```
import os
import numpy as np
import matplotlib.pyplot as plt
import random
import cv2
import PIL
import glob
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
!ls /content/
```

```
      model   sample_data   Test  'Test (2).zip'   Train
```

```
!mkdir '/content/Train'
```

```
!mv '/content/Train.zip' '/content/Train/'
```

```
!mkdir '/content/Test'
```

```
!mkdir '/content/model/'
```

```
!mv '/content/Test.zip' '/content/Test/'
```

```
!unzip '/content/Train/Train.zip'
```

```
  inflating: Twentynote/11.jpg
  inflating: Twentynote/12.jpg
  inflating: Twentynote/13.jpg
  inflating: Twentynote/15.jpg
  inflating: Twentynote/16.jpg
  inflating: Twentynote/17.jpg
  inflating: Twentynote/2.jpg
  inflating: Twentynote/20.jpg
  inflating: Twentynote/23.jpg
  inflating: Twentynote/26.jpg
  inflating: Twentynote/28.jpg
  inflating: Twentynote/29.jpg
  inflating: Twentynote/3.jpg
  inflating: Twentynote/32.jpg
  inflating: Twentynote/33.jpg
  inflating: Twentynote/4.jpg
  inflating: Twentynote/6.jpg
  inflating: Twentynote/7.jpg
  inflating: Twentynote/8.jpg
  inflating: Twentynote/9.jpg
```

```
!unzip '/content/Test/Test.zip'
```

```
Archive:  /content/Test/Test.zip
replace Test/1Hundrednote/1.jpg? [y]es, [n]o, [A]ll, [N]one, [r]ename: All
  inflating: Test/1Hundrednote/1.jpg
  inflating: Test/1Hundrednote/14.jpg
  inflating: Test/1Hundrednote/15.jpg
  inflating: Test/1Hundrednote/16.jpg
  inflating: Test/1Hundrednote/2.jpg
  inflating: Test/1Hundrednote/3.jpg
  inflating: Test/2Hundrednote/1.jpg
  inflating: Test/2Hundrednote/2.jpg
  inflating: Test/2Hundrednote/3.jpg
  inflating: Test/2Hundrednote/31.jpg
  inflating: Test/2Hundrednote/32.jpg
  inflating: Test/2Hundrednote/33.jpg
  inflating: Test/2Thousandnote/1.jpg
  inflating: Test/2Thousandnote/2.jpg
  inflating: Test/2Thousandnote/3.jpg
  inflating: Test/2Thousandnote/31.jpg
  inflating: Test/2Thousandnote/32.jpg
  inflating: Test/2Thousandnote/33.jpg
  inflating: Test/5Hundrednote/1.jpg
  inflating: Test/5Hundrednote/2.jpg
  inflating: Test/5Hundrednote/3.jpg
  inflating: Test/5Hundrednote/31.jpg
  inflating: Test/5Hundrednote/32.jpg
  inflating: Test/5Hundrednote/33.jpg
  inflating: Test/Fiftynote/1.jpg
  inflating: Test/Fiftynote/2.jpg
  inflating: Test/Fiftynote/27.jpg
  inflating: Test/Fiftynote/28.jpg
  inflating: Test/Fiftynote/29.jpg
  inflating: Test/Fiftynote/3.jpg
  inflating: Test/Tennote/1.jpg
  inflating: Test/Tennote/2.jpg
  inflating: Test/Tennote/3.jpg
  inflating: Test/Tennote/31.jpg
  inflating: Test/Tennote/32.jpg
  inflating: Test/Tennote/33.jpg
  inflating: Test/Twentynote/1.jpg
  inflating: Test/Twentynote/18.jpg
  inflating: Test/Twentynote/2.jpg
  inflating: Test/Twentynote/24.jpg
  inflating: Test/Twentynote/3.jpg
  inflating: Test/Twentynote/30.jpg
```

```
import os
import numpy as np
import matplotlib.pyplot as plt
import random
import cv2
import PIL
import glob
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from keras.preprocessing.image import ImageDataGenerator


ROOTPATH = '/content'
DATAPATH= ROOTPATH+'/Train'
TRAINPATH = ROOTPATH+'/Train'
TESTPATH =  ROOTPATH+'/Test'
MODELPATH = ROOTPATH+'/model/'


!ls /content/Train/
```

```
1Hundrednote  2Hundrednote  2Thousandnote  5Hundrednote  Fiftynote  Tennote  Twentynote
```

```
_1Hundrednote=glob.glob(DATAPATH+'/1Hundrednote/*')# [/content/Test/1Hundrednote/1.jpg,/content/Test/1Hundrednote/2.
