A PROJECT REPORT ON

Brain Tumor Classification Using Deep Learning Algorithms

SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY , PUNE IN THE PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE

BACHELOR OF ENGINEERING
(Computer Engineering)

BY

Ankita Kadam	Exam No: 71709661M
Sartaj Bhuvaji	Exam No: 71709702B
Prajakta Bhumkar	Exam No: 71709700F
Sameer Dedge	Exam No: 71709735J

Under the guidance of

Mr. Santosh Nagargoje

DEPARTMENT OF COMPUTER ENGINEERING

PES MODERN COLLEGE OF ENGINEERING
SHIVAJINAGAR,PUNE 411005

SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE
2019 - 2020

CERTIFICATE

This is to certify that the Project Entitled

Brain Tumor Classification Using Deep Learning Algorithms

Submitted by

Ankita Kadam	Exam No: 71709661M
Sartaj Bhuvaji	Exam No: 71709702B
Prajakta Bhumkar	Exam No: 71709700F
Sameer Dedge	Exam No: 71709735J

is a bonafide work carried out by them under the supervision of Mr. Santosh Nagargoje and it is approved for the partial fulfillment of the requirement of Savtribai Phule Pune university, Pune for the award of the degree of Bachelor of Engineering (Computer Engineering).

Mr. Santosh Nagargoje
Guide
Department of Computer Engineering

Prof. Dr. Mrs. S. A. ITKAR
Head
Department of Computer Engineering

Signature of Internal Examiner

Signature of External Examiner

PROJECT APPROVAL SHEET

A Project Report Titled as

Brain Tumor Classification Using Deep Learning Algorithms

is successfully completed by

Ankita Kadam	Exam No: 71709661M
Sartaj Bhuvaji	Exam No: 71709702B
Prajakta Bhumkar	Exam No: 71709700F
Sameer Dedge	Exam No: 71709735J

at

DEPARTMENT OF COMPUTER ENGINEERING

PES MODERN COLLEGE OF ENGINEERING

SAVITRIBAI PHULE PUNE UNIVERSITY,PUNE

ACADEMIC YEAR 2018-2019

Mr. Santosh Nagargoje
Guide
Department of Computer Engineering

Prof.Dr. Mrs. S. A. Itkar
Head
Department of Computer Engineering

Acknowledgement

It gives us pleasure in presenting the project report on **'Brain Tumor Classification Using Deep Learning Algorithms'**.

Firstly, we would like to express our indebtedness appreciation to our guide **Mr. Santosh Nagargoje**. His constant guidance and advice played very important role in successful completion of the project. He always gave us his suggestions, that were crucial in making this report as flawless as possible.

We would like to express our gratitude towards **Prof. Dr. Mrs. S. A. Itkar** Head of Computer Engineering Department, PES Modern College of Engineering for her kind co-operation and encouragement which helped us during the completion of this report.

Also we wish to thank our Principal, **Prof. Dr. Mrs. K. R. Joshi** and all faculty members for their whole hearted co-operation for completion of this report. We also thank our laboratory assistants for their valuable help in laboratory.

Last but not the least, the backbone of our success and confidence lies solely on blessings of dear parents and lovely friends.

Ankita Kadam
Sartaj Bhuvaji
Prajakta Bhumkar
Sameer Dedge

Abstract

A Brain tumor is considered as one of the aggressive disease, among children and adults. Brain tumors account for 85 to 90 percent of all primary Central Nervous System(CNS) tumors. Every year, around 11,700 people are diagnosed with a brain tumor. The 5-year survival rate for people with a cancerous brain or CNS tumor is approximately 34 percent for men and 36 percent for women. Brain Tumors are classified as: **Benign Tumor, Malignant Tumor, Pituitary Tumor, etc.** Proper treatment, planning and accurate diagnostics should be implemented to improve life expectancy of the patients.

The best technique to detect brain tumor is **Magnetic Resonance Imaging (MRI)**.A huge amount of image data is generated through the scans. These images are examined by the radiologist. Manual examination can be error-prone due to the level of complexities involved in brain tumors and their properties.

Application of automated classification techniques using **Machine Learning(ML) and Artificial Intelligence(AI)** has consistently shown higher accuracy than manual classification. Hence, we propose performing detection and classification by using Deep Learning Algorithms using Convolution Neural Network (CNN)[1], Artificial Neural Network (ANN)[2] and Transfer Learning(TL)[23] to achieve higher accuracy.

The MRI images are classified using different **'Deep ANN, CNN, TL models'**. These models have permutations and combinations of different **'Network Parameters'**. The model(s) with higher accuracy, lower loss and a good F1-Score is/are selected and deployed.

The aim of the project is to achieve higher accuracy and reliability for real world MRI data using AI and ML domain knowledge. Further to accurately indicate the position of the tumor and to provide some suggestions for treatment by providing ease of access to the software through cloud and mobile applications, web platforms.

Keywords

AI ..... Artificial Intelligence

ANN ..... Artificial Neural Network

AWS ..... Amazon Web Services

CNN ..... Convolution Neural Network

CNS ..... Central Nervous System

DNN ..... Deep Neural Network

EC2 ..... Elastic Compute Cloud

FPN ..... Feature Pyramid Networks

ML ..... Machine Learning

MRCNN ..... Mask Region-based CNN

MRI ..... Magnetic Resonance Imaging

NN ..... Neural Network

ROI ..... Region of interest

TL ..... Transfer Learning

URL ..... Uniform Resource Locator

Contents

List of Figures

List of Tables

CHAPTER 1
INTRODUCTION

1.1 Overview

1.2 Motivation

Brain Tumors are complex. There are a lot of abnormalities in the sizes and location of the brain tumor(s). This makes it really difficult for complete understanding about the nature of the tumor. Also, a professional Neurosurgeon is required for MRI analysis. Often times in developing countries the lack of skillful doctors and lack of knowledge about tumors makes it really challenging and time-consuming to generate reports from MRI's. So an automated system on Cloud[11] can solve this problem. The automated system would be able to resolve if the MRI has a tumor and the system would be able to pin-point the exact type of tumor identified.[3] The system would also be able to the locate position of the tumor in a given MRI. Providing a list of doctors, suggestions and further medical procedures would help eliminate any confusion among patients.

1.3 Problem Definition and Objectives

1.3.1 Definition

To Detect and Classify Brain Tumor using **ANN, CNN and TL** as an asset of Deep Learning and to examine the tumor position.

1.3.2 Objectives

(a) To classify MRI as: MRI with tumor or MRI without tumor.

(b) To classify the tumor into one of the four classes.

(c) To show exact position of tumor in the MRI.

(d) To provide information on symptoms and treatment.

(e) To provide information about available doctors.

1.4 Project Scope and Limitations

(a) **Scope :**

i. We cleaned and processed the available data and build multiple ANN,CNN and TL models.

ii. Selecting the best models with tuned hyper-parameters for tumor classification .

iii. AWS deployment.

iv. Display position of the tumor.

v. Reduction in Medical negligence or Human error.

(b) **Limitations :**

i. MRI can only be in 'jpg', 'png' formats.

ii. Random images other than MRI are not treated as invalid.

iii. The System can only classify MRI as per tumor class and accounts no weight to other medical conditions.

1.5 Methodologies for Problem Solving

1.5.1 Underfitting

1. **Description :**
   Underfitting occurs when the model has not trained enough. The causes for underfitting are a large model, a small data set size, improper hyper-parameters, inappropriate loss function.

2. **Value :**
   An under-fit model has lower accuracy and cannot be used in a system. Thus a model which cannot correctly classify the MRI is unusable and is considered as a failure.

3. **Solution :**
   Increase the size or number of parameters in the ML model. Increase the complexity of the model. Increasing the training time until the cost function in ML is minimized.

1.5.2 Overfitting

1. **Description :**
   Overfitting occurs when you achieve a good fit of your model on the training data, while it does not generalize well on new, unseen data. In other words, the model learned patterns specific to the training data, which are irrelevant in other data.

2. **Value :**
   Over-fit model is too specific and cannot predict on unseen data. While in real world, MRI is never a part of training set, hence model is inaccurate in classifying those real world MRI's. This is termed as failure.

3. **Solution :**
   Train with more data, Cross-validation, Early stopping, Regularization, Ensembling helps in eliminating overfitting.

1.5.3 Inadequate Infrastructure

1. **Description :**
   Inadequate Infrastructure in terms of GPU power causes models to take a lot of time to train. Also GPU with a lower RAM configuration crashes when it tries to load huge models and data sets.

2. **Value :**
   A system that cannot load the complete dataset proves incapable of handling the scale of the project. Also ML models take a lot of RAM and time while training. Waiting for hours for a model to train is not practically possible.

3. **Solution :**
   Using GPU providers like Google Colab, AWS Sagemaker would provide us with the necessary infrastructure need for the project. GPU there can be as large as 50GB which would be sufficient for our application.

CHAPTER 2
LITERATURE SURVEY

2.1 Literature Survey

No.	Research Papers/Journal	Authors	Publications	Comments/Analysis /Problems
1	Multi Classification Of Brain Tumor Images	Hossam H. Sultan, Nancy M. Salem, Walid Al-Atabany	IEEE 2019	CNN Based Classification With Augmentation
2	Brain Tumor Classification Using Convolutional Neural Networks	J.Seetha, S.Selvakumar Raja	Biomedical Pharmacology Journal, September 2018.	CNN Based Classification
3	Deep Learning based brain tumor classification and detection system	Ali Ari, Davut Hanbay	Turkish Journal of Electrical Engineering and Computer Sciences 2018	ELR-LRM Based Classification
4	A Review on Image Processing and Image Segmentation.	Jiss Kuruvilla,Dhanya Sukumaran,Anjali Sankar,Siji P Joy	IEEE,2016	Overview of Image Processing and Image Segmentation
5	Brain Tumor Segmentation Using Deep Learning by Type Specific Sorting of Images	Zahra Sobhaninia, Safiyeh Rezaei, Alireza Noroozi, Mehdi Ahmadi, Hamidreza Zarrabi, Nader Karimi, Ali Emami,	Research Gate 2019	Sorting Of Images 1. Sagittal View 2.Axial View 3.Coronal View

Table 2.1: Literature Survey

CHAPTER 3

SOFTWARE REQUIREMENTS SPECIFICATION

3.1 Assumptions and Dependencies

1. **Assumptions :**
   (a) Proper MRI images.
   (b) Updated Web Browser.
   (c) Android Phone(Kit-kat v(4.4) or higher).
2. **Dependencies :**
   (a) NVIDIA Cuda Toolkit(v9.0)
   (b) NVIDIA cuDNN(v7.6.5)
   (c) TensorFlow(v1.13.1)
   (d) Keras(v2.1.0)
   (e) Flask(v1.1.1)

3.2 Functional Requirements

1. **Open Architecture :**
   There is no standard or uniform infrastructure platform. The key consideration is whether the analytic solution works with multiple platforms.
2. **Alert Generation :**
   When a machine degradation or potential asset failure is detected, this is communicated to the relevant facility stakeholders.
3. **Human Error Correction :**
   The system confirming if the input from the user is valid. An invalid input will generate invalid results.

3.2.1 System Feature

1. **Availability :**
   The system deployed on the cloud platform is available to use all round the clock. There are no down-times to the system.
2. **Consistency :**
   The system outputs results that are consistent. The result does not change for a MRI no matter at what time it is uploaded. The system can also treat's all image formats equally.

3. **Free for all :**
   The project is free for everyone to use. All doctors and patients can access the complete system. No limitations on the number of tests or type of user is implemented.

3.3 External Interface Requirements

3.3.1 User Interfaces

1. **Front-end Software :**
   Mobile Operating System (Android/IOS), Web Browser.

2. **Back-end Software :**
   Anaconda Navigator (v5.3.0), Colaboratory Interface, Amazon EC2.

3.3.2 Hardware Interfaces

1. **Computer :**
   A computer to store images and access the system.

2. **Amazon Web Services :**
   AWS servers along with their specific hardware to develop and deploy the system.

3. **Android or iOS smartphones :**
   Android or ios devices to access the system through mobile applications.

4. **Heroku :**
   Heroku servers for website deployment.

3.3.3 Software Interfaces

1. **Operating system :**
   We have chosen Windows operating system for its best support and user-friendliness.

2. **File System :**
   We use a file system to store MRI images.

3. **BootStrap :**
   It contains CSS and JavaScript based design templates for typography, forms, buttons, navigation and other interface components and

JavaScript-based design templates for typography, forms, buttons, navigation and other interface components.

3.3.4 Communication Interfaces

1. Mobile Application (Android or iPhone).
2. Web Browsers.
3. Wifi.
4. Ethernet.

3.4 Nonfunctional Requirements

3.4.1 Performance Requirements

1. **Workload :**
   The software system consists of a large amount of data-sets of MRI images (.jpeg/.jpg/.png format).
2. **Scalability :**
   The software system should be able to process data above the system's workload.
3. **Platform :**
   The software system should be compatible with multiple mobile Operating Systems (Android, iOS).
4. **Reliability :**
   The system should be reliable and should provide accurate results. The information on the estimated cost and list of doctors should be appropriate.

3.4.2 Safety Requirements

1. Data-set Verification.
2. Bug-free Mobile Applications.

3.4.3 Security Requirements

1. Authorized AWS Login.
2. HTTPS enabled servers.

3.4.4 Software Quality Attributes

1. **Accuracy :**
   The software must accurately identify the tumor present in MRI and segregate the tumor into different types.
2. **Portability :**
   The software system should be available on different smart phones (Android and iPhones).
3. **Robustness :**
   Robustness reduces the impact of operational mistakes, erroneous input data and hardware errors.

3.5 System Requirements

3.5.1 Software Requirements

1. Anaconda Navigator (v5.3.0)
2. Jupyter Notebook (v6.0.1)
3. Python (v3.6) (64bit)
4. Google Colab
5. Android Studio (v3.5)
6. Swift Studio (v4.1)
7. Cuda Toolkit (v9.0)
8. Flask Framework (v1.1.1)

3.5.2 Hardware Requirements

1. Graphic Card : Nvidia's Titan V(12 GB RAM(GDDR6)), Tesla K80(12 GB RAM)
2. CPU : Intel
3. RAM : 25 GB (DDR4)
4. Other: Network Card.
5. iPhone / Android phone

3.5.3 Data Set Requirements

The data required for the project is MRI data which is gathered from the internet. The data is cleaned for any outliers. The data set is uploaded on Kaggle.com for use in ML community and other enthusiasts.

URL : www.kaggle.com/sartajbhuvaji/brain-tumor-classification-mri

3.6 Analysis Models : SDLC Model

We have used the following model of Software Development Life Cycle .

Incremental Model :

Incremental Model is a process of software development where requirements divided into multiple standalone modules of the software development cycle. In this model, each module goes through the requirements, design, implementation and testing phases. Every subsequent release of the module adds function to the previous release. The process continues until the complete system achieved.

The modules implemented are:

Module 1 : ANN Model Building.

Module 2 : CNN Model Building.

Module 3 : TL Model Building.

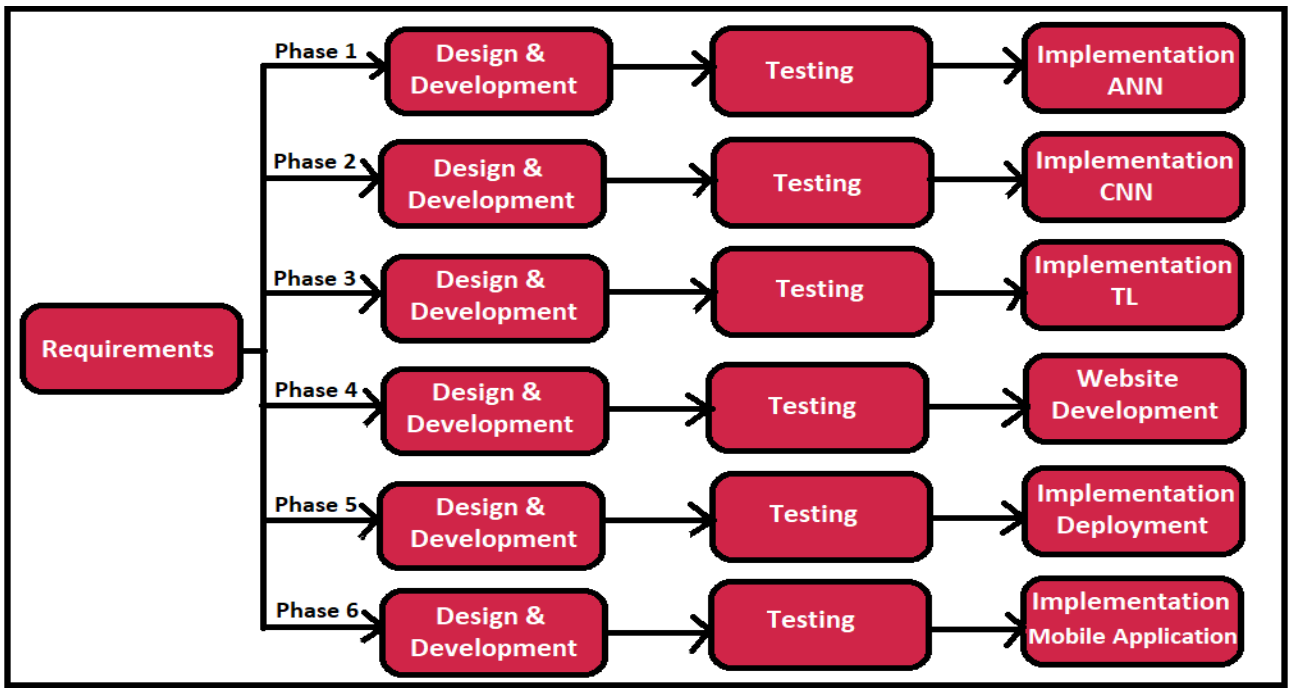


Figure 3.1: Incremental Model

Module 4 : Website Development.

Module 5: Deployment(Cloud Platform:AWS and Heroku)

Module 6: Mobile Application Development

The System is built after the implementation of all the 6 Modules.

