BRAIN TUMOR DETECTION OF CT-SCANS

VIT-AP UNIVERSITY

Name : **Gandrapu Sri Sai Lahari**
Reg No : **18BCN7120**
Date : **13-12-2021**
Guided by : **G.Muneeswari**

CONTENT

INTRODUCTION

- Brain tumor is one of the most dangerous diseases occurring among the human beings. Brain CT Scan plays a very important role for radiologists to diagnose and treat brain tumor patients.
- In this project, I built a Brain tumor detection system that can identify tumor in CT Scan images using several image processing techniques.
- Implementing this kind of systems in hospitals which has to be done without human intervention will be a great help for doctors.
- I have trained the dataset with different deep learning models and tried to look for the best possible technique for more accuracy.

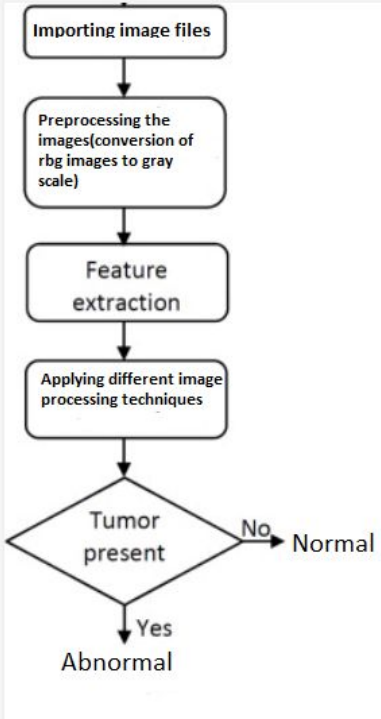
PROBLEM STATEMENT

- Study of the medical image by the radiologist is a time consuming process and also the accuracy depends upon their experience. Thus, the computer aided systems becomes very necessary as they overcome these limitations.
- Several automated methods are available, but automating this process is very difficult because of different appearance of the tumor among the different patients.
- There are various feature extraction and classification methods which are used for detection of brain tumor from CT Scan images.

