TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH
KHOA CƠ KHÍ CHẾ TẠO MÁY

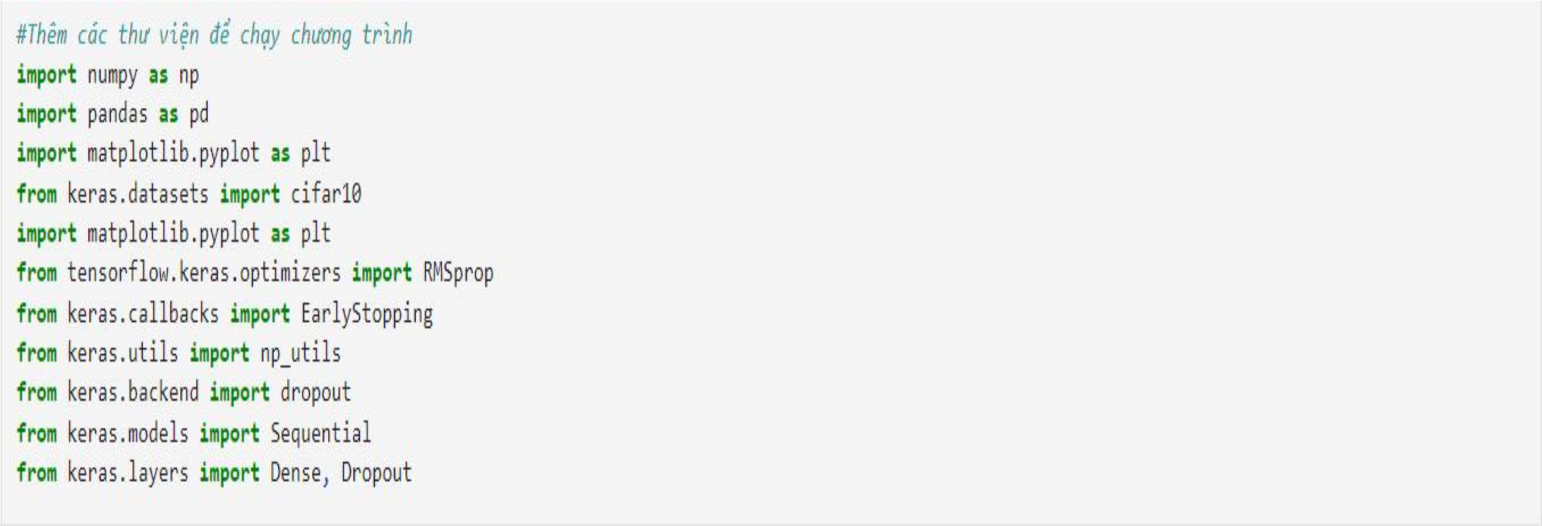
BÁO CÁO BÀI TẬP

Môn học: Trí tuệ nhân tạo
Họ và tên sinh viên: Lê Minh Trí
MSSV: 19146038
Lớp: Chiều thứ 2

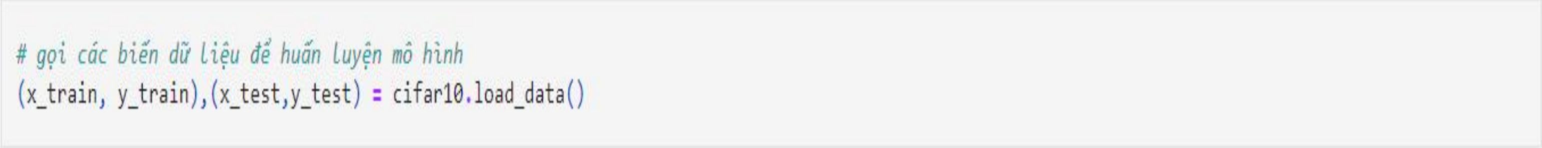
Giảng viên hướng dẫn: PGS.TS Nguyễn Trường Thịnh

Thành phố Hồ Chí Minh, ngày 23 tháng 05 năm 2021

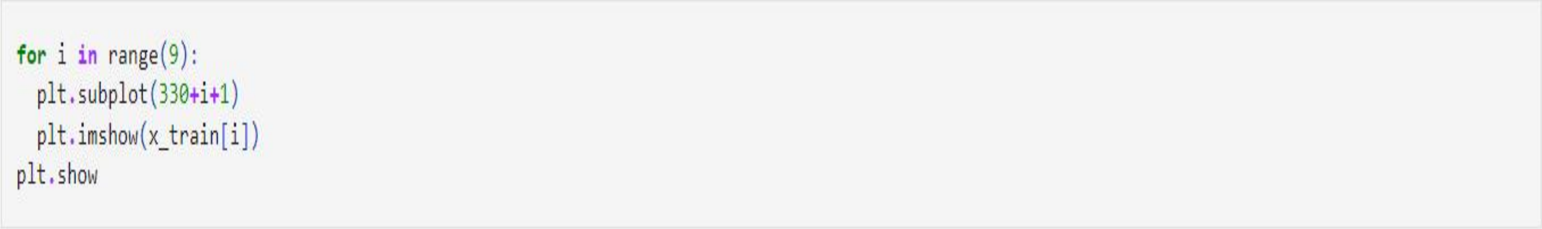
Bài 1: Cifar 10



```
#Thêm các thư viện để chạy chương trình
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from keras.datasets import cifar10
import matplotlib.pyplot as plt
from tensorflow.keras.optimizers import RMSprop
from keras.callbacks import EarlyStopping
from keras.utils import np_utils
from keras.backend import dropout
from keras.models import Sequential
from keras.layers import Dense, Dropout
```

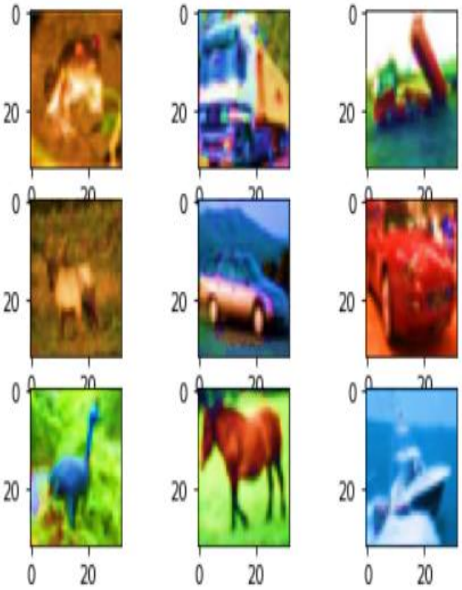


```
# gọi các biến dữ liệu để huấn luyện mô hình
(x_train, y_train),(x_test,y_test) = cifar10.load_data()
```



```
for i in range(9):
  plt.subplot(330+i+1)
  plt.imshow(x_train[i])
plt.show
```

```
<function matplotlib.pyplot.show>
```



In [46]:

```
# các biến định dạnh kích thước mô hình
x_train.shape , x_test .shape,  y_train.shape,  y_test.shape
```

Out[46]:

```
((50000, 32, 32, 3), (10000, 32, 32, 3), (50000, 1), (10000, 1))
```

In [47]:

```
x_train.shape
```

Out[47]:

```
(50000, 32, 32, 3)
```

In [48]:

```
y_train.shape
```

Out[48]:

```
(50000, 1)
```

In [49]:

```
x_test .shape
```

Out[49]:

```
(10000, 32, 32, 3)
```

In [50]:

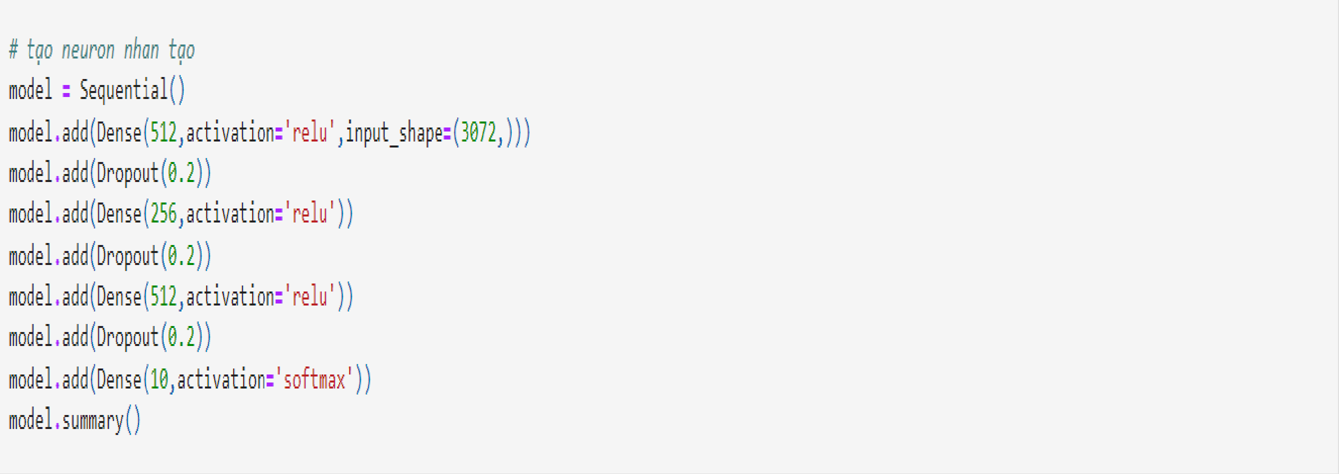
```
y_test.shape
```

Out[50]:

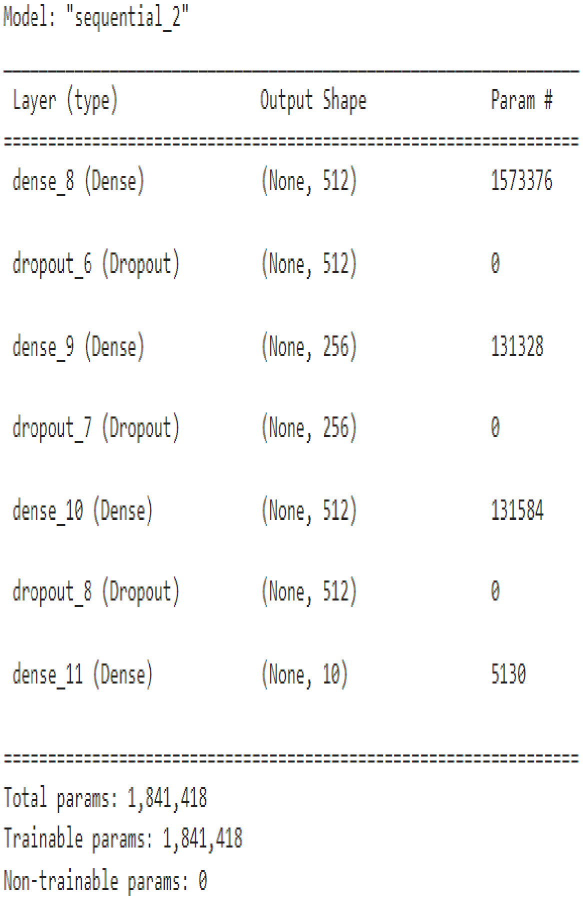
```
(10000, 1)
```

In [51]:

```
#  x_train , x_test  là mảng 4 chuyển sang mảng 2
x_train = x_train.reshape(50000 , 3072 )  #32*32*3
x_test = x_test.reshape(10000 ,  3072 )   #32*32*3

# chuyển hóa dữ liệu
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /=255
x_test /= 255

# chuyển y thành 10 lớp do 10 output
y_train =np_utils.to_categorical(y_train,10)
y_test = np_utils.to_categorical(y_test,10)
```



```
# tạo neuron nhan tạo
model = Sequential()
model.add(Dense(512,activation='relu',input_shape=(3072,)))
model.add(Dropout(0.2))
model.add(Dense(256,activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(512,activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(10,activation='softmax'))
model.summary()
```



```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense_8 (Dense)             (None, 512)               1573376   
                                                                 
 dropout_6 (Dropout)         (None, 512)               0         
                                                                 
 dense_9 (Dense)             (None, 256)               131328    
                                                                 
 dropout_7 (Dropout)         (None, 256)               0         
                                                                 
 dense_10 (Dense)            (None, 512)               131584    
                                                                 
 dropout_8 (Dropout)         (None, 512)               0         
                                                                 
 dense_11 (Dense)            (None, 10)                5130      
                                                                 
=================================================================
Total params: 1,841,418
Trainable params: 1,841,418
Non-trainable params: 0
```

# huấn luyện hóa mô hình

model.compile(loss='categorical_crossentropy',optimizer=RMSprop(), metrics=['accuracy'])

history = model.fit(x_train,y_train, batch_size=128, epochs=300 , verbose=1 , validation_split=0.2 , callbacks=[EarlyStopping(monitor='val_loss',patience=70)])

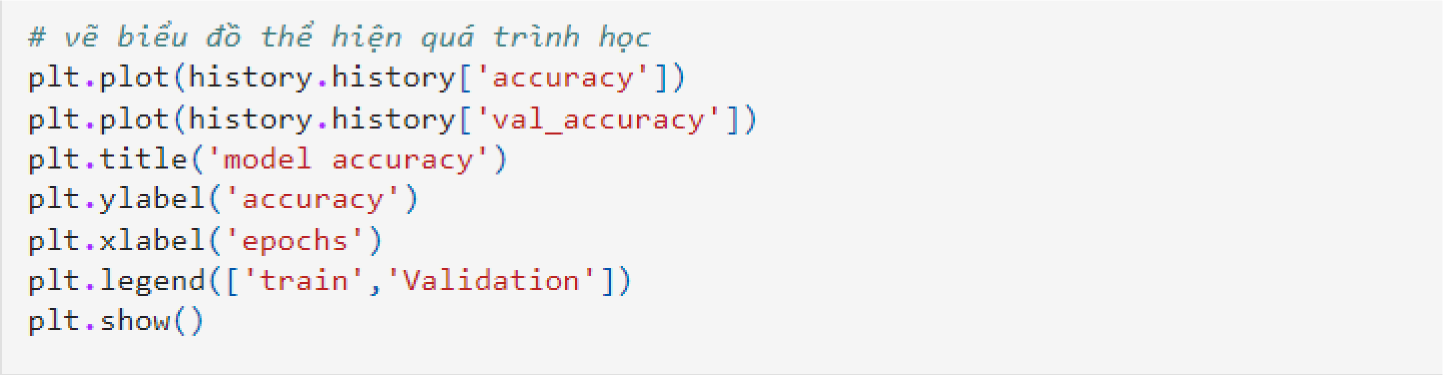
# verbose=1 hiên thị tiến trình huấn luyện

```
# lưu kết quả để kiểm tra
from tensorflow.keras.models import load_model
model.save('TRICifar10.h5')
load_model('TRICifar10.h5')
```

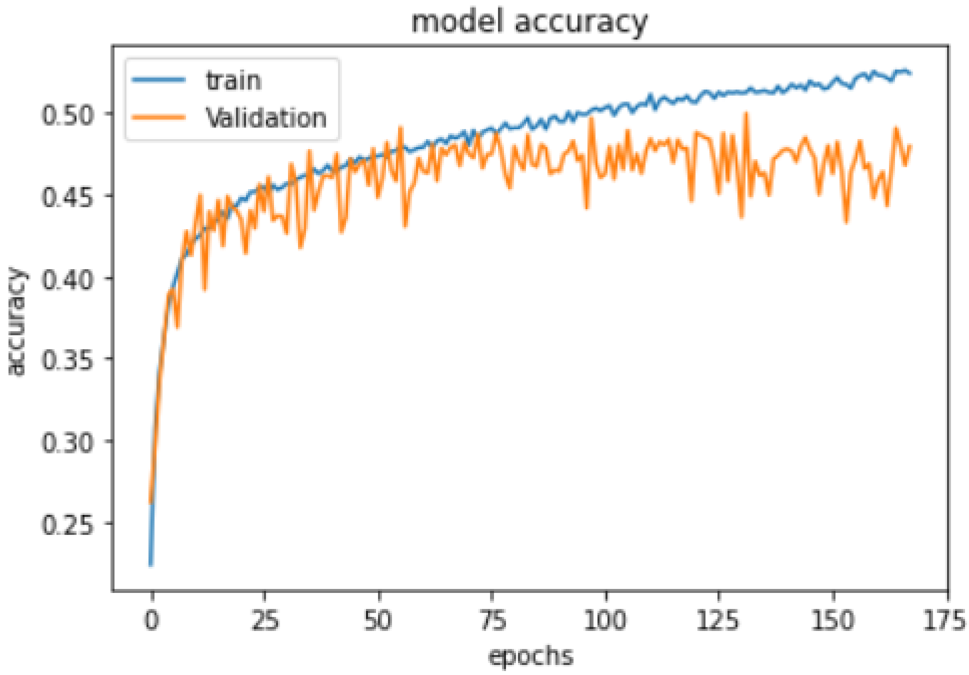
```
<keras.engine.sequential.Sequential at 0x7f9c27aea1d0>
```

