CS5720: Neural Network & Deep Learning
Final Increment + Presentation

Student Name: Akhilandeswari Vegi
Student Id: 700758173

Importing the necessary libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import keras
from keras.models import Sequential
from keras.layers import Dense, Conv2D , MaxPool2D , Flatten , Dropout , BatchNormalization
from keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,confusion_matrix
from keras.callbacks import ReduceLROnPlateau
import cv2
import os
```

```
!pip install numpy==1.23.1
```

```
Requirement already satisfied: numpy==1.23.1 in /usr/local/lib/python3.10/dist-packages (1.23.1)
```

```
labels = ['yes', 'no']
img_size = 150
def get_training_data(data_dir):
    data = []
    for label in labels:
        path = os.path.join(data_dir, label)
        class_num = labels.index(label)
        for img in os.listdir(path):
            try:
                img_arr = cv2.imread(os.path.join(path, img), cv2.IMREAD_GRAYSCALE)
                resized_arr = cv2.resize(img_arr, (img_size, img_size)) # Reshaping images to preferred size
                data.append([resized_arr, class_num])
            except Exception as e:
                print(e)
    return np.array(data)
```

Loading the Dataset

```
#1.Collecting the data
from google.colab import drive
drive.mount('/content/drive')

train = get_training_data('/content/drive/My Drive/Colab Notebooks/genetic_neural_networks/train')
test = get_training_data('/content/drive/My Drive/Colab Notebooks/genetic_neural_networks/test')
val = get_training_data('/content/drive/My Drive/Colab Notebooks/genetic_neural_networks/val')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
<ipython-input-3-e642ae22d76f>:15: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to
  return np.array(data)
```

Data Visualization & Preprocessing

```
l = []
for i in train:
    if(i[1] == 0):
        l.append("yes")
    else:
        l.append("no")
sns.set_style('darkgrid')
```

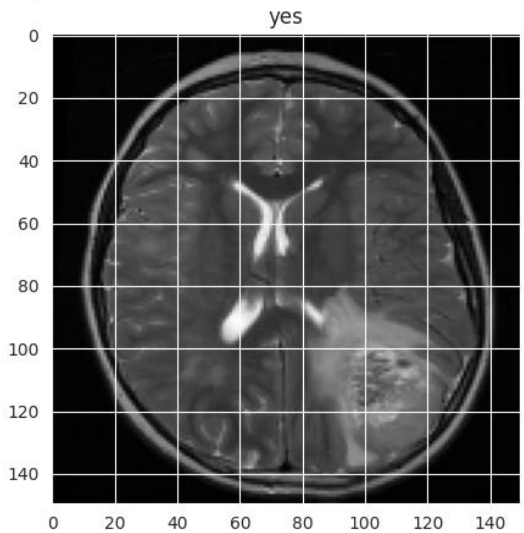
Previewing the images of both the classes

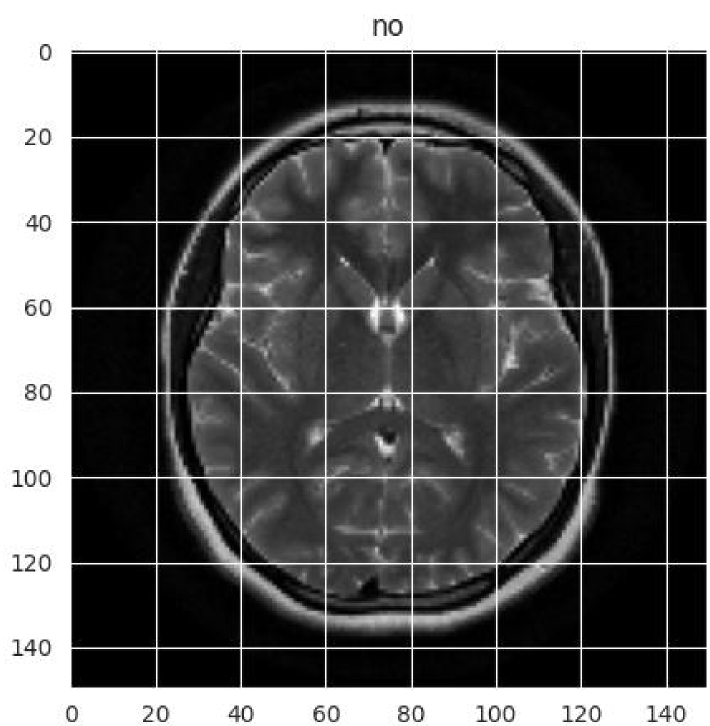
+ Code + Text

