```
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Activation, Dense, Flatten, BatchNormalization, Conv2D,
MaxPool2D, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.metrics import categorical_crossentropy
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import confusion_matrix
import itertools
import os
import shutil
import random
import glob
import matplotlib.pyplot as plt
import warnings
from tensorflow.python.keras.utils.data_utils import Sequence
warnings.simplefilter(action='ignore', category=FutureWarning)
%matplotlib inline
from google.colab import drive
drive.mount('/content/drive')
```

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

```
train_path = '/content/drive/MyDrive/Data_tienVN'
valid_path = '/content/drive/MyDrive/Data_tienVN'
test_path = '/content/drive/MyDrive/Data_tienVN'
```

```
train_datagen = ImageDataGenerator(rescale=1./255, rotation_range=40,
                                   idth_shift_range=0.2, height_shift_range=0.2,
                                   hear_range=0.2, zoom_range=0.2,
                                   orizontal_flip=True, fill_mode='nearest')
test_datagen = ImageDataGenerator(rescale=1./255)
train_batches = train_datagen.flow_from_directory(train_path, target_size=(224, 224),
                                                  batch_size=10, class_mode='categorical')

validation_batches = test_datagen.flow_from_directory(valid_path, target_size=(224, 224),
                                                      batch_size=10, class_mode='categorical')

test_batches = test_datagen.flow_from_directory(test_path, target_size=(224, 224),
                                                batch_size=10, class_mode='categorical')
```

```
Found 43 images belonging to 7 classes.
Found 43 images belonging to 7 classes.
Found 43 images belonging to 7 classes.
```

```
imgs, labels = next(train_batches)
def plotImages(images_arr):
    fig, axes = plt.subplots(1, 10, figsize=(20,20))
    axes = axes.flatten()
    for img, ax in zip( images_arr, axes):
        ax.imshow(img)
        ax.axis('off')
```

Saved successfully!

```
    plt.tight_layout()
    plt.show()
```

```
model = Sequential([
    Conv2D(filters=64, kernel_size=(3, 3), activation='relu', padding = 'same', input_shape=(224,224,3))
    MaxPool2D(pool_size=(2, 2), strides=2),
    Conv2D(filters=128, kernel_size=(3, 3), activation='relu', padding = 'same'),
    MaxPool2D(pool_size=(2, 2), strides=2),
    Conv2D(filters=256, kernel_size=(3, 3), activation='relu', padding = 'same'),
    MaxPool2D(pool_size=(2, 2), strides=2),
    Dropout(0.2),
    Conv2D(filters=256, kernel_size=(3, 3), activation='relu', padding = 'same'),
    Flatten(),
    Dropout(0.5),
    Dense(units=7, activation='softmax')])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 224, 224, 64)      1792

 max_pooling2d (MaxPooling2D  (None, 112, 112, 64)     0
 )

 conv2d_1 (Conv2D)           (None, 112, 112, 128)     73856

 max_pooling2d_1 (MaxPooling  (None, 56, 56, 128)      0
 2D)

 conv2d_2 (Conv2D)           (None, 56, 56, 256)       295168

                             None, 28, 28, 256)        0


 dropout (Dropout)           (None, 28, 28, 256)       0

 conv2d_3 (Conv2D)           (None, 28, 28, 256)       590080

 flatten (Flatten)           (None, 200704)            0

 dropout_1 (Dropout)         (None, 200704)            0

 dense (Dense)               (None, 7)                 1404935

=================================================================
Total params: 2,365,831
Trainable params: 2,365,831
Non-trainable params: 0
_________________________________________________________________
```

Saved successfully!

```
model.compile(optimizer=Adam(learning_rate=0.0005), loss='categorical_crossentropy',
              metrics=['accuracy'])
model.fit(x=train_batches, steps_per_epoch=len(train_batches),
          validation_data=validation_batches, validation_steps=len(validation_batches), epochs=1, verbos
```

```
5/5 [==============================] - 2s 409ms/step - loss: 0.9452 - accuracy: 0.6744 - val_loss:
<keras.callbacks.History at 0x7fd5067a9e90>
```

```
model.save('Train_tien.h5')
```

```
classes = ['1000','10000','100000','2000','','20000','5000','50000']
print("Image Processing.......Compleated")
```

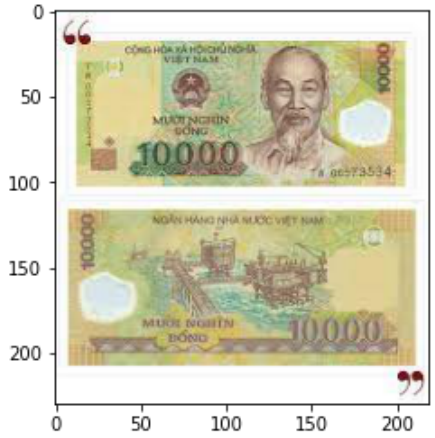
```
Image Processing.......Compleated
```

```
from google.colab import files
from keras.preprocessing import image
%matplotlib inline
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

uploaded=files.upload()

for fn in uploaded.keys():
  #predicting images
  path='/content/'+fn
  #In ảnh đọc được
  plt.imshow(mpimg.imread(path))

  img=image.load_img(path,target_size=(224,224))
  x=image.img_to_array(img)
  x=np.expand_dims(x,axis=0)
  images=np.vstack([x])
  y_predict = model.predict(images,batch_size=200000)
  print(y_predict)
```

```
[np.argmax(y_predict)])
```

Saved successfully!

```
n  Upload widget is only available when the cell has been executed in the current
browser session. Please rerun this cell to enable.
Saving tải xuống.jpg to tải xuống.jpg
[[0. 1. 0. 0. 0. 0. 0.]]
Giá trị dự đoán:  10000
```



Saved successfully!