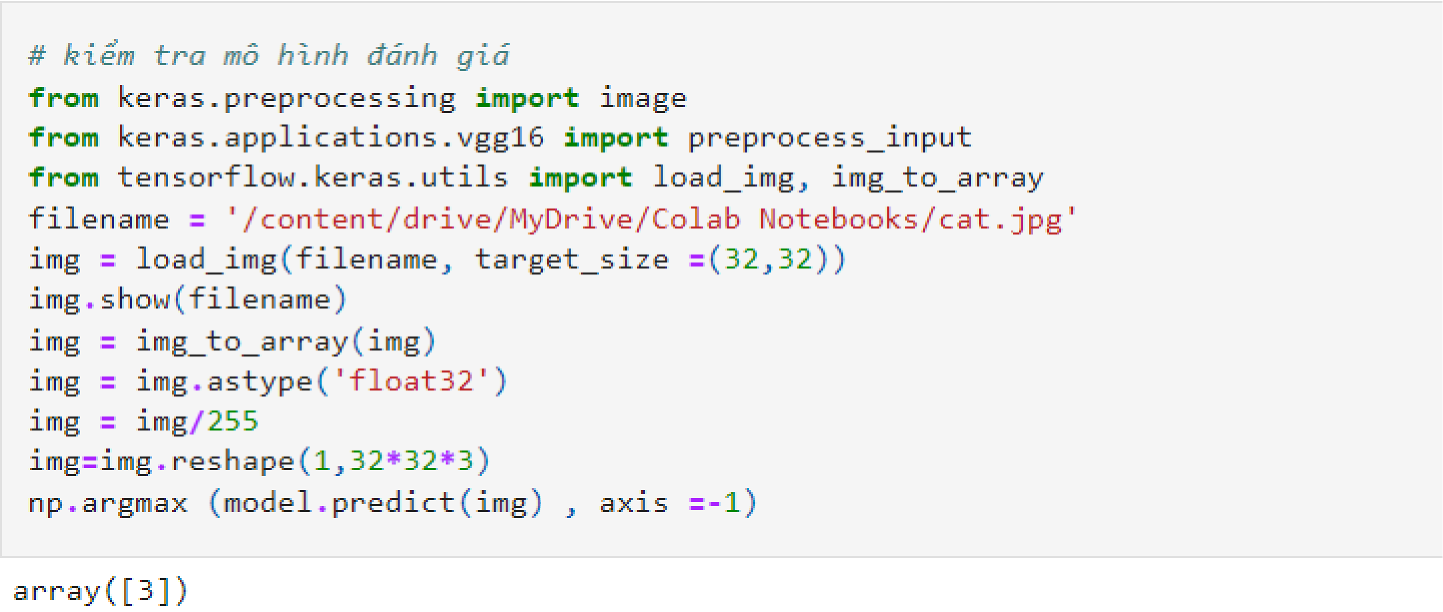
```
# kiểm tra đánh giá độ chính xác mô hình vừa huấn luyện
score = model.evaluate(x_test,y_test,verbose=0)
print('Sai số kiểm tra là: ',score[0])
print('Độ chính xác kiểm tra là: ',score[1])
```

```
Sai số kiểm tra là:  1.5234566926956177
Độ chính xác kiểm tra là:  0.4756999909877777
```



```python
# vẽ biểu đồ thể hiện quá trình học
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epochs')
plt.legend(['train','Validation'])
plt.show()
```

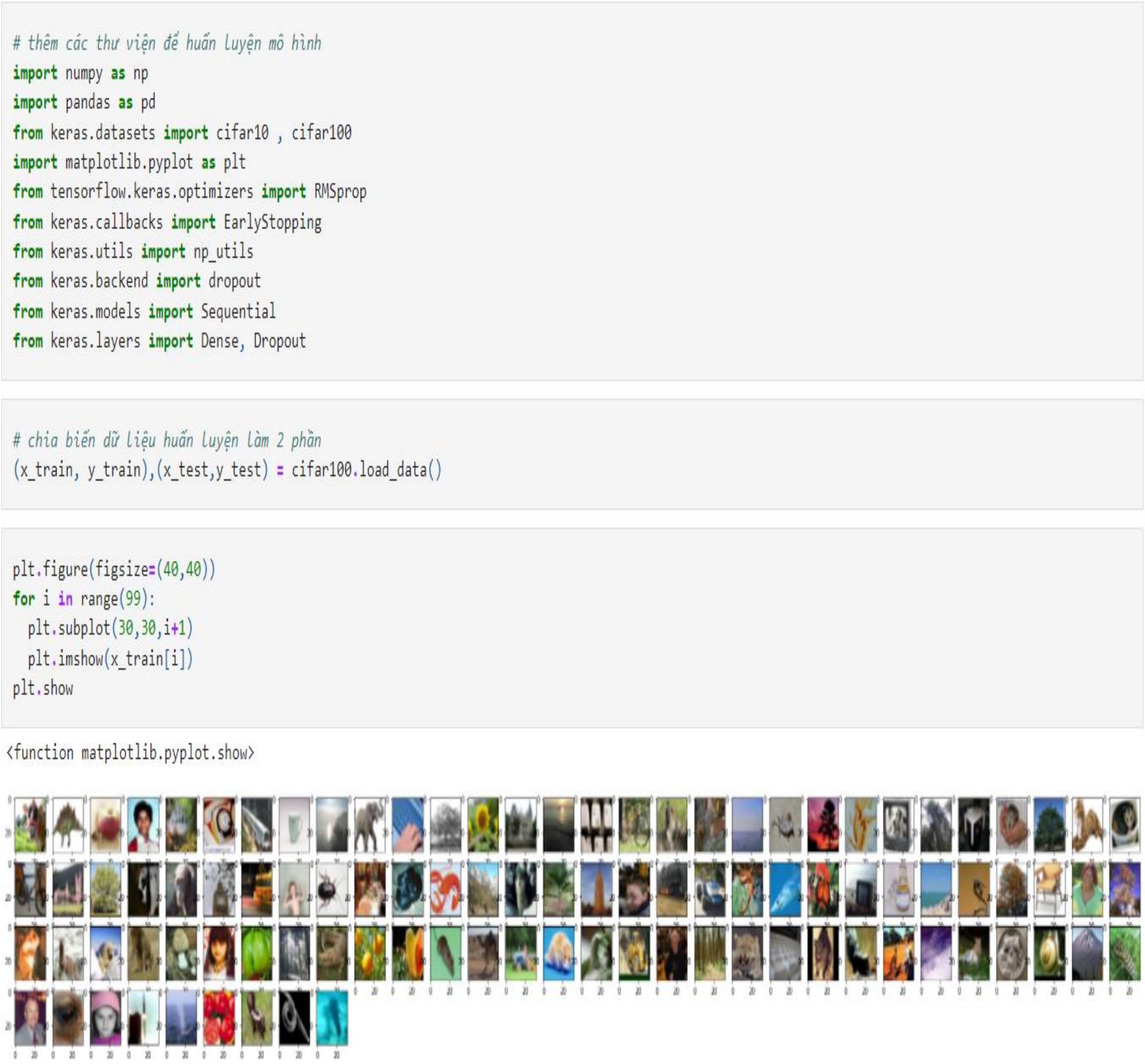




```python
# kiểm tra mô hình đánh giá
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input
from tensorflow.keras.utils import load_img, img_to_array
filename = '/content/drive/MyDrive/Colab Notebooks/cat.jpg'
img = load_img(filename, target_size =(32,32))
img.show(filename)
img = img_to_array(img)
img = img.astype('float32')
img = img/255
img=img.reshape(1,32*32*3)
np.argmax (model.predict(img) , axis =-1)
```

```
array([3])
```

Bài 2:Cifa100

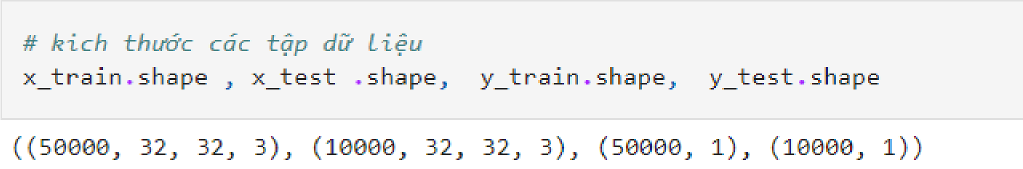


```
# thêm các thư viện để huấn luyện mô hình
import numpy as np
import pandas as pd
from keras.datasets import cifar10 , cifar100
import matplotlib.pyplot as plt
from tensorflow.keras.optimizers import RMSprop
from keras.callbacks import EarlyStopping
from keras.utils import np_utils
from keras.backend import dropout
from keras.models import Sequential
from keras.layers import Dense, Dropout
```

```
# chia biến dữ liệu huấn luyện làm 2 phần
(x_train, y_train),(x_test,y_test) = cifar100.load_data()
```

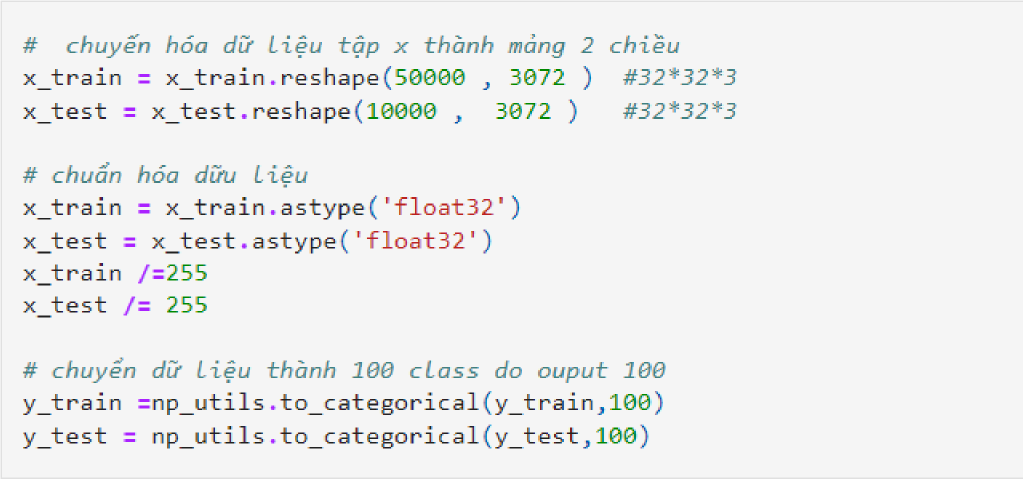
```
plt.figure(figsize=(40,40))
for i in range(99):
  plt.subplot(30,30,i+1)
  plt.imshow(x_train[i])
plt.show
```

```
<function matplotlib.pyplot.show>
```

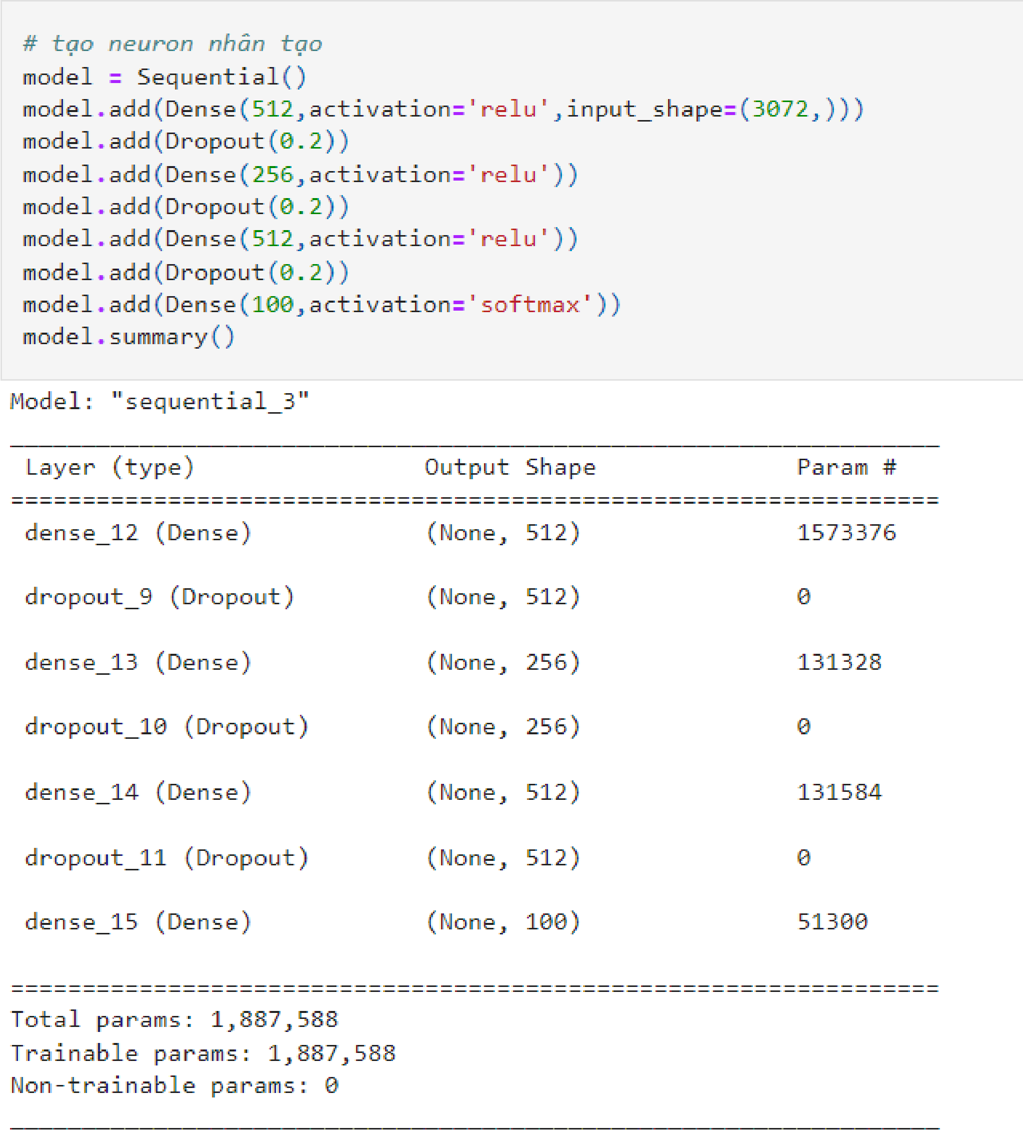


```
# kich thước các tập dữ liệu
x_train.shape , x_test .shape,  y_train.shape,  y_test.shape
```

```
((50000, 32, 32, 3), (10000, 32, 32, 3), (50000, 1), (10000, 1))
```



```
#  chuyến hóa dữ liệu tập x thành mảng 2 chiều
x_train = x_train.reshape(50000 , 3072 )  #32*32*3
x_test = x_test.reshape(10000 ,  3072 )   #32*32*3

# chuẩn hóa dữu liệu
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /=255
x_test /= 255

# chuyển dữ liệu thành 100 class do ouput 100
y_train =np_utils.to_categorical(y_train,100)
y_test = np_utils.to_categorical(y_test,100)
```



```
# tạo neuron nhân tạo
model = Sequential()
model.add(Dense(512,activation='relu',input_shape=(3072,)))
model.add(Dropout(0.2))
model.add(Dense(256,activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(512,activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(100,activation='softmax'))
model.summary()
```