_2Hundrednote=glob.glob(DATAPATH+'/2Hundrednote/*')
_2Thousandnote=glob.glob(DATAPATH+'/2Thousandnote/*')
_5Hundrednote=glob.glob(DATAPATH+'/5Hundrednote/*')
_Fiftynote=glob.glob(DATAPATH+'/Fiftynote/*')
_Tennote=glob.glob(DATAPATH+'/Tennote/*')
_Twentynote=glob.glob(DATAPATH+'/Twentynote/*')


print(len(_1Hundrednote),_1Hundrednote)
print(len(_2Hundrednote),_2Hundrednote)
print(len(_2Thousandnote),_2Thousandnote)
print(len(_2Thousandnote),_2Thousandnote)
print(len(_5Hundrednote),_5Hundrednote)
print(len(_Fiftynote),_Fiftynote)
print(len(_Tennote),_Tennote)
print(len(_Twentynote),_Twentynote)
```

```
22 ['/content/Train/1Hundrednote/4.jpg', '/content/Train/1Hundrednote/11.jpg', '/content/Train/1Hundrednote/21.jp
22 ['/content/Train/2Hundrednote/4.jpg', '/content/Train/2Hundrednote/26.jpg', '/content/Train/2Hundrednote/11.jp
21 ['/content/Train/2Thousandnote/4.jpg', '/content/Train/2Thousandnote/26.jpg', '/content/Train/2Thousandnote/11
21 ['/content/Train/2Thousandnote/4.jpg', '/content/Train/2Thousandnote/26.jpg', '/content/Train/2Thousandnote/11
22 ['/content/Train/5Hundrednote/4.jpg', '/content/Train/5Hundrednote/9.jpg', '/content/Train/5Hundrednote/26.jpg
22 ['/content/Train/Fiftynote/26.jpg', '/content/Train/Fiftynote/11.jpg', '/content/Train/Fiftynote/21.jpg', '/co
22 ['/content/Train/Tennote/4.jpg', '/content/Train/Tennote/26.jpg', '/content/Train/Tennote/21.jpg', '/content/T
22 ['/content/Train/Twentynote/4.jpg', '/content/Train/Twentynote/9.jpg', '/content/Train/Twentynote/26.jpg', '/c
```

```
dataset_classes=[_1Hundrednote,_2Hundrednote,_2Thousandnote,_5Hundrednote,_Fiftynote,_Tennote,_Twentynote]
total_class=len(dataset_classes)
print('Total dataset class: ',total_class)
```

```
Total dataset class:  7
```

```
IMAGE_SIZE=224
BATCH_SIZE=16

#pre_processing_training
train_datagen = ImageDataGenerator(
        rescale=1./255,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode='nearest',
        validation_split=0.2)

training_set = train_datagen.flow_from_directory(
        DATAPATH,
        shuffle=True,
        target_size=(IMAGE_SIZE,IMAGE_SIZE),
        batch_size=BATCH_SIZE,
        class_mode='categorical',
        subset='training')


validation_set = train_datagen.flow_from_directory(
        DATAPATH,
        shuffle=True,
        target_size=(IMAGE_SIZE, IMAGE_SIZE),
        batch_size=BATCH_SIZE,
        class_mode='categorical',
        subset='validation',
        )  # Specify the validation split
```

```
Found 125 images belonging to 8 classes.
