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

```
Mounted at /content/drive
```

```
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
from keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import adam_v2

from sklearn.metrics import classification_report,confusion_matrix

import tensorflow as tf

import cv2
import os

import numpy as np
```

```
labels = ['rugby', 'soccer']
img_size = 224
def get_data(data_dir):
    data = []
    for label in labels:
        path = os.path.join(data_dir, label)
        class_num = labels.index(label)
        print(path)
        for img in os.listdir(path):
            try:

                img_arr = cv2.imread(os.path.join(path, img))[...,::-1] #convert BGR to RGB f
                print(img_arr)
                resized_arr = cv2.resize(img_arr, (img_size, img_size)) # Reshaping images to
                data.append([resized_arr, class_num])
            except Exception as e:
                print(e)
    return np.array(data)
```

```
#Now we can easily fetch our train and validation data.
train = get_data('/content/drive/MyDrive/input/train')
```

```
val = get_data('/content/drive/MyDrive/input/test')
```

```
 [[190 198 161]
  [186 194 155]
  [173 184 141]
  ...
  [171 180 135]
  [167 176 129]
  [169 177 130]]]
[[[ 71  74  55]
  [ 75  77  55]
  [ 82  77  57]
  ...
  [232 242 251]
  [232 241 250]
  [233 239 251]]

 [[ 78  78  54]
  [ 80  78  55]
  [ 84  77  59]

  ...
  [230 240 250]
  [232 241 250]
  [232 241 250]]

 [[ 83  78  56]
  [ 82  80  55]
  [ 88  84  59]
  ...
  [231 241 250]
  [229 242 250]
  [230 240 250]]

 ...

 [[ 86  96  61]
  [ 89  89  63]
  [ 95  76  61]
  ...
  [ 54  70  44]
  [ 54  68  42]
  [ 68  85  49]]

 [[ 87  99  63]
  [110 121  78]
  [112 119  78]
  ...
  [ 71 104  51]
  [ 65  90  48]
  [ 65  93  53]]

 [[ 72  92  55]
  [ 74  91  59]
  [ 80 100  63]
  ...
```

```
        [ 74 103  55]
        [ 69  96  53]
        [ 65  96  55]]]
```

```
x_train = []
y_train = []
x_val = []
y_val = []

for feature, label in train:
  x_train.append(feature)
  y_train.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_train.reshape(-1, img_size, img_size, 1)
y_train = np.array(y_train)

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

```
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,padding="same", activation="relu", input_shape=(224,224,3)))
model.add(MaxPool2D())
```

```
model.add(Conv2D(32, 3, padding="same", activation="relu"))
model.add(MaxPool2D())

model.add(Conv2D(64, 3, padding="same", activation="relu"))
model.add(MaxPool2D())
model.add(Dropout(0.4))

model.add(Flatten())
model.add(Dense(128,activation="relu"))
model.add(Dense(2, activation="softmax"))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 224, 224, 32)      896

 max_pooling2d (MaxPooling2D  (None, 112, 112, 32)     0
 )

 conv2d_1 (Conv2D)           (None, 112, 112, 32)      9248

 max_pooling2d_1 (MaxPooling  (None, 56, 56, 32)       0
 2D)

 conv2d_2 (Conv2D)           (None, 56, 56, 64)        18496

 max_pooling2d_2 (MaxPooling  (None, 28, 28, 64)       0
 2D)

 dropout (Dropout)           (None, 28, 28, 64)        0

 flatten (Flatten)           (None, 50176)             0

 dense (Dense)               (None, 128)               6422656

 dense_1 (Dense)             (None, 2)                 258

=================================================================
Total params: 6,451,554
Trainable params: 6,451,554
Non-trainable params: 0
_________________________________________________________________
```

```
from keras.optimizer_v2.adam import Adam
opt = Adam(learning_rate=0.000001)
model.compile(optimizer = opt , loss = tf.keras.losses.SparseCategoricalCrossentropy(from_log
```

```
history = model.fit(x_train,y_train,epochs = 50 , validation_data = (x_val, y_val))
```

