Đại học quốc gia TP.HCM
Trường đại học công nghệ thông tin

Báo Cáo Đồ Án

Ngành Khoa học máy tính

Môn học: Thị giác máy tính nâng cao

Lớp: CS331.N21.KHCL

Đề tài: Sử dụng mạng học sâu để nhận biết loại bệnh ở cây cà chua

---

Sinh viên:
Quách Minh Triết – 20522057

Giảng viên:
Mai Tiến Dũng

Contents

1 Giới thiệu

1.1 Mô tả bài toán

Bài toán phân loại loại bệnh của cây cà chua là một bài toán quan trọng trong lĩnh vực nông nghiệp ứng dụng thị giác máy tính. Với mục đích phân loại, nhận diện các loại bệnh một cách nhanh chóng và chính xác để đưa ra các biện pháp phòng và xử lý thích hợp giúp nông dân tăng hiệu suất sản xuất, giảm các tổn thất trong quá trình trồng trọt.

Tính đến hiện tại, các phương pháp phân loại loại bệnh của cây cà chua chủ yếu được thực hiện bởi con người thông qua việc phân tích ảnh và đặt chẩn đoán dựa trên các triệu chứng hiển thị trên thành phần của cây. Tuy nhiên, phương pháp này không chỉ tốn kém về thời gian và nhân lực mà còn không đảm bảo tính chính xác và hiệu quả.

Vì vậy, sử dụng ứng dụng của thị giác máy tính trở thành một phương pháp khả thi trong việc phân loại loại bệnh cây cà chua. Thị giác máy tính được xây dựng dựa trên các mô hình học sâu, lớp tích chập, và các phương pháp deep learning khác, giúp phân tích dữ liệu ảnh một cách nhanh chóng và chính xác, cho phép nhận diện và đánh giá các loại bệnh khác nhau của cây cà chua.

Việc sử dụng ứng dụng của thị giác máy tính trong phân loại loại bệnh cây cà chua giúp giảm thiểu các sai sót do con người phát sinh, cải thiện độ chính xác của phương pháp, tiết kiệm thời gian và chi phí. Ngoài ra, đối với nông dân, sử dụng ứng dụng này còn giúp tăng hiệu suất sản xuất cây cảnh, giảm thiểu tổn thất và hỗ trợ phát triển nông nghiệp thông minh.

Tổng quan, hiện nay ứng dụng của thị giác máy tính như mạng học sâu, lớp tích chập và các phương pháp deep learning đang được sử dụng rộng rãi để phân loại loại bệnh cây cà chua. Các phương pháp này cho phép nông dân nhận diện và phân loại các loại bệnh một cách nhanh chóng và chính xác, tăng hiệu suất sản xuất và cải thiện đời sống của nông dân.

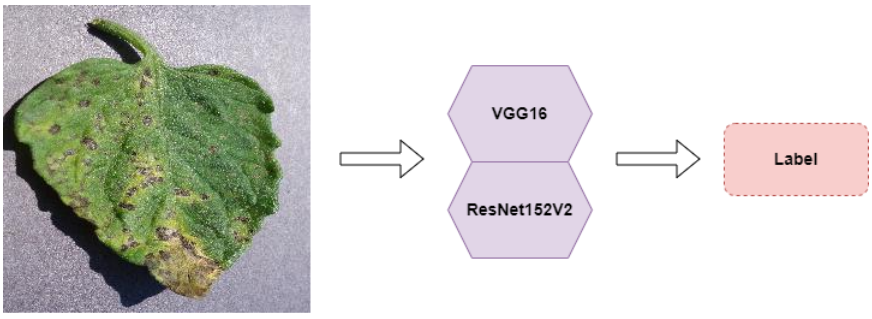


Figure 1: Mô tả bài toán

1.2 Tầm quan trọng của bài toán

Việc áp dụng các công nghệ mới như thị giác máy tính và các mô hình học sâu là cần thiết và quan trọng để giải quyết các thách thức liên quan đến sản lượng và hiệu suất trong ngành nông nghiệp hiện đại. Tầm quan trọng của bài toán phân loại loại bệnh của cây cà chua không chỉ giúp giảm thiểu chi phí và tăng hiệu suất sản xuất, mà còn giúp tăng cường độ chính xác và bảo đảm chất lượng sản phẩm, góp phần phát triển nông nghiệp bền vững.

Việc giải quyết bài toán này có thể mở ra cánh cửa tự do cho việc phân loại loại bệnh của cây cà chua và giúp giảm tình trạng chết cây và giảm tổn thất do bệnh tật do phân loại nhanh chóng, chính xác và đầy đủ. Các ứng dụng này cho phép việc phân loại trở nên đơn giản hơn mà không cần hiểu rõ cách thức, tiết kiệm thời gian và phát hiện bệnh rất sớm giúp người trồng cây có những quyết định chính xác, bảo vệ cà chua, cải thiện và bảo vệ môi trường.

1.3 Input và Output của bài toán

Input:

- Đầu vào là một bức ảnh màu.
- Chỉ chứa duy nhất một lá cà chua (thuộc một trong số 10 loại bệnh).
- Kích thước ảnh tối thiểu là 224x224 và tối đa là 256x256.
- Độ phân giải tối thiểu là 96x96 pixel.
- Lá cà chua phải nằm giữa bức ảnh chiếm ít nhất 50% diện tích bức ảnh.
- Background có duy nhất một màu và nên tránh dùng background có màu vàng hoặc xanh lá để hạn chế việc gây nhiễu

Output:

- Nhãn thuộc một trong 10 lớp của bức ảnh đầu vào.

2 Bộ dữ liệu

2.1 Giới thiệu về bộ dữ liệu

Tập dữ liệu mà chúng tôi sử dụng trong bài bái cáo này là New Plant Diseases Dataset(Augmented). Bộ data này chứa tổng cộng **22930** bức ảnh. Đây là một bộ dữ liệu có sẵn trên Kaggle đã được đông đảo người trong cộng đồng Data Science sử dụng trong việc đào tạo mô hình.

