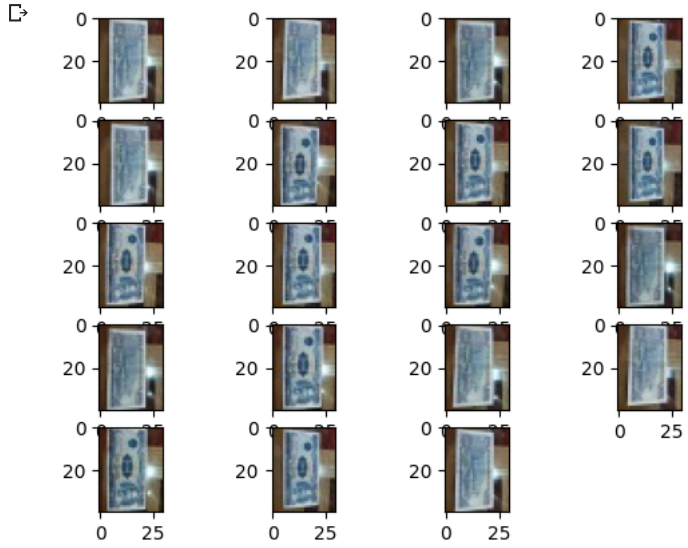
```
# kiểm tra dữ liệu
import matplotlib.pyplot as plt
from matplotlib.image import imread
folder = '/content/drive/MyDrive/AI_baocao/images/images_tienVN/'
for i in range(19):
  plt.subplot(5,4,i+1)
  filename = folder + 'tien5k_'+str(i+1)+'.jpg'
  img = plt.imread(filename)
  plt.imshow(img)
plt.show()
```



```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remou
```

```
import matplotlib.pyplot as plt
from matplotlib.image import imread
folder = '/content/drive/MyDrive/AI_baocao/images/images_tienVN/'
for i in range(14):
  plt.subplot(5,3,i+1)
  filename = folder + 'tien100k_'+str(i+1)+'.jpg'
  img = plt.imread(filename)
  plt.imshow(img)
plt.show()
```

Gan nhan

```
from os import listdir
```

```
from os.path import isdir
from numpy import asarray
from numpy import save
from keras.utils import load_img, img_to_array
folder = '/content/drive/MyDrive/AI_baocao/images/images_tienVN/'
photos, labels = list(), list()
for file in listdir(folder):
  output = 0.0
  if file.startswith('tien5k'):
    output = 1
  if file.startswith('tien10k'):
    output = 2
  if file.startswith('tien20k'):
    output = 3
  if file.startswith('tien50k'):
    output = 4
  if file.startswith('tien100k'):
    output = 5
  if file.startswith('tien200k'):
    output = 6
  if file.startswith('tien500k'):
    output = 7
  img =load_img(folder+file,target_size=(40,30))
# print(img)
  photo = img_to_array(img)
  photos.append(photo)
  labels.append(output)
x_train = photos = asarray(photos)
y_train = labels = asarray(labels)
print(photos.shape,labels.shape)
# # save('/content/drive/MyDrive/AI_baocao/npy_files/predict_tienVN_photos.npy', photos)
# save('/content/drive/MyDrive/AI_baocao/npy_files/predict_tienVN_labels.npy', labels)
```

```
    (140, 40, 30, 3) (140,)
```

```
import numpy as np
# x_train = np.load('/content/drive/MyDrive/AI_baocao/npy_files/predict_food_photos.npy')
# y_train = np.load('/content/drive/MyDrive/AI_baocao/npy_files/predict_food_labels.npy')
print(x_train.shape)
print(y_train.shape)
```

```
    (140, 40, 30, 3)
    (140,)
```

```
x_train = x_train.astype('float32')/255
from keras.utils import to_categorical
y_train = to_categorical(y_train,10)
```

Tao lop tich chap - CNN

```
from keras.models import Sequential
from keras.layers import Dense,Dropout,Flatten,Conv2D,MaxPooling2D,Normalization,LeakyReLU
from keras.optimizers import Adam

#32 lan tich chap
model = Sequential()
model.add(Conv2D(32,kernel_size = (3,3),activation = 'relu',input_shape=(40,30,3),padding='Same'))
model.add(MaxPooling2D((2,2),padding='same'))
model.add(Dropout(0.25))

#64 lan tich chap
model.add(Conv2D(64,(3,3),activation ='relu',padding ='same'))
model.add(MaxPooling2D((2,2),padding='same'))
model.add(Dropout(0.25))

#128 lan tich chap
model.add(Conv2D(128,(3,3),activation ='relu',padding ='same'))
model.add(MaxPooling2D((2,2),padding='same'))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(128,activation = 'relu'))
model.add(Dropout(0.25))
model.add(Dense(10,activation='softmax'))
```

```
from keras.losses import categorical_crossentropy
model.compile(loss = categorical_crossentropy,optimizer = Adam(),metrics=['accuracy'])
model.summary()
