ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN CUỐI KỲ

MÔN: Thị giác máy tính nâng cao

ĐỀ TÀI: Tomato leaf disease detection

Lớp: CS331.N12

Giảng viên: Mai Tiến Dũng

Sinh viên thực hiện:

Hồ Hồng Hà – 20520480

Hà Văn Linh – 20521529

TP. HỒ CHÍ MINH, THÁNG 1 NĂM 2023

I. Tổng quan đề tài

- Trong nông nghiệp thì không thể tránh khỏi những loại cây trồng bị mắc bệnh, đặc biệt là các bệnh trên lá cây.
- Hiện nay, cà chua là một loại cây trồng khá phổ biến, có nhu cầu sử dụng cao. Nếu lá cây bị bệnh sẽ gây ảnh hưởng rất lớn đến chất lượng và năng suất của cây.
- Do đó cần phải chuẩn đoán bệnh để đưa ra giải pháp phù hợp. Nếu chuẩn đoán bệnh thực hiện bởi con người thì sẽ dẫn đến có nhiều sai sót do số lượng bệnh rất nhiều và có nhiều loại bệnh có biểu hiện khá giống nhau, tốn kém chi phí do phải được thực hiện bởi người có chuyên môn…
- Từ đó, cần phải xây dựng một hệ thống giúp chuẩn đoán bệnh trên lá cây cà chua để hạn chế sai sót.
- Input và output của bài toán:
  + Input: Ảnh chụp lá cây cà chua
  + Output: Loại bệnh được dự đoán là lá cây cà chua đó mắc phải

II. Dữ liệu

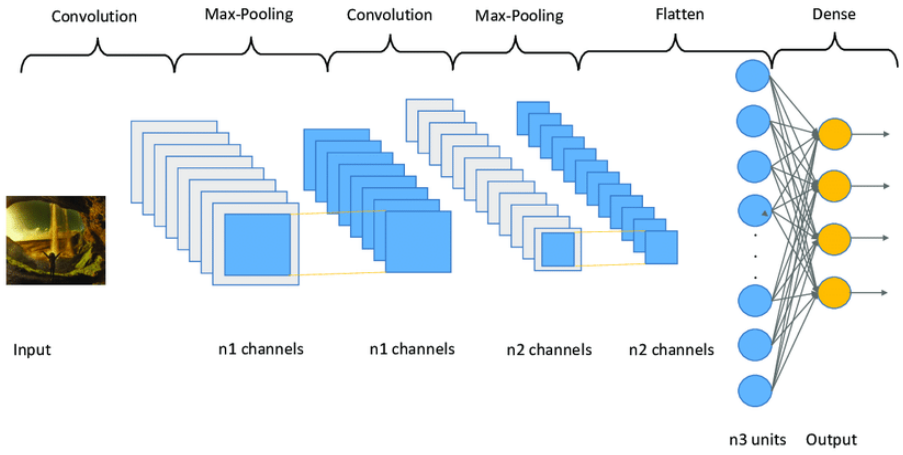
- Bộ dữ liệu được lấy từ Kaggle: Tomato Leaf Disease Detection
- Bộ dữ liệu gồm 11000 ảnh: 10000 ảnh trong tập train và 1000 trong tập val
- Số lượng ảnh được chia đều cho 10 class:
    - Tomato___Bacterial_spot
    - Tomato___Early_blight
    - Tomato___Late_blight
    - Tomato___Leaf_Mold
    - Tomato___Septoria_leaf_spot
    - Tomato___Spider_mites Two-spotted_spider_mite
    - Tomato___Target_Spot
    - Tomato___Tomato_Yellow_Leaf_Curl_Virus
    - Tomato___Tomato_mosaic_virus
    - Tomato___healthy
- Để thực hiện cho đồ án nhóm tiến hành chia bộ dữ liệu trên thành 3 tập
    - Train: 8000 ảnh được lấy từ tập train ban đầu
    - Val: 2000 ảnh được lấy từ tập train ban đầu
    - Test: 1000 ảnh là tập val ban đầu
- Sử dụng ImageDataGenerator để tăng thêm sự đa dạng dữ liệu .
- Một số hình ảnh trong tập dữ liệu:



III. Phương pháp giải quyết

1. Xây dựng model CNN tự do:

- Convolutional Neural Network (CNNs – Mạng nơ-ron tích chập) là một trong những mô hình Deep Learning tiên tiến. Nó giúp cho chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao.



