

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



BÁO CÁO ĐỒ ÁN
FRESH AND ROTTEN FRUIT CLASSIFICATION
CS114 – MÁY HỌC

Giảng viên hướng dẫn: PGS. TS. Lê Đình Duy
ThS. Phạm Nguyễn Trường An
Nhóm sinh viên thực hiện: Lớp CS114.N11.KHCL

STT	MSSV	Họ và tên
1	20520767	Trương Thị Thanh Thanh
2	20520785	Tổng Phúc Thịnh
3	20520795	Nguyễn Minh Thuận

MỤC LỤC

BẢNG ĐÓNG GÓP VÀ ĐÁNH GIÁ	iii
CHƯƠNG I: GIỚI THIỆU	1
1. Đặt vấn đề.....	1
2. Fresh and rotten fruit classification.....	2
2.1. Khái niệm	2
2.2. Input và output.....	3
2.2.1. Input.....	3
2.2.2. Output.....	3
2.2.3. Mục đích	3
2.2.4. Ứng dụng	4
3. Mục tiêu đề tài.....	4
CHƯƠNG II: VGG16.....	5
1. Giới thiệu	5
2. Kiến trúc mạng VGG16:.....	5
3. Mô tả cách hoạt động	9
CHƯƠNG III: DATASET.....	16
1. Bộ dữ liệu trên Mendeley Data.....	16
1.1. Giới thiệu.....	16
1.2. Mô tả chi tiết	16
1.3. Các khó khăn khi sử dụng bộ dataset	19
2. PreProcessing	20
CHƯƠNG IV: HUẤN LUYỆN VÀ KIỂM THỬ MÔ HÌNH.....	22
1. Lý do chọn mô hình	22
2. Preprocess dataset	22

3. Huấn luyện mô hình	24
3.1. Cài đặt mô hình huấn luyện	26
3.2. Kết quả huấn luyện	27
4. Đánh giá mô hình huấn luyện.....	28
4.1. Phương pháp đánh giá.....	28
4.2. Kết quả đánh giá	29
KẾT LUẬN.....	34
TÀI LIỆU THAM KHẢO.....	35

BẢNG ĐÓNG GÓP VÀ ĐÁNH GIÁ

STT	MSSV	Họ và tên	Phân công nhiệm vụ	Tỉ lệ hoàn thành
1	20520767	Trương Thị Thanh Thanh	<ul style="list-style-type: none"> • Viết báo cáo: <ul style="list-style-type: none"> ▪ Chương I, III, IV. ▪ Định dạng bài báo cáo. • Coding: tiền xử lý dữ liệu 	100%
2	20520785	Tổng Phúc Thịnh	<ul style="list-style-type: none"> • Viết báo cáo: chương II. 	70%
3	20520795	Nguyễn Minh Thuận	<ul style="list-style-type: none"> • Viết báo cáo: chương III, IV. • Coding: Huấn luyện model VGG16 và đánh giá mô hình 	100%

CHƯƠNG I: GIỚI THIỆU

1. Đặt vấn đề

- Ngày nay với sự phát triển của khoa học và công nghệ, cuộc sống của con người ngày càng ấm no và tiện nghi hơn. Sự phát triển của khoa học và công nghệ kéo theo sự phát triển của các ngành nghề khác, trong đó có cả nông nghiệp và vận tải. Sự phát triển của nông nghiệp với việc ứng dụng khoa học đã cho ra nhiều giống cây trồng mới và tăng năng suất của nông nghiệp lên. Cùng với sự phát triển của ngành vận tải đã giúp cho việc phân phối các loại trái cây đặc trưng của vùng này có thể đến được các vùng nơi không có hoặc không thể trồng được loại cây đó, qua đó giúp mọi người có thể thưởng thức chúng dù ở đâu trên thế giới.
- Tuy nhiên, một điều dễ thấy đó là các loại trái cây có thể bị hư hỏng, dập trong quá trình vận chuyển, mà mỗi loại trái cây khi bị hư hỏng, không thể sử dụng có đặc điểm riêng của mỗi loại. Đối với người dùng, điều này có thể rất khó khăn, vì không phải ai cũng biết và hiểu chính xác từng loại trái cây mà họ ăn để biết quả nào có thể ăn được hay đã bị hư.
- Chính vì lẽ đó, nhóm chúng em quyết định xây dựng 1 mô hình máy học, sử dụng dữ liệu được thu thập về thông tin trạng thái của các loại quả, và nhận diện chúng thông qua hình ảnh. Để người dùng có thể kiểm tra chúng thông qua việc sử dụng mô hình này, giúp cải thiện chất lượng cuộc sống của mọi người.

2. Fresh and rotten fruit classification

2.1. Khái niệm

Trái cây tươi là những trái cây chưa bị chín hoàn toàn, không bị hư hỏng, không bị thối rữa và được bảo quản đúng cách. Những trái cây tươi thường có mùi thơm, vị ngon và chứa nhiều dưỡng chất hữu ích cho sức khỏe, bao gồm vitamin, khoáng chất và chất xơ.



Hình 1.1. Ảnh trái cây tươi

Trái cây thối rữa là những trái cây bị phân hủy và mục nát do bị nhiễm khuẩn hoặc nấm. Những trái cây thối rữa thường có mùi hôi thối, màu sắc thay đổi, và thường không an toàn để ăn. Khi một trái cây bắt đầu thối rữa, khu vực thối có thể bao phủ bởi nấm mốc hoặc khuẩn, gây ra các vết đen hoặc xám trên bề mặt trái cây.



Hình 1.2: Ảnh trái cây hư

2.2. Input và output

2.2.1. Input

Một bức ảnh màu có định dạng đuôi ảnh là định dạng jpg.

Ảnh được chụp từ camera của điện thoại hoặc máy ảnh

Ảnh chỉ có duy nhất 1 loại quả nhưng không giới hạn số lượng của loại quả đó trong tấm ảnh.

2.2.2. Output

Dự đoán tên của loại quả trong ảnh và trạng thái “Fresh” (tươi) hoặc “Rotten” (hư) của loại quả ấy. Kết quả đầu ra sẽ có dạng như sau “Trạng thái của quả” “Tên quả” trong tiếng anh.

Ví dụ: “FreshApple”, “RottenBanana”

2.2.3. Mục đích

Trong thời đại công nghệ 4.0, tự động hóa sử dụng machine learning là một việc thường thấy trong thực tế và xuất hiện ở trong mọi lĩnh vực như nông nghiệp, công nghiệp,... Đặc biệt là trong nông nghiệp thì machine learning đã xuất hiện rất nhiều bài toán khó cần giải quyết để đáp ứng nhu cầu tự động hóa trong nông nghiệp như phân loại giống, phân loại hoa quả,...

Bài toán phân loại trái cây tươi hay hư ra đời vì trái cây nếu không được bảo quản đúng cách thì sẽ rất dễ bị hư, vì thế để đảm bảo chất lượng của trái cây và sức khỏe của người ăn thì bài toán rất cấp thiết.

Bài toán được ứng dụng rất nhiều như phân loại các trái cây sau khi phân loại ở trạng thái nào để có thể mang ra ngoài tiêu thụ, hay ở trong các siêu thị khi nhập hàng lượng lớn các loại trái cây thì sẽ có khả năng xuất hiện trái cây hư, vì thế bài toán này sẽ giải quyết việc phân loại trái cây tươi hoặc hư để mang đến cho an toàn cho sức khỏe của người dùng khi dùng cái loại thực phẩm nhanh hư như trái cây.

Việc phân loại trái cây tươi và trái cây hư là giảm lượng thực phẩm bị lãng phí, cải thiện an toàn thực phẩm và tiết kiệm thời gian bằng cách nhanh chóng xác định được trái cây tươi và hư. Bằng cách giảm lượng thực phẩm bị lãng phí, ứng dụng có thể có

tác động lớn đến ngành thực phẩm. Giúp đảm bảo rằng các sản phẩm trái cây và trái cây hư đều được đặt theo các nhóm đúng để phù hợp với các tiêu chuẩn của thị trường. Ngoài ra, nó tiết kiệm thời gian bằng cách nhanh chóng xác định được trái cây tươi và hư, giảm thời gian cần thiết cho việc kiểm tra thủ công.

2.2.4. Ứng dụng

1. *Giúp kiểm soát chất lượng thực phẩm*: Phân loại trái cây giúp kiểm soát chất lượng thực phẩm và đảm bảo rằng sản phẩm đến tay người tiêu dùng là an toàn và tươi ngon.
2. *Giảm lãng phí thực phẩm*: Phân loại trái cây giúp loại bỏ những sản phẩm đã hư hỏng hoặc có chất lượng kém, giảm thiểu lãng phí thực phẩm và tối ưu hóa sản xuất.
3. *Tăng năng suất*: Phân loại trái cây giúp tăng năng suất trong quá trình sản xuất. Với sự trợ giúp của máy móc tự động, việc phân loại có thể được thực hiện nhanh chóng và chính xác hơn, giúp tăng năng suất sản xuất.
4. *Nâng cao giá trị thương mại*: Sản phẩm được phân loại theo đúng chất lượng sẽ nâng cao giá trị thương mại của chúng. Điều này giúp tăng lợi nhuận cho các doanh nghiệp và nâng cao chất lượng cuộc sống cho người tiêu dùng.
5. *Tăng tính cạnh tranh*: Sử dụng máy móc tự động để phân loại trái cây tươi và hư sẽ giúp các doanh nghiệp nông nghiệp và thực phẩm cạnh tranh với những đối thủ khác trên thị trường bằng cách cung cấp sản phẩm chất lượng tốt hơn và giá cả cạnh tranh hơn.

3. Mục tiêu đề tài

Mục tiêu của *Fresh and Rotten fruit classification* là phát hiện, xác định tình trạng của loại quả. Một số mục tiêu cụ thể của *Fresh and Rotten fruit classification* là:

- Phát hiện loại quả trong bức ảnh được chụp từ nhiều thiết bị khác nhau
- Nhận diện được loại quả trong nhiều điều kiện chụp khác nhau (ánh sáng, góc chụp, ...)

