# NHẬN DIỆN TRÁI CÂY BẰNG PHƯƠNG PHÁP CNN

SVTH: Võ Nguyễn Phú Thịnh – Trường Đại học Sư phạm Kỹ Thuật.

## *Tóm tắt sơ lược*

Trước kia, người ta thường sử dụng phương pháp nhận diện trái cây phân loại truyền thống do nhân công phân loại trên dây chuyền. Phương pháp này có nhược điểm cao do sức người hạn chế, tốn nhiều thời gian và nếu số lượng trái cây lớn thì phải sử dụng nhiều nhân công lao động, tốn kém chi phí về con người và khó có thể đạt năng suất cao. Việc phân loại như vậy rất tốn thời gian cũng như nguồn nhân lực. Vì vậy, cần một phương pháp mới hiện đại nhằm mục đích phân loại được trái cây nhanh chóng mà tiết kiệm được những chi phí không cần thiết.

Nghiên cứu được thực hiện từ những cơ sở lý thuyết dựa trên nền tảng đã học từ bộ môn Trí tuệ nhân tạo (Artificial Intelligence) và Học máy (Machine Learning), áp dụng vào việc xây dựng mô hình và xử lí dữ liệu đầu vào. Nghiên cứu cũng dựa trên các đề tài đã được thực hiên trước đây, tham khảo từ các đề tài trong và ngoài nước cũng như các nguồn tài liệu tham khảo trên Internet.

Sau quá trình nghiên cứu và thực hiện tác giả đã thu được những kết quả từ cơ sở lý thuyết và chọn lựa được phương pháp xây dựng mô hình huấn luyện, tác giả cũng hoàn thành việc xây dựng và viết code chạy mô hình trên Google Colab. Bên cạnh đó, trong quá trình nghiên cứu cũng không thể tránh khỏi những điểm yếu cần phải khắc phục do thời gian thực hiện có hạn và nhiều thiếu xót về kiến thức nên độ chính xác chưa cao và hàm lượng khoa học chưa được đánh giá cao.

Kết luận rút ra được sau quá trình nghiên cứu rằng tác giả đã đã thiết lập được một mô hình nhận diện trái cây qua camera có thể áp dụng vào thực tiễn.

### *Đặt vấn đề*

Trong thực tế hiện nay, việc phân loại trái cây thủ công dựa vào vỏ ngoài, điểm đặc thù và kích thước để phân loại trái cây đó chín hay chưa. Cách làm này cũng có ưu điểm nhỏ như việc phân loại và lựa chọn loại hàng tươi tốt tránh các tác nhân hoa quả bị va đập, trầy xước. Tuy nhiên đối với số lượng lớn thì việc phân loại thủ công gây tốn kém nhiều về chi phí, thời gian và nguồn nhân lực mà không đạt được hiệu quả và năng suất.

Những năm gần đây, trí tuệ nhân tạo (AI) dần nổi lên như một minh chứng cho cuộc cách mạng công nghiệp lần thứ tư, sau động cơ hơi nước, năng lượng điện và công

nghệ thông tin. Trí tuệ nhân tạo đã và đang trở thành nhân tố cốt lõi trong các hệ thống công nghệ cao. Thậm chí, nó đã len lỏi vào hầu hết các lĩnh vực của đời sống mà có thể chúng ta không nhận ra. Nhằm mục đích dễ dàng vận chuyển, phân loại, giảm thiểu thời gian xử lý, bài toán nhận diên hoa quả đã ra đời. Việc ứng dụng AI vào việc nhận diên trái cây đã dần phổ biến trên thế giới. Tuy nhiên bên cạnh đó cũng gặp nhiều khó khăn trong việc xây dựng hệ thống thi công phức tạp và chi phí xây dựng đắt đỏ. Trong đề tài này tác giả bước tiếp các vấn đề về lĩnh vực trí tuệ nhân tạo đã nêu áp dụng trong ngành khoa học máy tính để hoàn thiện và cải tiến, giúp người dùng có thể tiếp cận và sử dụng trong cuộc sống hàng ngày một cách dễ dàng.

