```python
import os
import keras
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
```

```python
encoder = OneHotEncoder()
encoder.fit([[0], [1]])

# 0 - Tumor
# 1 - Normal
```

```
OneHotEncoder
OneHotEncoder()
```

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

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

for r, d, f in os.walk(r'../content/yes'):
    for file in f:
        if '.jpg' in file:
            paths.append(os.path.join(r, file))
```

```python
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())
```

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

paths = []
for r, d, f in os.walk(r"../content/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())
```

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

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

```python
result = np.array(result)
result = result.reshape(139,2)
```

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

```python
model = Sequential()

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_1"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 128, 128, 32)	416
conv2d_1 (Conv2D)	(None, 128, 128, 32)	4,128
batch_normalization (BatchNormalization)	(None, 128, 128, 32)	128
max_pooling2d (MaxPooling2D)	(None, 64, 64, 32)	0
dropout (Dropout)	(None, 64, 64, 32)	0
conv2d_2 (Conv2D)	(None, 64, 64, 64)	8,256
conv2d_3 (Conv2D)	(None, 64, 64, 64)	16,448
batch_normalization_1 (BatchNormalization)	(None, 64, 64, 64)	256
max_pooling2d_1 (MaxPooling2D)	(None, 32, 32, 64)	0
dropout_1 (Dropout)	(None, 32, 32, 64)	0
flatten (Flatten)	(None, 65536)	0
dense (Dense)	(None, 512)	33,554,944
dropout_2 (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 2)	1,026

Total params: 33,585,602 (128.12 MB)
Trainable params: 33,585,410 (128.12 MB)
Non-trainable params: 192 (768.00 B)
None

```
y_train.shape
```

(111, 2)

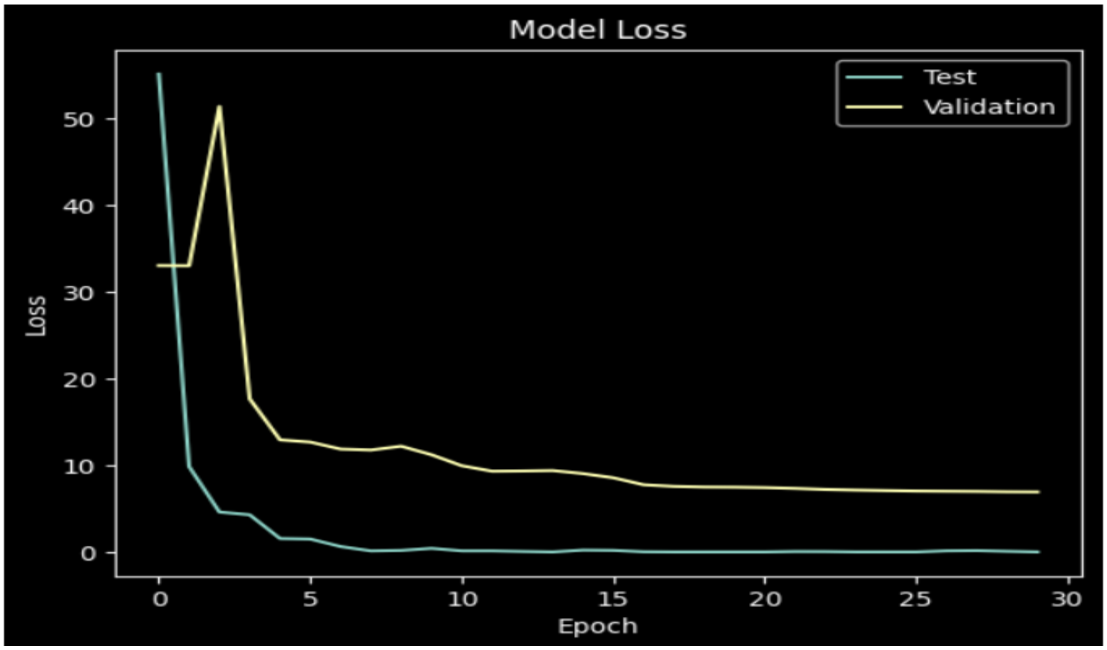
```
history = model.fit(x_train, y_train, epochs = 30, batch_size = 40, verbose
= 1,validation_data = (x_test, y_test))
```

