**NHẬN DIỆN TRÁI CÂY**

Vũ Đình Phúc   
Khoa Công Nghệ Thông Tin  
Trường Đại Học Đà LạtĐà Lạt, Lâm Đồng  
1812824@dlu.edu.vnNguyễn Hồng Phượng  
Khoa Công Nghệ Thông Tin  
Trường Đại Học Đà LạtĐà Lạt, Lâm Đồng  
1812829@dlu.edu.vn

-------------------  
Đặng Ngọc Thạch  
Khoa Công Nghệ Thông Tin  
Trường Đại Học Đà LạtĐà Lạt, Lâm Đồng  
1812843@dlu.edu.vn

*Tóm tắt:* Trong bài báo cáo này, chúng tôi giới thiệu về một bộ dữ liệu hình ảnh trái cây và một số phương pháp nhận diện trái cây. Ngoài ra, chúng tôi cũng trình bày kết quả của quá trình sử dụng phương pháp mạng nơ-ron trong nhận diện trái cây.

Từ khoá: Nhận diện trái cây, CNN, VGG16, tensorflow, deep learning, machine learning, object detection, image processing.

# **GIỚI THIỆU**

Tài liệu [1] nhận định: Bài toán nhận diện trái cây là một tường hợp cụ thể của bài toán nhận dạng và phân lớp hình ảnh. Bài toán này kế thừa khó khăn vốn có của bài toán gốc, kèm theo đó là các khó khăn riêng của chính nó như: số lượng khổng lồ về chủng loại theo mùa, vùng miền, địa hình,… với vô vàn loại trái cây có kiểu dáng, màu sắc, kết cấu giống nhau, dải màu thay đổi theo chu kỳ phát triển đa dạng,….

Bài toán nhận diện trái cây đã xuất hiện từ lâu và đã có rất nhiều bài báo, công trình khoa học được đưa ra nhằm đề xuất hoặc cải tiến thuật toán nhận dạng. Trong đó xuất hiện sớm nhất là các phương pháp “Xử lý ảnh – Image Processing”, các phương pháp này tập trung vào phát triển các thuật toán nhằm trích xuất thông tin, ví dụ như tham số về màu sắc, hình dạng, kết cấu, kích thước,… từ bức ảnh đầu để nhận dạng. Do chỉ đơn thuần xử lý trên một vài ảnh đầu trong khi sự biến thiên về màu sắc hình dạng, kích thước,… của trái cây quá phức tạp nên kết quả của phương pháp này không được cao và phạm vi áp dụng trên số lượng loại trái cây cũng bị hạn chế. Trong tài liệu [2], tác giả Patel và các cộng sự đã sử dụng thuật toán Algorithm để nhận diện trái cây. Trong bài báo này, các tác giả coi màu sắc, cường độ, cạnh và hướng là các đặc điểm của hình ảnh. Tuy nhiên, rất có thể có một số các tính năng khác như các tính năng đối xứng cũng phải xem xét. Thuật toán thông thường chỉ trả về giá trị chính xác khoảng 80%, tuy nhiên với sự cải tiến nhiều tính năng, các tác giả vẩn thu được kết quả có độ chính xác lên tới 95%.

Bên cạnh đó, sau khi xuất hiện nhiều bài báo khoa học đề xuất áp dụng phương pháp Machine learning vào bài toán nhận diện trái cây với độ chính xác cao, hướng giải quyết đã tập trung vào ứng dụng vào cải tiến thuật toán, cụ thể là nghiên cứu, thử nghiệm trích chọn các đặc trưng phù hợp nhất để đưa vào bộ huấn luyện nhận dạng tự động. Tài liệu [3], tác giả Yamamoto vả các cộng sự thực hiện nhận diện cà chua bằng phương pháp Image analysis và Blob-Based Segmentation. Trong nghiên cứu này, các tác giả mục đích phát triển phương pháp để phát hiện chính xác từng quả cà chua nguyên vẹn bao gồm cả quả trưởng thành và quả non trên cây sử dụng một máy ảnh kỹ thuật số RGB thông thường kết hợp với phương pháp Blob-Based Segmentation. Các phương pháp này phát triển không yêu cầu điều chỉnh các giá trị ngưỡng để phát hiện trái cây từ mỗi hình ảnh vì việc phân đoạn hình ảnh được thực hiện dựa trên các mô hình phân loại được tạo ra phù hợp với màu sắc, hình dạng, kết cấu và kích thước của hình ảnh. Kết quả từ phát hiện cà chua trong các hình ảnh thử nghiệm cho thấy phương pháp được phát triển đạt được độ thu hồi là 80%, trong khi độ chính xác là 88%. Ngoài ra, kết quả phát hiện quả non là 78%, mặc dù việc phát hiện quả non là rất khó vì kích thước nhỏ và bề ngoài của chúng giống với thân cây.