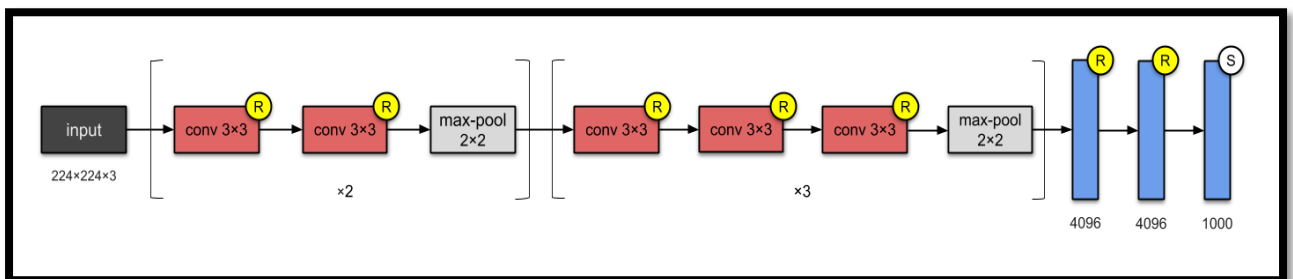
CHƯƠNG II: VGG16

1. Giới thiệu

Mạng VGG16 là một kiến trúc mạng CNN (Convolutional Neural Network) được đề xuất vào năm 2014, được thiết kế với quan điểm quan điểm về một mạng nơ ron sâu hơn sẽ giúp ích cho cải thiện độ chính xác của mô hình tốt hơn và giới thiệu 1 khái niệm lần đầu xuất hiện là khối tích chập (block).

VGG16 được sử dụng trong cuộc thi ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) vào năm 2014 và giành được giải nhất trong hạng mục Object Detection và giải nhì trong hạng mục Classification.

Mạng VGG16 được thiết kế với quan điểm cho rằng một mạng nơ ron sâu hơn sẽ giúp ích cho cải thiện độ chính xác của mô hình tốt hơn và giới thiệu 1 khái niệm lần đầu xuất hiện là khối tích chập (block), gồm nhiều Convolutional Layer kết hợp với nhau sau đó là 1 lớp Pooling.



Hình II.1: Kiến trúc mạng VGG16

2. Kiến trúc mạng VGG16:

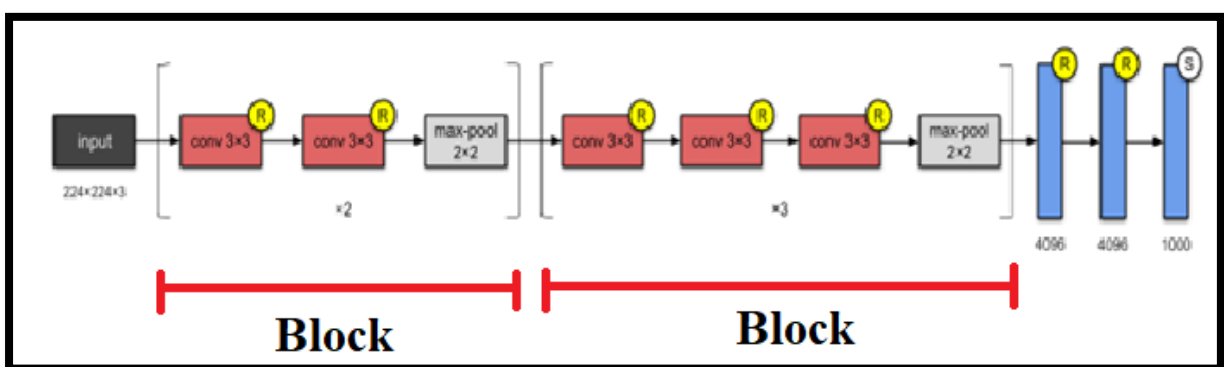
Mạng VGG16 bao gồm: 13 Lớp tích chập (Convolutional Layer), 5 lớp Pooling (Pooling Layer), 1 lớp Flatten (Flatten Layer), 3 Lớp Dense (Dense Layer) và cuối cùng là 1 lớp Dropout được nhóm thêm vào để cải thiện việc training:

- Convolutional Layer:

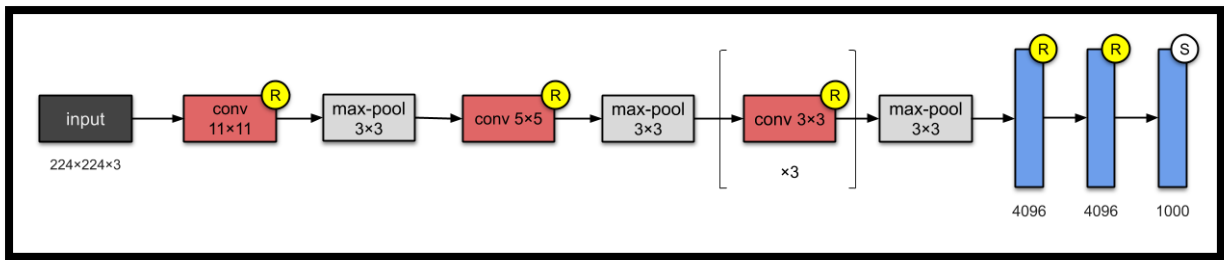
- + Đây là 1 trong những thành phần quan trọng nhất của mạng VGG16 nói riêng và họ mạng CNN nói chung. Nhiệm vụ chính của các Convolutional Layer là khi đưa hình ảnh vào Convolutional Layer, thông qua các bộ lọc (Filter) chúng

sẽ thực hiện việc trích xuất ra các đặc trưng của tấm hình, kết quả sau khi trích xuất, ta được 1 Feature Map của hình ảnh đó.

- + Điểm đặc biệt ở VGG16 là sử dụng bộ lọc là các ma trận có kích thước 3×3 , điều này khác so với người tiền nhiệm của nó là AlexNet (2012), để giúp giảm số lượng tham số cho mô hình, tăng hiệu quả tính toán hơn.
- Pooling Layer:
 - + Thành phần quan trọng tiếp theo cấu tạo nên mạng. Trong VGG16, các Pooling Layer được chọn là Max Pooling với kích thước filter là 2×2 và bước trượt (stride) là 2.
 - + Sau khi đi qua các Convolutional Layer, các Feature Maps được tạo ra và đi đến Pooling Layer, khi filter đến 1 vùng trong Feature Map, tại đây vùng này sẽ chọn giá trị lớn nhất trong khu vực vùng bao phủ và đưa vào vị trí tương ứng trong Feature Map mới. Kết quả là giúp làm giảm kích thước Feature Map, và theo đó giảm số lượng tham số trong mô hình và giúp mô hình đơn giản hơn.
 - + Điểm đặc biệt nhất của VGG16 là sự xuất hiện của khối tích chập (block), khác với các mô hình CNN trước đó, sau 1 lớp Convolutional Layer là 1 lớp Pooling Layer. Các Convolutional Layers xếp chồng lên nhau cho đến khi gặp 1 Pooling Layer được gọi là 1 Block.

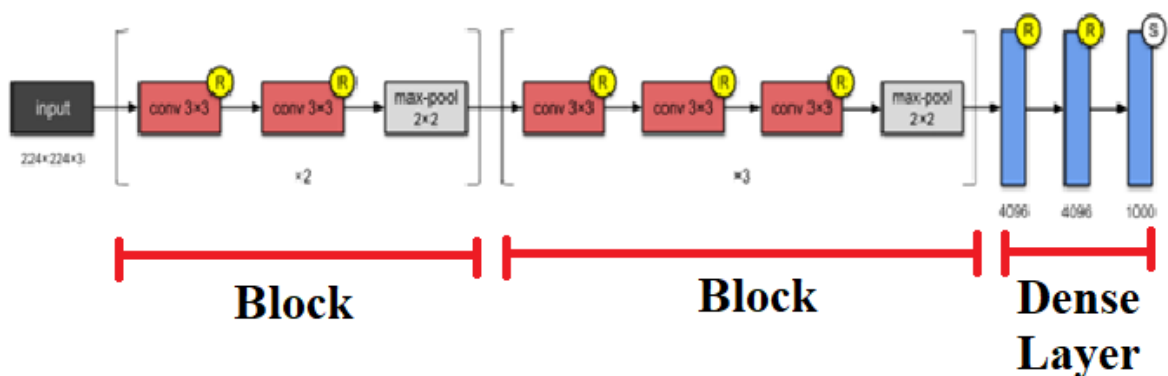


Hình II.1: Các Blocks ở trong mạng VGG16



Hình II.2: Kiến trúc mạng AlexNet. So với người tiền nhiệm của mình là AlexNet (2012), VGG16 đã có bước đột phá với sự xuất hiện của kiến trúc Blocks

- Việc xếp chồng các Convolutional Layers giúp trích xuất đặc trưng tốt hơn, so với chỉ 1 Convolutional Layer theo sau đó là 1 Pooling Layer.
- Flatten Layer:
 - + Trước khi đưa đến Dense Layer để thực hiện việc tính toán và đưa ra kết quả dự đoán của mô hình, chúng ta cần phải chuyển các ma trận thành các vector ma trận 1 chiều để phù hợp với yêu cầu đầu vào của Dense Layer.
 - + Là nơi để chuyển các Feature Maps thành các vector qua đó giúp đưa các Feature Map đến được lớp cuối cùng. Mặc dù trong hình minh họa kiến trúc VGG16 không có hiển thị Flatten Layer, nhưng vị trí của Layer này sẽ nằm ở ngay sau block cuối cùng của mạng VGG16.