PROPOSED SYSTEM

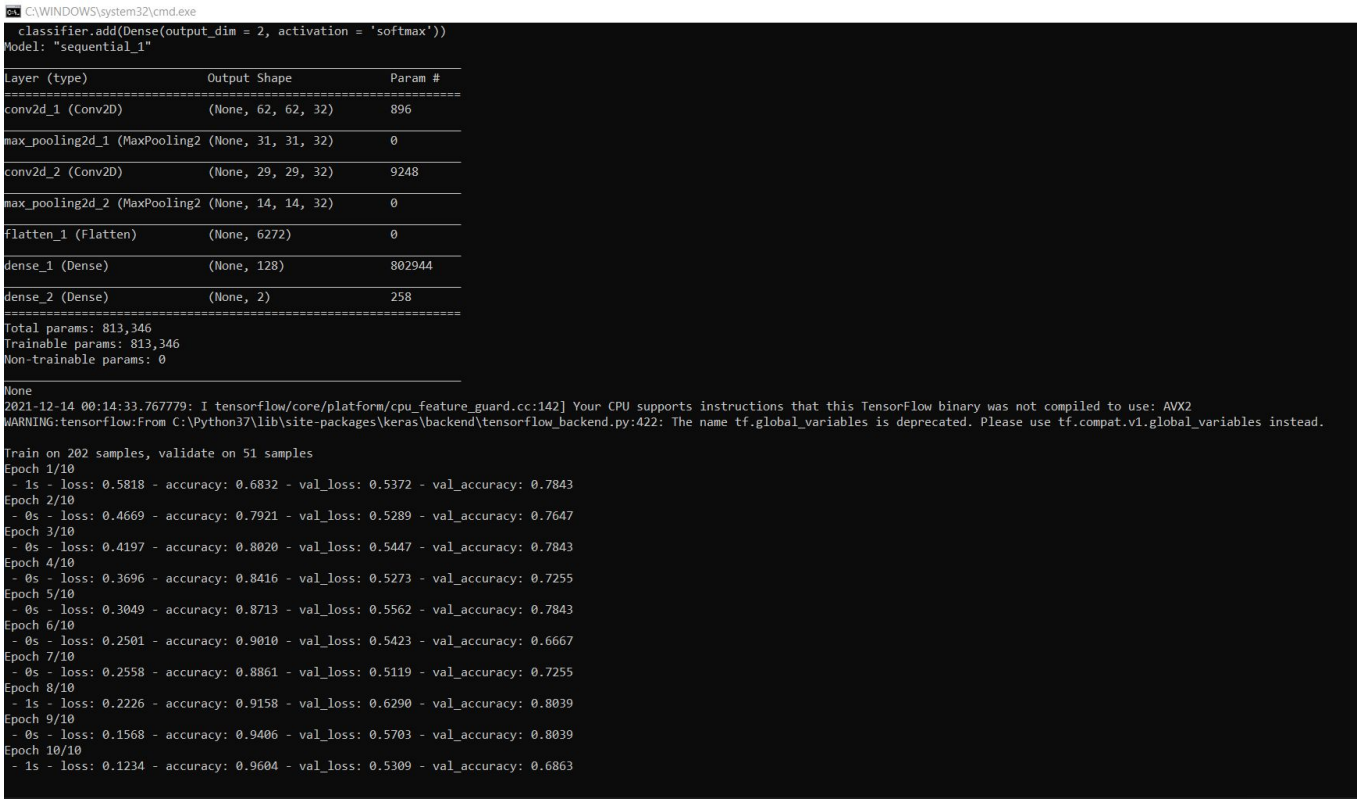
- I have trained the CT Scan images with gaussian filter, Adaptive median filtering and AMF gradient boosting algorithms, AMF GB got the highest accuracy among the three techniques.
- I have plotted the graphs for accuracy and error rate of the three techniques for better understanding.
- I have used tkinter to display the outputs using a batch file.
- The final output we get would be if the CT Scan is normal or abnormal, where normal indicates the absence of tumor and abnormal indicates presence of brain tumor.

DESIGN



After applying all the models we check for the best model by plotting accuracy graph and error rate graph