train = model.fit(x_train,y_train,batch_size=250,epochs = 100,verbose = 1)
```

```
1/1 [==============================] - 0s 18ms/step - loss: 1.3872 - accuracy: 0.4500
Epoch 19/100
1/1 [==============================] - 0s 17ms/step - loss: 1.2623 - accuracy: 0.5214
Epoch 20/100
1/1 [==============================] - 0s 17ms/step - loss: 1.2212 - accuracy: 0.5357
Epoch 21/100
1/1 [==============================] - 0s 18ms/step - loss: 1.1739 - accuracy: 0.5000
Epoch 22/100
1/1 [==============================] - 0s 18ms/step - loss: 1.0531 - accuracy: 0.6071
Epoch 23/100
1/1 [==============================] - 0s 18ms/step - loss: 1.0732 - accuracy: 0.5643
Epoch 24/100
1/1 [==============================] - 0s 17ms/step - loss: 0.9494 - accuracy: 0.6000
Epoch 25/100
1/1 [==============================] - 0s 18ms/step - loss: 0.9502 - accuracy: 0.6214
Epoch 26/100
1/1 [==============================] - 0s 18ms/step - loss: 0.8700 - accuracy: 0.6571
Epoch 27/100
1/1 [==============================] - 0s 19ms/step - loss: 0.8618 - accuracy: 0.6500
Epoch 28/100
1/1 [==============================] - 0s 18ms/step - loss: 0.7288 - accuracy: 0.6500
Epoch 29/100
1/1 [==============================] - 0s 17ms/step - loss: 0.7101 - accuracy: 0.6714
Epoch 30/100
1/1 [==============================] - 0s 17ms/step - loss: 0.6901 - accuracy: 0.7357
Epoch 31/100
1/1 [==============================] - 0s 17ms/step - loss: 0.5993 - accuracy: 0.7929
Epoch 32/100
1/1 [==============================] - 0s 17ms/step - loss: 0.6602 - accuracy: 0.7643
Epoch 33/100
1/1 [==============================] - 0s 17ms/step - loss: 0.6214 - accuracy: 0.7643
Epoch 34/100
1/1 [==============================] - 0s 16ms/step - loss: 0.5987 - accuracy: 0.7786
Epoch 35/100
1/1 [==============================] - 0s 17ms/step - loss: 0.5749 - accuracy: 0.7500
Epoch 36/100
1/1 [==============================] - 0s 17ms/step - loss: 0.4813 - accuracy: 0.8214
Epoch 37/100
1/1 [==============================] - 0s 20ms/step - loss: 0.4454 - accuracy: 0.8429
Epoch 38/100
1/1 [==============================] - 0s 19ms/step - loss: 0.5223 - accuracy: 0.7714
Epoch 39/100
1/1 [==============================] - 0s 21ms/step - loss: 0.4514 - accuracy: 0.8286
Epoch 40/100
1/1 [==============================] - 0s 18ms/step - loss: 0.4710 - accuracy: 0.7857
Epoch 41/100
1/1 [==============================] - 0s 17ms/step - loss: 0.3710 - accuracy: 0.8929
Epoch 42/100
1/1 [==============================] - 0s 18ms/step - loss: 0.3514 - accuracy: 0.9071
Epoch 43/100
1/1 [==============================] - 0s 20ms/step - loss: 0.4265 - accuracy: 0.7857
Epoch 44/100
1/1 [==============================] - 0s 16ms/step - loss: 0.3515 - accuracy: 0.8571
Epoch 45/100
1/1 [==============================] - 0s 17ms/step - loss: 0.3521 - accuracy: 0.8786
Epoch 46/100
1/1 [==============================] - 0s 17ms/step - loss: 0.3050 - accuracy: 0.8786
Epoch 47/100
```

```
img = load_img("tien3.jpeg",target_size=(30,40))
plt.imshow(img)
img = img_to_array(img)
img=img.reshape(1,40,30,3)
img = img.astype('float32')
img =img/255
result  = np.argmax(model.predict(img),axis=1)
vat[result[0]]
```

```
1/1 [==============================] - 0s 22ms/step
'tien100k'
```