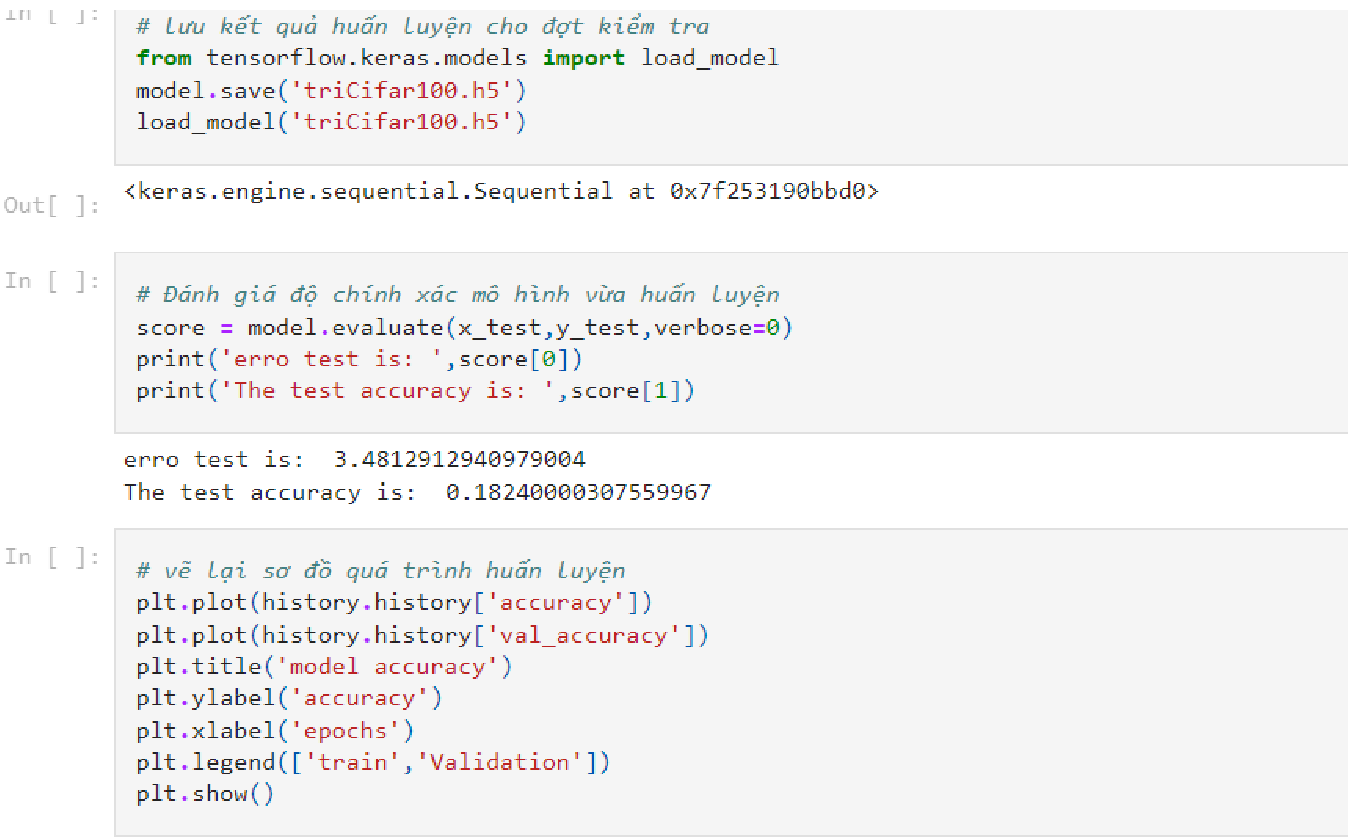
```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_12 (Dense)            (None, 512)               1573376

 dropout_9 (Dropout)         (None, 512)               0

 dense_13 (Dense)            (None, 256)               131328

 dropout_10 (Dropout)        (None, 256)               0

 dense_14 (Dense)            (None, 512)               131584

 dropout_11 (Dropout)        (None, 512)               0

 dense_15 (Dense)            (None, 100)               51300

=================================================================
Total params: 1,887,588
Trainable params: 1,887,588
Non-trainable params: 0
_________________________________________________________________
```

# Huấn Luyện Mô Hình Quá Trình Học 500

model.compile(loss='categorical_crossentropy',optimizer=RMSprop(), metrics=['accuracy'])

history = model.fit(x_train,y_train, batch_size=128, epochs=500 , verbose=1 , validation_split=0.2 , callbacks=[EarlyStopping(monitor='val_loss',patience=70)])

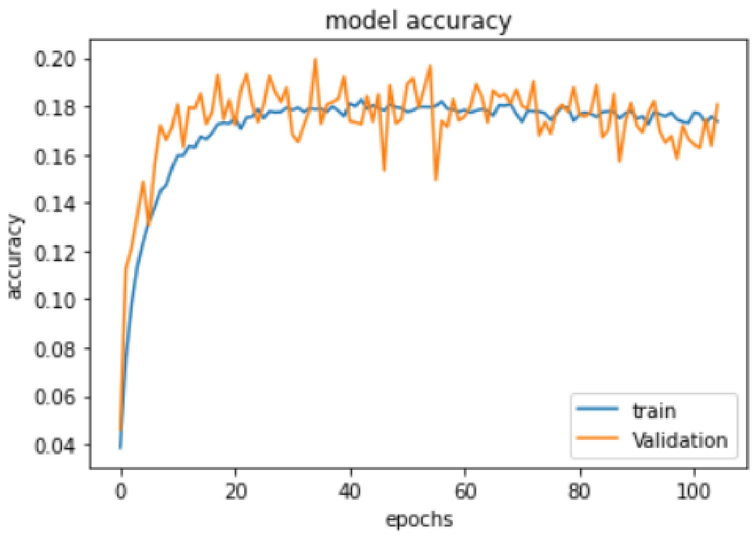
# Verbose=1 Hiển Thị Quá Trình Huấn Luyện

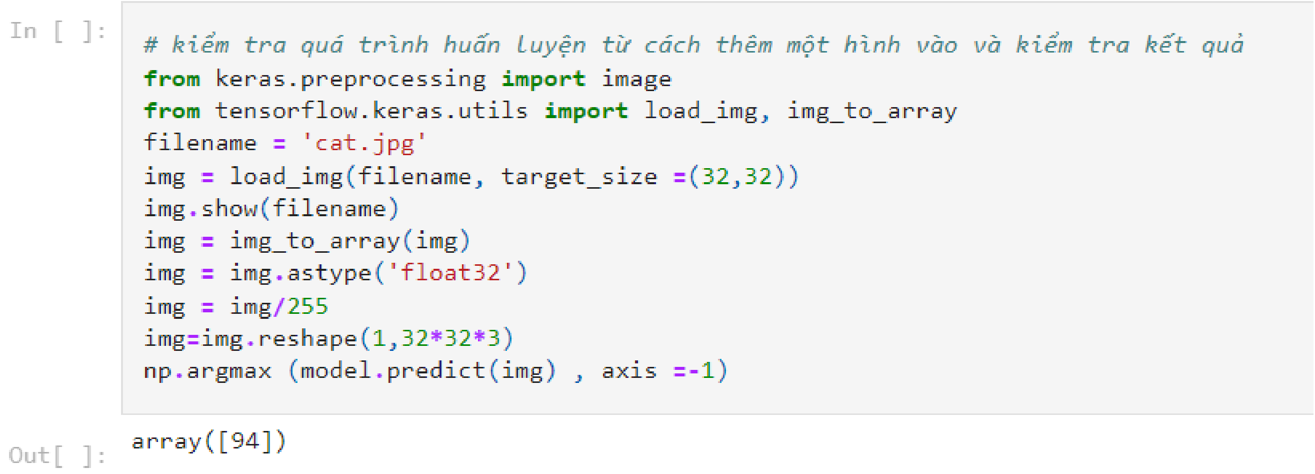


```
In [ ]:  # Lưu kết quả huấn luyện cho đợt kiểm tra
         from tensorflow.keras.models import load_model
         model.save('triCifar100.h5')
         load_model('triCifar100.h5')

Out[ ]:  <keras.engine.sequential.Sequential at 0x7f253190bbd0>
```

```
In [ ]:  # Đánh giá độ chính xác mô hình vừa huấn luyện
         score = model.evaluate(x_test,y_test,verbose=0)
         print('erro test is: ',score[0])
         print('The test accuracy is: ',score[1])

         erro test is:  3.4812912940979004
         The test accuracy is:  0.18240000307559967
```

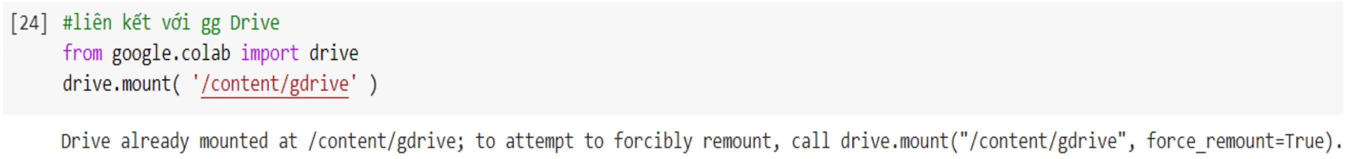
```
In [ ]:  # vẽ lại sơ đồ quá trình huấn luyện
         plt.plot(history.history['accuracy'])
         plt.plot(history.history['val_accuracy'])
         plt.title('model accuracy')
         plt.ylabel('accuracy')
         plt.xlabel('epochs')
         plt.legend(['train','Validation'])
         plt.show()
```



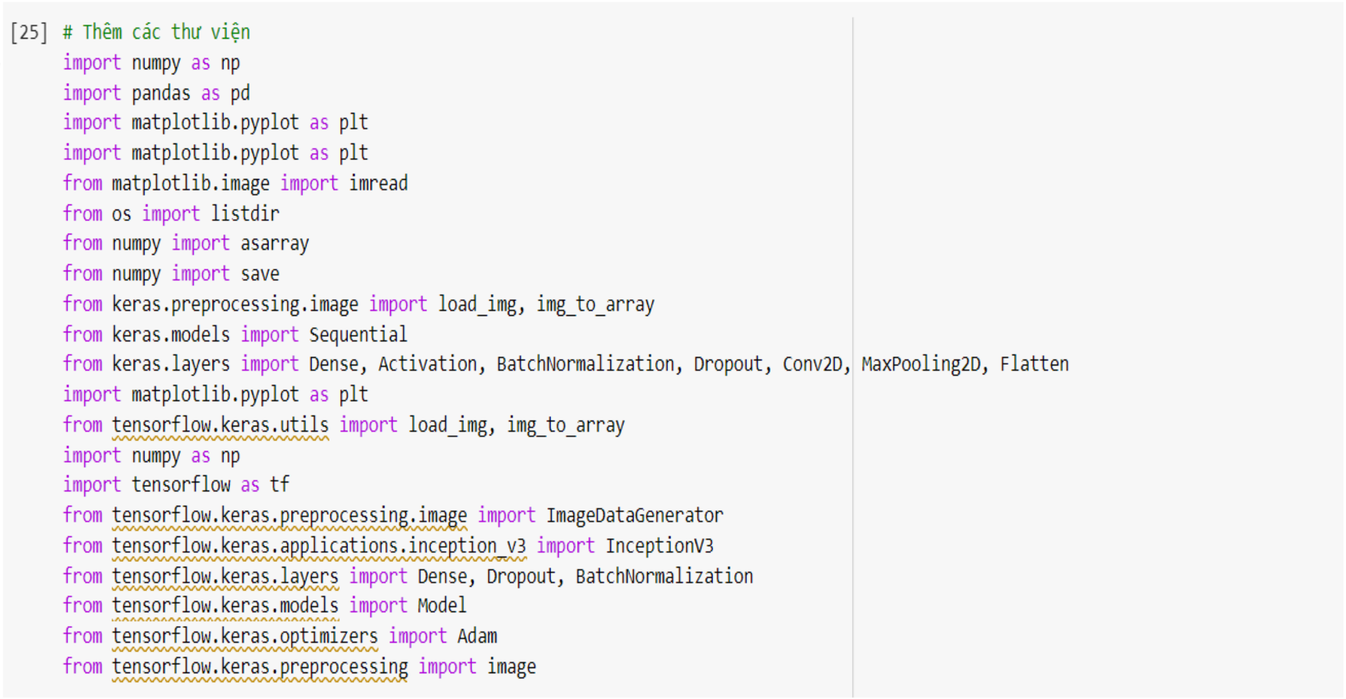


```
In [ ]:  # kiểm tra quá trình huấn luyện từ cách thêm một hình vào và kiểm tra kết quả
         from keras.preprocessing import image
         from tensorflow.keras.utils import load_img, img_to_array
         filename = 'cat.jpg'
         img = load_img(filename, target_size =(32,32))
         img.show(filename)
         img = img_to_array(img)
         img = img.astype('float32')
         img = img/255
         img=img.reshape(1,32*32*3)
         np.argmax (model.predict(img) , axis =-1)

Out[ ]:  array([94])
```

Bài 3: Nhân diện món ăn VN



```
[24] #liên kết với gg Drive
     from google.colab import drive
     drive.mount( '/content/gdrive' )

     Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).
```



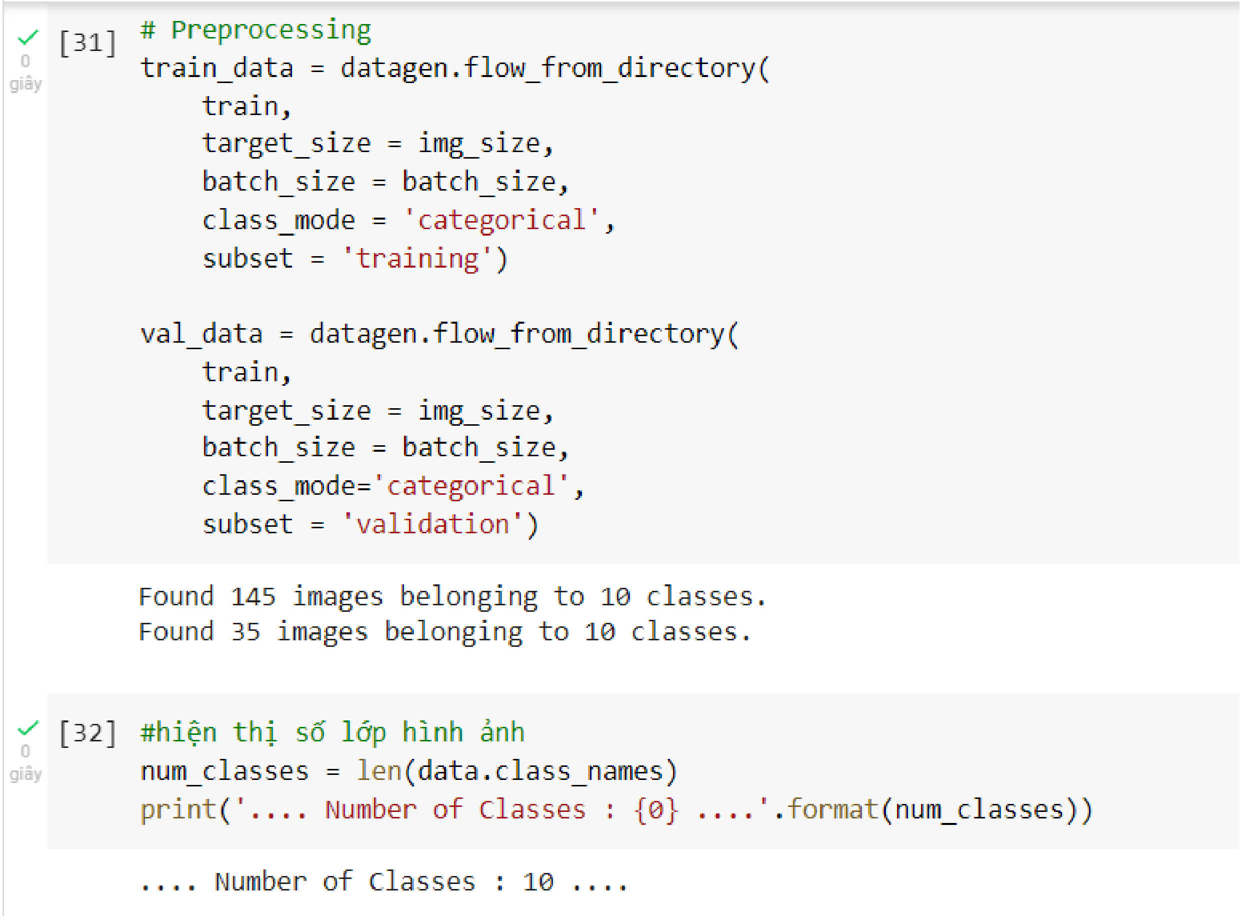
```
[25] # Thêm các thư viện
     import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import matplotlib.pyplot as plt
     from matplotlib.image import imread
     from os import listdir
     from numpy import asarray
     from numpy import save
     from keras.preprocessing.image import load_img, img_to_array
     from keras.models import Sequential
     from keras.layers import Dense, Activation, BatchNormalization, Dropout, Conv2D, MaxPooling2D, Flatten
     import matplotlib.pyplot as plt
     from tensorflow.keras.utils import load_img, img_to_array
     import numpy as np
     import tensorflow as tf
     from tensorflow.keras.preprocessing.image import ImageDataGenerator
     from tensorflow.keras.applications.inception_v3 import InceptionV3
     from tensorflow.keras.layers import Dense, Dropout, BatchNormalization
     from tensorflow.keras.models import Model
     from tensorflow.keras.optimizers import Adam
     from tensorflow.keras.preprocessing import image
```

```
[26] # Xem hinh anh tap training và val
     train = '/content/gdrive/MyDrive/Colab Notebooks/10monanvn/test'
     val_data = '/content/gdrive/MyDrive/Colab Notebooks/10monanvn/train'
     data = tf.keras.preprocessing.image_dataset_from_directory(train)
```

```
Found 180 files belonging to 10 classes.
```

```
[30] #Định dạng hình ảnh đầu vào
     datagen = ImageDataGenerator(
             rescale = 1./255,
             rotation_range=40,
             width_shift_range=0.2,
             height_shift_range=0.2,
             shear_range=0.2,
             zoom_range=0.2,
             horizontal_flip=True,
             fill_mode='nearest',
             validation_split = 0.2)
```

```
[29] #định dạng kích thước hình ảnh
     height = 228
     width = 228
     channels = 3
     batch_size = 32
     img_shape = (height, width, channels)
     img_size = (height, width)
```

