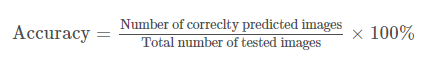
**AI MINI PROJECT :**

# OVERVIEW AND OBJECTIVES :

The main purpose of this project was to build a CNN model that would classify if subject has a haemorrhage or not base on MRI scan. I used the [VGG-16](https://www.kaggle.com/navoneel/brain-mri-images-for-brain-tumor-detection) model architecture and weights to train the model for this binary problem. I used accuracy as a metric to justify the model performance which can be defined as:



Final results look as follows:

|  |  |
| --- | --- |
| Set | Accuracy |
| Validation Set\* | ~88% |
| Test Set\* | ~80% |

* validation set - is the set used during the model training to adjust the hyperparameters.
* test set - is the small set that I don't touch for the whole training process at all. It's been used for final model performance evaluation.

## 

## 

## DATA SET DESCRIPTION :

## The image data that was used for this problem is [Brain MRI Images for Brain haemorrhage Detection](https://www.kaggle.com/navoneel/brain-mri-images-for-brain-tumor-detection). It conists of MRI scans of two classes:

* NO - no haemorrhage, encoded as 0
* YES - haemorrhage, encoded as 1

**PROGRAM :**

Import pandas as pd

import numpy as np

import cv2

import os

import matplotlib.pyplot as plt

import seaborn as sns

import tensorflow as tf

os.environ['TF\_CPP\_MIN\_LOG\_LEVEL'] = '3'

from keras.models import Sequential

from keras.layers import Dense, Flatten, Conv2D, MaxPooling2D

from keras.layers import Input, Activation, LeakyReLU, Dropout

from keras.losses import BinaryCrossentropy

from tensorflow.keras.optimizers import Adam

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import classification\_report, confusion\_matrix

import warnings

warnings.filterwarnings('ignore')

MAIN\_DIR = "../input/brain-mri-images-for-brain-haemorrhage-detection"

SEED = 40

## 

## PREPARING THE IMAGE DATA

os.listdir(MAIN\_DIR)

**OUTPUT :**

['no', 'yes', 'brain\_haemorrhage\_dataset']

subdirs = os.listdir(MAIN\_DIR)[:2]

for subdir in subdirs:

print(f"{subdir} contains {len(os.listdir(MAIN\_DIR+'/'+subdir))} images")

**OUTPUT :**

no contains 98 images

yes contains 155 images

def load\_images(folder):

imgs = []

target = 0

labels = []

for i in os.listdir(folder):

subdir = os.path.join(folder, i)

for j in os.listdir(subdir):

img\_dir = os.path.join(subdir,j)

try:

img = cv2.imread(img\_dir)

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

img = cv2.resize(img, (128,128))

imgs.append(img)

labels.append(target)

except:

continue

target += 1

imgs = np.array(imgs)

labels = np.array(labels)

return imgs, labels

data, labels = load\_images(MAIN\_DIR)

data.shape, labels.shape

**OUTPUT:**

((253, 128, 128), (253,))

**Visualisation Data**

def plot\_images(start, end):

plt.figure(figsize=(22,8))

for i in range(10):

axs = plt.subplot(2,5, i+1)

idx = np.random.randint(start, end)

plt.imshow(data[idx], cmap='gray')

plt.axis('on')

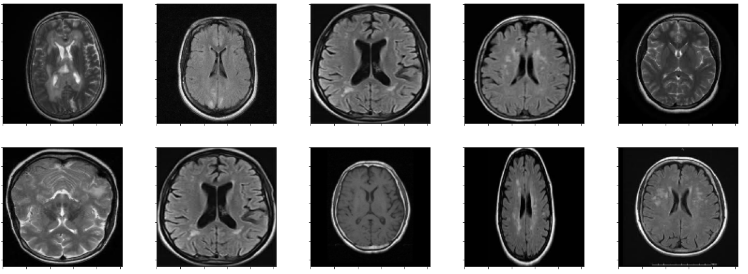
axs.set\_xticklabels([])

axs.set\_yticklabels([])

plt.subplots\_adjust(wspace=None, hspace=None)

