Dự đoán bệnh của cây táo dựa trên hình ảnh lá cây

Thông tin

LECTURE

TS. Mai Tiến Dũng

MEMBERS

Nguyễn Thanh Điền	18520600
Lê Trung Hiếu	18520738

Nội dung trình bày

01 Giới thiệu bài toán

02 Định nghĩa bài toán

03 Tập dữ liệu

04 Phương pháp giải quyết

05 Cài đặt thử nghiệm

06 Demo

07 Kết luận

1. Giới thiệu bài toán

1.Giới thiệu bài toán

Việc chẩn đoán các loại bệnh của cây táo thường do con người => tỉ lệ **chẩn đoán sai** cao, **tốn kém** chi phí, thời gian.

Làm sao để giải quyết các vấn đề trên?

Hình 1. Bệnh nấm (Apple scab - trái), bệnh gỉ sắt (Apple rush – phải) là những bệnh phổ biến trên vườn táo ảnh hưởng đến vẻ ngoài thẩm mỹ, chất lượng và sản lượng quả

1. Giới thiệu bài toán

Giải pháp

Xây dựng hệ thống thị giác máy tính để chẩn đoán bệnh dựa trên hình ảnh lá cây táo

1.Giới thiệu bài toán

Cuộc thi Plant Pathology 2020 - FGVC7

Cho ảnh lá của một cây táo , chẩn đoán các loại bệnh cho cây táo vào **4 loại**:

1. Healthy (khoẻ mạnh)
2. Rush (bệnh rỉ sắt)
3. Scab (bệnh nấm)
4. Multiple_diseases (nhiều loại bệnh)

2. Định nghĩa bài toán

2. Định nghĩa

Input & Output

Ảnh chụp lá của
một cây táo

VS

Loại bệnh được
chẩn đoán của cây
táo đó

3. Tập dữ liệu

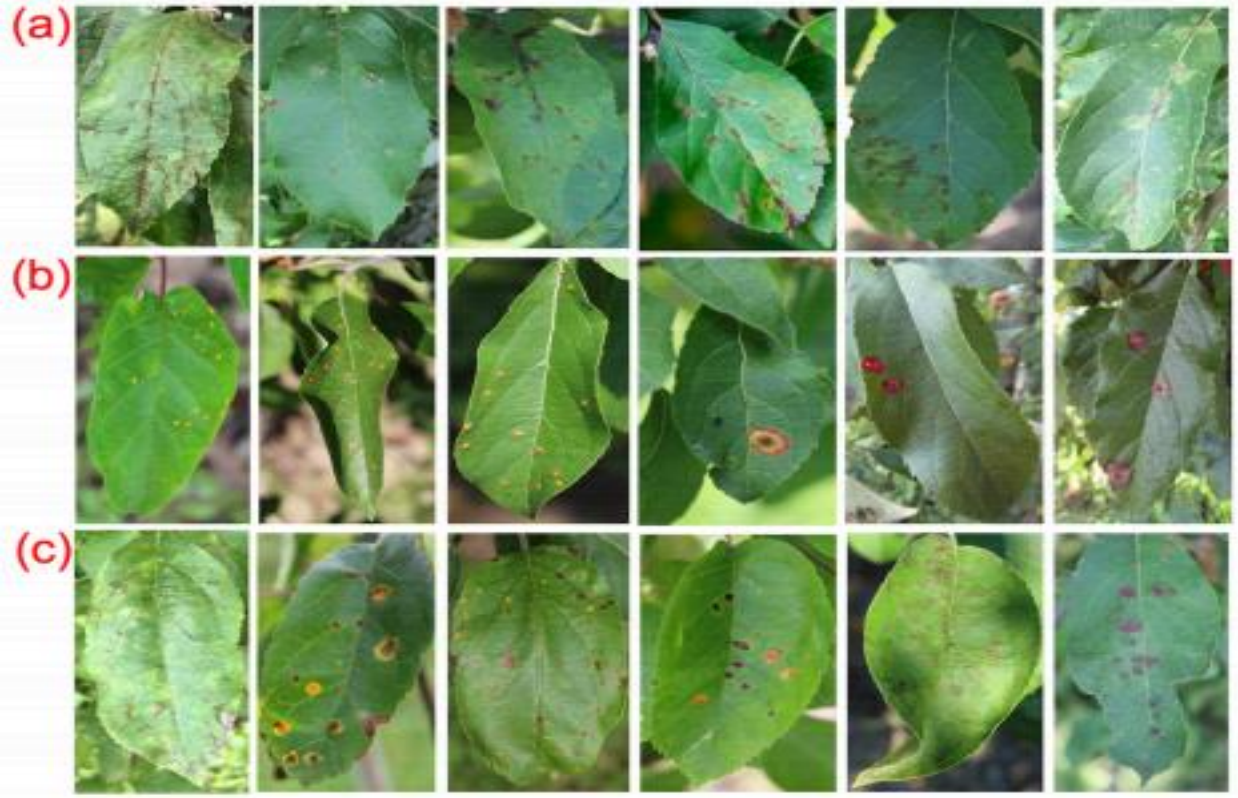
3.Tập dữ liệu

- Lấy từ cuộc thi Cuộc thi **Plant Pathology 2020 - FGVC7**
- Gồm **3642** hình ảnh lá cây táo được chụp với nhiều góc độ, độ sáng khác khác nhau
- Kích thước ảnh phần lớn là **(1365, 2048, 3)**, còn lại là **(2048, 1365, 3)**

Hình 2: Hình ảnh từ tập dữ liệu với nhiều góc độ, độ sáng khác nhau

3.Tập dữ liệu

Tập train: 1821 ảnh và 1 file .csv chứa nhãn của từng ảnh



Hình 3: Hình ảnh mẫu từ tập dữ liệu cho thấy các loại bệnh

a) Bệnh rỉ sắt (rush)
b) Bệnh nấm (scab)
c) Nhiều bệnh trên một lá (multiple_diseases)

	A	B	C	D	E	F
1	image_id	healthy	multiple_	rust	scab	
2	Train_0	0	0	0	1	
3	Train_1	0	1	0	0	
4	Train_2	1	0	0	0	
5	Train_3	0	0	1	0	
6	Train_4	1	0	0	0	
7	Train_5	1	0	0	0	
8	Train_6	0	1	0	0	
9	Train_7	0	0	0	1	
10	Train_8	0	0	0	1	

Hình 4. Một phần từ file .csv

- Healthy: 516 ảnh
- Rust: 622 ảnh
- Scab: 592 ảnh
- **Multiple diseases: 91**

Tập dữ liệu không cân bằng

3.Tập dữ liệu

Tập test: 1821 ảnh không có nhãn

Hình 4: Một số hình ảnh mẫu từ tập test

4. Phương pháp giải quyết

4. Phương pháp giải quyết

4.1 Cơ sở lý thuyết

Transfer learning

Transfer learning là việc sử dụng lại mô hình đã được đào tạo trước để giải quyết một vấn đề mới
VD: Đào tạo trước mô hình trên một tập dữ liệu khổng lồ như ImageNet và sau đó trích xuất các đặc trưng từ nó để sử dụng cho một mục đích khác

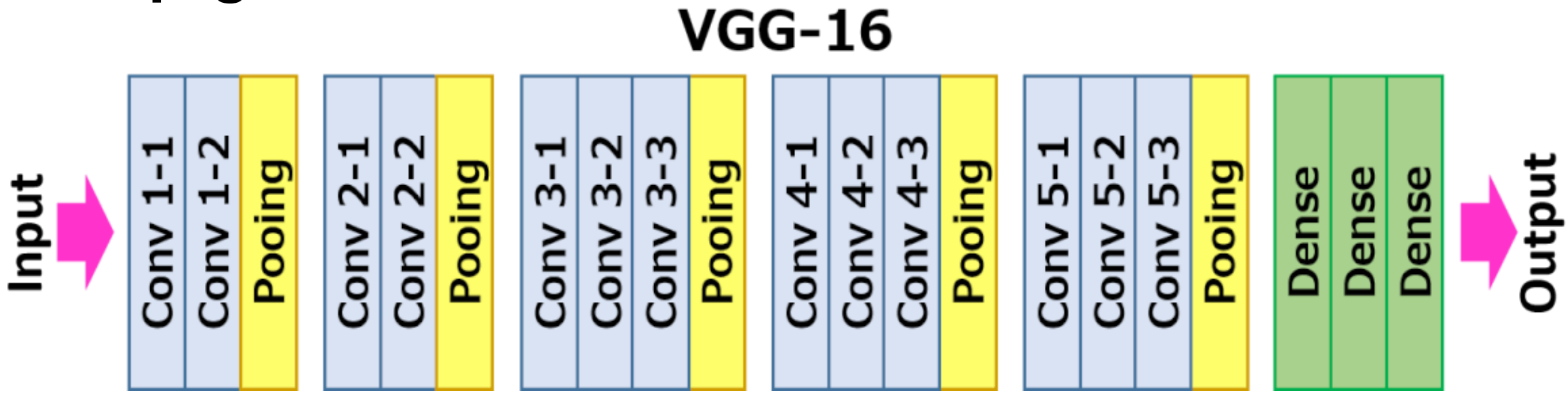
Lợi ích :

- Tiết kiệm thời gian
- Cải thiện hiệu suất học tập và giảm công sức thu thập dữ liệu

