Automatic Tumor Detection

Author Sayed Reza

University of Texas at El Paso

The project is to identify the which image has the tumor. For this the dataset has some tumor images from which you need to use deep learning to identif tumor.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", forc
```

```
# Author Name: Sayed Reza
# University: University of Texas at El Paso
# Importing necessary Libraries
import os
import keras

# ----------- CODE GOES HERE -----------------
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, BatchNormalization
from PIL import Image
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
#plt.style.use('dark_background')
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
```

Encoding the Target Classes

```
# Identification of Target classes
# 0 - Tumor
# 1 - Normal
# ----------- CODE GOES HERE -----------------
encoder = OneHotEncoder()
encoder.fit([[0], [1]])
```

```
OneHotEncoder()
```

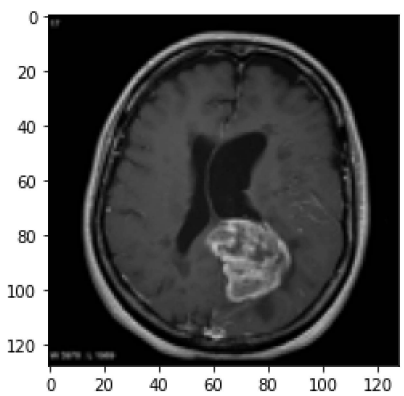
```
# dataset path

dataset_path = "/content/drive/MyDrive/Datasets-for-colab/tumor-detection/";
```

Creating 3 Important Lists --

1. data list for storing image data in numpy array form
2. paths list for storing paths of all images
3. result list for storing one hot encoded form of target class whether normal or tumor

```
# This cell updates result list for images with tumor


data = []
paths = []
result = []

# ----------- CODE GOES HERE -----------------
for r, d, f in os.walk(r'/content/drive/MyDrive/Datasets-for-colab/tumor-detection/yes'):
    for file in f:
        if '.jpg' in file:
            paths.append(os.path.join(r, file))

for path in paths:
    img = Image.open(path)
    img = img.resize((128,128))
    img = np.array(img)
    if(img.shape == (128,128,3)):
        data.append(np.array(img))
        result.append(encoder.transform([[0]]).toarray())

print(len(data))
plt.imshow(data[0])
```

```
70
<matplotlib.image.AxesImage at 0x7f1447839750>
```




```
# This cell updates result list for images without tumor

paths = []

# ----------- CODE GOES HERE -----------------
for r, d, f in os.walk(r"/content/drive/MyDrive/Datasets-for-colab/tumor-detection/no"):
    for file in f:
        if '.jpg' in file:
            paths.append(os.path.join(r, file))

for path in paths:
    img = Image.open(path)
    img = img.resize((128,128))
    img = np.array(img)
    if(img.shape == (128,128,3)):
        data.append(np.array(img))
        result.append(encoder.transform([[1]]).toarray())

print(len(data))
plt.imshow(data[71])
```

```
139
<matplotlib.image.AxesImage at 0x7f144797d0d0>
```

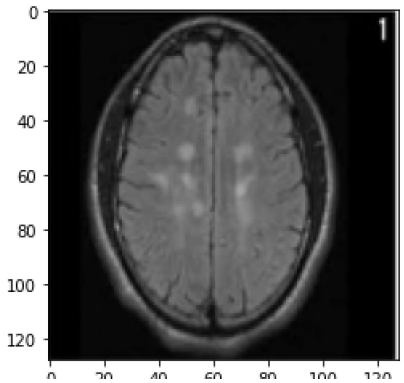


Image Data Preprocessing

```
data = np.array(data)
data.shape
```

```
(139, 128, 128, 3)
```

```
result = np.array(result)
result.shape
```

```
(139, 1, 2)
```

```
result = result.reshape((139,2))
```

Prepare Training & Testing Data

```
x_train,x_test,y_train,y_test = train_test_split(data, result, test_size=0.2, shuffle=True, random_state=0)
```

Deep learning Model Development

```
# We need to use sequential model
model = Sequential()