CHAPTER 4
SYSTEM DESIGN

4.1 System Architecture

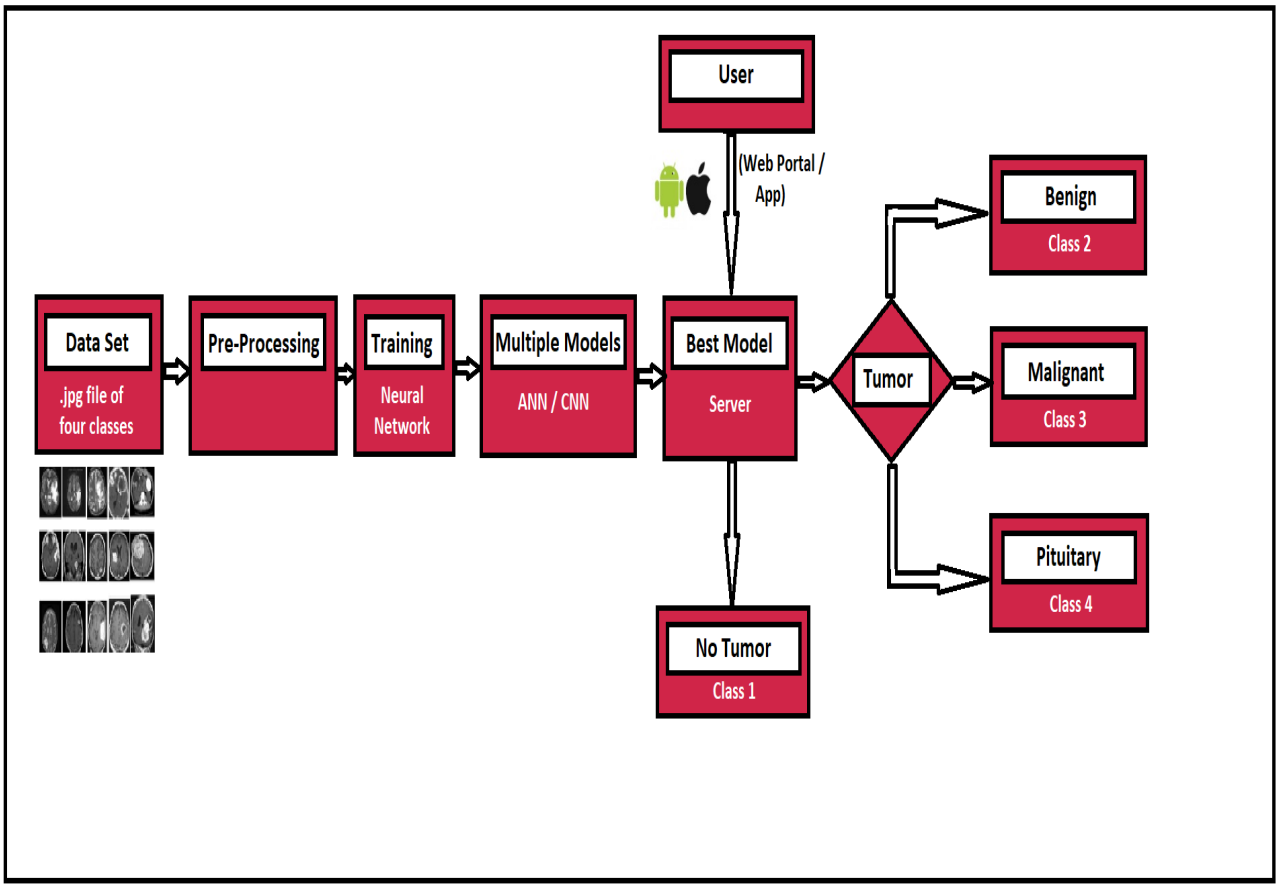


Figure 4.1: Architecture Diagram (Fig.1)

- **Data-set Block :**
  This block represents the data collection process. Image data is gathered and validated in this process block.[6]
- **Pre-Processing Block :**
  The validated data is cleaned and augmented. Various data pre processing algorithms like edge detection, padding are applied on data.[4]
- **Training :**
  Various ANN and CNN models are trained on pre processed data. The models are monitored for accuracy and many other parameters.

- **Multiple Models :**
  Training generates multiple models. The best model is selected based on multiple parameters. Alex Net.[15], Res Net.[16], Image Net.[17] , custom model net are all tested. The best model has the highest accuracy value and lowest loss value, etc.

- **Best model :**
  The best model[8] is hosted on the cloud along with other python scripts. The model accepts an image as input from the user and generates output as class titles.

- **User :**
  The user connects to the model using web service or mobile applications. The user sends his/her MRI to the model and expects a class label as an answer.

- **No Tumor :**
  This is our class 1 of output. This indicated no tumor was found in the MRI.

- **Benign :**
  This is our class 2 of output. This indicated benign tumor(s) were found in the MRI.

- **Malignant :**
  This is our class 3 of output. This indicated malignant tumor(s) were found in the MRI.

- **Pituitary :**
  This is our class 4 of output. This indicated pituitary tumor(s) were found in the MRI.

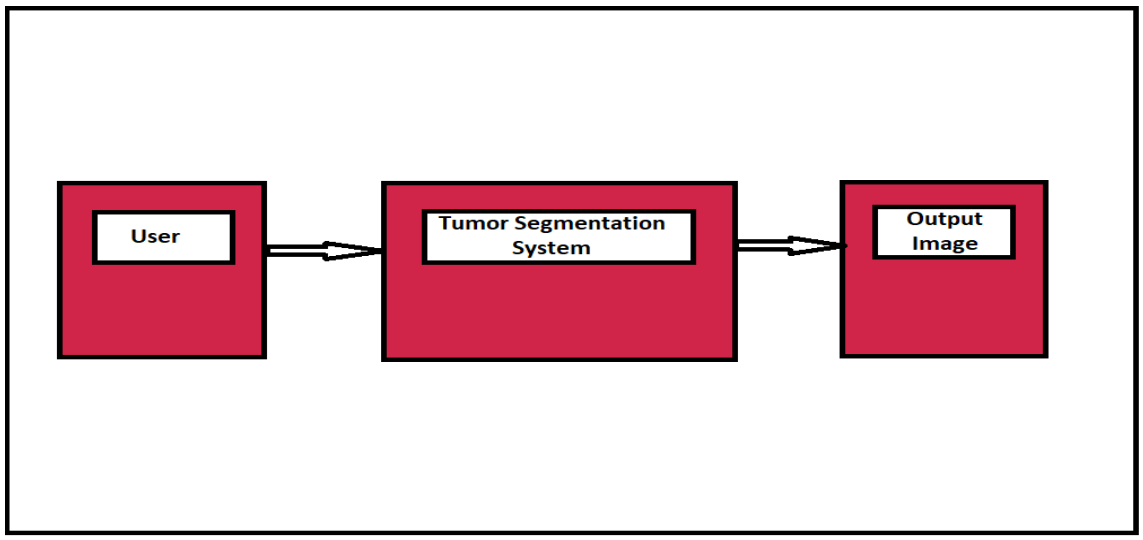


Figure 4.2: Architecture Diagram (Fig.2)

- **User :**
  The user inputs a MRI to the system. The MRI so given surely belongs to class 2 ,3 or 4.

- **Tumor Segmentation System :**
  This system receives the MRI and it segments the tumor from rest of the image. A rectangle is drawn around tumors and the area of tumor is colored.

- **Output Image :**
  The output image has the tumor colored and marked inside a rectangle to indicate size of the tumor. Also a probability value is specified to indicate how sure the system is about the segmentation.

4.2 Model Architectures

4.2.1 Best Artificial Neural Network Model

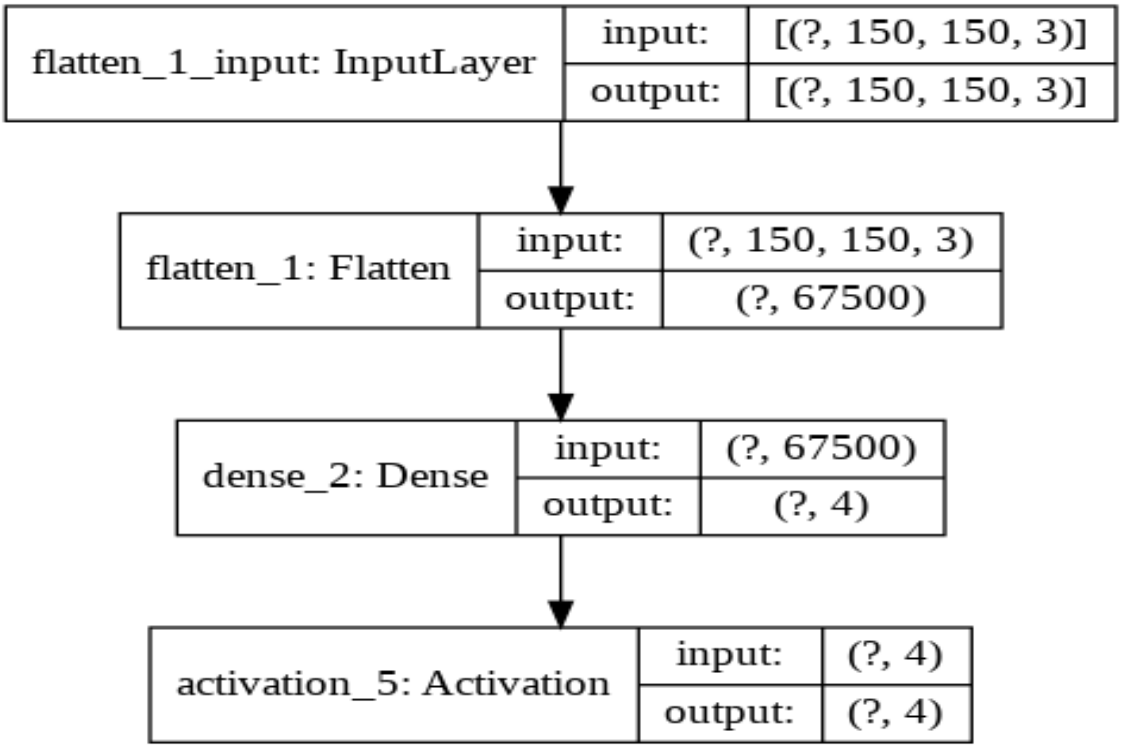


Figure 4.3: ANN Architecture

- **Input Layer Shape :** (150,150,3)
- **Number of Convolution Layers :** 0
- **Number of Dense Layers :** 1
- **Total Number of Parameters :** 270,004
- **Model Accuracy :** 80 percent
- **Model Loss :** 181.14
- **Model F1-Score :** 80

4.2.2 Best Convolution Neural Networks Model

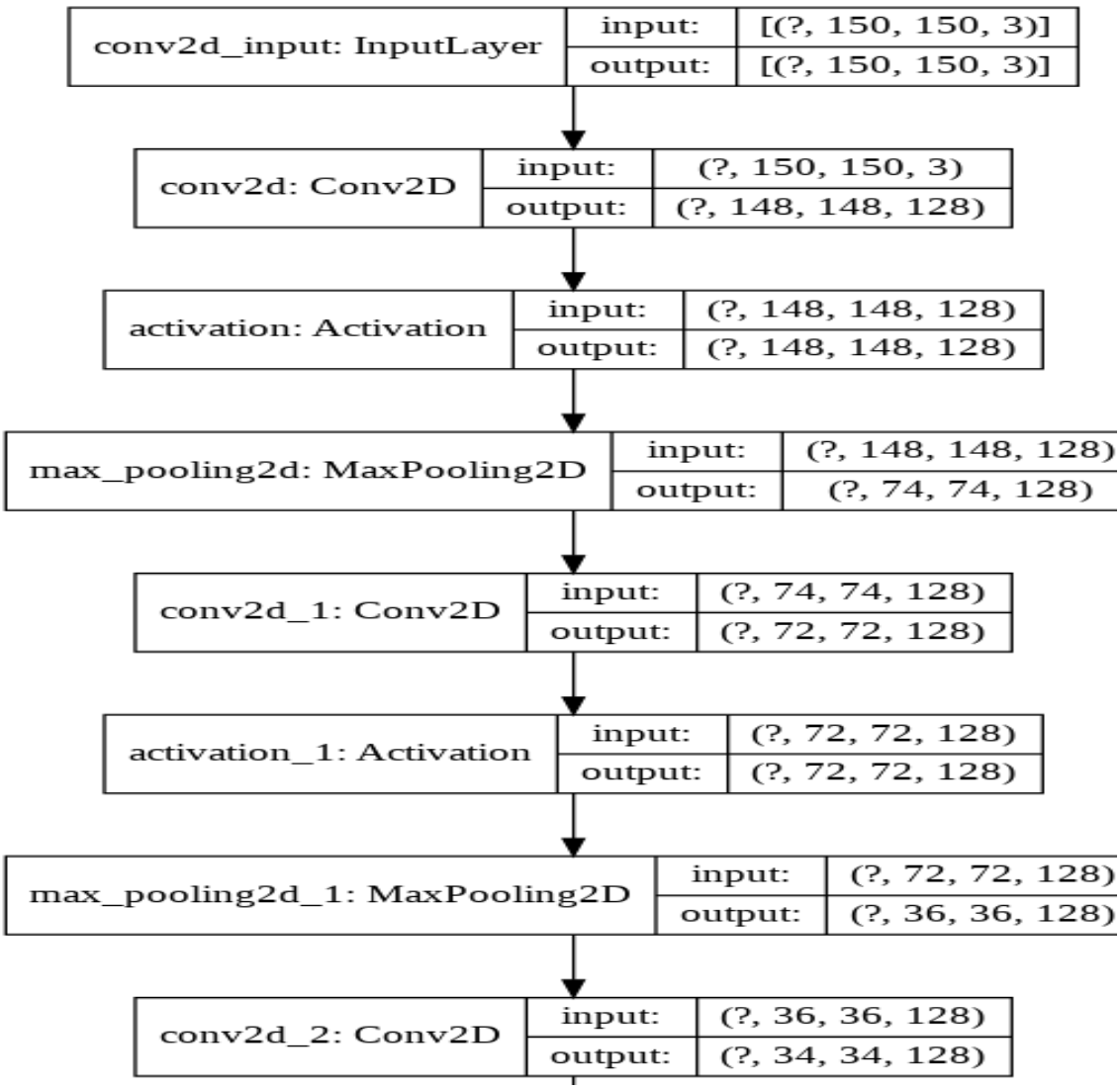


Figure 4.4: CNN Architecture(Fig:1)

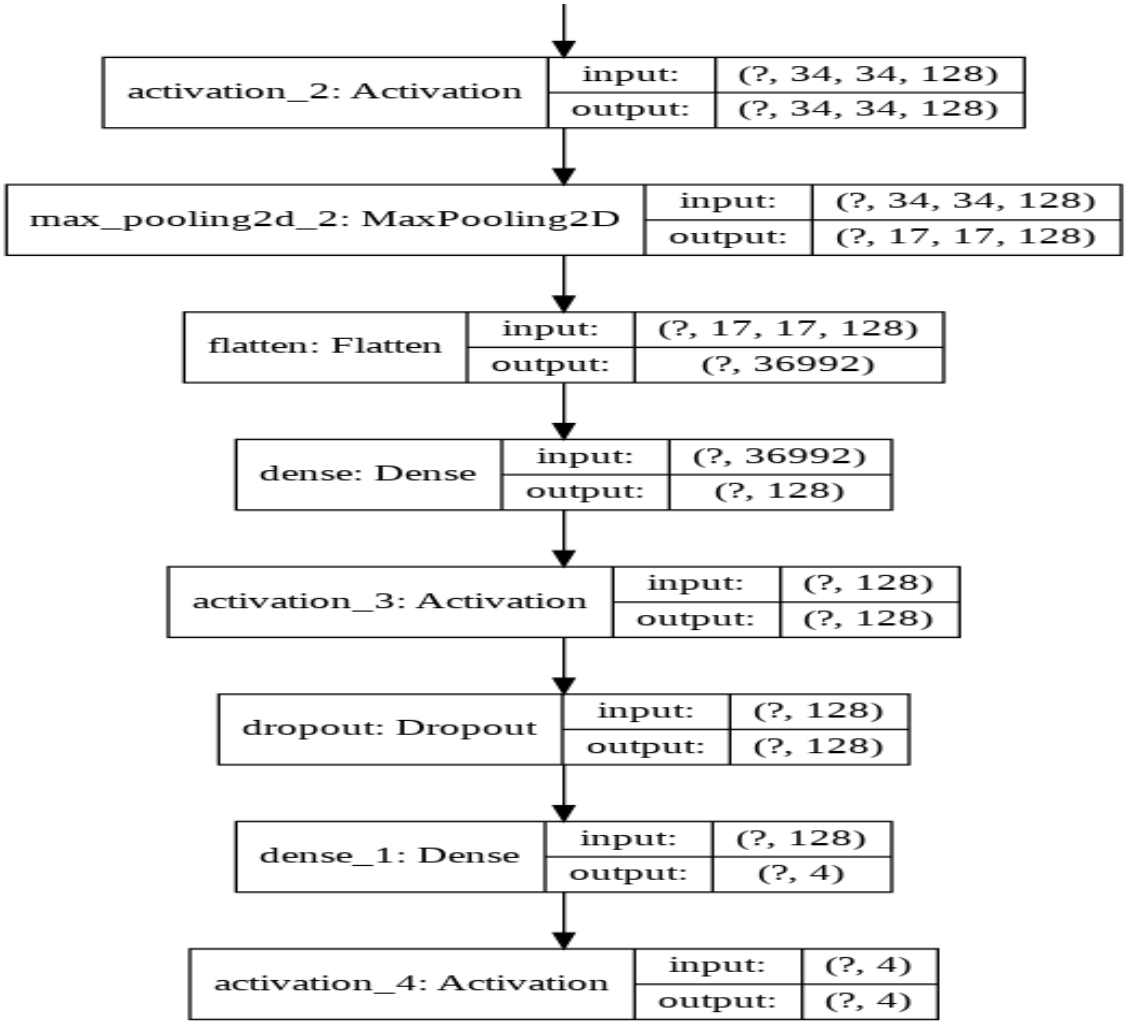


Figure 4.5: CNN Architecture(Fig:2)

- **Input Layer Shape :** (150,150,3)
- **Number of Convolution Layers :** 3
- **Number of Dense Layers :** 1
- **Total Number of Parameters :** 5,034,372
- **Model Accuracy :** 90 percent
- **Model Loss :** 0.43
- **Model F1-Score :** 91