=> VGG16, VGG19

4. Phương pháp giải quyết

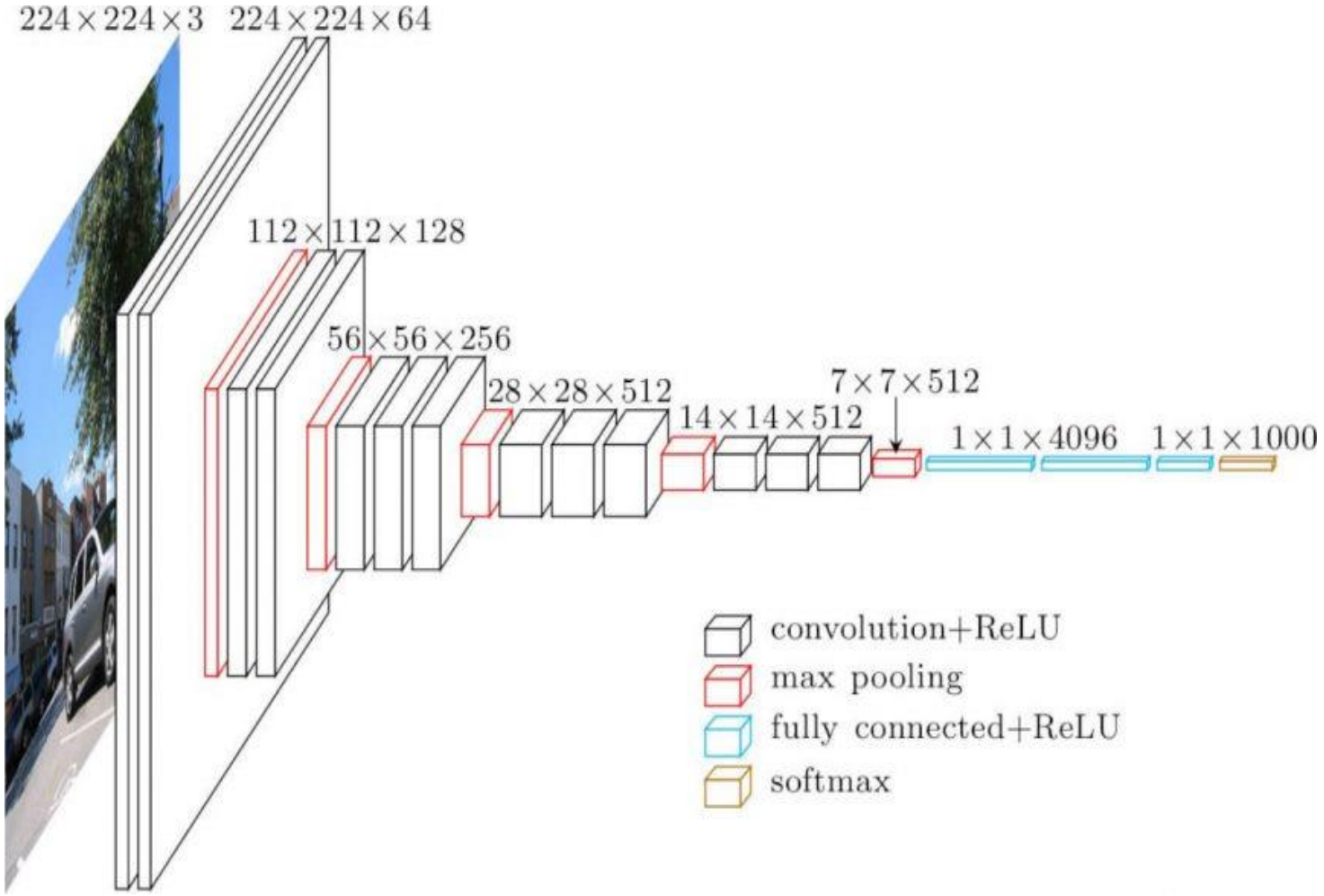
4.2 Mạng VGG16



Hình 5: Kiến trúc mạng VGG16

- VGG16 là một mô hình mạng nơ-ron tích chập được đề xuất bởi Karen **Simonyan** và Andrew **Zisserman** từ Đại học Oxford trong bài báo “Very Deep Convolutional Networks for Large-Scale Image Recognition”, và ILSVRC-2014
- Mô hình này đạt độ chính xác top 5 là 92,7% trong ImageNet (14 triệu hình ảnh, 1000 lớp)
- VGG16 đã được train trong nhiều tuần và đang sử dụng GPU NVIDIA Titan Black
- VGG16 được sử dụng nhiều trong trong cái bài toán phân lớp