# ----------- CODE GOES HERE -----------------
model.add(Conv2D(32, kernel_size=(2, 2), input_shape=(128, 128, 3), padding = 'Same'))
model.add(Conv2D(32, kernel_size=(2, 2),  activation ='relu', padding = 'Same'))


model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, kernel_size = (2,2), activation ='relu', padding = 'Same'))
model.add(Conv2D(64, kernel_size = (2,2), activation ='relu', padding = 'Same'))

model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))
```

```
model.add(Dropout(0.25))

model.add(Flatten())

model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(2, activation='softmax'))

model.compile(loss = "categorical_crossentropy", optimizer='Adamax')
print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 128, 128, 32)      416

 conv2d_1 (Conv2D)           (None, 128, 128, 32)      4128

 batch_normalization (BatchN  (None, 128, 128, 32)     128
 ormalization)

 max_pooling2d (MaxPooling2D  (None, 64, 64, 32)       0
 )

 dropout (Dropout)           (None, 64, 64, 32)        0

 conv2d_2 (Conv2D)           (None, 64, 64, 64)        8256

 conv2d_3 (Conv2D)           (None, 64, 64, 64)        16448

 batch_normalization_1 (Batc  (None, 64, 64, 64)       256
 hNormalization)

 max_pooling2d_1 (MaxPooling  (None, 32, 32, 64)       0
 2D)

 dropout_1 (Dropout)         (None, 32, 32, 64)        0

 flatten (Flatten)           (None, 65536)             0

 dense (Dense)               (None, 512)               33554944

 dropout_2 (Dropout)         (None, 512)               0

 dense_1 (Dense)             (None, 2)                 1026

=================================================================
Total params: 33,585,602
Trainable params: 33,585,410
Non-trainable params: 192
_________________________________________________________________
None
```

```
y_train.shape
```

```
(111, 2)
```

Deep Learning Model Run

```
# ----------- CODE GOES HERE -----------------

history = model.fit(x_train, y_train, epochs = 30, batch_size = 40, verbose = 1,validation_data = (x_test, y_test))
```

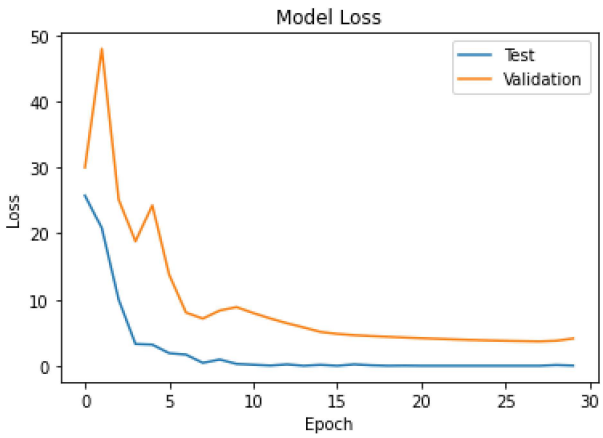
```
Epoch 2/30
```