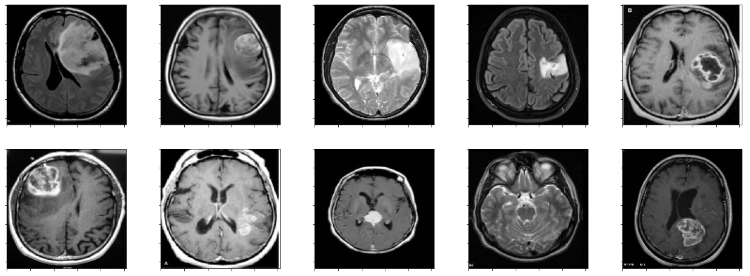
plot\_images(0, 97)

**OUTPUT :**

****

**OBSERVATION :** 0 to 97 for images with no haemorrhage

plot\_images(98, 252)

****

**OBSERVATION :** 98 to 252 for images with haemorrhage

norm\_data = data / 255.

norm\_data = np.expand\_dims(norm\_data, axis=3)

norm\_data.shape, norm\_data[0]

**OUTPUT :**

((253, 128, 128, 1),

array([[[0.01568627],

[0.01568627],

[0.01568627],

...,

[0.01176471],

[0.01176471],

[0.01176471]],

[[0.01568627],

[0.01568627],

[0.01568627],

...,

[0.01568627],

[0.01568627],

[0.01568627]],

[[0.01568627],

[0.01568627],

[0.01568627],

...,

[0.01568627],

[0.01568627],

[0.01568627]],

...,

[[0.01176471],

[0.01176471],

[0.01176471],

...,

[0.01176471],

[0.01176471],

[0.01176471]],

[[0.01176471],

[0.01176471],

[0.01176471],

...,

[0.01176471],

[0.01176471],

[0.01176471]],

[[0.01176471],

[0.01176471],

[0.01176471],

...,

[0.01176471],

[0.01176471],

[0.01176471]]]))

## Convolutional Neural Network

tf.random.set\_seed(SEED)

model = tf.keras.models.Sequential([

tf.keras.layers.Conv2D(filters=64,

kernel\_size=3,

activation='relu',

input\_shape=(128,128,1)),

tf.keras.layers.Conv2D(32,3,activation='relu'),

tf.keras.layers.MaxPool2D(pool\_size=2,

padding='valid'),

tf.keras.layers.Conv2D(32,3,activation='relu'),

tf.keras.layers.Conv2D(16,3,activation='relu'),

tf.keras.layers.MaxPool2D(2),

tf.keras.layers.Flatten(),

tf.keras.layers.Dense(1, activation='softmax')

])

# Compile the model

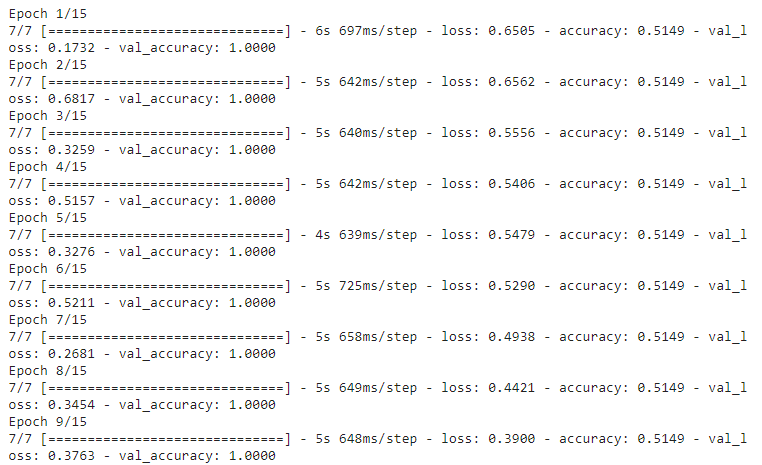
model.compile(loss=tf.keras.losses.BinaryCrossentropy(),

optimizer=tf.keras.optimizers.Adam(),

metrics=["accuracy"])

# Fit the model

history = model.fit(norm\_data, labels, epochs=15, validation\_split=0.20)



## EVALUATION

result = model.evaluate(norm\_data, labels, verbose=0)

print(f"Accuracy on Evaluation: {result[1]\*100:.2f}%\nLoss: {result[0]:.4f}")