4. Phương pháp giải quyết

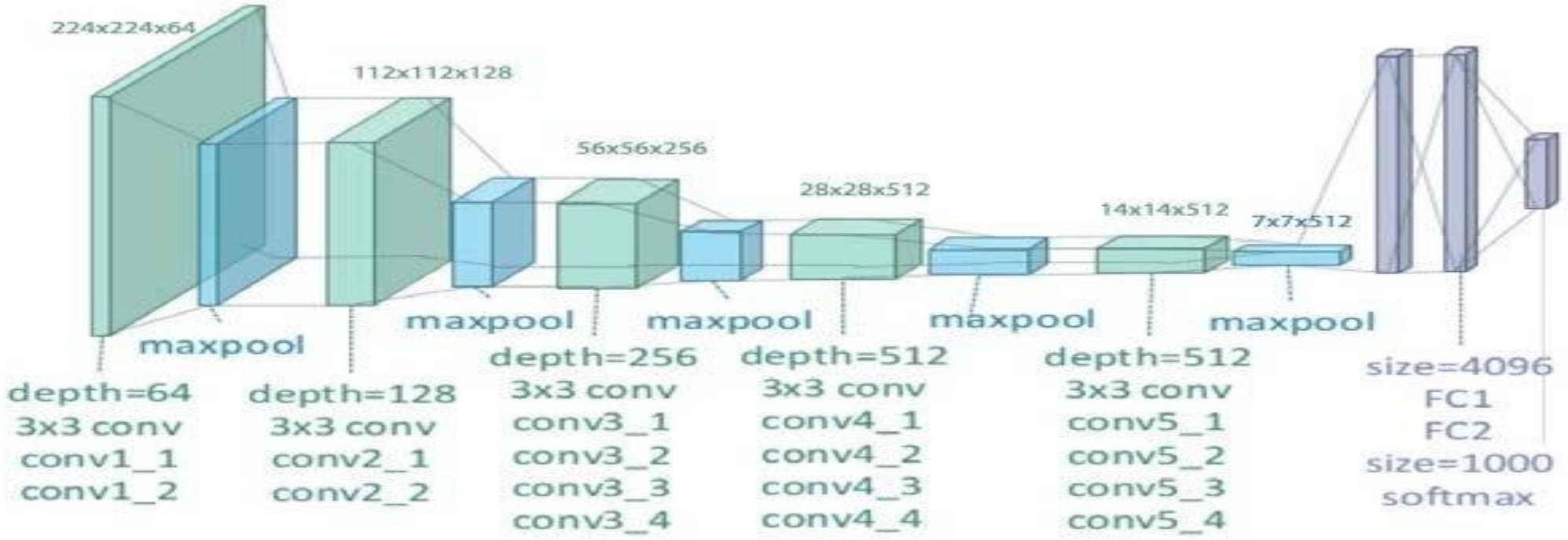


- Input: một bức ảnh RGB kích thước 224x224
- Outptut: một véc tơ 1000 giá trị biểu diễn xác suất phân lớp

Hình 6: Kiến trúc của mạng VGG16

4. Phương pháp giải quyết

4.3 Mạng VGG19



Hình 7: Kiến trúc của mạng VGG19

- VGG19 là một biến thể của mô hình VGG, ngắn gọn là bao gồm 19 lớp cũng được train trên tập dữ liệu ImageNet cùng năm với VGG16
- VGG19 có cấu trúc tương tự như VGG16

4. Phương pháp giải quyết

4.4 Xây dựng mô hình mạng dựa trên VGG16, VGG19

```
Model type: VGG16
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 7, 7, 512)         14714688
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
dense (Dense)                (None, 4)                 100356
=================================================================
Total params: 14,815,044
Trainable params: 100,356
Non-trainable params: 14,714,688
```

Hình 8: Chi tiết mạng VGG16

```
Model type: VGG19
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg19 (Functional)           (None, 7, 7, 512)         20024384
_________________________________________________________________
flatten_1 (Flatten)          (None, 25088)             0
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 100356
=================================================================
Total params: 20,124,740
Trainable params: 100,356
Non-trainable params: 20,024,384
```

Hình 9: Chi tiết mạng VGG19

4. Phương pháp giải quyết

4.5 Mô hình mạng CNN tham khảo

Chương 5 - "Deep learning for computer vision", trong sách "Deep Learning with Python" phiên bản đầu tiên, của tác giả François Chollet - tác giả của thư viện Keras

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(len(CLASS_NAMES), activation='softmax'))