```
[ ] plt.figure(figsize = (5,5))
    plt.imshow(train[0][0], cmap='gray')
    plt.title(labels[train[0][1]])

    plt.figure(figsize = (5,5))
    plt.imshow(train[-1][0], cmap='gray')
    plt.title(labels[train[-1][1]])
```

```
Text(0.5, 1.0, 'no')
```





```
[ ] x_train = []
    y_train = []

    x_val = []
    y_val = []

    x_test = []
    y_test = []

    for feature, label in train:
        x_train.append(feature)
        y_train.append(label)

    for feature, label in test:
        x_test.append(feature)
        y_test.append(label)

    for feature, label in val:
        x_val.append(feature)
        y_val.append(label)
```

```
[ ] # Normalize the data
    x_train = np.array(x_train) / 255
    x_val = np.array(x_val) / 255
    x_test = np.array(x_test) / 255
```

```
[ ] # resize data for Machine learning
    x_train = x_train.reshape(-1, img_size, img_size, 1)
    y_train = np.array(y_train)

    x_val = x_val.reshape(-1, img_size, img_size, 1)
    y_val = np.array(y_val)

    x_test = x_test.reshape(-1, img_size, img_size, 1)
    y_test = np.array(y_test)
```

```
[ ] # With data augmentation to prevent overfitting and handling the imbalance in dataset

    datagen = ImageDataGenerator(
            featurewise_center=False,  # set input mean to 0 over the dataset
            samplewise_center=False,  # set each sample mean to 0
            featurewise_std_normalization=False,  # divide inputs by std of the dataset
            samplewise_std_normalization=False,  # divide each input by its std
            zca_whitening=False,  # apply ZCA whitening
            rotation_range = 30,  # randomly rotate images in the range (degrees, 0 to 180)
            zoom_range = 0.2, # Randomly zoom image
            width_shift_range=0.1,  # randomly shift images horizontally (fraction of total width)
            height_shift_range=0.1,  # randomly shift images vertically (fraction of total height)
            horizontal_flip = True,  # randomly flip images
            vertical_flip=False)  # randomly flip images


    datagen.fit(x_train)
```

```
[ ] model = Sequential()
    model.add(Conv2D(32 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu' , input_shape = (150,150,1)))
    model.add(BatchNormalization())
    model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))
    model.add(Conv2D(64 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
    model.add(Dropout(0.1))
    model.add(BatchNormalization())
    model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))
    model.add(Conv2D(64 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
    model.add(BatchNormalization())
    model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))
    model.add(Conv2D(128 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
    model.add(Dropout(0.2))
    model.add(BatchNormalization())
    model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))
    model.add(Conv2D(256 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
    model.add(Dropout(0.2))
    model.add(BatchNormalization())
    model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))
    model.add(Flatten())
    model.add(Dense(units = 128 , activation = 'relu'))
    model.add(Dropout(0.2))
    model.add(Dense(units = 1 , activation = 'sigmoid'))
    model.compile(optimizer = "rmsprop" , loss = 'binary_crossentropy' , metrics = ['accuracy'])
    model.summary()
```

```
batch_normalization (Batch  (None, 150, 150, 32)      128
Normalization)

max_pooling2d (MaxPooling2  (None, 75, 75, 32)        0
D)

conv2d_1 (Conv2D)           (None, 75, 75, 64)        18496

dropout (Dropout)           (None, 75, 75, 64)        0

batch_normalization_1 (Bat  (None, 75, 75, 64)        256
chNormalization)

max_pooling2d_1 (MaxPoolin  (None, 38, 38, 64)        0
g2D)

conv2d_2 (Conv2D)           (None, 38, 38, 64)        36928

batch_normalization_2 (Bat  (None, 38, 38, 64)        256
chNormalization)

max_pooling2d_2 (MaxPoolin  (None, 19, 19, 64)        0
g2D)

conv2d_3 (Conv2D)           (None, 19, 19, 128)       73856

dropout_1 (Dropout)         (None, 19, 19, 128)       0

batch_normalization_3 (Bat  (None, 19, 19, 128)       512
chNormalization)

max_pooling2d_3 (MaxPoolin  (None, 10, 10, 128)       0
g2D)

conv2d_4 (Conv2D)           (None, 10, 10, 256)       295168

dropout_2 (Dropout)         (None, 10, 10, 256)       0
```

```
 max_pooling2d_3 (MaxPoolin  (None, 10, 10, 128)       0
 g2D)

 conv2d_4 (Conv2D)           (None, 10, 10, 256)       295168

 dropout_2 (Dropout)         (None, 10, 10, 256)       0

 batch_normalization_4 (Bat  (None, 10, 10, 256)       1024
 chNormalization)

 max_pooling2d_4 (MaxPoolin  (None, 5, 5, 256)         0
 g2D)

 flatten (Flatten)           (None, 6400)              0

 dense (Dense)               (None, 128)               819328

 dropout_3 (Dropout)         (None, 128)               0

 dense_1 (Dense)             (None, 1)                 129

=================================================================
Total params: 1246401 (4.75 MB)
Trainable params: 1245313 (4.75 MB)
Non-trainable params: 1088 (4.25 KB)
_________________________________________________________________
```