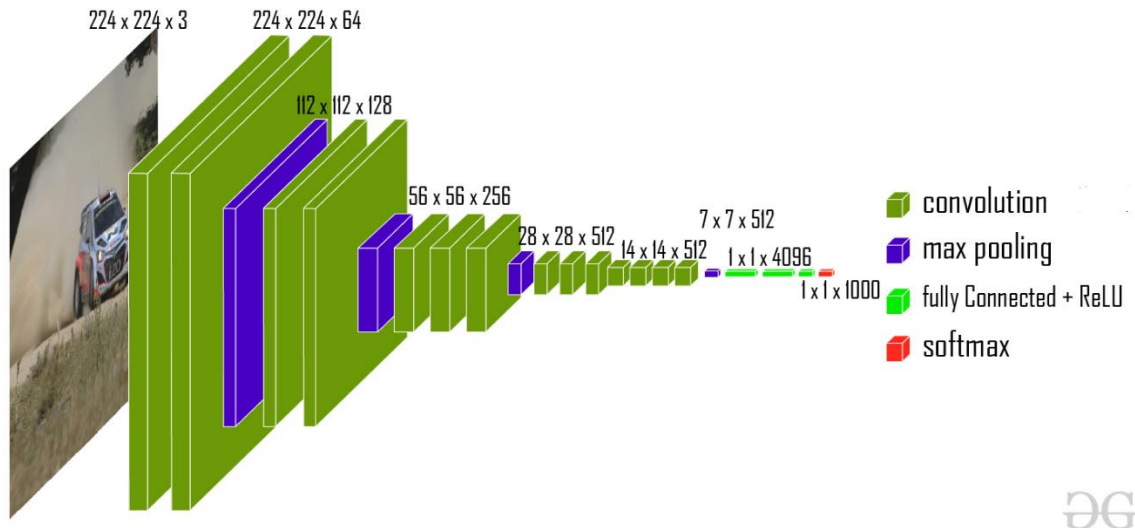


Hình II.3. Dense Layer, lớp cuối cùng thực hiện việc tính toán và đưa ra dự đoán của mạng VGG16

- Dense Layer: Còn có tên gọi khác là Fully Connected Layer, thành phần quan trọng cuối cùng của mạng VGG16.
- Tại đây chúng ta có hai loại Dense Layer gồm: ReLU Dense Layer (kí hiệu là R trong hình minh họa) và Softmax Dense Layer (kí hiệu là S):
 - + ReLU Dense Layer: có vai trò là 1 activation function trong kiến trúc VGG16. Kế thừa từ AlexNet, việc sử dụng ReLU thay cho hàm Sigmoid giúp gia tăng tốc độ tính toán. Đầu ra của ReLU dense layer này là kết quả để mô hình có thể tính toán và đưa ra dự đoán ở Softmax Dense Layer.
 - + Softmax Dense Layer: Layer cuối cùng trong mạng VGG16, tại đây sau khi nhận kết quả từ các ReLU Dense Layers, việc tính toán và đưa ra kết quả dự đoán xác suất cho từng nhãn được thực hiện tại đây. Số lượng neurals trong tầng này phụ thuộc vào bài toán.
- Đối với bài toán của nhóm chúng em, là việc thực hiện dự đoán loại quả được chụp trong bức ảnh đầu vào là loại nào trong 8 loại quả được học, ứng với 8 labels với xác suất riêng từng nhãn. Vì vậy trong Softmax Dense Layer của nhóm chúng em cũng sẽ gồm 8 neurals.
- Ngoài ra, nhóm chúng em cũng thêm vào 1 lớp Dropout trong mô hình mạng VGG16 sau khi thực hiện việc tính toán ở các ReLU Dense Layers trước khi đưa ra kết quả tính toán vào Softmax Dense Layer với mục đích tránh việc Overfitting trong quá trình training mô hình.

3. Mô tả cách hoạt động

Mô tả lại cách thức hoạt động khi đưa 1 bức ảnh vào mạng VGG16 đối với bài toán phân loại trái cây của nhóm.

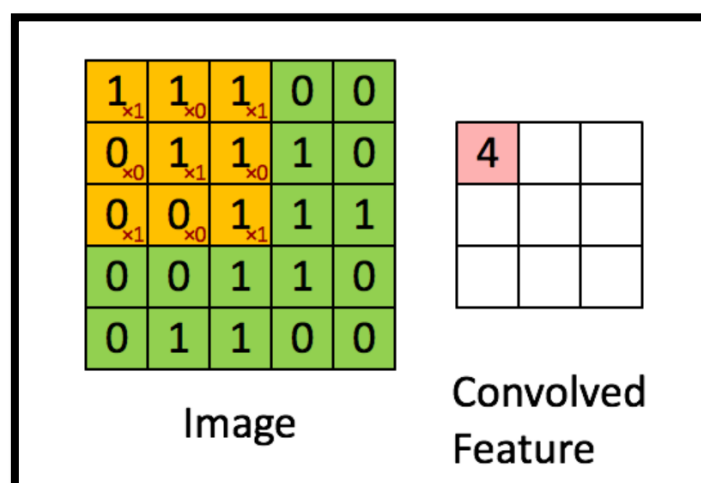


Hình II.4: Mạng VGG16

Ta sẽ tiến hành quan sát block đầu tiên của mạng VGG16, gồm 2 Convolutional Layer và 1 Pooling Layer để mô tả cách hoạt động ở phần trích xuất đặc trưng của mạng:

- Khi bức ảnh được đưa vào mạng VGG16, kích thước đầu vào yêu cầu bức ảnh có kích thước 224×224 với 3 kênh màu RGB hay $224 \times 224 \times 3$, vì nhóm sử dụng phương pháp transfer learning, sử dụng mô hình VGG16 đã được học các trọng số quan trọng từ tập dữ liệu ImageNet, nhưng ảnh từ tập dữ liệu ImageNet là 224×224 với 3 kênh màu RGB
- Sau đó bức ảnh được đưa đến Convolutional Layer, Layer này sử dụng các bộ lọc Filter với kích thước 3×3 để trích xuất đặc trưng ra khỏi bức ảnh. Tuy nhiên với bộ lọc kích thước 3×3 , có thể sẽ xảy ra việc mất mát thông tin trong quá trình trích xuất đặc trưng.

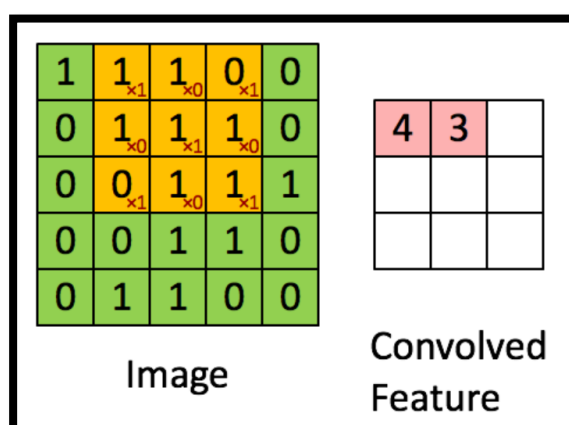
- Việc tính tích chập được thực hiện bằng cách sử dụng Filter bắt đầu từ việc đặt Filter trùng với biên trái và biên trên của ma trận cần tính. Sau đó tiến hành nhân các phần tử tương ứng của ma trận với phần tử tương ứng của bộ lọc và sau đó cộng tất cả chúng lại. Quá trình thực hiện được minh họa như sau:



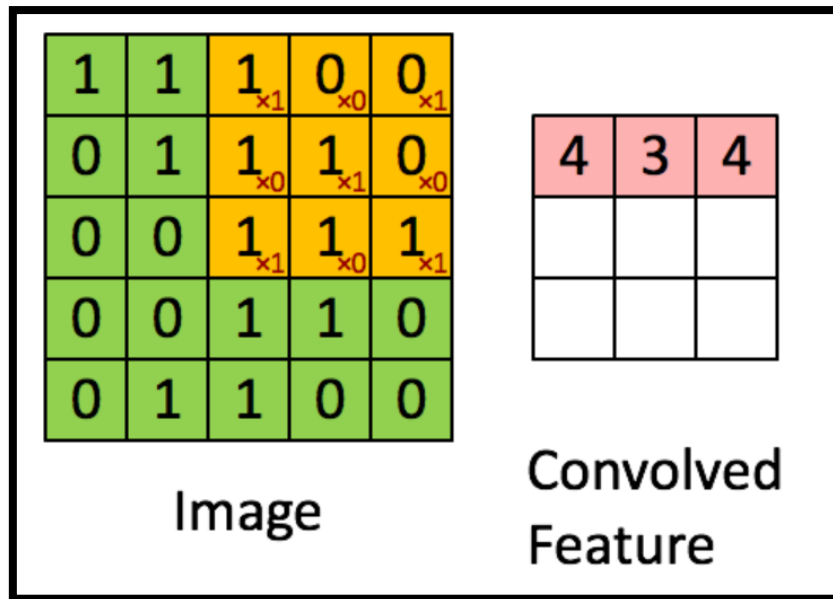
Hình II.5. Bắt đầu việc tính tích chập. Bằng việc nhân các phần tử tương ứng của ma trận với phần tử tương ứng trong Filter và cộng tất cả chúng lại.

- Để tính giá trị đầu tiên trong Feature map, ta thực hiện phép tính như sau:

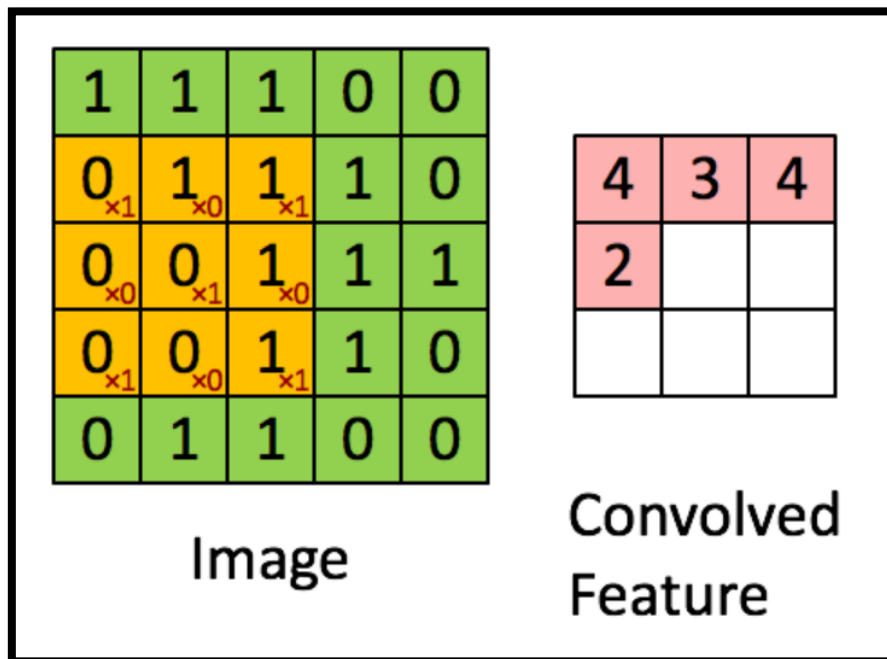
$$1 \times 1 + 1 \times 0 + 1 \times 1 + 0 \times 0 + 1 \times 1 + 1 \times 0 + 0 \times 1 + 0 \times 0 + 1 \times 1 = 4$$
- Như vậy ta được giá trị đầu tiên trong feature map. Tiếp theo ta sẽ cho filter trượt qua trái bằng với stride, trong model VGG16, do stride = 1, vậy nên kết quả của việc này là:



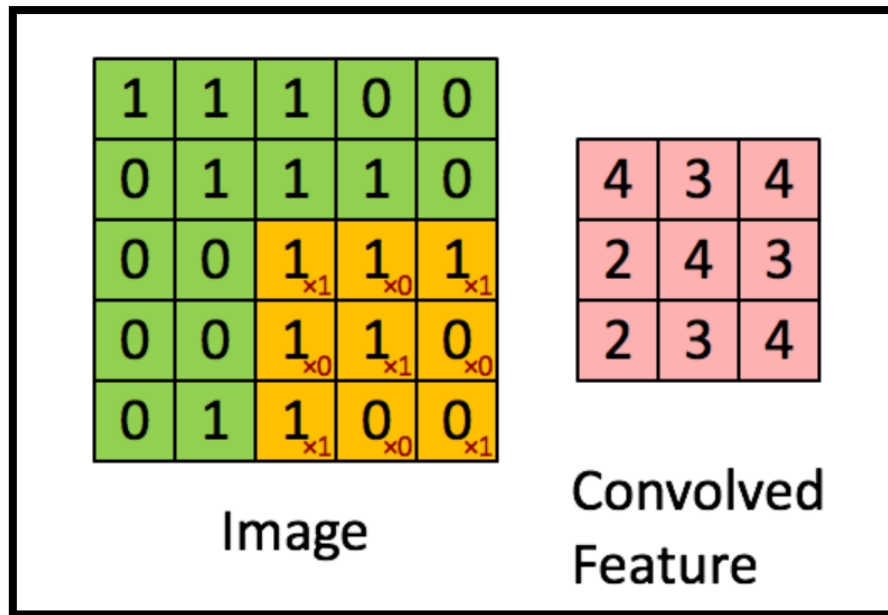
Hình II.6. Trượt filter qua trái bằng với stride, và tính tiếp giá trị tiếp theo trong Feature Map với cách làm tương tự như trên, ta thu được phần tử thứ 2 trong Feature map



Hình II.7. Thực hiện tương tự, ta cũng trượt filter qua trái, và tính được giá trị tiếp theo. Nhưng ở đây chúng ta đã đụng biên phải của ma trận. Vì vậy ta sẽ dịch filter lại trùng với biên trái và dịch filter xuống 1 dòng



Hình 8. Kết quả sau khi thực hiện việc trượt Filter được đề cập ở hình 8. Bây giờ ta sẽ tiếp tục lặp lại các bước đã được minh họa ở các hình 6, 7, 8, 9 cho đến khi hoàn thành việc tính tích chập



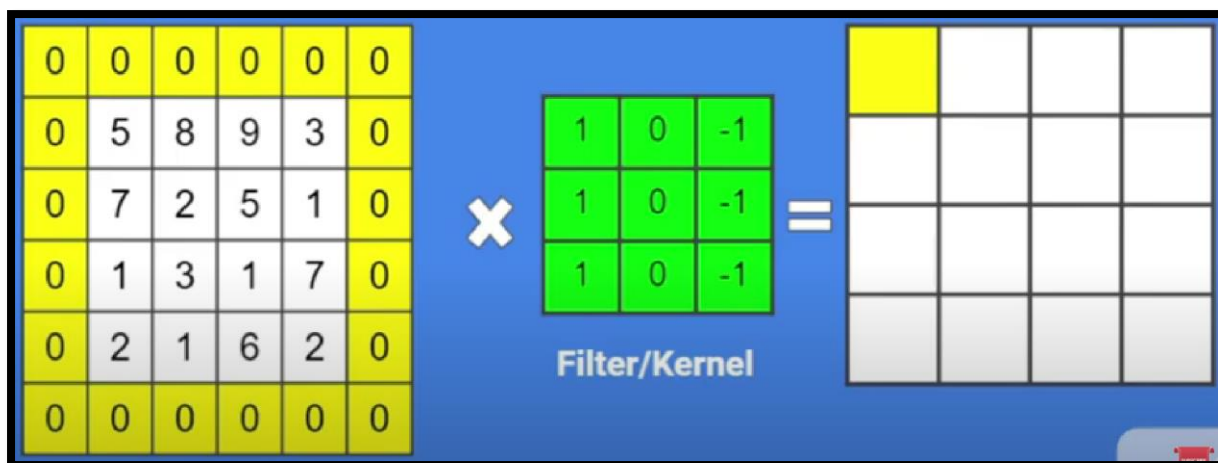
Hình II.9. Feature Map ta nhận được sau khi hoàn thành tính tích chập tất cả các phần tử trong ma trận sau khi Filter chạm biên dưới và biên phải của ma trận

- Tuy nhiên ta nhận thấy, so với ma trận đầu vào kích thước đã bị giảm đi. Xét trong việc thực hiện tích chập để lấy các đặc trưng của ảnh, điều này đồng nghĩa việc có thể đánh mất đặc trưng quan trọng trong quá trình trích xuất.
- Để khắc phục điều đó, ta sẽ sử dụng phương pháp Padding, tức là thêm các giá trị 0 vào viền của input image để giữ nguyên kích thước của output feature map, được sử dụng để không bỏ sót đặc trưng trong quá trình trích xuất đặc trưng. Việc tính toán Padding được thực hiện dựa theo công thức sau:

$$padding = \frac{filter_size - 1}{2}$$

Trong đó:

- *filter_size* là kích thước của bộ lọc (Filter)
- *padding* là số lớp bao quanh ma trận, lớp bao quanh này mang giá trị 0 cần được thêm vào để đảm bảo sau khi trích xuất đặc trưng, feature map kết quả sẽ có kích thước bằng với kích thước bức ảnh đầu vào



Hình 11. Minh họa việc thực hiện Padding trong Convolutional Layer của VGG16.

Nguồn ảnh: video youtube StudyGyann

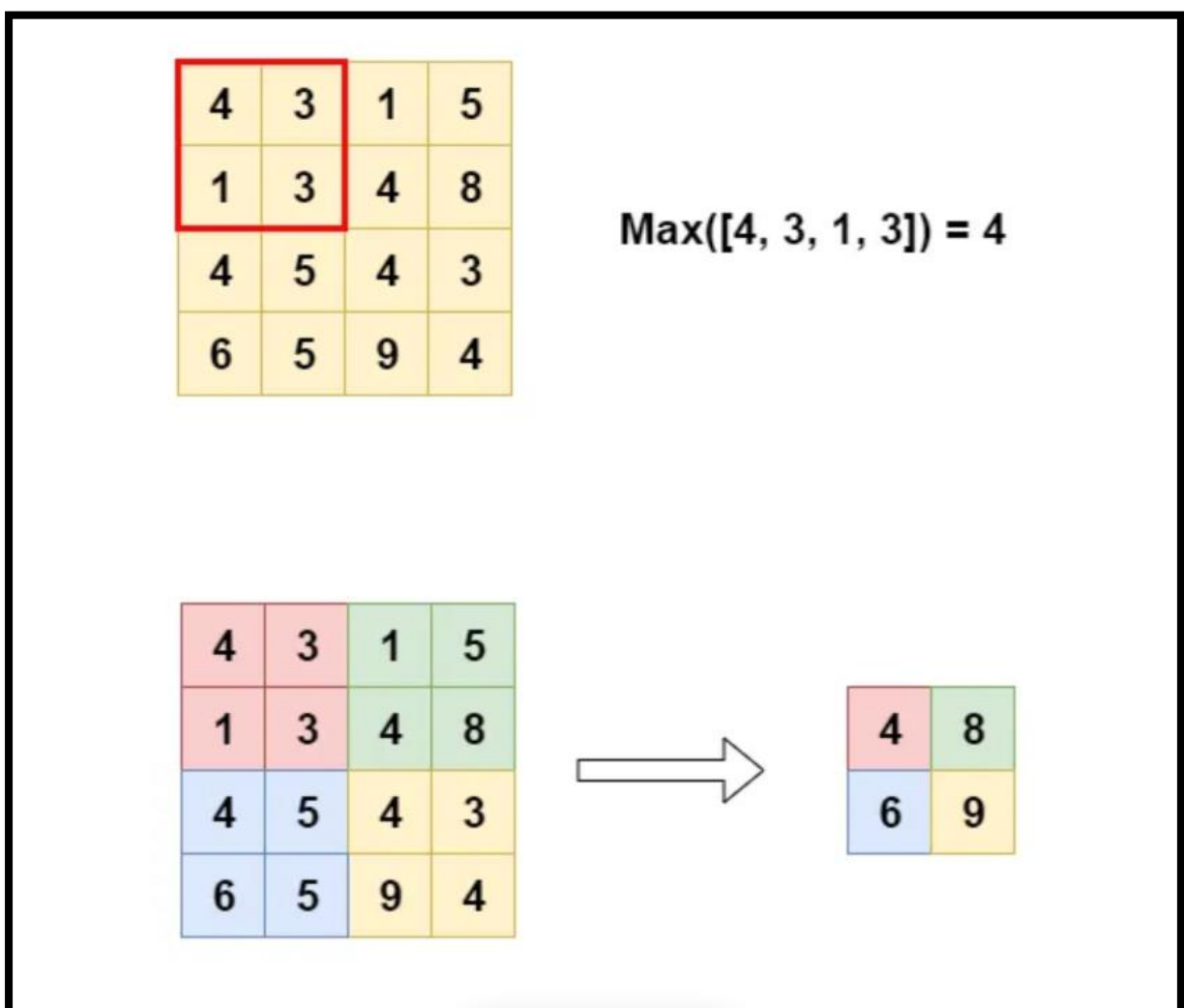
Ở hình 11, do filter có kích thước là 3×1 , vậy nên theo công thức ta có padding = $(3 - 1) / 2 = 1$, vì vậy ta thêm 1 lớp mang giá trị 0 bao quanh ma trận, vậy nên sau khi tích chập, kết quả thu được là 1 Feature map có kích thước 4×1 , bằng với kích thước ma trận đầu vào là 4×1 .

Việc thực hiện padding đảm bảo kích thước Feature map không bị thay đổi so với kích thước của bức ảnh đầu vào, đảm bảo việc trích xuất đặc trưng trong ảnh không bị thiếu sót, làm ảnh hưởng đến kết quả sau này của cả hệ thống

Sau khi hình ảnh qua được Convolutional Layer đầu tiên, ta sẽ thu được 64 Feature maps có kích thước 224×224 . Lí do ta có 64 Feature maps sau khi trích xuất đặc trưng ở Convolutional Filter đầu tiên là do, Convolutional Filter đầu tiên có 64 Filter 3×1 hoạt động, do đó sau khi trích xuất đặc trưng, mỗi 1 Filter cho ra 1 Feature map, ta sẽ thu được 64 Feature maps.