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 222, 222, 32)      896
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 111, 111, 32)      0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 109, 109, 64)      18496
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 54, 54, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 52, 52, 64)        36928
_________________________________________________________________
flatten_1 (Flatten)          (None, 173056)            0
_________________________________________________________________
dense_2 (Dense)              (None, 64)                11075648
_________________________________________________________________
dense_3 (Dense)              (None, 64)                4160
_________________________________________________________________
dense_4 (Dense)              (None, 4)                 260
=================================================================
Total params: 11,136,388
Trainable params: 11,136,388
Non-trainable params: 0
_________________________________________________________________
```

Hình 10: Chi tiết mạng CNN tham khảo

4. Phương pháp giải quyết

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(len(CLASS_NAMES), activation='softmax'))

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 222, 222, 32)      896
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 111, 111, 32)      0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 109, 109, 64)      18496
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 54, 54, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 52, 52, 64)        36928
_________________________________________________________________
flatten_1 (Flatten)          (None, 173056)            0
_________________________________________________________________
dense_2 (Dense)              (None, 64)                11075648
_________________________________________________________________
dense_3 (Dense)              (None, 64)                4160
_________________________________________________________________
dense_4 (Dense)              (None, 4)                 260
=================================================================
Total params: 11,136,388
Trainable params: 11,136,388
Non-trainable params: 0
_________________________________________________________________
```

Hình 11: Mô hình mạng tham khảo ban đầu

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(len(CLASS_NAMES), activation='softmax'))

model.summary()
```

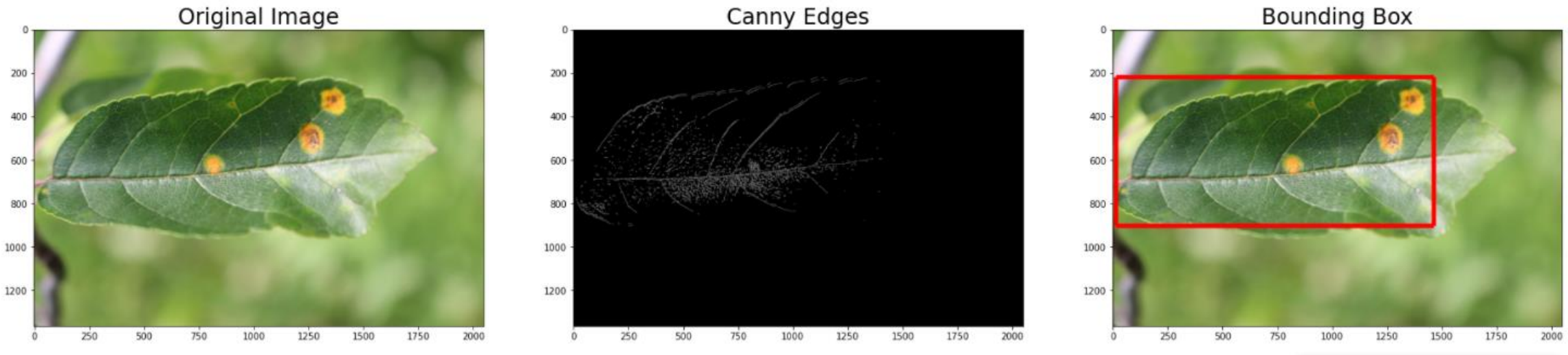
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 222, 222, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 111, 111, 32)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 109, 109, 64)      18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 54, 54, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 186624)            0
_________________________________________________________________
dense (Dense)                (None, 64)                11944000
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 260
=================================================================
Total params: 11,963,652
Trainable params: 11,963,652
Non-trainable params: 0
_________________________________________________________________
```

Hình 12: Mô hình mạng tham khảo đã thay đổi

4. Phương pháp giải quyết

4.6 Tiền xử lý

a) Edge dectection: Canny



Hình 13: Cắt hình ảnh chiếc lá bằng cách áp dụng Canny

4. Phương pháp giải quyết

4.6 Tiền xử lý

b) Resize + Padding

Hình 14: Ảnh gốc 1365x2048

Hình 15: Ảnh 224x224 (Resize → Pading)