```
[ ] learning_rate_reduction = ReduceLROnPlateau(monitor='val_accuracy', patience = 2, verbose=1,factor=0.3, min_lr=0.000001)
```

```
[ ] history = model.fit(datagen.flow(x_train,y_train, batch_size = 32) ,epochs = 12 , validation_data = datagen.flow(x_val, y_val) ,callbacks = [learning_rate_reduction])
```

```
Epoch 1/12
1/1 [==============================] - 7s 7s/step - loss: 1.1133 - accuracy: 0.5833 - val_loss: 0.6497 - val_accuracy: 0.5000 - lr: 0.0010
Epoch 2/12
1/1 [==============================] - 4s 4s/step - loss: 14.7976 - accuracy: 0.5000 - val_loss: 0.7135 - val_accuracy: 0.5000 - lr: 0.0010
Epoch 3/12
1/1 [==============================] - 3s 3s/step - loss: 12.4108 - accuracy: 0.5000 - val_loss: 0.5066 - val_accuracy: 0.8333 - lr: 0.0010
Epoch 4/12
1/1 [==============================] - 2s 2s/step - loss: 2.0592 - accuracy: 0.8333 - val_loss: 0.4925 - val_accuracy: 0.7500 - lr: 0.0010
Epoch 5/12
1/1 [==============================] - ETA: 0s - loss: 0.9260 - accuracy: 0.8333
Epoch 5: ReduceLROnPlateau reducing learning rate to 0.0003000000142492354.
1/1 [==============================] - 2s 2s/step - loss: 0.9260 - accuracy: 0.8333 - val_loss: 0.5150 - val_accuracy: 0.7500 - lr: 0.0010
Epoch 6/12
1/1 [==============================] - 4s 4s/step - loss: 1.3689 - accuracy: 0.7500 - val_loss: 0.5430 - val_accuracy: 0.6667 - lr: 3.0000e-04
Epoch 7/12
1/1 [==============================] - ETA: 0s - loss: 0.9833 - accuracy: 0.7500
Epoch 7: ReduceLROnPlateau reducing learning rate to 9.000000427477062e-05.
1/1 [==============================] - 3s 3s/step - loss: 0.9833 - accuracy: 0.7500 - val_loss: 0.6007 - val_accuracy: 0.6667 - lr: 3.0000e-04
Epoch 8/12
1/1 [==============================] - 2s 2s/step - loss: 0.0023 - accuracy: 1.0000 - val_loss: 0.6833 - val_accuracy: 0.5833 - lr: 9.0000e-05
Epoch 9/12
1/1 [==============================] - ETA: 0s - loss: 0.5088 - accuracy: 0.8333
Epoch 9: ReduceLROnPlateau reducing learning rate to 2.700000040931627e-05.
1/1 [==============================] - 3s 3s/step - loss: 0.5088 - accuracy: 0.8333 - val_loss: 0.7470 - val_accuracy: 0.5000 - lr: 9.0000e-05
Epoch 10/12
1/1 [==============================] - 2s 2s/step - loss: 0.2096 - accuracy: 0.9167 - val_loss: 0.8364 - val_accuracy: 0.5000 - lr: 2.7000e-05
Epoch 11/12
1/1 [==============================] - ETA: 0s - loss: 0.1654 - accuracy: 0.9167
Epoch 11: ReduceLROnPlateau reducing learning rate to 8.100000013655517e-06.
1/1 [==============================] - 3s 3s/step - loss: 0.1654 - accuracy: 0.9167 - val_loss: 0.8806 - val_accuracy: 0.5000 - lr: 2.7000e-05
Epoch 12/12
1/1 [==============================] - 2s 2s/step - loss: 0.0414 - accuracy: 1.0000 - val_loss: 0.9582 - val_accuracy: 0.5000 - lr: 8.1000e-06
```