Tác giả hướng tới xây dựng mô hình mạng Nơ-ron tích chập (Convolutional Neural Network) dùng ứng dụng trong phân loại trái cây, xử lý tập dữ liệu bao gồm 47745 hình ảnh về 70 loại trái cây phổ biến trên thế giới. Tác giả tối ưu mô hình xây dựng dựng ở trên đưa ra kết quả dự đoán với độ chính xác không dưới 80%, có thể phân loại trái cây thời gian thực (realtime) bằng camera trên máy tính. Bên cạnh đó, Sử dụng google colaboratory để xử lí dữ liệu, xây dựng mô hình mạng nơ-ron tích chập (Convolutional Neural Network), đánh giá mô hình đưa ra độ chính xác và xử lí dữ liệu thời gian thực.

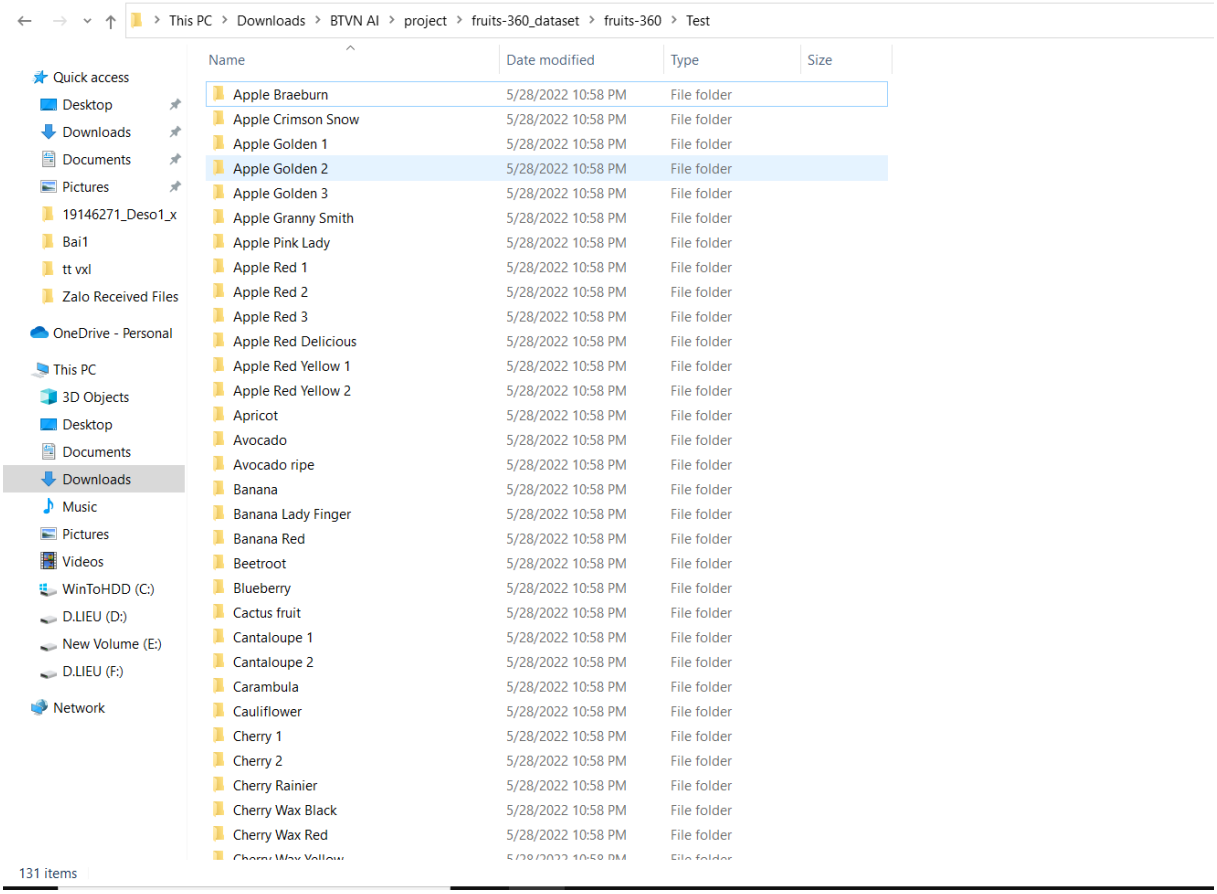
***Phương pháp và kết quả***

+Thiết kế mô hình

Google Colaboratory (gọi tắt là Google Colab hay Colab) là một sản phẩm của Google Research. Colab dựa trên Jupyter Notebook, người dùng có thể viết và thực thi đoạn mã python thông qua trình duyệt và đặc biệt rất phù hợp với data analysis, machine learning và giáo dục.