Found 28 images belonging to 8 classes.
```

```
test_datagen = ImageDataGenerator(rescale=1./255)
test_set = test_datagen.flow_from_directory(
        TESTPATH,
         shuffle=False,
        target_size=(IMAGE_SIZE,IMAGE_SIZE),
        batch_size=BATCH_SIZE,
        class_mode='categorical')
```

```
Found 42 images belonging to 8 classes.
```

```
training_set.class_indices
```

```
{'.ipynb_checkpoints': 0,
 '1Hundrednote': 1,
 '2Hundrednote': 2,
 '2Thousandnote': 3,
 '5Hundrednote': 4,
 'Fiftynote': 5,
 'Tennote': 6,
 'Twentynote': 7}
```

```
total_class=len(training_set.class_indices)
print('Number of classes in dataset: ',total_class)
```

```
Number of classes in dataset:  8
```

```
print("Shape of training set:", training_set.image_shape)
```

```
Shape of training set: (224, 224, 3)
```

```
print("Shape of training set:", validation_set.image_shape)
```

```
Shape of training set: (224, 224, 3)
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.layers import Activation, Dense, Flatten
```

```
vgg16_model = Sequential()

pretrained_model= tf.keras.applications.VGG16(include_top=False,
                   input_shape=(224,224,3),
                   pooling='max',classes=8,
                   weights='imagenet')
for layer in pretrained_model.layers:
        layer.trainable=False

vgg16_model.add(pretrained_model)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_or
58889256/58889256 [==============================] - 0s 0us/step
```

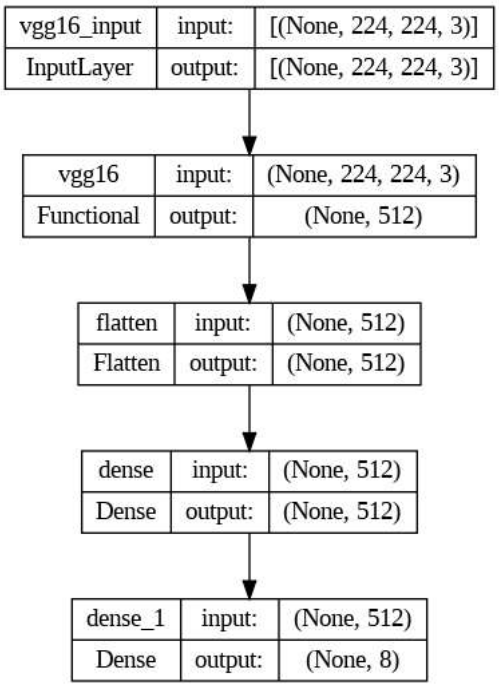
```
vgg16_model.add(Flatten())
vgg16_model.add(Dense(512, activation='relu'))
vgg16_model.add(Dense(8, activation='softmax'))
```

```
vgg16_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 512)               14714688

 flatten (Flatten)           (None, 512)               0

 dense (Dense)               (None, 512)               262656

 dense_1 (Dense)             (None, 8)                 4104

=================================================================
Total params: 14981448 (57.15 MB)
Trainable params: 266760 (1.02 MB)
Non-trainable params: 14714688 (56.13 MB)
_________________________________________________________________
```

```
vgg16_model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
```

```
from tensorflow.keras.utils import plot_model

# Plot the model and save it to an image file
plot_model(vgg16_model, to_file='vgg16_model.png', show_shapes=True, show_layer_names=True)
```



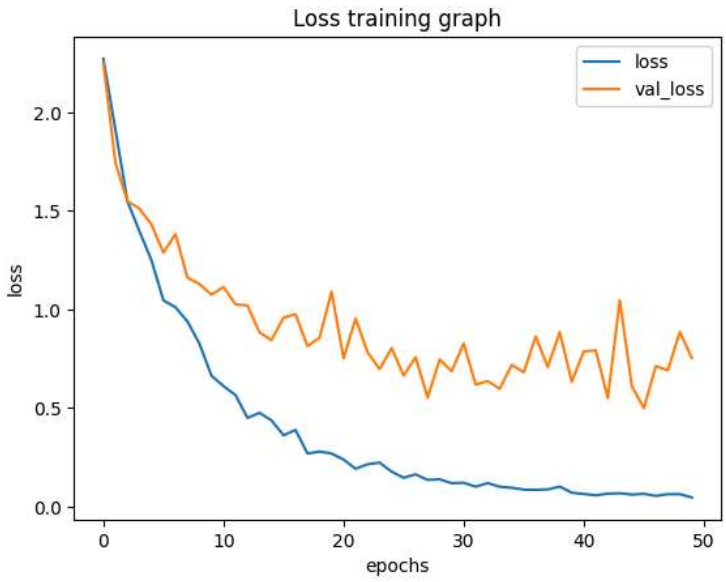

```
from tensorflow.keras.callbacks import ModelCheckpoint
checkpointer = ModelCheckpoint(filepath=MODELPATH+'VGG16_Pretrained.model.best.hdf5', verbose=1 ,save_best_only=True)
```

```
history=vgg16_model.fit(training_set,
                epochs=50,
