Trần Ngọc Đoàn - 19146175 - Lớp T5 - tiết 12-15

https://github.com/DoanAI/Fruit.git

```
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.layers import Flatten, Dense, Dropout, Activation
from google.colab import drive
```

```
drive.mount('/content/drive',force_remount=True)
from tensorflow.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/CNN/FR/training_set
                                               target_size=(256,256),
                                               batch_size=32,
                                               class_mode ='categorical')
test_set=train_datagen.flow_from_directory('/content/drive/MyDrive/CNN/FR/test_set',
                                               target_size=(256,256),
                                               batch_size=32,
                                               class_mode ='categorical')
```

```
Mounted at /content/drive
Found 50 images belonging to 10 classes.
Found 10 images belonging to 10 classes.
```

```
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m
```

```
model=Sequential()
model.add(Conv2D(128,(3,3),activation='relu',kernel_initializer='he_uniform',padding='same
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(64,(3,3),activation='relu',kernel_initializer='he_uniform',padding='same'
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(32,(3,3),activation='relu',kernel_initializer='he_uniform',padding='same'
model.add(MaxPooling2D((2,2)))
model.add(Flatten())
model.add(Dense(128,activation='relu',kernel_initializer = 'he_uniform'))
#model.add(Dropout(0,2))
model.add(Dense(10,activation='Softmax'))
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.callbacks import EarlyStopping
#opt = SGD(lr = 0.01, momentum = 0.9)
model.compile(optimizer = 'adam', loss ='categorical_crossentropy',metrics = ['accuracy'])
```

```
callbacks=[EarlyStopping(monitor='val_loss',patience=100)]
history=model.fit(training_set,
                  steps_per_epoch=len(training_set),
                  batch_size = 64,
                  epochs=20,
                  validation_data=test_set,
                  validation_steps=len(test_set),
                  callbacks=callbacks,
                  verbose = 1)
```

```
Epoch 1/20
2/2 [==============================] - 19s 8s/step - loss: 4.7581 - accuracy: 0.1000
Epoch 2/20
2/2 [==============================] - 17s 6s/step - loss: 4.2544 - accuracy: 0.0600
Epoch 3/20
2/2 [==============================] - 16s 10s/step - loss: 2.0557 - accuracy: 0.2600
Epoch 4/20
2/2 [==============================] - 16s 10s/step - loss: 2.0273 - accuracy: 0.2400
Epoch 5/20
2/2 [==============================] - 16s 10s/step - loss: 1.8692 - accuracy: 0.3800
Epoch 6/20
2/2 [==============================] - 16s 6s/step - loss: 1.8103 - accuracy: 0.3600
Epoch 7/20
2/2 [==============================] - 16s 6s/step - loss: 1.7800 - accuracy: 0.4000
Epoch 8/20
2/2 [==============================] - 16s 6s/step - loss: 1.6337 - accuracy: 0.5000
Epoch 9/20
2/2 [==============================] - 16s 6s/step - loss: 1.4618 - accuracy: 0.4600
Epoch 10/20
2/2 [==============================] - 16s 10s/step - loss: 1.4707 - accuracy: 0.6200
Epoch 11/20
2/2 [==============================] - 16s 6s/step - loss: 1.1584 - accuracy: 0.6400
Epoch 12/20
2/2 [==============================] - 16s 6s/step - loss: 0.8814 - accuracy: 0.7600
Epoch 13/20
2/2 [==============================] - 16s 10s/step - loss: 0.8184 - accuracy: 0.8000
Epoch 14/20
2/2 [==============================] - 16s 10s/step - loss: 0.6904 - accuracy: 0.7600
Epoch 15/20
2/2 [==============================] - 16s 10s/step - loss: 0.4306 - accuracy: 0.9000
Epoch 16/20
2/2 [==============================] - 16s 6s/step - loss: 0.4552 - accuracy: 0.8800
Epoch 17/20
2/2 [==============================] - 16s 10s/step - loss: 0.2742 - accuracy: 0.9000
Epoch 18/20
2/2 [==============================] - 16s 10s/step - loss: 0.2244 - accuracy: 0.9400
Epoch 19/20
2/2 [==============================] - 16s 10s/step - loss: 0.2134 - accuracy: 0.9000
Epoch 20/20
2/2 [==============================] - 16s 6s/step - loss: 0.1902 - accuracy: 0.9200
```