Google Colab là một sản phẩm từ Google Research, nó cho phép chạy các dòng code python thông qua trình duyệt, đặc biệt phù hợp với Data analysis, machine learning và giáo dục. Colab không cần yêu cầu cài đặt hay cấu hình máy tính, mọi thứ có thể chạy thông qua trình duyệt, có thể sử dụng tài nguyên máy tính từ CPU tốc độ cao và cả GPUs và cả TPUs đều được cung cấp. Colab cung cấp nhiều loại GPU, thường là Nvidia K80s, T4s, P4s and P100s, tuy nhiên người dùng không thể chọn loại GPU trong Colab, GPU trong Colab thay đổi theo thời gian.

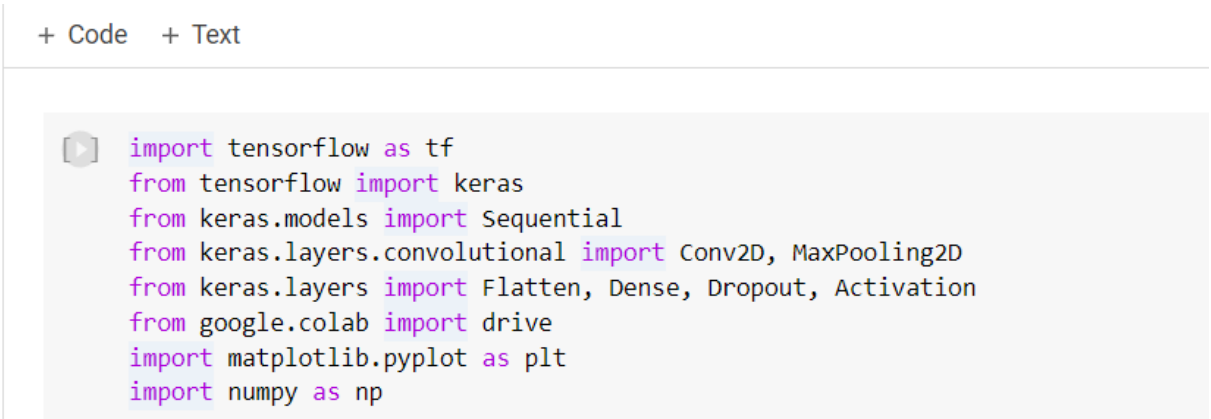
Sau khi tải tập dữ liệu bao gồm 70 class và 47745 dữ liệu hình ảnh từ kaggle ta có thể đánh giá sơ qua bộ giữ liệu.




*Hình 1. 70 classes từ tập dữ liệu.*

## Truy xuất dữ liệu từ drive và chia thành các tập train, test

Khai báo một số thư viện cần thiết:



Truy xuất tập dữ liệu đã tải từ google 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)
    train=train_datagen.flow_from_directory('/content/drive/MyDrive/train10/Training',
                                                    target_size=(256,256),
                                                    batch_size=32,
                                                    class_mode ='categorical')
    test=train_datagen.flow_from_directory('/content/drive/MyDrive/test10/Test',
                                                    target_size=(256,256),
                                                    batch_size=32,
                                                    class_mode ='categorical')
```

```
Mounted at /content/drive
Found 35755 images belonging to 70 classes.
Found 11990 images belonging to 70 classes.
```

Phân chia dữ liệu đã tải thành 2 tập là train và test với tập test có giá trị bằng 25% giá trị của tập train bằng train_test_split từ thư viện sklearn.

Có thể thấy được sau khi tách ta được 35755 dữ liệu thuộc 70 classes ở tập dữ liệu train và 11990 dữ liệu thuộc 70 classes ở tập dữ liệu test.

**Xây dựng mô hình**

Xây dựng mô hình CNN bằng Sequential từ thư viện keras.models với 1 lớp input với dữ liệu đầu vào là dữ liệu với kích thước 256x256x3, 3 lớp ẩn và lớp output với 70 classes đầu ra:

```
model=Sequential()
model.add(Conv2D(128,(3,3),activation='relu',kernel_initializer='he_uniform',padding='same',input_shape=(256,256,3)))
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(70,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)]
```

Huấn luyện với mô hình đã tạo với tập huấn luyện là train, tập thẩm định là test, batch size =64, số lần huấn luyện là 8 lần:

Kết quả thu được ở năm lần huấn luyện thu được ta có thể thấy độ chính xác đối với tập train là 90% và độ chính xác của mô hình khi chạy với tập thẩm định là gần 70%

```
Epoch 1/8
648/648 [==============================] - 9415s 15s/step - loss: 2.2061 - accuracy: 0.4328 - val_loss: 0.9558 - val_accuracy: 0.7341
Epoch 2/8
648/648 [==============================] - 443s 684ms/step - loss: 0.2851 - accuracy: 0.9110 - val_loss: 0.7096 - val_accuracy: 0.8194
Epoch 3/8
648/648 [==============================] - 439s 678ms/step - loss: 0.1370 - accuracy: 0.9597 - val_loss: 0.7081 - val_accuracy: 0.8501
Epoch 4/8
648/648 [==============================] - 440s 679ms/step - loss: 0.0799 - accuracy: 0.9755 - val_loss: 0.3884 - val_accuracy: 0.9107
Epoch 5/8
648/648 [==============================] - 438s 677ms/step - loss: 0.0840 - accuracy: 0.9763 - val_loss: 0.2917 - val_accuracy: 0.9245
Epoch 6/8
648/648 [==============================] - 440s 679ms/step - loss: 0.0605 - accuracy: 0.9830 - val_loss: 0.5715 - val_accuracy: 0.8852
Epoch 7/8
648/648 [==============================] - 438s 676ms/step - loss: 0.0613 - accuracy: 0.9841 - val_loss: 0.4342 - val_accuracy: 0.9226
Epoch 8/8
648/648 [==============================] - 438s 676ms/step - loss: 0.0591 - accuracy: 0.9849 - val_loss: 0.4348 - val_accuracy: 0.9281
```

**Đánh giá mô hình**

Từ kết quả mô hình sau khi huấn luyện như trên ta có thể đánh giá sơ qua kết quả của mô hình như sau: khi huấn luyện độ chính xác chính xác của mô hình khá cao đạt gần 93% ở kết quả thu được nhưng khi thẩm định thì kết quả thu được độ chính xác chỉ giao động ở 80% cho thấy được chất lượng mô hình và data chưa thực sự tối ưu nhưng vẫn trong khoảng chấp nhận được.

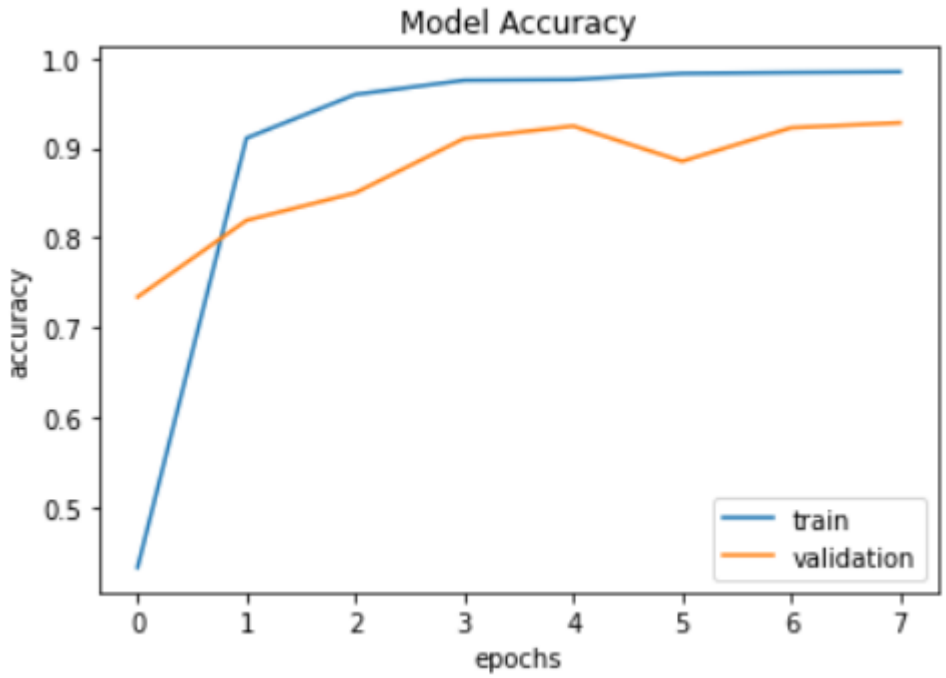
```
#đánh giá chất lượng của mô hình và vẽ lại
score = model.evaluate(test,verbose=0)
print('Sai số kiểm tra là: ',score[0])
print('Độ chính xác kiểm tra là: ',score[1])
```

