TABLE OF CONTENTS

LIST OF FIGURES

NOMENCLATURE

CNN	:	Convolution Neural Network
DL	:	Deep Learning
FC	:	Fully Connected
MRI	:	Magnetic Resonance Imaging
ReLU	:	Rectified Linear Unit
SVM	:	Support Vector Machine

ACKNOWLEDGEMENT

It is our privilege and pleasure to express a profound sense of respect, gratitude, and indebtedness to our Vice Chancellor **Prof. S. Ramachandram**, Anurag Group of Institutions for his indefatigable inspiration, guidance, cogent discussion, constructive criticisms, and encouragement throughout this dissertation work.

We would like to thank **Prof. Syed Sameen Fatima,** Registrar and Head, Department of Artificial Intelligence for giving us the provision to carry out this project and for moral support throughout the period for study in Anurag Group of Institutions.

We extend our sincere thanks to our guide **Asst. Prof. Mr. M. Hari Prasad** for the motivation and co-operation for the successful completion of the project.

Finally, we wish to acknowledge our management and friends for helping us to complete the dissertation.

ABSTRACT

One of the most leading death causes in the world is brain tumor. Tumor Detection is one of the most difficult tasks in medical image processing. in fact, the manual classification with human assisted support can be improper prediction and diagnosis shown by medical evidence. The detection task is too difficult to perform because there is a lot of diversity in the images as brain tumors come in different shapes and textures.

Recently, deep learning techniques showed promising results towards improving accuracy of detection and classification r brain tumor from magnetic resonance imaging (MRI), In this project, we propose a deep learning model for the classification of brain tumors from MRI images using convolutional neural network (CNN) based on transfer learning. The implemented system explores a few CNN architectures, image pre-processing and transfer learning model named Mobil Net to achieve the better performance and accuracy.

CHAPTER 1

INTRODUCTION

A brain tumor is an abnormal growth or mass of cells in or around the brain. It is also called a Central Nervous System Tumor. Brain tumors can be malignant (cancerous) or benign (not cancerous), Some tumors grow quickly, others are slow growing. Tumor in the brain can seriously disrupt the central nervous system. Furthermore, the mass of tumor-cells can affect the brain's regular functionalities. It should also be noted that many types of tumors make the brain tissue subjected to a scaling-up occurring over time, which leads to brain cells damage.

The cause of brain tumors is exposure to large amounts of radiation from X-rays or earlier cancer treatment. Some brain tumors occur when hereditary conditions are passed down among family members and symptoms of a brain tumor vary depending on the tumor's location and type, size and what the affected part of the brain controls. It is observed that brain tumors occur more often in men than women.

Although they are most common among older adults, they can develop at any age. However, early discovery of brain tumors helps significantly improve the possibility of treatment and survival rate of the patients. In spite of this, manual classification of tumor using a significant quantity of MRI scans, generated in clinical routine, is time and labor consuming task. In fact, the use of the magnetic resonance imaging (MRI) technique in medicine produces high quality images. This kind of imaging is often used by scientists in detecting brain tumors and showing their progress over time.

MRI images play a crucial role in automatic medical analysis field as they facilitate visualizing the different brain structure, thus providing detailed information about it. Scientists have developed different techniques for detecting and classifying brain tumors using MRI images. These approaches range from classical medical image processing to advanced machine learning techniques.

Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans like learn by example. In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Deep learning models are trained by using large sets of labeled data and neural network architectures that learn features directly from the data without the need for manual feature extraction. In ML Algorithms everything is flattened and in single dimension array while in deep learning uses something called a tensor. It has basically small matrices inside a big matrix, so it can be considered as matrix nested inside a matrix.

Deep learning is a specialized form of machine laming. A machine learning workflow relevant features are automatically extracted from images. In addition, deep learning is performed end-to-end where a network is given raw data and a task to perform, such as classification, and it learns how to do this automatically.

A key advantage of deep learning networks is that they often continue to improve as the size of your data increases. Deep learning has been applied in many applications, such as pattern classification object detection, speech recognition and other decision-making tasks. However, the main challenge for DL is the huge amount of data necessary for training.

