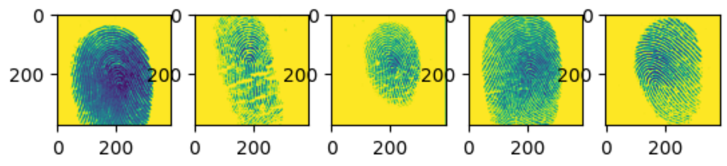
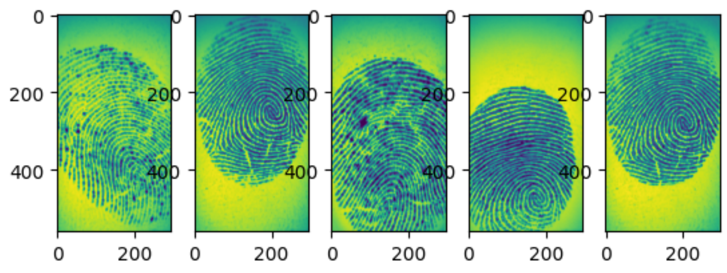
Kiem tra du lieu dau vao

```
import matplotlib.pyplot as plt
from matplotlib.image import imread
folder = '/content/drive/MyDrive/AI_baocao/images/images_fingerprint/'
for i in range(5):
  plt.subplot(1,5,i+1)
  filename = folder + 'db1_'+str(i+1)+'.jpg'
  img = plt.imread(filename)
  print(img.shape)
  plt.imshow(img)
plt.show()
```

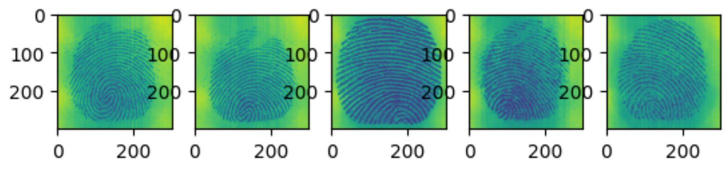
```
(374, 388)
(374, 388)
(374, 388)
(374, 388)
(374, 388)
```



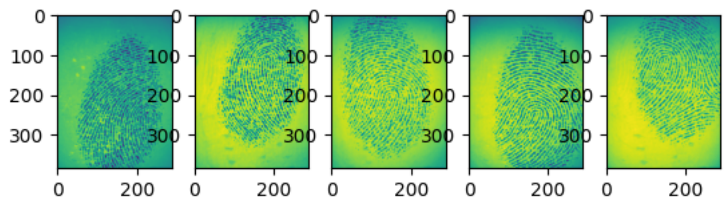
```
for i in range(5):
  plt.subplot(1,5,i+1)
  filename = folder + 'db2_'+str(i+1)+'.jpg'
  img = plt.imread(filename)
  plt.imshow(img)
plt.show()
```



```
for i in range(5):
  plt.subplot(1,5,i+1)
  filename = folder + 'db3_'+str(i+1)+'.jpg'
  img = plt.imread(filename)
  plt.imshow(img)
plt.show()
```



```
for i in range(5):
  plt.subplot(1,5,i+1)
  filename = folder + 'db4_'+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_fingerprint/'
photos, labels = list(), list()
for file in listdir(folder):
  output = 0.0
  if file.startswith('db1'):
    output = 1
  if file.startswith('db2'):
    output = 2
  if file.startswith('db3'):
    output = 3
  if file.startswith('db4'):
    output = 4
  img =load_img(folder+file,target_size=(374,388))
  photo = img_to_array(img)
  photos.append(photo)
  labels.append(output)
photos = asarray(photos)
labels = asarray(labels)
print(photos.shape,labels.shape)
save('predict_food_photos.npy', photos)
save('predict_food_labels.npy', labels)
```

```
(320, 374, 388, 3) (320,)
```

```
import numpy as np
x_train = np.load('predict_food_photos.npy')
y_train = np.load('predict_food_labels.npy')
print(x_train.shape)
print(y_train.shape)
```

```
(320, 374, 388, 3)
(320,)
```

```
x_train = x_train.astype('float32')/255
from keras.utils import to_categorical
y_train = to_categorical(y_train,11)
```