```
Sai số kiểm tra là:  0.4282713830471039
Độ chính xác kiểm tra là:  0.9273504018783569
```

Tiến hành vẽ biểu đồ thể hiện giá thu được sau mỗi lần huấn luyện với thư viện matplotlib:

```
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'],loc='upper_left')
```

Kết quả thu được:




*Hình 5. Đồ thị biểu diễn độ chính xác của mô hình*

**Thử nghiệm kết quả thu được**

Ta tiến hành thử nghiệm kết quả thu được của mô hình đã huấn luyện bằng cách tải mô hình đã huấn luyện bằng load_model từ thư viện keras

```
[ ]  from tensorflow.keras.models import load_model
     model=load_model('model_fruit70.h5')
```

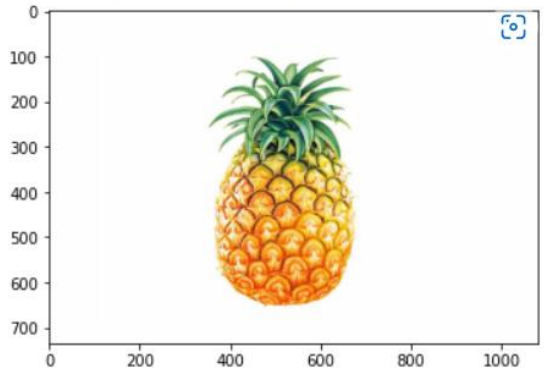
Tiến hành đặt tên cho 70 class ứng với 70 loại trái cây:

```
fruit = ['Apple Braeburn','Grape Pink','Hazelnut','Kaki','Kiwi','Kohlrabi','Kumquats','Lemon Meyer',
         'Limes','Lychee','Mandarine','Mango Red','Mangostan','Maracuja','Melon Piel de Sapo','Mulberry',
         'Nectarine','Nectarine Flat','Nut Forest','Nut Pecan','Onion Red','Onion Red Peeled','Onion White',
         'Orange','Papaya','Passion Fruit','Peach','Peach Flat','Pear','Pear Abate','Pear Forelle','Pear Kaiser',
         'Pear Monster','Pear Red','Pear Stone','Pear Williams','Pepino','Pepper Green','Pepper Orange','Pepper Red',
         'Pepper Yellow','Physalis','Physalis with Husk','Pineapple','Pineapple Mini','Pitahaya Red','Plum',
         'Pomegranate','Pomelo Sweetie','Potato Red','Potato Red Washed','Potato Sweet','Potato White',
         'Quince','Rambutan','Raspberry','Redcurrant','Salak','Strawberry','Strawberry Wedge','Tamarillo',
         'Tangelo','Tomato 1','Tomato Cherry Red','Tomato Heart','Tomato Maroon','Tomato not Ripened',
         'Tomato Yellow','Walnut','Watermelon']
```

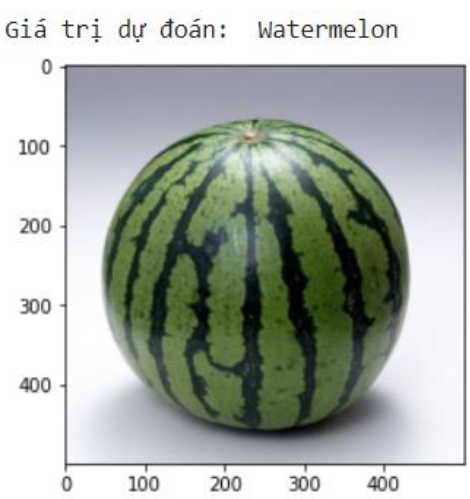
Sử dụng load_img và img_to_array từ thư viện tensorflow để thử nghiệm với các kết quả bên ngoài:

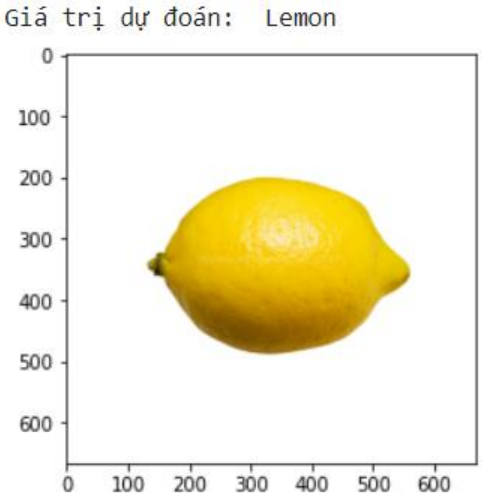
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from keras.preprocessing import image
%matplotlib inline
uploaded=files.upload()
for fn in uploaded.keys():
  #predicting images
  path='/content/' + fn
  #In ảnh đọc được
  plt.imshow(mpimg.imread(path))
  img=image.load_img(path,target_size=(256,256))
  x=image.img_to_array(img)
  x=np.expand_dims(x,axis=0)
  images=np.vstack([x])
  y_predict = model.predict(images,batch_size=10)
  print(y_predict)
  print('Giá trị dự đoán: ', fruit[np.argmax(y_predict)])
```

