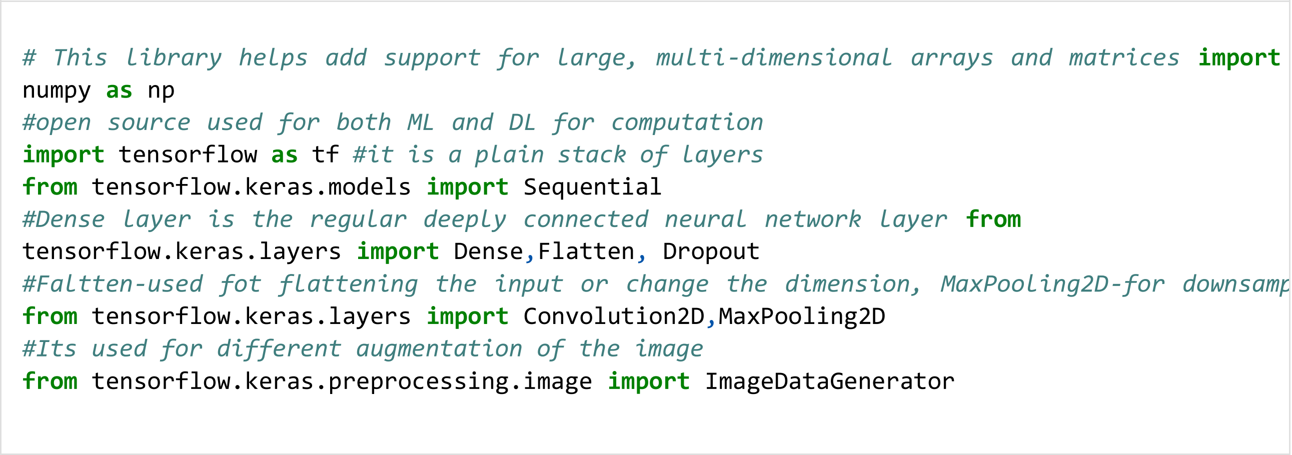
TEAM ID : PNT2022TMID24103

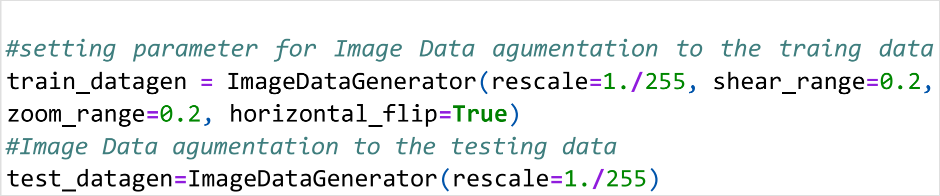
Model Building

Importing libraries



```
# This library helps add support for large, multi-dimensional arrays and matrices import
numpy as np
#open source used for both ML and DL for computation
import tensorflow as tf #it is a plain stack of layers
from tensorflow.keras.models import Sequential
#Dense layer is the regular deeply connected neural network layer from
tensorflow.keras.layers import Dense,Flatten, Dropout
#Faltten-used fot flattening the input or change the dimension, MaxPooling2D-for downsamp
from tensorflow.keras.layers import Convolution2D,MaxPooling2D
#Its used for different augmentation of the image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

Augmenting the data



```
#setting parameter for Image Data agumentation to the traing data
train_datagen = ImageDataGenerator(rescale=1./255, shear_range=0.2,
zoom_range=0.2, horizontal_flip=True)
#Image Data agumentation to the testing data
test_datagen=ImageDataGenerator(rescale=1./255)
```

Loading our data and performing data agumentation

```
#performing data agumentation to train data
x_train = train_datagen.flow_from_directory(r'C:\\Users\\Midhun\\OneDrive\\Desktop\\Gestu
                                              target_size=(64, 64), batch_size=3,
                                              color_mode='grayscale',
                                              class_mode='categorical')
#performing data agumentation to test data
x_test = test_datagen.flow_from_directory(r'C:\\Users\\Midhun\\OneDrive\\Desktop\\Gesture
                                            target_size=(64, 64), batch_size=3,
                                            color_mode='grayscale',
                                            class_mode='categorical')
```

```
Found 594 images belonging to 6 classes. Found
30 images belonging to 6 classes.
```

```
print(x_train.class_indices)#checking the number of classes
```

```
{'0': 0, '1': 1, '2': 2, '3': 3, '4': 4, '5': 5}
```

Model Creation

```
# Initializing the CNN model = Sequential()
```

```
# First convolution layer and pooling
model.add(Convolution2D(32,   (3,   3),   input_shape=(64,   64,   1),   activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
# Second convolution layer and pooling
model.add(Convolution2D(32, (3, 3), activation='relu'))
# input_shape is going to be the pooled feature maps from the previous convolution layer
model.add(MaxPooling2D(pool_size=(2,2)))
```

```
# Flattening the layers i.e. input layer model.add(Flatten())
```

```
# Adding a fully connected layer, i.e. Hidden Layer model.add(Dense(units=512
, activation='relu'))
```

```
# softmax for categorical analysis, Output Layer model.add(Dense(units=6,
activation='softmax'))
```

```
model.summary()#summary of our model
```

```
Model: "sequential"
```

```
Layer (type)        Output Shape                Param #

 conv2d (Conv2D)              (None, 62, 62, 32)        320

max_pooling2d (MaxPooling2D) (None, 31, 31, 32)        0

 conv2d_1 (Conv2D)            (None, 29, 29, 32)        9248

max_pooling2d_1 (MaxPooling2 (None, 14, 14, 32)        0
        flatten (Flatten) (None, 6272)