**OUTPUT :**

Accuracy on Evaluation: 61.26%

Loss: 0.1316

np.random.seed(SEED)

idxs = np.random.randint(0, 252, 20)

y\_pred\_prob = model.predict(norm\_data[idxs])

y\_pred = np.array([1 if prob>0.5 else 0 for prob in y\_pred\_prob])

y\_true = labels[idxs]

y\_pred.shape, y\_true.shape

**OUTPUT :**

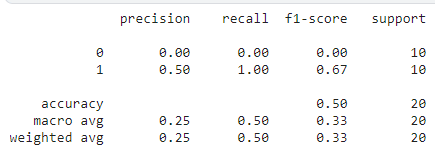
((20,), (20,))

y\_pred

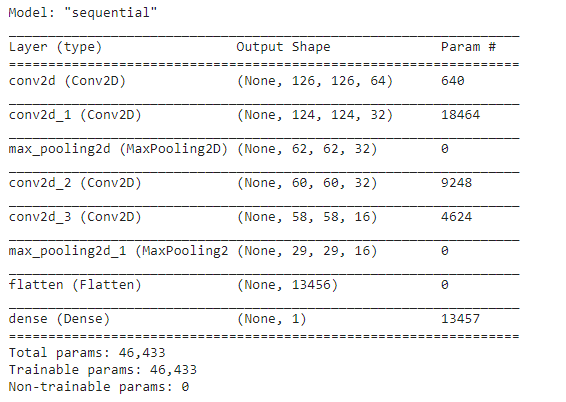
**OUTPUT :**

array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])

print(classification\_report(y\_true, y\_pred))



model.summary()



histdf = pd.DataFrame(history.history)

plt.figure(figsize=(12,8))

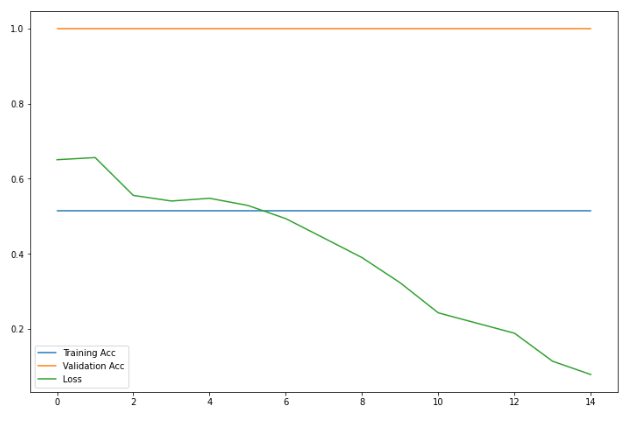
plt.plot(histdf['accuracy'], label='Training Acc')

plt.plot(histdf['val\_accuracy'], label='Validation Acc')

plt.plot(histdf['loss'], label='Loss')

plt.legend()

plt.show()



plt.figure(figsize=(15,6))

plt.subplot(141)

plt.imshow(img)

plt.xticks([])

plt.yticks([])

plt.title('Step 1. Get the original image')

plt.subplot(142)

plt.imshow(img\_cnt)

plt.xticks([])

plt.yticks([])

plt.title('Step 2. Find the biggest contour')

plt.subplot(143)

plt.imshow(img\_pnt)

plt.xticks([])

plt.yticks([])

plt.title('Step 3. Find the extreme points')

plt.subplot(144)

plt.imshow(new\_img)

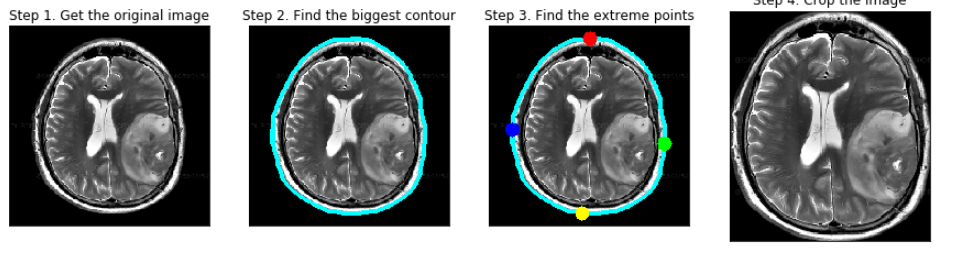
plt.xticks([])

plt.yticks([])

plt.title('Step 4. Crop the image')

