**How Code Works?**

**Code:**

import numpy as np

import cv2

import time

import os

import sys

from PIL import Image

from random import shuffle

import pandas as pd

file\_name = '../Data/training\_data.npy'

if os.path.isfile(file\_name):

print("File exists, loading previous data")

training\_data = list(np.load(file\_name))

else:

print("File does not exist, starting fresh")

training\_data = []

npath = "../Dataset/brain-mri-images-for-brain-tumor-detection/no/"

ypath = "../Dataset/brain-mri-images-for-brain-tumor-detection/yes/"

no = [os.path.join(npath, f) for f in os.listdir(npath)]

yes = [os.path.join(ypath, f) for f in os.listdir(ypath)]

**Explanation:**

In the first 8 lines of code are related to importing the necessary files. “Numpy” is the package which is used for the management of the array files. “cv2” is the Open CV package which is related for the management of the computer vision related work like reading the images, converting the images into array format and making necessary changes to the images. “Time” is for the time management. “OS” /” Sys” is for the management of the files in the OS and also for importing the data in the folders. “PIL” is the Image management library. “Random” is to make some random changes. “Pandas” is the data management library which is also used to manipulated the data in the csv format.

Next, I am creating a “Numpy” file to save the data. We need to check whether the file already exists or not, if it exists we need to append the data to that file else we need to create and add data, and save the file. “Npath” is for the list of image paths where the images are tended to “No”, “Ypath” is for the list of image paths where the images are tended to “Yes”.

**Code:**

for file in no:

try:

img = cv2.imread(file)

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

cv2.imshow('img',img)

img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

img = cv2.resize(img, (30,30))

#img = cv2.imread('Images/'+file)

output = 0

training\_data.append([img, output])

print('Frame took {} seconds'.format(time.time()-last\_time))

last\_time = time.time()

if cv2.waitKey(30) & 0xff == 'q' == 27:

break

except:

np.save(file\_name, training\_data)

for file in yes:

try:

img = cv2.imread(file)

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

cv2.imshow('img',img)

img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

img = cv2.resize(img, (30,30))

#img = cv2.imread('Images/'+file)

output = 1

training\_data.append([img, output])

print('Frame took {} seconds'.format(time.time()-last\_time))

last\_time = time.time()

if cv2.waitKey(30) & 0xff == 'q' == 27:

break

except:

np.save(file\_name, training\_data)

np.save(file\_name, training\_data)

cv2.destroyAllWindows()

**Explaination:**

First for loop is to read the images along with the class name as “NO” and the image data along with its class is appended to an array “training\_data”. In the next for loop also images along with the class name as “YES” and the image data along with its class is appended to an array “training\_data”. After the above process the data is saved into the “Numpy file”.

**Code:**

train\_data = np.load('../Data/training\_data.npy')

TOTAL = []

for img, ind in train\_data:

TOTAL.append([img, ind])

if cv2.waitKey(30) & 0xff == 'q' == 27:

break

cv2.destroyAllWindows()

shuffle(TOTAL)

np.save('../Data/training\_data\_cleaned.npy', TOTAL)

**Explaination:**

The saved data need to be shuffled for the creation of training and testing dataset.

**Code:**

import tensorflow as tf

from tensorflow import keras

import matplotlib.pyplot as plt

# Import the data

data\_name = '../Data/training\_data\_cleaned.npy'

data = np.load(data\_name)

# Declare the required arrays

img = []

label = []

test\_imgs = []

test\_labs = []

# Class names

class\_names = ['Benign', 'Malignant']

# Input to the arrays

for item, index in data:

img.append(item)

label.append(index)

# Train and Test data

train\_images = img

train\_labels = label

test\_images = img[-100:]

test\_labels = label[-100:]

train\_images = np.asarray(train\_images)

test\_images = np.asarray(test\_images)

train\_images = train\_images.reshape((-1, 30, 30, 1))

test\_images = test\_images.reshape((-1, 30, 30, 1))

# Image Processing

train\_images = train\_images / 255.0

test\_images = test\_images / 255.0

**Explaination:**

The shuffled data is read in the array format and made into train data and test data. But for the images in both the train and test data we need to reshape the images as required for the Convolutional Neural Network Model.