1.1. Classification of Tumor

There are three basic types of tumors: 1) Benign; 2) Pre-Malignant; 3) Malignant (cancer can only be malignant)

1.1.1. Benign Tumor

A Benign Tumor is not always Malignant or cancerous. It might not invade close tissue or unfold to alternative components of the body the way cancer can. In most cases, the outlook with benign tumors is not at all serious but it can be serious if it presses on vital structures such as blood vessels or nerves.

1.1.2. Pre-Malignant Tumor

In these tumors, the cells are not cancerous. However, they need the potential to become malignant. The cells will grow and unfold to alternative components of the body.

1.1.3. Malignant Tumor

Malignancy (mal- = “bad” and ignis = “fire”) Malignant tumors area is cancerous. They develop once cells grow uncontrollably. If the cells still grow and unfold, the malady will become dangerous. Malignant tumors will grow quickly and unfold to alternative components of the body during a method known as metastasis.

The latest research in the year 2021 says that in United States among 24530 adults (13840 men & 10690 Women) will be identified with cancerous tumors of brain and in the spinal cord. A person's probability of developing this type of brain tumor in their lifespan is less than 1%. It causes 85% to 90% of all primary central nervous system (CNS) tumors. A number of 3,460 children under the age of 15 will also be identified with a brain or CNS tumor this year, other than this deals with adult primary brain tumors. Brain and alternative system nervous cancer is the tenth leading reason behind death for men and women. It is evaluated that 18,600 adults (10,500 men & 8,100 women) may die from primary cancerous brain and CNS tumors in the year 2021.

Hence, it's important to improve the accuracy of previously proposed methods for the betterment of medical image research. In our project, our proposed CNN-based algorithm is 90.74% accurate, it will help medical representatives in their treatment job without manually analyzing the MRI images so that the treatment speed can be enhanced.

CHAPTER 2

PROBLEM STATEMENT

Brain detection in real-time is a very challenging task. As our desired object is a small size, detecting it in an image is also very challenging in the presence of other objects, especially those objects that can be confused with it. Deep learning models faced several below mentioned challenges for detection and classification task: The first and main problem is the data through which CNN learn its features to be used later for classification and detection. Manually was a very long and time-consuming process. Labeling the desired database is not an easy task, as all data needs to be labeled manually.

2.1 EXISTING SYSTEM

MRI, also known as Magnetic Resonance Imaging, is mostly used for brain tumor or lesion detection. Brain tumor segmentation from MRI is one of the most crucial tasks in medical image processing as it generally involves a considerable amount of data. Moreover, tumors can be ill-defined with soft tissue boundaries. So, it is a very extensive task to obtain the exact segmentation of tumors from the human brain.

The large-scale manual examination method can often lead to misinterpretation due to some factors such as fatigue and excessive abundance of MRI slices. In addition, it is non-repeatable and results in intra- and inter-reader variability. Alleviating these concerns requires developing a detection system method to diagnose various brain abnormalities.

2.2 PROPOSED SYSTEM

The human brain is modelled by using design and implementation of neural network. The neural network is mainly used for vector quantization, approximation, data clustering, pattern matching, optimization functions and classification techniques. The neural network is divided into three types based on their interconnections. Three type neural networks are feedback, feed forward and recurrent network. The Feed Forward Neural network is further divided into single layer network and multilayer network. In the single layer network, the hidden layer is not presented. But it contains only input and output layer. However, the multilayer consists of input layer, hidden layer, and output layer. The closed loop-based feedback network is called as recurrent network.

In the normal neural network, image cannot scalable. But in convolution neural network, image can scalable i.e., it will take 3D input volume to 3D output volume (length, width, height)). The Convolution Neural Network (CNN) consists of input layer, convolution layer, Rectified Linear Unit (ReLU) layer, pooling layer and fully connected layer. In the convolution layer, the given input image is separated into various small regions. Element wise activation function is carried out in ReLU layer. Pooling layer is optional. However, the pooling layer is mainly used for down sampling. In the final layer i.e., fully connected layer is used to generate the class score or label score value based on the probability in-between 0 to 1.

