```
#10 loại mó ăn
import numpy as np
from keras_preprocessing import image
import cv2
import os
import tensorflow as tf
import time
from keras_preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)


training_set = train_datagen.flow_from_directory('/content/drive/MyDrive/AI/test4',
                                                 target_size = (64, 64),
                                                 batch_size = 12,
                                                 class_mode = 'categorical')
```

```
     Found 10 images belonging to 6 classes.
```

```
test_datagen = ImageDataGenerator(rescale = 1./255)
test_set = test_datagen.flow_from_directory('/content/drive/MyDrive/AI/train4',
                                            target_size = (64, 64),
                                            batch_size = 12,
                                            class_mode = 'categorical')
```

```
     Found 10 images belonging to 6 classes.
```

```
from keras.models import Sequential
from keras.layers import Dense, Dropout, Conv2D, MaxPooling2D, Flatten

model=tf.keras.models.Sequential()

# lớp CNN1
model.add(Conv2D(32,(3,3), activation='relu',input_shape=(200,200,3)))
model.add(MaxPooling2D(2,2))

# lớp CNN2
model.add(Conv2D(64,(3,3), activation='relu'))
model.add(MaxPooling2D(2,2))

# lớp CNN3
model.add(Conv2D(128,(3,3), activation='relu'))
model.add(MaxPooling2D(2,2))
```

```
# chuyển dữ liệu đầu ra của mạng CNN từ mảng 2 chiều về mảng 1 chiều
model.add(Flatten())
# lớp ẩn
model.add(Dense(512, activation=tf.nn.relu))
# lớp output
model.add(Dense(10, activation=tf.nn.softmax)) # 4 đầu ra


classes = ['10000vnd','1000vnd','2000vnd','5000vnd','500vnd','100000vnd','100','20000vnd','2(
print("Image Processing.......Compleated")
```