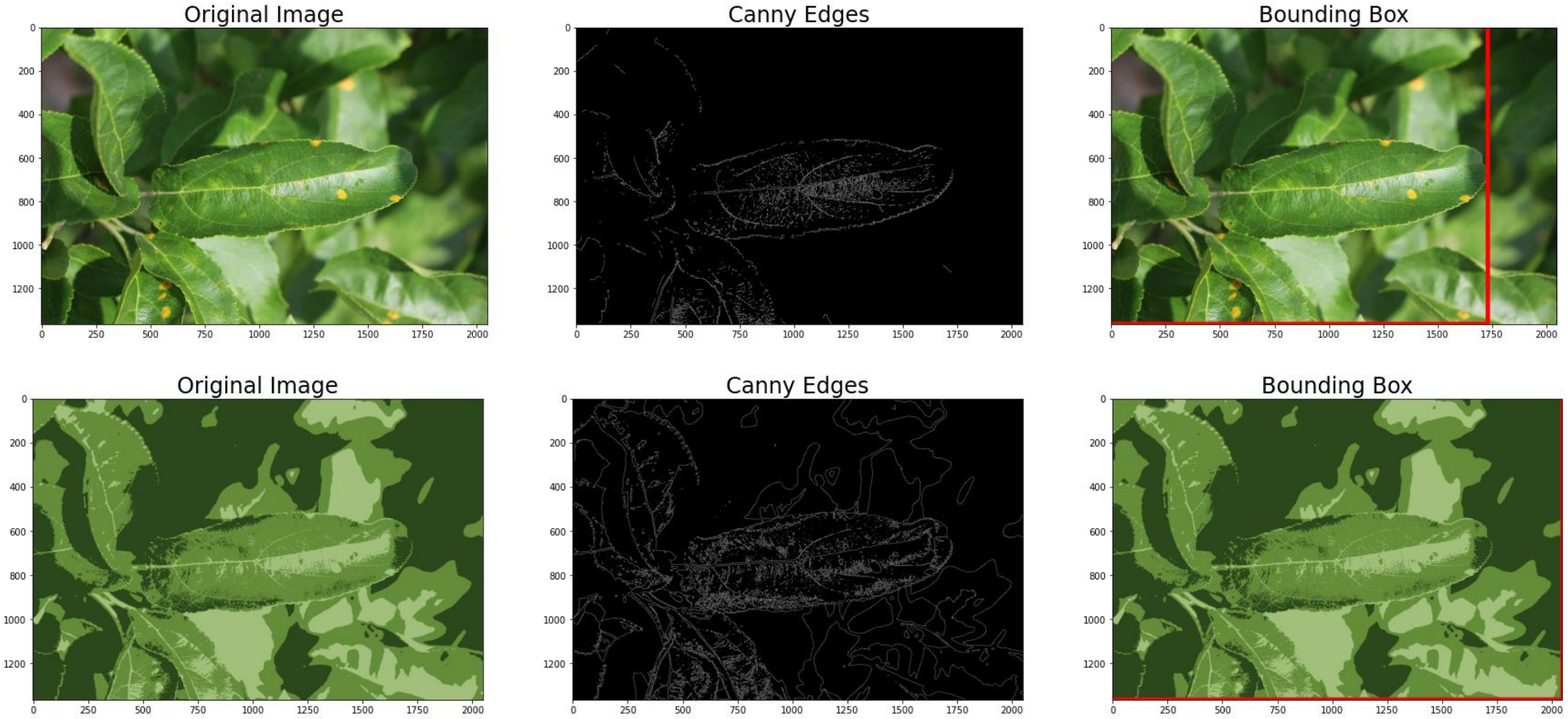
Hình 16: Ảnh 224x224 (Canny → Resize → Pading)

