
    matplotlib.pyplot.legend(loc="lower left")
    matplotlib.pyplot.savefig("plot.png")

for X_model in [{'name':'VGG-16','history':history_1,'model':vgg16},
                {'name':'Inception_v3','history':history_2,'model':inception_v3},
                {'name':'Resnet','history':history_3,'model':resnet34}]:
    ModelGraphTrainngSummary(X_model['history'],30,X_model['name'])
    ModelGraphTrainngSummaryAcc(X_model['history'],30,X_model['name'])
    predictions=X_model['model'].predict(X_val_prep)
    predictions=[1 if x>0.5 else 0 for x in predictions]
    accuracy=accuracy_score(y_val, predictions)
    print('Val Accuracy = %.2f%% accuracy)
    confusion_mtx=confusion_matrix(y_val, predictions)
        cm=plot_confusion_matrix(confusion_mtx, classes =list(labels.items()), normalize=False)

def plot_metrics(history):
    train_loss=history['loss']
    val_loss=history['val_loss']
    train_acc=history['accuracy']
    val_acc=history['val_accuracy']
    plt.figure()
    plt.plot(train_loss, label='Training Loss')
    plt.plot(val_loss, label='Validation Loss')
    plt.title('Loss')
    plt.legend()
    plt.show()
    plt.figure()
    plt.plot(train_acc, label='Training Accuracy')
    plt.plot(val_acc, label='Validation Accuracy')
    plt.title('Accuracy')
    plt.legend()
    plt.show()

plot_metrics(model.history.history)
plt.figure()
plt.plot(train_hist,color='r',linewidth=2,label='train')
plt.plot(val_hist,color='g',linewidth=2,label='test')
plt.xlabel('iterations')
plt.legend()
plt.show()

```

```
vgg16.save('2020-04-24_VGG_model.h5')
inception_v3.save('2020-04-24_inception_v3.h5')
resnet34.save('2020-04-24_resnet34.h5')
```

```
filepath="tumor-detection-{epoch:02d}-{val_acc:.2f}.model"
checkpoint=ModelCheckpoint(filepath, monitor='val_acc', verbose=1,save_best_only=True,
mode='max')
```

```
from mpl_toolkits.axes_grid1 import make_axes_locatable
def implot(mp,ax,cmap='gray'):
im=ax.imshow(mp.astype(np.float32),cmap=cmap)
divider=make_axes_locatable(ax)