IMPLEMENTATION

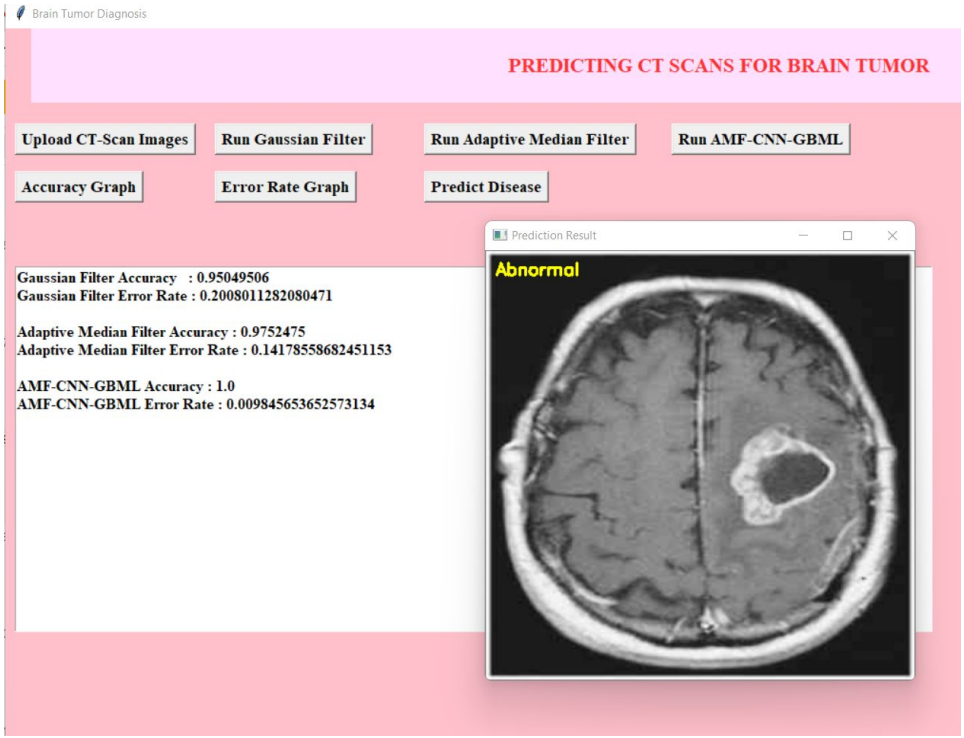


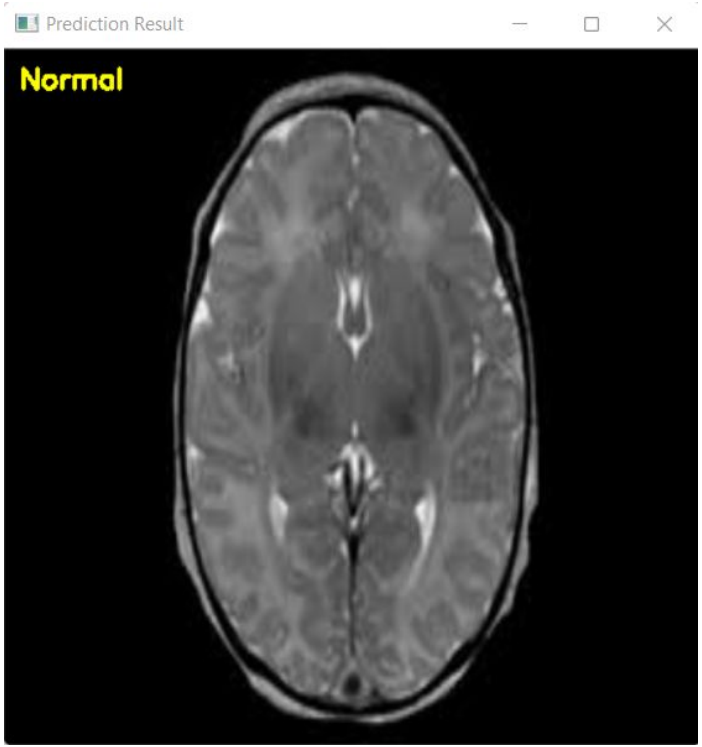
C:\WINDOWS\system32\cmd.exe

```
final_code.py:177: UserWarning: Update your `Dense` call to the Keras 2 API: `Dense(activation="relu", units=128)`
  classifier.add(Dense(output_dim = 128, activation = 'relu'))
final_code.py:178: UserWarning: Update your `Dense` call to the Keras 2 API: `Dense(activation="softmax", units=2)`
  classifier.add(Dense(output_dim = 2, activation = 'softmax'))
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 62, 62, 32)        896
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 31, 31, 32)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 29, 29, 32)        9248
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
flatten_3 (Flatten)          (None, 6272)              0
_________________________________________________________________
dense_5 (Dense)              (None, 128)               802944
_________________________________________________________________
dense_6 (Dense)              (None, 2)                 258
=================================================================