- Cài đặt:

```
CNN_model = Sequential()
inputShape = (256, 256, 3)
chanDim = -1
if K.image_data_format() == "channels_first":
    inputShape = (3, 256, 256)
    chanDim = 1
CNN_model.add(Conv2D(32, (3, 3), padding="same",input_shape=inputShape))
CNN_model.add(Activation("relu"))
CNN_model.add(BatchNormalization(axis=chanDim))
CNN_model.add(MaxPooling2D(pool_size=(3, 3)))
CNN_model.add(Dropout(0.25))
CNN_model.add(Conv2D(64, (3, 3), padding="same"))
CNN_model.add(Activation("relu"))
CNN_model.add(BatchNormalization(axis=chanDim))
CNN_model.add(Conv2D(64, (3, 3), padding="same"))
CNN_model.add(Activation("relu"))
CNN_model.add(BatchNormalization(axis=chanDim))
CNN_model.add(MaxPooling2D(pool_size=(2, 2)))
CNN_model.add(Dropout(0.25))
CNN_model.add(Conv2D(128, (3, 3), padding="same"))
CNN_model.add(Activation("relu"))
CNN_model.add(BatchNormalization(axis=chanDim))
CNN_model.add(Conv2D(128, (3, 3), padding="same"))
CNN_model.add(Activation("relu"))
CNN_model.add(BatchNormalization(axis=chanDim))
CNN_model.add(MaxPooling2D(pool_size=(2, 2)))
CNN_model.add(Dropout(0.25))
CNN_model.add(Flatten())
CNN_model.add(Dense(1024))
CNN_model.add(Activation("relu"))
CNN_model.add(BatchNormalization())
CNN_model.add(Dropout(0.5))
CNN_model.add(Dense(n_classes))
CNN_model.add(Activation("softmax"))
```

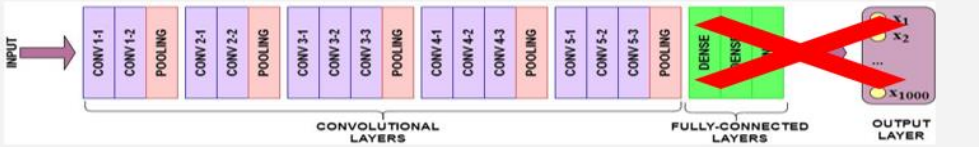
Đây là mô hình CNN tự do. Mô hình này bao gồm nhiều lớp Conv2D để tính toán các bộ lọc (filter) trên hình ảnh đầu vào, sau đó sử dụng các lớp Activation, BatchNormalization và MaxPooling2D để xử lý dữ liệu đầu ra từ

các lớp Conv2D. Mô hình còn sử dụng các lớp Dropout để giảm overfitting và lớp Dense để xử lý dữ liệu đầu ra từ lớp MaxPooling2D và đưa ra dự đoán cuối cùng.