4.2.3 Best Transfer Learning Model (VGG16)

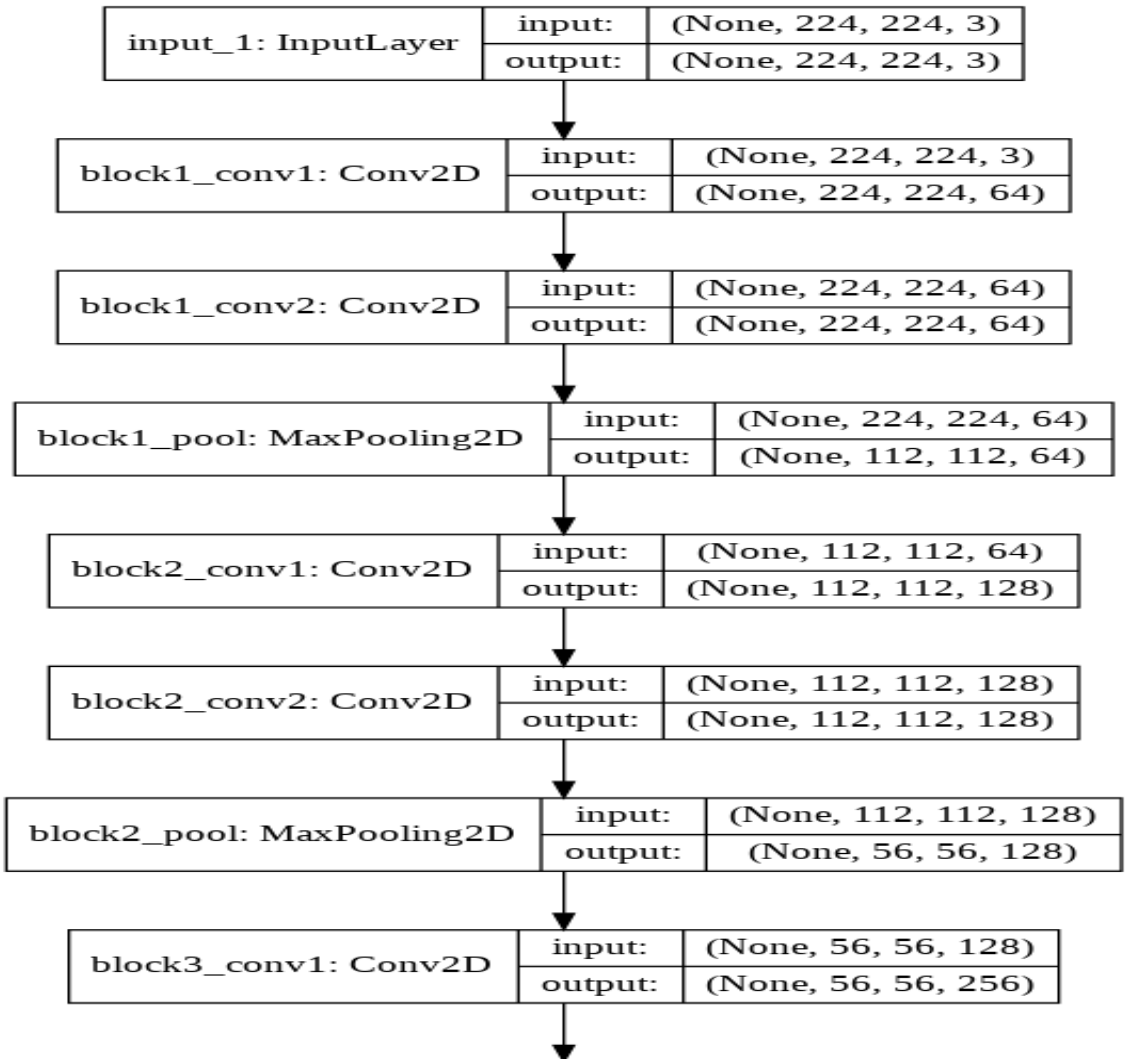


Figure 4.6: TL Architecture(Fig:1)

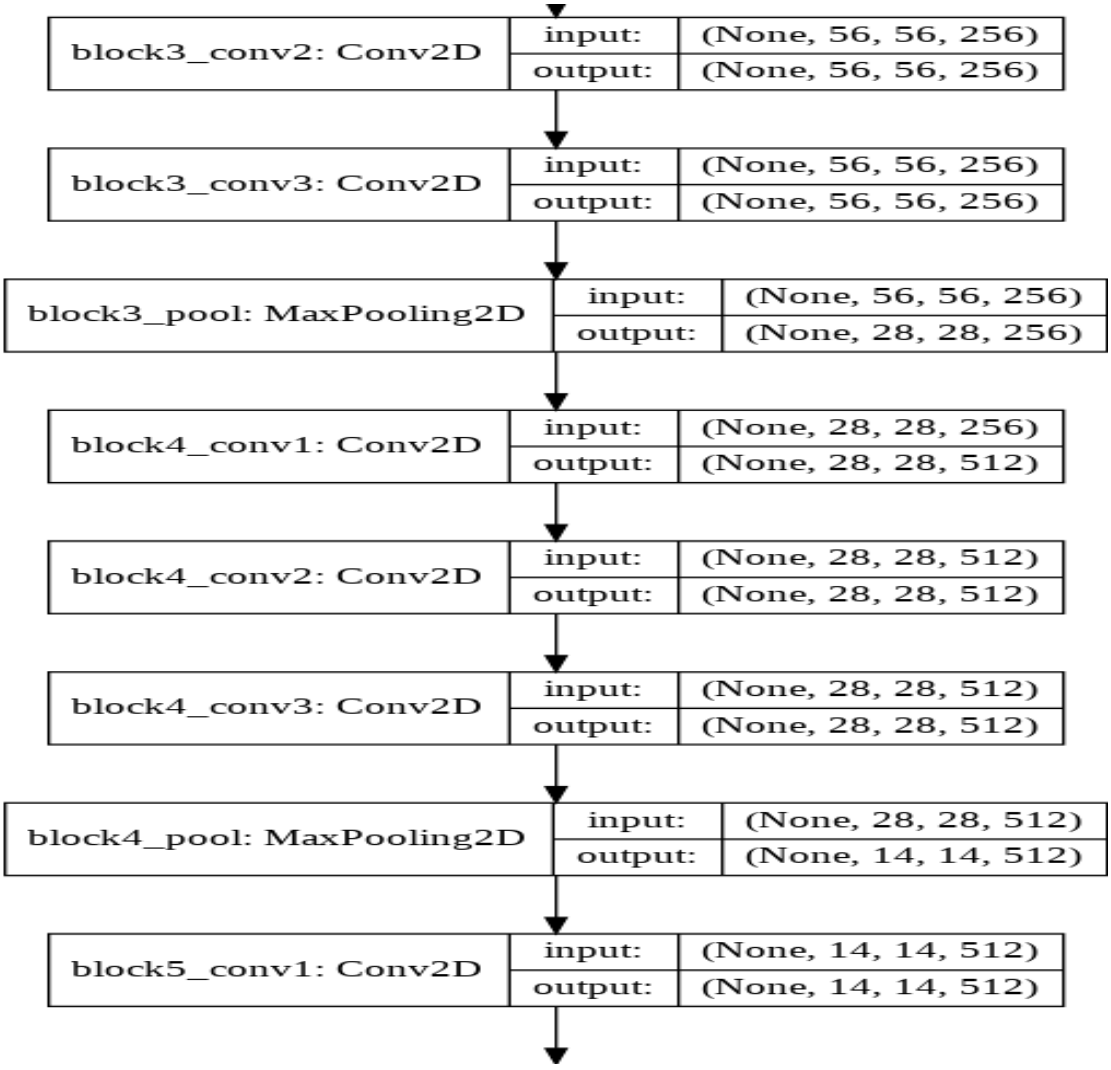


Figure 4.7: TL Architecture(Fig:2)

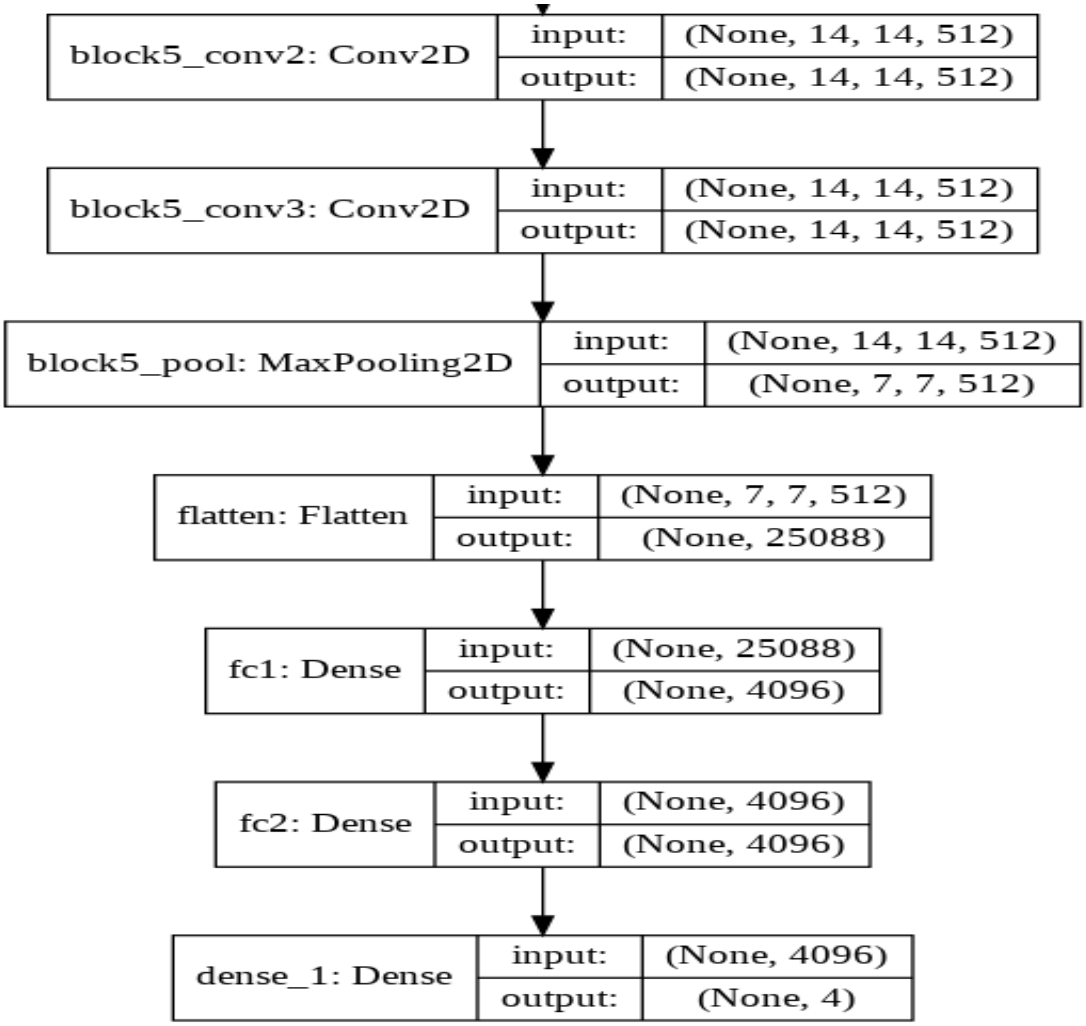


Figure 4.8: TL Architecture(Fig:3)

- **Input Layer Shape :** (224,224,3)
- **Number of Convolution Layers :** 13
- **Number of Dense Layers :** 2
- **Total Number of Parameters :** 134,276,932
- **Model Accuracy :** 94 percent
- **Model Loss :** 0.89
- **Model F1-Score :** 94

4.3 Common Attributes

All ANN, CNN, TL models had the following attributes and values unchanged:

- **Loss :** Sparse Categorical Cross-entropy
- **Optimizer :** Adam
- **Epochs :** 20
- **Batch Size :** 64
- **Learning Rate :** 0.001
- **Regularization :** None
- **Activation :** ReLu
- **Final Activation :** Softmax
- **Drop Out :** 33 percent
- **Call Backs :** Tensorboard, Early stopping

4.4 Visualizing intermediate activation

In this technique[21], given an input image, we will simply plot what each filter has extracted (output features) after a convolution operation in each layer. Eg. In VGG16, the input layer dimension is 224x224x3 and the output dimension after the first convolution operation is 224x224x64 (block1-conv1). Here, 64 is the number of filters that are used to extract input features after 1st convolution operation, so we will just plot these sixty-four 224x224 outputs.