Trong những năm gần đây, với sự phát triển vượt bậc về công nghệ, máy tính và sự bùng nổ dữ liệu trên Internet, Deep learning đã đạt được nhiều thành tựu trong lĩnh vực Xử lý hình ảnh và Ngôn ngữ tự nhiên. Deep learning cũng được áp dụng rất thành công trong nhận diện trái cây, trong phạm vi hạn chế về số lượng trái cây cần nhận dạng, phương pháp này đã đạt được kết quả rất cao.

Bài báo cáo [4], tác giả I. Sa và các cộng sự trình bày một cách tiếp cận mới để phát hiện trái cây bằng cách sử dụng mạng nơ-ron phức hợp sâu (CNN). Phương pháp Faster–RCNN (dựa trên CNN) được các tác giả điều chỉnh, thông qua transfer learning, tạo nhiệm vụ phát hiện trái cây bằng cách sử dụng hình ảnh qua 2 phương thức ảnh RGB và ảnh hồng ngoại NIR. Phương pháp này đem lại độ chính xác luôn trên 83%. Mô hình được huấn luyện thực hiện phát hiện 7 loại trái cây, toàn bộ quy trình mất 4 giờ để chú thích và tạo mô hình mới cho mỗi loại trái cây. Tài liệu [5] cũng dùng phương pháp trên và đạt kết quả độ chính xác trên 85%.

Trong bài báo cáo [6], tác giả Yossy các cộng sự đã kết hợp CNN và Computer Vision để phân loại xoài chín và chưa chín. Computer vision mặc dù đã tồn tại từ lâu nhưng những hiệu quả đem lại cho bài toán nhận diện không thể phủ nhận. Hệ thống của các tác giả tận dụng mạng nơ-ron và phương pháp Computer vision để phân loại xoài phù hợp với độ chính xác cao đạt tới 94%.

Ngoài ra, trong bài báo cáo [7], tác giả Zhang và các cộng sự đã đưa ra một hướng khác để giải quyết bài toán với phương pháp You Only Lock One (YOLO). Fast YOLO là công cụ phát hiện vật thể đa năng nhanh nhất. Trong khi đó, YOLOv2 cung cấp sự cân bằng tốt nhất giữa tốc độ thời gian thực và độ chính xác tuyệt vời để phát hiện đối tượng so với các hệ thống phát hiện khác trên nhiều bộ dữ liệu phát hiện. Với một hình ảnh thông thường, kết quả đạt được có thể lên đến 85%. Với một video 155FPS, độ chính xác vẫn đạt được lên đến 78.6%.

Bài báo cáo tập trung về nghiên cứu mô hình mạng nơ-ron (CNN) và đánh giá mô hình đã được xây dựng trước đó trong lĩnh vực nhận diện trái cây.

# **CƠ SỞ LÝ THUYẾT**

## **Deep learning**

3 Học kĩ càng

Trong lĩnh vực nhận dạng hình ảnh và phân loại, cách tái tạo thành công nhất thu được là bằng cách sử dụng mạng nơron nhân tạo [4]. Các mạng này tạo cơ sở cho hầu hết các mô hình học sâu.

Học sâu là một lớp thuật toán máy học sử dụng đa lớp có chứa các đơn vị xử lý phi tuyến tính. Mỗi cấp độ học để chuyển đổi dữ liệu đầu vào của nó thành một đại diện tổng hợp và trừu tượng hơn. Các phương pháp học sâu đã quản lý để hoạt động tốt hơn các mạng khác các thuật toán học máy. Họ cũng đạt được danh hiệu đầu tiên - nhận dạng chim nhạn trong một số lĩnh vực nhất định [4]. Điều này càng củng cố bởi thực tế rằng học sâu được coi là một bước quan trọng để xây dựng AI mạnh mẽ. Thứ hai, mạng nơ-ron sâu đã được chứng minh là thu được kết quả tuyệt vời trong lĩnh vực hình ảnh.

Trong phần còn lại của phần này, chúng tôi sẽ mô tả tóm tắt mô hình của mạng nơ-ron nhân tạo cùng với một số kết quả liên quan.

## **Mạng nơ-ron tích chập (CNN)**

Mạng nơ-ron tích chập (CNN) là một phần của mô hình học sâu. Một mạng như vậy có thể bao gồm các lớp phức hợp, các lớp gồm: Convolutional layer, Rectiﬁed Linear Unit (ReLU) layer, Pooling layer và Connected layer.

Trong một CNN điển hình, kiến trúc mỗi lớp phức hợp được cấu tạo bởi một ReLU layer, sau đó là một lớp Pooling layer rồi một hoặc nhiều lớp Convolutional layer và

cuối cùng là một hoặc nhiều lớp Connected layer.