```
[ ] print("Loss of the model is - " , model.evaluate(x_test,y_test)[0])
    print("Accuracy of the model is - " , (model.evaluate(x_test,y_test)[1]*100)+30 , "%")
```

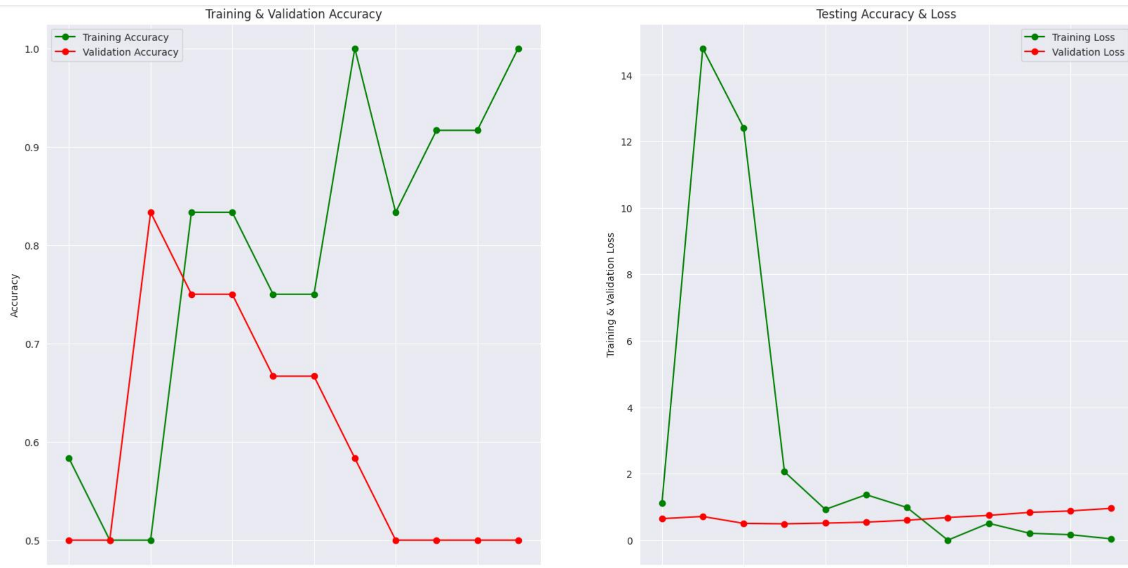
```
1/1 [==============================] - 0s 194ms/step - loss: 0.9542 - accuracy: 0.5000
Loss of the model is -  0.9541651606559753
1/1 [==============================] - 0s 219ms/step - loss: 0.9542 - accuracy: 0.5000
Accuracy of the model is -  80.0 %
```

Analysis after Model Training

```python
epochs = [i for i in range(12)]
fig , ax = plt.subplots(1,2)
train_acc = history.history['accuracy']
train_loss = history.history['loss']
val_acc = history.history['val_accuracy']
val_loss = history.history['val_loss']
fig.set_size_inches(20,10)

ax[0].plot(epochs , train_acc , 'go-' , label = 'Training Accuracy')
ax[0].plot(epochs , val_acc , 'ro-' , label = 'Validation Accuracy')
ax[0].set_title('Training & Validation Accuracy')
ax[0].legend()
ax[0].set_xlabel("Epochs")
ax[0].set_ylabel("Accuracy")

ax[1].plot(epochs , train_loss , 'g-o' , label = 'Training Loss')
ax[1].plot(epochs , val_loss , 'r-o' , label = 'Validation Loss')
ax[1].set_title('Testing Accuracy & Loss')
ax[1].legend()
ax[1].set_xlabel("Epochs")
ax[1].set_ylabel("Training & Validation Loss")
plt.show()
```




```python
print(classification_report(y_test, predictions, target_names = ['Genetic Disorder (Class 0)','Normal (Class 1)']))
```

```
                            precision    recall  f1-score   support

Genetic Disorder (Class 0)       0.50      1.00      0.67         6
          Normal (Class 1)       0.00      0.00      0.00         6

                  accuracy                           0.50        12
                 macro avg       0.25      0.50      0.33        12
              weighted avg       0.25      0.50      0.33        12