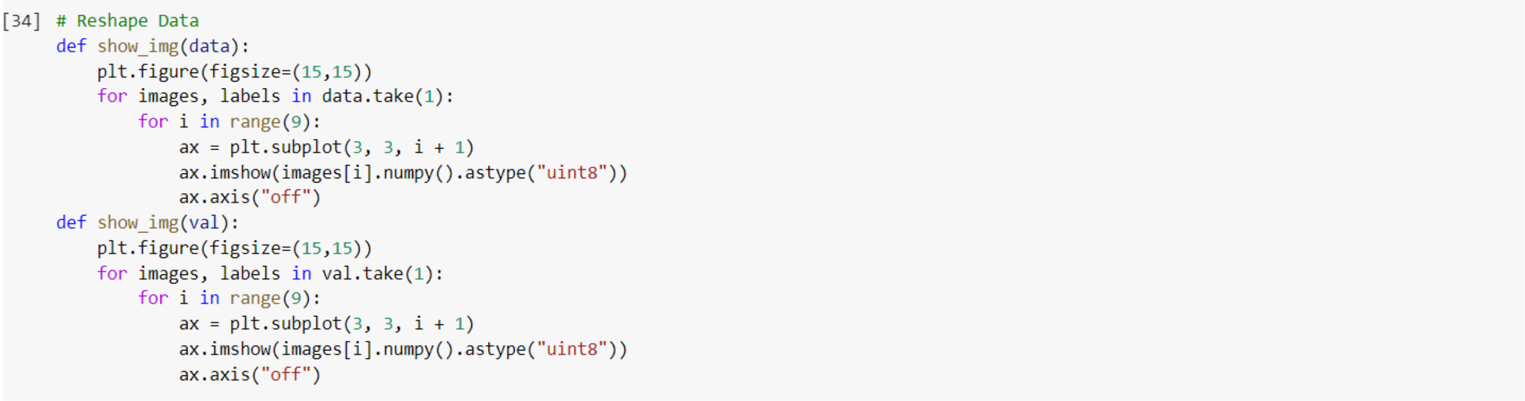


```
[31] # Preprocessing
     train_data = datagen.flow_from_directory(
         train,
         target_size = img_size,
         batch_size = batch_size,
         class_mode = 'categorical',
         subset = 'training')

     val_data = datagen.flow_from_directory(
         train,
         target_size = img_size,
         batch_size = batch_size,
         class_mode='categorical',
         subset = 'validation')
```

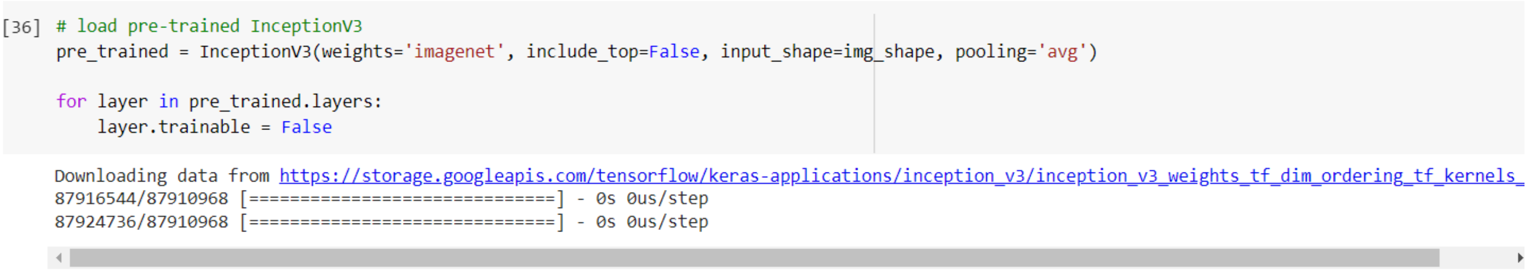
```
Found 145 images belonging to 10 classes.
Found 35 images belonging to 10 classes.
```

```
[32] #hiện thị số lớp hình ảnh
     num_classes = len(data.class_names)
     print('.... Number of Classes : {0} ....'.format(num_classes))
```

```
.... Number of Classes : 10 ....
```

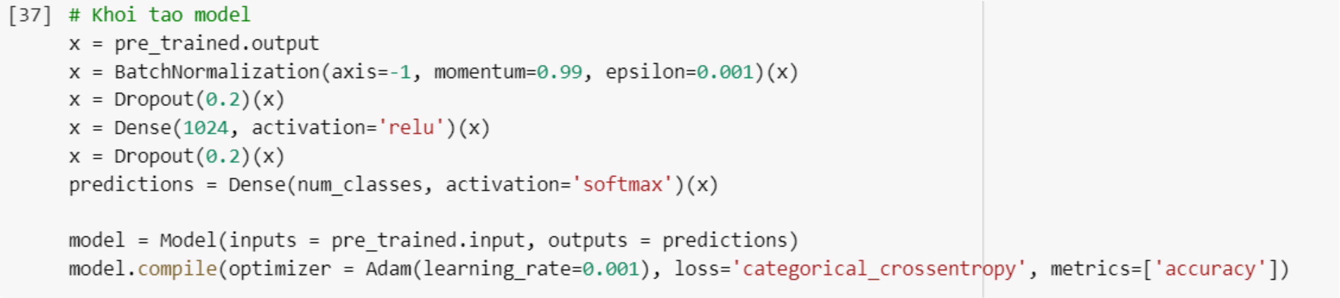


```
[34] # Reshape Data
     def show_img(data):
         plt.figure(figsize=(15,15))
         for images, labels in data.take(1):
             for i in range(9):
                 ax = plt.subplot(3, 3, i + 1)
                 ax.imshow(images[i].numpy().astype("uint8"))
                 ax.axis("off")
     def show_img(val):
         plt.figure(figsize=(15,15))
         for images, labels in val.take(1):
             for i in range(9):
                 ax = plt.subplot(3, 3, i + 1)
                 ax.imshow(images[i].numpy().astype("uint8"))
                 ax.axis("off")
```

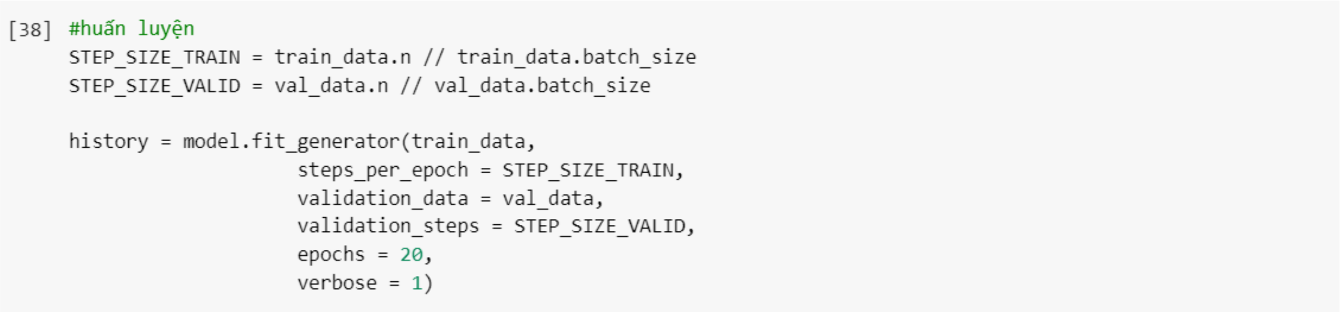


```
[36] # load pre-trained InceptionV3
     pre_trained = InceptionV3(weights='imagenet', include_top=False, input_shape=img_shape, pooling='avg')

     for layer in pre_trained.layers:
         layer.trainable = False
```

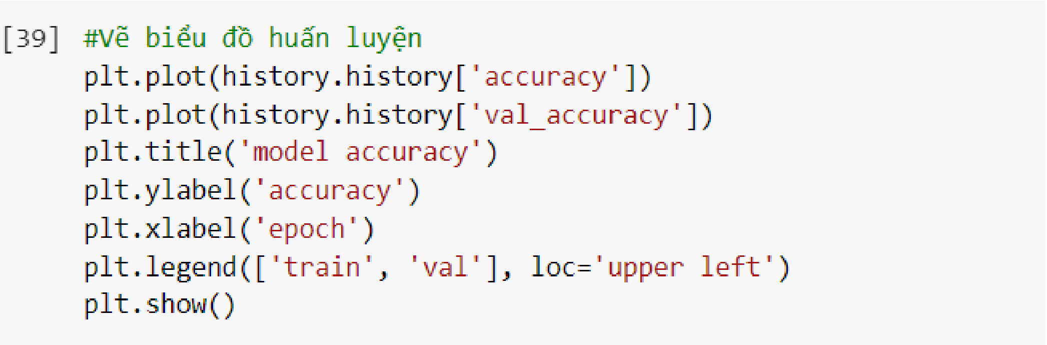
```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception_v3/inception_v3_weights_tf_dim_ordering_tf_kernels_
87916544/87910968 [==============================] - 0s 0us/step
87924736/87910968 [==============================] - 0s 0us/step
```



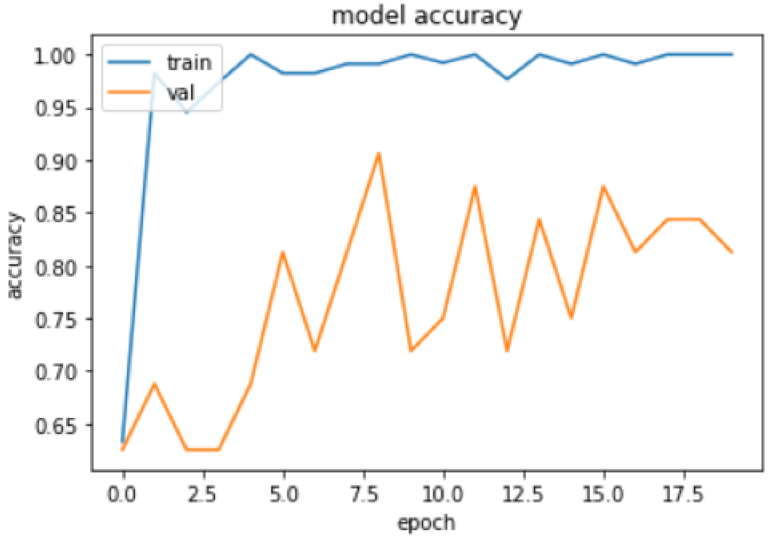
```
[37] # Khoi tao model
     x = pre_trained.output
     x = BatchNormalization(axis=-1, momentum=0.99, epsilon=0.001)(x)
     x = Dropout(0.2)(x)
     x = Dense(1024, activation='relu')(x)
     x = Dropout(0.2)(x)
     predictions = Dense(num_classes, activation='softmax')(x)

     model = Model(inputs = pre_trained.input, outputs = predictions)
     model.compile(optimizer = Adam(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy'])
```

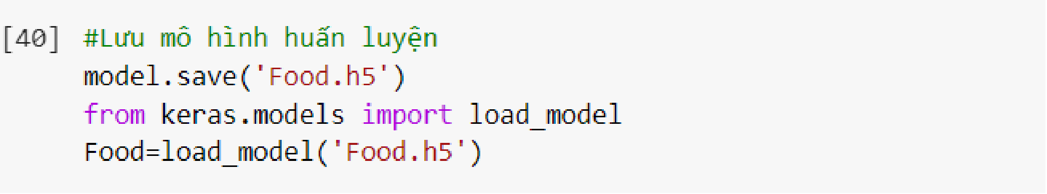


```
[38] #huấn luyện
     STEP_SIZE_TRAIN = train_data.n // train_data.batch_size
     STEP_SIZE_VALID = val_data.n // val_data.batch_size

     history = model.fit_generator(train_data,
                         steps_per_epoch = STEP_SIZE_TRAIN,
                         validation_data = val_data,
                         validation_steps = STEP_SIZE_VALID,
                         epochs = 20,
                         verbose = 1)
```

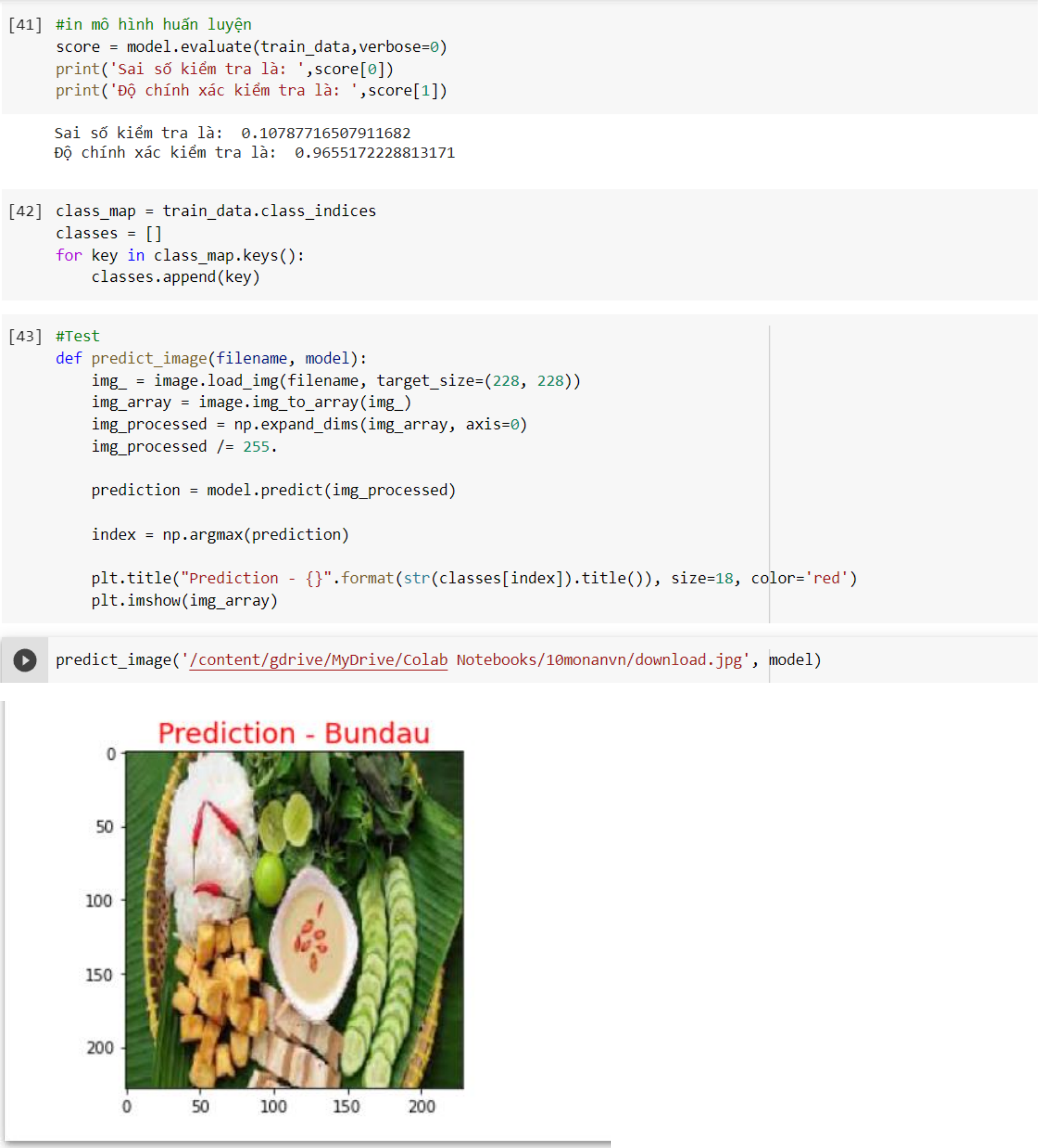


```
[39] #Vẽ biểu đồ huấn luyện
     plt.plot(history.history['accuracy'])
     plt.plot(history.history['val_accuracy'])
     plt.title('model accuracy')
     plt.ylabel('accuracy')
     plt.xlabel('epoch')
     plt.legend(['train', 'val'], loc='upper left')
     plt.show()
```





```
[40] #Lưu mô hình huấn luyện
     model.save('Food.h5')
     from keras.models import load_model
     Food=load_model('Food.h5')
```



```
[41] #in mô hình huấn luyện
     score = model.evaluate(train_data,verbose=0)
     print('Sai số kiểm tra là: ',score[0])
     print('Độ chính xác kiểm tra là: ',score[1])
```