An efficient and skilful method is proposed which helps in the segmentation and detection of the brain tumor without any human aid based on Convolutional Neural Network. It aims to classify brain MRI images into two classes, images with tumor and images without tumor or healthy.

First, different pre-processing steps are applied to the MRI images for image augmentation and enhancement. The original dataset consists of MRI images from which some of them are having tumor and some of them are non-tumorous. The datasets are further split into train, validation, and test sets. The pre-trained CNN architectures are used to test and evaluate the proposed model.

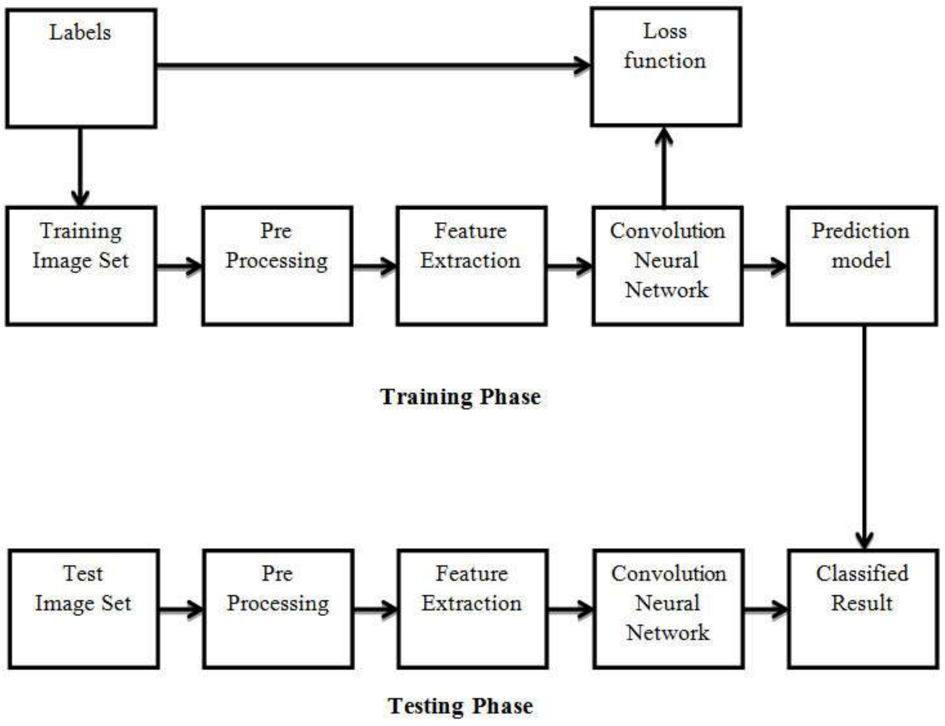

Figure 2.1 Block Diagram of Proposed Brain Tumor Classification Using CNN

The block diagram of brain tumor classification based on convolution neural network is shown in figure.1.1. The CNN based brain tumor classification is divided into two phases such as training and testing phases. The number of images is divided into different category by using labels name such as tumor and non-tumor brain image…etc. In the training phase, preprocessing, feature exaction and classification with Loss function is performed to make a prediction model. Initially, label the training image set. In the preprocessing image resizing is applied to change size of the image.

Finally, the convolution neural network is used for automatic brain tumor classification. The brain image dataset is taken from image net. Image net is a one of the pre-trained models. If you want to train from the starting layer, train the entire layer i.e., up to ending layer. So, time consumption is very high. It will affect the performance. To avoid this kind of problem, pre-trained model-based brain dataset is used for classification steps. So, computation time is low meanwhile the performance is high in the proposed automatic brain tumor classification scheme.

The loss function is calculated by using gradient descent algorithm. The raw image pixel is mapping with class scores by using a score function. The quality of set of parameters is measured by loss function. It is based on how well the induced scores approved with the ground truth labels in the training data. The loss function calculation is very important to improve the accuracy. If the loss function is high, when the accuracy is low. Similarly, the accuracy is high when the loss function is low. The gradient value is calculated for loss function to compute gradient descent algorithm. Repeatedly evaluate the gradient value to compute the gradient of loss function.

