```
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Activation, Dense, Flatten, BatchNormalization, Conv2D, MaxPool2D, D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.metrics import categorical crossentropy
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import confusion matrix
import itertools
import os
import shutil
import random
import glob
import matplotlib.pyplot as plt
import warnings
from tensorflow.python.keras.utils.data utils import Sequence
warnings.simplefilter(action='ignore', category=FutureWarning)
%matplotlib inline
from google.colab import drive
drive.mount('/content/drive')
    Mounted at /content/drive
train path = '/content/drive/MyDrive/fruits/train'
valid_path = '/content/drive/MyDrive/fruits/train'
test_path = '/content/drive/MyDrive/fruits/train'
train_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')
test datagen = ImageDataGenerator(rescale=1./255)
train batches = train datagen.flow from directory(train path, target size=(224, 224),
                                                  batch_size=10, class_mode='categorical')
validation_batches = test_datagen.flow_from_directory(valid_path, target_size=(224, 224),
                                                      batch_size=10, class_mode='categorical')
test batches = test datagen.flow from directory(test path, target size=(224, 224),
                                                batch size=10, class mode='categorical')
     Found 127 images belonging to 9 classes.
     Found 127 images belonging to 9 classes.
     Found 127 images belonging to 9 classes.
imgs, labels = next(train batches)
def plotImages(images_arr):
```

```
fig, axes = plt.subplots(1, 10, figsize=(20,20))
    axes = axes.flatten()
    for img, ax in zip( images_arr, axes):
        ax.imshow(img)
        ax.axis('off')
    plt.tight layout()
    plt.show()
model = Sequential([
    Conv2D(filters=64, kernel size=(3, 3), activation='relu', padding = 'same', input shape=(224,224,3))
    MaxPool2D(pool_size=(2, 2), strides=2),
    Conv2D(filters=128, kernel_size=(3, 3), activation='relu', padding = 'same'),
    MaxPool2D(pool_size=(2, 2), strides=2),
    Conv2D(filters=256, kernel_size=(3, 3), activation='relu', padding = 'same'),
    MaxPool2D(pool_size=(2, 2), strides=2),
    Dropout(0.2),
    Conv2D(filters=256, kernel size=(3, 3), activation='relu', padding = 'same'),
    Flatten(),
    Dropout(0.5),
    Dense(units=9, activation='softmax')])
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 224, 224, 64)	1792
<pre>max_pooling2d (MaxPooling2D)</pre>	(None, 112, 112, 64)	0
conv2d_1 (Conv2D)	(None, 112, 112, 128)	73856
<pre>max_pooling2d_1 (MaxPooling 2D)</pre>	(None, 56, 56, 128)	0
conv2d_2 (Conv2D)	(None, 56, 56, 256)	295168
<pre>max_pooling2d_2 (MaxPooling 2D)</pre>	(None, 28, 28, 256)	0
dropout (Dropout)	(None, 28, 28, 256)	0
conv2d_3 (Conv2D)	(None, 28, 28, 256)	590080
flatten (Flatten)	(None, 200704)	0
dropout_1 (Dropout)	(None, 200704)	0
dense (Dense)	(None, 9)	1806345

Total params: 2,767,241 Trainable params: 2,767,241 Non-trainable params: 0

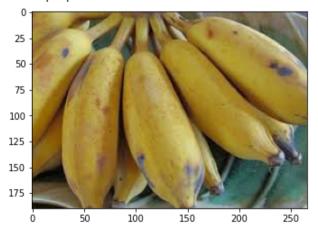
model.compile(optimizer=Adam(learning_rate=0.0005), loss='categorical_crossentropy', metrics=['accuracy'
model.fit(x=train_batches, steps_per_epoch=len(train_batches), validation_data=validation_batches, valid

```
model.save('Train_MonAnVN.h5')
classes = ['chuối','măng cut','ổi','táo','thanh long','thơm','trái cam','trái sampoche','trái vải']
print("Image Processing......Compleated")
     Image Processing.....Compleated
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 anh đọc được
 plt.imshow(mpimg.imread(path))
 img=image.load img(path,target size=(224,224))
 x=image.img to array(img)
 x=np.expand dims(x,axis=0)
 images=np.vstack([x])
 y predict = model.predict(images,batch size=200000)
 print(y_predict)
 print('Giá tri dự đoán: ', classes[np.argmax(y_predict)])
```

Chọn tệp Không có tệp nào được chọn Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving images (1).jpg to images (1).jpg [[1. 0. 0. 0. 0. 0. 0. 0. 0.]]

Giá trị dự đoán: chuối



•