Bộ dữ liệu gồm có 3 tập là train, valid và test, trong mỗi tập lại có 10 class tương ứng với 10 loại bệnh khác nhau ở cà chua mà chúng ta sẽ phân lớp, bao gồm:

1. Bệnh đốm vi khuẩn(Bacterial spot)
2. Bệnh bạc lá sớm(Early blight)
3. Khoẻ mạnh(Healthy)
4. Bệnh héo(úa) muộn(Late blight)
5. Bệnh mốc lá (Leaf Mold)
6. Bệnh đốm lá(Septoria leaf spot)
7. Bệnh ve nhện(Spider mites Two-spotted spider mite)
8. Bệnh đốm mục tiêu(đốm vòng mạch gỗ)(Target Spot)
9. Bệnh "xoăn lá" do virus khảm cà chua(Tomato mosaic virus)
10. Bệnh vọng xoắn vàng lá(Tomato Yellow Leaf Curl Virus)

Về số lượng ảnh cụ thể trong mỗi tập dữ liệu được thể hiện qua bảng sau:

Table 1: Bảng thống kê số lượng ảnh trong mỗi tập dữ liệu

Class name	**Train**	**Valid**	Test
Bacterial spot	1191	340	171
Early blight	1316	384	192
Healthy	1348	385	193
Late blight	1295	370	186
Leaf Mold	1317	376	189
Septoria leaf spot	1221	349	175
Spider mites Two-spotted spider mite	1218	348	175
Target Spot	1264	365	184
Tomato mosaic virus	1253	358	179
Tomato Yellow Leaf Curl Virus	1372	392	197
Total	**12795**	**3667**	**1841**

Bacterial spot | Early blight | Healthy | Late blight | Leaf Mold

Septoria leaf spot | Spider mites Two-spotted spider mite | Target Spot | Tomato mosaic virus | Tomato Yellow Leaf Curl Virus

Table 2: 10 loại bệnh ở cây cà chua

2.2 Tiền xử lý dữ liệu

1. Sử dụng lớp ImageDataGenerator để tạo ra các hình ảnh mới từ tập dữ liệu huấn luyện hiện có thông qua các phép biến đổi khác nhau như quay, dịch chuyển, co giãn, phóng to/giảm, lật ngang, ... Những tham số sử dụng trong class ImageDataGenerator là:

   - $rescale = 1./255$: Tiêu chuẩn hóa giá trị pixel của ảnh bằng cách chia tất cả các giá trị pixel cho 255.
   - $rotation_range = ROT\_RANGE$: Trong đó ROT_RANGER bằng 10, số độ quay ảnh tối đa (0-180 độ).
   - $width\_shift\_range = 0.2$: Phạm vi dịch chuyển theo chiều rộng của hình ảnh.
   - $height\_shift\_range = 0.2$: Phạm vi dịch chuyển theo chiều cao của hình ảnh.
   - $shear\_range = 0.2$: Độ uốn ảnh.
   - $zoom\_range = 0.2$: Phạm vi phóng to hoặc thu nhỏ ảnh.
   - $horizontal\_flip = True$: Lật ngang ảnh.
   - $fill\_mode =' nearest'$: Chiến lược để điền giá trị cho các chỗ trống khi phép biến đổi hình ảnh di chuyển vào vùng không có giá trị.

   Trong quá trình huấn luyện, lớp ImageDataGenerator sẽ tạo ra các trường hợp mới từ tập dữ liệu huấn luyện thay vì sử dụng các hình mẫu dữ liệu ban đầu để đưa ra các quyết định. Điều này giúp mô hình đào tạo học cách giải quyết các bài toán trong một loạt các tình huống khác nhau.

2. Tiếp theo ta sử dụng các trờng hợp mới vừa được tạo ra để tạo ra dữ liệu huấn luyện dưới dạng luồng từ thư mục được chỉ định trong biến $TRAINING\_DIR$. Các tham số sử dụng trong phương thức $flow\_from\_directory()$ như sau đối vối với cả 3 tập train, valid và test:

   - $target\_size = (224, 224)$: kích thước đầu vào của những tấm hình huấn luyện, trong trường hợp này là 224x224.

- $class\_mode = 'categorical'$: loại của các nhãn, trong trường hợp này, chúng ta muốn phân loại thành các lớp được đánh số từ 0 đến 9.
- $color\_mode = 'rgb'$: ảnh được đọc với 3 kênh màu red, green, blue.
- $batch\_size = 128$ đối với tập train và $batch\_size = 32$ đối với tập valid là kích thước của mỗi batch dữ liệu, giúp giảm thời gian tính toán và làm tăng khả năng tối ưu của mô hình.
- $shuffle = True$ đối với train và valid để dữ liệu được xáo trộn ngẫu nhiên trước khi chia thành các lô (batches) cho huấn luyện mỗi lần lặp lại (epoch). $shuffle = False$ đối với tập test.

3 2 mô hình học sâu được sử dụng trong đồ án

3.1 VGG-16

VGG16 là một mô hình mạng nơ-ron tích chập được đề xuất bởi K. Simonyan và A. Zisserman từ Đại học Oxford trong bài báo "Very Deep Convolutional Networks for Large-Scale Image Recognition". Nó đứng đầu trong cuộc thi hình ảnh ImageNet 2014, đạt được accuracy đáng kinh ngạc ở mức 92,7%. Mạng VGG16 được thiết kế để phân loại hình ảnh và bao gồm 16 lớp tích chập, các lớp kích hoạt phi tuyến và lớp fully connected layer.

Cấu trúc của mạng VGG16 bao gồm:

1. Input layer: Là layer đầu tiên của mạng, layer này nhận được dữ liệu đầu vào hình ảnh.
2. Convolutional layer: Là layer chính của mạng, sử dụng các bộ lọc tích chập để trích xuất các đặc trưng từ hình ảnh đầu vào.
3. Activation layer: Là layer sử dụng hàm kích hoạt phi tuyến để tạo ra các non-linearities để tăng tính phi tuyến của mạng.
4. Max pooling layer: Là layer sử dụng để giảm kích thước của đầu ra từ các lớp tích chập, giảm độ phức tạp của mô hình và giúp tăng tốc độ tính toán.