Total params: 813,346
Trainable params: 813,346
Non-trainable params: 0
_________________________________________________________________
None
Train on 202 samples, validate on 51 samples
Epoch 1/10
 - 1s - loss: 0.6253 - accuracy: 0.6931 - val_loss: 0.5219 - val_accuracy: 0.7843
Epoch 2/10
 - 0s - loss: 0.5307 - accuracy: 0.7921 - val_loss: 0.5116 - val_accuracy: 0.8039
Epoch 3/10
 - 1s - loss: 0.4548 - accuracy: 0.8267 - val_loss: 0.5060 - val_accuracy: 0.8039
Epoch 4/10
 - 1s - loss: 0.4131 - accuracy: 0.8218 - val_loss: 0.5468 - val_accuracy: 0.7647
Epoch 5/10
 - 1s - loss: 0.3703 - accuracy: 0.8564 - val_loss: 0.4894 - val_accuracy: 0.7647
Epoch 6/10
 - 1s - loss: 0.3079 - accuracy: 0.8663 - val_loss: 0.6568 - val_accuracy: 0.7647
Epoch 7/10
 - 1s - loss: 0.2927 - accuracy: 0.8911 - val_loss: 0.4832 - val_accuracy: 0.8039
Epoch 8/10
 - 1s - loss: 0.2279 - accuracy: 0.9010 - val_loss: 0.5336 - val_accuracy: 0.7647
Epoch 9/10
 - 0s - loss: 0.1804 - accuracy: 0.9307 - val_loss: 0.5965 - val_accuracy: 0.8235
Epoch 10/10
 - 0s - loss: 0.1418 - accuracy: 0.9752 - val_loss: 0.5824 - val_accuracy: 0.8039
```