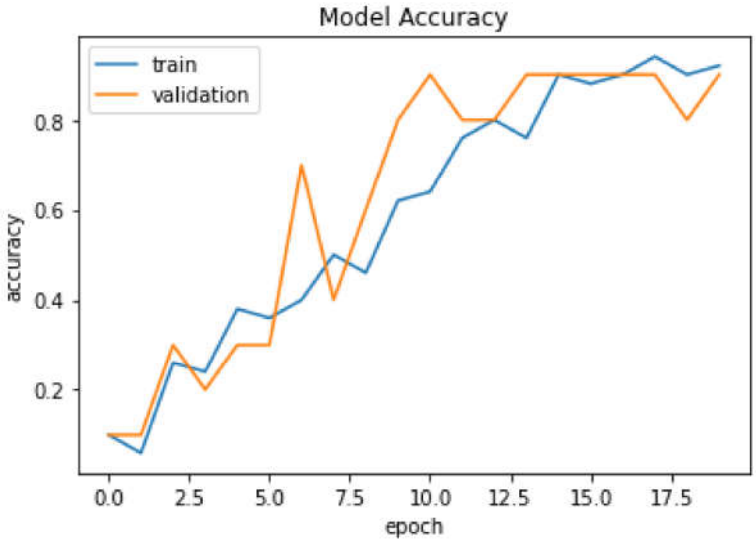
```
score = model.evaluate(test_set,verbose=0)
print('Sai số kiểm tra là: ',score[0])
print('Độ chính xác kiểm tra là: ',score[1])
```

```
Sai số kiểm tra là:  0.2416674792766571
Độ chính xác kiểm tra là:  0.8999999761581421
```

```
model.save('model_fruit.h5')
from tensorflow.keras.models import load_model
model=load_model('model_fruit.h5')
```

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train','validation'], loc='upper-left')
```

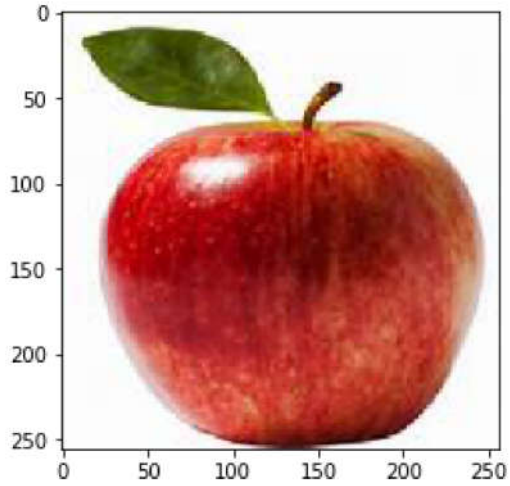
```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: MatplotlibDeprecatior
        best
        upper right
        upper left
        lower left
        lower right
        right
        center left
        center right
        lower center
        upper center
        center
This will raise an exception in 3.3.

<matplotlib.legend.Legend at 0x7f71347ba8d0>
```



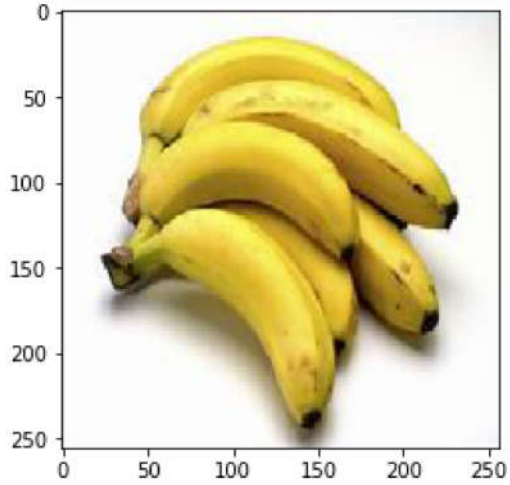

```
from tensorflow.keras.utils import load_img
from tensorflow.keras.utils import img_to_array
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import math

img_0 = load_img('/content/drive/MyDrive/CNN/FR/prediction/Apple_1.jpg', target_size=(256,
img_1 = load_img('/content/drive/MyDrive/CNN/FR/prediction/Banana.jpg', target_size=(256,2
img_2 = load_img('/content/drive/MyDrive/CNN/FR/prediction/Cherry.jpg', target_size=(256,2
img_3 = load_img('/content/drive/MyDrive/CNN/FR/prediction/Durian.jpg', target_size=(256,2
img_4 = load_img('/content/drive/MyDrive/CNN/FR/prediction/Lemon.jpg', target_size=(256,25
```

```
img_5 = load_img('/content/drive/MyDrive/CNN/FR/prediction/Lime.jpg', target_size=(256,256
img_6 = load_img('/content/drive/MyDrive/CNN/FR/prediction/Mango.jpg', target_size=(256,25
img_7 = load_img('/content/drive/MyDrive/CNN/FR/prediction/Melon.jpg', target_size=(256,25
img_8 = load_img('/content/drive/MyDrive/CNN/FR/prediction/Orange.jpg', target_size=(256,2
img_9 = load_img('/content/drive/MyDrive/CNN/FR/prediction/Plum.jpg', target_size=(256,256

img = [img_0,img_1,img_2,img_3,img_4,img_5,img_6,img_7,img_8,img_9]
fruit = ['Apple','Banana','Cherry','Durian','Lemon','Lime','Mango','Melon','Orange','Plum'

for i in range(10):
  plt.imshow(img[i])
  imga = img_to_array(img[i])
  imga = imga/255
  imga = np.expand_dims(imga,axis=0)
  result = model.predict(imga)