cax=divider.append_axes("right", size="5%", pad=0.05)
cbar=plt.colorbar(im,cax=cax)
xb,yb=get_batch(X_p_test,Y_p_test,X_n_test,Y_n_test,n=Nbatch)
yh=sess.run(yhat,{x:xb})
ypred=np.argmax(yh,axis=3)
for i in range(7):
plt.figure()
fig,(ax1, ax2, ax3)=plt.subplots(1,3,sharey=True,figsize=(10,3))
implot(xb[i,:,:],ax1)
implot(yb[i,:,:],ax2,cmap='Spectral')
implot(ypred[i,:,:],ax3,cmap='Spectral')
plt.grid('off')
plt.tight_layout()
plt.savefig('images_{ }.pdf'.format(i),dpi=600)
plt.show()
```

---

# EVALUATION OF THE PREDICTIVE MODEL PERFORMANCE

## CNN

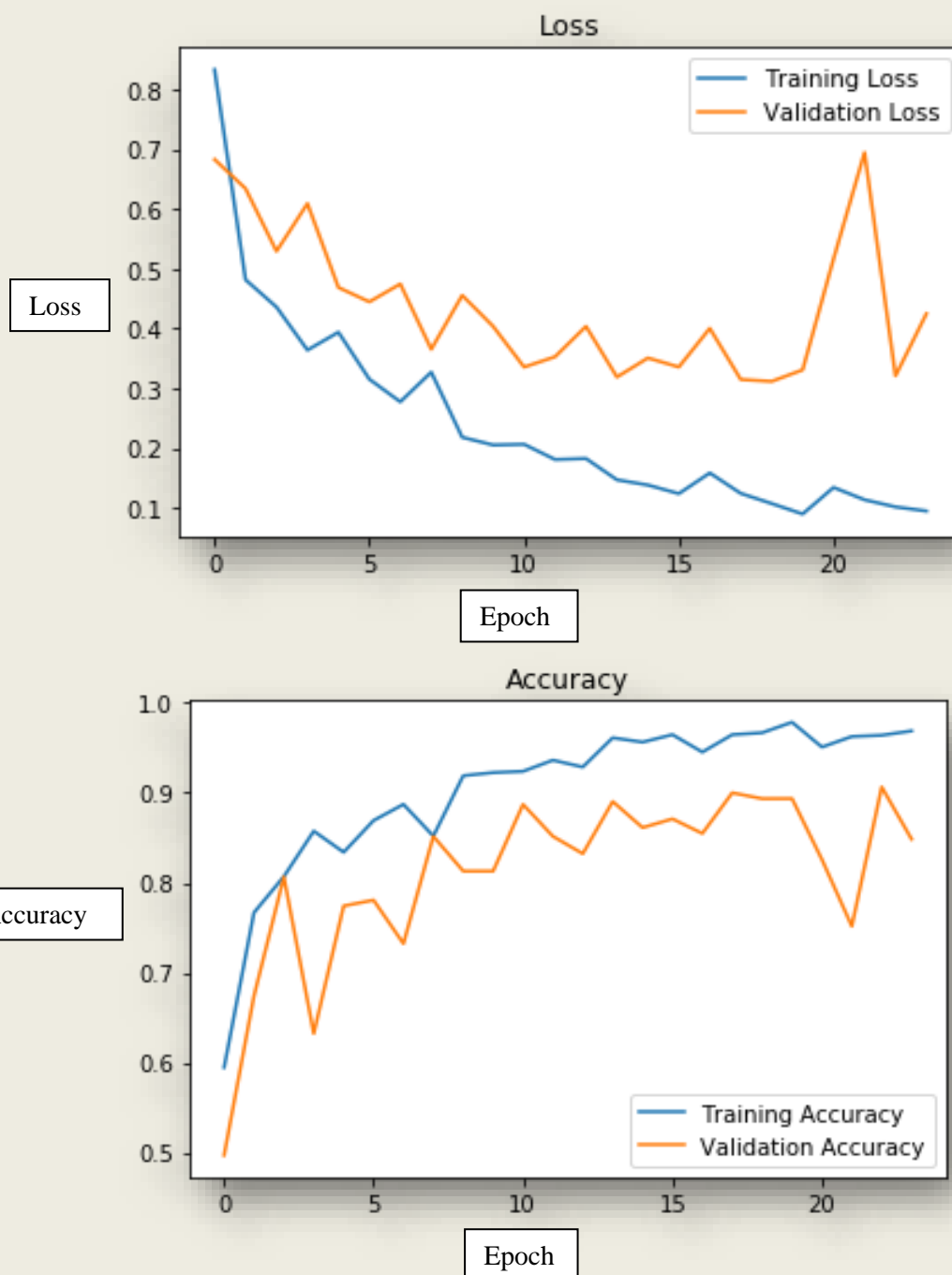
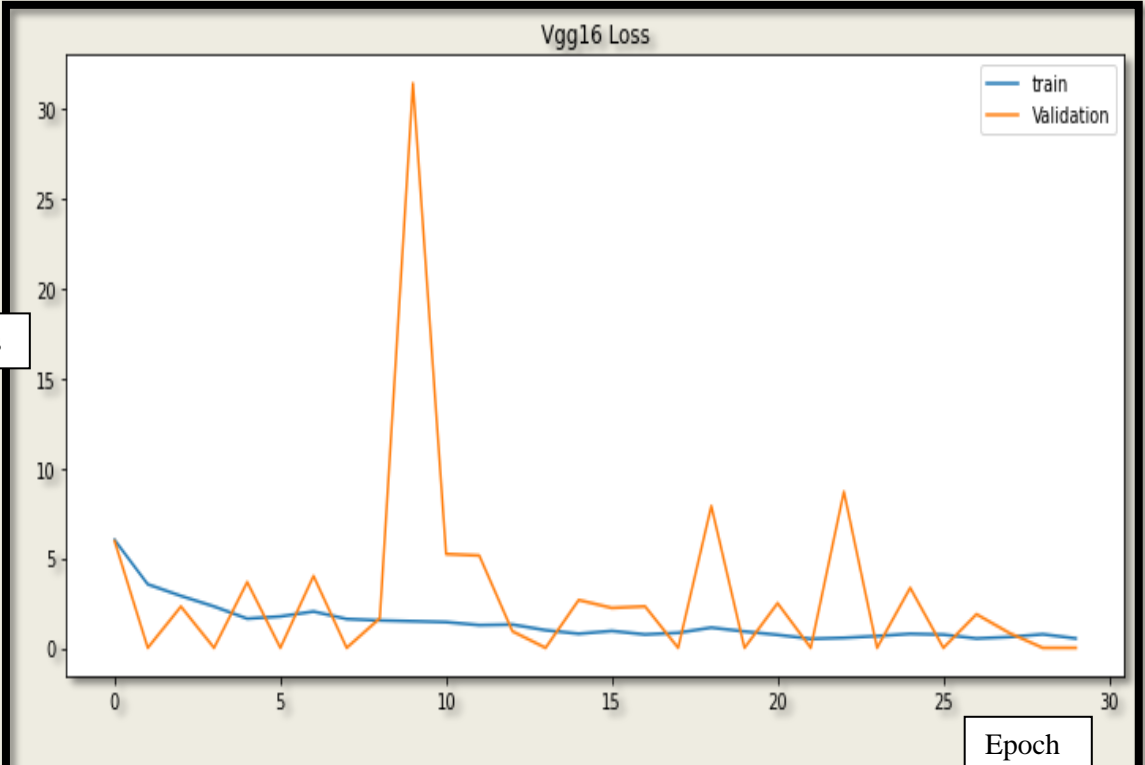


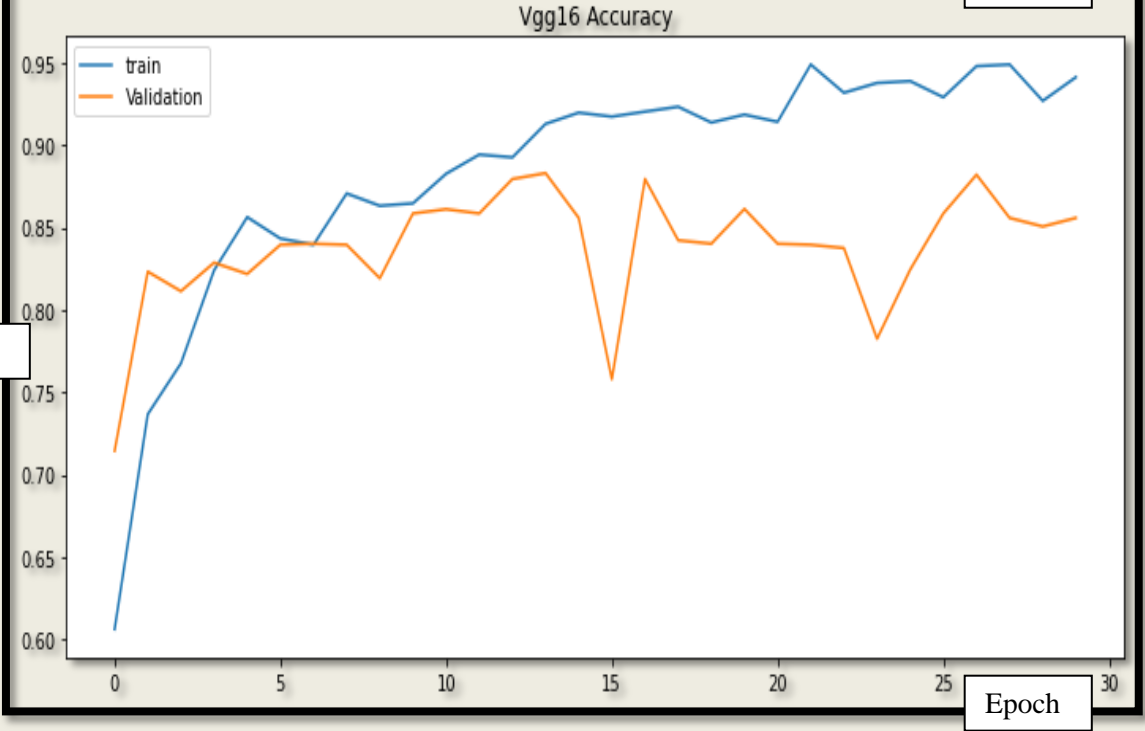
Fig 8: Loss and Accuracy Vs Epoch plots of a **CNN model** without pre-trained Keras models like VGG16, ResNet 34 and Inception v3

Loss



Epoch

Accuracy



Epoch

Fig9 : Loss and Accuracy Vs Epoch plots of VGG-16

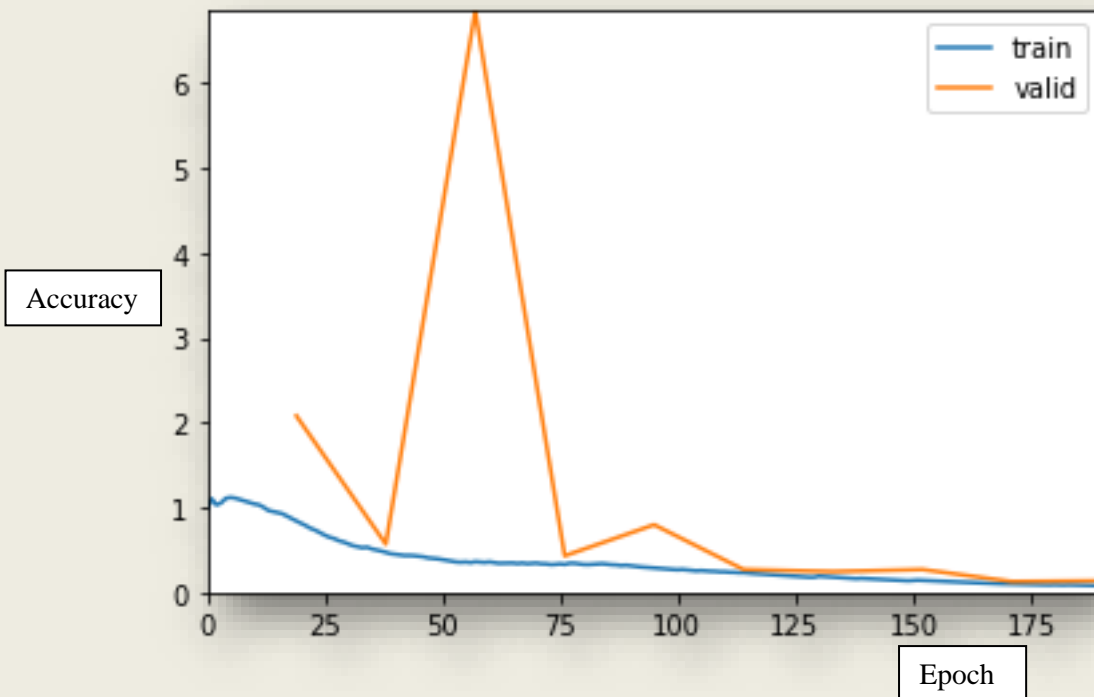
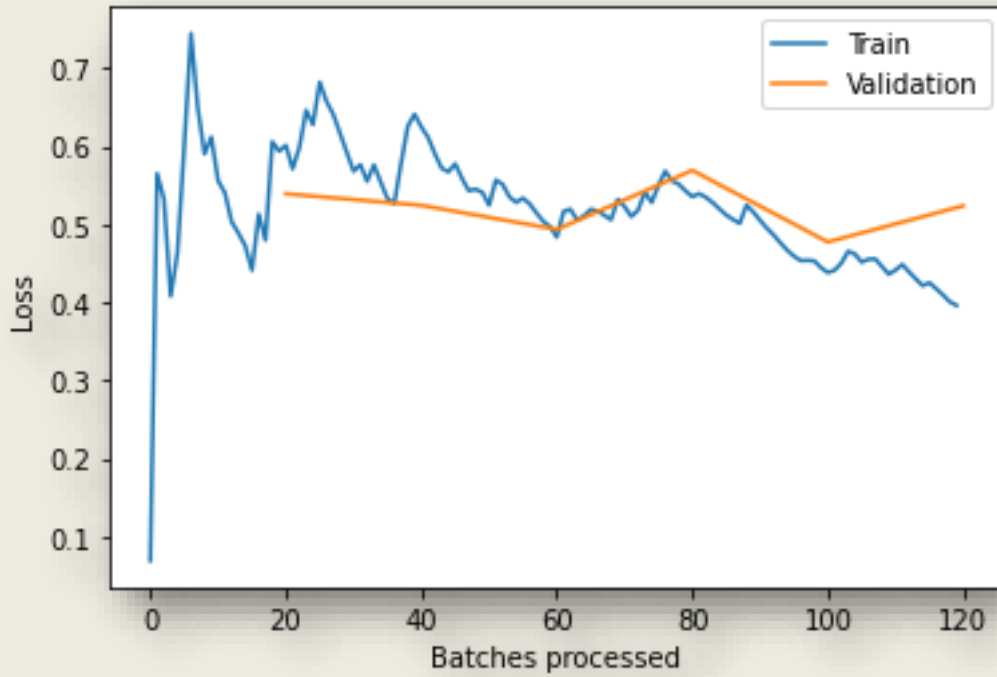


Fig 9: Loss and Accuracy vs Epoch of ResNet 34 model.