                validation_data=validation_set,
                callbacks=[checkpointer])
```

```
Epoch 45/50
8/8 [==============================] - ETA: 0s - loss: 0.0604 - accuracy: 1.0000
Epoch 45: val_loss did not improve from 0.55021
8/8 [==============================] - 3s 332ms/step - loss: 0.0604 - accuracy: 1.0000 - val_loss: 0.6120 - va
Epoch 46/50
8/8 [==============================] - ETA: 0s - loss: 0.0644 - accuracy: 1.0000
Epoch 46: val_loss improved from 0.55021 to 0.49901, saving model to /content/model/VGG16_Pretrained.model.bes
8/8 [==============================] - 3s 387ms/step - loss: 0.0644 - accuracy: 1.0000 - val_loss: 0.4990 - va
Epoch 47/50
8/8 [==============================] - ETA: 0s - loss: 0.0535 - accuracy: 1.0000
Epoch 47: val_loss did not improve from 0.49901
8/8 [==============================] - 3s 317ms/step - loss: 0.0535 - accuracy: 1.0000 - val_loss: 0.7122 - va
Epoch 48/50
8/8 [==============================] - ETA: 0s - loss: 0.0625 - accuracy: 1.0000
Epoch 48: val_loss did not improve from 0.49901
8/8 [==============================] - 2s 312ms/step - loss: 0.0625 - accuracy: 1.0000 - val_loss: 0.6910 - va
Epoch 49/50
8/8 [==============================] - ETA: 0s - loss: 0.0625 - accuracy: 0.9920
Epoch 49: val_loss did not improve from 0.49901
8/8 [==============================] - 4s 468ms/step - loss: 0.0625 - accuracy: 0.9920 - val_loss: 0.8863 - va
Epoch 50/50
8/8 [==============================] - ETA: 0s - loss: 0.0455 - accuracy: 0.9920
Epoch 50: val_loss did not improve from 0.49901
8/8 [==============================] - 2s 309ms/step - loss: 0.0455 - accuracy: 0.9920 - val_loss: 0.7534 - va
```

```
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.title('Training accuracy graph')
plt.plot(history.history['accuracy'],label='accuracy')
plt.plot(history.history['val_accuracy'],label='val_accuracy')
plt.legend()
plt.show()
```


```
plt.xlabel('epochs')
plt.ylabel('loss')
plt.title('Loss training graph')
plt.plot(history.history['loss'],label='loss')
plt.plot(history.history['val_loss'],label='val_loss')
plt.legend()
plt.show()
```




```
test_loss,test_accuracy=vgg16_model.evaluate(test_set)
print('Test Loss: ',test_loss)
print('Test Accuracy: ',test_accuracy)
```

```
3/3 [==============================] - 3s 1s/step - loss: 1.1546 - accuracy: 0.6429
Test Loss:  1.1546310186386108
Test Accuracy:  0.6428571343421936
```

```
print('Accuracy of the model is : ',test_accuracy*100)
```

```
Accuracy of the model is :  64.28571343421936
```

```
predicted_result=vgg16_model.predict(test_set)
predicted_result[:5]
```

```
3/3 [==============================] - 0s 87ms/step
array([[1.6346118e-07, 9.0310216e-01, 9.4300704e-03, 1.5244492e-03,