5. Fully connected layer: Là layer kết nối hoàn toàn, tạo ra các đầu ra phân loại.

Mạng VGG16 sử dụng các lớp tích chập có kernel là 3x3, stride = 1 và chọn padding sao cho kích thước ảnh input bằng với kích thước ảnh output. Pooling layer được sử dụng là loại có kích thước 2x2, stride = 2 và là max pooling. Cụ thể:

- 2 bước CONV liên tiếp, mỗi bước gồm 64 filter
- 1 max pooling layer
- 2 bước CONV liên tiếp, mỗi bước gồm 128 filter
- max pooling layer
- 3 bước CONV liên tiếp, mỗi bước gồm 256 filter
- max pooling layer
- 3 bước CONV liên tiếp, mỗi bước gồm 512 filter
- max pooling layer
- 3 bước CONV liên tiếp, mỗi bước gồm 512 filter
- max pooling layer
- 3 lớp fully connected layer mỗi lớp có 4096 node

Cuối cùng, hàm softmax để tính xác suất của các lớp đầu ra.

Trong quá trình huấn luyện, mạng VGG16 sử dụng kỹ thuật dropout để giảm quá khớp và giúp mạng học được các đặc trưng chung của tập dữ liệu, làm tăng tính tổng quát của mạng.

Số 16 trong cái tên VGG-16 là đang nói đến 16 layer có tham sô có thể học được trong mô hình, bao gồm: 13 convolutional layer và 3 fully connected layer.

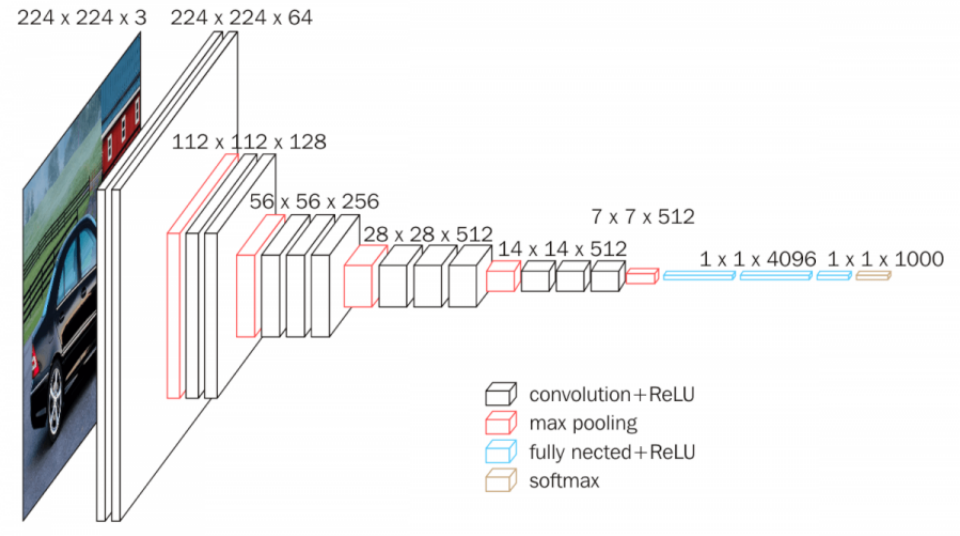


Figure 2: Minh hoạ kiến trúc VGG-16

3.2 ResNet-152

ResNet152 là một kiến trúc mạng neuron sâu (deep neural network) được đề xuất bởi Kaiming He, Xiangyu Zhang, Shaoqing Ren và Jian Sun trong bài báo có sức ảnh hưởng lớn "Deep Residual Learning for Image Recognition" tại Microsoft Research vào năm 2015. Đây là một phiên bản của kiến trúc ResNet cực kì phổ biến trong lĩnh vực máy học và thị giác máy tính.

Nếu sử dụng một mạng học sâu với càng nhiều layer thì model sẽ học được càng nhiều đặc trưng của đối tượng, tuy nhiên đồng thời cũng sẽ gây ra vấn đề mất mát thông tin (vanishing gradient). ResNet (Residual Network) là một kiến trúc được phát triển để giải quyết vấn đề này trong quá trình huấn luyện mạng học sâu có rất nhiều layer.

Giải pháp mà ResNet đưa ra là sử dụng kết nối "tắt" để xuyên qua một hay nhiều lớp. Một khối như vậy được gọi là một Residual Block(khối phần dư), như trong hình sau :

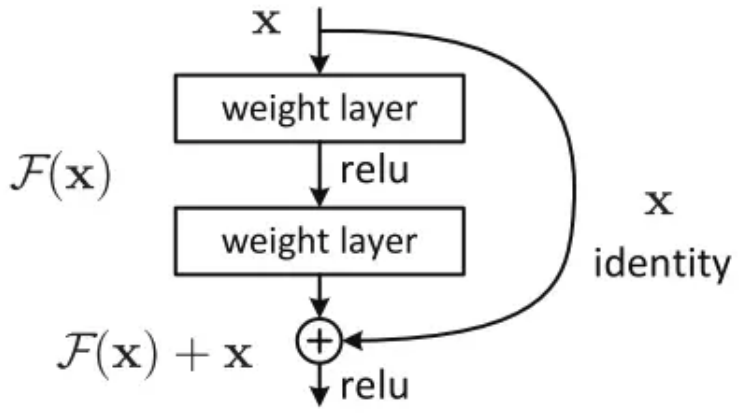


Figure 3: Minh hoạ kiến trúc ResNet50

ResNet gần như tương tự với các mạng gồm có convolution, pooling, activation và fully-connected layer. Ảnh bên trên hiển thị khối phần dư được sử dụng trong mạng. Xuất hiện một mũi tên cong xuất phát từ đầu và kết thúc tại cuối khối dư. Hay nói cách khác là sẽ bổ sung Input X vào đầu ra của layer, hay chính là phép cộng mà ta thấy trong hình minh họa, việc này sẽ chống lại việc đạo hàm bằng 0, do vẫn còn cộng thêm X.