## SEGMENTATION USING MASKRCNN

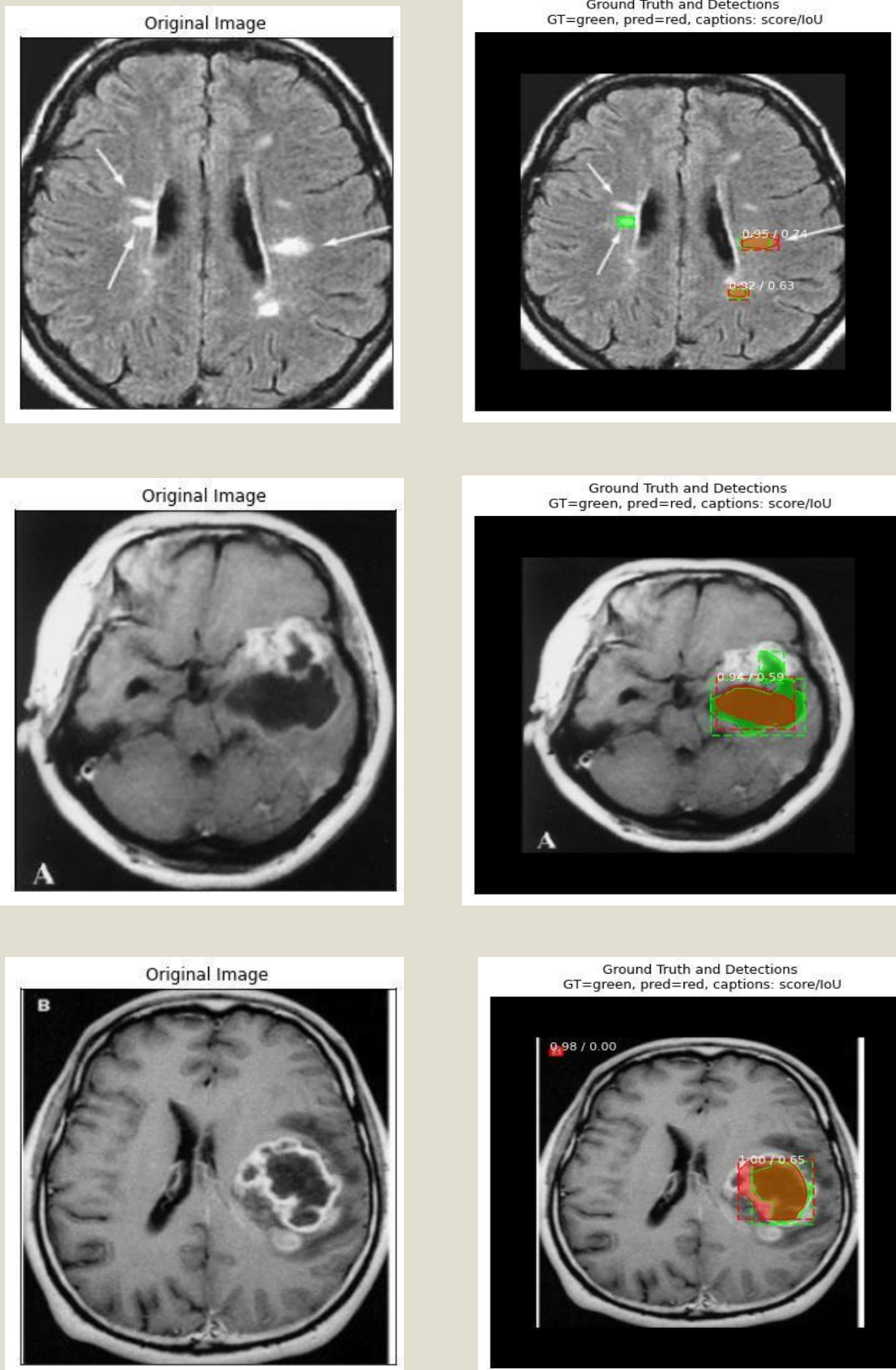


Fig 10: Tumor segmentation using Mask RCNN



## SEGMENTATION USING OPENCV

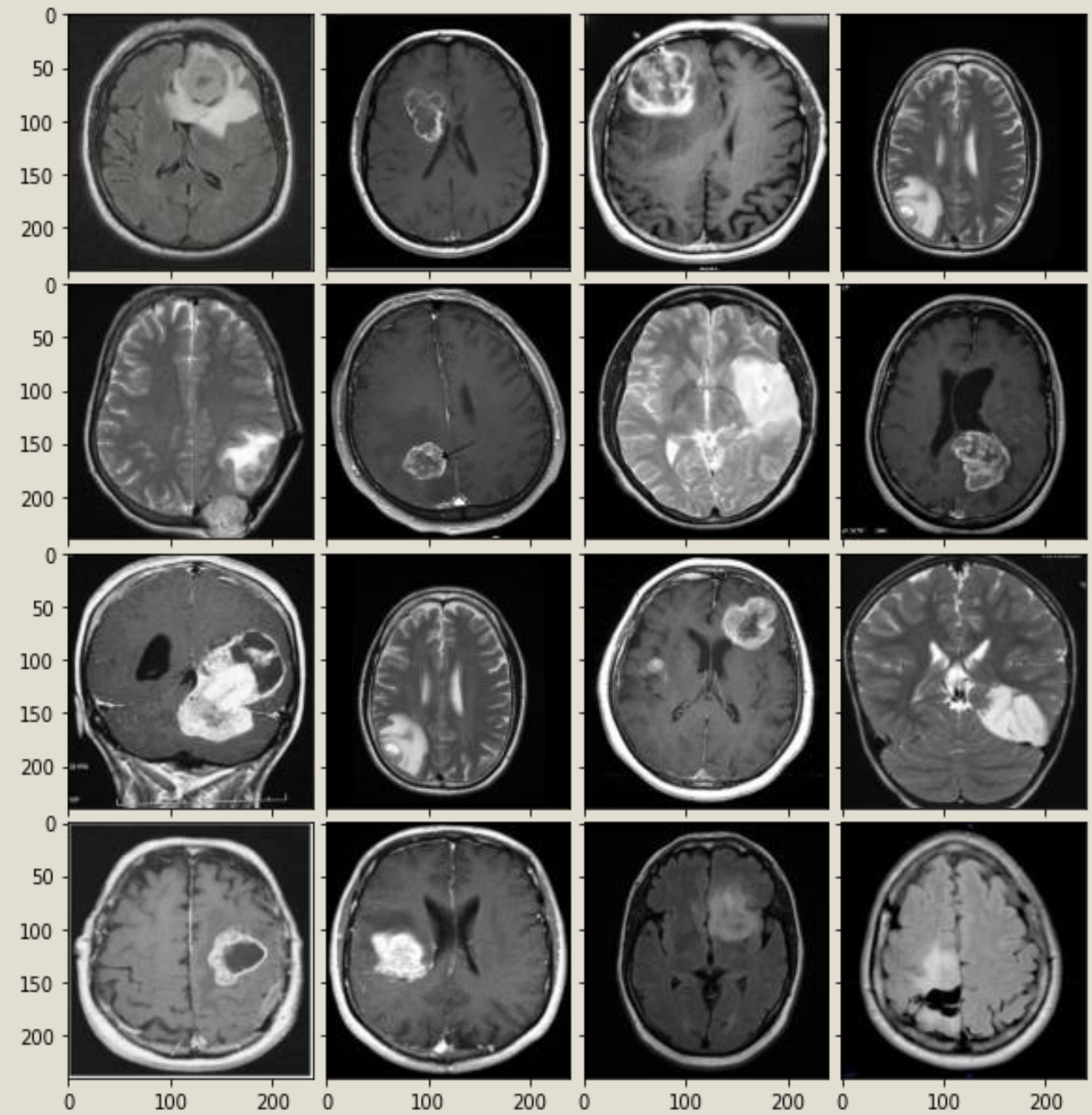


Fig 11: Segmentation using OpenCV

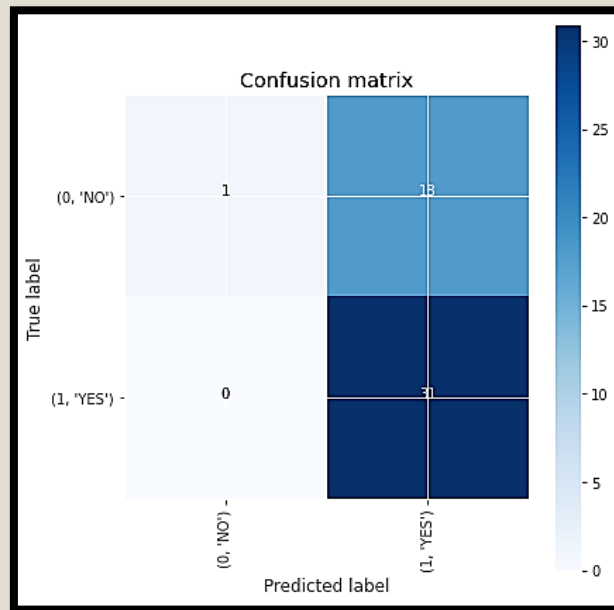
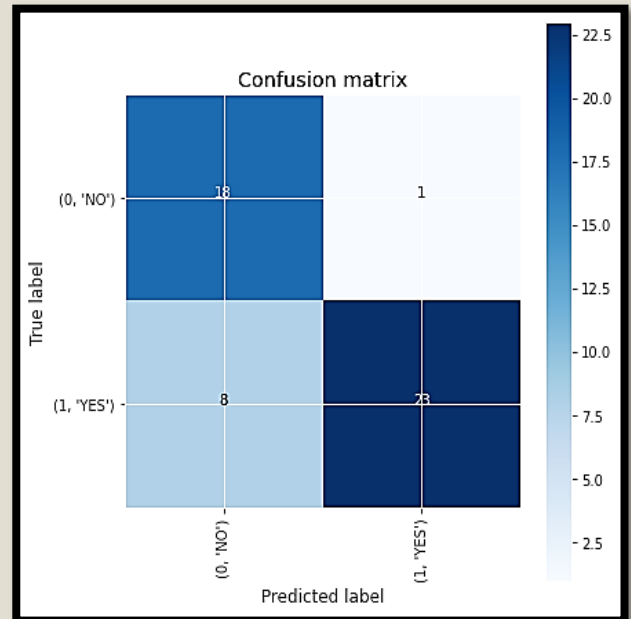
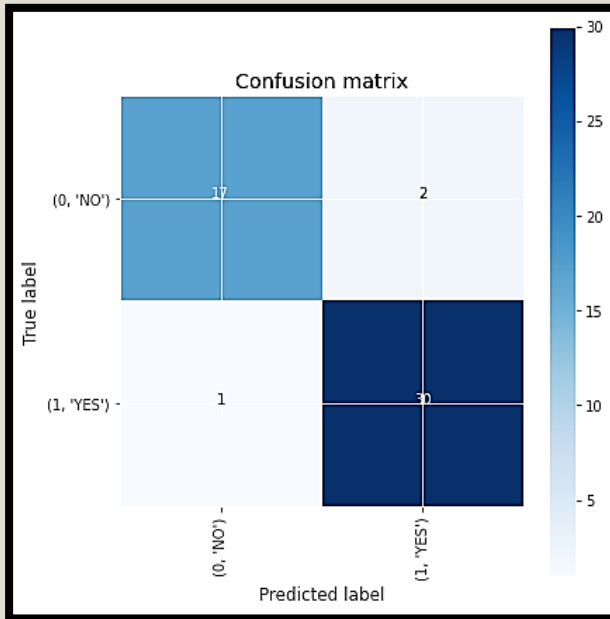


Fig12: Confusion matrix plots (from top left) – VGG16, ResNet34 and Inception v3

**TABLE 1: PERFORMANCE ANALYSIS OF THE PROPOSED MODEL**

<b><u>Metric</u></b>	<b><u>Vgg 16</u></b>	<b><u>Res- Net34</u></b>	<b><u>CNN</u></b>
Train accuracy	0.940	0.9873	0.8800
Test accuracy		0.9900	0.8700
Overall accuracy	0.8608	0.9900	0.8800
Validation Loss	0.4057	0.1354	0.31940
Test Loss	0.3583	0.80023	0.31580