```
Epoch 2/30
3/3 [==============================] - 9s 3s/step - loss: 20.8294 - val_loss: 47.9428
Epoch 3/30
3/3 [==============================] - 9s 3s/step - loss: 10.0198 - val_loss: 25.1805
Epoch 4/30
3/3 [==============================] - 12s 4s/step - loss: 3.2950 - val_loss: 18.8018
Epoch 5/30
3/3 [==============================] - 8s 3s/step - loss: 3.1986 - val_loss: 24.2800
Epoch 6/30
3/3 [==============================] - 8s 2s/step - loss: 1.9051 - val_loss: 13.7638
Epoch 7/30
3/3 [==============================] - 8s 3s/step - loss: 1.6860 - val_loss: 8.0411
Epoch 8/30
3/3 [==============================] - 8s 3s/step - loss: 0.4423 - val_loss: 7.1274
Epoch 9/30
3/3 [==============================] - 8s 3s/step - loss: 0.9473 - val_loss: 8.3542
Epoch 10/30
3/3 [==============================] - 8s 3s/step - loss: 0.2907 - val_loss: 8.8689
Epoch 11/30
3/3 [==============================] - 10s 3s/step - loss: 0.1648 - val_loss: 7.9640
Epoch 12/30
3/3 [==============================] - 7s 2s/step - loss: 0.0362 - val_loss: 7.1577
Epoch 13/30
3/3 [==============================] - 7s 2s/step - loss: 0.2187 - val_loss: 6.4230
Epoch 14/30
3/3 [==============================] - 7s 2s/step - loss: 3.1571e-05 - val_loss: 5.7972
Epoch 15/30
3/3 [==============================] - 7s 2s/step - loss: 0.1487 - val_loss: 5.1218
Epoch 16/30
3/3 [==============================] - 7s 2s/step - loss: 7.7803e-04 - val_loss: 4.8156
Epoch 17/30
3/3 [==============================] - 8s 3s/step - loss: 0.2224 - val_loss: 4.6312
Epoch 18/30
3/3 [==============================] - 8s 3s/step - loss: 0.0801 - val_loss: 4.4940
Epoch 19/30
3/3 [==============================] - 8s 2s/step - loss: 0.0026 - val_loss: 4.3837
Epoch 20/30
3/3 [==============================] - 8s 2s/step - loss: 0.0260 - val_loss: 4.2687
Epoch 21/30
3/3 [==============================] - 7s 2s/step - loss: 8.2522e-04 - val_loss: 4.1647
Epoch 22/30
3/3 [==============================] - 7s 2s/step - loss: 3.6199e-06 - val_loss: 4.0680
Epoch 23/30
3/3 [==============================] - 7s 2s/step - loss: 1.3904e-04 - val_loss: 3.9814
Epoch 24/30
3/3 [==============================] - 7s 2s/step - loss: 1.7613e-07 - val_loss: 3.9052
Epoch 25/30
3/3 [==============================] - 7s 2s/step - loss: 0.0010 - val_loss: 3.8350
Epoch 26/30
3/3 [==============================] - 7s 2s/step - loss: 6.9200e-06 - val_loss: 3.7731
Epoch 27/30
3/3 [==============================] - 7s 2s/step - loss: 1.4004e-06 - val_loss: 3.7239
Epoch 28/30
3/3 [==============================] - 7s 2s/step - loss: 6.4486e-05 - val_loss: 3.6787
Epoch 29/30
3/3 [==============================] - 7s 2s/step - loss: 0.1193 - val_loss: 3.7844
Epoch 30/30
3/3 [==============================] - 7s 2s/step - loss: 0.0299 - val_loss: 4.1195
```

Plotting Losses

```
# Plot the model lossess

# ----------- CODE GOES HERE -----------------

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

```
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Test', 'Validation'], loc='upper right')
plt.show()
```

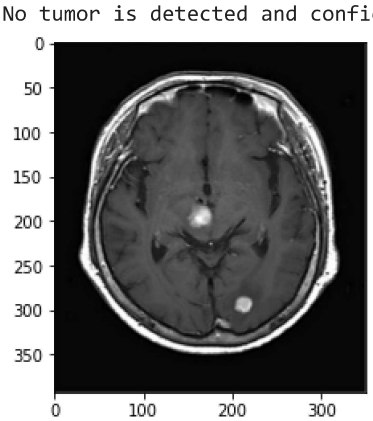


Realtime automatic detection of Tumor

```
def names(number):
    if number==0:
        return 'Tumor is automatically detected'
    else:
        return 'No tumor is detected'
```

```
from matplotlib.pyplot import imshow
img = Image.open(r"/content/drive/MyDrive/Datasets-for-colab/tumor-detection/no/N17.jpg")
x = np.array(img.resize((128,128)))
x = x.reshape(1,128,128,3)
res = model.predict_on_batch(x)
classification = np.where(res == np.amax(res))[1][0]
imshow(img)
print( names(classification) + ' and confidence is '+str(res[0][classification]*100)+'%')
```

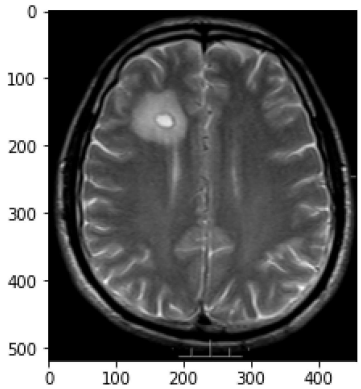
```
No tumor is detected and confidence is 100.0%
```




```
from matplotlib.pyplot import imshow
img = Image.open(r"/content/drive/MyDrive/Datasets-for-colab/tumor-detection/yes/Y3.jpg")
x = np.array(img.resize((128,128)))
x = x.reshape(1,128,128,3)
res = model.predict_on_batch(x)
```

```
classification = np.where(res == np.amax(res))[1][0]
imshow(img)
print( names(classification) + ' and confidence is '+str(res[0][classification]*100)+'%')
```

```
Tumor is automatically detected and confidence is 99.98321533203125%
```



✓ 0s completed at 11:34 PM