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

```
cnn = tf.keras.models.Sequential()
print("Building Neural Network.....")
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation='relu', input_shape=[64,
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
cnn.add(tf.keras.layers.Flatten())
cnn.add(tf.keras.layers.Dense(units=32, activation='relu'))
cnn.add(tf.keras.layers.Dense(units=64, activation='relu'))
cnn.add(tf.keras.layers.Dense(units=128, activation='relu'))
cnn.add(tf.keras.layers.Dense(units=256, activation='relu'))
cnn.add(tf.keras.layers.Dense(units=256, activation='relu'))
cnn.add(tf.keras.layers.Dense(units=6, activation='softmax'))
cnn.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
print("Training cnn")
cnn.fit(x = training_set, validation_data = test_set, epochs = 100,verbose=1)
cnn.save("model11.h5")
```

```
Epoch 37/100
1/1 [==============================] - 0s 180ms/step - loss: 0.5323 - accuracy: 0.900
Epoch 38/100
1/1 [==============================] - 0s 191ms/step - loss: 0.5474 - accuracy: 0.700
Epoch 39/100
1/1 [==============================] - 0s 174ms/step - loss: 0.5611 - accuracy: 0.800
Epoch 40/100
1/1 [==============================] - 0s 182ms/step - loss: 0.4884 - accuracy: 0.900
Epoch 41/100
1/1 [==============================] - 0s 185ms/step - loss: 0.3932 - accuracy: 0.900
Epoch 42/100
1/1 [==============================] - 0s 180ms/step - loss: 0.4772 - accuracy: 0.800
Epoch 43/100
1/1 [==============================] - 0s 195ms/step - loss: 0.3610 - accuracy: 0.900
Epoch 44/100
1/1 [==============================] - 0s 184ms/step - loss: 0.3432 - accuracy: 0.900
Epoch 45/100
1/1 [==============================] - 0s 189ms/step - loss: 0.2902 - accuracy: 0.900
Epoch 46/100
1/1 [==============================] - 0s 190ms/step - loss: 0.2903 - accuracy: 0.900
Epoch 47/100
1/1 [==============================] - 0s 193ms/step - loss: 0.5480 - accuracy: 0.800
```

```
Epoch 48/100
1/1 [==============================] - 0s 178ms/step - loss: 0.5263 - accuracy: 0.700
Epoch 49/100
1/1 [==============================] - 0s 183ms/step - loss: 0.2710 - accuracy: 0.900
Epoch 50/100
1/1 [==============================] - 0s 182ms/step - loss: 0.4665 - accuracy: 0.700
Epoch 51/100
1/1 [==============================] - 0s 196ms/step - loss: 0.2986 - accuracy: 0.800
Epoch 52/100
1/1 [==============================] - 0s 184ms/step - loss: 0.5493 - accuracy: 0.700
Epoch 53/100
1/1 [==============================] - 0s 183ms/step - loss: 0.3716 - accuracy: 0.800
Epoch 54/100
1/1 [==============================] - 0s 187ms/step - loss: 0.2256 - accuracy: 0.900
Epoch 55/100
1/1 [==============================] - 0s 181ms/step - loss: 0.2568 - accuracy: 0.900
Epoch 56/100
1/1 [==============================] - 0s 176ms/step - loss: 0.3611 - accuracy: 0.800
Epoch 57/100
1/1 [==============================] - 0s 209ms/step - loss: 0.4052 - accuracy: 0.800
Epoch 58/100
1/1 [==============================] - 0s 196ms/step - loss: 0.2991 - accuracy: 0.900
Epoch 59/100
1/1 [==============================] - 0s 200ms/step - loss: 0.2510 - accuracy: 0.900
Epoch 60/100
1/1 [==============================] - 0s 174ms/step - loss: 0.1835 - accuracy: 0.900
Epoch 61/100
1/1 [==============================] - 0s 186ms/step - loss: 0.3225 - accuracy: 0.900
Epoch 62/100
1/1 [==============================] - 0s 206ms/step - loss: 0.2770 - accuracy: 0.900
Epoch 63/100
1/1 [==============================] - 0s 186ms/step - loss: 0.4272 - accuracy: 0.800
Epoch 64/100
1/1 [==============================] - 0s 207ms/step - loss: 0.1847 - accuracy: 0.900
Epoch 65/100
```

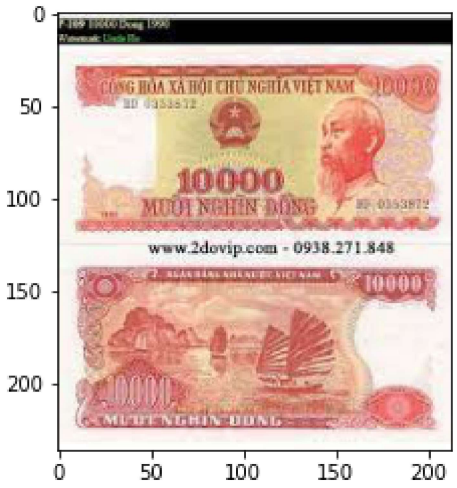
```
from google.colab import files
from keras.preprocessing import image
%matplotlib inline
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

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=(200,200))
  x=image.img_to_array(img)
  x=np.expand_dims(x,axis=0)
```

```
images=np.vstack([x])
y_predict = model.predict(images,batch_size=10000)
print(y_predict)
print('Giá trị dự đoán: ', classes[np.argmax(y_predict)])
```

Choose Files download (11).jpg

- **download (11).jpg**(image/jpeg) - 11172 bytes, last modified: 5/19/2022 - 100% done

```
Saving download (11).jpg to download (11).jpg
[[1.0000000e+00 2.3960486e-38 1.4996754e-11 2.3280168e-37 1.6170245e-36
  1.7494112e-32 2.6969743e-22 0.0000000e+00 0.0000000e+00 2.3678870e-18]]
Giá trị dự đoán:  10000vnd
```



✓ 8 giây hoàn thành lúc 14:05