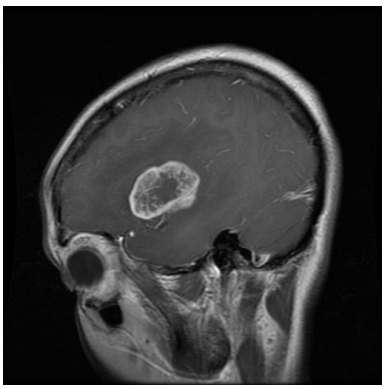


Figure 4.9: Input

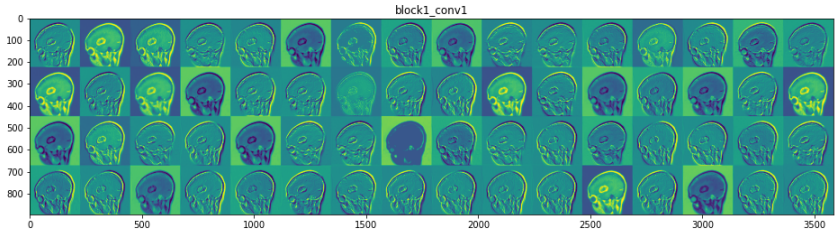


Figure 4.10: block1 conv1

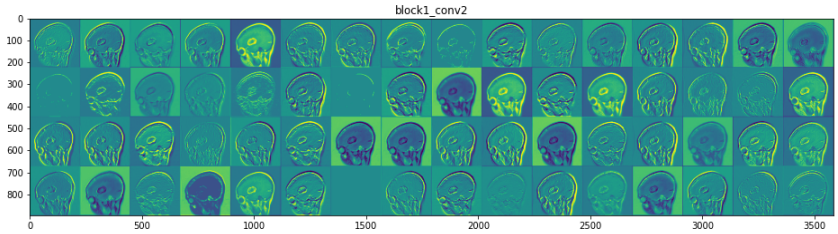


Figure 4.11: block1 conv2

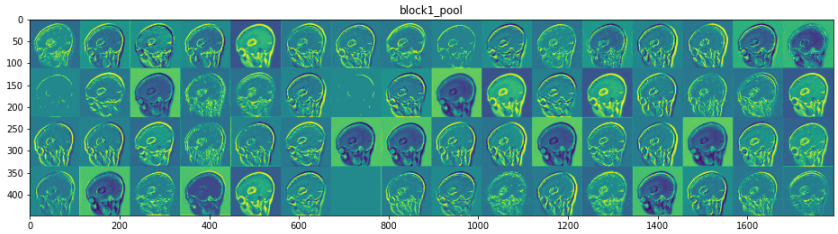


Figure 4.12: block1 pool

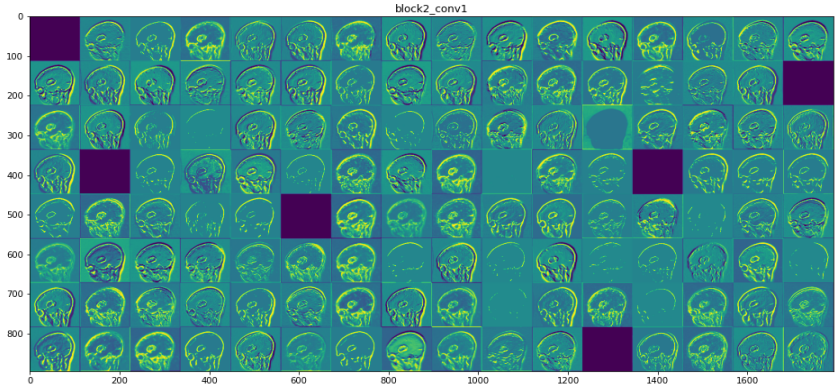


Figure 4.13: block2 conv1

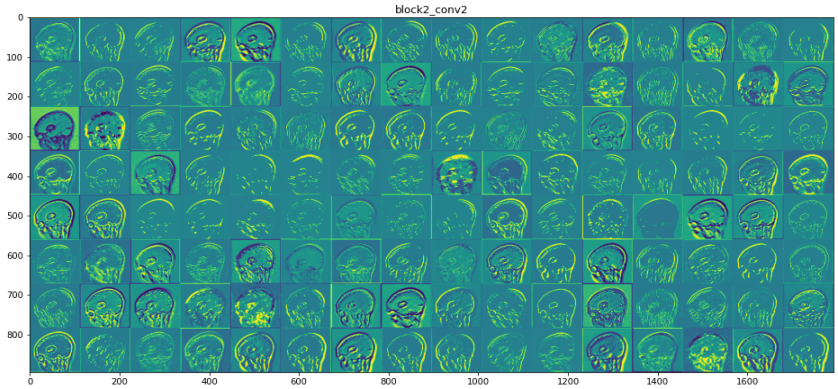


Figure 4.14: block2 conv2

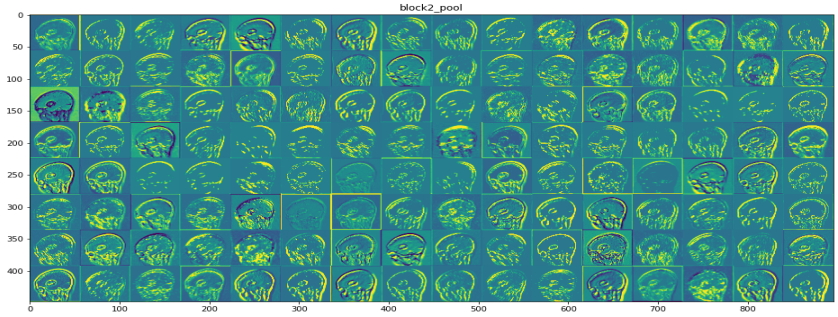


Figure 4.15: block2 pool

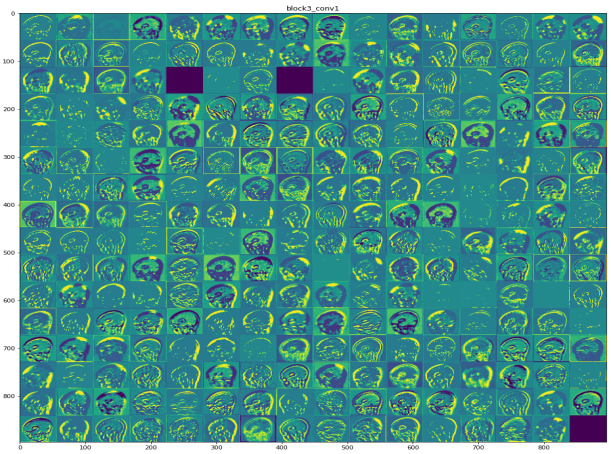


Figure 4.16: block3 conv1

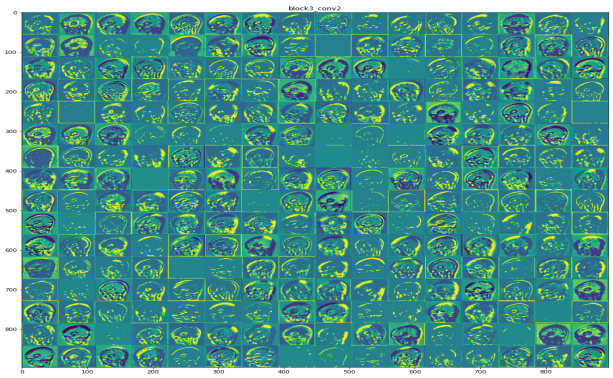


Figure 4.17: block3 conv2

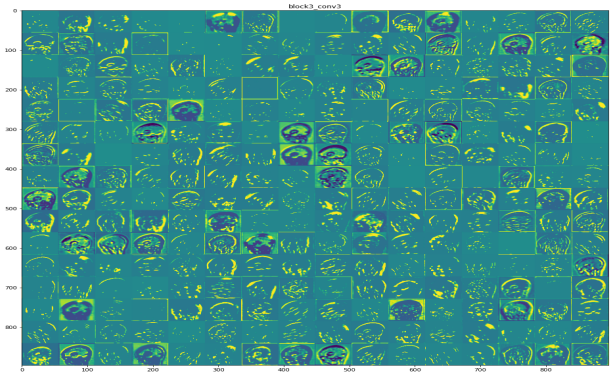


Figure 4.18: block3 conv3

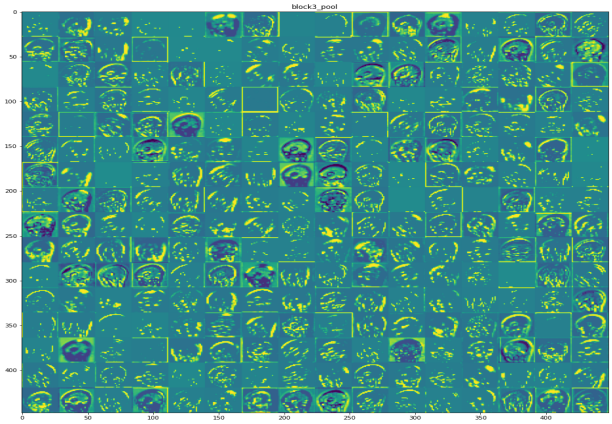


Figure 4.19: block3 pool

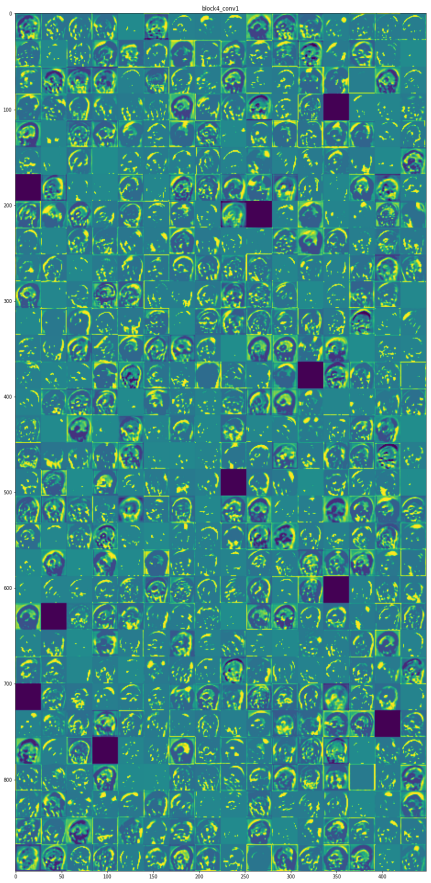


Figure 4.20: block4 conv1

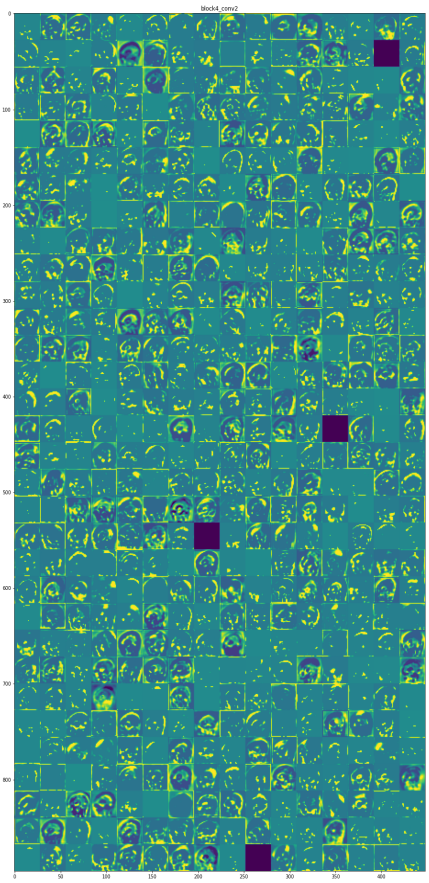


Figure 4.21: block4 conv2

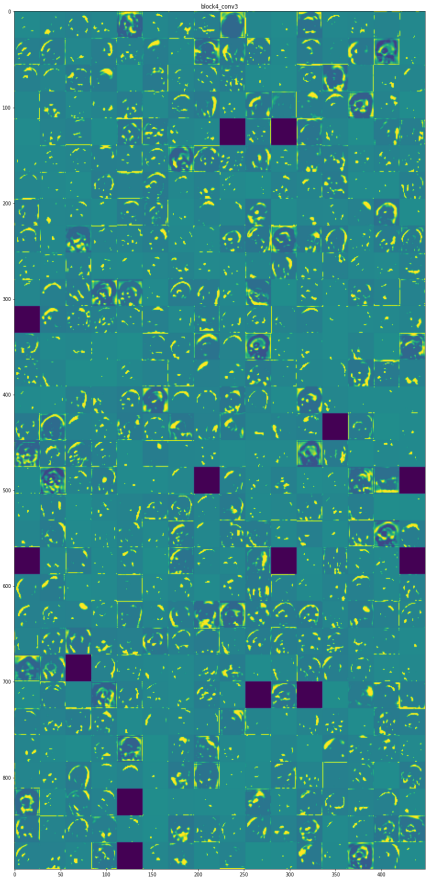


Figure 4.22: block4 conv3

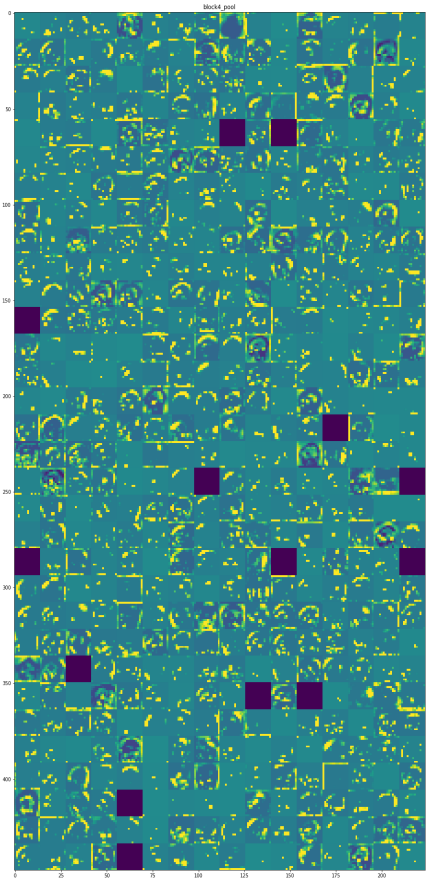


Figure 4.23: block4 pool

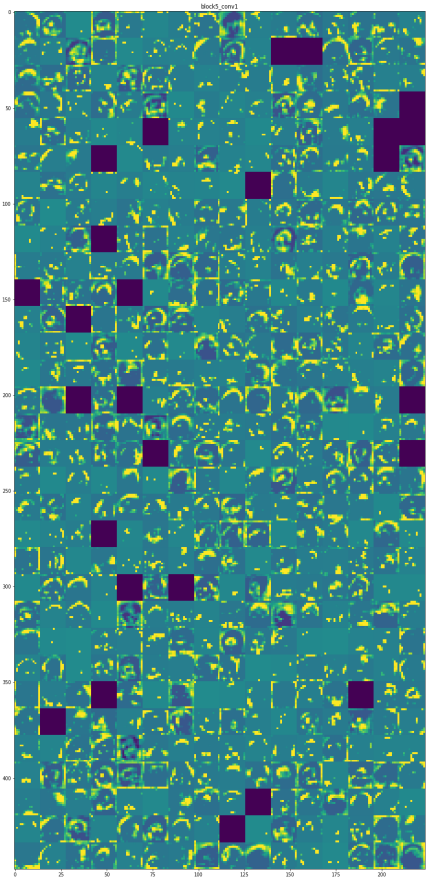


Figure 4.24: block5 conv1

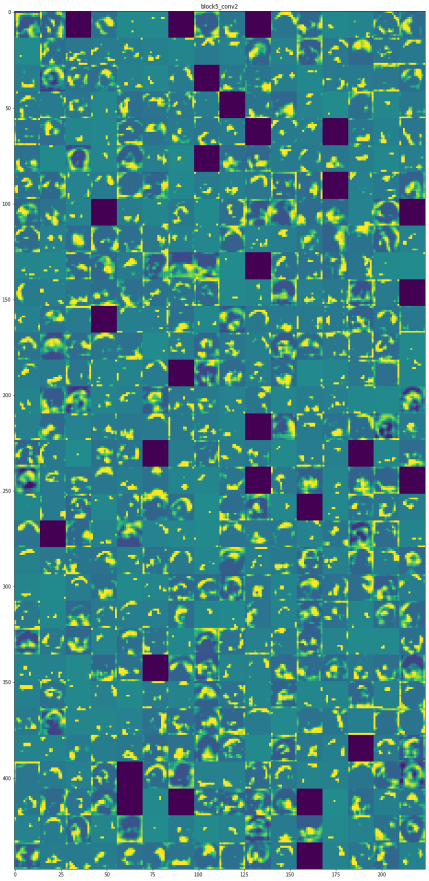


Figure 4.25: block5 conv2

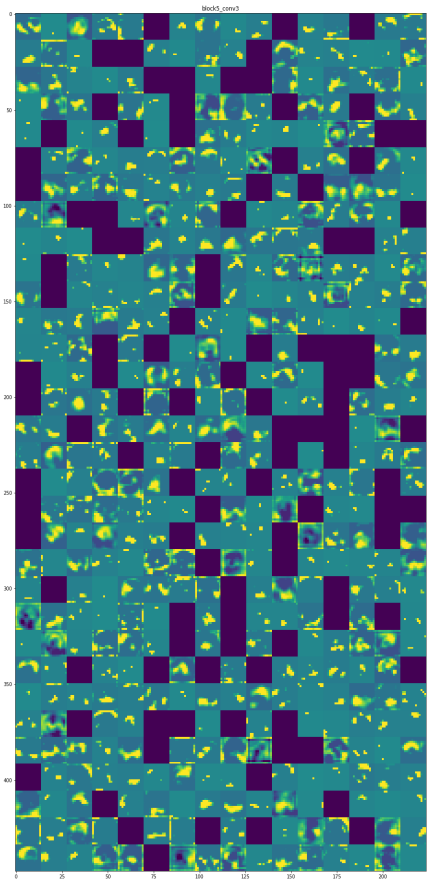


Figure 4.26: block5 conv3

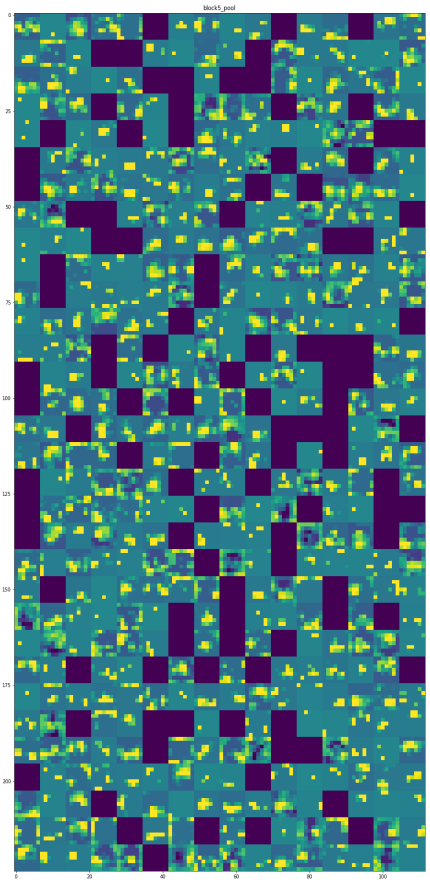


Figure 4.27: block5 pool

4.4.1 Interpretations

- The initial layers (block1 and block2) retain most of the input image features. It looks like the convolution filters are activated at every part of the input image. This gives us an intuition that these initial filters might be primitive edge detectors (since we can consider a complex figure to be made up of small edges, with different orientations, put together.)
- As we go deeper (block3 and block4) the features extracted by the filters become visually less interpretative. An intuition for this can be that the convnet is now abstracting away visual information of the input image and trying to convert it to the required output classification domain.
- In block5 (especially block5-conv3) we see a lot of blank convolution outputs. This means that the pattern encoded by the filters were not found in the input image. Most probably, these patterns must be complex shapes that are not present in this input image.

4.5 Mask R-CNN Architecture

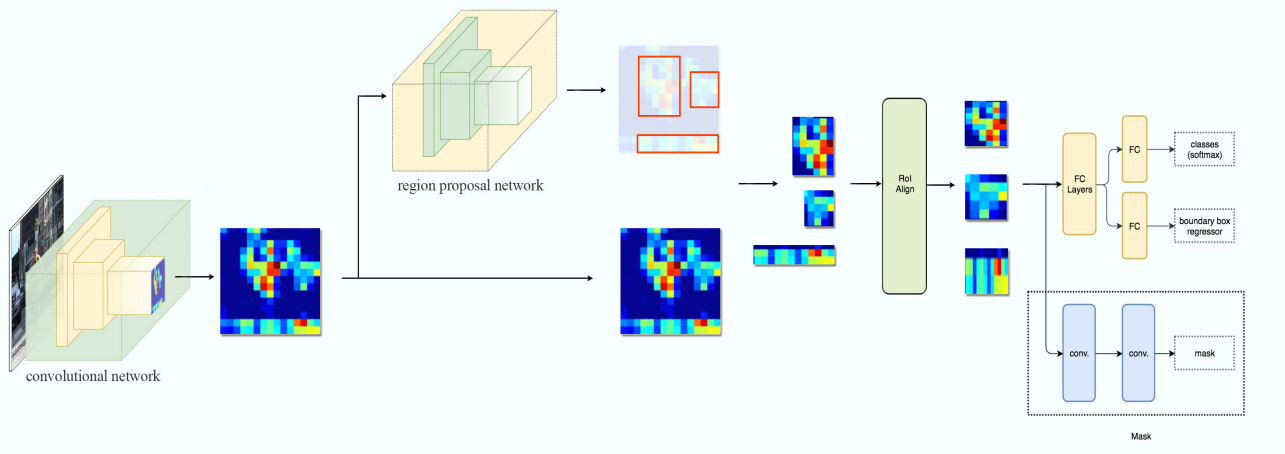


Figure 4.28: M R-CNN

Mask R-CNN[18], extends Faster R-CNN by adding a branch for predicting segmentation masks on each Region of Interest (ROI), in parallel with the existing branch for classification and bounding box regression. The mask branch is a small FCN applied to each ROI, predicting a segmentation mask in a pixel-top pixel manner. Mask R-CNN is simple to implement and train given the Faster R-CNN framework, which facilitates a wide range of flexible architecture designs. Additionally,the mask branch only adds a small computational overhead, enabling a fast system and rapid experimentation.

4.5.1 Region proposal network

The region proposal network (RPN) in the faster region-based convolutional neural network (Faster R-CNN)[19] is used to decide "where" to look in order to reduce the computational requirements of the overall inference process. The RPN quickly and efficiently scans every location in order to assess whether further processing needs to be carried out in a given region. It does that by outputting k bounding box proposals each with 2 scores representing probability of object or not at each location.

The anchor boxes are just references, they are selected to have different aspect ratios and scales in order to accommodate different types of objects, elongated objects like buses, for example, cannot be properly represented by a square bounding box. In Faster R-CNN they used k = 9 representing 3 scales and 3 aspect ratios. Each regressor in the RPN it only computes 4 offset values (w, h, x, y) to the corresponding reference anchor box.

where w = width, h = height, (x,y) = center

The RPN uses a 3x3 window that slides over a high-level convolutional feature map, the effective size of that small window is actually 177x177 when re-projected back to the input layer, so the RPN is actually using a lot of context when making the proposals. This 3x3 window is re-sampled to a 256 dimensional vector before feeding into two fully connected layers, a box regression layer, that computes the box offsets, and the box classification layer(cls) that computes the confidence scores that are related to probability of objectness. The regression layer has 4k outputs while the cls layer has 2k outputs making the total RPN output per position to 4k + 2k.

4.5.2 Region of Interest Pooling

Region of Interest (ROI) pooling is used for utilising single feature map for all the proposals generated by RPN in a single pass. ROI pooling solves the problem of fixed image size requirement for object detection network. ROI pooling produces the fixed-size feature maps from non-uniform inputs by doing max-pooling on the inputs. The number of output channels is equal to the number of input channels for this layer. ROI pooling layer takes two inputs:

1. A feature map obtained from a Convolutional Neural Network after multiple convolutions and pooling layers.
2. 'N' proposals or Region of Interests from Region proposal network. Each proposal has five values, the first one indicating the index and the rest of the four are proposal coordinates. Generally, it represents the top-left and bottom-right corner of the proposal.

4.6 Mathematical Model

4.6.1 Convolution

Kernel convolution is not only used in CNNs, but is also a key element of many other Computer Vision algorithms. It is a process where we take a small matrix of numbers (called kernel or filter), we pass it over our image and transform it based on the values from filter. Subsequent feature map values are calculated according to the following formula.

2.

- An image matrix (volume) of dimension **(h x w x d)**
- A filter **(f_h x f_w x d)**
- Outputs a volume dimension **(h - f_h + 1) x (w - f_w + 1) x 1**

Figure 4.29: Convolution

4.6.2 Pooling

Pooling layers provide an approach to down sampling feature maps by summarizing the presence of features in patches of the feature map. A pooling layer is a new layer added after the convolution layer. The pooling operation is specified, rather than learned. Two common functions used in the pooling operation are:

- Average Pooling
- Maximum Pooling

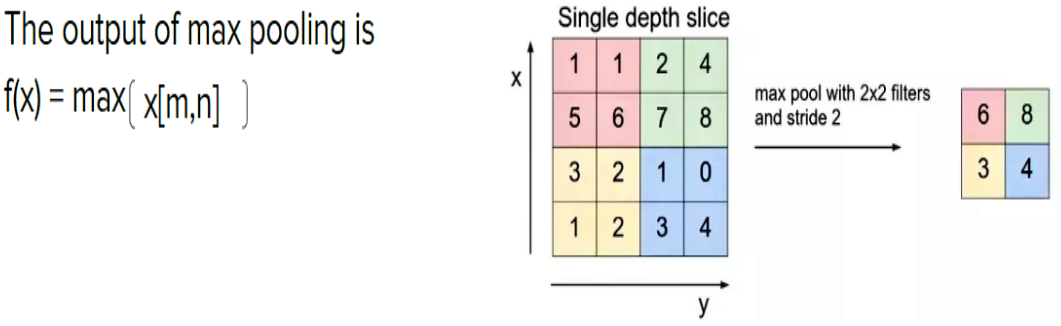


Figure 4.30: Max Pooling

4.6.3 ReLU

ReLU stands for rectified linear unit, and is a type of activation function. ReLU is linear for all positive values, and zero for all negative values.