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=(374,388,3),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(Conv2D(256,(3,3),activation ='relu',padding ='same'))
model.add(MaxPooling2D((2,2),padding='same'))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(256,activation = 'relu'))
model.add(Dropout(0.25))
model.add(Dense(11,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=100,epochs = 30,verbose = 1)
```

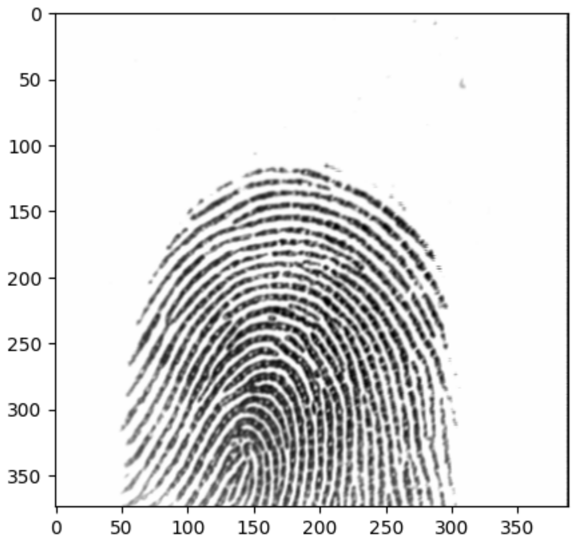
```
    Epoch 2/30
    4/4 [==============================] - 3s 696ms/step - loss: 2.0160 - accuracy: 0.3000
    Epoch 3/30
    4/4 [==============================] - 3s 690ms/step - loss: 1.5935 - accuracy: 0.2875
    Epoch 4/30
    4/4 [==============================] - 3s 675ms/step - loss: 1.4108 - accuracy: 0.2844
    Epoch 5/30
    4/4 [==============================] - 3s 672ms/step - loss: 1.4334 - accuracy: 0.2594
    Epoch 6/30
    4/4 [==============================] - 3s 669ms/step - loss: 1.3940 - accuracy: 0.2500
    Epoch 7/30
    4/4 [==============================] - 3s 675ms/step - loss: 1.2554 - accuracy: 0.3469
    Epoch 8/30
    4/4 [==============================] - 3s 681ms/step - loss: 1.1813 - accuracy: 0.4062
    Epoch 9/30
    4/4 [==============================] - 3s 665ms/step - loss: 1.0514 - accuracy: 0.5719
    Epoch 10/30
    4/4 [==============================] - 3s 666ms/step - loss: 0.9110 - accuracy: 0.5906
    Epoch 11/30
    4/4 [==============================] - 3s 667ms/step - loss: 0.8527 - accuracy: 0.6750
    Epoch 12/30
    4/4 [==============================] - 3s 684ms/step - loss: 0.6554 - accuracy: 0.7406
    Epoch 13/30
    4/4 [==============================] - 3s 670ms/step - loss: 0.5456 - accuracy: 0.7906
    Epoch 14/30
    4/4 [==============================] - 3s 668ms/step - loss: 0.5617 - accuracy: 0.7688
    Epoch 15/30
    4/4 [==============================] - 3s 666ms/step - loss: 0.4717 - accuracy: 0.8344
    Epoch 16/30
    4/4 [==============================] - 3s 671ms/step - loss: 0.3886 - accuracy: 0.8438
    Epoch 17/30
    4/4 [==============================] - 3s 682ms/step - loss: 0.3578 - accuracy: 0.8375
    Epoch 18/30
    4/4 [==============================] - 3s 666ms/step - loss: 0.3504 - accuracy: 0.8500
    Epoch 19/30
    4/4 [==============================] - 3s 665ms/step - loss: 0.3250 - accuracy: 0.8500
    Epoch 20/30
    4/4 [==============================] - 3s 699ms/step - loss: 0.3062 - accuracy: 0.8687
    Epoch 21/30
    4/4 [==============================] - 3s 676ms/step - loss: 0.2715 - accuracy: 0.8844
    Epoch 22/30
    4/4 [==============================] - 3s 678ms/step - loss: 0.2630 - accuracy: 0.8938
    Epoch 23/30
    4/4 [==============================] - 3s 661ms/step - loss: 0.2243 - accuracy: 0.9187
    Epoch 24/30
    4/4 [==============================] - 3s 663ms/step - loss: 0.2564 - accuracy: 0.9094
    Epoch 25/30
    4/4 [==============================] - 3s 674ms/step - loss: 0.2128 - accuracy: 0.9031
    Epoch 26/30
    4/4 [==============================] - 3s 677ms/step - loss: 0.2168 - accuracy: 0.9094
    Epoch 27/30
    4/4 [==============================] - 3s 665ms/step - loss: 0.2329 - accuracy: 0.9062
    Epoch 28/30
    4/4 [==============================] - 3s 670ms/step - loss: 0.2772 - accuracy: 0.8906
    Epoch 29/30
    4/4 [==============================] - 3s 680ms/step - loss: 0.1775 - accuracy: 0.9344
    Epoch 30/30
    4/4 [==============================] - 3s 669ms/step - loss: 0.1846 - accuracy: 0.9375
```