F(x) có được chính là kết quả sau khi dữ liệu đầu vào đi qua khối Residual, còn lối tắt sẽ truyền dữ liệu x từ đầu vào của khối residual đến đầu ra của khối đó. Khi dữ liệu ở 2 đường này gặp nhau thì sẽ được cộng vào để tạo ra đầu ra cuối cùng của khối residual.

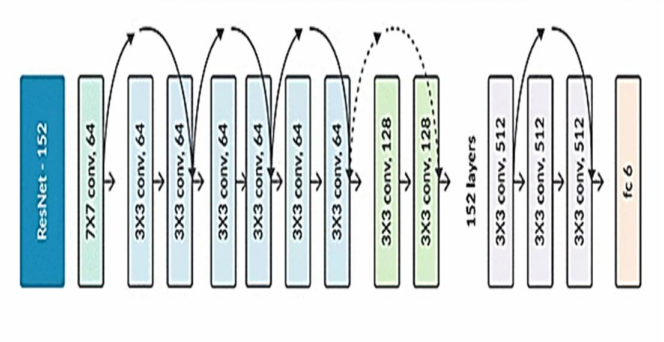


Figure 4: Minh hoạ kiến trúc ResNet50

Mô hình ResNet152V2 bao gồm 152 lớp, được chia thành 5 tầng (stage) chính, với mỗi tầng chứa nhiều khối residual block. Mỗi khối residual block bao gồm các thành phần như convolutional layer, batch normalization, hàm kích hoạt, shortcut connection và hàm kích hoạt cuối cùng. Mô hình kết thúc với một lớp Global Average Pooling và lớp fully-connected để phân loại đầu vào.

4 Thực nghiệm và độ đo

4.1 Lần huấn luyện đầu tiên

1. Sử dụng 2 kiến trúc mạng đã được đề cập ở phần 3 là VGG16 và ResNet152V2 đã được huấn luyện sẵn trên tập dữ liệu imagnet.

   - input_shape: 224x224 là kích thước của ảnh đầu vào.
   - weights: Tùy chọn trọng số của mô hình, ở đây là 'imagenet' (đã được huấn luyện trên imagenet).
   - include_top: Có bao gồm một số lớp fully connected ở đầu và cuối mạng hay không, ở đây là False (loại bỏ các lớp này để tùy chỉnh cho bài toán riêng).

   Những lớp của mạng đã được đóng băng (không được tinh chỉnh) bằng cách đặt thuộc tính 'trainable' của chúng thành False. Phương thức này giúp giữ lại các trọng số đã học được trên imagenet và tránh bị đánh mất thông tin quan trọng giữa các lớp đã huấn luyện.

   Một lớp Flatten được thêm vào bên dưới mạng VGG16 để chuyển từ tensor 3D thành vector 1D, giúp dữ liệu có thể được đưa vào lớp fully connected. Cuối cùng, một lớp fully connected (Dense) được thêm vào với 10 nơi đầu ra và hàm kích hoạt softmax được sử dụng để tính toán xác suất của từng lớp, điều này phù hợp với bài toán phân loại 10 lớp.

   Cuối cùng, một đối tượng mô hình được tạo ra từ input và output đã được xác định để có thể sử dụng cho quá trình huấn luyện.

2. Cấu hình các siêu tham số cho quá trình huấn luyện của model với compile() method:

$layer.trainable = False$: Những lớp của mạng đã được đóng băng (không được tinh chỉnh) bằng cách đặt thuộc tính 'trainable' của chúng thành False. Phương thức này giúp giữ lại các trọng số đã học được trên imagenet và tránh bị đánh mất thông tin quan trọng giữa các lớp đã huấn luyện.

$model.compile(optimizer = opt, loss = keras.losses.categorical\_crossentropy, metrics = ['accuracy'])$

- optimizer: thuật toán tối ưu Graident Descent, ở đây sử dụng Adam optimizer có learning rate (tốc độ học) là $LEARNING\_RATE$.
- loss: Hàm mất mát được sử dụng để đánh giá mức độ sai khác giữa output dự đoán và kết quả thực tế, ở đây sử dụng $categorical\_crossentropy$ vì bài toán là phân loại có nhiều hơn 2 lớp đối tượng.
- metrics: Các chỉ số để đánh giá mô hình, ở đây sử dụng độ chính xác (accuracy) để xác định tỉ lệ phân loại đúng trên tổng số lượng ảnh được phân loại.

Cuối cùng, model hoàn chỉnh với các siêu tham số đã cấu hình sẵn được tạo ra để sử dụng cho quá trình huấn luyện.

3. Huấn luyện mô hình trên dữ liệu đã được chuẩn bị. Các thông số:
   - $Epochs = 10$.
   - $train\_batch = 128$.
   - $valid\_batch = 32$

4.2 Lần huấn luyện thứ hai

Do mô hình vẫn chưa đạt được mức độ chính xác yêu cầu sau khi huấn luyện trên tập train và đánh giá trên tập valid. Trong trường hợp đó, ta có thể thực hiện huấn luyện lại trên mô hình để cải thiện độ chính xác dự đoán của mô hình. Phương pháp này được gọi là fine-tuning, tức là điều chỉnh lại các trọng số của mô hình để hoạt động tốt hơn trên tập dữ liệu của chúng ta, đồng thời giữ nguyên những đặc trưng mà mô hình đã học được ở lần huấn luyện trước. Điều này giúp tiết kiệm thời gian và cho phép chúng ta sử dụng lại kiến thức đã học được từ lần huấn luyện trước.

$layer.trainable = True$: ta phải đặt giá trị các trainable attributes của các lớp của mô hình thành True bằng vòng lặp for.