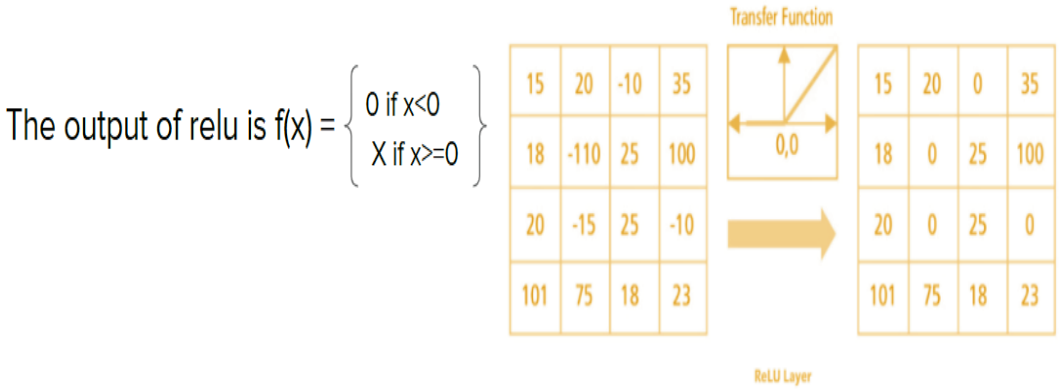


Figure 4.31: ReLU

4.6.4 Backpropagation

According to the paper from 1989, backpropagation[24]: repeatedly adjusts the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector. The error function in classic backpropagation is the mean squared error:

$$E(X,\theta) = \frac{1}{2N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

where y_i is the target value for input-output pair $(\overrightarrow{x_i}, y_i)$ and $\hat{y}_i$ is the computed output of the network on input $\overrightarrow{x_i}$.Again, other error functions can be used, but the mean squared error's historical association with backpropagation and its convenient mathematical properties make it a good choice for learning the method.

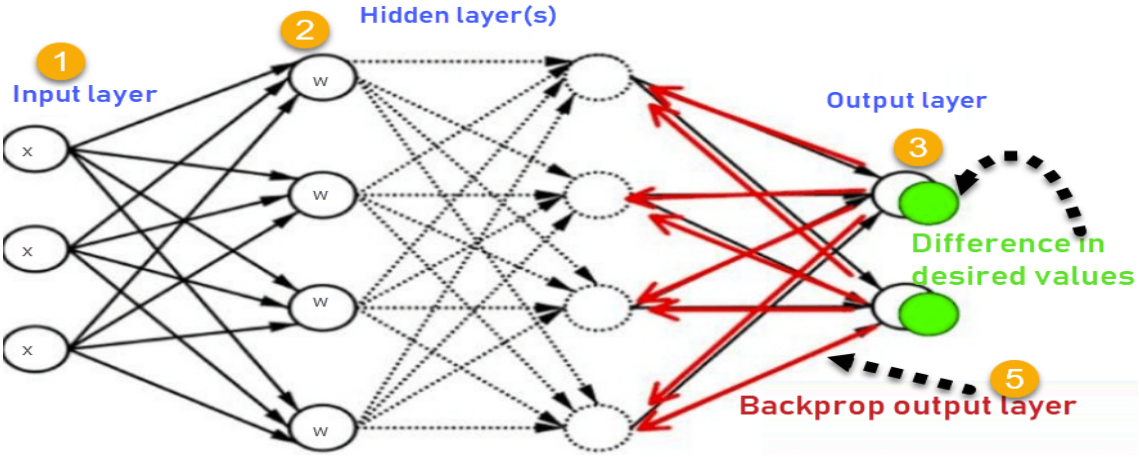


Figure 4.32: Backpropagation

4.6.5 Region of interest Align

Another major contribution of Mask R-CNN is the refinement of the ROI pooling[22]. In ROI, the warping is digitalized (top left diagram below): the cell boundaries of the target feature map are forced to realign with the boundary of the input feature maps. Therefore, each target cells may not be in the same size (bottom left diagram). Mask R-CNN uses ROI Align which does not digitalize the boundary of the cells (top right) and make every target cell to have the same size (bottom right). It also applies interpolation to calculate the feature map values within the cell better. For example, by applying interpolation, the maximum feature value on the top left is changed from 0.8 to 0.88 now.

ROI Align makes significant improvements in the accuracy.

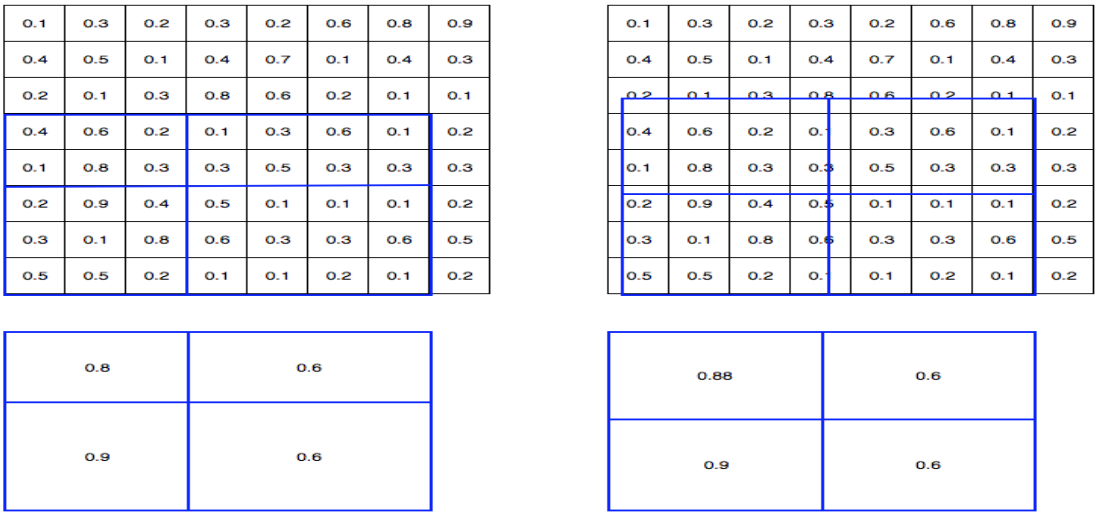


Figure 4.33: ROI Align

4.7 Data Flow Diagram

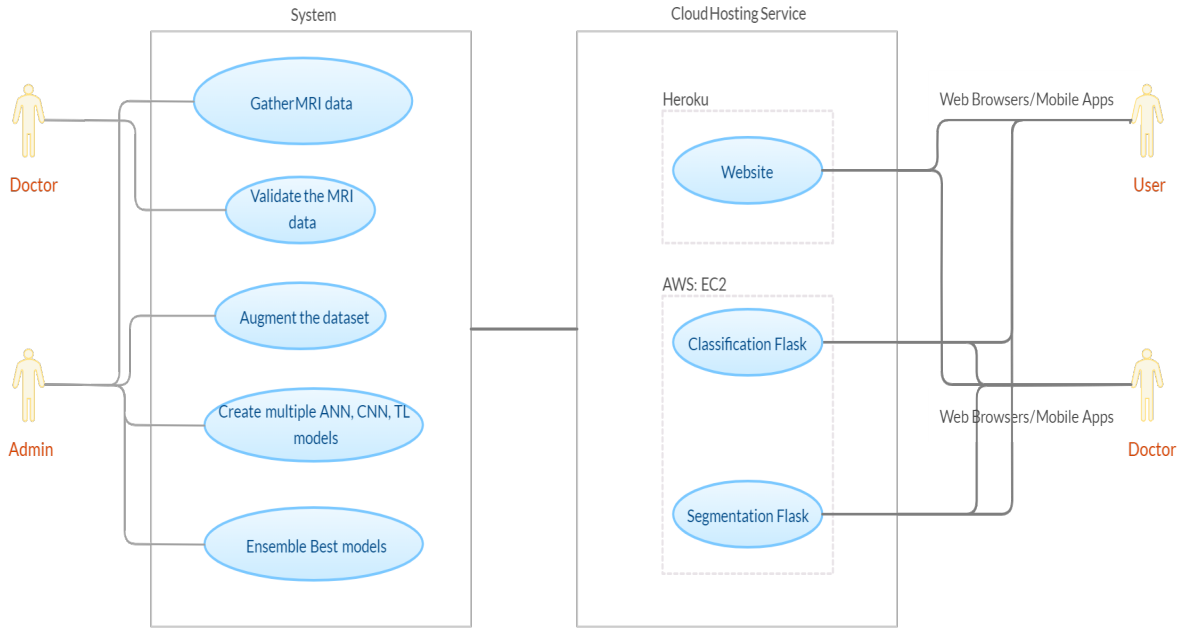


Figure 4.34: Data Flow Diagram

DFD Level: Zero (0) Level.

- **Administrator :**
  The administrator is responsible to collect valid image data from online sources as well as request data from hospitals. The group augments the data and creates various ML models and selects the best model.

- **Doctor :**
  The doctor in the system design phase is responsible to validate the MRI data gathered by the group and provides data validation certificate to the group.

- **User :**
  The user is a patient who uses Web or mobile applications to connect to the model which is hosted on cloud.

- **Doctor(s) :**
  The doctor connecting to cloud services represent a handler of patients.

Accordingly the doctor can act as an user or can access records of valid patients.

4.7.1 Data Flow Stages

1. **Pre-Processing Stage :** Pre processing of Image data helps clean the data to be fed as input into NN models. Models with clean and accurate data yield high F1 scores. Parts in Pre-Processing Stage are:

   (a) Normalization.

   (b) Edge Detection (Canny/Sobel).

   (c) Data Augmentation.

   (d) ZCA Whitening.

2. **Model Building Stage :** Model Building stage[5] is where we create multiple models using permutations and combinations of various hyper parameters. We also look at accuracy graphs over epochs. Parts in Model Building are:

   (a) ANN Model .

   (b) CNN Model.

   (c) TL Model.

   (d) MRCNN Model.

   (e) Hyper-parameter Tuning.

3. **Deployment Stage :** The best model is deployed on Cloud Platform[9] along with its weights.Parts in Deployment Stage are:

   (a) AWS Cloud.[10]

   (b) Paper Space.

   (c) Flask.

4. **Mobile Application Stage :** For ease of access user can download mobile applications which would be supported by the Cloud Provider.[14]

   (a) Android Platform (.apk file). [12]

   (b) iPhone Platform (.ipa file).[13]

CHAPTER 5
PROJECT PLAN

5.1 Project Estimate

5.1.1 Reconciled Estimates

1. **Cost Estimation :** The data set was freely made available. Cleaning and pre-processing of data was also done in house. The project is completely developed on AWS using AWS Sagemaker and deployed using AWS EC2 instances which cost a total of approximate 2000 Rs. The deployment of website was done on a free provider.

2. **Time Estimation :** The data gathering and cleaning took 2 months. The complete neural network model(s) building took 2 months. Deployment and documentation took 2 months.

5.1.2 Project Resources

1. **Development :** The project was developed on AWS SageMaker on instance **'ml.p2.xlarge'**. GPU used for training of models was NVIDIA V100.

   (a) Total time: 18 hrs

   (b) Total cost: 1717.38 Rs

2. **Deployment :** The project was deployed on AWS EC2 on demand Linux **'t2.2xlarge'** instance.

   (a) Total time: 10 hrs

   (b) Total cost: 281.10 Rs

5.2 Risk Management

5.2.1 Risk Identification

1. **Lack of Data :** The original data-set collected is small in size, this can lead to the machine learning model not generalizing.

2. **Type I error :** A type 1 error is also known as a false positive and occurs when a researcher incorrectly rejects a true null hypothesis. This means that your report that your findings are significant when in fact they have occurred by chance.

3. **Type II error :** A type II error is also known as a false negative and occurs when a researcher fails to reject a null hypothesis which is really false. Here a researcher concludes there is not a significant effect, when actually there really is.

4. **Internal Server Error :** When the server is unable to complete the request.This happens when the server is overloaded or there is an error in the application.

5. **Permission Denied :** This happens when the port 8080 is closed.

5.2.2 Risk Analysis

1. **Augmentation :** Data augmentation can be used to increase the size of data set. This helps provide much information to the neural networks.

2. **Learning Rate and Batch size :** Increasing the values of learning rate and batch size can help overcome under-fitting. Also we can decrease thee amount of drop-out layers and train the model for a few more epochs.

3. **Regularization and network size :** We can introduce L1, L2 regularization methods along with decreasing the size of the network by reducing number of dense layers. Thus reducing the ability of network to over fit.

4. **Rendering Templates :** We can request for the HTML pages with the help of url tag.

5. **Security Groups :** Adding new inbound and outbound rules in the Security groups. Add the HTTP security group for the port 80.In case some other process is running on the same port,kill the previous process on the port.

5.2.3 Overview of Risk Mitigation, Monitoring, Management

The tabular representation of all associated risk and the probability of their occurrences along with solutions proves as a summary for risk management.

1. **Risk Summary :** Definition of risk.
2. **Risk Category :** Risk is categorized in Development- Occurs in development stage, Technical-Occurs due to technical difficulty, Business-Risks that occur in business management.
3. **Probability :** It defines how likely the following risks can occur.Low-Least probable of occurrence. Medium-The risk can occur anytime during the operation. High-The risk is most likely to occur and a solution must be decided.
4. **Impact :** It deals with the aftermath of risk. Negligible-The risk did no damage. Marginal-The risk did minor damage to the system. Critical-The risk did considerable damage and must be fixed. Catastrophic-The risk has broken the system, all necessary tests must be redone to ensure everything works.
5. **Reference :** Detailed description on the associated risk.

Risk Summary	**Risk Category**	**Probability**	**Impact**	**Reference**
Improper Data set	Development	Medium	Critical	5.2.1
Under Fitting	Development	High	Marginal	6.3.1
Over Fitting	Development	Medium	Marginal	1.5.2
Lack of computation power	Technical	Medium	Negligible	1.5.3
Unavailability of Servers	Business	Low	Catastrophic	5.2.1

Table 5.1: Risk Overview

5.3 Project Schedule

5.3.1 Project Task Set

Task No.	Task	Start Date	End Date
1	Project Group Registration and Domain Registration	21st June, 2019	21st June, 2019
2	Domain Research	25th June, 2019	15th August, 2019
3	Guide Allocation and Title Finalization	24th August, 2019	26th August, 2019
4	Data Analysis	29th August, 2019	5th September, 2019
5	Data Cleaning	29th August, 2019	25th September, 2019
6	CNN Model Building	20th September, 2019	15th October, 2019
7	ANN Model Building	17th October,2019	31st October, 2019
8	TL Model Building	2nd November, 2019	20th November, 2019
9	Flask Application Classification	1st December, 2019	15th December, 2019
10	Flask Application Segmentation	1st January, 2020	20th January, 2020
11	AWS Deployment Flask Segmentation	1st February, 2020	10th February, 2020
12	AWS Deployment Flask Classification	15th February, 2020	27th February, 2020
13	Website Development	15th September, 2019	10th April, 2020
14	Website Deployment Heroku	12th April, 2020	20th April, 2020
15	Website Testing Selenium	21st April, 2020	27th April, 2020
16	Research Paper Publication	1st December, 2020	Till Date

Table 5.2: Project Task Set

5.3.2 Task Network

Task Network is a graphic representation of the tasks for the project. It depicts sequence, concurrency and dependency. It points out inter-task dependencies to help the team ensure continuous progress towards project completion. Below is the task network diagram for the project. The Tasks are allocated with respect to the Project Task Set Table mentioned above.

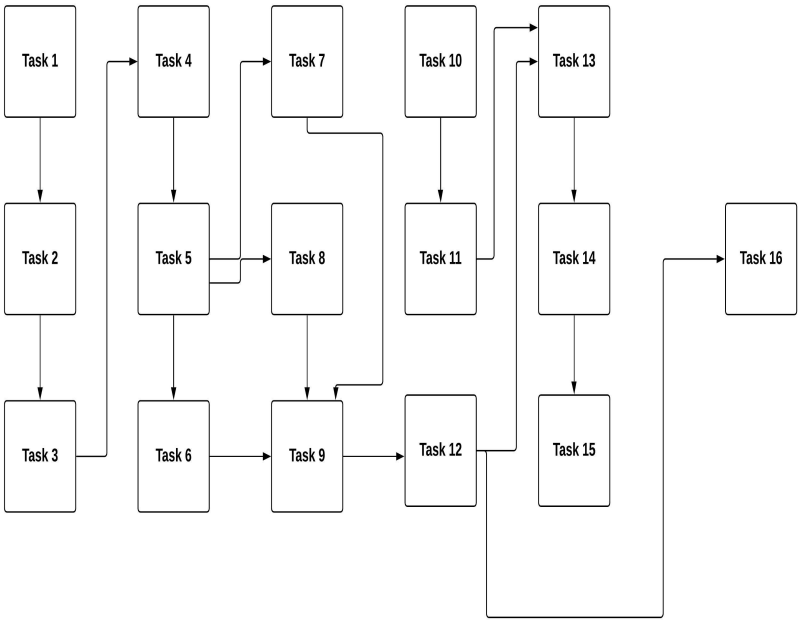


Figure 5.1: Task Network

5.3.3 Timeline Chart

A timeline chart is an effective way to visualize a process using chronological order. It is used for managing a project's schedule, timeline charts function as a sort of calendar of events within a specific period of time. Below is the timeline chart for the project.

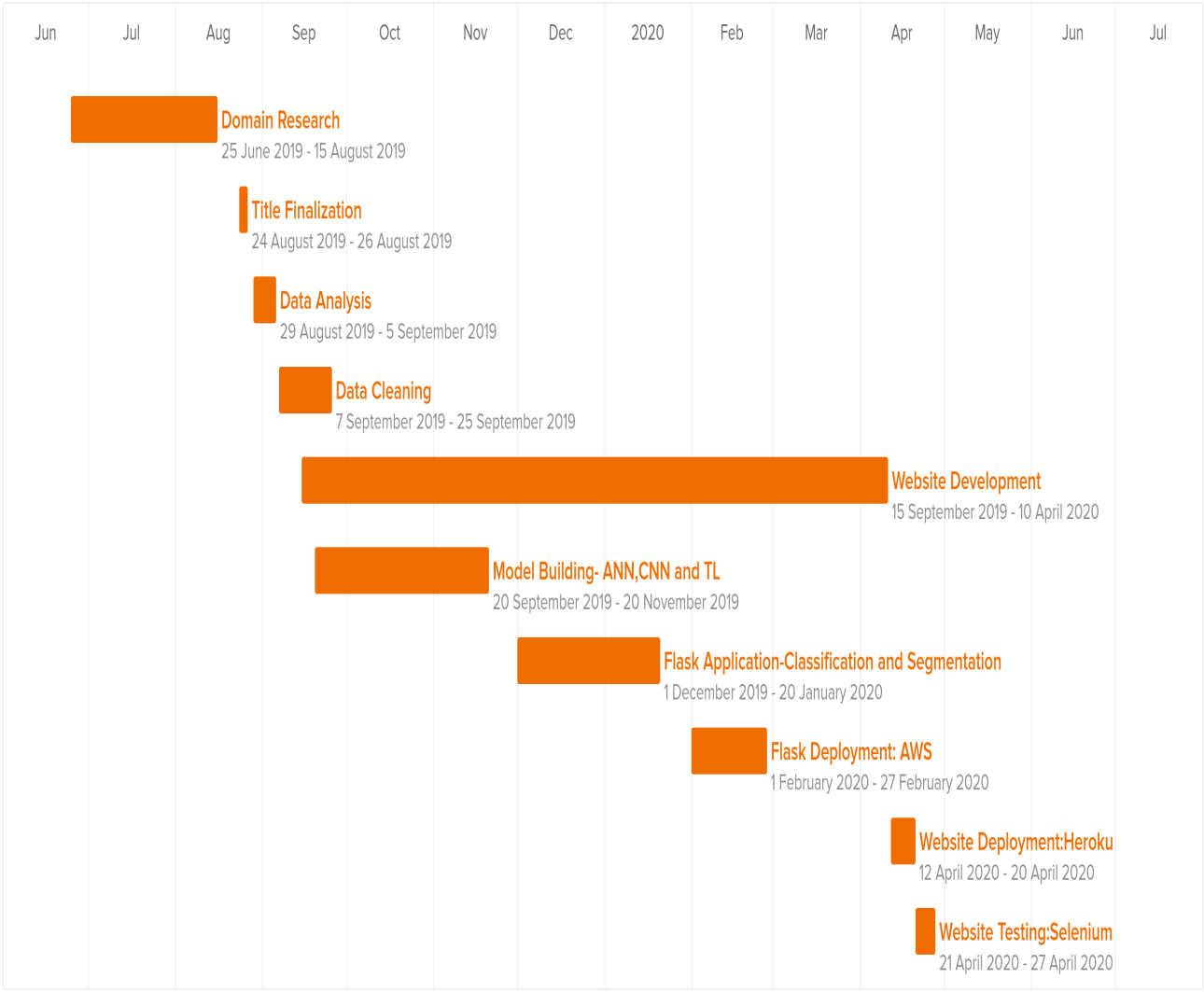


Figure 5.2: Timeline Chart

5.4 Team Organization

5.4.1 Team Structure

1. **Sartaj Bhuvaji :** (Leader, ML Models)

   Performed the pre-processing of data, using cropping and augmentation. Created the Deep learning models using ANN,CNN, TL. Tuned the hyper-parameters to get best accuracy possible.