Điều tương tự đối với Convolutional Layer tiếp theo, do cũng có 64 Filter vậy nên kết quả của nó cũng sẽ là $224 \times 224 \times 64$.

- Sau đó ta sẽ đến lớp Pooling đầu tiên, trong VGG16, loại Pooling được sử dụng là Max Pooling với $\text{stride} = 2$. Nhiệm vụ của lớp Pooling sẽ giảm kích thước của feature map, giảm số lượng tham số và giúp trích xuất đặc trưng tổng quát hơn. Việc sử dụng Max Pooling nhằm giúp giữ lại thông tin quan trọng nhất trong của Feature Map ở mỗi vùng được quét qua, với $\text{stride} = 2$, mỗi lần hoàn thành Pooling, kích thước của Feature maps sẽ giảm đi 1 nửa:



Hình III.12. Minh họa kết quả sau khi đưa Feature map qua Pooling Layer trong VGG16

Vậy là ta đã hoàn thành xong Block đầu tiên của mạng VGG16, đối với các Block còn lại. Việc thực hiện cũng tương tự như Block 1, tuy nhiên đối với các Block sau, số lượng Filters của Convolutional Layer của chúng là gấp đôi so với Block trước của chúng, như Convolutional Layer ở Block 2 có 128 Filters, còn ở Block 3 con số đó là 256.

- Sau khi qua được tất cả các Block, ta hoàn thành việc trích xuất đặc trưng, việc tiếp theo là phải chuyển các Feature maps thành vector 1 chiều, vì lớp Dense yêu cầu việc input đầu vào phải là 1 vector để có thể tính toán
- Tiếp theo đó, ta sẽ đưa các vector vào các Dense Layer của VGG16 để đưa ra kết quả cuối cùng. Dense Layer của VGG16 gồm 2 ReLU dense layer và 1 Softmax Dense Layer, thêm vào đó nhóm chúng em cũng thêm vào 1 lớp Dropout:
 - + Nhiệm vụ của ReLU dense layer trong VGG16 đóng vai trò là activation function, vì ReLU là 1 hàm phi tuyến đóng vai trò quyết định có kích hoạt neural để cho mô hình học hay không, qua đó giúp tạo ra các đặc trưng phi tuyến, giúp mạng học được các quan hệ không tuyến tính giữa các đặc trưng trong dữ liệu. Công thức của hàm ReLU là:

$$f(x) = \max(0, x)$$

Trong đó:

- $f(x)$ là giá trị đầu ra của hàm ReLU với đầu vào x
- x là giá trị đầu vào của hàm ReLU
- + Sau đó trước khi đưa vào Softmax Layer
- + Cuối cùng các kết quả của hàm ReLU sẽ được sử dụng trong hàm Softmax Dense Layer, đây là nơi tính toán và đưa ra kết quả dự đoán cuối cùng của mạng VGG16 cho bài toán Classification của nhóm.
- + Kết quả đầu ra của Softmax Dense Layer là 1 vector có độ dài là 8, ứng với 8 loại quả được nhóm em sử dụng cho mô hình học. Các giá trị trong vector đại diện cho xác suất của từng label mà mô hình dự đoán loại quả trong bức ảnh có thể thuộc về label nào.

CHƯƠNG III: DATASET

1. Bộ dữ liệu trên Mendeley Data

1.1. Giới thiệu

Phần này nhóm chúng em xin giới thiệu sơ lược về bộ dữ liệu *Fresh and Rotten Fruits Dataset for Machine-Based Evaluation of Fruit Quality*. Bộ dữ liệu được lấy từ web *Mendeley Data* được đăng tải vào ngày 8 tháng 4 năm 2022 bởi nhóm tác giả *Nusrat Sultana, Musfika Jahan, Mohammad Shorif Uddin* thuộc *Jahangirnagar University*.

Web *Mendeley Data* là một nền tảng lưu trữ và chia sẻ dữ liệu khoa học, trực tuyến được phát triển Elsevier, nó cung cấp cho các nhà nghiên cứu và tổ chức khoa học một cách để quản lý, lưu trữ dữ liệu một cách dễ dàng và đáng tin cậy. Ảnh được nhóm tác giả chụp bằng máy ảnh (Nikon D5600) ở các sạp trái cây ở Bangladesh với điều kiện khi chụp ảnh là ánh sáng, góc chụp bất kì.









Bộ dữ liệu gồm 15,535 ảnh đã được gán nhãn bao gồm 16 nhãn dán. Bộ dữ liệu *Fresh and Rotten Fruits Dataset for Machine-Based Evaluation of Fruit Quality* được nhóm tác giả phân chia thành 2 bộ dữ liệu khác nhau là: Ảnh gốc [*Original Images* (3200)] và ảnh đã qua tiền xử lý [*Augmented Images* (12,335)]. Đối với bộ bài toán này, nhóm chúng em chỉ sử dụng bộ ảnh gốc (*Original Images*) gồm 3200 ảnh.



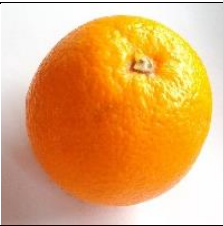




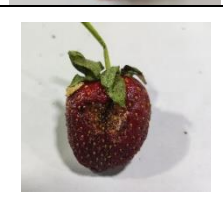
Bộ dữ liệu *Fresh and Rotten Fruits Dataset for Machine-Based Evaluation of Fruit Quality* có cả những bức ảnh có nhiều quả. Chi tiết về các hình ảnh và các nhãn tương ứng sẽ được mô tả ở phần dưới đây.

1.2. Mô tả chi tiết

Những hình ảnh trong bộ dữ liệu được chụp bởi chính nhóm tác giả. Được phân làm 16 nhóm là: táo tươi (*Fresh Apple*), chuối tươi (*Fresh Banana*), nho tươi (*Fresh Grape*), ổi tươi (*Fresh Guava*), táo tàu tươi (*Fresh Jujube*), cam tươi (*Fresh Orange*), lựu tươi (*Fresh Pomegranate*), dâu tươi (*Fresh Strawberry*), táo hư (*Rotten Apple*), chuối hư (*Rotten Banana*), nho hư (*Rotten Grape*), ổi hư (*Rotten Guava*), táo tàu hư (*Rotten Jujube*), cam hư (*Rotten Orange*), lựu hư (*Rotten Pomegranate*), dâu hư (*Rotten Strawberry*) và số lượng ảnh của mỗi nhóm là 200 tấm ảnh và đã được nhóm tác giả chia sẵn thành 16 nhóm ảnh riêng biệt.

Dưới đây là bảng mô tả một số đặc điểm của các loại trái cây tươi và trái cây hư:





Loại	Đặc điểm	Hình ảnh
táo tươi (<i>Fresh Apple</i>)	Vỏ của táo tươi thường có màu đỏ, vàng, xanh, hồng, hoặc màu nâu đỏ, tuy nhiên có một số giống có hai hoặc ba màu và có thể có những đốm nhỏ.	
táo hư (<i>Rotten Apple</i>)	Vỏ của quả táo hư có thể có một hoặc nhiều vết thâm đen hoặc nâu, thường có một lỗ khá lớn có màu nâu hoặc đen.	
chuối tươi (<i>Fresh Banana</i>)	Có hình dáng cong, tùy vào độ chín của quả vỏ có thể là màu xanh lá cây hoặc chuyển dần sang màu vàng, có thể có những đốm đen hoặc nâu.	
chuối hư (<i>Rotten Banana</i>)	Có hình dáng cong, vỏ chuyển dần sang màu vàng đục có nhiều vết thâm đen hoặc nâu trên bề mặt quả.	
nho tươi (<i>Fresh Grape</i>)	Vỏ thường có màu xanh. Trái đầy đặn.	
nho hư (<i>Rotten Grape</i>)	Vỏ xuất hiện màu nâu sẫm hoặc đen, có thể nhăn nheo hơn và có thể xuất hiện 1 số vết nứt.	
ổi tươi (<i>Fresh Guava</i>)	Có hình tròn hoặc hình cầu, vỏ thường có màu xanh, có 1 số quả sẽ ngả dần sang màu vàng có thể có những đường màu xanh nhạt ở giữa, căng bóng.	
ổi hư (<i>Rotten Guava</i>)	Có hình tròn hoặc hình cầu, vỏ ngoài có những vết thâm đen hoặc nâu, nhăn nheo, không còn bóng như khi tươi.	

táo tàu tươi (<i>Fresh Jujube</i>)	Thường có hình nón đầu nhọn hoặc hình cầu, vỏ có sự pha trộn giữa các màu đỏ, xanh lá và vàng.	
táo tàu hư (<i>Rotten Jujube</i>)	Thường có hình nón đầu nhọn hoặc hình cầu, vỏ ngoài có những vết thâm đen hoặc nâu, nhẵn nhéo, không còn bóng như khi tươi, có thể xuất hiện một số vết nứt	
cam tươi (<i>Fresh Orange</i>)	Có hình tròn dẹt, vỏ thường là màu cam, một trái có màu xanh hoặc chuyển dần từ xanh qua cam.	
cam hư (<i>Rotten Orange</i>)	Có hình tròn dẹt, vỏ có vết thâm đen hoặc nâu, có thể có màu bạc mờ và những vết nhăn. Hoặc bị mất đi hình dạng ban đầu.	
lựu tươi (<i>Fresh Pomegranate</i>)	Vỏ có màu đỏ thẫm, thay đổi từ màu vàng sang màu tím.	
lựu hư (<i>Rotten Pomegranate</i>)	Vỏ có vết thâm đen, có thể có vết nứt	
dâu tươi (<i>Fresh Strawberry</i>)	Thường có hình nón, vỏ có màu đỏ, có các chấm vàng trên vỏ	
dâu hư (<i>Rotten Strawberry</i>)	Có vỏ ngoài thường nhẵn nhéo, thâm đen có thể xuất hiện vết bạc mờ	

Bảng III.1: Một số đặc trưng của bộ dữ liệu

1.3. Các khó khăn khi sử dụng bộ dataset

Vì bộ dataset được chụp từ các sạp trái cây ở các điều kiện chụp khác nhau nên ảnh trong bộ dataset sẽ ở trong các điều kiện trạng thái, ánh sáng và góc chụp và số lượng khác nhau. Một số loại quả ở trạng thái tươi và hư chỉ có thể nhận diện thông qua các vết sạm trên vỏ của quả hoặc kích thước quá nhỏ nên khó nhận biết được trạng thái tươi hoặc hư của loại quả chỉ thông qua hình dạng, một số trái có hình dạng giống nhau.