4. Phương pháp giải quyết

4.6 Tiền xử lý

c) Áp dụng Segmentation: K-Means

Không áp dụng

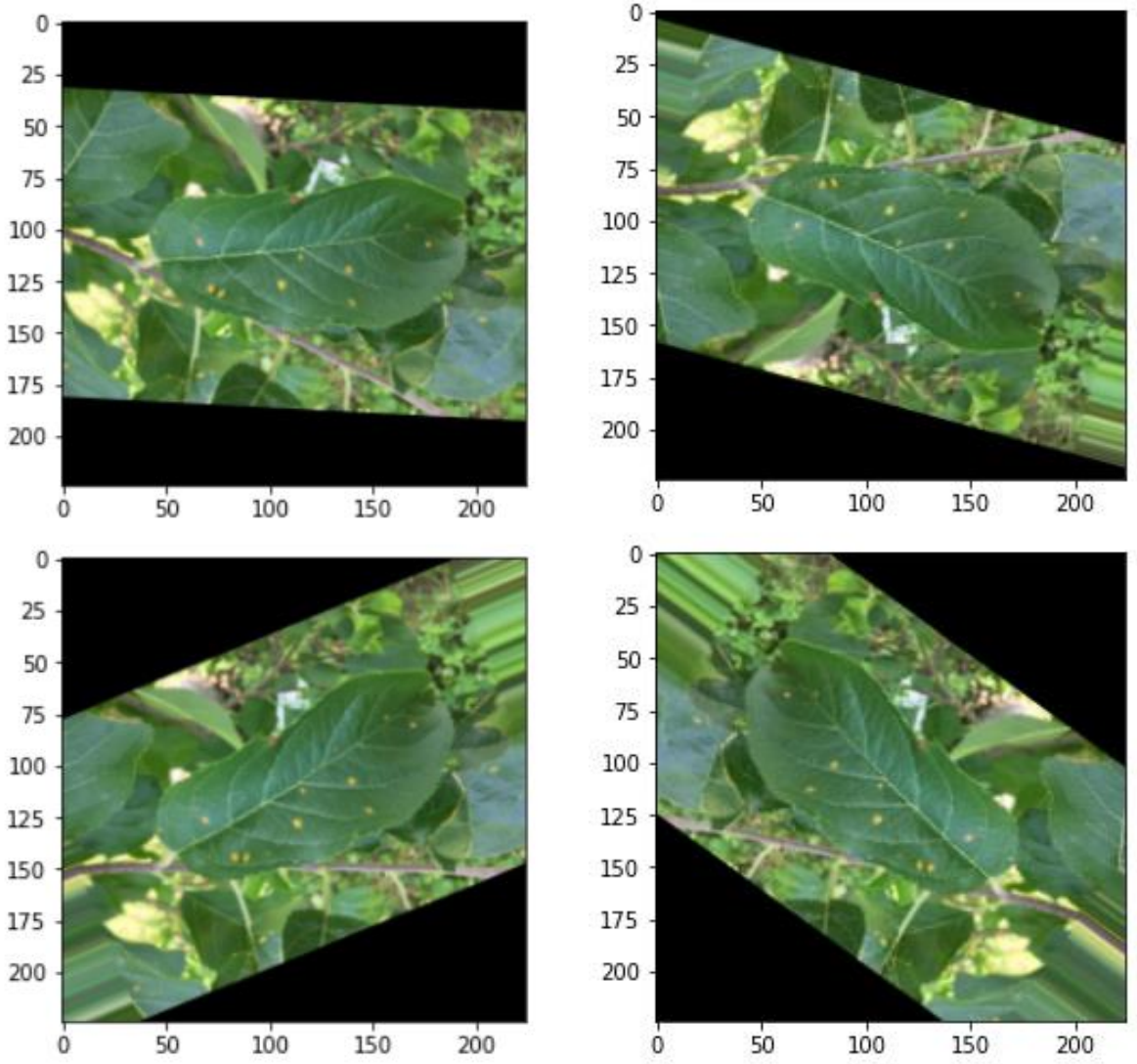


Hình 17: Áp K-means để segment ảnh, k = 3, sau đó áp dụng Canny để cắt ảnh

4. Phương pháp giải quyết

4.7 Data augmented

Hình 18: Ảnh gốc 224x224



Hình 19: Ảnh đã được áp dụng các phép biến đổi

5.Cài đặt thử nghiệm

5. Cài đặt thử nghiệm

5.1 Cài đặt chương trình

Ngôn ngữ: Python 3
Môi trường cài đặt: Google Colab
Phần cứng: GPU miễn phí của Google Colab

5. Cài đặt thử nghiệm

5.2 Phương thức đánh giá

ROC (receiver operating characteristic curve): là một đồ thị biểu diễn sự hiệu quả (performance) của mô hình phân lớp theo giá trị ngưỡng phân lớp

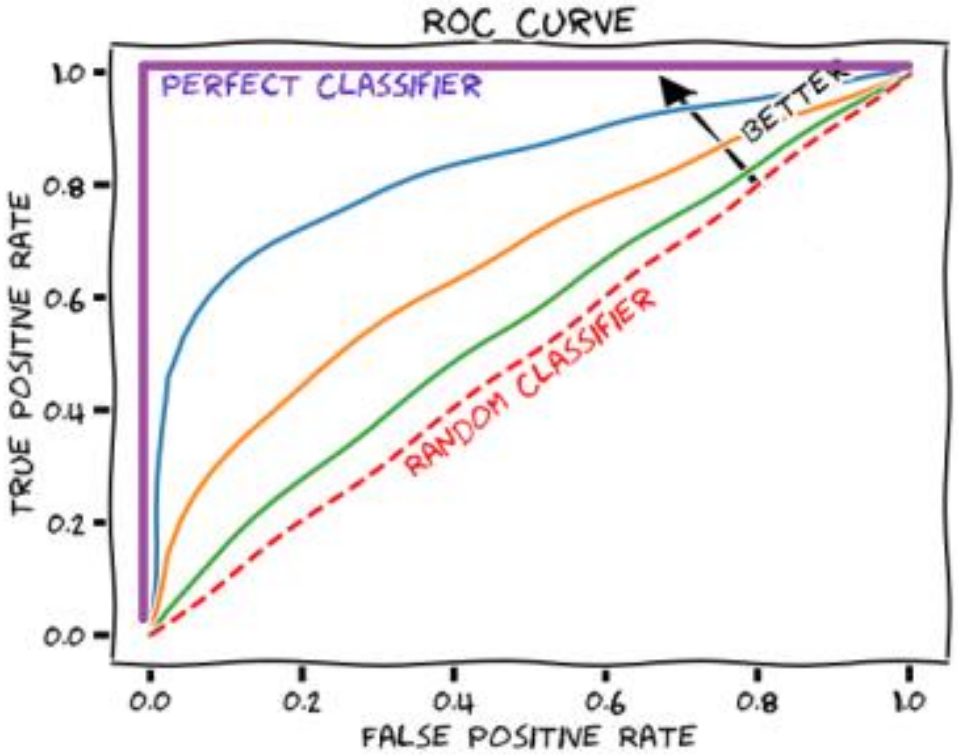
AUC (Area under the ROC Curve): đo diện tích của phần bên dưới đường cong ROC