F1 score	0.8700	0.8900	0.8863
AUC	0.8431	0.9400	0.87000

Table 1 : Comparison of 3 pre-trained Keras models

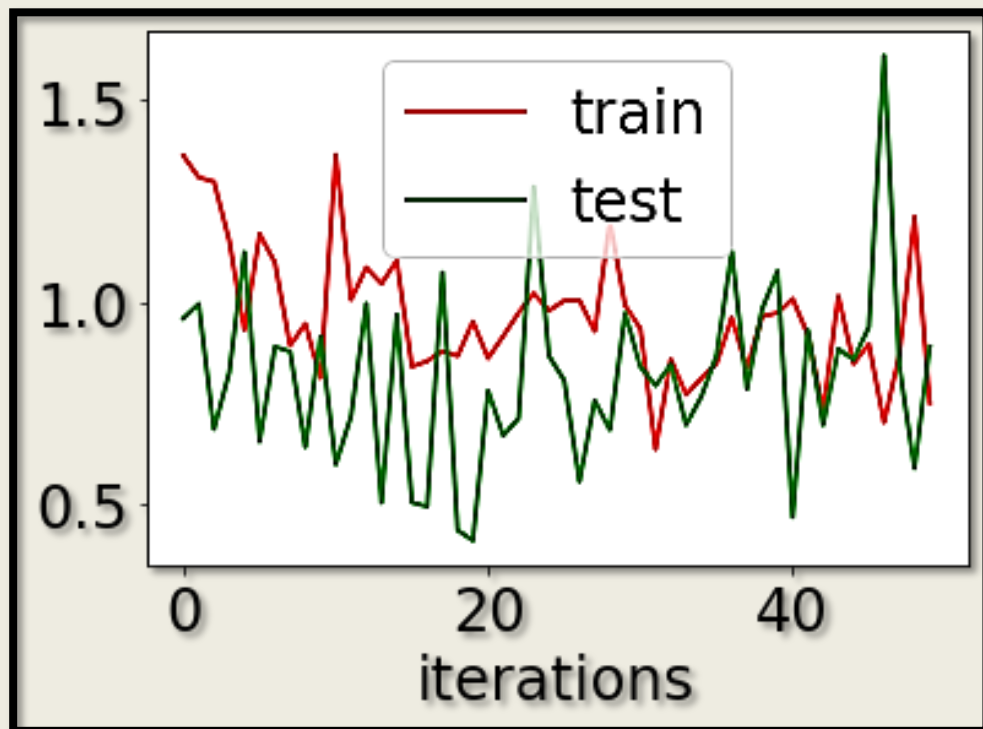


Fig13: Dice loss Vs Epoch after training with BRaTS MICCAI dataset

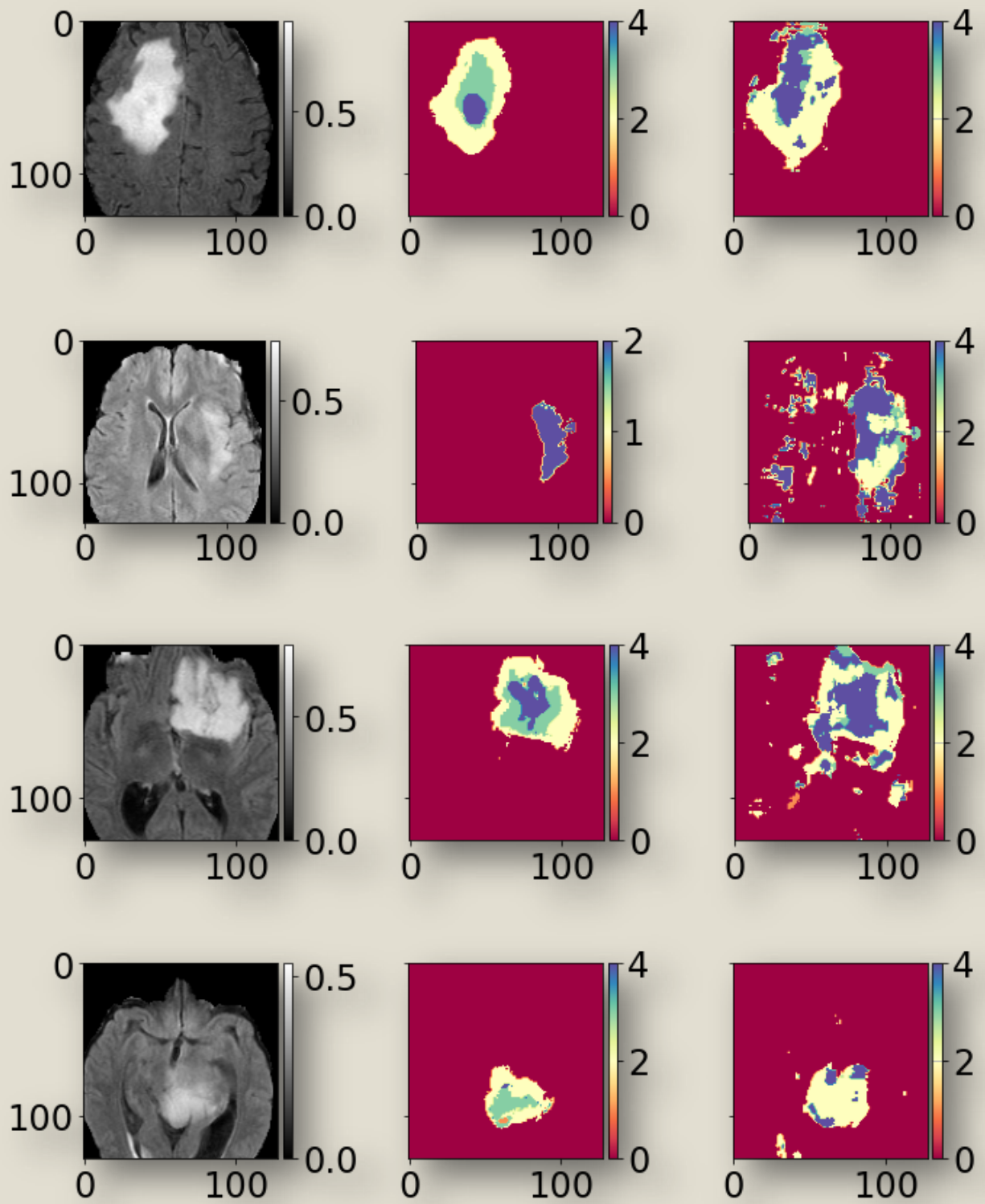


Fig 14 : Original (left), Ground Truth (middle) and our network model on the BRaTS 2015 dataset(right)

## CONCLUSION

Without pre-trained Keras model, the train accuracy is 97.5% and validation accuracy is 90.0%.The validation result had a best figure of 91.09% as accuracy. It is observed that without using pre-trained Keras model, although the training accuracy is >90%, the overall accuracy is low unlike where pre-trained model is used.

Also, when we trained our dataset without Transfer learning, the computation time was 40 min whereas when we used Transfer Learning, the computation time was 20min. Hence, training and computation time with pre-trained Keras model was 50% lesser than without. Chances over over-fitting the dataset is higher when training the model from scratch rather than using pre-trained Keras. Keras also provides an easy interface for data augmentation. Amongst the Keras models, it is seen that ResNet 34 has the best overall accuracy as well as F1 score. ResNet is a powerful backbone model that is used very frequently in many computervision tasks.

Precision and Recall both cannot be improved as one comes at the cost of the other .So,we use F1 score too. Transfer learning can only be applied if low-level features from Task 1(image recognition) can be helpful for Task 2(radiology diagnosis).For a large dataset, Dice loss is preferred over Accuracy. For small size of data, we should use simple models, pool data, clean up data, limit experimentation, use regularization/model averaging ,confidence intervals and single number evaluation metric. To avoid overfitting, we need to ensure we have plenty of testing and validation of data i.e. dataset is not generalized. This is solved by Data Augmentation. If the training accuracy too high, we can conclude that it the model might be over fitting the dataset. To avoid this, we can monitor testing accuracy, use outliers and noise, train longer, compare variance (=train performance-test performance).