Steps In CNN Based Classification

- Apply convolution filter in first layer.
- The sensitivity of filter is reduced by smoothing the convolution filter i.e., subsampling.
- The signal transfers from one layer to another layer is controlled by activation layer.
- Fasten the training period by using rectified linear unit (RELU).
- The neurons in proceeding layer is connected to every neuron in subsequent layer.

- During training Loss layer is added at the end to give feedback to neural network.

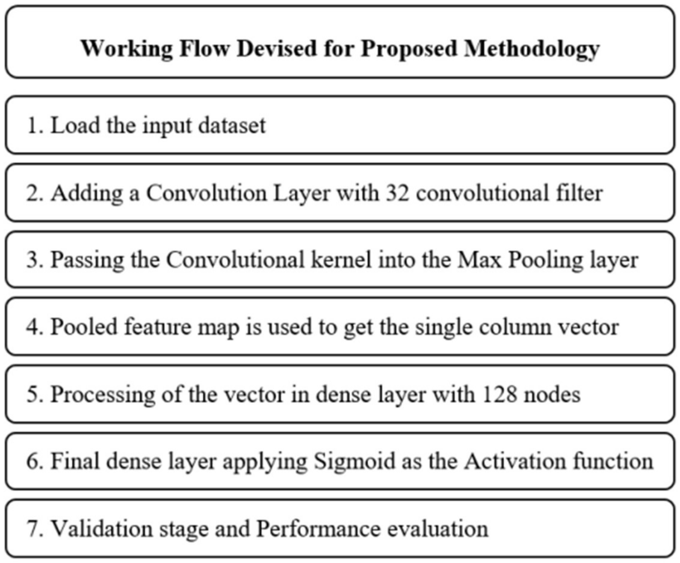
Working Flow Devised for Proposed Methodology
1. Load the input dataset
2. Adding a Convolution Layer with 32 convolutional filter
3. Passing the Convolutional kernel into the Max Pooling layer
4. Pooled feature map is used to get the single column vector
5. Processing of the vector in dense layer with 128 nodes
6. Final dense layer applying Sigmoid as the Activation function
7. Validation stage and Performance evaluation

Figure 2.2. Working Flow of The Proposed CNN Model

CHAPTER 3

SPECIFICATION

3.1 HARDWARE REQUIREMENTS

- Processor: i3 or above.
- Ram: 4GB or above.
- Hard Disk: 100GB or above.

3.2 SOFTWARE REQUIREMENTS

The software requirements are a description of features and functionalities of the target system. Requirements convey the expectations of users from the software product. The requirements can be obvious or hidden, known or unknown, expected, or unexpected from the client's point of view. The key software requirements needed for the project are:

- Python

  Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.

- Anaconda

  Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment.

- Jupyter Notebook

  The Jupyter Notebook is an open-source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter.

  Jupyter Notebooks are a spin-off project from the IPython project, which used to have an IPython Notebook project itself. The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python, and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use.

- Pandas

  Pandas is a software library written for the Python programming language for data manipulation and analysis. It offers data structures and operations for manipulating numerical tables and time series. Pandas is a Python library used for working with data sets.

- Matplotlib

  Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython.

- NumPy

  NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. It also discusses the various array functions, types of indexing, etc.

CHAPTER 4

DESIGN

4.1 SYSTEM ARCHITECTURE

To bring down whole knowledge of requirements and analysis on the desk and design the software product. The inputs from users and information gathered in requirement gathering phase are the inputs of this step. The output of this step comes in the form of two designs: logical design and physical design. Engineers produce meta-data and data dictionaries, logical diagrams, data-flow diagrams and in some cases pseudo codes.