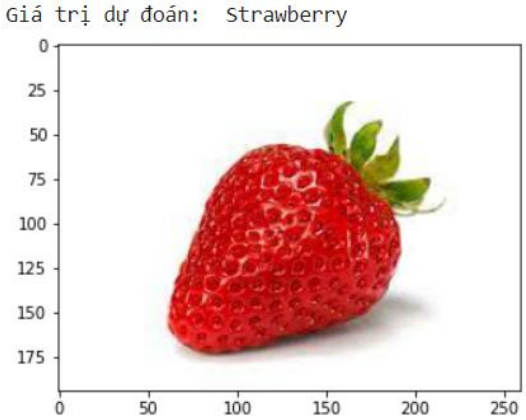
```
Giá trị dự đoán:  Pear Abate
```

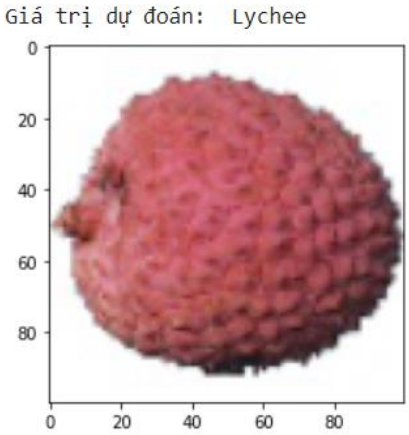


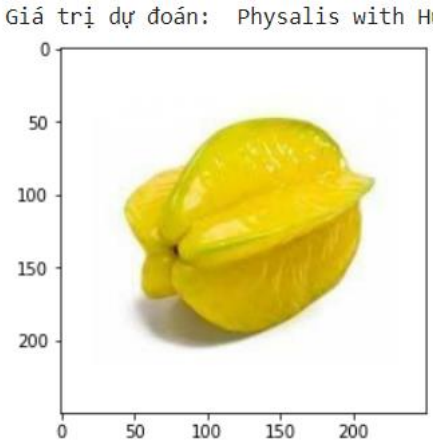

```
Giá trị dự đoán:  Pitahaya Red
```

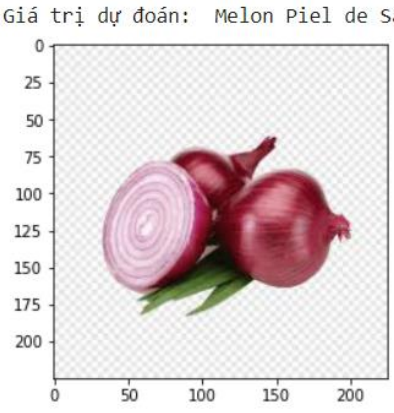



Có thể thấy được từ 8 mẫu thử bất kì như trên mô hình đem lại độ chính xác không quá cao nhưng vẫn trong khoảng có thể chấp nhận được.