```
Epoch 1/50
/usr/local/lib/python3.7/dist-packages/tensorflow/python/util/dispatch.py:1082: UserW
  return dispatch_target(*args, **kwargs)
77/77 [==============================] - 154s 2s/step - loss: 0.7003 - accuracy: 0.50
Epoch 2/50
77/77 [==============================] - 156s 2s/step - loss: 0.7003 - accuracy: 0.50
Epoch 3/50
77/77 [==============================] - 156s 2s/step - loss: 0.7003 - accuracy: 0.50
Epoch 4/50
77/77 [==============================] - 156s 2s/step - loss: 0.7003 - accuracy: 0.50
Epoch 5/50
77/77 [==============================] - 158s 2s/step - loss: 0.7000 - accuracy: 0.50
Epoch 6/50
77/77 [==============================] - 156s 2s/step - loss: 0.7000 - accuracy: 0.50
Epoch 7/50
77/77 [==============================] - 155s 2s/step - loss: 0.7000 - accuracy: 0.50
Epoch 8/50
77/77 [==============================] - 156s 2s/step - loss: 0.7003 - accuracy: 0.50
Epoch 9/50
77/77 [==============================] - 156s 2s/step - loss: 0.6999 - accuracy: 0.50
Epoch 10/50
77/77 [==============================] - 154s 2s/step - loss: 0.7003 - accuracy: 0.50
Epoch 11/50
77/77 [==============================] - 156s 2s/step - loss: 0.6999 - accuracy: 0.50
Epoch 12/50
77/77 [==============================] - 154s 2s/step - loss: 0.6999 - accuracy: 0.50
Epoch 13/50
77/77 [==============================] - 153s 2s/step - loss: 0.7003 - accuracy: 0.50
Epoch 14/50
77/77 [==============================] - 154s 2s/step - loss: 0.7003 - accuracy: 0.50
Epoch 15/50
77/77 [==============================] - 153s 2s/step - loss: 0.7003 - accuracy: 0.50
Epoch 16/50
77/77 [==============================] - 153s 2s/step - loss: 0.6999 - accuracy: 0.50
Epoch 17/50
77/77 [==============================] - 154s 2s/step - loss: 0.7002 - accuracy: 0.50
Epoch 18/50
77/77 [==============================] - 153s 2s/step - loss: 0.7002 - accuracy: 0.50
Epoch 19/50
77/77 [==============================] - 156s 2s/step - loss: 0.6999 - accuracy: 0.50
Epoch 20/50
77/77 [==============================] - 157s 2s/step - loss: 0.6999 - accuracy: 0.50
Epoch 21/50
77/77 [==============================] - 154s 2s/step - loss: 0.7002 - accuracy: 0.50
Epoch 22/50
77/77 [==============================] - 155s 2s/step - loss: 0.7002 - accuracy: 0.50
Epoch 23/50
77/77 [==============================] - 154s 2s/step - loss: 0.7002 - accuracy: 0.50
Epoch 24/50
77/77 [==============================] - 152s 2s/step - loss: 0.6999 - accuracy: 0.50
Epoch 25/50
77/77 [==============================] - 152s 2s/step - loss: 0.6999 - accuracy: 0.50
Epoch 26/50
77/77 [==============================] - 152s 2s/step - loss: 0.7002 - accuracy: 0.50
Epoch 27/50
```

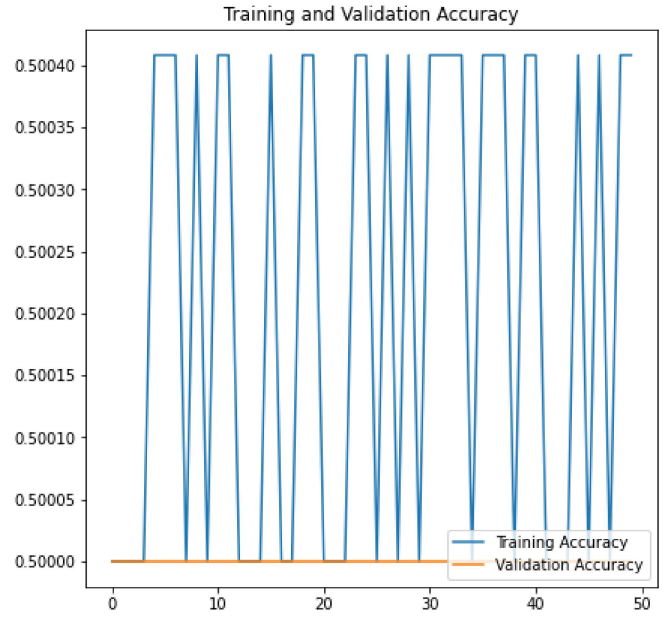
```
77/77 [==============================] - 151s 2s/step - loss: 0.6999 - accuracy: 0.50
```

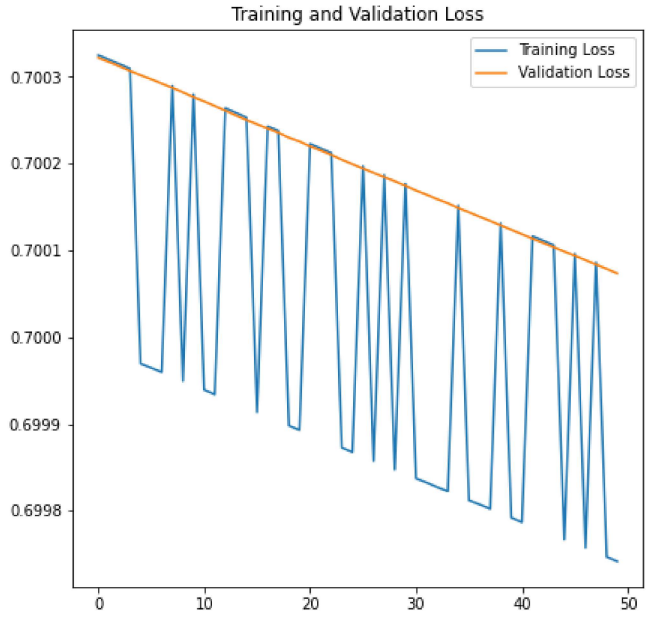
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(50)

plt.figure(figsize=(15, 15))
plt.subplot(2, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(2, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```






```
predictions = np.argmax(model.predict(x_val), axis=-1),
predictions = np.asarray(predictions)
predictions = predictions.reshape(1,-1)[0]
```

```
print(classification_report(y_val, predictions, target_names = ['Rugby (Class 0)','Soccer (Cl
```

```
                  precision    recall  f1-score   support

 Rugby (Class 0)       0.50      1.00      0.67       305
Soccer (Class 1)       0.00      0.00      0.00       305

        accuracy                           0.50       610
       macro avg       0.25      0.50      0.33       610
    weighted avg       0.25      0.50      0.33       610

/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Undefine
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Undefine
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Undefine
  _warn_prf(average, modifier, msg_start, len(result))
```

Task 1: Run the above code with given dataset.

Task 2: Run the code with different dataset