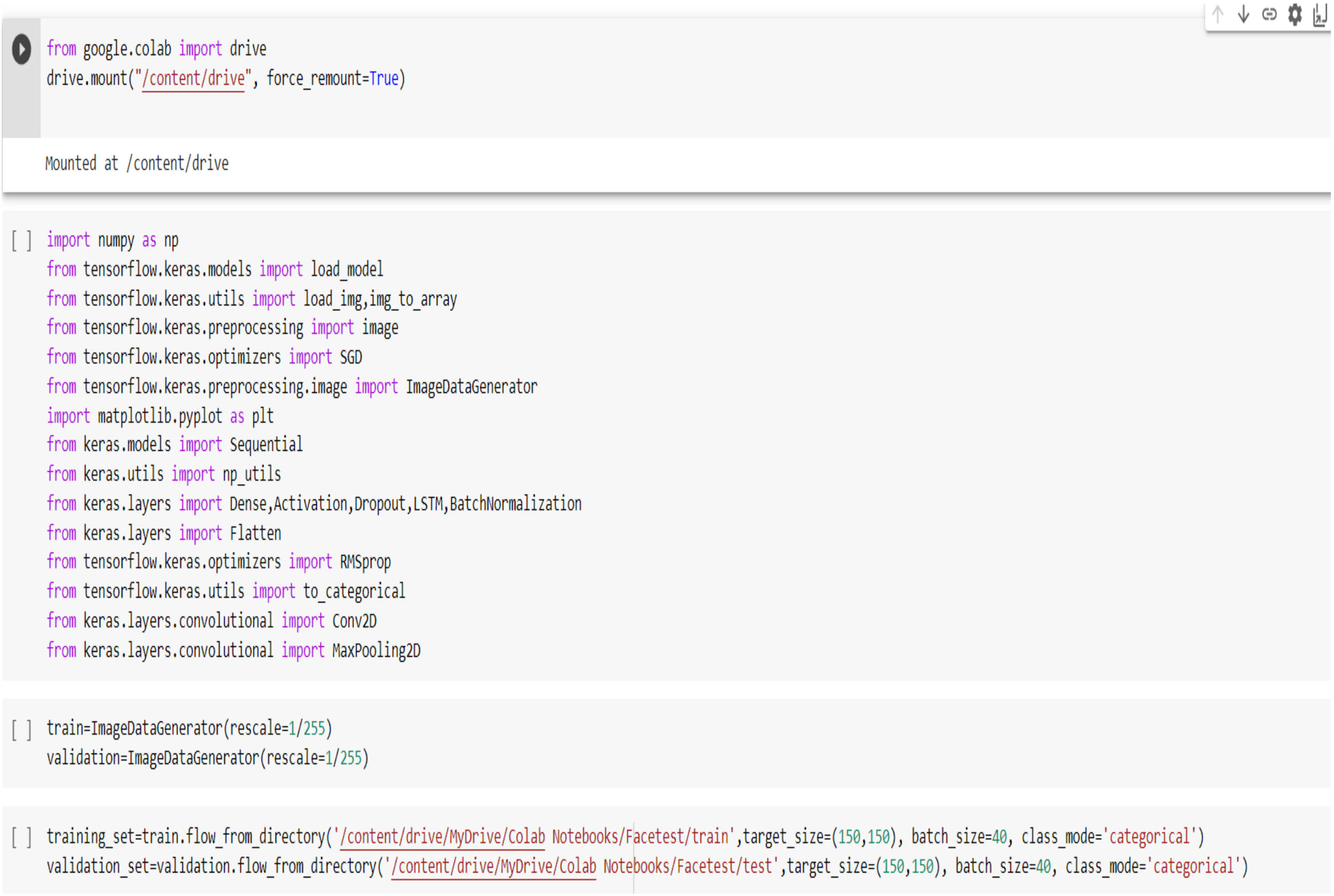
```
Sai số kiểm tra là:  0.10787716507911682
Độ chính xác kiểm tra là:  0.9655172228813171
```

```
[42] class_map = train_data.class_indices
     classes = []
     for key in class_map.keys():
         classes.append(key)
```

```
[43] #Test
     def predict_image(filename, model):
         img_ = image.load_img(filename, target_size=(228, 228))
         img_array = image.img_to_array(img_)
         img_processed = np.expand_dims(img_array, axis=0)
         img_processed /= 255.

         prediction = model.predict(img_processed)

         index = np.argmax(prediction)

         plt.title("Prediction - {}".format(str(classes[index]).title()), size=18, color='red')
         plt.imshow(img_array)
```

```
predict_image('/content/gdrive/MyDrive/Colab Notebooks/10monanvn/download.jpg', model)
```

Bài 4: Nhận diện khuôn mặt



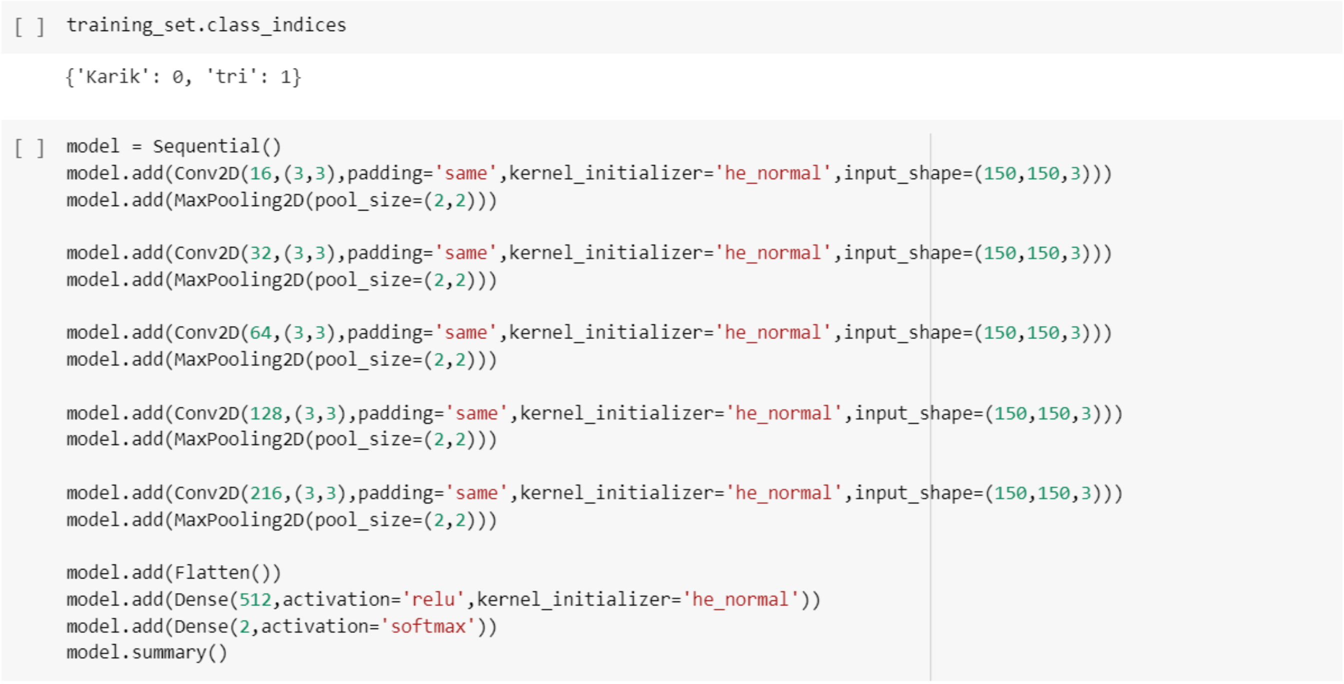
```
from google.colab import drive
drive.mount("/content/drive", force_remount=True)
```

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

```
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.utils import load_img,img_to_array
from tensorflow.keras.preprocessing import image
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.utils import np_utils
from keras.layers import Dense,Activation,Dropout,LSTM,BatchNormalization
from keras.layers import Flatten
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.utils import to_categorical
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
```

```
train=ImageDataGenerator(rescale=1/255)
validation=ImageDataGenerator(rescale=1/255)
```

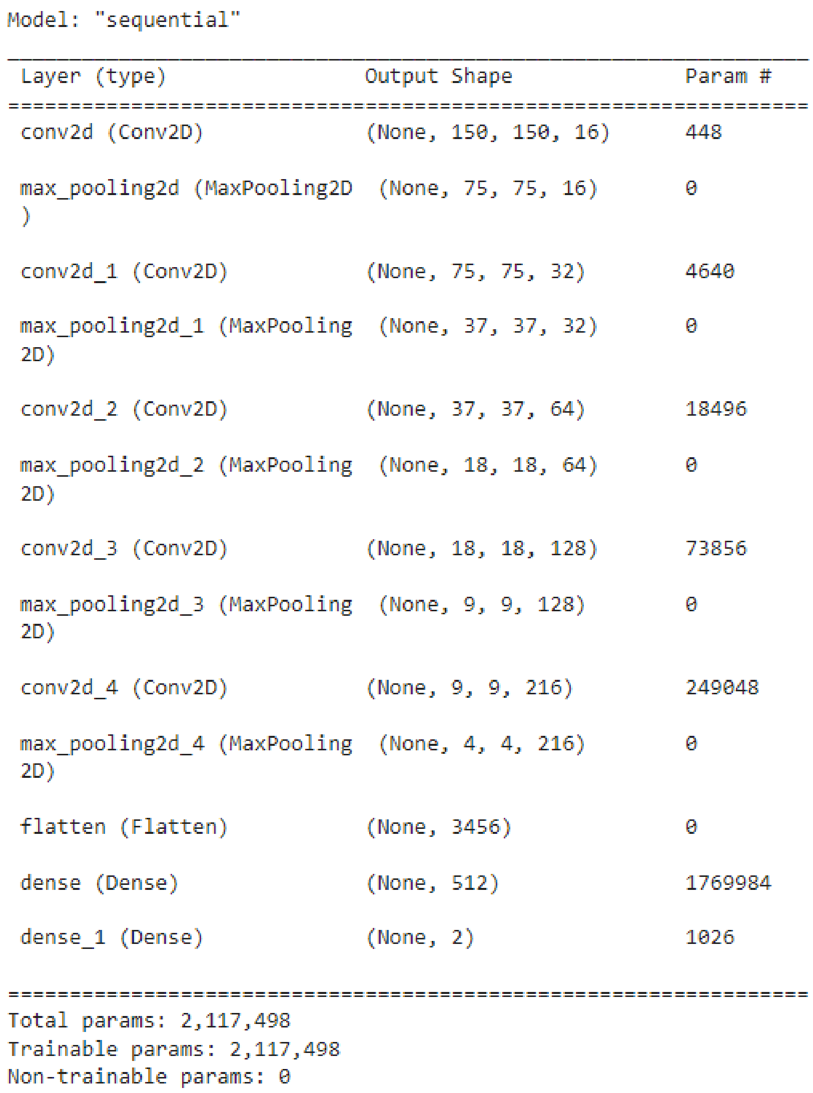
```
training_set=train.flow_from_directory('/content/drive/MyDrive/Colab Notebooks/Facetest/train',target_size=(150,150), batch_size=40, class_mode='categorical')
validation_set=validation.flow_from_directory('/content/drive/MyDrive/Colab Notebooks/Facetest/test',target_size=(150,150), batch_size=40, class_mode='categorical')
```



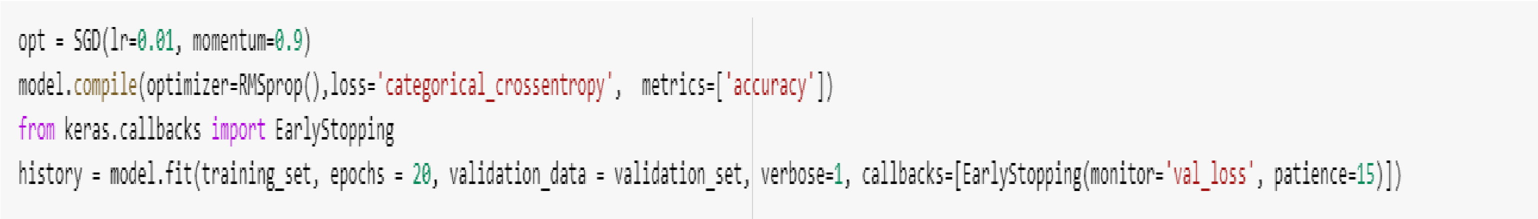
```
training_set.class_indices
```

```
{'Karik': 0, 'tri': 1}
```

```
model = Sequential()
model.add(Conv2D(16,(3,3),padding='same',kernel_initializer='he_normal',input_shape=(150,150,3)))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(32,(3,3),padding='same',kernel_initializer='he_normal',input_shape=(150,150,3)))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(64,(3,3),padding='same',kernel_initializer='he_normal',input_shape=(150,150,3)))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(128,(3,3),padding='same',kernel_initializer='he_normal',input_shape=(150,150,3)))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(216,(3,3),padding='same',kernel_initializer='he_normal',input_shape=(150,150,3)))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Flatten())
model.add(Dense(512,activation='relu',kernel_initializer='he_normal'))
model.add(Dense(2,activation='softmax'))
model.summary()
```



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 150, 150, 16)      448       
                                                                 
 max_pooling2d (MaxPooling2D  (None, 75, 75, 16)       0         
 )                                                               
                                                                 
 conv2d_1 (Conv2D)           (None, 75, 75, 32)        4640      
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 37, 37, 32)       0         
 2D)                                                             
                                                                 
 conv2d_2 (Conv2D)           (None, 37, 37, 64)        18496     
                                                                 
 max_pooling2d_2 (MaxPooling  (None, 18, 18, 64)       0         
 2D)                                                             
                                                                 
 conv2d_3 (Conv2D)           (None, 18, 18, 128)       73856     
                                                                 
 max_pooling2d_3 (MaxPooling  (None, 9, 9, 128)        0         
 2D)                                                             
                                                                 
 conv2d_4 (Conv2D)           (None, 9, 9, 216)         249048    
                                                                 
 max_pooling2d_4 (MaxPooling  (None, 4, 4, 216)        0         
 2D)                                                             
                                                                 
 flatten (Flatten)           (None, 3456)              0         
                                                                 
 dense (Dense)               (None, 512)               1769984   
                                                                 
 dense_1 (Dense)             (None, 2)                 1026      
                                                                 
=================================================================
Total params: 2,117,498
Trainable params: 2,117,498
Non-trainable params: 0
```



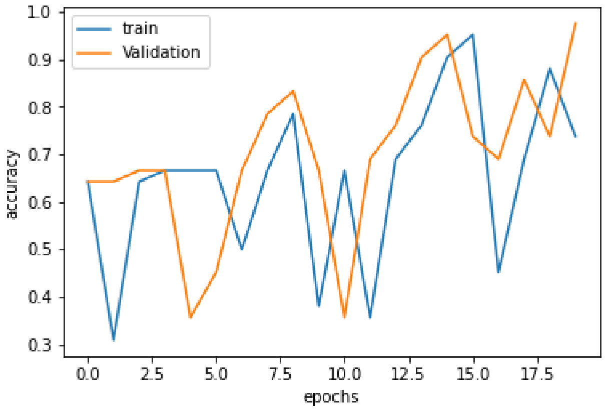
```python
opt = SGD(lr=0.01, momentum=0.9)
model.compile(optimizer=RMSprop(),loss='categorical_crossentropy',  metrics=['accuracy'])
from keras.callbacks import EarlyStopping
history = model.fit(training_set, epochs = 20, validation_data = validation_set, verbose=1, callbacks=[EarlyStopping(monitor='val_loss', patience=15)])
```

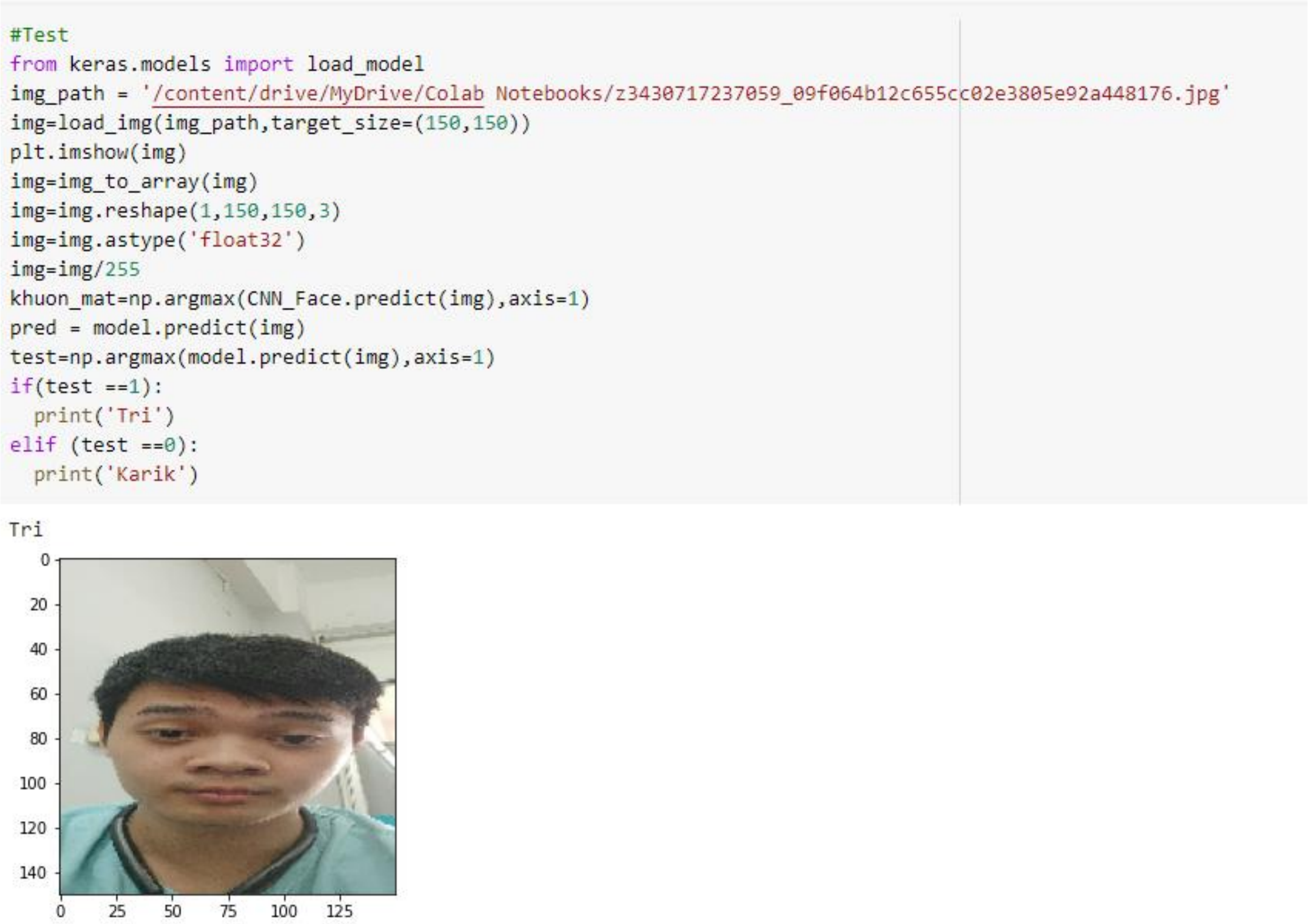
```
[ ]  model.save('Face.h5')
     from keras.models import load_model
     CNN_Face=load_model('Face.h5')
```