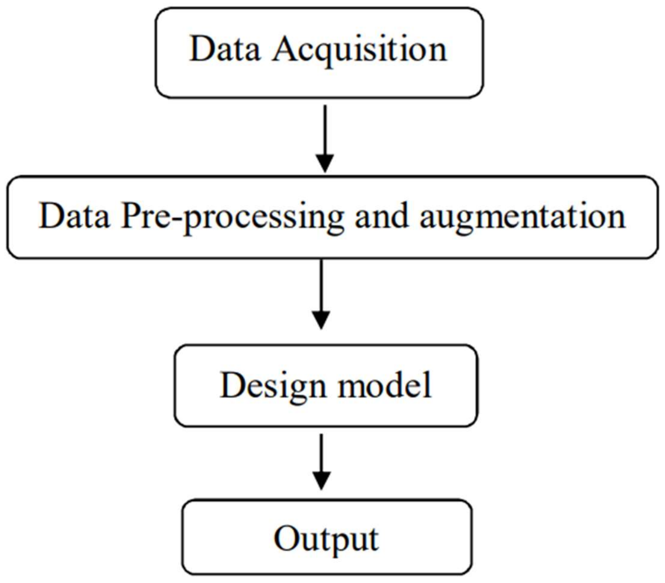

Fig 4.1.1 System Architecture Diagram

4.2 DATA ACQUISITION

The dataset used for training and testing was collected from Kaggle. It contains brain MRI images in which some of them are images containing tumor (tumorous images) and some images are normal (without tumor). Tumorous images are segregated in folder named “Yes” and normal images are kept in “No” folder. The images are in different formats and of variable sizes.

4.3 DATA PRE-PROCESSING AND AUGMENTATION

For any machine learning project data pre-processing is the most crucial and initial step. In this the raw data was collected and making it useful for machine learning model. As mentioned, the dataset contains images of different formats and sizes which may contain noise. This can lead to errors in classification and segmentation. Pre-processing the image will reduce this problem and data can be transformed in a standard format acceptable for classification and segmentation.

Deep Learning needs a large dataset for producing accurate results. Image augmentation is a process of increasing the size of the dataset by producing copies of images through different ways of processing like random rotation, shifts, shear, and flips by using the Image Data Generator tool in Keras TensorFlow. This process boosts the model to generalize better and helps prevent overfitting.

4.3 DESIGN MODEL

CNN, transfer learning and its architecture called Mobilenet are used to design the model which is used to improve the accuracy of our model. Before training the model, the whole dataset was divided into three parts called Training, Testing and Validation Dataset. Training data is used to train the model and Testing data and Validation data was used to test the model. In this project 70% of the data was taken as training data and 15% was taken as testing data as well as validation data.

4.4 DEPLOYMENT

The last step of machine learning life cycle is deployment, where deploying is done in the real-world system. If the above-prepared model is producing an accurate result as per our requirement with acceptable speed, then the deployment is done. But before deploying the project, a check is insisted on whether it is improving its performance using available data or not. The deployment phase is similar to making the final report for a project.

CHAPTER 5

ALGORITHMS

5.1 CNN CLASSIFIER ALGORITHM:

A CNN is a kind of network architecture for deep learning algorithms and is specifically used for image recognition and tasks that involve the processing of pixel data. There are other types of neural networks in deep learning, but for identifying and recognizing objects, CNNs are the network architecture of choice.

Step 1: Choosing the Datasets.

Step 2: Preparing Dataset for training.

Step 3: Create dataset.

Step 4: Assigning Labels and Features.

Step 5: Normalizing X and converting labels to categorical data.

Step 6: Split X and Y for use in CNN.

Step 7: Define, compile, and train the CNN model.