Fresh Fruit	Rotten Fruit
	
	

2. PreProcessing

Để training mô hình máy học, cần 1 lượng dữ liệu có thể bao quát được một số trường hợp phổ biến như: ảnh chụp nghiêng, ảnh được chụp trong điều kiện thiếu sáng hay ánh sáng quá mạnh, máy ảnh mờ, ... Để đáp ứng với nhu cầu của dữ liệu nhóm chúng em sử dụng phương pháp như xoay ảnh, điều chỉnh mức sáng, điều chỉnh độ tương phản, điều chỉnh độ nét của bức ảnh để làm đầy bộ dữ liệu.

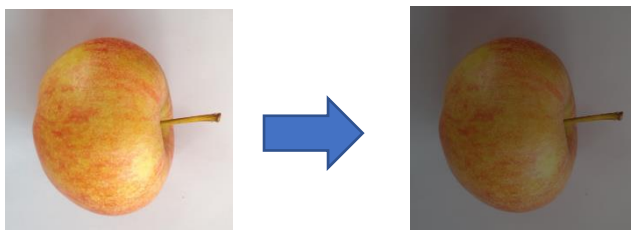
- Xoay ảnh

- + Chuyển đổi hình ảnh bằng cách xoay hình ảnh đó theo một góc độ xác định.
- + Xoay ảnh để tạo ra các góc khác nhau của bức ảnh.

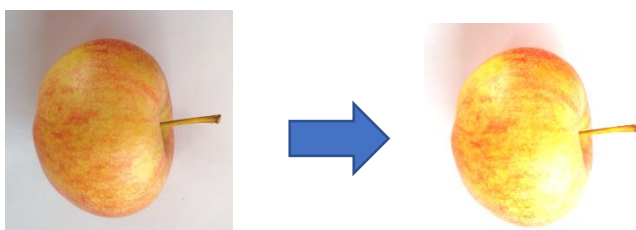


- Điều chỉnh mức sáng

- + Điều chỉnh độ sáng của ảnh
- + Nếu giá trị < 1 → ảnh trở nên tối hơn, nếu hệ số bằng 0 → ảnh màu đen.



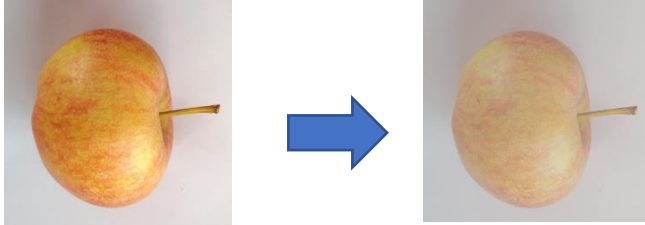
- + Giá trị > 1 → ảnh được làm sáng



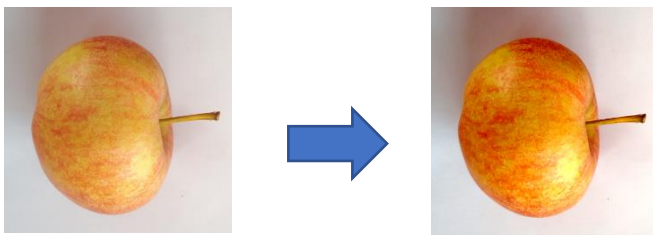
- Điều chỉnh độ tương phản

+ Điều chỉnh độ tương phản của ảnh

+ Nếu giá trị $< 1 \rightarrow$ ảnh trở nên xám hơn



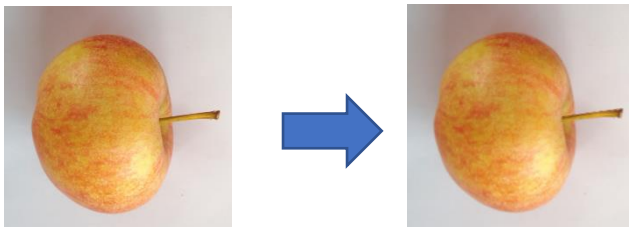
+ Nếu giá trị $> 1 \rightarrow$ tăng độ tương phản của ảnh



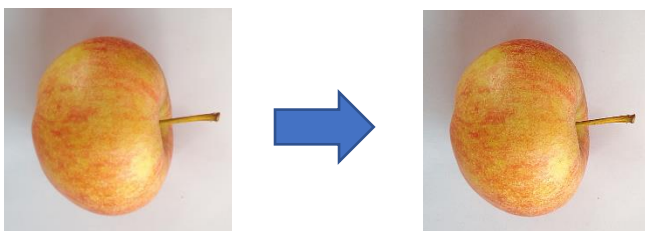
- Điều chỉnh độ nét

+ Điều chỉnh độ nét của ảnh

+ giá trị $< 1 \rightarrow$ ảnh trở nên nhòe hơn



+ giá trị $> 1 \rightarrow$ ảnh trở nên nét hơn



CHƯƠNG IV: HUẤN LUYỆN VÀ KIỂM THỬ MÔ HÌNH

1. Lý do chọn mô hình

Sau khi phân tích bộ dữ liệu sử dụng, chúng tôi quyết định sử dụng mạng VGG16. Vì mạng VGG16 là mô hình có mạng CNN nên có thể trích xuất đặc trưng ảnh sau khi đi qua lớp của mạng. VGG16 còn là một mạng đã được huấn luyện sẵn trên dữ liệu của imagenet, vì thế VGG16 còn có thể trích xuất các đặc trưng cần thiết để có thể thích ứng với đa dạng yêu cầu của những bài toán cùng lĩnh vực phân loại và nhận dạng.

2. Preprocess dataset

Dataset gốc có tổng cộng 3200 tấm ảnh, được chia thành 16 thư mục được đặt tên theo nhãn của loại quả mà thư mục chứa, mỗi thư mục lưu 200 tấm ảnh có cùng nhãn với tên thư mục.

Chia dataset thành 2 tập train và test với tỉ lệ là train chiếm 60% và test chiếm 40% với tập train dùng để huấn luyện mô hình và tập test dùng để kiểm thử mô hình sau khi huấn luyện. Sau đó lưu trữ tập train và tập test vào 2 thư mục train và test tương ứng.

Thư mục train và test đều có cấu trúc giống với cấu trúc của thư mục lưu trữ bộ dataset gốc là chia thành 16 thư mục.

Thư mục train có tổng cộng 1920 tấm ảnh, mỗi thư mục chứa 120 tấm ảnh có cùng nhãn với tên thư mục.

Thư mục test có tổng cộng 1280 tấm ảnh, mỗi thư mục chứa 80 tấm ảnh có cùng nhãn với tên thư mục.

Với mỗi tập train và test, các tấm ảnh trong được qua các kĩ thuật dùng để biến đổi hình ảnh gồm:

- + Xoay ảnh một góc 30°, 60° và 90°

```
#rotate image
angle = 0 #In degrees counter clockwise
for i in range (0,4):
    im_ro = im.rotate(angle , expand=True)
    count += 1
    im_ro.save(f"{path_image_save}" +str(count) + ".jpg")
    angle = angle + 30 #rotation with 30°, 60°, and 90° angles
```

- + Điều chỉnh mức sáng xuống 1 nửa và tăng mức sáng lên 1 nửa.

```
#Brightness image
for i in (0.5, 1.5): #0.5-darkens the image, 1.5-brightens the image
    im_Br = ImageEnhance.Brightness(im).enhance(i)
    count += 1
    im_Br.save(f"{path_image_save}" +str(count) +".jpg")
```

- + Điều chỉnh độ tương phản xuống 1 nửa và tăng mức sáng lên 1 nửa.

```
#Contrast image
for i in (0.5, 1.5): #0.5-decrease constrast, 1.5-increase contrast
    im_Co = ImageEnhance.Contrast(im).enhance(i)
    count += 1
    im_Co.save(f"{path_image_save}" +str(count) +".jpg")
```

- + Điều chỉnh độ nét của tấm ảnh là -10 và +10

```
#Sharpness image
for i in (-10,10): #-10-blurred image, 10-sharpened image
    im_Sh = ImageEnhance.Sharpness(im).enhance(i)
    count += 1
    im_Sh.save(f"{path_image_save}" +str(count) +".jpg")
```

Thư mục train sau khi qua bước biến đổi hình ảnh:

- Chứa tổng cộng 19,200 tấm ảnh.
- Chia thành 16 thư mục, mỗi thư mục đại diện cho nhãn của dataset và chứa ảnh có nhãn cùng loại.
- Mỗi thư mục chứa 1200 tấm ảnh.

Thư mục test sau khi qua bước biến đổi hình ảnh:

- Chứa tổng cộng 12,800 tấm ảnh.
- Chia thành 16 thư mục, mỗi thư mục đại diện cho nhãn của dataset và chứa ảnh có nhãn cùng loại.
- Mỗi thư mục chứa 800 tấm ảnh.

3. Huấn luyện mô hình

Mô hình được huấn luyện trên nền tảng google colab.

Để load dataset đã được chia thành 2 mục train và test vào huấn luyện mạng, chúng em sử dụng ImageDataGenerator của thư viện keras để đưa dữ liệu từ 2 thư mục train và test để đưa vào huấn luyện và kiểm thử mô hình. Đối với tập train thì tiếp tục được chia thành train và validation với validation chiếm 10% trong tập train dùng để điều chỉnh tham số lý tưởng của mạng mỗi khi train xong 1 iterator.