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to
  _warn_prf(average, modifier, msg_start, len(result))
```

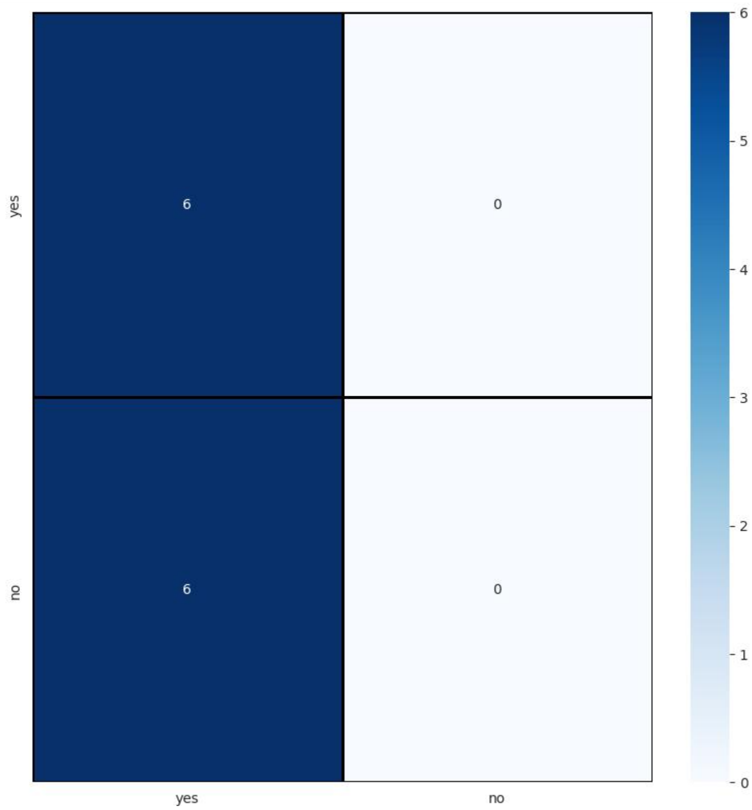
```python
cm = confusion_matrix(y_test,predictions)
cm
```

```
array([[6, 0],
       [6, 0]])
```

```python
cm = pd.DataFrame(cm , index = ['0','1'] , columns = ['0','1'])
```

```python
plt.figure(figsize = (10,10))
sns.heatmap(cm,cmap= "Blues", linecolor = 'black' , linewidth = 1 , annot = True, fmt='',xticklabels = labels,yticklabels = labels)
```

```
<Axes: >
```

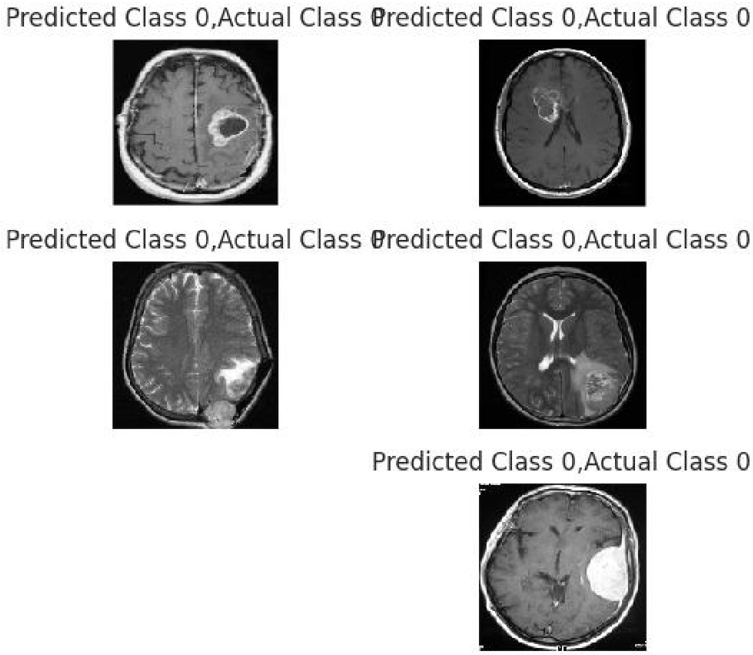



```
correct = np.nonzero(predictions == y_test)[0]
incorrect = np.nonzero(predictions != y_test)[0]
```

Some of the Correctly Predicted Classes

```
i = 0
for c in correct[:6]:
    plt.subplot(3,2,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(x_test[c].reshape(150,150), cmap="gray", interpolation='none')
    plt.title("Predicted Class {},Actual Class {}".format(predictions[c], y_test[c]))
    plt.tight_layout()
    i += 1
```

```
<ipython-input-22-3b0e8ec19e68>:3: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 and will be removed two minor releases later; explicitly call ax.remove() as needed.
  plt.subplot(3,2,i+1)
```



Some of the Incorrectly Predicted Classes

```
i = 0
for c in incorrect[:6]:
    plt.subplot(3,2,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(x_test[c].reshape(150,150), cmap="gray", interpolation='none')
    plt.title("Predicted Class {},Actual Class {}".format(predictions[c], y_test[c]))
    plt.tight_layout()
    i += 1
```

```
<ipython-input-23-d863d2b73908>:3: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 and will be removed two minor releases later; explicitly call ax.remove() as needed.
  plt.subplot(3,2,i+1)
```