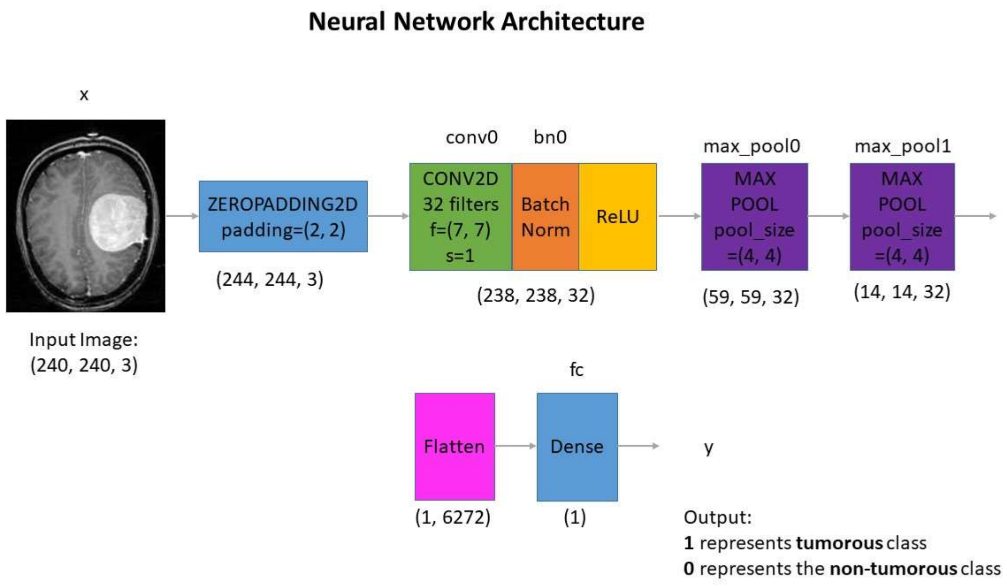

Fig 5.1.1 CNN Architecture

CHAPTER 6

IMPLEMENTATION

6.1 DATA ACQUISITION:

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.

6.2 DATA AUGMENTATION:

Since this is a small dataset, there weren't enough examples to train the neural network. Also, data augmentation was useful in tackling the data imbalance issue in the data. Before data augmentation, the dataset consisted of 155 positive and 98 negative examples, resulting in 253 example images. After data augmentation, the dataset now consists of 1085 positive and 980 examples, resulting in 2065 example images.

Note: these 2065 examples contain also the 253 original images.

6.3 DATA PREPROCESSING:

For every image, the following preprocessing steps were applied:

1. Crop the part of the image that contains only the brain (which is the most important part of the image).
2. Resize the image to have a shape of (240, 240, 3) = (image_width, image_height, number of channels): because images in the dataset come in different sizes. So, all images should have the same shape to feed it as an input to the neural network.
3. Apply normalization: to scale pixel values to the range 0-1.

6.4 DATA SPLIT:

The data was split in the following way:

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

6.5 CNN ARCHITECTURE:

Each input x (image) has a shape of (240, 240, 3) and is fed into the neural network. And it goes through the following layers:

1. A Zero Padding layer with a pool size of (2, 2).
2. A convolutional layer with 32 filters, with a filter size of (7, 7) and a stride equal to 1.
3. A batch normalization layer to normalize pixel values to speed up computation.
4. A ReLU activation layer.
5. A Max Pooling layer with f=4 and s=4.
6. A Max Pooling layer with f=4 and s=4, same as before.
7. A flatten layer in order to flatten the 3-dimensional matrix into a one-dimensional vector.
8. A Dense (output unit) fully connected layer with one neuron with a sigmoid activation (since this is a binary classification task).

6.6 TRAINING THE MODEL:

The model was trained for 24 epochs with the batch size of 32 to get the best results.