2. **Ankita Kadam :** (Data Analysis, Flask)
   Performed cleaning and pre-processing on Data. Built the flask applications for Classification and Segmentation from scratch. Deployed the flask application on AWS.

3. **Prajakta Bhumkar :** (Web Development, Website)
   Executed Data refinement along with selective extraction of best data samples. Built the website and its associated components. Management of data-flow.

4. **Sameer Dedge :** (Android App development, Testing)
   Performed data augmentation, testing of the complete system and generation of reports. Created the android application and its associated components.

5.4.2 Management reporting and communication

We have to report to the following staff:

1. **Internal Guide :**
   Guidance on deciding project path, documentation, testing and deployment.

2. **Project Coordinator :**
   Guidance on documentation and coding style. Any progress or doubts are cleared by internal guide. Any changes or suggestions are made adhered to.

CHAPTER 6
PROJECT IMPLEMENTATION

6.1 Overview of Project Modules

6.1.1 Machine Learning models

Neural Network was used for pre-processing of images and both classification and segmentation. Multiple ANN, CNN, TL models were built to find best possible model. The ML models developed were examined on basis of validation accuracy and F1-Scores.

6.1.2 MRI Classification

Neural network consisting of four neurons in the output layer was trained to classify a MRI into one of the four classes. The neuron with highest value in output layer was selected and adjoining class label was displayed. Various hyper-parameters were fine tuned to achieve best results.

6.1.3 MRI Segmentation

COCO data-set is considered one of the best for image segmentation. So for tumor segmentation in our MRI we applied learning on the available dataset. The segmentation includes color coding the part of tumor in the image and labelling it with a probability.

6.1.4 Flask Development

Flask is a python frame work, which acts as an inter median between HTML and Python code. Flask application deployed on AWS are the machine learning models which accept image input and output a prediction.

6.1.5 Website Deployment

Web pages were built using simple HTML, CSS. As project is deployed on cloud, user must enter the system through the website. Website also provides information about the tumor, its treatment along with city wise information about available doctors and their contact information.

6.1.6 Android Application

A light weight android application was coded for ease of access. The application works on smartphones and directly links to cloud servers. Users can directly access out system by launching our mobile app.

6.2 Tools and Technologies Used

6.2.1 Development

1. **Amazon Web Services :** AWS was extensively used for development of the project. All pre-processing of data, model creation and storage was done in SageMaker. Also AWS EC2 was used for deployment of the models. EC2's Linux instance is a Infrastructure as a Service (IaaS) platform which helps in complete control over the system.

2. **Heroku :** Heroku is a platform as a service (PaaS). It was used to deploy our website. Heroku provides easy deployment of web-pages on a free domain.

6.2.2 Management

1. **Trello :** Trello is a web-based list-making application which is used for management. Tasks are divided and assigned to team members. It helps in visual representation of work process.

2. **Medium.com :** Medium is an online publishing platform. We published articles related to our project. Complete explanation of every step in the project is documented in the article.

6.2.3 Communication :

1. **Discord :** Discord is a VoIP application which also allows to share files and other data for a group.

6.2.4 Storage

1. **Google Drive :** Google Drive was used for storage of data, jupyter notebooks, models, other files of the project.

2. **GitHub :** GitHub helped provide version control to the project. All files were committed to GitHub for storage and redundancy.

6.3 Algorithm Details

6.3.1 Contour cropping and Augmentation

Our data is unclean and has few incorrectly placed images, which were corrected, deleted or replaced. Also, a few images from other medical publications were added. Getting an approximately equal number of images in each class is necessary to avoid any bias in the neural networks.

The MRIs contain a black background around the central image of the brain. This black background provides no useful information about the tumor and would be waste if fed to neural networks. Hence cropping the images around the main contour would be useful.

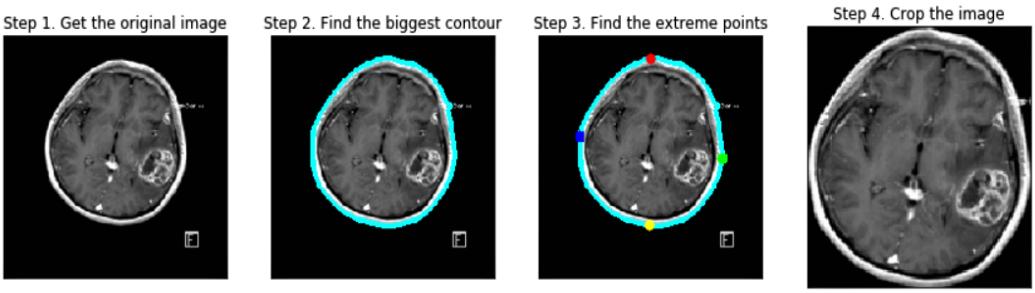


Figure 6.1: Contour Cropping

Here the biggest contour is selected and marked. Next, we find the extreme points of the contour and crop the image on those endpoints. Thus we have removed most of the unwanted background and some noise present in the original image. This process is done for each image in the dataset. However, note that sometimes the function may not be able to correctly recognize the correct contours and makes a mistake and wrongly crops the image. Such images should be removed by manual inspection before entering the **'augmentation'** phase.

The amount of data gathered was very low and could cause the models to under-fit. Hence, we would use a brilliant technique of Data Augmentation to increase the amount of data. This technique relies on rotations, flips, change in exposure, etc to create similar images.

The output image of the cropping stage is given as input to **'ImageDataGenerator'** which is a function in **'keras.preprocessing.image'** library. This function takes multiple arguments that decide how Augmentation takes place.

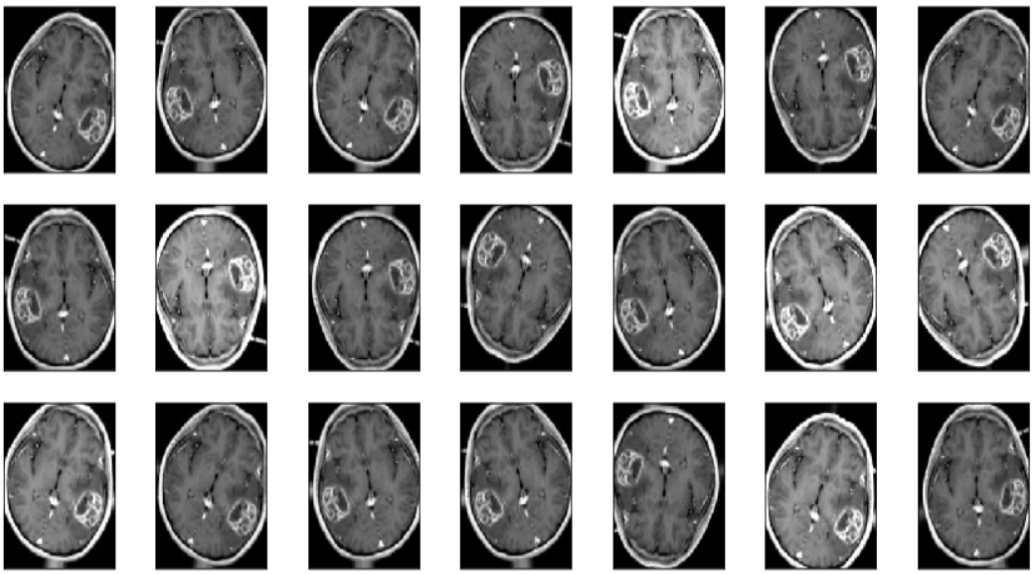


Figure 6.2: Augmentation

The output of **'ImageDataGenerator'** with defined necessary arguments is multiple images which are then saved to a separate folder. The process of augmentation is applied to every image of the data-set hence increasing the size of the data-set. Thus with such a huge amount of data, the chances of under-fitting can be lowered.

No(Set)	**Class**	**Initial Count**	**Augmented Count**	**Discarded Count**	**Final Count**
1(Training)	Glioma	826	17346	7100	10246
2(Training)	Meningioma	822	17262	7000	10262
3(Training)	No Tumor	395	8295	16	8279
4(Training)	Pituitary	827	17367	7100	10267
5(Testing)	Glioma	100	2100	10	2090
6(Testing)	Meningioma	115	2415	65	2350
7(Testing)	No Tumor	105	2205	100	2105
8(Testing)	Pituitary	75	1575	5	1570

Table 6.1: Metadata

6.3.2 Artificial Neural Network

Artificial Neural Networks are multi-layer fully-connected neural nets. They consist of an input layer, multiple hidden layers, and an output layer. Every node in one layer is connected to every other node in the next layer. We make the network deeper by increasing the number of hidden layers.

The training procedure works as follows:
Randomly initialize the weights for all the nodes. There are smart initialization methods which we will explore in another article. For every training example, perform a forward pass using the current weights and calculate the output of each node going from left to right. The final output is the value of the last node. Compare the final output with the actual target in the training data, and measure the error using a loss function. Perform a backwards pass from right to left and propagate the error to every individual node using backpropagation. Calculate each weight's contribution to the error, and adjust the weights accordingly using gradient descent. Propagate the error gradients back starting from the last layer.

Artificial neural networks use backpropagation as a learning algorithm to compute a gradient descent with respect to weights. Desired outputs are compared to achieved system outputs, and then the systems are tuned by adjusting connection weights to narrow the difference between the two as much as possible. Because backpropagation requires a known, desired output for each input value in order to calculate the loss function gradient.

Models built using ANN Algorithm:

ANN Models	Test Accuracy	Test Loss	F1-Score
32-nodes-0-dense	76	213.54	76
64-nodes-0-dense	77	229.56	77
128-nodes-0-dense	80	181.14	80
32-nodes-1-dense	24	1.38	10
64-nodes-1-dense	25	1.38	13
128-nodes-1-dense	25	1.39	10
32-nodes-2-dense	26	1.33	10
64-nodes-2-dense	24	1.3	10
128-nodes-2-dense	25	1.38	10

Table 6.2: Artificial Nerural Network

6.3.3 Convolution Neural Network

Convolutional Neural Networks is one of the variants of neural networks used heavily in the field of Computer Vision. It derives its name from the type of hidden layers it consists of. The hidden layers of a CNN typically consist of convolutional layers, pooling layers, fully connected layers, and normalization layers. Here it simply means that instead of using the normal activation functions, convolution and pooling functions are used as activation functions.

CNN learns the filters automatically without mentioning it explicitly. These filters help in extracting the right and relevant features from the input data. CNN captures the spatial features from an image. Spatial features refer to the arrangement of pixels and the relationship between them in an image. They help us in identifying the object accurately, the location of an object, as well as its relation with other objects in an image. CNN also follows the concept of parameter sharing. A single filter is applied across different parts of an input to produce a feature map.

The training of CNNs becomes the task of learning filters (deciding what features you should look for in the data.) Pooling and ReLU: CNNs have two non-linearities: pooling layers and ReLU functions. Pooling layers consider a block of input data and simply pass on the maximum value. Doing this reduces the size of the output and requires no added parameters to learn, so pooling layers are often used to regulate the size of the network and keep the system below a computational limit.

Models built using CNN Algorithm:

CNN Models	**Test Accuracy**	**Test Loss**	**F1-Score**
1-conv-32-nodes-0-dense	86	1.77	86
2-conv-32-nodes-0-dense	85	0.76	85
3-conv-32-nodes-0-dense	90	0.48	90
1-conv-64-nodes-0-dense	85	2.39	85
2-conv-64-nodes-0-dense	89	0.55	89
3-conv-64-nodes-0-dense	89	0.55	89
1-conv-128-nodes-0-dense	84	1.36	84
2-conv-128-nodes-0-dense	82	1.01	81
3-conv-128-nodes-0-dense	86	0.95	86
1-conv-32-nodes-1-dense	24	1.38	10
2-conv-32-nodes-1-dense	88	0.59	89
3-conv-32-nodes-1-dense	86	0.58	86
1-conv-64-nodes-1-dense	26	1.36	13
2-conv-64-nodes-1-dense	88	0.48	89
3-conv-64-nodes-1-dense	88	0.54	89
1-conv-128-nodes-1-dense	85	0.73	85
2-conv-128-nodes-1-dense	85	0.77	83
3-conv-128-nodes-1-dense	90	0.43	91
1-conv-32-nodes-2-dense	24	1.38	10
2-conv-32-nodes-2-dense	24	1.38	10
3-conv-32-nodes-2-dense	63	0.84	57
1-conv-64-nodes-2-dense	24	1.38	10
2-conv-64-nodes-2-dense	83	0.70	83
3-conv-64-nodes-2-dense	85	0.55	86
1-conv-128-nodes-2-dense	24	0.00	10
2-conv-128-nodes-2-dense	85	0.78	87
3-conv-128-nodes-2-dense	85	0.64	85

Table 6.3: Convolutional Nerural Network

6.3.4 Transfer Learning

Transfer Learning is a machine learning technique where a model trained on one task is re-purposed on a second related task. A pre-trained source model is chosen from available models. Many research institutions release models on large and challenging datasets that may be included in the pool of candidate models from which to choose from. The model pre-trained model can then be used as the starting point for a model on the second task of interest. This may involve using all or parts of the model, depending on the modeling technique used. Optionally, the model may need to be adapted or refined on the input-output pair data available for the task of interest.

One of the fundamental requirements for transfer learning is the presence of models that perform well on source tasks. Pre-trained models are available for everyone to use through different means. The famous deep learning Python library, keras, provides an interface to download some popular models.

This approach is effective because the images were trained on a large corpus of photographs and require the model to make predictions on a relatively large number of classes, in turn, requiring that the model efficiently learn to extract features from photographs in order to perform well on the problem. For computer vision, you can leverage some popular models including, VGG-16[25], VGG-19, Inception V3, XCeption, ResNet-50. Deep learning has been quite successfully utilized for various computer vision tasks, such as object recognition and identification, using different CNN architectures.

Models built using TL Algorithm:

TL Models	**Test Accuracy**	**Test Loss**	**F1-Score**
VGG16	94	0.81	94
InceptionV3	27	1.40	19
VGG19	47	0.46	33
ResNet50	88	0.41	88
MobileNet v2	37	1.73	30

Table 6.4: Transfer Learning Model

6.3.5 Mask Region-based Convolution Neural Network

Mask RCNN is a deep neural network aimed to solve instance segmentation problem in machine learning or computer vision.

There are two stages of Mask RCNN.[19] First, it generates proposals about the regions where there might be an object based on the input image. Second, it predicts the class of the object, refines the bounding box and generates a mask in pixel level of the object based on the first stage proposal. Both stages are connected to the backbone structure.

Backbone is a FPN(Feature Pyramid Networks) style deep neural network. It consists of a bottom-up pathway , a top-bottom pathway and lateral connections. Bottom-up pathway can be any ConvNet, usually ResNet or VGG, which extracts features from raw images. Top-bottom pathway generates feature pyramid map which is similar in size to bottom-up pathway. Lateral connections are convolution and adding operations between two corresponding levels of the two pathways. FPN outperforms other single ConvNets mainly for the reason that it maintains strong semantically features at various resolution scales.

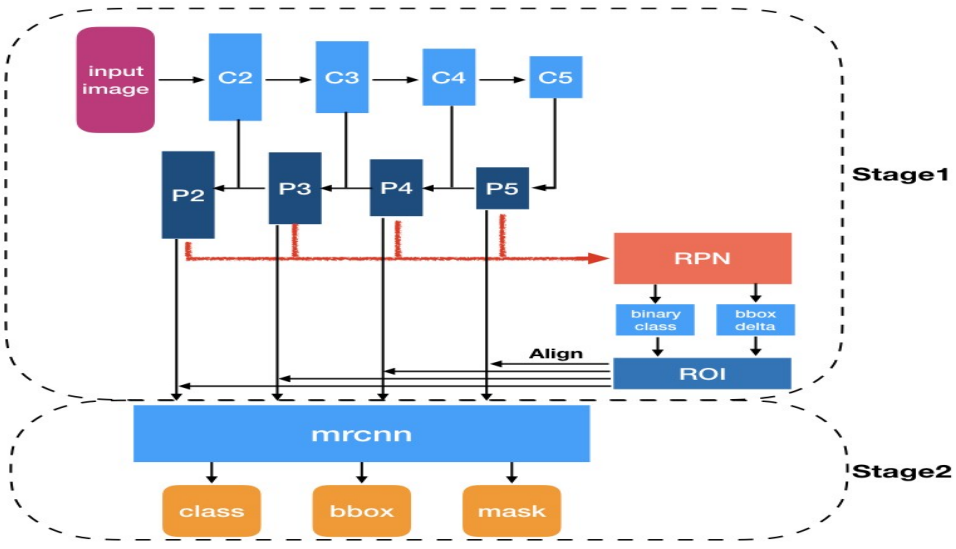


Figure 6.3: MRCNN Stages

Consider the first stage. A light weight neural network called RPN (Region Proposal Network) scans all FPN top-bottom pathway and proposes regions which may contain objects. While scanning feature map is an efficient way, we need a method to bind features to its raw image location. Here come the anchors. Anchors are a set of boxes with predefined locations and scales relative to images. Ground-truth classes(only object or background binary classified at this stage) and bounding boxes are assigned to individual anchors according to some IoU value. As anchors with different scales bind to different levels of feature map, RPN uses these anchors to figure out where of the feature map 'should' get an object and what size of its bounding box is.