  if round(result[0][i])==1: prediction = fruit[i]
  print(prediction)
  plt.show()
```

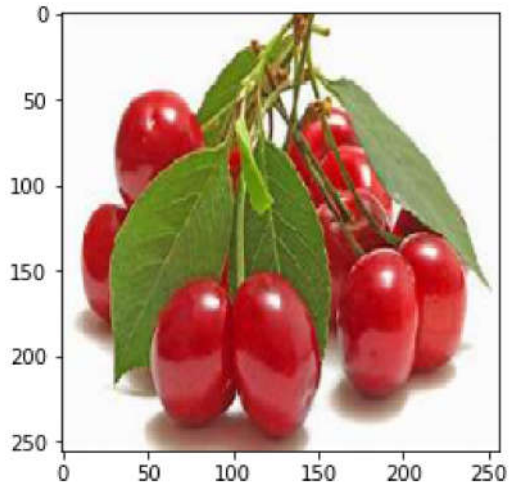
Apple



Banana

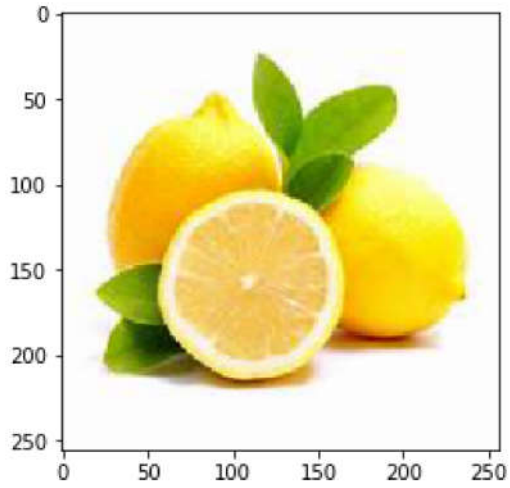


Cherry

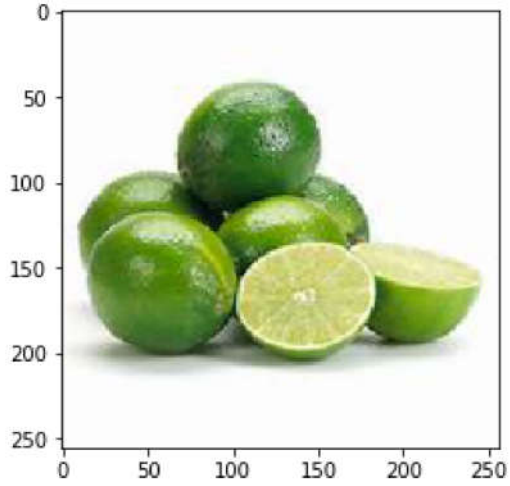


Durian

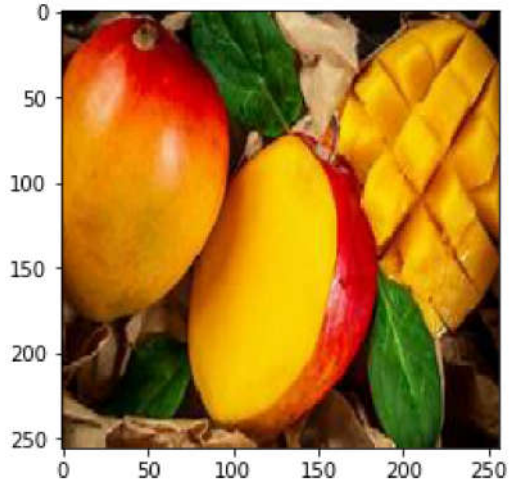
Lemon



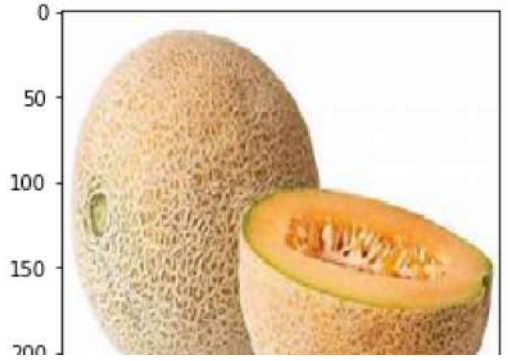
Lime

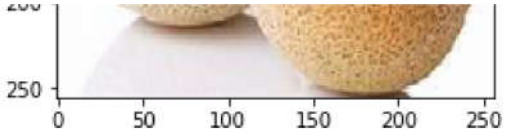


Mango

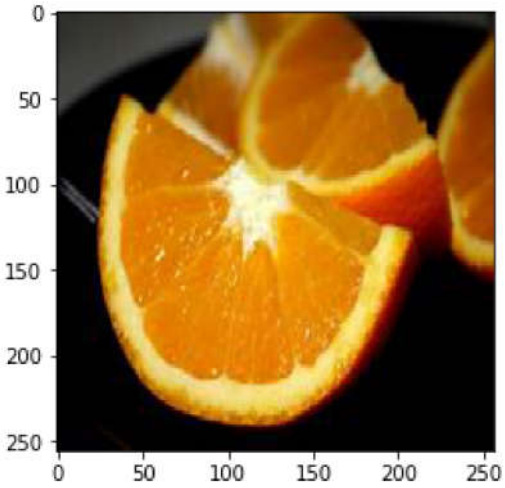


Melon






```
Orange
```




```
Plum
```