Ta tiến hành compile lại mô hình với các tham số tương ứng như:

- $loss =' categorical\_crossentropy'$.
- optimizer mới với learning rate là 1e-4 và momentum là 0.9.
- $metrics = ['accuracy']$.

4.3 Độ đo

Các trường hợp dự đoán có thể xảy ra:

- True Positive (TP): Đối tượng ở lớp Positive, mô hình phân đối tượng vào lớp Positive (dự đoán đúng).
- True Negative (TN): Đối tượng ở lớp Negative, mô hình phân đối tượng vào lớp Negative (dự đoán đúng).
- False Positive (FP): Đối tượng ở lớp Negative, mô hình phân đối tượng vào lớp Positive (dự đoán sai).
- False Negative (FN): Đối tượng ở lớp Positive, mô hình phân đối tượng vào lớp Negative (dự đoán sai).

Các độ đo được sử dụng trong bài.

1. Độ chính xác (Accuracy)

   Cách tính: Accuracy là tỉ lệ giữa số điểm được phân loại đúng(TP + TN) và tổng số điểm (TP + TN + FP + FN).

   Công thức: $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$

   Ý nghĩa: Accuracy càng cao, tức là mô hình của chúng ta có khả năng dự đoán đúng một số lượng lớn các điểm dữ liệu được cung cấp. Tuy nhiên, accuracy không phải là độ đo tốt nhất nếu phân loại các lớp trong tập dữ liệu của chúng ta có sự chênh lệch về số lượng điểm dữ liệu giữa các lớp.

2. Precision

   Cách tính: Precision được tính toán bằng số lượng dự đoán positive chính xác (true positive) chia cho tổng số lượng dự đoán là póitive(true positive + false positive).

   Công thức: $Precision = \frac{TP}{TP+FP}$

   Ý nghĩa: Precision càng cao thì những dự đoán positive mà mô hình của chúng ta đưa ra có khả năng thực sự là positive càng cao.

3. Recall

   Cách tính: Recall được tính toán bằng số lượng dự đoán positive chính xác(true positive) chia cho những điểm thực sự là positive (true positive + false negative).

   Công thức: $Recall = \frac{TP}{TP+FN}$

   Ý nghĩa: Recall càng cao đồng nghĩa tỉ lệ bỏ sót các điểm thực sự positive càng thấp.

4. F1-Score

   Cách tính: F1-score có giá trị từ 0 đến 1, là một độ đo kết hợp giữa precision và recall, và được sử dụng để đánh giá hiệu suất của mô hình phân loại. F1-score được tính bằng trung bình điều hoà của precision và recall, và có thể được hiểu là một trọng số cho sự cân bằng giữa precision và recall.

   Công thức: $F1 - score = \frac{2*(precision*recall)}{precision+recall}$

   Ý nghĩa: recall cao hơn thường đi kèm với precision thấp, và ngược lại, precision cao thường đi kèm với recall thấp, vì vậy một giá trị F1-score lớn được coi là một hiệu suất tốt của mô hình. Nếu mô hình có precision và recall là cân bằng, thì giá trị của F1-score sẽ lớn.

5. Confusion matrix

   Trong các bài toán phân loại, confusion matrix (ma trận lỗi) cũng được sử dụng để đánh giá mô hình. Confusion matrix giúp có cái nhìn rõ hơn về việc các điểm dữ liệu được phân loại đúng/sai như thế nào. Các lớp nào dễ bị nhầm lẫn nhất, ở những class nào mô hình phân loại tốt nhất và không tốt nhất.

5 Kết quả và đánh giá

5.1 Trên tập Train và Validation

- Độ chính xác(Accuracy)
  - ***Lần train thứ nhất***

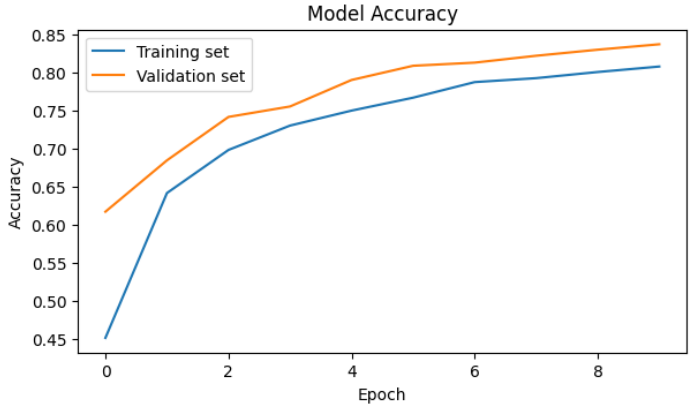


Figure 5: Biểu đồ biểu diễn accuracy của model VGG16 sau lần train thứ nhất.

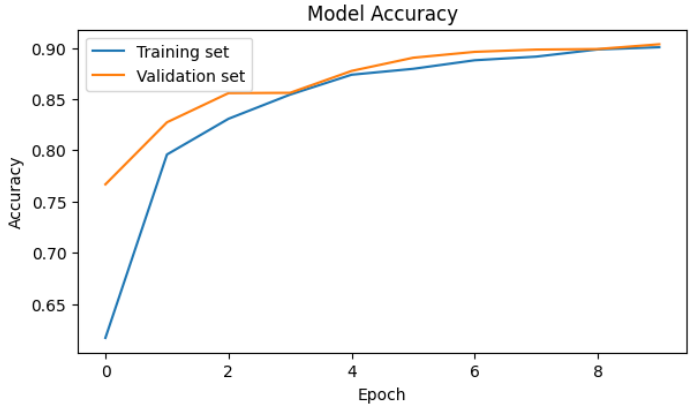


Figure 6: Biểu đồ biểu diễn accuracy của model ResNet152 sau lần train thứ nhất.

Sau 10 lần lặp ta có thể dễ dàng nhận thấy model ResNet152 cho kết quả tốt hơn nhiều so với model VGG16 cả trên tập train và tập validation.