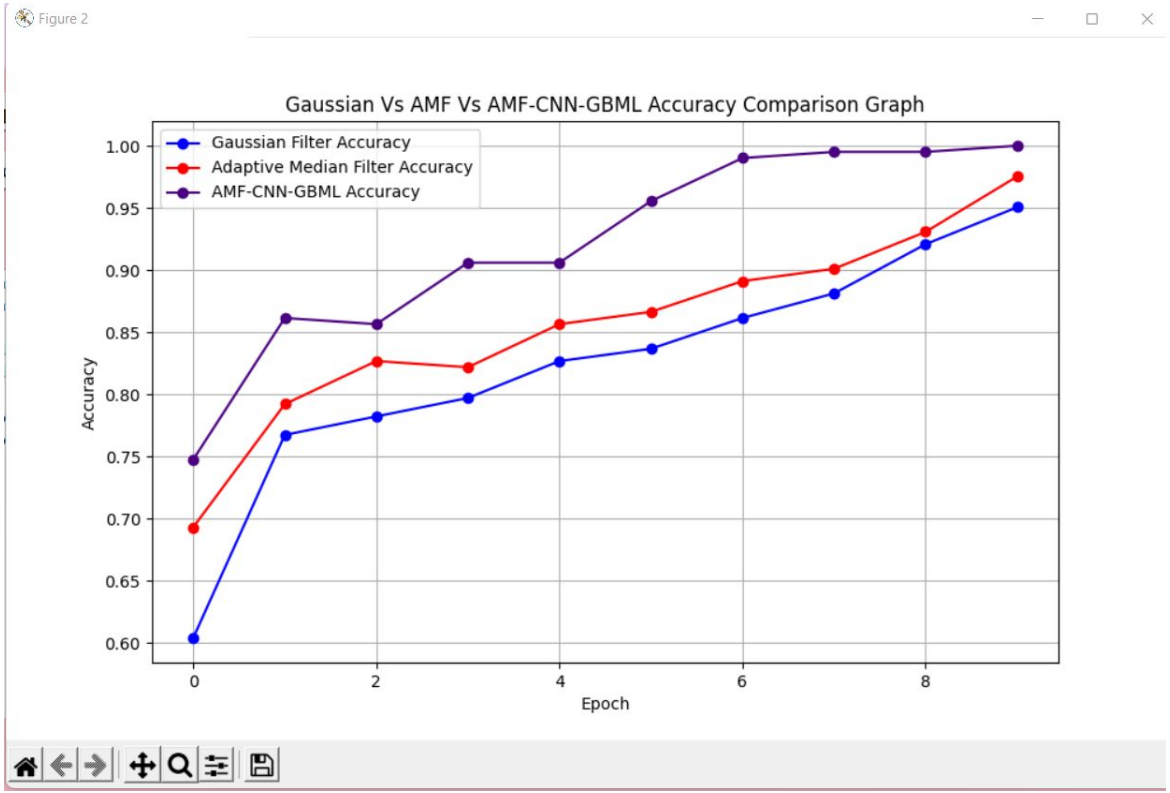
```
final_code.py:206: UserWarning: Update your `Dense` call to the Keras 2 API: `Dense(activation="relu", units=128)`
  classifier.add(Dense(output_dim = 128, activation = 'relu'))
final_code.py:207: UserWarning: Update your `Dense` call to the Keras 2 API: `Dense(activation="softmax", units=2)`
  classifier.add(Dense(output_dim = 2, activation = 'softmax'))
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_7 (Conv2D)            (None, 62, 62, 32)        896
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 31, 31, 32)        0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 29, 29, 32)        9248
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
flatten_4 (Flatten)          (None, 6272)              0
_________________________________________________________________
dense_7 (Dense)              (None, 128)               802944
_________________________________________________________________
dense_8 (Dense)              (None, 2)                 258
=================================================================
Total params: 813,346
Trainable params: 813,346
Non-trainable params: 0
_________________________________________________________________
None
Train on 202 samples, validate on 51 samples
Epoch 1/10
 - 1s - loss: 0.5305 - accuracy: 0.7475 - val_loss: 0.5408 - val_accuracy: 0.7843
Epoch 2/10
 - 1s - loss: 0.3970 - accuracy: 0.8614 - val_loss: 0.5300 - val_accuracy: 0.7647
Epoch 3/10
 - 1s - loss: 0.3332 - accuracy: 0.8564 - val_loss: 0.5184 - val_accuracy: 0.8039
Epoch 4/10
 - 1s - loss: 0.2421 - accuracy: 0.9059 - val_loss: 0.5956 - val_accuracy: 0.8431
Epoch 5/10
 - 1s - loss: 0.2164 - accuracy: 0.9059 - val_loss: 0.4818 - val_accuracy: 0.8235
Epoch 6/10
 - 1s - loss: 0.1174 - accuracy: 0.9554 - val_loss: 0.5359 - val_accuracy: 0.7647
Epoch 7/10
 - 1s - loss: 0.0628 - accuracy: 0.9901 - val_loss: 0.5973 - val_accuracy: 0.8039
Epoch 8/10
 - 1s - loss: 0.0340 - accuracy: 0.9950 - val_loss: 0.5272 - val_accuracy: 0.8039
Epoch 9/10
 - 1s - loss: 0.0200 - accuracy: 0.9950 - val_loss: 0.6067 - val_accuracy: 0.7647
Epoch 10/10
 - 1s - loss: 0.0098 - accuracy: 1.0000 - val_loss: 0.5834 - val_accuracy: 0.7647
```