CHAPTER 7

RESULT

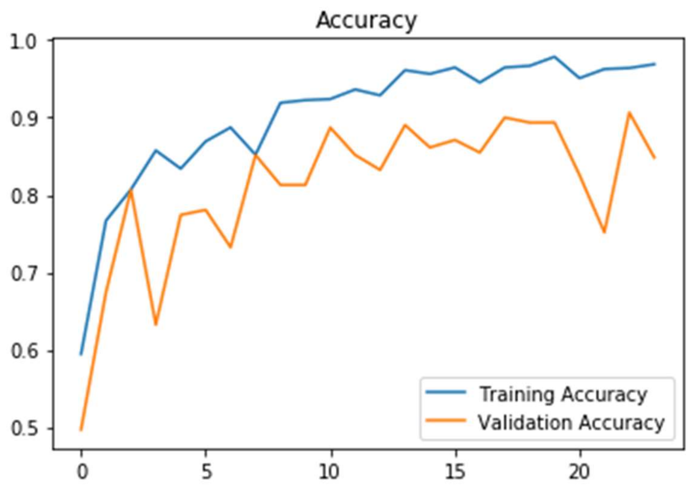

Fig 7.1 Accuracy Line Graph

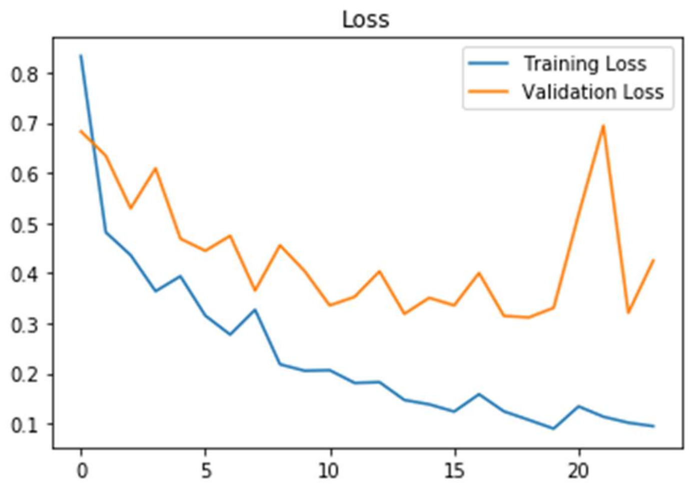

Fig 7.2 Loss Line Graph

The best validation accuracy was achieved on the **23**rd iteration.

CHAPTER 8

CONCLUSION AND FUTURE SCOPE

8.1 CONCLUSION:

Brain tumor detection system was developed using deep learning to detect tumors and find accuracy. Here, an efficient method for automatic brain tumor classification using MRI images is used. The method is based on Transfer Learning and implemented on well-known CNN architectures. Transfer Learning has the benefit of decreasing the training time for a neural network model and can result in lower generalization error. The time-consuming process of brain tumor detection is thus simplified by automation. The comparative analysis of training accuracy versus validation accuracy is done. An accuracy of 89% on testing data and 91% on validation data is achieved by the proposed model for detecting brain tumor.

8.2 FUTURE SCOPE:

In future work, a model can be built based on brain tumor for detecting several types of brain tumors and their region and how much percentage of the brain is affected through the cancerous cell. For ease of use, it can also be further deployed as an application.

CHAPTER 9

APPENDIX

Importing Necessary Modules

```
Import tensorflow as tf
from tensorflow.keras.layers import Conv2D, Input, ZeroPadding2D, BatchNormalization, 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
%matplotlib inline
```

Data Preparation & Preprocessing

```
def crop_brain_contour(image, plot=False):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    gray = cv2.GaussianBlur(gray, (5, 5), 0)
    thresh = cv2.threshold(gray, 45, 255, cv2.THRESH_BINARY)
    thresh = cv2.erode(thresh, None, iterations=2)
    thresh = cv2.dilate(thresh, None, iterations=2)
    cnts =cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    cnts = imutils.grab_contours(cnts)
    c = max(cnts, key=cv2.contourArea)
    extLeft = tuple(c[c[:, :, 0].argmin()][0])
    extRight = tuple(c[c[:, :, 0].argmax()][0])
    extTop = tuple(c[c[:, :, 1].argmin()][0])
    extBot = tuple(c[c[:, :, 1].argmax()][0])
    new_image = image[extTop[1]:extBot[1], extLeft[0]:extRight[0]]
    if plot:
        plt.figure()
        plt.subplot(1, 2, 1)
        plt.imshow(image)
        plt.tick_params(axis='both', which='both',
        top=False, bottom=False, left=False, right=False,
        labelbottom=False, labeltop=False, labelleft=False, labelright=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, labelright=False)
        plt.title('Cropped Image')
        plt.show()
    return new_image
ex_img = cv2.imread('yes/Y1.jpg')
ex_new_img = crop_brain_contour(ex_img, True)
```