– ***Lần train thứ hai***

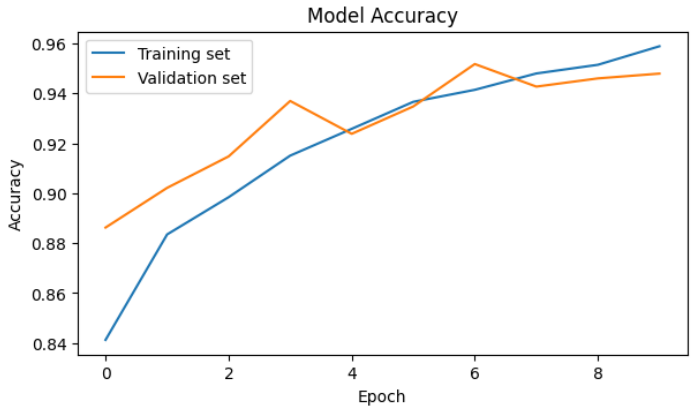


Figure 7: Biểu đồ biểu diễn accuracy của model VGG16 sau lần train thứ nhất.

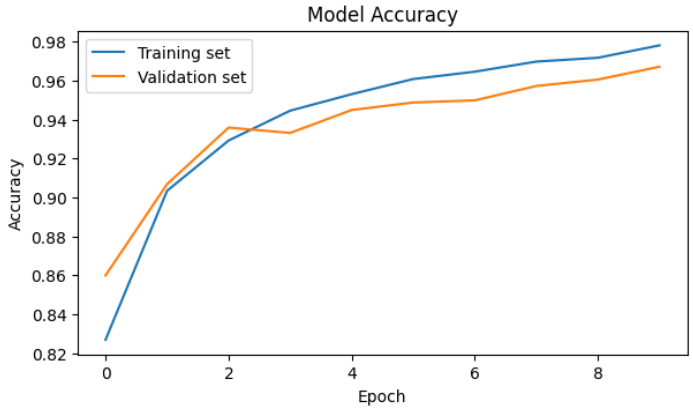


Figure 8: Biểu đồ biểu diễn accuracy của model ResNet152 sau lần train thứ nhất.

Ở lần train thứ hai độ chính xác của cả hai mô hình đã được nâng lên đáng kể đặc biệt là ở model VGG16. Tuy nhiên vẫn là model ResNet152 có độ chính xác nhỉnh hơn vài phần trăm.

- Độ lỗi(Loss)
  - ***Lần train thứ nhất***

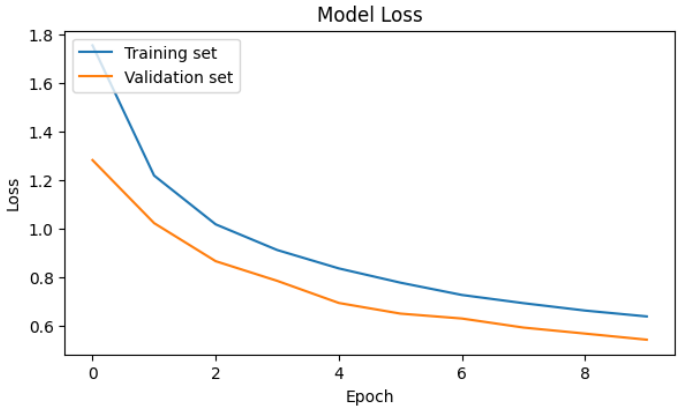


Figure 9: Biểu đồ biểu diễn độ lỗi của model VGG16 sau lần train thứ nhất.

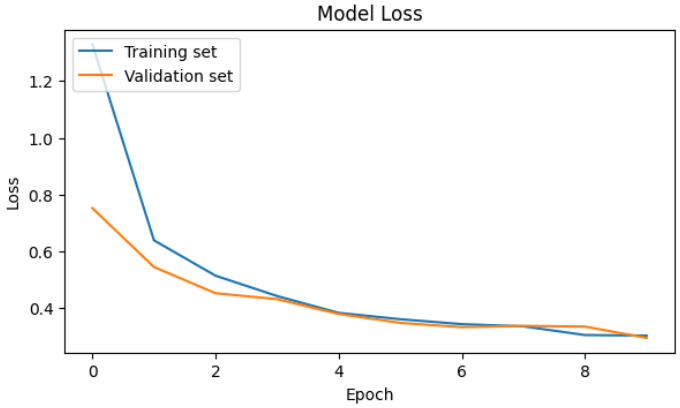


Figure 10: Biểu đồ biểu diễn độ lỗi của model ResNet152 sau lần train thứ nhất.

Model ResNet152 chứng tỏ sự vượt trội của mình khi có độ lỗi thấp hơn gần một nửa so với model VGG16.

– ***Lần train thứ hai***

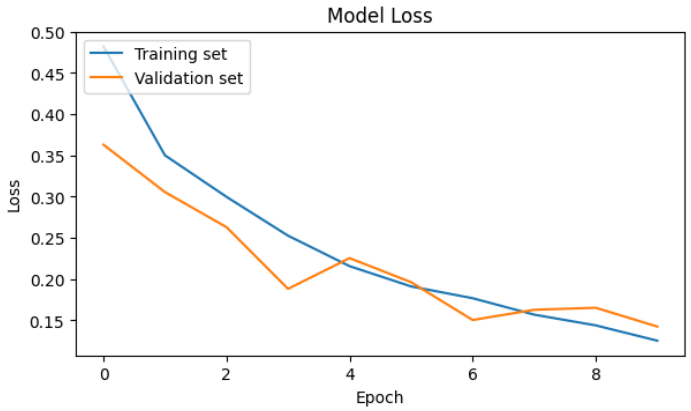


Figure 11: Biểu đồ biểu diễn độ lỗi của model VGG16 sau lần train thứ hai.