RESULTS

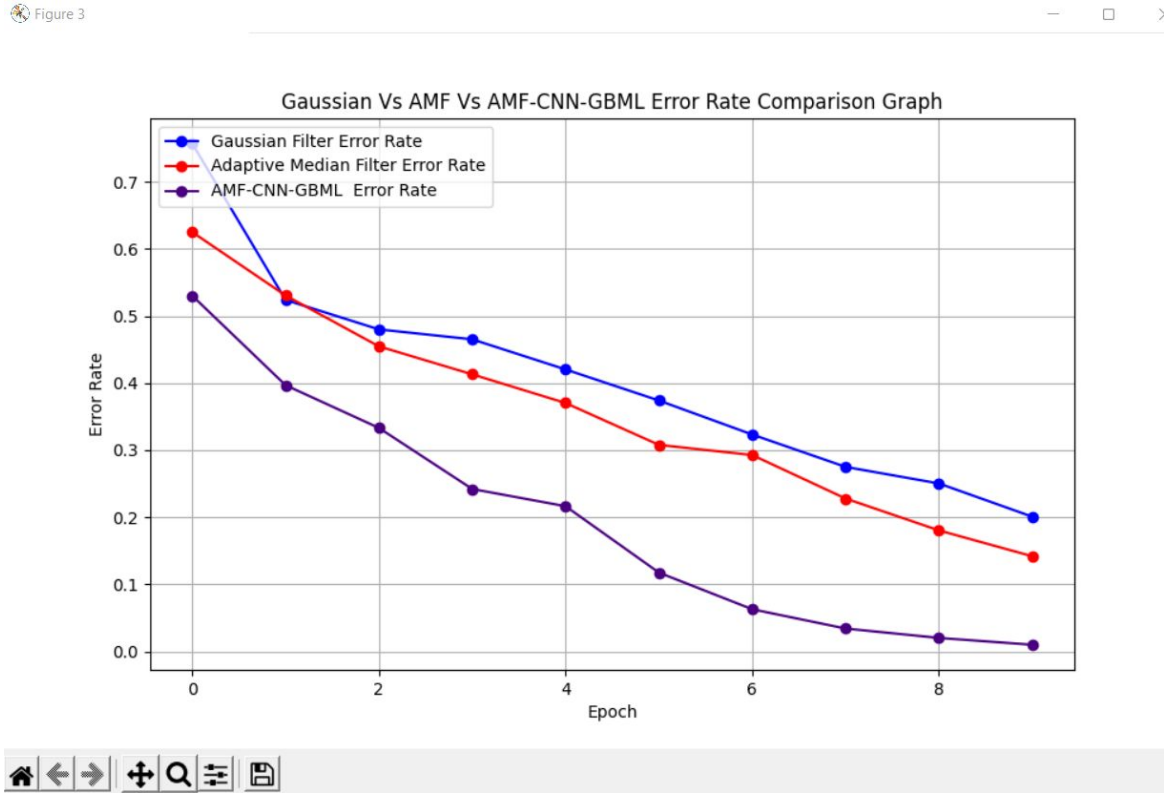

Brain Tumor Diagnosis
PREDICTING CT SCANS FOR BRAIN TUMOR
Upload CT-Scan Images
Run Gaussian Filter
Run Adaptive Median Filter
Run AMF-CNN-GBML
Accuracy Graph
Error Rate Graph
Predict Disease
Gaussian Filter Accuracy : 0.95049506
Gaussian Filter Error Rate : 0.2008011282080471
Adaptive Median Filter Accuracy : 0.9752475
Adaptive Median Filter Error Rate : 0.14178558682451153
AMF-CNN-GBML Accuracy : 1.0
AMF-CNN-GBML Error Rate : 0.009845653652573134
Prediction Result
Abnormal


Prediction Result
Normal

ACCURACY GRAPH



ERROR RATE GRAPH



CODE SCREENSHOTS

```
def runGaussian():
  global gaussian
  classifier = Sequential()
  classifier.add(Convolution2D(32, 3, 3, input_shape = (64, 64, 3), activation = 'relu'))
  classifier.add(MaxPooling2D(pool_size = (2, 2)))
  classifier.add(Convolution2D(32, 3, 3, activation = 'relu'))
  classifier.add(MaxPooling2D(pool_size = (2, 2)))
  classifier.add(Flatten())
  classifier.add(Dense(output_dim = 128, activation = 'relu'))
  classifier.add(Dense(output_dim = 2, activation = 'softmax'))
  classifier.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
  print(classifier.summary())
  gaussian = classifier.fit(X_gaussian, Y, batch_size=32, epochs=10, validation_split=0.2, shuffle=True, verbose=2)
  gaussian = gaussian.history
  acc = gaussian['accuracy']
  loss = gaussian['loss']
  text.delete('1.0', END)
  text.insert(END,'Gaussian Filter Accuracy   : '+str(acc[9])+"\n")
  text.insert(END,'Gaussian Filter Error Rate : '+str(loss[9])+"\n")
  img = cv2.imread('test.jpeg')
  gau = cv2.GaussianBlur(img,(5,5),0)
```