```
import matplotlib.pyplot as plt
from keras.utils import load_img
from keras.utils.image_utils import img_to_array
import numpy as np
vat = {1: 'db1 - ',2:'db2 - ',3:'db3 - ',4:'db4 - '}
img = load_img('fp1.jpeg',target_size=(374,388))
plt.imshow(img)
img = img_to_array(img)
img=img.reshape(1,374,388,3)
img = img.astype('float32')
img =img/255
result  = np.argmax(model.predict(img),axis=1)
vat[result[0]]
```

```
1/1 [==============================] - 0s 144ms/step
'db4 - '
```

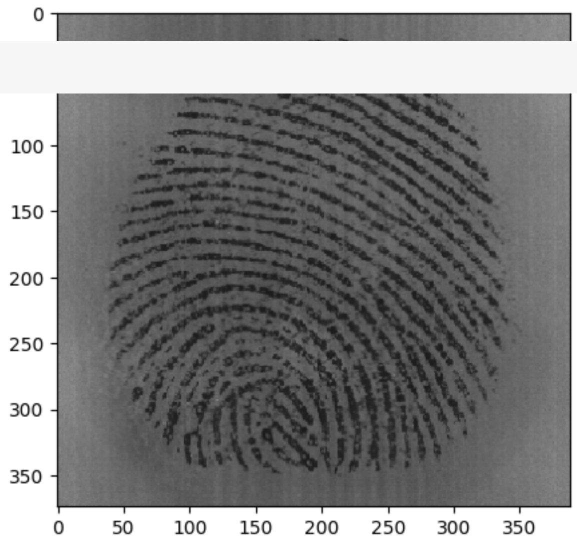
```
img = load_img('db1_18.jpg',target_size=(374,388))
plt.imshow(img)
img = img_to_array(img)
img=img.reshape(1,374,388,3)
img = img.astype('float32')
img =img/255
result  = np.argmax(model.predict(img),axis=1)
vat[result[0]]
```

```
1/1 [==============================] - 0s 74ms/step
'db1 - '
```



```
img = load_img('db3_16.jpg',target_size=(374,388))
plt.imshow(img)
img = img_to_array(img)
img=img.reshape(1,374,388,3)
img = img.astype('float32')
img =img/255
result  = np.argmax(model.predict(img),axis=1)
vat[result[0]]
```

```
1/1 [==============================] - 0s 19ms/step
'db3 - '
```


0
100
150
200
250
300
350
0
50
100
150
200
250
300
350