Load up the data:

```
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):
            image = cv2.imread(directory + '\\' + filename)
            image = crop_brain_contour(image, plot=False)
            image = cv2.resize(image, dsize=(image_width, image_height), interpolation=cv2.INTER_CUBIC)
            image = image / 255.

            X.append(image)
            if directory[-3:] == 'yes':
                y.append([1])
            else:
                y.append([0])
```

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

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

augmented_path = 'augmented data/'


augmented_yes = augmented_path + 'yes'
augmented_no = augmented_path + 'no'




IMG_WIDTH, IMG_HEIGHT = (240, 240)
X, y = load_data([augmented_yes, augmented_no], (IMG_WIDTH, IMG_HEIGHT))
```

Split the data:

```
    def split_data(X, y, test_size=0.2):
            X_train, X_test_val, y_train, y_test_val = train_test_split(X, y, test_size=test_size)
        X_test, X_val, y_test, y_val = train_test_split(X_test_val, y_test_val, test_size=0.5)


      return X_train, y_train, X_val, y_val, X_test, y_test

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))

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)}"
```

Build the model:

```
 def build_model(input_shape):
X_input = Input(input_shape)

X = ZeroPadding2D((2, 2))(X_input)
X = Conv2D(32, (7, 7), strides = (1, 1), name = 'conv0')(X)
X = BatchNormalization(axis = 3, name = 'bn0')(X)
X = Activation('relu')(X)
X = MaxPooling2D((4, 4), name='max_pool0')(X)
X = MaxPooling2D((4, 4), name='max_pool1')(X)
X = Flatten()(X)
X = Dense(1, activation='sigmoid', name='fc')(X)
model = Model(inputs = X_input, outputs = X, name='BrainDetectionModel'
return model
```

Train the model:

```
start_time = time.time()
model.fit(x=X_train, y=y_train, batch_size=32, epochs=24, 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)}")
```

Plot Loss & Accuracy:

```
def plot_metrics(history):

  train_loss = history['loss']
  val_loss = history['val_loss']
  train_acc = history['acc']
  val_acc = history['val_acc']
  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()
```

Accuracy of the best model on the testing data:

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

The models are stored as .model files. They can be restored as follows:

```
from tensorflow.keras.models import load_model
best_model = load_model(filepath='models/cnn-parameters-improvement-23
0.91.model')
```

CHAPTER 10

REFERENCES

1. Sufiyan Salim Akbani, Adeeba Naaz, Nazish Kausar, Prof.Abdul Razzaque, **“Brain Tumor Detection using Deep Learning**" 2022 International Journal for Research In Applied Science and Engineering Technology | Volume 10 Issue IV Apr 2022 | ISSN: 2321-9653.
2. Ali Mohammad Alqudah, Hiam Alquraan, Isam Abu Qasmieh, Amin Alqudah, Wafaa Al-Sharu,"**Brain tumor detection using deep learning technique- a comparison between cropped, uncropped, segmented lesion images with different sizes**" 2019 International Journal of Advanced Trends in Computer Science and Engineering |Volume 8 Issue VI Nov-Dec 2019| ISSN: 2278-3091.
3. Dr. Ganesh Khekare, Yash Kumar Patel, Patel Nisht, Darsh Engineer, Prajapati Badal, "**Brain Tumor Detection using Data Science**", 2022 International Journal of Engineering Applied Sciences and Technology | Volume 6, Issue IX|ISSN: 2455-2143.
4. Khurram Shahzad and Imran Siddique “**Efficient Brain Tumor Detection Using Image Processing Techniques”,** International Journal of Scientific & Engineering Research | Dec 2019.
5. Masoumeh Siar “**Brain Tumor Detection Using Deep Neural Network and Machine Learning Algorithm**”, International Conference on Computer and Knowledge Engineering | October 2019.
6. Sneha Grampurohit “**BRAIN TUMOR DETECTION USING DEEP LEARNING MODELS”,** Institute of Electrical and Electronics Engineers | May 2020.
7. T. Logeswari and M. Karnan, “**An improved implementation of Brain Tumor Detection using segmentation based on hierarchical self organizing map**”,Int. J. Comput. Theory Eng | Volume 2, No.4 May 2010.