2. Trainsfer learning sử dụng VGG16:

- Trainsfer learning là sử dụng mô hình đã được đào tạo trước đó để giải quyết bài toán mới tương tự.
- VD: Bạn có bài toán cần nhận diện 1000 người nổi tiếng ở Việt Nam, tuy nhiên dữ liệu để train chỉ khoảng 10 ảnh / 1 người. Số lượng dữ liệu là quá ít để train một mô hình CNN hoàn chỉnh. Bạn tìm trên mạng thấy VGGFace2 dataset có 3.31 triệu ảnh của 9131 người, với trung bình 362.6 ảnh cho mỗi người. Họ làm bài toán tương tự đó là nhận diện ảnh từng người và họ đã train được CNN model với accuracy hơn 99%. Bạn nhớ ra là trong convolutional neural network, convolutional layer có tác dụng lấy ra các đặc trưng của ảnh, và sau hàng loạt các convolutional layer + pooling layer (ConvNet) thì model sẽ học được các đặc điểm của ảnh, trước khi được cho vào fully connected layer => ConvNet trong VGGFace2 model cũng lấy ra được các đặc điểm của mặt người (tai, mũi, tóc,...) => Ta cũng có thể áp dụng phần ConvNet của VGGFace2 model vào bài toán nhận diện mặt người nổi tiếng ở Việt Nam để lấy ra các đặc điểm của mặt.
- VGG16 là mạng convolutional neural network được đề xuất bởi K. Simonyan and A.Zisserman, University of Oxford. Model sau khi train bởi mạng VGG16 đạt độ chính xác 92.7% top-5 test trong dữ liệu ImageNet gồm 14 triệu hình ảnh thuộc 1000 lớp khác nhau. Giờ áp dụng kiến thức ở trên để phân tích mạng VGG 16.

- Phân tích:
  - Convolutional layer: kích thước 3*3, padding=1, stride=1.
  - Pool/2 : max pooling layer với size 2*2
  - 3*3 conv, 64: thì 64 là số kernel áp dụng trong layer đấy, hay depth của output của layer đấy.
  - Càng các convolutional layer sau thì kích thước width, height càng giảm nhưng depth càng tăng.
- Loại bỏ FCs và chỉ giữ lại phần ConvNet trong model VGG16




```
input_1 (InputLayer)         [(None, 224, 224, 3)]     0
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
```

- Fatten output của ConvNet trong model
- Thêm fully connected(FC)

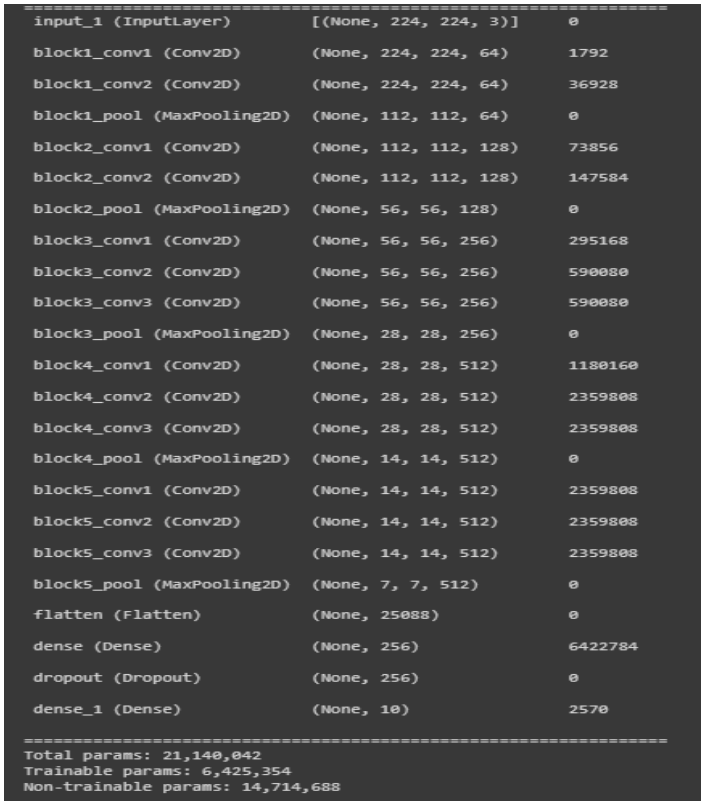
```
x = Dense(256, activation='relu')(x)
x = Dropout(0.5)(x)
```

