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

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

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

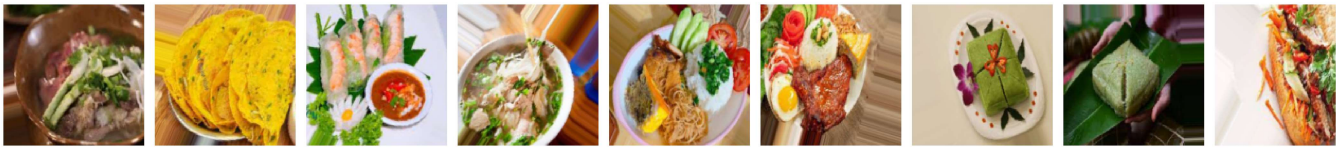
```
Drive already mounted at /content/drive/; to attempt to forcibly remount, call drive.mo
```

```
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, M
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


train_path = '/content/drive/MyDrive/mon_an'
valid_path = '/content/drive/MyDrive/mon_an'
test_path = '/content/drive/MyDrive/mon_an'


# Data generators
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')

# Note that the validation data should not be augmented!
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 200 images belonging to 10 classes.
Found 200 images belonging to 10 classes.
Found 200 images belonging to 10 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()


plotImages(imgs)
print(labels)
```



```
[[0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]]
```

```
model = Sequential([
    Conv2D(filters=32, kernel_size=(3, 3), activation='relu', padding = 'same', input_shape=(
    MaxPool2D(pool_size=(2, 2), strides=2),
    Conv2D(filters=64, kernel_size=(3, 3), activation='relu', padding = 'same'),
    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),
    Dropout(0.2),
    Conv2D(filters=128, kernel_size=(3, 3), activation='relu', padding = 'same'),
    Flatten(),
    Dropout(0.5),
    Dense(units=10, activation='softmax')
])
model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_20 (Conv2D)          (None, 224, 224, 32)      896

 max_pooling2d_15 (MaxPoolin  (None, 112, 112, 32)     0
 g2D)

 conv2d_21 (Conv2D)          (None, 112, 112, 64)      18496

 max_pooling2d_16 (MaxPoolin  (None, 56, 56, 64)       0
 g2D)

 conv2d_22 (Conv2D)          (None, 56, 56, 128)       73856

 max_pooling2d_17 (MaxPoolin  (None, 28, 28, 128)      0
 g2D)

 dropout_10 (Dropout)        (None, 28, 28, 128)       0

 conv2d_23 (Conv2D)          (None, 28, 28, 128)       147584

 flatten_5 (Flatten)         (None, 100352)            0

 dropout_11 (Dropout)        (None, 100352)            0

 dense_5 (Dense)             (None, 10)                1003530

=================================================================
Total params: 1,244,362
Trainable params: 1,244,362
Non-trainable params: 0
_________________________________________________________________
```

```
model.compile(optimizer=Adam(learning_rate=0.0005), loss='categorical_crossentropy', metrics=
model.fit(x=train_batches, steps_per_epoch=len(train_batches), validation_data=validation_bat
          validation_steps=len(validation_batches), epochs=10, verbose=1)
```

```
Epoch 1/10
20/20 [==============================] - 4s 183ms/step - loss: 2.3729 - accuracy: 0.095
Epoch 2/10
20/20 [==============================] - 3s 170ms/step - loss: 2.2685 - accuracy: 0.130
Epoch 3/10
20/20 [==============================] - 3s 173ms/step - loss: 2.1127 - accuracy: 0.220
Epoch 4/10
```

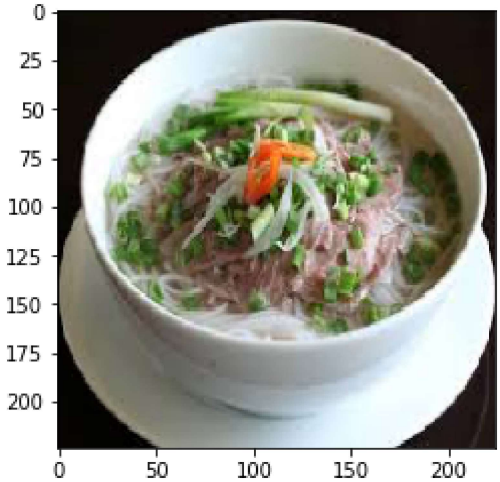
```
20/20 [==============================] - 3s 173ms/step - loss: 1.9418 - accuracy: 0.290
Epoch 5/10
20/20 [==============================] - 3s 175ms/step - loss: 1.7596 - accuracy: 0.360
Epoch 6/10
20/20 [==============================] - 3s 172ms/step - loss: 1.6914 - accuracy: 0.355
Epoch 7/10
20/20 [==============================] - 3s 173ms/step - loss: 1.5065 - accuracy: 0.485
Epoch 8/10
20/20 [==============================] - 3s 171ms/step - loss: 1.4902 - accuracy: 0.400
Epoch 9/10
20/20 [==============================] - 3s 175ms/step - loss: 1.4340 - accuracy: 0.450
Epoch 10/10
20/20 [==============================] - 3s 170ms/step - loss: 1.3442 - accuracy: 0.485
<keras.callbacks.History at 0x7f12f041ab50>
```

```
model.save('mon_an.h5')
```

```
from keras.models import load_model

model5 = load_model('mon_an.h5')
```

```
from keras.preprocessing.image import load_img,img_to_array
img=load_img('/content/drive/MyDrive/mon_an/pho/images (10).jpg',target_size=(224,224))
plt.imshow(img)
img=img_to_array(img)
img=img.reshape(1,224,224,3)
img=img.astype('float32')
img=img/255
img.shape
```

```
(1, 224, 224, 3)
```




```
a=np.argmax(model5.predict(img),axis=1)
if a == 0:
  print('Oc xao')
```

```
    print('bo_xao')
if a == 1:
  print('banh_xeo')
if a == 2:
  print('banh_mi')
if a == 3:
  print('ca_kho')
if a == 4:
  print('bo_kho')
if a == 5:
  print('com_tam')
if a == 6:
  print('goi_cuon')
if a == 7:
  print('mi_quang')
if a == 8:
  print('nem_ran')
if a == 9:
  print('Pho')
```

```
WARNING:tensorflow:5 out of the last 5 calls to <function Model.make_predict_function.<
Pho
```

✓ 0 giây hoàn thành lúc 14:50