**Code:**

# Sequential Model

# Convolutional Neural Network

model = keras.Sequential([

keras.layers.Conv2D(32, (2, 2), activation='relu', input\_shape=(30, 30, 1)),

keras.layers.MaxPooling2D((2, 2)),

keras.layers.Conv2D(64, (3, 3), activation='relu'),

keras.layers.MaxPooling2D((2, 2)),

keras.layers.Conv2D(64, (3, 3), activation='relu'),

keras.layers.Dropout(0.25),

keras.layers.Flatten(),

keras.layers.Dense(512, activation=tf.nn.relu),

keras.layers.Dense(128, activation=tf.nn.relu),

keras.layers.Dense(128, activation=tf.nn.relu),

keras.layers.Dense(len(class\_names), activation=tf.nn.softmax)

])

# Compile the model

model.compile(optimizer=tf.train.AdamOptimizer(),

loss='sparse\_categorical\_crossentropy',

metrics=['accuracy'])

# Train the Model

history = model.fit(train\_images, train\_labels, epochs = 15)

# Save Model

model.save('../Model/BrainTumorDetection-Tensorflow.model')

# Print the Summary

model.summary()

# Accuracy of the Model

test\_loss, test\_acc = model.evaluate(test\_images, test\_labels)

# Print Test accuracy

print('Test accuracy:', test\_acc\*100)

# Make Predictions

predictions = model.predict([test\_images])[0]

predicted\_label = class\_names[np.argmax(predictions)]

# Compare the predictions

print("Predictions : ",predicted\_label)

print("Actual : ", class\_names[test\_labels[0]])

##print(history.history.keys())

# summarize history for accuracy

plt.plot(history.history['acc'])

plt.plot(history.history['loss'])

**Explaination:**

The Convolutional Neural Network Model is build using the “Tensorflow library” which is a Google’s Neural Network Framework. In the first layer of the model, **Conv2D** is defined which is a 32 layers Convolutional Neural Network layer with the activation function **relu** which is a short form for Rectified Linear. The activation function is used to activate the Neural Network layer to work as per the given instructions. Next, **MaxPooling2D** is for the pooling of image. Pooling is the process of compression of images in the smaller format to for the easy identification of the specification in the image. In the next lines the same **Conv2D** is repeated twice with 64 layers of Neurons. At last the weights of the neural network layer is reduced by 25% using Dropout method. In the next line the total Convolutional model is flatten and passed through 3 layers of Sequential Dense layers with 512, 128, 128 neurons respectively. At last the final line is used for the classifications which is activated by **SoftMax** activation function. The whole model is compiled using Adam Optimizer and the metrics of loss are evaluated using **Sparse cross entropy** method. The whole model is fitted by compiling it with 15 iterations using train dataset and test dataset. Finally, the model is saved as **BrainTumorDetection-Tensorflow.model.** The summary of the training is printed using summary function. And the model accuracy is evaluated and the test accuracy is printed. The training accuracy is plotted using **matplotlib** library which is a predefined plotting library.

**Code:**

Im = “image.jpg”

Class\_names = [‘No’, ‘Yes’]

model = tf.keras.models.load\_model('../Model/BrainTumorDetection-Tensorflow.model')

img = cv2.imread(im)

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

cv2.imshow('img',img)

img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

img = cv2.resize(img, (30,30))

img = np.asarray(img)

img = img.reshape((-1, 30, 30, 1))

img = img / 255.0

prediction = model.predict([img])[0]

predicted\_label = class\_names[np.argmax(prediction)]

print(predicted\_label)

if cv2.waitKey(30) & 0xff == 'q' == 27:

break

**Explaination:**

The saved model is read and the new image path is given. The given image is read using OpenCV method **imread** and changed into input format which is accepted by the Convolutional Neural Network Model. And the prediction is performed.