        5.8278019e-05, 4.6440167e-03, 1.1646249e-02, 6.9594547e-02],
       [7.8369970e-07, 8.2756177e-02, 1.2216610e-02, 1.8757635e-03,
        1.9149543e-04, 8.0890574e-02, 8.1139974e-02, 7.4092865e-01],
       [1.0971662e-06, 3.1259072e-01, 5.9706833e-02, 1.4478811e-04,
        1.9144529e-04, 3.0253306e-01, 7.0643939e-02, 2.5418821e-01],
       [1.5412869e-06, 7.4867809e-01, 8.6827306e-03, 3.2772534e-02,
        2.7515375e-04, 2.3179189e-03, 1.0074695e-02, 1.9719736e-01],
       [1.7059615e-07, 9.8380512e-01, 4.8967809e-03, 7.8356266e-04,
        8.5782576e-03, 5.0648488e-04, 1.2500278e-03, 1.7957791e-04]],
      dtype=float32)
```

```
predicted_class=np.argmax(predicted_result,axis=-1)
predicted_class[:5]
```

```
array([1, 7, 1, 1, 1])
```

```
test_classes=test_set.classes
test_classes
```

```
array([1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4,
       4, 4, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 7, 7, 7, 7, 7, 7],
      dtype=int32)
```

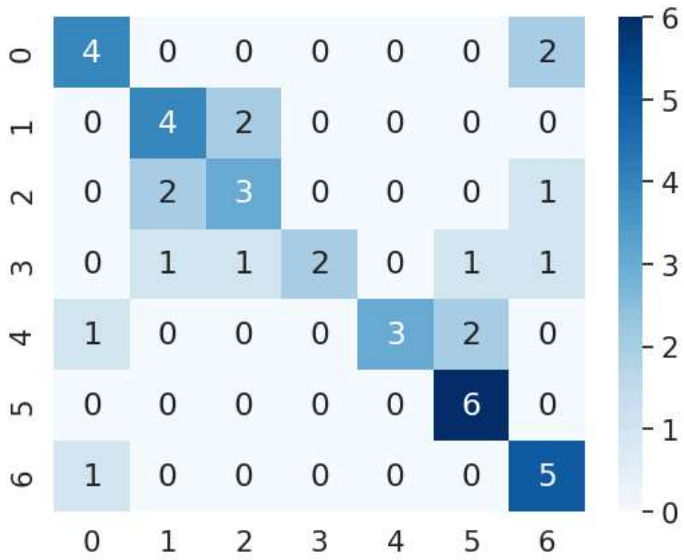
```
from sklearn.metrics import confusion_matrix
```

```
cm=confusion_matrix(test_classes,predicted_class)
print(cm)
```

```
[[4 0 0 0 0 0 2]
 [0 4 2 0 0 0 0]
 [0 2 3 0 0 0 1]
 [0 1 1 2 0 1 1]
 [1 0 0 0 3 2 0]
 [0 0 0 0 0 6 0]
 [1 0 0 0 0 0 5]]
```

```
import seaborn as sns
sns.set(font_scale=1.4)
sns.heatmap(cm, annot=True,fmt='d',cmap="Blues")
```

```
<Axes: >
```




```
from sklearn.metrics import accuracy_score
print('Accuracy score: ',accuracy_score(test_classes,predicted_class))
```

```
Accuracy score:  0.6428571428571429
```

```
from sklearn.metrics import classification_report
print('Classification Report \n',classification_report(test_classes,predicted_class))
```

```
Classification Report
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        72
           1       0.95      1.00      0.98        60
           2       1.00      1.00      1.00        60
           3       1.00      1.00      1.00        60
           4       0.95      0.95      0.95        60
           5       1.00      1.00      1.00        72
           6       1.00      1.00      1.00        47
           7       1.00      0.96      0.98        72
           8       1.00      1.00      1.00        72

    accuracy                           0.99       575
   macro avg       0.99      0.99      0.99       575
weighted avg       0.99      0.99      0.99       575
```

```
import time
t = time.time()

export_path_keras = "/content/model/Model_11_vgg16_Pretrained{}_model_{}.h5".format(test_accuracy,int(t))
print(export_path_keras)
vgg16_model.save(export_path_keras)
```

```
Final Thesis/Saved Model/Model_11_vgg16_Pretrained0.9895651936531067_model_1647614762.h5
```

```
from tensorflow.keras.models import load_model