                                                        0
 dense (Dense)                (None, 512)               3211776

 dense_1 (Dense)              (None, 6)                 3078

Total params: 3,224,422
Trainable params: 3,224,422
Non-trainable params: 0
```

Model Compilation

```
# Compiling the CNN
# categorical_crossentropy for more than 2
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
```

Model fitting

```
# It will generate packets of train and test data for training
model.fit_generator(x_train, steps_per_epoch = 594/3 , epochs =
25,
                    validation_data = x_test, validation_steps =
                    30/3 )
```

```
C:\Users\Midhun\anaconda3\lib\site-packages\tensorflow\python\keras\engine\training.py:194
0: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future versi
on. Please use `Model.fit`, which supports generators.
  warnings.warn('`Model.fit_generator` is deprecated and '
Epoch 1/25
198/198 [==============================] - 7s 34ms/step - loss: 1.3144 - accuracy: 0.4798
- val_loss: 0.7614 - val_accuracy: 0.7000
 Epoch 2/25
198/198 [==============================] - 7s 34ms/step - loss: 0.6828 - accuracy: 0.7155
- val_loss: 0.5644 - val_accuracy: 0.8000
 Epoch 3/25
198/198 [==============================] - 7s 33ms/step - loss: 0.4049 - accuracy: 0.8552
- val_loss: 0.7858 - val_accuracy: 0.7667
 Epoch 4/25
198/198 [==============================] - 7s 34ms/step - loss: 0.3155 - accuracy: 0.8721 -
val_loss: 0.2433 - val_accuracy: 0.9667
 Epoch 5/25
198/198 [==============================] - 7s 34ms/step - loss: 0.1929 - accuracy: 0.9327
- val_loss: 0.3210 - val_accuracy: 0.9667
 Epoch 6/25
```

```
198/198 [==============================] - 7s 34ms/step - loss: 0.1761 - accuracy: 0.9495
- val_loss: 0.5928 - val_accuracy: 0.9000
 Epoch 7/25
198/198 [==============================] - 7s 34ms/step - loss: 0.1257 - accuracy: 0.9613
- val_loss: 0.3547 - val_accuracy: 0.9333
 Epoch 8/25
198/198 [==============================] - 7s 34ms/step - loss: 0.0959 - accuracy: 0.9630 -
val_loss: 0.4215 - val_accuracy: 0.9667
 Epoch 9/25
198/198 [==============================] - 7s 35ms/step - loss: 0.1353 - accuracy: 0.9444
- val_loss: 0.3127 - val_accuracy: 0.9667
 Epoch 10/25
198/198 [==============================] - 7s 34ms/step - loss: 0.0985 - accuracy: 0.9697
- val_loss: 0.3157 - val_accuracy: 0.9667
 Epoch 11/25
198/198 [==============================] - 7s 34ms/step - loss: 0.0824 - accuracy: 0.9747
- val_loss: 0.3259 - val_accuracy: 0.9667
 Epoch 12/25
198/198 [==============================] - 7s 34ms/step - loss: 0.0474 - accuracy: 0.9865 -
val_loss: 0.4769 - val_accuracy: 0.9333
 Epoch 13/25
198/198 [==============================] - 7s 35ms/step - loss: 0.0570 - accuracy: 0.9848
- val_loss: 0.1794 - val_accuracy: 0.9667
 Epoch 14/25
198/198 [==============================] - 7s 34ms/step - loss: 0.0704 - accuracy: 0.9714
- val_loss: 0.4142 - val_accuracy: 0.9333
 Epoch 15/25
198/198 [==============================] - 7s 34ms/step - loss: 0.0749 - accuracy: 0.9697
- val_loss: 0.4670 - val_accuracy: 0.9667
 Epoch 16/25
198/198 [==============================] - 7s 34ms/step - loss: 0.0529 - accuracy: 0.9832
- val_loss: 0.3779 - val_accuracy: 0.9667
 Epoch 17/25
198/198 [==============================] - 7s 34ms/step - loss: 0.0412 - accuracy: 0.9899
- val_loss: 0.4824 - val_accuracy: 0.9000
 Epoch 18/25
198/198 [==============================] - 7s 35ms/step - loss: 0.0394 - accuracy: 0.9798
- val_loss: 0.2046 - val_accuracy: 0.9667
 Epoch 19/25
198/198 [==============================] - 7s 35ms/step - loss: 0.0870 - accuracy: 0.9747
- val_loss: 0.3575 - val_accuracy: 0.9667
 Epoch 20/25
198/198 [==============================] - 7s 34ms/step - loss: 0.0308 - accuracy: 0.9899 -
val_loss: 0.3367 - val_accuracy: 0.9667
 Epoch 21/25
198/198 [==============================] - 7s 35ms/step - loss: 0.0381 - accuracy: 0.9865
- val_loss: 0.2932 - val_accuracy: 0.9333   Epoch 22/25
198/198 [==============================] - 7s 35ms/step - loss: 0.0422 - accuracy: 0.9865
- val_loss: 0.5818 - val_accuracy: 0.8667
 Epoch 23/25
198/198 [==============================] - 7s 34ms/step - loss: 0.0319 - accuracy: 0.9865
- val_loss: 0.3459 - val_accuracy: 0.9667
 Epoch 24/25
198/198 [==============================] - 7s 35ms/step - loss: 0.0385 - accuracy: 0.9815 -
val_loss: 0.3301 - val_accuracy: 0.9667
 Epoch 25/25
198/198 [==============================] - ETA: 0s - loss: 0.0138 - accuracy: 0.99 - 7s 3
<tensorflow.python.keras.callbacks.History at 0x1b89d20da60>
```

Saving model

```
# Save the model model.save('gesture.h5')
```

```
model_json = model.to_json() with open("model-
bw.json", "w") as json_file:
    json_file.write(model_json)
```