```
[ ]  score = model.evaluate(validation_set,verbose=0)
     print('Sai số kiểm tra là: ',score[0])
     print('Độ chính xác kiểm tra là: ',score[1])
```

```
Sai số kiểm tra là:  0.09346793591976166
Độ chính xác kiểm tra là:  0.976190447807312
```

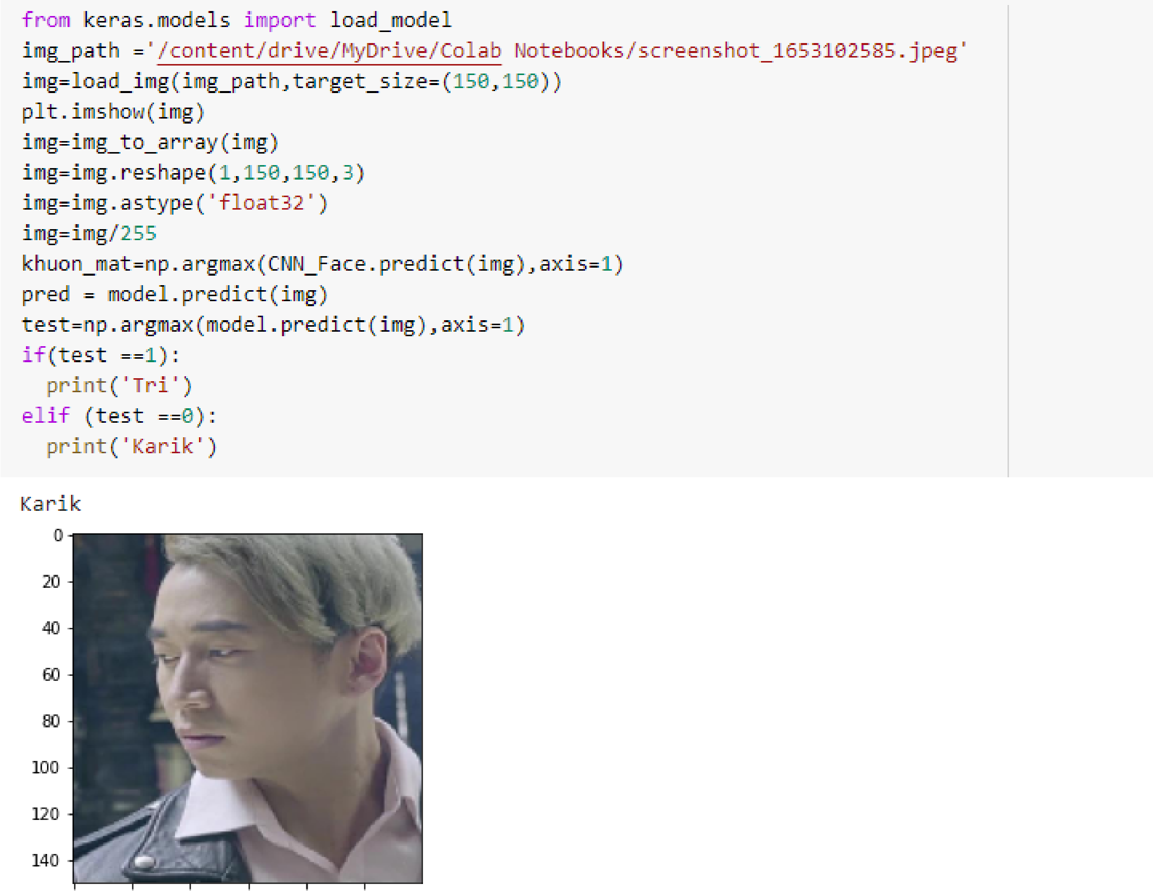
```
[ ]  plt.plot(history.history['accuracy'])
     plt.plot(history.history['val_accuracy'])
     plt.ylabel('accuracy')
     plt.xlabel('epochs')
     plt.legend(['train','Validation'])
     plt.show()
```





```
#Test
from keras.models import load_model
img_path = '/content/drive/MyDrive/Colab Notebooks/z3430717237059_09f064b12c655cc02e3805e92a448176.jpg'
img=load_img(img_path,target_size=(150,150))
plt.imshow(img)
img=img_to_array(img)
img=img.reshape(1,150,150,3)
img=img.astype('float32')
img=img/255
khuon_mat=np.argmax(CNN_Face.predict(img),axis=1)
pred = model.predict(img)
test=np.argmax(model.predict(img),axis=1)
if(test ==1):
  print('Tri')
elif (test ==0):
  print('Karik')
```

```
Tri
```

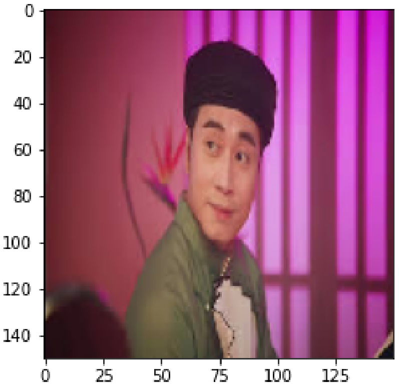


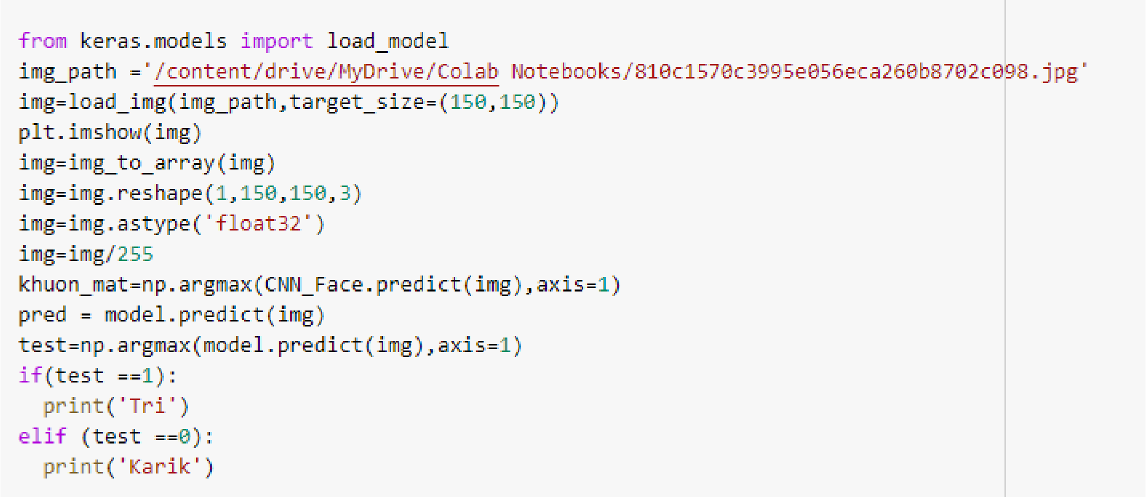
```
from keras.models import load_model
img_path ='/content/drive/MyDrive/Colab Notebooks/screenshot_1653102585.jpeg'
img=load_img(img_path,target_size=(150,150))
plt.imshow(img)
img=img_to_array(img)
img=img.reshape(1,150,150,3)
img=img.astype('float32')
img=img/255
khuon_mat=np.argmax(CNN_Face.predict(img),axis=1)
pred = model.predict(img)
test=np.argmax(model.predict(img),axis=1)
if(test ==1):
  print('Tri')
elif (test ==0):
  print('Karik')
```

```
Karik
```

```
from keras.models import load_model
img_path ='/content/drive/MyDrive/Colab Notebooks/images (1).jpg'
img=load_img(img_path,target_size=(150,150))
plt.imshow(img)
img=img_to_array(img)
img=img.reshape(1,150,150,3)
img=img.astype('float32')
img=img/255
khuon_mat=np.argmax(CNN_Face.predict(img),axis=1)
pred = model.predict(img)
test=np.argmax(model.predict(img),axis=1)
if(test ==1):
  print('Tri')
elif (test ==0):
  print('Karik')
```

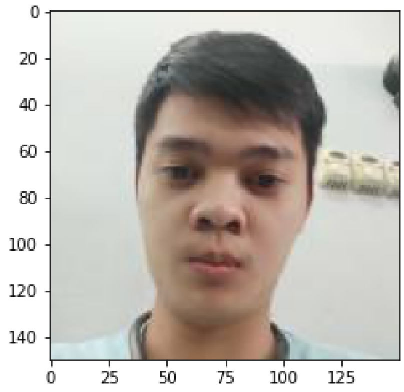
```
Karik
```



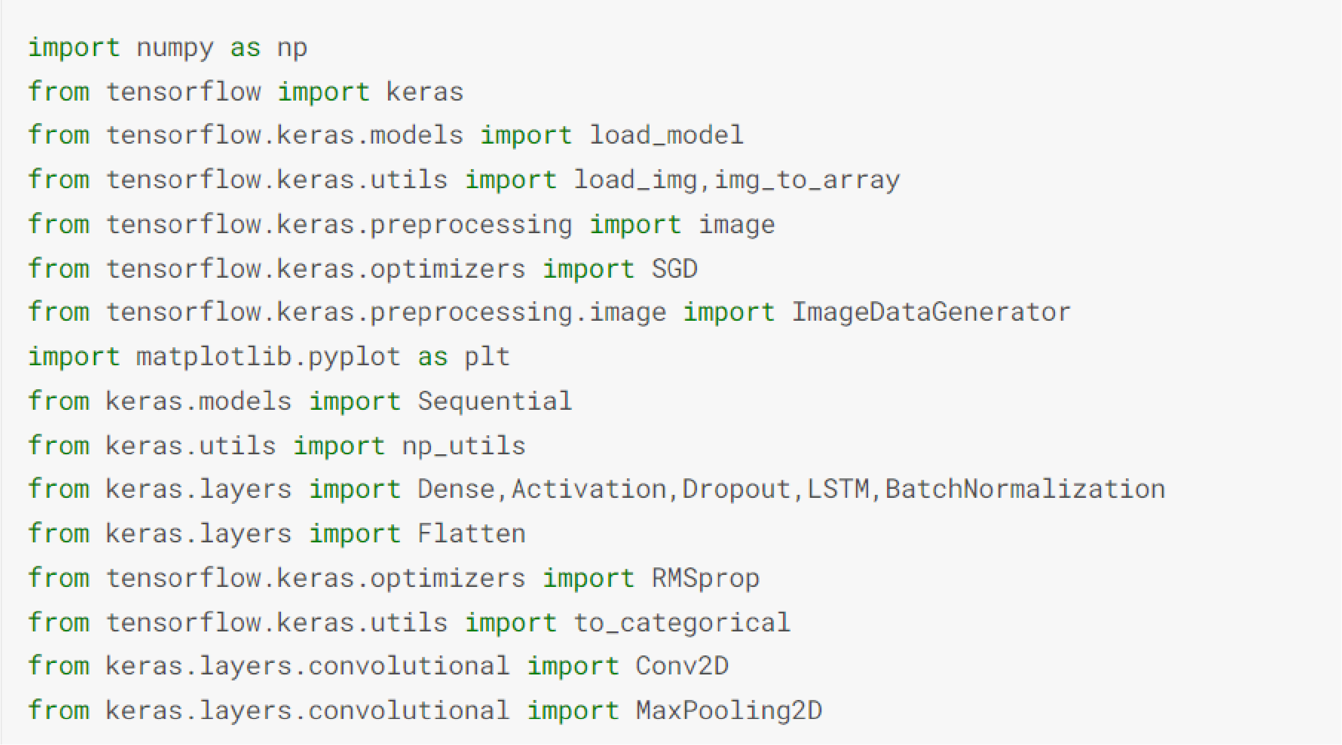


```
from keras.models import load_model
img_path ='/content/drive/MyDrive/Colab Notebooks/810c1570c3995e056eca260b8702c098.jpg'
img=load_img(img_path,target_size=(150,150))
plt.imshow(img)
img=img_to_array(img)
img=img.reshape(1,150,150,3)
img=img.astype('float32')
img=img/255
khuon_mat=np.argmax(CNN_Face.predict(img),axis=1)
pred = model.predict(img)
test=np.argmax(model.predict(img),axis=1)
if(test ==1):
  print('Tri')
elif (test ==0):
  print('Karik')
```

```
Tri
```



Bài 5: Fruit

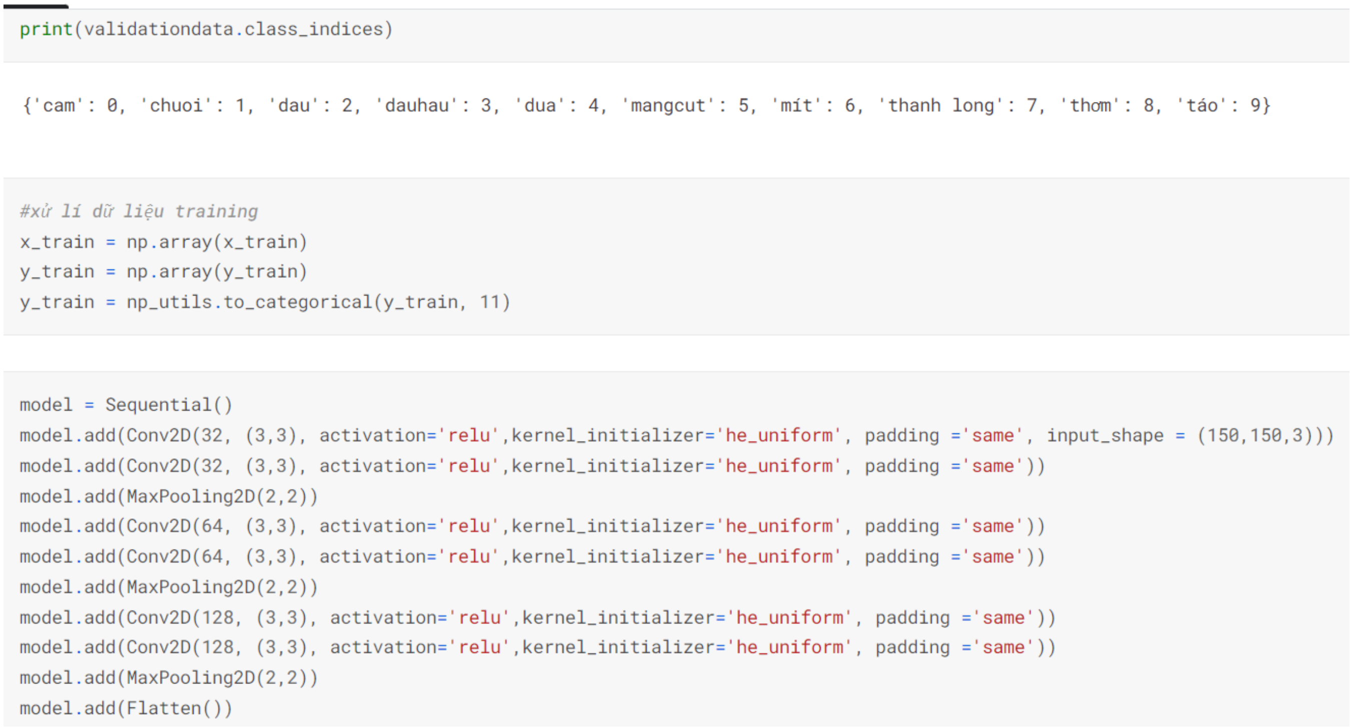


```
import numpy as np
from tensorflow import keras
from tensorflow.keras.models import load_model
from tensorflow.keras.utils import load_img,img_to_array
from tensorflow.keras.preprocessing import image
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.utils import np_utils
from keras.layers import Dense,Activation,Dropout,LSTM,BatchNormalization
from keras.layers import Flatten
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.utils import to_categorical
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
```