model_path=export_path_keras
reload_model=load_model(model_path)
reload_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 512)               14714688

 flatten (Flatten)           (None, 512)               0

 dense (Dense)               (None, 512)               262656

 dense_1 (Dense)             (None, 9)                 4617

=================================================================
Total params: 14,981,961
Trainable params: 267,273
Non-trainable params: 14,714,688
_________________________________________________________________
```

```
print(len(reload_model.weights))
print(reload_model.output_shape)
```

```
30
(None, 9)
```

```
reload_model.layers
```

```
[<keras.engine.functional.Functional at 0x7fec35ecaa90>,
 <keras.layers.core.flatten.Flatten at 0x7fec35ec7f50>,
 <keras.layers.core.dense.Dense at 0x7fec35e9ced0>,
 <keras.layers.core.dense.Dense at 0x7fec35e4b3d0>]
```

```
t = time.time()

export_path_sm = "/content/model/Model_11_vgg16_Pretrained {} Model {}".format(test_accuracy,int(t))
print(export_path_sm)

tf.saved_model.save(vgg16_model, export_path_sm)
```

```
Final Thesis/Saved Model/Model_11_vgg16_Pretrained 0.9895651936531067 Model 1647614786
INFO:tensorflow:Assets written to: Final Thesis/Saved Model/Model_11_vgg16_Pretrained 0.9895651936531067 Model 16
```

```
reload_tf_saved_model=tf.saved_model.load(export_path_sm)


reload_tf_saved_model.signatures['serving_default']
```

```
<ConcreteFunction signature_wrapper(*, vgg16_input) at 0x7FEC33365250>
```

```
reload_tf_saved_model
```

```
<tensorflow.python.saved_model.load.Loader._recreate_base_user_object.<locals>._UserObject at 0x7fec35bc25d0>
```

```
model=reload_model
```

```
def noteclass(cls):
    txt=pyttsx3.init()
    # if cls==0:
    #     ans="Two Taka"
    #     print(ans)
    #     txt.say(ans)
    #     txt.runAndWait()
    # el
    if cls==1:
        ans="1Hundrednote"
        print(ans)
        txt.say(ans)
        txt.runAndWait()
    elif cls==2:
        ans="2Hundrednote"
        print(ans)
        txt.say(ans)
        txt.runAndWait()
    elif cls==3:
        ans="2Thousandnote"
        print(ans)
        txt.say(ans)
        txt.runAndWait()
    elif cls==4:
        ans="5Hundrednote"
        print(ans)
        txt.say(ans)
        txt.runAndWait()
    elif cls==5:
        ans="Fiftynote"
        print(ans)
        txt.say(ans)
        txt.runAndWait()
    elif cls==6:
        ans="Tennote"
        print(ans)
        txt.say(ans)
        txt.runAndWait()
    elif cls==7:
        ans="Twentynote"
        print(ans)
        txt.say(ans)
        txt.runAndWait()
    else:
        ans="NA"
        print(ans)
        txt.say(ans)
        txt.runAndWait()


#""WIN 20220326 11 18 21 Pro"""
```