At the second stage, another neural network takes proposed regions by the first stage and assign them to several specific areas of a feature map level, scans these areas, and generates objects classes(multi-categorical classified), bounding boxes and masks. The procedure looks similar to RPN. Differences are that without the help of anchors, stage-two used a trick called ROIAlign to locate the relevant areas of feature map, and there is a branch generating masks for each objects in pixel level.

CHAPTER 7
SOFTWARE TESTING

7.1 Types of Testing

7.1.1 Automation Testing

Automation testing is the management and performance of test activities, to include the development and execution of test scripts so as to verify test requirements, using an automation testing tool. It helps in the comparison of actual outcomes with predicted outcomes. Automated software testing can increase the depth and scope of tests to help improve software quality. Lengthy tests can be run on multiple computers with different configurations. Automated software testing can examine an application and investigate memory contents, data tables, file contents, and internal program states to determine if the product is behaving as expected. Automated software tests can easily execute thousands of different complex test cases during a test run. Automation testing is used to simulate a typical user environment using categorically deployed mouse clicks and keystrokes.

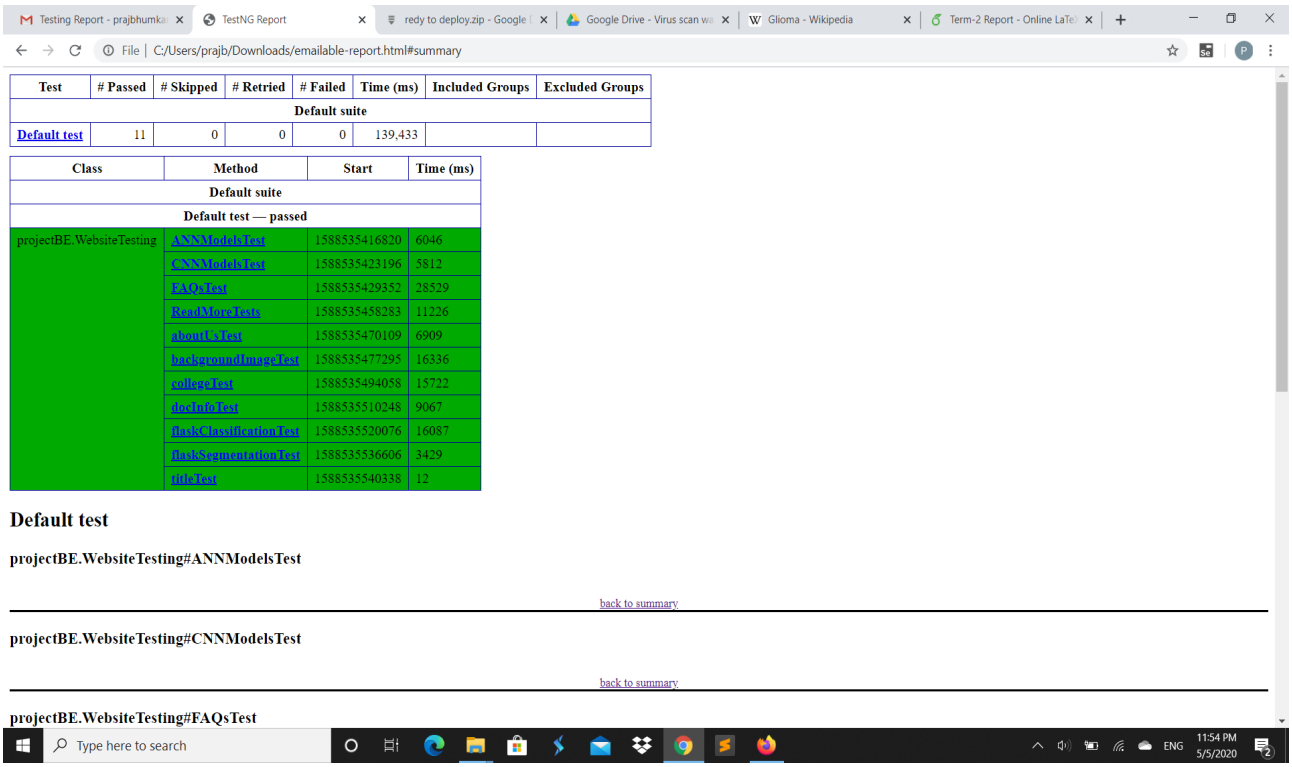
In this project we have used **Selenium Automation Testing Tool** to perform Automation Testing.

7.1.2 Manual Testing

Manual testing involving the team to stress test the system by accessing the system all at once. Various MRI were uploaded to the system and its output was cross validated. Various file-types of images were tested. Also navigation around the website was tested. The segmentation model was tested with batches of MRI to validate consistency of segmentation of tumor.

In this project the type of Manual Testing performed is **White Box Testing.**

7.2 Test Cases and Test Results



Test	# Passed	# Skipped	# Retried	# Failed	Time (ms)	Included Groups	Excluded Groups
Default suite							
Default test	11	0	0	0	139,433		

Class	Method	Start	Time (ms)
Default suite			
Default test — passed			
projectBE.WebsiteTesting	ANNModelsTest	1588535416820	6046
	CNNModelsTest	1588535423196	5812
	FAQsTest	1588535429352	28529
	ReadMoreTests	1588535458283	11226
	aboutUsTest	1588535470109	6909
	backgroundImageTest	1588535477295	16336
	collegeTest	1588535494058	15722
	dogInfoTest	1588535510248	9067
	flaskClassificationTest	1588535520076	16087
	flaskSegmentationTest	1588535536606	3429
	titleTest	1588535540338	12

Default test

projectBE.WebsiteTesting#ANNModelsTest

back to summary

projectBE.WebsiteTesting#CNNModelsTest

back to summary

projectBE.WebsiteTesting#FAQsTest

Figure 7.1: Test Result (Fig.1)

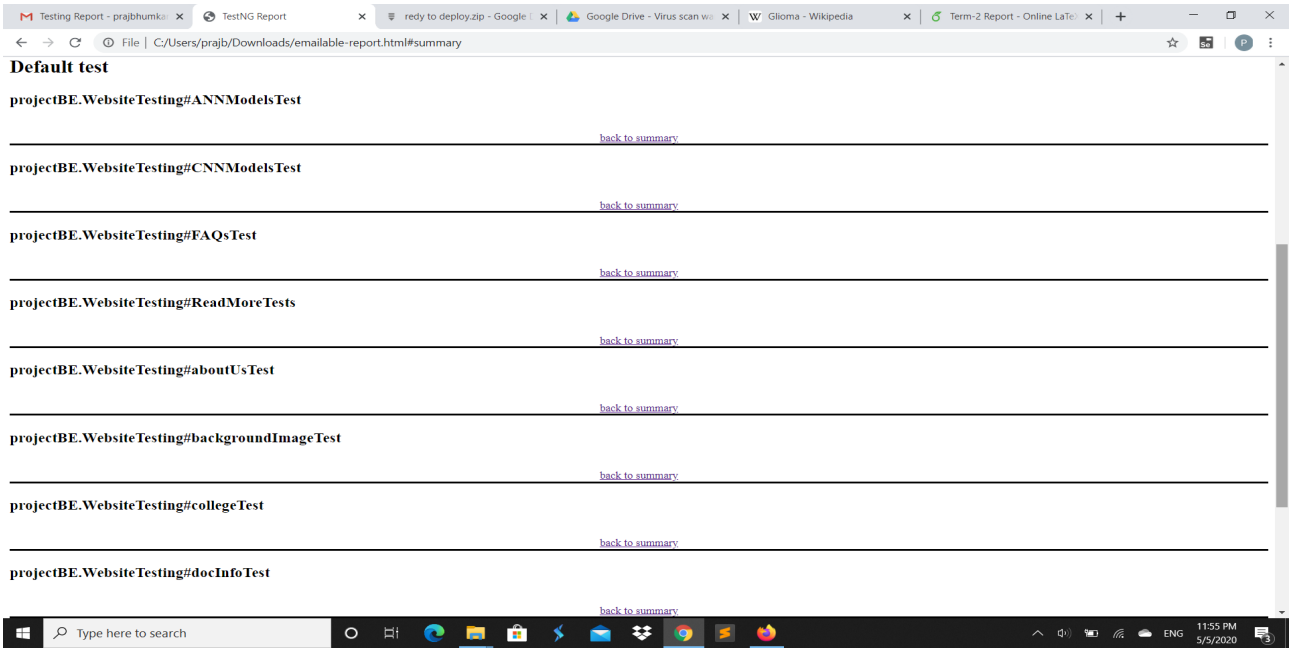


Figure 7.2: Test Result (Fig.2)

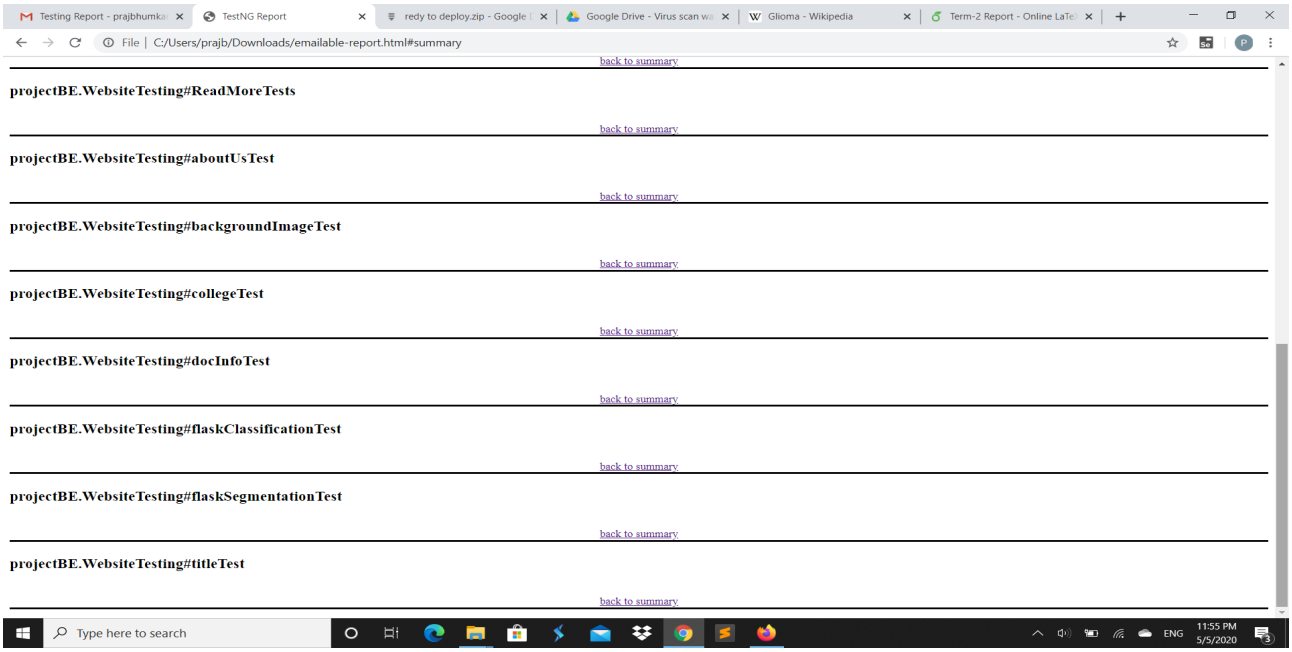


Figure 7.3: Test Result (Fig.3)

No.	Test Case Description	Expected Result	Actual Result	Executed Date	Test Case Priority	Test Results
1	Upload the MRI image.	The image is uploaded in (.jpg,.jpeg, .png) format	The image is uploaded in the given format	25-4-2020	High	Pass
2	The output is returned with the image.	The image is returned with the tumor class.	The image is returned with the tumor class.	25-4-2020	High	Pass
3	Navigation to the information page	The page with hyperlinks of the type tumor is shown.	The page with hyperlinks of the type tumor is shown.	25-4-2020	Medium	Pass
4	Navigation to the information page of Glioma Tumor.	The page with the detailed information of Glioma Tumor is shown.	The page with the detailed information of Glioma Tumor is shown.	25-4-2020	Medium	Pass
5	Navigation to the information page of Meningioma Tumor.	The page with the detailed information of Meningioma Tumor is shown.	The page with the detailed information of Pituitary Tumor is shown.	25-4-2020	Medium	Pass
6	Navigation to the information page of Pituitary Tumor.	The page with the detailed information of Pituitary Tumor is shown.	The page with the detailed information of meningioma Tumor is shown.	25-4-2020	Medium	Pass

Table 7.1: Manual Testing Report(1)

No.	Test Case Description	Expected Result	Actual Result	Executed Date	Test Case Priority	Test Results
7	Navigation back to home page.	The page is redirected from the Glioma Informative page to home.	The page is redirected from the Glioma Informative page to home.	25-4-2020	Low	Pass
8	Navigation back to home page.	The page is redirected from the Meningioma Informative page to home.	The page is redirected from the Meningioma Informative page to home.	25-4-2020	Low	Pass
9	Navigation back to home page.	The page is redirected from the Pituitary Informative page to home.	The page is redirected from the Pituitary Informative page to home.	25-4-2020	Low	Pass

Table 7.2: Manual Testing Report(2)

CHAPTER 8

RESULTS

8.1 Outcomes

1. Thus an **ensemble system** consisting of **3 models** was deployed achieving an overall accuracy of **94 percent**.
2. The system is able to classify MRI into tumor types.
3. The system is able to segregate tumor from MRI.
4. Mobile application are working correctly.
5. All information about treatment is provided.
6. Website link(URL): https://website-demo-one.herokuapp.com/

8.2 Screen Shots

8.2.1 Website

This screenshot represents the front page of our website hosted on **Heroku servers**. The complete project is tied together with this website. Access of both flask systems is provided here. Also the website hosts information about different types of tumors along with its systems and treatments. Information about doctors is also provided.

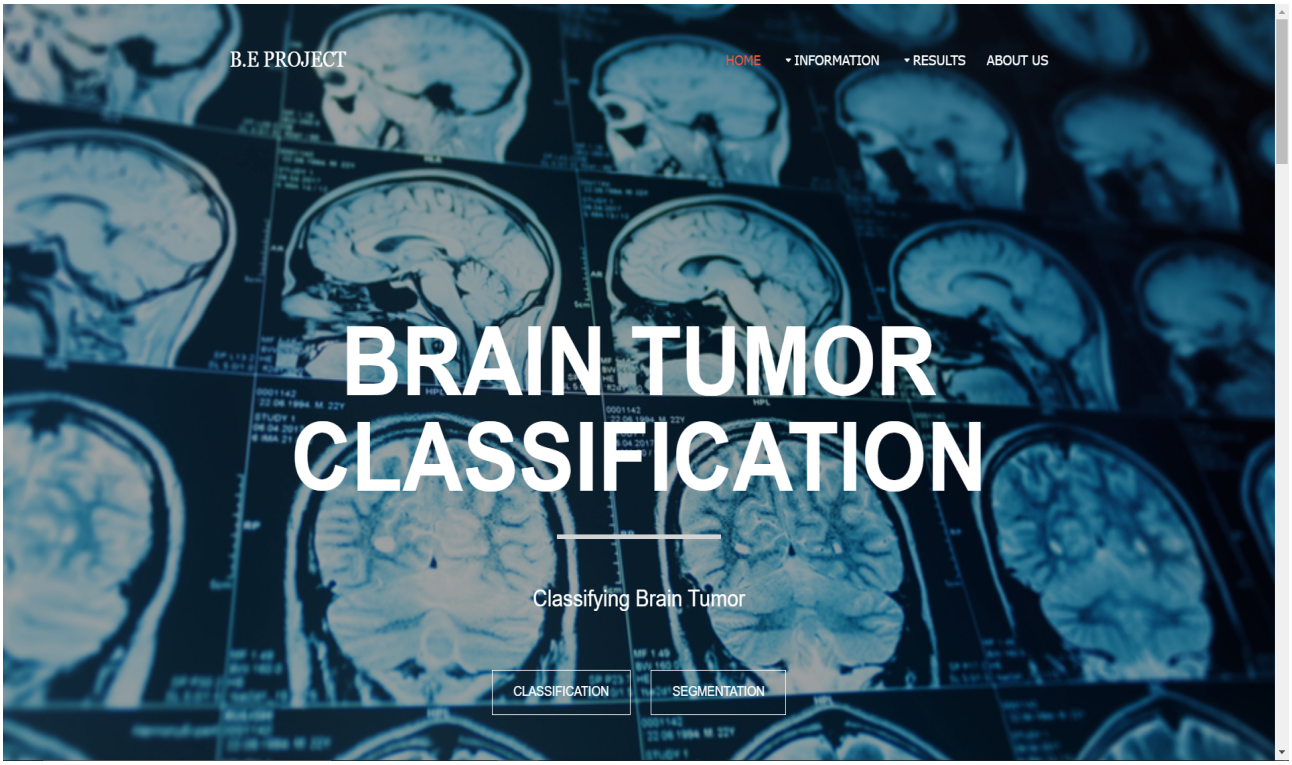


Figure 8.1: Website

8.2.2 AWS Instance

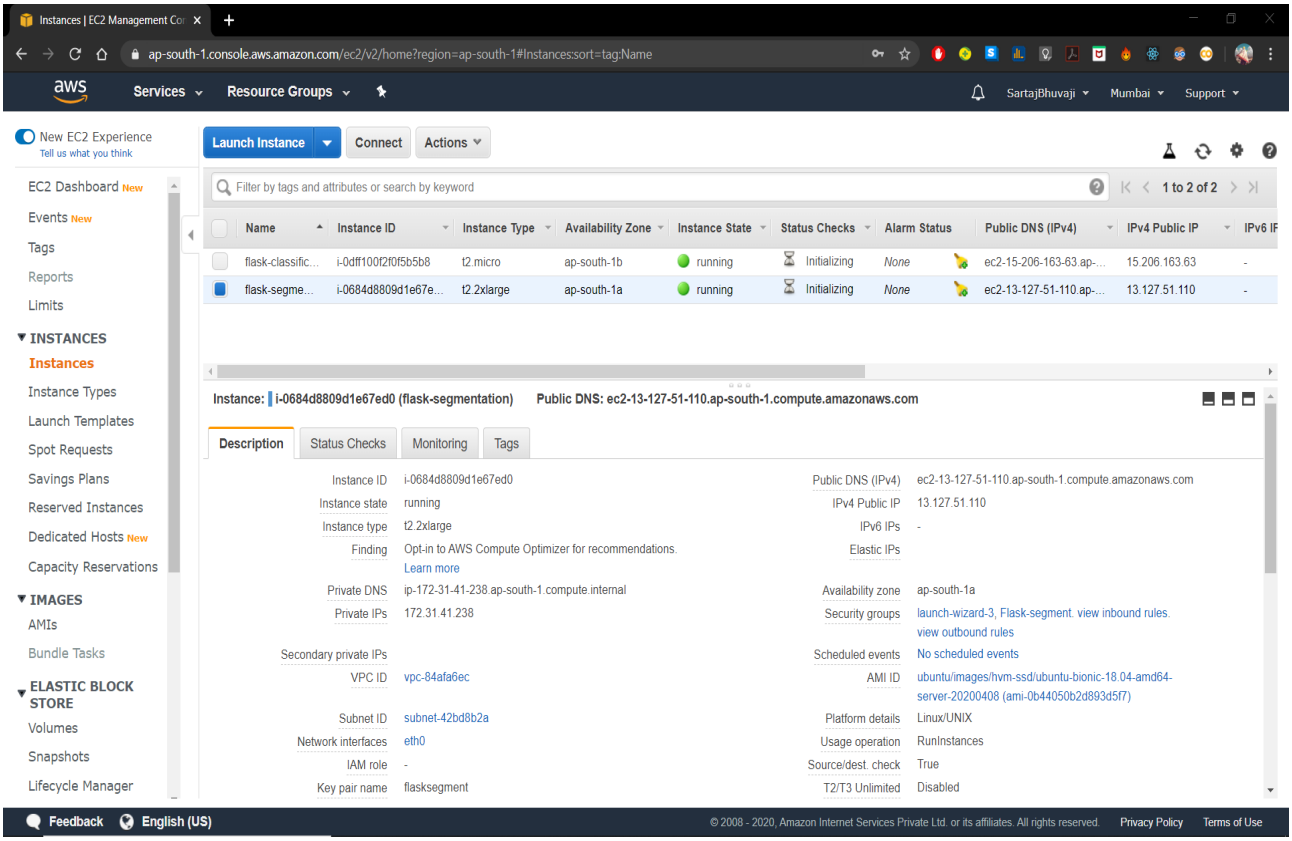


Figure 8.2: AWS Instances

The AWS EC2 instances host our flask applications for **Brain Tumor Classification** and **Brain Tumor Segmentation**. They are hosted on two separate Virtual Machines and can be accessed using the website or our mobile application.