```
data_train='../input/traicayvietnam2/train'
data_validation='../input/traicayvietnam2/test'
train=ImageDataGenerator(rescale=1/255)
validation=ImageDataGenerator(rescale=1/255)
```

```
'../input/traicayvietnam2/test'
```

```
# tạo lí tạo dữ liệu training
traindata=train.flow_from_directory(data_train,
                                    target_size=(150,150),
                                    batch_size=10,
                                    class_mode='categorical',)
validationdata=validation.flow_from_directory(data_validation,
                                              target_size=(150,150),
                                              batch_size=10,
                                              class_mode='categorical',)
```



```
print(validationdata.class_indices)
```

```
{'cam': 0, 'chuoi': 1, 'dau': 2, 'dauhau': 3, 'dua': 4, 'mangcut': 5, 'mít': 6, 'thanh long': 7, 'thơm': 8, 'táo': 9}
```

```
#xử lí dữ liệu training
x_train = np.array(x_train)
y_train = np.array(y_train)
y_train = np_utils.to_categorical(y_train, 11)
```

```
model = Sequential()
model.add(Conv2D(32, (3,3), activation='relu',kernel_initializer='he_uniform', padding ='same', input_shape = (150,150,3)))
model.add(Conv2D(32, (3,3), activation='relu',kernel_initializer='he_uniform', padding ='same'))
model.add(MaxPooling2D(2,2))
model.add(Conv2D(64, (3,3), activation='relu',kernel_initializer='he_uniform', padding ='same'))
model.add(Conv2D(64, (3,3), activation='relu',kernel_initializer='he_uniform', padding ='same'))
model.add(MaxPooling2D(2,2))
model.add(Conv2D(128, (3,3), activation='relu',kernel_initializer='he_uniform', padding ='same'))
model.add(Conv2D(128, (3,3), activation='relu',kernel_initializer='he_uniform', padding ='same'))
model.add(MaxPooling2D(2,2))
model.add(Flatten())
model.add(Flatten())
model.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))
model.add(Dense(10, activation='softmax'))
model.summary()
```

```
Model: "sequential_10"
-----------------------------------------------------------------
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_30 (Conv2D)           (None, 150, 150, 32)      896
-----------------------------------------------------------------
conv2d_31 (Conv2D)           (None, 150, 150, 32)      9248
-----------------------------------------------------------------
max_pooling2d_18 (MaxPooling (None, 75, 75, 32)        0
-----------------------------------------------------------------
conv2d_32 (Conv2D)           (None, 75, 75, 64)        18496
-----------------------------------------------------------------
conv2d_33 (Conv2D)           (None, 75, 75, 64)        36928
-----------------------------------------------------------------
max_pooling2d_19 (MaxPooling (None, 37, 37, 64)        0
-----------------------------------------------------------------
conv2d_34 (Conv2D)           (None, 37, 37, 128)       73856
-----------------------------------------------------------------
conv2d_35 (Conv2D)           (None, 37, 37, 128)       147584
-----------------------------------------------------------------
max_pooling2d_20 (MaxPooling (None, 18, 18, 128)       0
-----------------------------------------------------------------
flatten_9 (Flatten)          (None, 41472)             0
-----------------------------------------------------------------
dense_12 (Dense)             (None, 128)               5308544
-----------------------------------------------------------------
dense_13 (Dense)             (None, 10)                1290
=================================================================
Total params: 5,596,842
Trainable params: 5,596,842
Non-trainable params: 0
-----------------------------------------------------------------
```

```
model.compile(loss='categorical_crossentropy',optimizer='rmsprop',metrics=['accuracy'])
history=model.fit(traindata,batch_size=10,epochs=10,verbose=1,validation_data=validationdata)
```

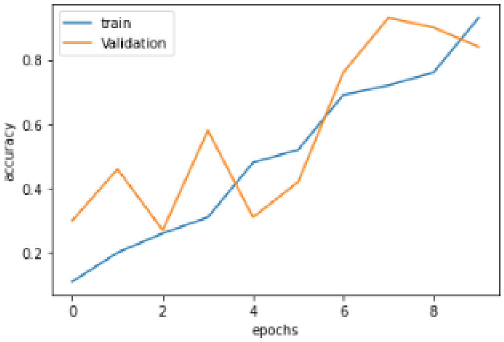
```
Epoch 1/10
10/10 [==============================] - 4s 352ms/step - loss: 30.5276 - accuracy: 0.1100 - val_loss: 2.2653 - val_accuracy: 0.300
0
Epoch 2/10
10/10 [==============================] - 3s 326ms/step - loss: 2.2619 - accuracy: 0.2000 - val_loss: 1.8307 - val_accuracy: 0.4600
Epoch 3/10
10/10 [==============================] - 3s 289ms/step - loss: 2.9626 - accuracy: 0.2600 - val_loss: 2.2101 - val_accuracy: 0.2700
Epoch 4/10
10/10 [==============================] - 3s 290ms/step - loss: 2.0307 - accuracy: 0.3100 - val_loss: 1.3298 - val_accuracy: 0.5800
Epoch 5/10
10/10 [==============================] - 4s 370ms/step - loss: 1.4688 - accuracy: 0.4800 - val_loss: 2.2932 - val_accuracy: 0.3100
Epoch 6/10
10/10 [==============================] - 3s 295ms/step - loss: 1.5897 - accuracy: 0.5200 - val_loss: 2.0340 - val_accuracy: 0.4200
Epoch 7/10
10/10 [==============================] - 3s 283ms/step - loss: 1.1161 - accuracy: 0.6900 - val_loss: 0.5442 - val_accuracy: 0.7600
Epoch 8/10
10/10 [==============================] - 3s 288ms/step - loss: 0.9575 - accuracy: 0.7200 - val_loss: 0.3129 - val_accuracy: 0.9300
Epoch 9/10
10/10 [==============================] - 3s 301ms/step - loss: 0.6664 - accuracy: 0.7600 - val_loss: 0.4130 - val_accuracy: 0.9000
Epoch 10/10
10/10 [==============================] - 3s 273ms/step - loss: 0.2804 - accuracy: 0.9300 - val_loss: 0.4405 - val_accuracy: 0.8400
```

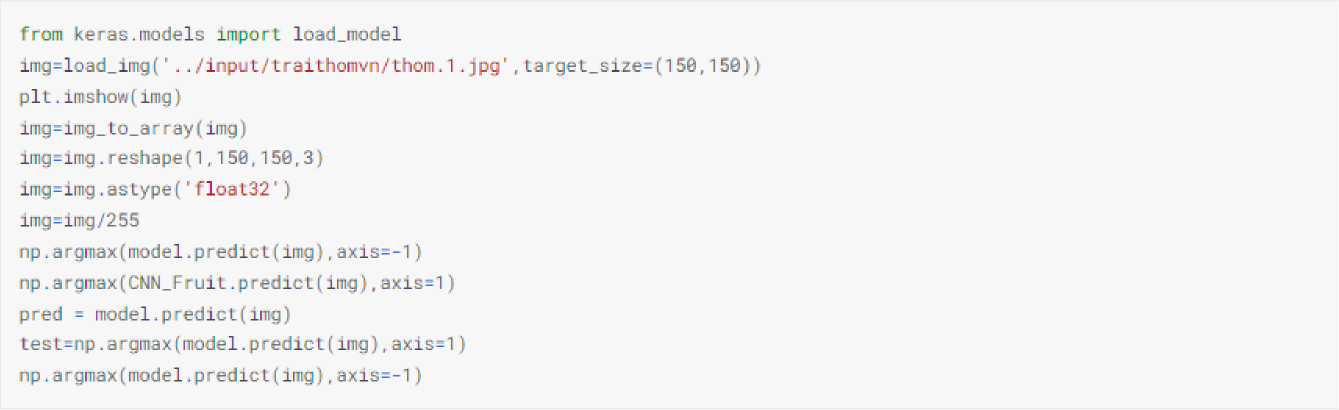
```
model.save('10LOAITRAICAY.h5')
from keras.models import load_model
CNN_Fruit=load_model('10LOAITRAICAY.h5')
```

```
score = model.evaluate(validationdata,verbose=0)
print('Sai số kiểm tra là: ',score[0])
print('Độ chính xác kiểm tra là: ',score[1])
```

```
Sai số kiểm tra là:  0.4404717683792114
Độ chính xác kiểm tra là:  0.8399999737739563
```

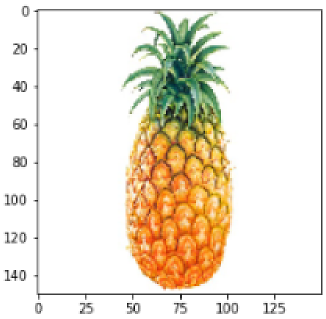
```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.ylabel('accuracy')
plt.xlabel('epochs')
plt.legend(['train','Validation'])
plt.show()
```



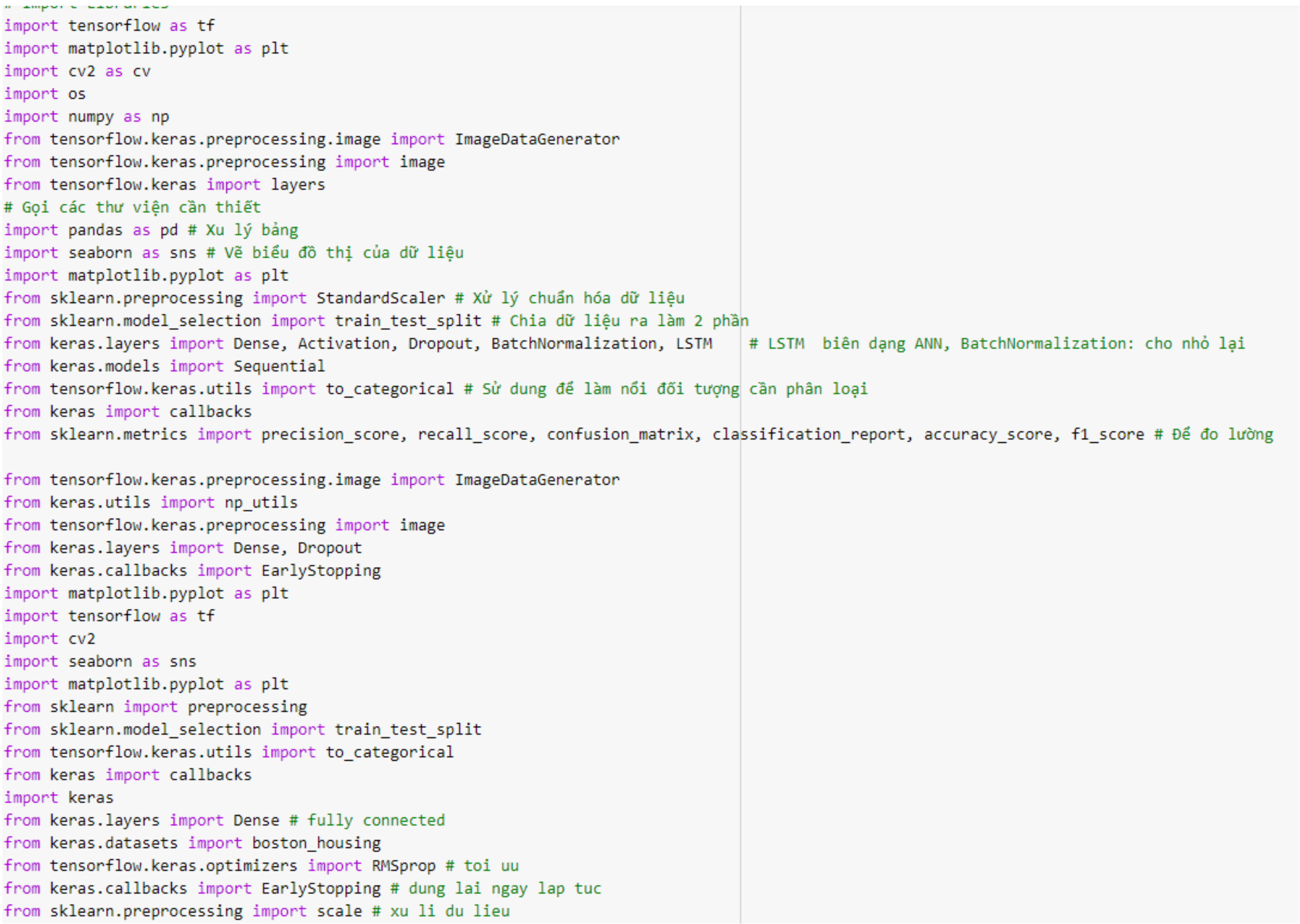


```
from keras.models import load_model
img=load_img('../input/traithomvn/thom.1.jpg',target_size=(150,150))
plt.imshow(img)
img=img_to_array(img)
img=img.reshape(1,150,150,3)
img=img.astype('float32')
img=img/255
np.argmax(model.predict(img),axis=-1)
np.argmax(CNN_Fruit.predict(img),axis=1)
pred = model.predict(img)
test=np.argmax(model.predict(img),axis=1)
np.argmax(model.predict(img),axis=-1)
```

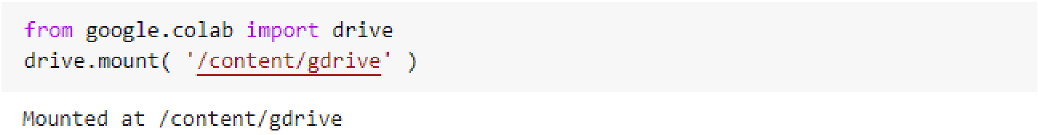
```
array([8])
```



Bài 6 : Money

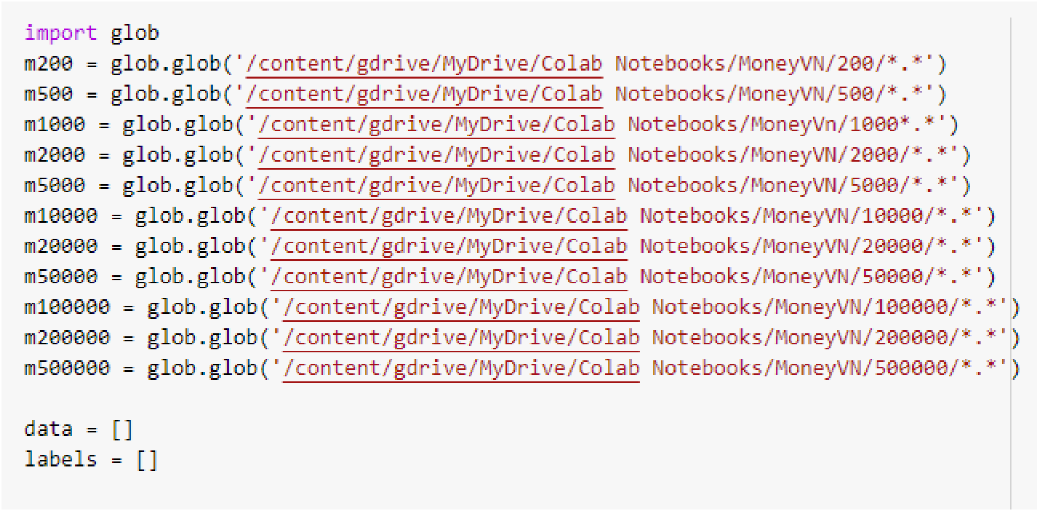