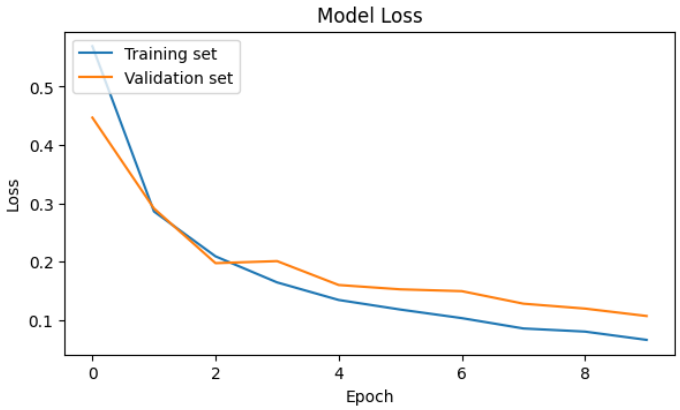


Figure 12: Biểu đồ biểu diễn độ lỗi của model ResNet152 sau lần train thứ hai.

Ở lần train thứ hai độ lỗi của cả hai mô hình đã giảm mạnh so với lần đầu, model VGG16 giảm gấp 3,6 lần và model ResNet152 mặc dù đã có độ lỗi rất thấp ngay từ lần train đầu tiên nhưng vẫn giảm được thêm gấp 2,7 lần.

5.2 Trên tập test

```
1841/1841 [==============================]
Loss on test set:  0.1348511278629303
Accuracy on test set:  0.951656699180603
```

Figure 13: Loss và Accuracy của model VGG16

```
1841/1841 [==============================]
Loss on test set:  0.0791219025850296
Accuracy on test set:  0.9690385460853577
```

Figure 14: Loss và Accuracy của model ResNet152

Sau 2 lần huấn luyện hai mô hình bằng cách thay đổi learning rate cho phù hợp thì cả hai mô hình đều đã cho ra những kết quả thực nghiệm rất cao trên tập test, và ResNet152 là mô hình cho thấy sự vượt trội hoàn toàn của mình.

5.3 Các số liệu kết quả cụ thể

1. Số liệu Accuracy và Loss trong quá trình huấn luyện

		VGG16		ResNet152	
		Acc(%)	Loss	Acc(%)	Loss
Lần 1	Train	80,8	0,6378	90,05	0,3022
	Valid	83,72	0,5419	90,35	0,2943
Lần 2	Train	94,14	0,1768	97,81	0,0662
	Valid	95,18	0,1504	96,71	0,1071

2. Precision, Recall và F1-score

   Qua 2 bảng số liệu dưới đây ta dễ dàng nhận thấy model ResNet152 tốt hơn so với VGG16 vì độ đo F1-score của ResNet152 luôn lớn hơn hoặc bằng VGG16.

Classification Report

	precision	recall	f1-score	support
Tomato___Bacterial_spot	0.98	0.96	0.97	171
Tomato___Early_blight	0.96	0.94	0.95	192
Tomato___Late_blight	0.97	0.94	0.96	186
Tomato___Leaf_Mold	0.99	0.91	0.95	189
Tomato___Septoria_leaf_spot	0.94	0.99	0.96	175
Tomato___Spider_mites Two-spotted_spider_mite	0.93	0.91	0.92	175
Tomato___Target_Spot	0.83	0.91	0.87	184
Tomato___Tomato_Yellow_Leaf_Curl_Virus	1.00	0.96	0.98	197
Tomato___Tomato_mosaic_virus	0.97	0.99	0.98	179
Tomato___healthy	0.95	1.00	0.97	193
accuracy			0.95	1841
macro avg	0.95	0.95	0.95	1841
weighted avg	0.95	0.95	0.95	1841

Figure 15: VGG16

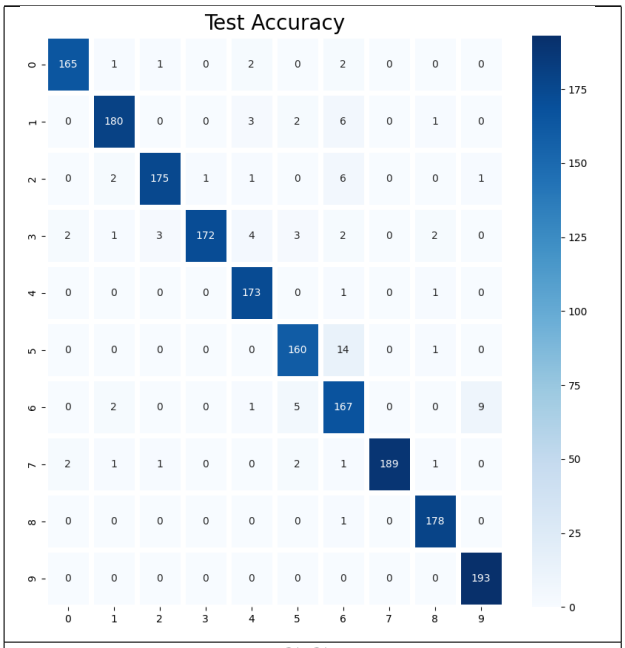
Classification Report

	precision	recall	f1-score	support
Tomato___Bacterial_spot	0.98	0.98	0.98	171
Tomato___Early_blight	0.97	0.93	0.95	192
Tomato___Late_blight	0.96	0.98	0.97	186
Tomato___Leaf_Mold	0.99	0.96	0.98	189
Tomato___Septoria_leaf_spot	0.97	0.99	0.98	175
Tomato___Spider_mites Two-spotted_spider_mite	0.98	0.93	0.95	175
Tomato___Target_Spot	0.89	0.95	0.92	184
Tomato___Tomato_Yellow_Leaf_Curl_Virus	0.99	0.98	0.99	197
Tomato___Tomato_mosaic_virus	0.99	1.00	1.00	179
Tomato___healthy	0.96	0.99	0.97	193
accuracy			0.97	1841
macro avg	0.97	0.97	0.97	1841
weighted avg	0.97	0.97	0.97	1841

