Pattern Recognition (COM511)
Project-Report

GROUP-26

CED17I023
CED17I034
CED17I042

Classify Input dance image into respective Indian classical dance style using CNN

```
#Import the libraries
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense,Flatten,Conv2D,MaxPooling2D,Dropout
from tensorflow.keras import layers
from keras.utils import to_categorical
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import pandas as pd
import os.path
import numpy as np
import cv2
```

Importing the required packages

```
#Load the data
dataset=pd.read_csv(r'C:\Users\duvvu\OneDrive\Desktop\PR\667015_1175090_bundle_archive\train.csv')
```

Loading data from .csv file (you need to change the path in order to run the code

```
images = np.array(pd.read_csv(r'C:\Users\duvvu\OneDrive\Desktop\PR\667015_1175090_bundle_archive\train.csv', usecols = ['Image'])
targets = np.array(pd.read_csv(r'C:\Users\duvvu\OneDrive\Desktop\PR\667015_1175090_bundle_archive\train.csv', usecols = ['target'
```

Loading data into two seperate arrays

```
x1=[]
y=[]
datasetpath="C:\\Users\\duvvu\\OneDrive\\Desktop\\PR\\667015_1175090_bundle_archive\\train\\"
for i in range(len(images)):
    x1.append(datasetpath+str(images[i][0]))
for i in range(len(targets)):
    y.append(targets[i][0])
```

Adding path to the images.

An array y_unique is to identify the unique elements in the y array.

```
# Images migh be in different size. In this section I assigning all image at same size of 224*224
img_width = 224
img_height = 224
```

```
print(x1[3])
print(y[3])
x2=[]
x=[]

for i in range(len(x1)):
    x.append(cv2.resize(cv2.imread(x1[i]), (img_width,img_height), interpolation=cv2.INTER_CUBIC))
```

Resizing the images to the same size.

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = None)
```

```
y_train_one_hot=to_categorical(y_train)
y_test_one_hot=to_categorical(y_test)
```

Choosing the train and test data set .
Converting y_train,y_test inorder to give it to the neural network

```
#Normalizing the pixel to values between 0 and 1
x_train=np.divide(x_train,255)
x_test=np.divide(x_test,255)
```

Normalizing the pixel values between 0 to 1

```
#create the models

from keras.applications.vgg16 import VGG16

vggmodel =VGG16(weights='imagenet', include_top=False, input_shape = (224, 224, 3),pooling='max')

vggmodel.trainable = False
model = Sequential([
  vggmodel,
  Dense(1024, activation='relu'),
  Dropout(0.25),
  Dense(256, activation='relu'),
  Dropout(0.25),
  Dense(8, activation='softmax'),
```

VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman

Tried in another way but there wa an error that resources(RAM) aren't enough
So We choose VGG16

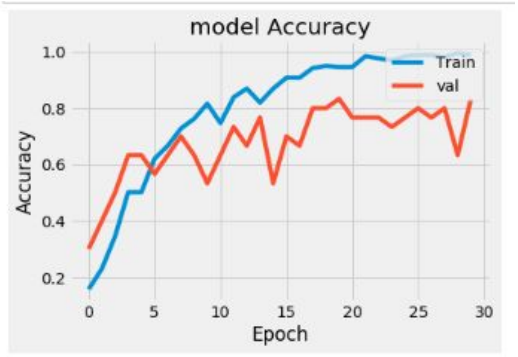
```
]: #compile the model
   model.compile(loss='categorical_crossentropy',
                 optimizer='adam',
                 metrics=['accuracy'])
```

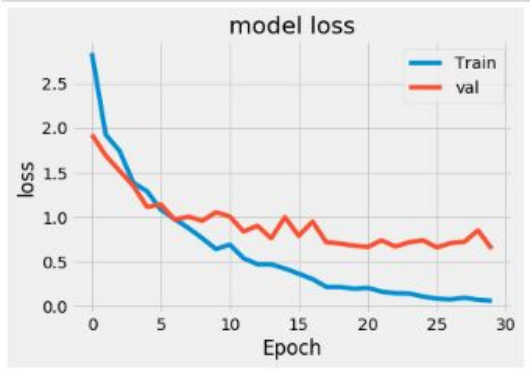
```
]: #train the model
   hist = model.fit(x_train,y_train_one_hot,
                    batch_size=16,
                    epochs=30,
                    validation_split=0.1)
```

Compiling and training the model

```
In [162]: #Evaluate the model using the test data set
          model.evaluate(x_test,y_test_one_hot)[1]

          73/73 [==============================] - 36s 491ms/step

Out[162]: 0.6301369667053223
```

Evaluate the model using test data

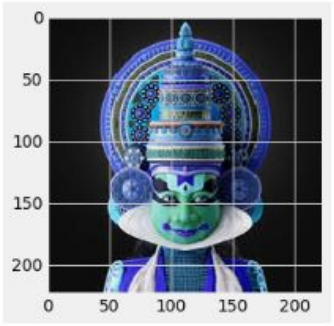




Graphs for the model

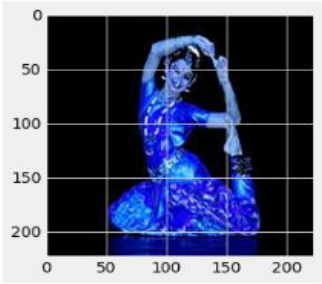
```
: #Get the model predictions
predictions=model.predict(np.array([resized_test_image]))

#Sort the predictions from least to greatest
list_index=[0,1,2,3,4,5,6,7]
x=predictions

for i in range(8):
    for j in range(8):
        if x[0][list_index[i]]>x[0][list_index[j]]:
            list_index[i],list_index[j]=list_index[j],list_index[i]

print(y_unique[list_index[0]])

#show the sorted list
print(list_index)

#prediction
for i in range(7):
    print(y_unique[list_index[i]])
```

This is used to predict the image



In [172]:
```
#Get the model predictions
predictions=model.predict(np.array([resized_test_image]))

#Sort the predictions from least to greatest
list_index=[0,1,2,3,4,5,6,7]
x=predictions

for i in range(8):
    for j in range(8):
        if x[0][list_index[i]]>x[0][list_index[j]]:
            list_index[i],list_index[j]=list_index[j],list_index[i]

print(y_unique[list_index[0]])

#show the sorted list
#print(list_index)

#prediction
#for i in range(7):
    #print(y_unique[list_index[i]])
```

```
kathakali
```



In [186]:
```
#Get the model predictions
predictions=model.predict(np.array([resized_test_image]))

#Sort the predictions from least to greatest
list_index=[0,1,2,3,4,5,6,7]
x=predictions

for i in range(8):
    for j in range(8):
        if x[0][list_index[i]]>x[0][list_index[j]]:
            list_index[i],list_index[j]=list_index[j],list_index[i]

print(y_unique[list_index[0]])

#show the sorted list
print(list_index)

#prediction
for i in range(7):
    print(y_unique[list_index[i]])
```

```
bharatanatyam
```