- Thêm Output layer với softmax activation

```
x = Dense(10, activation='softmax')(x)
vgg16_model=Model(inputs=model.input,outputs=x)
```

- Train model với Adam optimizer, Learning Rate = 0.0001, batch_size = 32
- Giai đoạn 1: Đóng băng/Freeze các ConvNet, chỉ train các layer mới thêm vào với số epoch là 25

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



```
=================================================================
input_1 (InputLayer)         [(None, 224, 224, 3)]     0
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
flatten (Flatten)            (None, 25088)             0
dense (Dense)                (None, 256)               6422784
dropout (Dropout)            (None, 256)               0
dense_1 (Dense)              (None, 10)                2570
=================================================================
Total params: 21,140,042
Trainable params: 6,425,354
Non-trainable params: 14,714,688
```

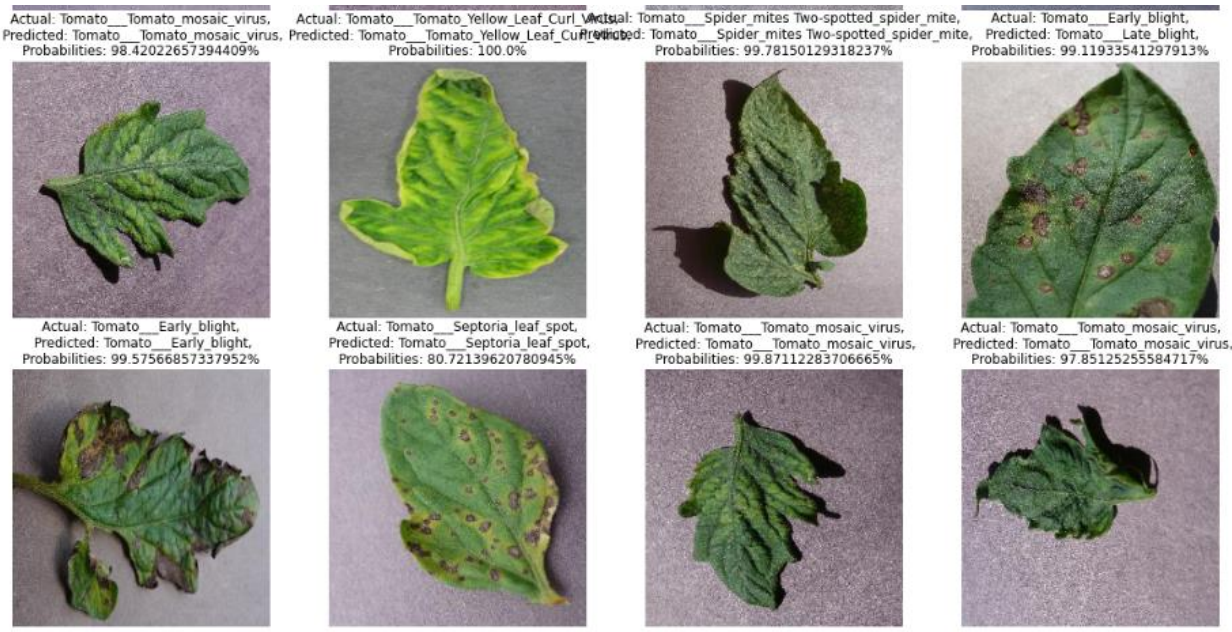
- Giai đoạn 2: Unfreeze các layer của pre-trained model, train ở các layer trong ConvNet của pre-trained model và các fully connected layer mới

```
for layer in model.layers[15:]:
  layer.trainable=True
```