CODES SCREENSHOTS

```
def AMF():
  global amf
  image_org = Image.open("test.jpeg")
  image = np.array(image_org)
  grayscale_image = rgb2gray(image)
  output = adaptivemf(grayscale_image, 3, 11)
  output = cv2.medianBlur(output, 5)
  cv2.imwrite("clean.jpg",output)
  classifier = Sequential()
  classifier.add(Convolution2D(32, 3, 3, input_shape = (64, 64, 3), activation = 'relu'))
  classifier.add(MaxPooling2D(pool_size = (2, 2)))
  classifier.add(Convolution2D(32, 3, 3, activation = 'relu'))
  classifier.add(MaxPooling2D(pool_size = (2, 2)))
  classifier.add(Flatten())
  classifier.add(Dense(output_dim = 128, activation = 'relu'))
  classifier.add(Dense(output_dim = 2, activation = 'softmax'))
  classifier.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
  print(classifier.summary())
  amf = classifier.fit(X_amf, Y, batch_size=24, epochs=10, validation_split=0.2, shuffle=True, verbose=2)
  amf = amf.history
  acc = amf['accuracy']
  loss = amf['loss']
  text.insert(END,'\nAdaptive Median Filter Accuracy : '+str(acc[9])+"\n")
  text.insert(END,'Adaptive Median Filter Error Rate : '+str(loss[9])+"\n")
  first = cv2.imread("test.jpeg",0)
  second = cv2.imread("clean.jpg",0)



def AMFCNN():
  global cnngb
  global model
  image_org = Image.open("test.jpeg")
  image = np.array(image_org)
  grayscale_image = rgb2gray(image)
  output = adaptivemf(grayscale_image, 3, 11)
  cv2.imwrite("clean.jpg",output)
  classifier = Sequential()
  classifier.add(Convolution2D(32, 3, 3, input_shape = (64, 64, 3), activation = 'relu'))
  classifier.add(MaxPooling2D(pool_size = (2, 2)))
  classifier.add(Convolution2D(32, 3, 3, activation = 'relu'))
  classifier.add(MaxPooling2D(pool_size = (2, 2)))
```

CODE SCREENSHOTS

```
def AMFCNN():
  global cnngb
  global model
  image_org = Image.open("test.jpeg")
  image = np.array(image_org)
  grayscale_image = rgb2gray(image)
  output = adaptivemf(grayscale_image, 3, 11)
  cv2.imwrite("clean.jpg",output)
  classifier = Sequential()
  classifier.add(Convolution2D(32, 3, 3, input_shape = (64, 64, 3), activation = 'relu'))
  classifier.add(MaxPooling2D(pool_size = (2, 2)))
  classifier.add(Convolution2D(32, 3, 3, activation = 'relu'))
  classifier.add(MaxPooling2D(pool_size = (2, 2)))
  classifier.add(Flatten())
  classifier.add(Dense(output_dim = 128, activation = 'relu'))
  classifier.add(Dense(output_dim = 2, activation = 'softmax'))
  classifier.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
  print(classifier.summary())
  cnngb = classifier.fit(X_cnngb, Y, batch_size=8, epochs=10, validation_split=0.2, shuffle=True, verbose=2)
  cnngb = cnngb.history
  acc = cnngb['accuracy']
  loss = cnngb['loss']
  text.insert(END,'\nAMF-CNN-GBML Accuracy : '+str(acc[9])+"\n")
  text.insert(END,'AMF-CNN-GBML Error Rate : '+str(loss[9])+"\n")
  first = cv2.imread("test.jpeg",0)
  second = cv2.imread("clean.jpg",0)
  cnn_data = get_feature_layer(classifier,X_cnngb)#getting features from CNN
  gb = GradientBoostingClassifier()
  gb = gb.fit(cnn_data, Y1) #passing CNN deep features to gradient boosting algorithm for better prediction or classification
  prediction = gb.predict(cnn_data);
  cnn_gb_acc =  accuracy_score(prediction,Y1)
  model = classifier
```

REFERENCES

- https://www.kaggle.com/navoneel/brain-mri-images-for-brain-tumor-detection
- https://www.geeksforgeeks.org/python-image-classification-using-keras/
- https://www.tutorialsteacher.com/python/create-gui-using-tkinter-python
- https://data-flair.training/blogs/brain-tumor-classification-machine-learning/
- https://www.irjet.net/archives/V6/i10/IRJET-V6I10148.pdf
- https://www.cs.auckland.ac.nz/courses/compsci373s1c/PatricesLectures/Gaussian%20Filtering_1up.pdf
- https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/