```
Epoch 1/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 14s 4s/step - loss: 43.9371 - val_loss: 33.0558
Epoch 2/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 19s 3s/step - loss: 11.1397 - val_loss: 33.0328
Epoch 3/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 12s 3s/step - loss: 3.8994 - val_loss: 51.4155
Epoch 4/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 21s 4s/step - loss: 4.7124 - val_loss: 17.6670
Epoch 5/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 9s 3s/step - loss: 1.1518 - val_loss: 12.9371
Epoch 6/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 19s 7s/step - loss: 1.8276 - val_loss: 12.6790
Epoch 7/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 15s 4s/step - loss: 0.6626 - val_loss: 11.8728
Epoch 8/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 20s 3s/step - loss: 0.1005 - val_loss: 11.7623
Epoch 9/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 20s 4s/step - loss: 0.2055 - val_loss: 12.2009
Epoch 10/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 9s 3s/step - loss: 0.3978 - val_loss: 11.2193
Epoch 11/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 12s 3s/step - loss: 0.1762 - val_loss: 9.9467
Epoch 12/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 20s 4s/step - loss: 0.1807 - val_loss: 9.3177
Epoch 13/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 21s 3s/step - loss: 0.0878 - val_loss: 9.3459
Epoch 14/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 20s 4s/step - loss: 6.0032e-04 - val_loss: 9.3933
Epoch 15/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 9s 3s/step - loss: 0.2113 - val_loss: 9.0411
Epoch 16/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 11s 3s/step - loss: 0.2054 - val_loss: 8.5610
Epoch 17/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 11s 4s/step - loss: 0.0331 - val_loss: 7.7541
```

3/3 ━━━━━━━━━━━━━━━━━━━━ 11s 4s/step - loss: 0.0331 - val_loss: 7.7541

```
Epoch 18/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 19s 3s/step - loss: 3.6638e-04 - val_loss: 7.5669
Epoch 19/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 11s 3s/step - loss: 7.9314e-06 - val_loss: 7.4931
Epoch 20/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 11s 3s/step - loss: 0.0013 - val_loss: 7.4764
Epoch 21/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 20s 4s/step - loss: 2.5508e-04 - val_loss: 7.4257
Epoch 22/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 9s 3s/step - loss: 0.0791 - val_loss: 7.3326
Epoch 23/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 10s 3s/step - loss: 0.0667 - val_loss: 7.2127
Epoch 24/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 20s 4s/step - loss: 0.0034 - val_loss: 7.1384
Epoch 25/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 9s 3s/step - loss: 8.3413e-04 - val_loss: 7.0804
Epoch 26/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 12s 3s/step - loss: 0.0014 - val_loss: 7.0307
Epoch 27/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 11s 4s/step - loss: 0.1105 - val_loss: 7.0056
Epoch 28/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 19s 3s/step - loss: 0.0764 - val_loss: 6.9848
Epoch 29/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 12s 3s/step - loss: 0.0633 - val_loss: 6.9436
Epoch 30/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 21s 4s/step - loss: 0.0015 - val_loss: 6.9264
```

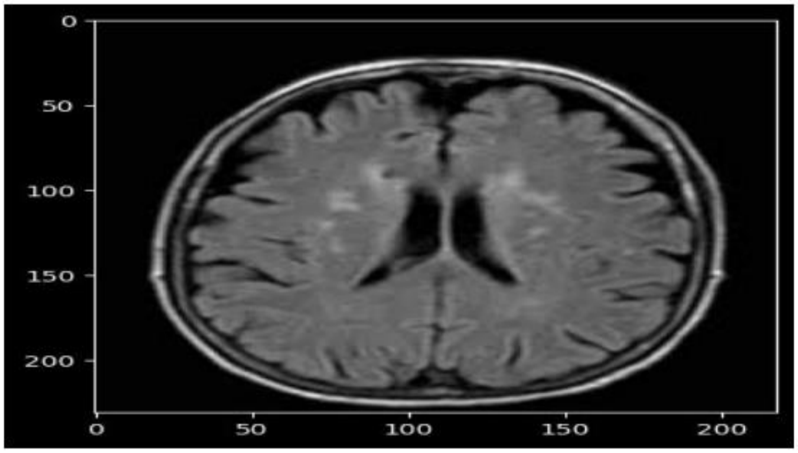
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Test', 'Validation'], loc='upper right')
plt.show()
```




```
def names(number):
    if number==0:
        return 'Its a Tumor'
    else:
        return 'No, Its not a tumor'
```

```
from matplotlib.pyplot import imshow
img = Image.open(r"../content/no/17 no.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(str(res[0][classification]*100) + '% Confidence This Is ' +
names(classification))
```

```
99.98847246170044% Confidence This Is No, Its not a tumor
```




```python
from matplotlib.pyplot import imshow
img = Image.open(r"../content/yes/Y117.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(str(res[0][classification]*100) + '% Confidence This Is A ' +
names(classification))
```

```
100.0% Confidence This Is A Its a Tumor
```