Data original

```
train_datagen = ImageDataGenerator(rescale=1. / 255, validation_split=0.1)
test_datagen = ImageDataGenerator(rescale=1. / 255)

# load and iterate training dataset has process
train_it_original = train_datagen.flow_from_directory("Data/Split Original/train",
                                                    target_size=(224,224),
                                                    color_mode='rgb',
                                                    class_mode="categorical",
                                                    batch_size=32,
                                                    subset = "training")

# Validation Data has process
val_it_original = train_datagen.flow_from_directory("Data/Split Original/train",
                                                    target_size=(224,224),
                                                    color_mode='rgb',
                                                    class_mode="categorical",
                                                    batch_size=32,
                                                    subset='validation')

# load and iterate test dataset has process
test_it_original = test_datagen.flow_from_directory("Data/Split Original/test",
                                                    target_size=(224,224),
                                                    color_mode='rgb',
                                                    class_mode="categorical")

Found 1728 images belonging to 16 classes.
Found 192 images belonging to 16 classes.
Found 1280 images belonging to 16 classes.
```

Data preprocess

```
train_datagen = ImageDataGenerator(rescale=1. / 255, validation_split=0.2)
test_datagen = ImageDataGenerator(rescale=1. / 255)

# load and iterate training dataset
train_it = train_datagen.flow_from_directory("Data/Split Train_Test/train",
                                             target_size=(224,224),
                                             color_mode='rgb',
                                             class_mode="categorical",
                                             batch_size=32,
                                             subset = "training")

# Validation Data
val_it = train_datagen.flow_from_directory("Data/Split Train_Test/train",
                                           target_size=(224,224),
                                           color_mode='rgb',
                                           class_mode="categorical",
                                           batch_size=32,
                                           subset='validation')

# load and iterate test dataset
test_it = test_datagen.flow_from_directory("Data/Split Train_Test/test",
                                           target_size=(224,224),
                                           color_mode='rgb',
                                           class_mode="categorical")

Found 15360 images belonging to 16 classes.
Found 3840 images belonging to 16 classes.
Found 12800 images belonging to 16 classes.
```

3.1. Cài đặt mô hình huấn luyện

- Lớp đầu của mạng là lớp Input để nhận ảnh đầu vào. Output của lớp này feature có kích thước là [224,224,3] là ảnh đầu vào có chiều dài là 224 pixel và có chiều rộng là 224 pixel và có 3 chiều đại diện cho màu của ảnh.
- Lớp thứ hai là mạng VGG16 đã được huấn luyện sẵn trên dữ liệu của ImageNet. Output của lớp đã được thiết lập cứng kích thước là [7,7,512]. Các trọng số của lớp này được thiết lập không thể thay đổi khi huấn luyện mô hình.
- Lớp thứ ba là lớp Flatten với mục đích là làm phẳng feature từ lớp trước về không gian 1 chiều. Lớp này có kích thước đầu ra là (25088).
- 3 lớp còn theo thứ tự là Dense (256), lớp Dropout(0.5) và lớp Dense(16).
- Tối ưu mô hình bằng Adam optimizer và độ đo đánh giá mô hình là accuracy và loss được đo bằng categorical_crossentropy đều có sẵn trong thư viện keras.
- Huấn luyện mô hình trong 20 epochs.

```
model.summary()
```

```
Model: "model"
```

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 224, 224, 3)]	0
vgg16 (Functional)	(None, 7, 7, 512)	14714688
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 256)	6422784
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 16)	4112

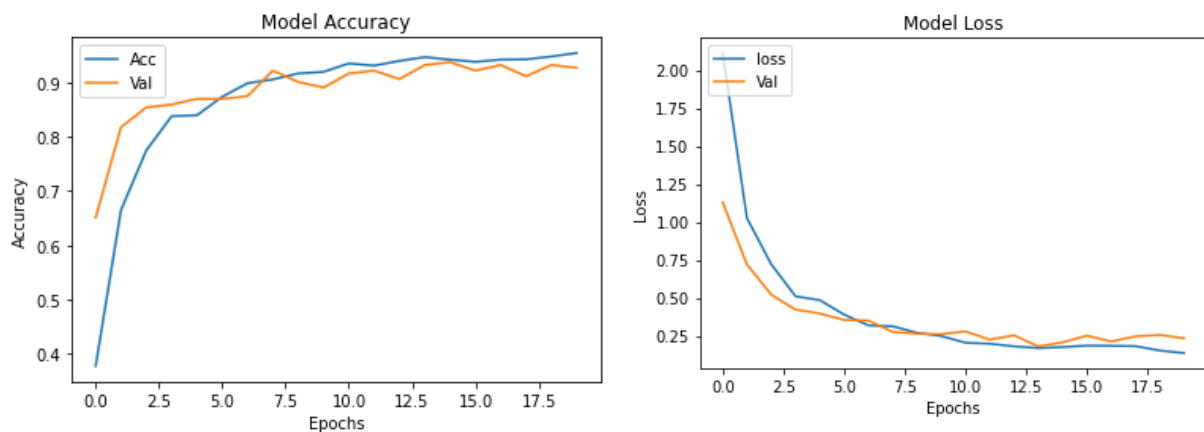
```
=====  
Total params: 21,141,584  
Trainable params: 6,426,896  
Non-trainable params: 14,714,688  
=====
```

```
model.compile(loss='categorical_crossentropy',  
              metrics=["accuracy"],  
              optimizer = 'nadam')
```

3.2. Kết quả huấn luyện

Đối với data original

Epoch 15/20	
54/54 [=====]	- 157s 3s/step - loss: 0.1764 - accuracy: 0.9421 - val_loss: 0.2069 - val_accuracy: 0.9375
Epoch 16/20	
54/54 [=====]	- 157s 3s/step - loss: 0.1858 - accuracy: 0.9381 - val_loss: 0.2515 - val_accuracy: 0.9219
Epoch 17/20	
54/54 [=====]	- 156s 3s/step - loss: 0.1846 - accuracy: 0.9421 - val_loss: 0.2133 - val_accuracy: 0.9323
Epoch 18/20	
54/54 [=====]	- 157s 3s/step - loss: 0.1824 - accuracy: 0.9427 - val_loss: 0.2466 - val_accuracy: 0.9115
Epoch 19/20	
54/54 [=====]	- 156s 3s/step - loss: 0.1537 - accuracy: 0.9479 - val_loss: 0.2560 - val_accuracy: 0.9323
Epoch 20/20	
54/54 [=====]	- 157s 3s/step - loss: 0.1372 - accuracy: 0.9543 - val_loss: 0.2346 - val_accuracy: 0.9271



Đối với data preprocess

Vì tài nguyên GPU của colab cung cấp có hạn, nên chúng em không thể train hết 20 epochs vì mỗi epochs sẽ tốn một lượng đơn vị tính toán GPU đồng thời cần một lượng thời gian rất lớn để hoàn thành việc huấn luyện. Đồng thời vì để thiết lập ban đầu là 20 epochs, nên khi dừng huấn luyện mô hình khi xong epoch 5 thì các giá trị loss và accuracy khi đang train không thể lưu vào biến đã thiết lập.

```
history = model.fit_generator(generator = train_it,
                             steps_per_epoch=train_it.samples/train_it.batch_size,
                             epochs=20,
                             validation_data=val_it,
                             validation_steps=test_it.samples/test_it.batch_size,
                             )

<ipython-input-12-e564f3ff746b>:1: UserWarning: `Model.fit_generator` is deprecated and will
history = model.fit_generator(generator = train_it,
Epoch 1/20
480/480 [=====] - ETA: 0s - loss: 1.2491 - accuracy: 0.5814WARNING:
480/480 [=====] - 5565s 12s/step - loss: 1.2491 - accuracy: 0.5814
Epoch 2/20
480/480 [=====] - 1233s 3s/step - loss: 0.6598 - accuracy: 0.7671
Epoch 3/20
480/480 [=====] - 1225s 3s/step - loss: 0.5150 - accuracy: 0.8178
Epoch 4/20
480/480 [=====] - 1228s 3s/step - loss: 0.4385 - accuracy: 0.8447
Epoch 5/20
480/480 [=====] - 1236s 3s/step - loss: 0.3746 - accuracy: 0.8622
Epoch 6/20
```

4. Đánh giá mô hình huấn luyện

4.1. Phương pháp đánh giá

Để đánh giá mô hình, nhóm sử dụng các độ đo như *Accuracy*, *Precision*, *Recall*, *F1*.

Trước khi đi vào công thức tính toán các độ đo, ta phải tính được các giá trị **True Positive (TP)**, **False Positive (FP)** và **False Negative (FN)**, **True Negative (TN)**.

Trong đó, *positive* là loại quả dự đoán đúng được “tình trạng” và “tên” của loại quả đó, ngược lại nếu dự đoán sai “tình trạng” hoặc “tên” của loại quả được dự đoán là *negative*.

Ví dụ: Ta xét phân lớp “FreshApple”

Nếu loại quả được đưa vào mô hình dự đoán là “FreshApple”

True Positive là dự đoán ra chính xác là “FreshApple”.

False Negative là dự đoán ra không phải là “FreshApple”.

Nếu loại quả được đưa vào mô hình dự đoán không là “FreshApple”

False Positive là dự đoán ra là “FreshApple”.

True Negative là dự đoán ra không phải là “FreshApple”.

Từ các giá trị đã tính toán được, ta có công thức của *Accuracy*:

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

Giá trị *Accuracy* sẽ cho ta biết được mô hình dự đoán được đúng bao nhiêu phần trăm trên toàn bộ phân lớp.

Công thức của *Precision* và *Recall*:

$$Precision = \frac{TP}{TP + FP}; Recall = \frac{TP}{TP + FN}$$

Giá trị *Precision* sẽ cho ta biết được mô hình dự đoán được đúng bao nhiêu phần trăm trong phân lớp đang xét. Giá trị của *Recall* cho ta biết được mô hình dự đoán được bao nhiêu kết quả có trong phân lớp đang được xét.

Bên cạnh ba độ đo, nhóm còn sử dụng một độ đo khác để đánh giá, đó là *F1*. Độ đo này sẽ phản ánh sự tương quan giữa *Precision* và *Recall*. Nếu *Precision* và *Recall* cùng lớn thì *F1* sẽ lớn và ngược lại, nếu *Precision* và *Recall* cùng nhỏ thì *F1* sẽ nhỏ. Độ đo *F1* được tính theo công thức sau:

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

4.2. Kết quả đánh giá

Data original

Kết quả đánh giá

	precision	recall	f1-score	support
FreshApple	0.90	0.95	0.93	80
FreshBanana	0.98	0.99	0.98	80
FreshGrape	0.88	0.96	0.92	80
FreshGuava	1.00	0.99	0.99	80
FreshJujube	0.91	0.93	0.92	80
FreshOrange	0.91	0.93	0.92	80
FreshPomegranate	0.92	0.97	0.95	80
FreshStrawberry	1.00	1.00	1.00	80
RottenApple	0.88	0.89	0.88	80
RottenBanana	0.99	0.95	0.97	80
RottenGrape	0.97	0.82	0.89	80
RottenGuava	0.87	0.94	0.90	80
RottenJujube	0.88	0.86	0.87	80
RottenOrange	0.91	0.90	0.91	80
RottenPomegranate	0.91	0.80	0.85	80
RottenStrawberry	0.95	0.97	0.96	80
accuracy			0.93	1280
macro avg	0.93	0.93	0.93	1280
weighted avg	0.93	0.93	0.93	1280

Nhìn vào bảng trên thì kết quả với tập kiểm thử cho ra accuracy là 0.93 thì chúng ta có thể thấy mô hình phân loại có độ chính xác cao. Tuy nhiên khi nhìn vào từng loại nhãn của tập kiểm thử thì chúng ta có thể thấy nhãn “RottenPomegranate” bị phân loại sai nhiều nhất, còn độ chính xác của “FreshStrawberry” là cao nhất.