## **FUTURE SCOPE**

Build an app-based user interface in hospitals which allows doctors to easily determine the impact of tumor and suggest treatment accordingly

Since performance and complexity of ConvNets depend on the input data representation we can try to predict the location as well as stage of the tumor from Volume based 3D images. By creating three dimensional (3D) anatomical models from individual patients, training, planning and computer guidance during surgery is improved. Using Volume Net with LOPO (Leave-One-Patient-Out) scheme has proved to give a high training as well as validation accuracy(>95%). In LOPO test scheme, in each iteration, one patient is used for testing and remaining patients are used for training the ConvNets, this iterates for each patient. Although LOPO test scheme is computationally expensive, using this we can have more training data which is required for ConvNets training. LOPO testing is robust and most applicable to our application, where we get test result for each individual patient. So, if classifier misclassifies a patient then we can further investigate it separately. Improve testing accuracy and computation time by using classifier boosting techniques like using more number images with more data augmentation, fine-tuning hyper parameters, training for a longer time i.e. using more epochs, adding more appropriate layers etc.. Classifier boosting is done by building a model from the training data then creating a second model that attempts to correct the errors from the first model for faster prognosis. Such techniques can be used to raise the accuracy even higher and reach a level that will allow this tool to be a significant asset to any medical facility dealing with brain tumors. For more complex datasets, we can use U-Net architecture rather than CNN where the max pooling layers are just replaced by up sampling ones.

Ultimately we would like to use very large and deep convolutional nets on video sequences where the temporal structure provides very helpful information that is missing or far less obvious in static images.

Unsupervised transfer learning may attract more and more attention in the future.

## **BIBLIOGRAPHY**

- [1]. S. Bauer et al., -A survey of MRI-based medical image analysis for brain tumor studies,|| Phys. Med. Biol., vol. 58, no. 13, pp.97–129, 2013.
- [2]. B. Menze et al., -The multimodal brain tumor image segmentation benchmark (BRATS),|| IEEE Trans. Med. Imag., vol. 34, no.10, pp. 1993–2024, Oct. 2015
- [3]. B. H. Menze et al., -A generative model for brain tumor segmentation in multi-modal images,|| in Medical Image Computing and Comput.- Assisted Intervention-MICCAI 2010. New York: Springer, 2010, pp. 151–159
- [4]. S. Bauer, L.-P. Nolte, and M. Reyes, -Fully automatic segmentation of brain tumor images using support vector machine classification in combination with hierarchical conditional random field regularization,|| in Medical Image Computing and Comput.-Assisted Intervention-MICCAI 2011. New York: Springer, 2011, pp. 354–361.
- [5]. C.-H. Lee et al., -Segmenting brain tumors using pseudo-conditional random fields,|| in Medical Image Computing and Comput.-Assisted Intervention-MICCAI 2008. New York: Springer, 2008, pp. 359–366
- [7]. R. Meier et al., -A hybrid model for multimodal brain tumor segmentation,|| in Proc. NCI-MICCAI BRATS, 2013, pp. 31–37.
- [8]. Vinod Kumar, JainySachdeva, Indra Gupta -Classification of brain tumors using PCA-ANN|| 2011 World Congress on Information and Communication Technologies
- [9]. Sergio Pereira, Adriano Pinto, Victor Alves, and Carlos A. Silva -Brain Tumor Segmentation Using Convolutional Neural Networks in MRI Images||IEEE TRANSACTIONS ON MEDICAL IMAGING, VOL. 35, NO. 5, MAY 2016
- [10]. RaselAhmmed, Md. Foisal Hossain -Tumor Detection in Brain MRI Image Using Template based K-means and Fuzzy C-means Clustering Algorithm|| 2016 International Conference on Computer Communication and Informatics (ICCCI -2016), Jan. 07 –09, 2016, Coimbatore, INDIA
- [11]. S.C. Turaga, J.F. Murray, V. Jain, F. Roth, M. Helmstaedter, K. Briggman,W. Denk, and H.S. Seung. Convolutional networks can learn to generate affinity graphs for image segmentation. Neural Computation, 22(2):511–538, 2010
- [12] A Reliable Method for Brain Tumor Detection Using Cnn Technique NeethuOuseph C1, Asst. Prof. Mrs.Shruti K2 1(Digital electronics ECE, Malabar Institute of Technology, India) 2(Electronics and Communication Engineering, Malabar Institute of Technology, India)

## **REFERENCES**

- [1] <https://keras.io/applications/>
- [2] <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>
- [3] <https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751>
- [4] <https://simpleitk.org>
- [5] <https://neurohive.io/en/popular-networks/resnet/>
- [6] <https://scikit-learn.org/stable/modules/svm.html>
- [7] <http://builtin.com/data-science/transfer-learning>
- [8] <https://openreview.net/forum?id=BJIRs34Fvr>
- [9] <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6640210>
- [10] <https://arxiv.org/pdf/1409.1556.pdf>
- [11] [https://www.cse.ust.hk/~qyang/Docs/2009/tkde\\_transfer\\_learning.pdf](https://www.cse.ust.hk/~qyang/Docs/2009/tkde_transfer_learning.pdf)
- [12] <https://arxiv.org/pdf/1409.4842.pdf>
- [13] <https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>
- [14] <https://arxiv.org/pdf/1512.03385.pdf>