8.2.3 Flask

The below stated MRI is the input for both Classification and Segmentation flask applications hosted on AWS servers.

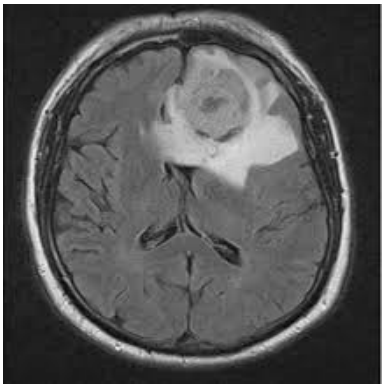


Figure 8.3: MRI Input

Below is the screenshot showing the result of classification algorithm. The ensemble models are called on the uploaded image and the output label is generated.

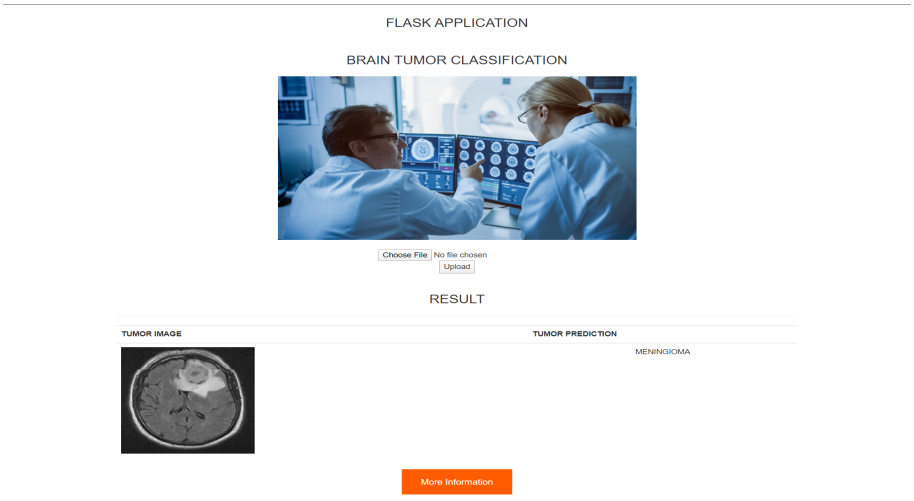


Figure 8.4: Classification Page

In this case the output label is Meningioma. Which is correct! Thus our models have correctly classified the tumor present in the MRI provided!

The following is the screenshot of the Segmentation part of MRI. The same MRI used in classification is uploaded for segmentation of tumor(s).

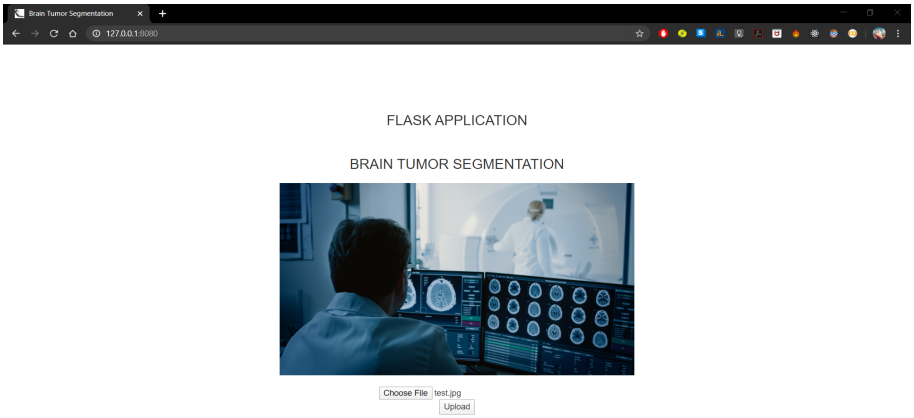


Figure 8.5: Segmentation Page

Below is the output file generated from Segmentation model. The image generated has colored, parts of MRI that model predicts to be tumor(s) or part(s) of tumor.

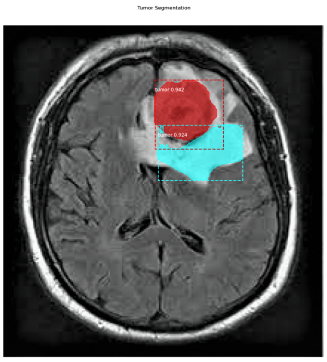


Figure 8.6: MRI Output

We can see, the image generated from segmentation model is quite close to the actual size and actual location of existing tumor(s). Thus model can correctly segment tumors in an MRI!

8.2.4 Mobile Application

Here we represent the website as viewed from our mobile application and segmentation module.

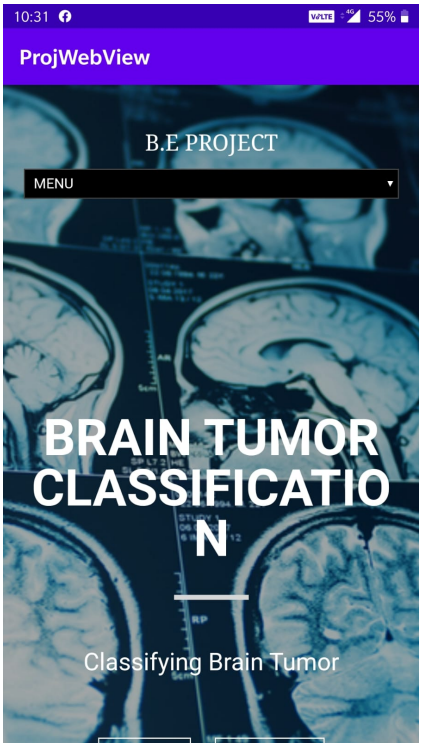


Figure 8.7: Mobile Screenshot(1)

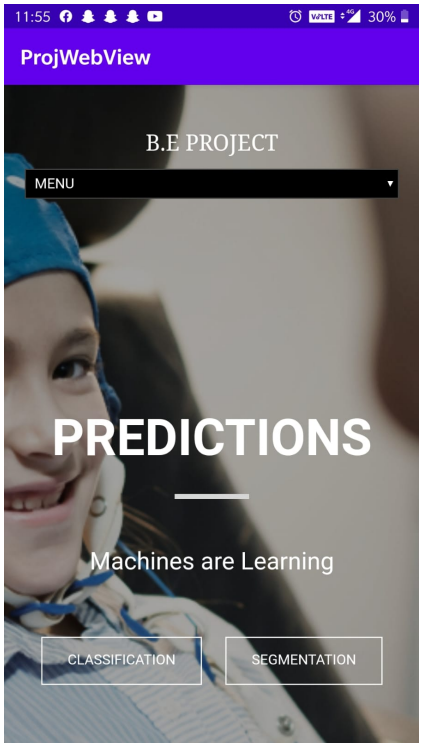


Figure 8.8: Mobile Screenshot(2)

CHAPTER 9
CONCLUSIONS

9.1 Conclusions

1. We have created multiple deep earning models using Artificial Neural Network, Convolution Neural Network, and Transfer Learning.
2. We select the best of ML model(s) with **94** percent accuracy and lowest error rate for deployment on Cloud Platform.
3. The system can be accessed by users and doctors, through Web-Browsers or Android application.
4. The system is able to correctly **classify** Brain Tumors and also examine about changes in **tumor position.**
5. The best possible Neural Network architecture for the automation of classification of brain tumor MR images into 4 classes : **No Tumor, Benign Tumor, Malignant Tumor and Pituitary Tumor** is studied.

9.2 Future Work

1. Increase number of tumor types that can be detected.
2. Provide feature to detect other medical conditions in MRI.
3. Complete the system by linking doctors directly to patients.
4. Build database to store and retrieve records.
5. Develop mobile app to run offline.

9.3 Applications

1. Patients with MRI can immediately check their report through the mobile applications.
2. Doctors with in-confident knowledge about tumor can guide the patient through validation using the system.
3. Patients can view alternative hospitals and doctors which would provide treatment.
4. Users no longer have to wait for results of MRI analysis.

ANNEXURE A

A.1 Computation complexity

- **P Problem :**
  In our project if the input MRI provided by the user is valid and our system strikes all confidence in the first iteration then our problem is completed in polynomial time and hence it is a 'P Problem'.

- **NP Problem :**
  If our system does not receive a valid MRI or if our system is not completely confident about the tumor class in first iteration it reruns the iteration with some augmentation, in such case the problem does not run in polynomial time so it becomes a 'NP Problem'.

ANNEXURE B

B.1 Article Publication

The following articles are published in **Medium.com** which is an online publishing platform for developers and enthusiast.

B.1.1 Brain Tumor Classification

This articles provides detailed description about all the steps taken in creation of the ML models. Details about pre-processing stage along with code snippets of ANN, CNN, TL model are mentioned. Also the graphs of accuracy and loss of best models are plotted for better visual understanding.

- **Author :** Sartaj Bhuvaji
- **Article :** https://medium.com/@sartajbhuvaji/brain-tumor-classification-546a72d4103b
- **Views :** 700+

B.1.2 Deploying a Flask web application on AWS

This articles provides a detailed description about the steps for development of a flask web application.The details about deployment of the flask web application on AWS cloud platform are briefly mentioned.

- **Author :** Ankita Kadam
- **Article :** https://medium.com/@ankitakadam/deploying-a-flask-web-application-on-aws-771909373a4
- **Views :** 300+

B.1.3 Brain Tumor Segmentation in MRI

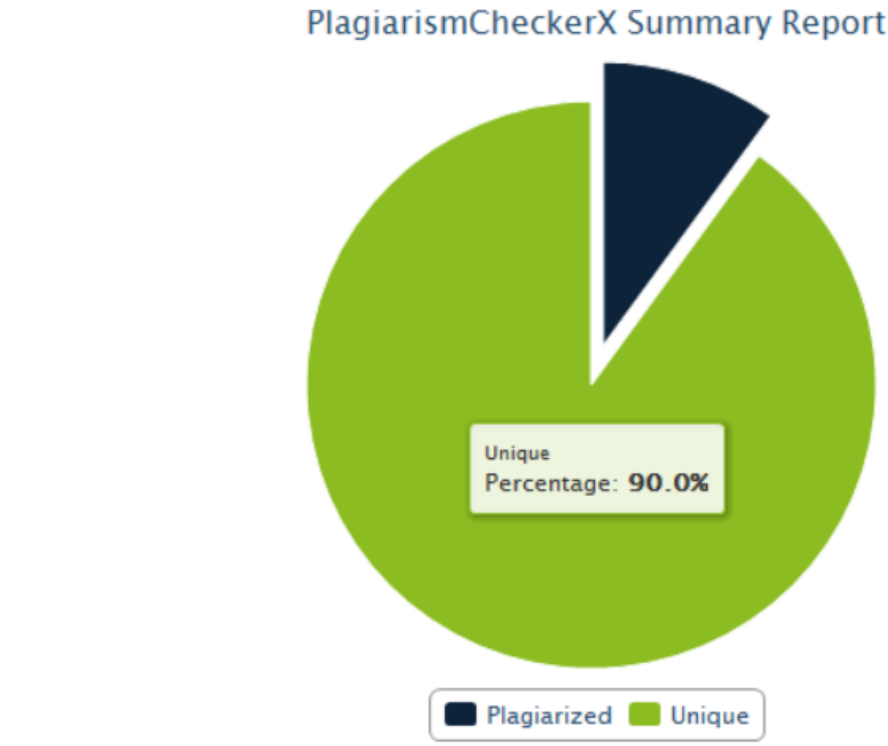
This article presents detailed description about the use of mask R-CNN to segment tumor from MRI. Description about tumor identification and segregation is provided. All necessary steps involved are described briefly.

- **Author :** Prajakta Bhumkar
- **Article :** https://medium.com/@prajbhumkar/brain-tumor-segmentation-in-mri-abc268faa304
- **Views :** 300+

B.2 Research Paper

- **Title :** Brain Tumor Classification using Deep Learning Algorithms
- **Journal :** International Journal of Artificial Intelligence and Soft Computing

ANNEXURE C



Date	Tuesday, May 26, 2020
Words	1282 Plagiarized Words / Total 13143 Words
Sources	More than 130 Sources Identified.

Figure C.1: Plagiarism Report

CHAPTER 10
REFERENCES

[1] J. Seetha and S.Selvakumar Raja," Brain Tumor Classification Using Convolutional Neural Networks" in Biomedical Pharmacology Journal, September 2018,Vol. 11(3), p. 1457-1461.
http://biomedpharmajournal.org/vol11no3/brain-tumor-classification-using-convolutional-neural-networks/

[2] Ali Ari and Davyut Hanbay, "Deep learning based brain tumor classification and detection system" in Turkish Journal of Electrical Engineering Computer Sciences,2018, 26: 2275 – 2286.
https://journals.tubitak.gov.tr/elektrik/abstract.htm?id=23203

[3] Hossam H. Sultan , Nancy M. Salem and Walid Al-atabany, "Multi-Classification of Brain Tumor Images Using Deep Neural Network" IEEE,2019.
https://ieeexplore.ieee.org/abstract/document/8723045

[4] Jiss Kuruvilla, Dhanya Sukumaran, Anjali Sankar, Siji P Joy,"A review on image processing and image segmentation",IEEE,2016.
https://ieeexplore.ieee.org/document/7684170

[5] Sentdex : "Machine Learning with python"
https://www.youtube.com/user/sentdex

[6] Karan Chauhan1, Shrwan Ram2 , "Image Classification with Deep Learning and Comparison between Different Convolutional Neural Network Structures using Tensorflow and Keras",
International Journal of Advance Engineering and Research Development. (Volume 5, Issue 02, February -2018)

[7] Pulkit Sharma, "Computer Vision Tutorial: A Step-by-Step Introduction to Image Segmentation Techniques (Part 1,2,3)"
https://www.analyticsvidhya.com/blog/2019/04/introduction-image-segmentation-techniques-python

[8] Parul Pandey, "Image Segmentation using Python's scikit-image module."
https://towardsdatascience.com/image-segmentation-using-pythons-scikit-image-module-533a61ecc980

[9] Donal Byrne, "Building Your First Neural Network On The Cloud."
https://medium.com/coinmonks/building-your-first-neural-network-on-the-cloud-ffb9fcfef945

[10] Cynthya Peranandam, "Get Started with Deep Learning Using the AWS Deep Learning AMI."
https://aws.amazon.com/blogs/machine-learning/get-started-with-deep-learning-using-the-aws-deep-learning-ami/

[11] Vaibhav Kumar, "Deploy Machine Learning Models"
https://medium.com/analytics-vidhya/how-to-deploy-simple-machine-learning-models-for-free -56cdccc62b8d

[12] Vinay Somawat, Convert a website into an Android app from scratch.
https://hackernoon.com/how-to-convert-a-website-into-an-android-app-from-scratch-de19c84a5801

[13] Karl Penzhorn, "Build an iOS App with React Native and Publish it to the App Store."
https://developer.okta.com/blog/2019/04/05/react-native-ios-app-store

[14] Danielsson, W., Froberg, A., Berglund, E. (2016)." React Native Application Development-A comparison between native Android and React Native,(pp. 1-70)"
http://www.divaportal.org/smash/get/diva2:998793/FULLTEXT02

[15] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton "ImageNet Classification with Deep Convolutional Neural Networks"
https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[16] Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun "Deep Residual Learning for Image Recognition"
https://arxiv.org/pdf/1512.03385.pdf

[17] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li and Li Fei-Fei "ImageNet: A Large-Scale Hierarchical Image Database"
http://www.image-net.org/papers/imagenet-cvpr09.pdf

[18] Kaiming He, Georgia Gkioxari, Piotr Dollar, Ross Girshick for 'Facebook AI Research' "Mask RCNN"
https://arxiv.org/pdf/1703.06870

[19] Jonathan Hui "What do we learn from region based object detectors (Faster R-CNN, R-FCN, FPN)?"
https://medium.com/@jonathan-hui/what-do-we-learn-from-region-based-object-detectors-faster-r-cnn-r-fcn-fpn-7e354377a7c9

[20] Xiang Zhang "Simple Understanding of Mask RCNN"
https://medium.com/@alittlepain833/simple-understanding-of-mask-rcnn-134b5b330e95

[21] Himanshu Rawlani "Visual Interpretability for Convolutional Neural Networks"
https://towardsdatascience.com/visual-interpretability-for-convolutional-neural-networks-2453856210ce

[22] Sambasivarao. K "Region of Interest Pooling"
https://towardsdatascience.com/region-of-interest-pooling-f7c637f409af

[23] Karl Weiss, Taghi M. Khoshgoftaar, DingDing Wang "A survey of transfer learning"
https://journalofbigdata.springeropen.com/articles/10.1186/s40537-016-0043-6

[24] Grant Sanderson "What is backpropagation really doing?"
https://www.3blue1brown.com/videos-blog/2017/11/3/what-is-backpropagation-really-doing-deep-learning-chapter-3

[25] Karen Simonyan and Andrew Zisserman "Very deep convolutional networks for large-scale image recognition"
https://arxiv.org/pdf/1409.1556.pdf

[26] Sartaj Bhuvaji, Ankita Kadam, Prajakta Bhumkar, Sameer Dedge, and Swati Kanchan, "Brain Tumor Classification (MRI)."
Kaggle, doi: 10.34740/KAGGLE/DSV/1183165.

Data Sets:

- Kaggle - https://www.kaggle.com/sartajbhuvaji/brain-tumor-classification-mri