Một số ảnh phân loại đúng của mô hình

FreshApple

True: FreshApple



Predicted: FreshApple

True: FreshApple



Predicted: FreshApple

True: FreshApple



Predicted: FreshApple

True: FreshApple



Predicted: FreshApple

True: FreshApple



Predicted: FreshApple

FreshBanana

True: FreshBanana



Predicted: FreshBanana

True: FreshBanana



Predicted: FreshBanana

True: FreshBanana



Predicted: FreshBanana

True: FreshBanana



Predicted: FreshBanana

True: FreshBanana



Predicted: FreshBanana

Một số ảnh bị phân loại sai của mô hình

FreshApple

True: FreshApple



Predicted: RottenPomegranate

True: FreshApple



Predicted: RottenApple

True: FreshApple



Predicted: RottenOrange

True: FreshApple



Predicted: RottenGuava

FreshBanana

True: FreshBanana



Predicted: RottenBanana

True: FreshGrape



Predicted: RottenGrape

True: FreshGrape

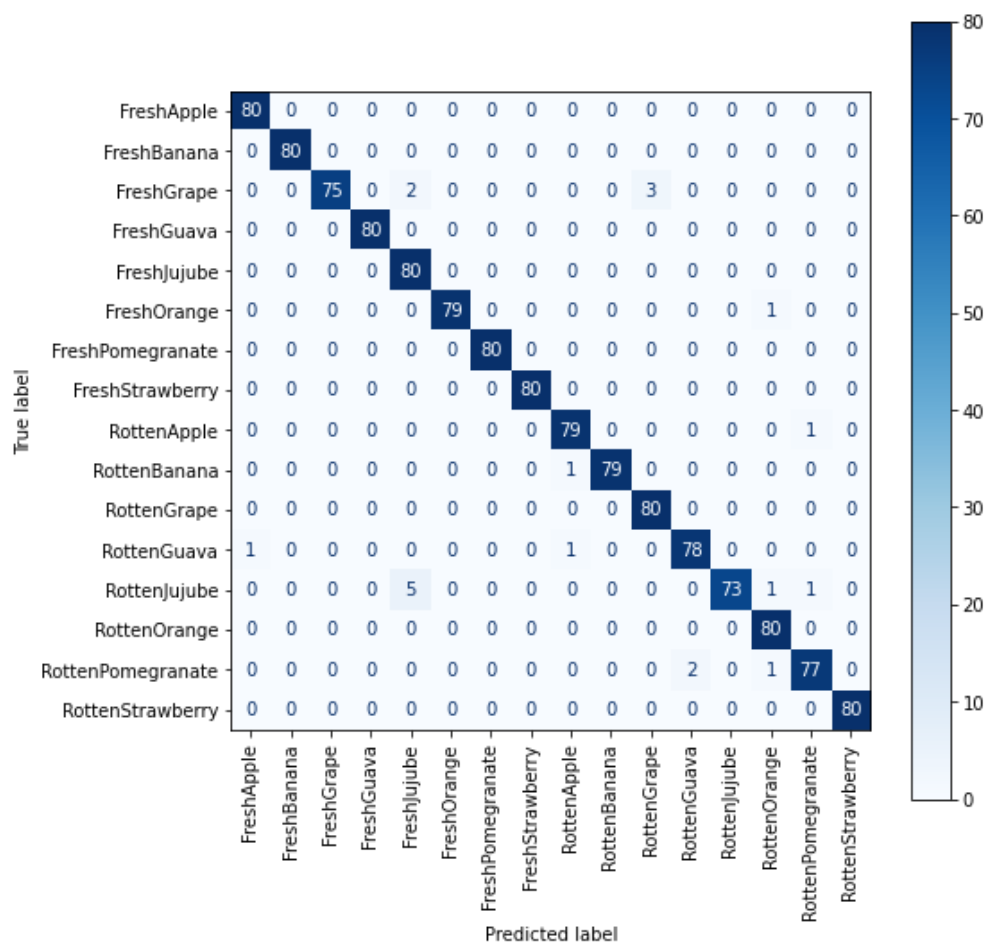


Predicted: RottenGrape

Data preprocess

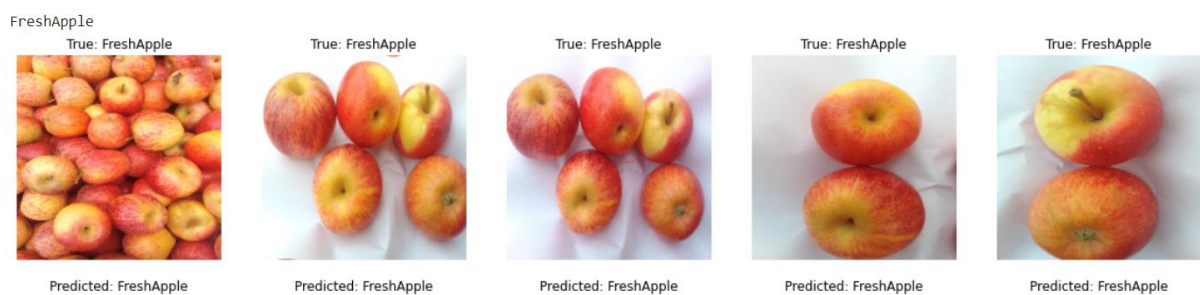
	precision	recall	f1-score	support
FreshApple	0.99	1.00	0.99	80
FreshBanana	1.00	1.00	1.00	80
FreshGrape	1.00	0.94	0.97	80
FreshGuava	1.00	1.00	1.00	80
FreshJujube	0.92	1.00	0.96	80
FreshOrange	1.00	0.99	0.99	80
FreshPomegranate	1.00	1.00	1.00	80
FreshStrawberry	1.00	1.00	1.00	80
RottenApple	0.98	0.99	0.98	80
RottenBanana	1.00	0.99	0.99	80
RottenGrape	0.96	1.00	0.98	80
RottenGuava	0.97	0.97	0.97	80
RottenJujube	1.00	0.91	0.95	80
RottenOrange	0.96	1.00	0.98	80
RottenPomegranate	0.97	0.96	0.97	80
RottenStrawberry	1.00	1.00	1.00	80
accuracy			0.98	1280
macro avg	0.98	0.98	0.98	1280
weighted avg	0.98	0.98	0.98	1280

Nhìn vào kết quả đánh giá tổng quát, chúng ta có thể thấy mô hình sau khi huấn luyện bằng data preprocess đã cho ra kết quả tốt hơn nhiều so với mô hình huấn luyện với data original, điều này là dữ liệu nhiều đã nhiều hơn và mô hình đã học tập tốt hơn để nhận biết những dữ liệu nhiều đó. Hầu như với mọi loại nhãn, mô hình phân loại gần như đúng hết, có vài nhãn thì tỉ lệ phân loại đúng là 100% và không phân loại nhầm sang loại nhãn khác.



Nhìn vào confusion matrix ta có thể thấy hầu như mọi nhãn đều phân loại đúng, chỉ có nhãn “RottenJujube” có 5 ảnh bị phân loại sai trạng thái thành “Fresh”.

Một số ảnh phân loại đúng của mô hình



Một số ảnh bị phân loại sai của mô hình

FreshGrape

True: FreshGrape



Predicted: FreshJujube

True: FreshGrape



Predicted: FreshJujube

True: FreshGrape



Predicted: RottenGrape

True: FreshGrape



Predicted: RottenGrape

True: FreshGrape



Predicted: RottenGrape

FreshOrange

True: FreshOrange



Predicted: RottenOrange

RottenBanana

True: RottenBanana



Predicted: RottenApple

KẾT LUẬN

Đối với bài toán phân loại trái cây tươi và trái cây hư thì mô hình sử dụng VGG16, một mô hình mạng có cấu trúc mạng CNN vô cùng phổ biến và hiệu quả trong các bài toán phân loại vì VGG16 đã huấn luyện sẵn trên một bộ dữ liệu rất lớn của ImageNet. Mô hình sau khi huấn luyện và kiểm thử trên bộ dataset *Fresh and Rotten Fruits Dataset for Machine-Based Evaluation of Fruit Quality* thì với data gốc mô hình cho kết quả kiểm thử khá tốt tuy nhiên vẫn còn một số nhãn bị phân loại sai nhiều. Với data preprocess thì mô hình đã làm quen được với nhiều của dataset, giúp cho việc phân loại chính xác hơn. Tuy nhiên vì dataset có số lượng ảnh còn hạn chế và số lượng nhãn còn thấp nên chưa đủ đưa ra thực tế để ứng dụng.

TÀI LIỆU THAM KHẢO

[1] Dataset

[Fresh and Rotten Fruits Dataset for Machine-Based Evaluation of Fruit Quality - Mendeley Data](#)

[2] Fruits fresh and rotten for classification

[Fruits fresh and rotten for classification | Kaggle](#)

[3] Classification-of-Fruit-images-using-VGG16

[vneogi199/Classification-of-Fruit-images-using-VGG16: Classifies images on fruits using the VGG16 model \(github.com\)](#)

[4] Bangkit-JKT2-D / fruits-fresh-rotten-classification

[fruits-fresh-rotten-classification/fresh_rotten_fruit_baseline.ipynb at master · Bangkit-JKT2-D/fruits-fresh-rotten-classification \(github.com\)](#)

[5] VGG16

[\[1409.1556\] Very Deep Convolutional Networks for Large-Scale Image Recognition \(arxiv.org\)](#)

[Everything you need to know about VGG16 | by Great Learning | Medium](#)

[Khoa học dữ liệu \(phamdinhhkhanh.github.io\)](#)

[Khoa học dữ liệu \(phamdinhhkhanh.github.io\)](#)