```
# import libraries
import tensorflow as tf
import matplotlib.pyplot as plt
import cv2 as cv
import os
import numpy as np
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image
from tensorflow.keras import layers
# Gọi các thư viện cần thiết
import pandas as pd # Xu lý bảng
import seaborn as sns # Vẽ biểu đồ thị của dữ liệu
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler # Xử lý chuẩn hóa dữ liệu
from sklearn.model_selection import train_test_split # Chia dữ liệu ra làm 2 phần
from keras.layers import Dense, Activation, Dropout, BatchNormalization, LSTM    # LSTM  biên dạng ANN, BatchNormalization: cho nhỏ lại
from keras.models import Sequential
from tensorflow.keras.utils import to_categorical # Sử dung để làm nối đối tượng cần phân loại
from keras import callbacks
from sklearn.metrics import precision_score, recall_score, confusion_matrix, classification_report, accuracy_score, f1_score # Để đo lường

from tensorflow.keras.preprocessing.image import ImageDataGenerator
from keras.utils import np_utils
from tensorflow.keras.preprocessing import image
from keras.layers import Dense, Dropout
from keras.callbacks import EarlyStopping
import matplotlib.pyplot as plt
import tensorflow as tf
import cv2
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
from keras import callbacks
import keras
from keras.layers import Dense # fully connected
from keras.datasets import boston_housing
from tensorflow.keras.optimizers import RMSprop # toi uu
from keras.callbacks import EarlyStopping # dung lai ngay lap tuc
from sklearn.preprocessing import scale # xu li du lieu
```

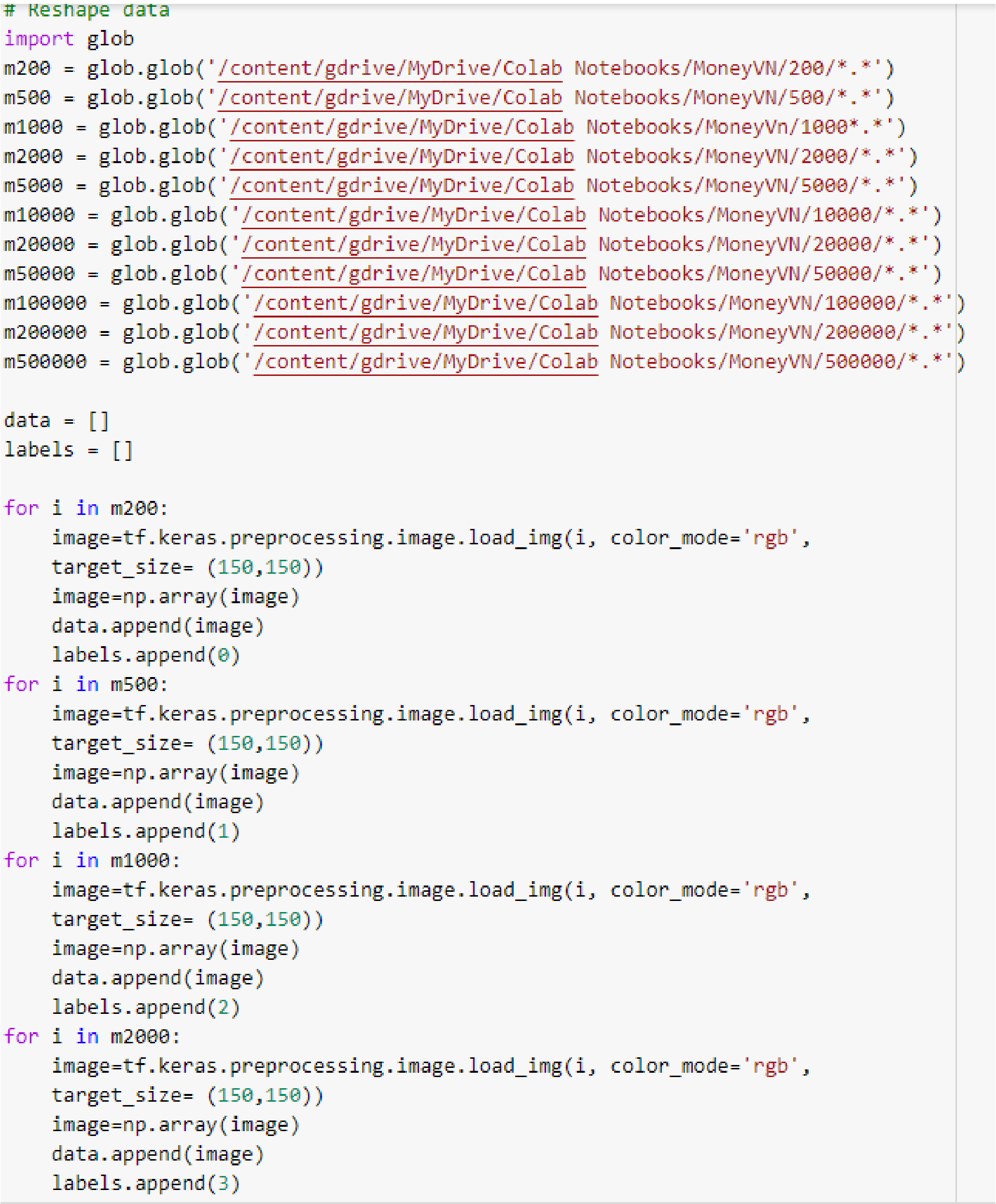


```
from google.colab import drive
drive.mount( '/content/gdrive' )
```

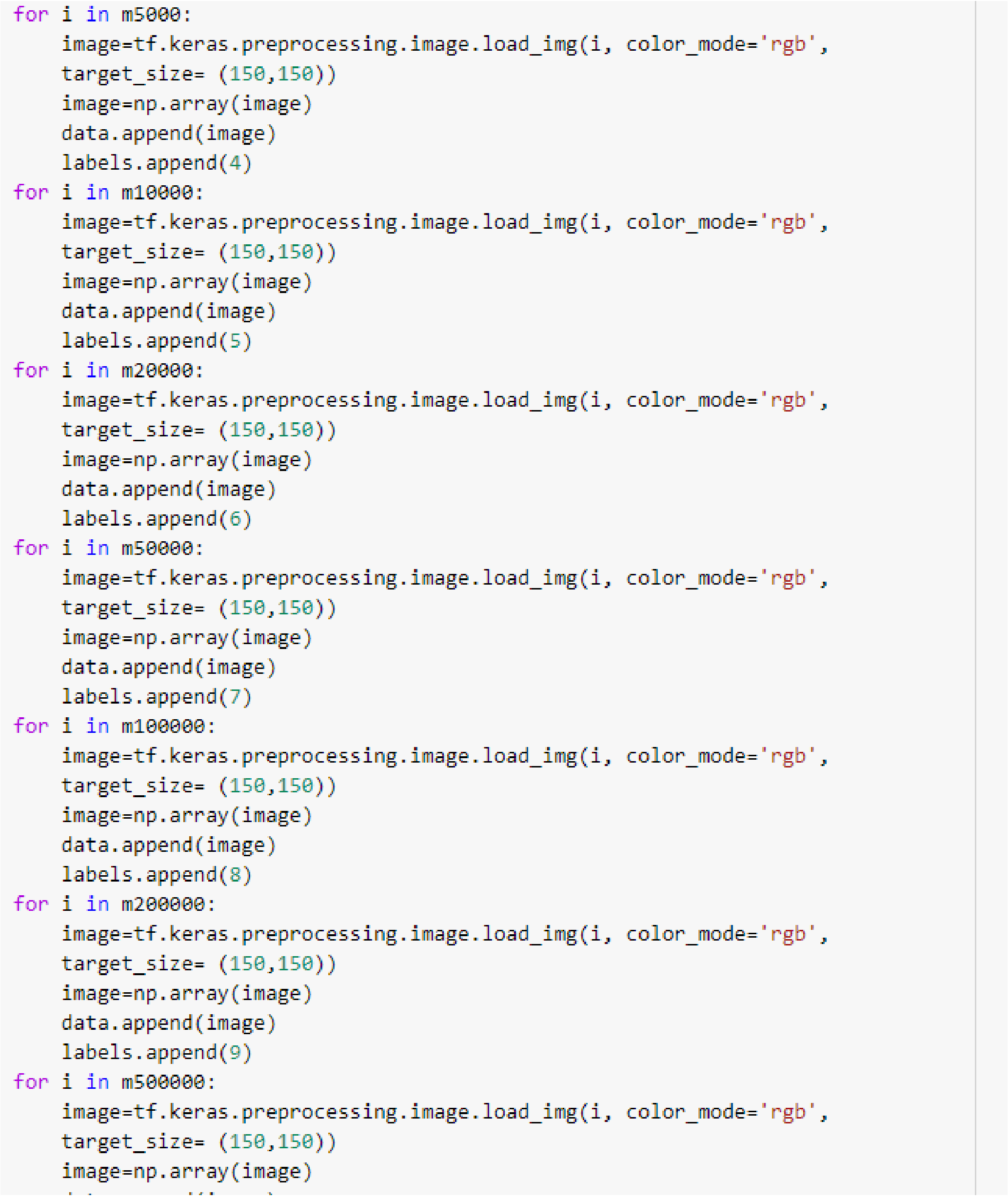
```
Mounted at /content/gdrive
```



```
import glob
m200 = glob.glob('/content/gdrive/MyDrive/Colab Notebooks/MoneyVN/200/*.*')
m500 = glob.glob('/content/gdrive/MyDrive/Colab Notebooks/MoneyVN/500/*.*')
m1000 = glob.glob('/content/gdrive/MyDrive/Colab Notebooks/MoneyVn/1000*.*')
m2000 = glob.glob('/content/gdrive/MyDrive/Colab Notebooks/MoneyVN/2000/*.*')
m5000 = glob.glob('/content/gdrive/MyDrive/Colab Notebooks/MoneyVN/5000/*.*')
m10000 = glob.glob('/content/gdrive/MyDrive/Colab Notebooks/MoneyVN/10000/*.*')
m20000 = glob.glob('/content/gdrive/MyDrive/Colab Notebooks/MoneyVN/20000/*.*')
m50000 = glob.glob('/content/gdrive/MyDrive/Colab Notebooks/MoneyVN/50000/*.*')
m100000 = glob.glob('/content/gdrive/MyDrive/Colab Notebooks/MoneyVN/100000/*.*')
m200000 = glob.glob('/content/gdrive/MyDrive/Colab Notebooks/MoneyVN/200000/*.*')
m500000 = glob.glob('/content/gdrive/MyDrive/Colab Notebooks/MoneyVN/500000/*.*')

data = []
labels = []
```



```
# Reshape data
import glob
m200 = glob.glob('/content/gdrive/MyDrive/Colab Notebooks/MoneyVN/200/*.*')
m500 = glob.glob('/content/gdrive/MyDrive/Colab Notebooks/MoneyVN/500/*.*')
m1000 = glob.glob('/content/gdrive/MyDrive/Colab Notebooks/MoneyVn/1000*.*')
m2000 = glob.glob('/content/gdrive/MyDrive/Colab Notebooks/MoneyVN/2000/*.*')
m5000 = glob.glob('/content/gdrive/MyDrive/Colab Notebooks/MoneyVN/5000/*.*')
m10000 = glob.glob('/content/gdrive/MyDrive/Colab Notebooks/MoneyVN/10000/*.*')
m20000 = glob.glob('/content/gdrive/MyDrive/Colab Notebooks/MoneyVN/20000/*.*')
m50000 = glob.glob('/content/gdrive/MyDrive/Colab Notebooks/MoneyVN/50000/*.*')
m100000 = glob.glob('/content/gdrive/MyDrive/Colab Notebooks/MoneyVN/100000/*.*')
m200000 = glob.glob('/content/gdrive/MyDrive/Colab Notebooks/MoneyVN/200000/*.*')
m500000 = glob.glob('/content/gdrive/MyDrive/Colab Notebooks/MoneyVN/500000/*.*')

data = []
labels = []

for i in m200:
    image=tf.keras.preprocessing.image.load_img(i, color_mode='rgb',
    target_size= (150,150))
    image=np.array(image)
    data.append(image)
    labels.append(0)
for i in m500:
    image=tf.keras.preprocessing.image.load_img(i, color_mode='rgb',
    target_size= (150,150))
    image=np.array(image)
    data.append(image)
    labels.append(1)
for i in m1000:
    image=tf.keras.preprocessing.image.load_img(i, color_mode='rgb',
    target_size= (150,150))
    image=np.array(image)
    data.append(image)
    labels.append(2)
for i in m2000:
    image=tf.keras.preprocessing.image.load_img(i, color_mode='rgb',
    target_size= (150,150))
    image=np.array(image)
    data.append(image)
    labels.append(3)
```



```
for i in m5000:
    image=tf.keras.preprocessing.image.load_img(i, color_mode='rgb',
    target_size= (150,150))
    image=np.array(image)
    data.append(image)
    labels.append(4)
for i in m10000:
    image=tf.keras.preprocessing.image.load_img(i, color_mode='rgb',
    target_size= (150,150))
    image=np.array(image)
    data.append(image)
    labels.append(5)
for i in m20000:
    image=tf.keras.preprocessing.image.load_img(i, color_mode='rgb',
    target_size= (150,150))
    image=np.array(image)
    data.append(image)
    labels.append(6)
for i in m50000:
    image=tf.keras.preprocessing.image.load_img(i, color_mode='rgb',
    target_size= (150,150))
    image=np.array(image)
    data.append(image)
    labels.append(7)
for i in m100000:
    image=tf.keras.preprocessing.image.load_img(i, color_mode='rgb',
    target_size= (150,150))
    image=np.array(image)
    data.append(image)
    labels.append(8)
for i in m200000:
    image=tf.keras.preprocessing.image.load_img(i, color_mode='rgb',
    target_size= (150,150))
    image=np.array(image)
    data.append(image)
    labels.append(9)
for i in m500000:
    image=tf.keras.preprocessing.image.load_img(i, color_mode='rgb',
    target_size= (150,150))
    image=np.array(image)
    data.append(image)
    labels.append(10)

data = np.array(data)
labels = np.array(labels)

from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(data, labels, test_size=0.2,
                                                random_state=42)
```

```
print(X_train.shape,Y_train.shape)
```

```
(144, 150, 150, 3) (144,)
```

```
X_train = X_train.reshape((X_train.shape[0],150,150,3)).astype('float32')/255
X_test = X_test.reshape((X_test.shape[0],150,150,3)).astype('float32')/255

Y_train = to_categorical(Y_train,11)
Y_test = to_categorical(Y_test,11)
```

```
print(X_train, Y_train)
```

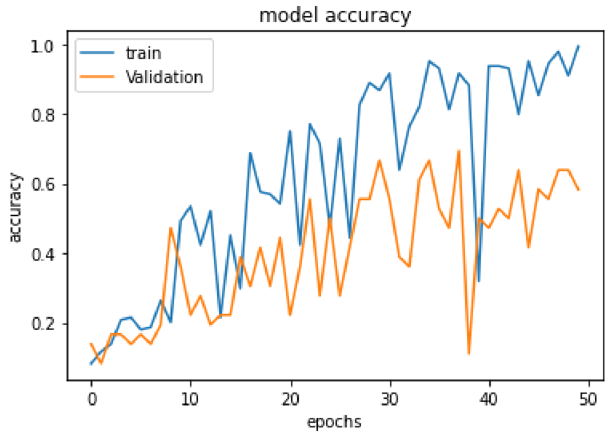
```
# Create model
from keras.layers import Conv2D, MaxPooling2D
model = Sequential()
model.add(Conv2D(32,(3,3), activation='relu',kernel_initializer='he_uniform',padding='same',input_shape=(150,150,3)))
model.add(Conv2D(32,(3,3), activation='relu',kernel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D(2,2))

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

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

model.add(Conv2D(256,(3,3), activation='relu',kernel_initializer='he_uniform',padding='same')) # 256 lan tich chap
model.add(Conv2D(256,(3,3), activation='relu',kernel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D(2,2))


from keras.layers import Dense, Activation, Flatten
model.add(Flatten())
model.add(Dense(128, activation = 'relu', kernel_initializer='he_uniform'))
model.add(Dense(11))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 150, 150, 32)      896

 conv2d_1 (Conv2D)           (None, 150, 150, 32)      9248

 max_pooling2d (MaxPooling2D  (None, 75, 75, 32)       0
 )

 conv2d_2 (Conv2D)           (None, 75, 75, 64)        18496

 conv2d_3 (Conv2D)           (None, 75, 75, 64)        36928

 max_pooling2d_1 (MaxPooling  (None, 37, 37, 64)       0
 2D)

 conv2d_4 (Conv2D)           (None, 37, 37, 128)       73856

 conv2d_5 (Conv2D)           (None, 37, 37, 128)       147584

 max_pooling2d_2 (MaxPooling  (None, 18, 18, 128)      0
 2D)

 conv2d_6 (Conv2D)           (None, 18, 18, 256)       295168

 conv2d_7 (Conv2D)           (None, 18, 18, 256)       590080

 max_pooling2d_3 (MaxPooling  (None, 9, 9, 256)        0
 2D)

 flatten (Flatten)           (None, 20736)             0

 dense (Dense)               (None, 128)               2654336

 dense_1 (Dense)             (None, 11)                1419

=================================================================
Total params: 3,828,011
Trainable params: 3,828,011
Non-trainable params: 0
_________________________________________________________________
```

```
# Training
model.compile(loss='mse',optimizer=RMSprop(),metrics=['accuracy'])
history = model.fit(X_train, Y_train, epochs =50, batch_size =128,validation_data=(X_test,Y_test) , verbose = 2)
```

```
# Save model
from tensorflow.keras.models import load_model
model.save('MoneyVN.h5')
MoneyVN = load_model('MoneyVN.h5')
```

```
# Draw plot
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epochs')
plt.legend(['train','Validation'])
plt.show()
```



```
# Check accuracy
from tensorflow.keras.utils import load_img, img_to_array
import numpy as np
filename = "/content/gdrive/MyDrive/Colab Notebooks/MoneyVN/20k.jpg"
predict = ['200','500','1000','2000','5000','10000','20000','50000','100000','200000','500000']
predict = np.array(predict)
```

```
img = load_img(filename,target_size=(150,150))
img = img_to_array(img)
img = img.reshape(1,150,150,3)
img = img.astype('float32')
img = img/255

result = np.argmax(MoneyVN.predict(img),axis=-1)
predict[result]
```

```
array(['20000'], dtype='<U6')
```

```
score = model.evaluate(X_test,Y_test, verbose=0)
print("Loss = ", score[0])
print("accuracy = ", score[1])
```

```
Loss =  0.05602594092488289
accuracy =  0.6111111044883728
```

Link Nộp Git: https://github.com/uyminhtri2702/AI