True Positive Rate (TPR)

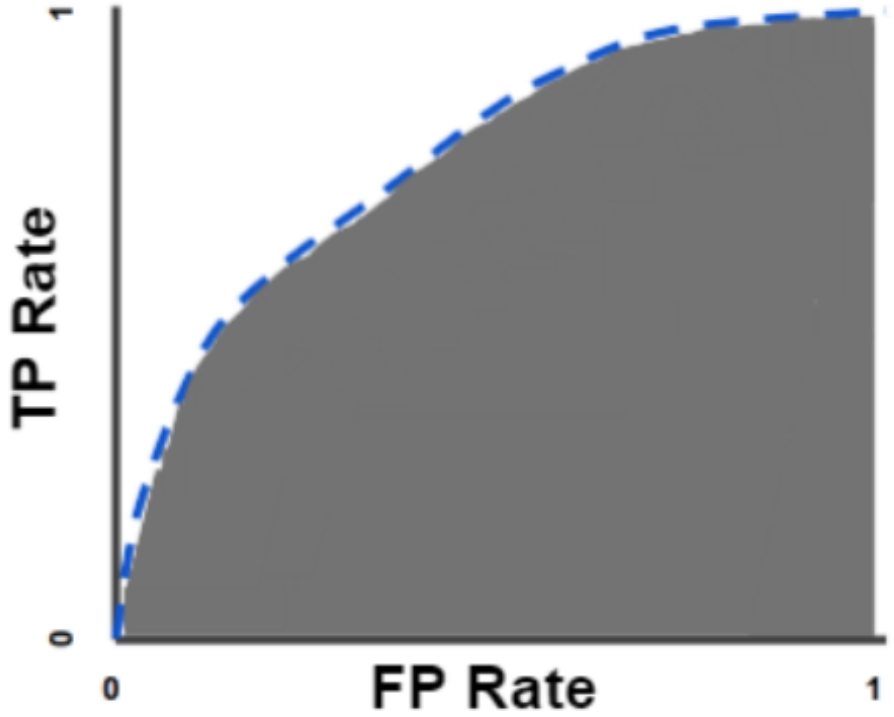
$$TPR = \frac{TP}{TP + FN}$$

False Positive Rate (FPR)

$$FPR = \frac{FP}{FP + TN}$$



Hình 20: Biểu đồ ROC



Hình 21: Phần diện tích AUC

5. Cài đặt thử nghiệm

5.3 Kết quả thực nghiệm

Conv base VGG16 + 1 FC:

- Train: 0.7394
- Test:
  - Private: 0.48835
  - Public: 0.51689

Mô hình mạng tham khảo ban đầu:

- Train: 0.6968
- Test:
  - Private: 0.49568
  - Public: 0.53622

Conv base VGG19 + 1 FC:

- Train: 0.7397
- Test:
  - Private: 0.46924
  - Public: 0.5015

Mô hình mạng tham khảo đã thay đổi:

- Train: 0.6892
- Test:
  - Private: 0.48455
  - Public: 0.53161

6. Demo

7. Kết luận

7. Kết luận

- Cả 3 model đã thử nghiệm chưa thể sử dụng trong thực tế
- Cần sử dụng thêm nhiều phương pháp xử lý ảnh và tinh chỉnh các hyper parameter của model
- Dataset khá ít và không cân bằng giữa các lớp khiến việc huấn luyện model gặp nhiều khó khăn

8. Tài liệu tham khảo

- https://www.geeksforgeeks.org/vgg-16-cnn-model/
- https://neurohive.io/en/popular-networks/vgg16/
- https://arxiv.org/ftp/arxiv/papers/2004/2004.11958.pdf
- https://www.quora.com/What-is-the-difference-between-VGG16-and-VGG19-neural-network
- https://developers.google.com/machine-learning/crash-course/classification/check-your-understanding-roc-and-auc
- https://stackoverflow.com/questions/44720580/resize-image-canvas-to-maintain-square-aspect-ratio-in-python-opencv
- https://docs.opencv.org/3.4/da/d22/tutorial_py_canny.html
- https://gist.github.com/pknowledge/05d68179cdb364e4fa4db608517f8d17
- https://stackoverflow.com/questions/9374747/optimal-approach-for-detecting-leaf-like-shapes-in-opencv
- https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/
- https://www.youtube.com/watch?v=A_ZKMsZ3f3o
- https://keras.io/api/metrics/classification_metrics/#auc-class

Thank you