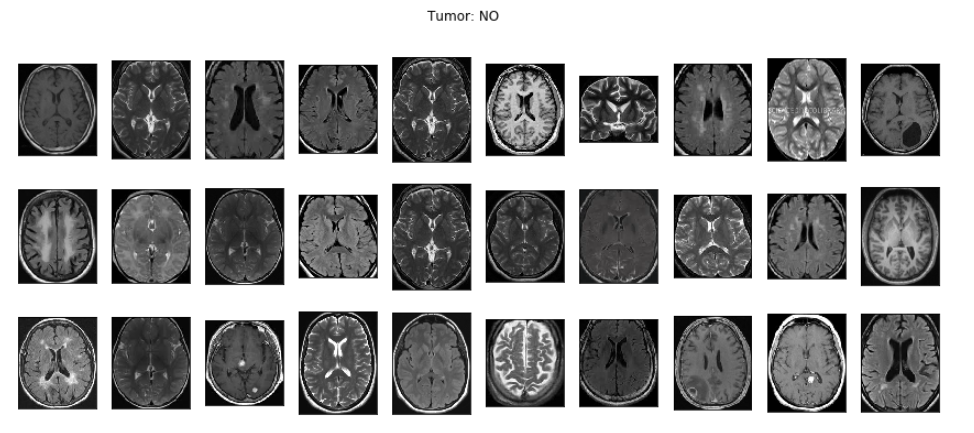
plt.show()

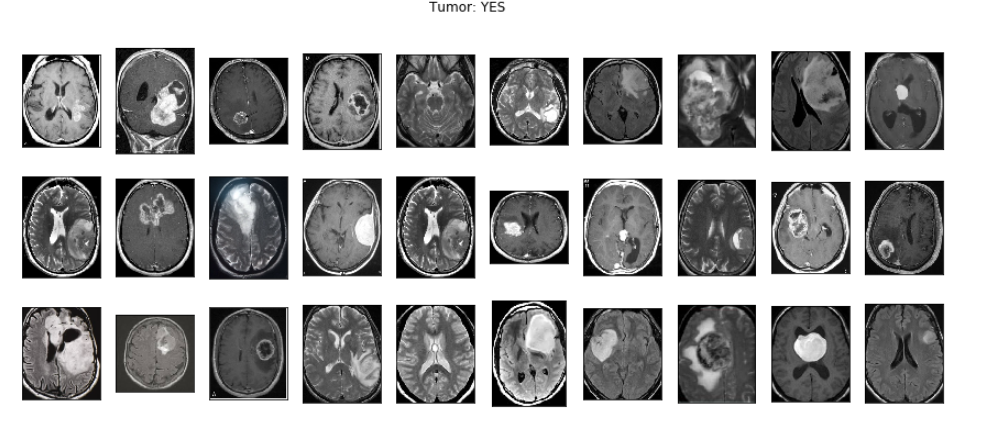
**OUTPUT :**



plot\_samples(X\_train\_crop, y\_train, labels, 30)

**OUTPUT :**





predictions = model.predict(X\_val\_prep)

predictions = [1 if x>0.5 else 0 for x **in** predictions]

accuracy = accuracy\_score(y\_val, predictions)

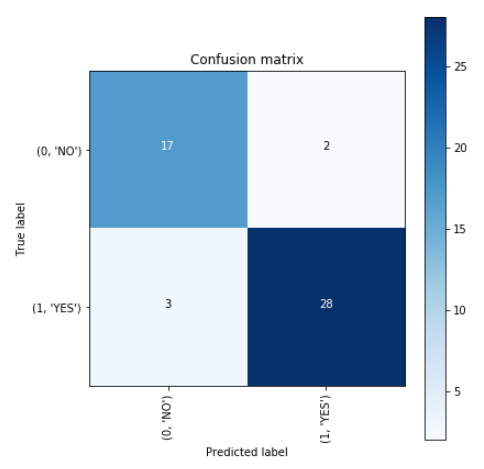
print('Val Accuracy = **%.2f**' % accuracy)

confusion\_mtx = confusion\_matrix(y\_val, predictions)

cm = plot\_confusion\_matrix(confusion\_mtx, classes = list(labels.items()), normalize=False)

Val Accuracy = 0.90

**OUTPUT :**



# 

# 

# CONCLUSIONS :

This project was a combination of CNN model classification problem (to predict wheter the subject has brain haemorrhage or not) & Computer Vision problem (to automate the process of brain cropping from MRI scans). The final accuracy is much higher than 50% baseline (random guess). However, it could be increased by larger number of train images or through model hyperparameters tuning.