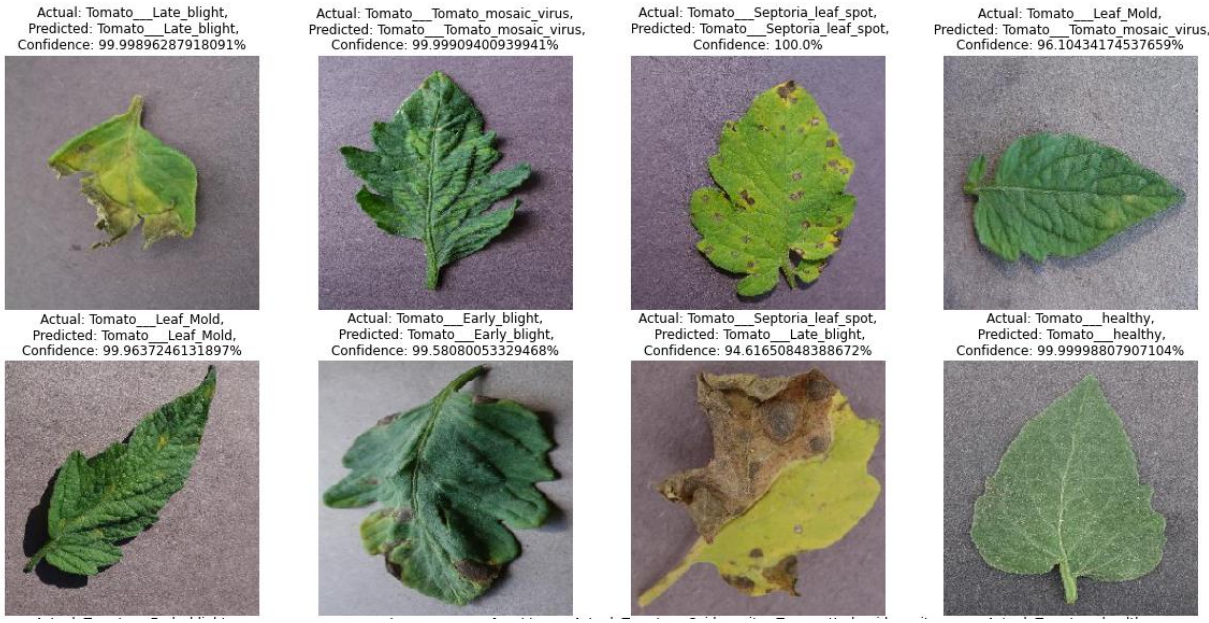
```
input_1 (InputLayer)        [(None, 224, 224, 3)]     0
block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792
block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928
block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0
block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856
block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584
block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0
block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168
block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080
block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080
block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0
block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160
block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808
block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808
block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0
block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808
block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808
block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808
block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0
flatten (Flatten)           (None, 25088)             0
dense (Dense)               (None, 256)               6422784
dropout (Dropout)           (None, 256)               0
dense_1 (Dense)             (None, 10)                2570
=================================================================
Total params: 21,140,042
Trainable params: 13,504,778
Non-trainable params: 7,635,264
```