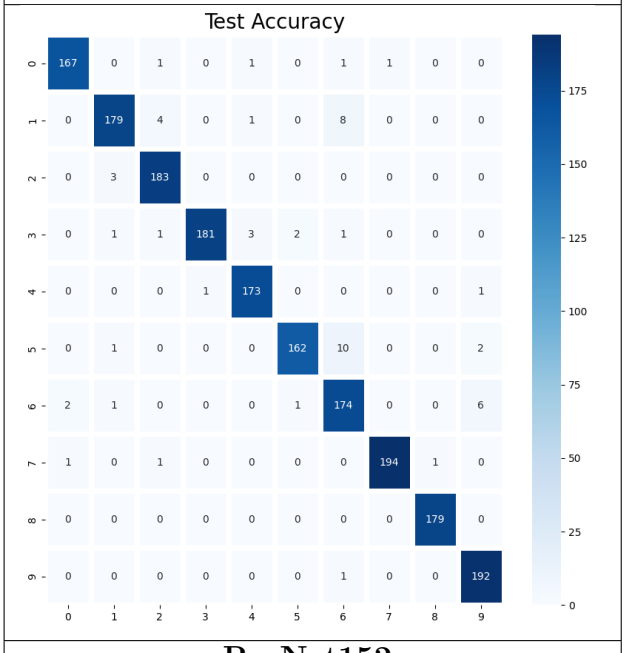
Figure 16: ResNet152

3. Confusion matrix

Table 3: Confusion matrix của VGG16 và ResNet152



VGG16



ResNet152

6 Demo thử nghiệm và kết luận

1. Demo trên tập test

```
1/1 [==============================] - 0s 161ms/step
Class 0: Tomato___Bacterial_spot - 0.00%
Class 1: Tomato___Early_blight - 0.00%
Class 2: Tomato___Late_blight - 0.00%
Class 3: Tomato___Leaf_Mold - 0.00%
Class 4: Tomato___Septoria_leaf_spot - 0.00%
Class 5: Tomato___Spider_mites Two-spotted_spider_mite - 0.00%
Class 6: Tomato___Target_Spot - 0.00%
Class 7: Tomato___Tomato_Yellow_Leaf_Curl_Virus - 0.00%
Class 8: Tomato___Tomato_mosaic_virus - 100.00%
Class 9: Tomato___healthy - 0.00%
Predicted class: Tomato___Tomato_mosaic_virus
```

```
1/1 [==============================] - 0s 143ms/step
Class 0: Tomato___Bacterial_spot - 0.99%
Class 1: Tomato___Early_blight - 0.00%
Class 2: Tomato___Late_blight - 0.00%
Class 3: Tomato___Leaf_Mold - 0.00%
Class 4: Tomato___Septoria_leaf_spot - 0.00%
Class 5: Tomato___Spider_mites Two-spotted_spider_mite - 0.00%
Class 6: Tomato___Target_Spot - 0.00%
Class 7: Tomato___Tomato_Yellow_Leaf_Curl_Virus - 0.00%
Class 8: Tomato___Tomato_mosaic_virus - 0.00%
Class 9: Tomato___healthy - 0.00%
Predicted class: Tomato___Bacterial_spot
```

Figure 17: Demo trên model VGG16

```
1/1 [==============================] - 2s 2s/step
Class 0: Tomato___Bacterial_spot - 0.00%
Class 1: Tomato___Early_blight - 99.80%
Class 2: Tomato___Late_blight - 0.20%
Class 3: Tomato___Leaf_Mold - 0.00%
Class 4: Tomato___Septoria_leaf_spot - 0.00%
Class 5: Tomato___Spider_mites Two-spotted_spider_mite - 0.00%
Class 6: Tomato___Target_Spot - 0.00%
Class 7: Tomato___Tomato_Yellow_Leaf_Curl_Virus - 0.00%
Class 8: Tomato___Tomato_mosaic_virus - 0.00%
Class 9: Tomato___healthy - 0.00%
Predicted class: Tomato___Early_blight
```

```
1/1 [==============================] - 0s 33ms/step
Class 0: Tomato___Bacterial_spot - 0.00%
Class 1: Tomato___Early_blight - 0.00%
Class 2: Tomato___Late_blight - 0.00%
Class 3: Tomato___Leaf_Mold - 0.00%
Class 4: Tomato___Septoria_leaf_spot - 0.00%
Class 5: Tomato___Spider_mites Two-spotted_spider_mite - 100.00%
Class 6: Tomato___Target_Spot - 0.00%
Class 7: Tomato___Tomato_Yellow_Leaf_Curl_Virus - 0.00%
Class 8: Tomato___Tomato_mosaic_virus - 0.00%
Class 9: Tomato___healthy - 0.00%
Predicted class: Tomato___Spider_mites Two-spotted_spider_mite
```

Figure 18: Demo trên model ResNet152

2. Kết luận
   Trước khi sử dụng 2 model là VGG16 và ResNet152 như trong bài đã nêu thì đồ án ban đầu đã từng thử qua 2 model là Inception và ResNet50 nhưng cho dù đã huấn luyện qua đến 2 lần nhưng độ chính xác vẫn chỉ xấp xỉ 50%, vì vậy đồ án đã chuyển hướng sang 2 mô hình là VGG16 và ResNet152. Đúng như kì vọng cả hai mô hình đều cho ra kết quả rất tốt và không chênh lệch quá nhiều.

   Trong tương lai có thể đồ án sẽ sử dụng đến các mô hình có số lớp mạng sâu hơn và dataset lớn hơn để mở rộng hơn nữa phạm vi nhận diện của bài toán.

References

1. Data: New Plant Diseases Dataset(Augmented)
2. Very Deep Convolutional Networks for Large-Scale Image Recognition
3. Deep Residual Learning for Image Recognition
4. Tiến trình phát triển của CNN
5. Mạng sử dụng Khối (VGG) - Đắm mình vào học sâu
6. Giới thiệu mạng ResNet - Viblo