IV. Kết quả đánh giá

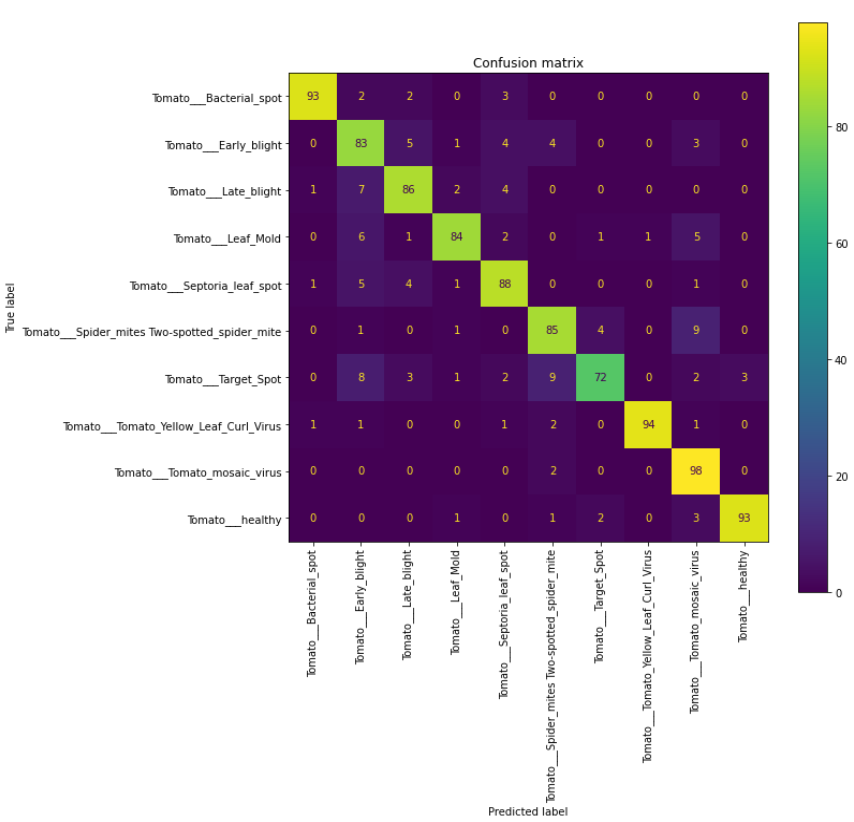
- **Model CNN tự do:**
    - Accuracy trên tập test đạt 89.9%
    - Một số hình ảnh test:



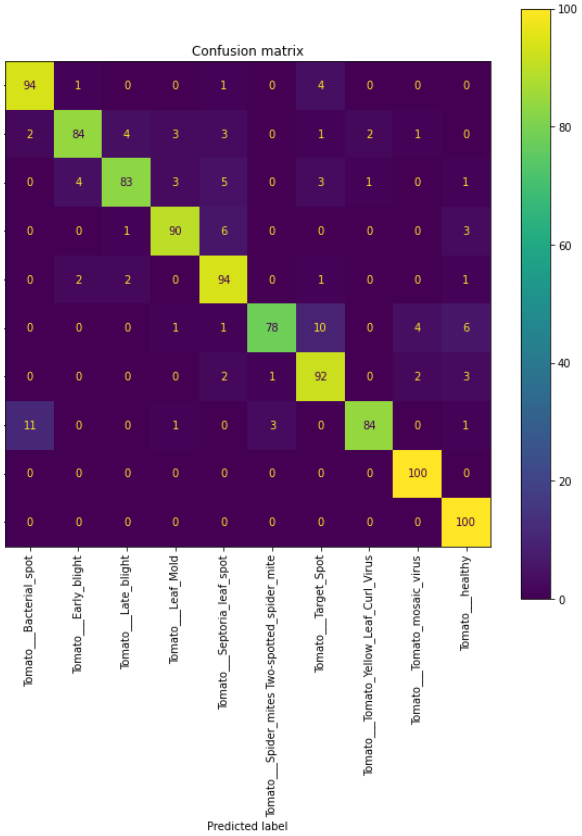
- **Trainsfer learning sử dụng VGG16:**
  - Accuracy trên tập test đạt 87.6%
  - Một số hình ảnh test:



Confusion matrix VGG16



Confusion matrix model CNN tự do



Nhận xét:

- Dựa vào confussion matrix có thể thấy rằng có một số lớp có sự khác biệt rất lướn giữa 2 model
- Vd:

  + ở phân lớp Tomato___Target_Spot model VGG16 có kết quả thấp hơn nhiều so với CNN tự do

  + ở phân lớp Tomato___Spider_mites Two-spotted_spider_mite VGG16 có kết quả tốt hơn so với CNN tự do

V. Tài liệu tham khảo

https://github.com/marcosdhiman/leaf_disease_detection
https://www.geeksforgeeks.org/vgg-16-cnn-model/
https://viblo.asia/p/transfer-learning-va-bai-toan-face-recognition-3Q75w7xD5Wb