### **Layer**

#### Convolutional layer

Convolutional layer được đặt tên theo thao tác tích chập. Trong mathe-matics, convolution là một tiến trình sử dụng 2 hàng sẵn có tạo nên chức năng mới thứ 3, đầy là một phiên bản phức tạp hơn của một trong các chức năng ban đầu.

Convolutional layer bao gồm các nhóm nơ-ron tạo nên kernel. Kernel có kích thước nhỏ nhưng có độ sâu tương tự như input. Các nơ-ron từ kernel được kết nối với một khu vực nhỏ của input được gọi là trường tiếp cận, bởi vì nó không hiệu quả nếu liên kết tất cả nơ-ron với tất cả output trong trường hợp đầu vào có kích thước cao như hình ảnh. Ví dụ, một hình ảnh 100x100 có 10000pixel vả nếu lớp đầu tiên có 100 nơ-ron thì kết quả trả về là 1000000 parameter.

Trong convolational layer, các kernel trượt qua chiều rộng và chiều dài của input, trích xuất các đặc trưng cấp cao và tạo ra bản đồ kích hoạt 2 chiều. Stride là tại đó, một trong chiếu của kernel được đưa ra tương đương với 1 parameter.

Đầu ra của một convolutional layer được tạo ra bằng cách xếp chồng các bản đồ kích hoạt được tạo ra mà lần này được sử dụng để xác định đầu vào của lớp tiếp theo.

Các stride khiến kernel bỏ qua các điểm ảnh trong hình ảnh và không bao gồm chúng trong output. Những bước tiến xác định các thao tác tích chập hoạt động với kernel khi hình ảnh lớn hơn và kernel phức tạp hơn được sử dụng. Khi một kernel đang đưa input vào, nó sử dụng tham số stride để xác định có bao nhiêu vị trí để bỏ qua.

#### ReLU layer

ReLU layer, hay lớp đơn vị tuyến tính được chỉnh sửa, áp dụng các hàm max activation function (0, x). Nó không làm giảm kích thước của mạng, nhưng làm tăng tính chất của nó.

#### Pooling layer

Các lớp pooling layer được sử dụng một mặt để giảm kích thước không gian biểu diễn vả để giảm lượng tính toán được thực hiện trong mạng. Ngoài ra, nó còn có chức năng khác là kiểm soát overfitting.

Pooling layer được sử dụng nhiều nhất với bộ lọc có kích thước 2x2 cùng một stride. Điều này làm cho input giảm so với kích thước ban đầu của nó.

#### Fully connected layer

Fully connected layer là các lớp từ mạng nơ-ron thường xuyên. Mỗi nơ-ron từ một tầng đầy đủ được liên kết với mỗi output của lớp trước đó. Các hoạt động của lớp này cũng giông như hoạt động đằng sau của convolutional layer. Do đó có thể chuyển đổi giữa 2 lớp này.

#### Loss layer

Loss layer dùng để xử lý mạng cho các hệ thống deviating từ output cũ. Đây thường là tầng cuối cùng của mạng. Các hàm trong lớp này gồm: sodtmax được dùng để dự đoán một lớp từ nhiều lớp disjunct, sigmoid cross-entropy được dùng để dự đoán nhiều giá trị xác suất độc lập.

### **Activation function**

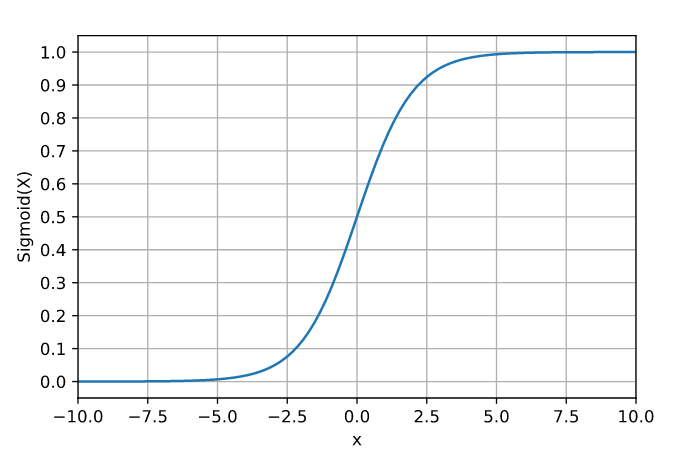
Hàm kích hoạt (activation function) mô phỏng tỷ lệ truyền xung qua axon của một nơ-ron nhân tạo, hành kích hoạt đóng vai trò là thành phần phi tuyến tại output của các nơ-ron. Trong bài viết này, chúng ta sẽ cùng tìm hiểu các hàm kích hoạt phổ biết nhất và các ưu, nhược điểm của chúng.

Nếu không có các hàm kích hoạt phi tuyến thì mạng nơ-ron dù có nhiều lớp vẫn sẽ có hiệu quả như một lớp tuyến tính.

#### Sigmoid

Công thức:

Phân tích:

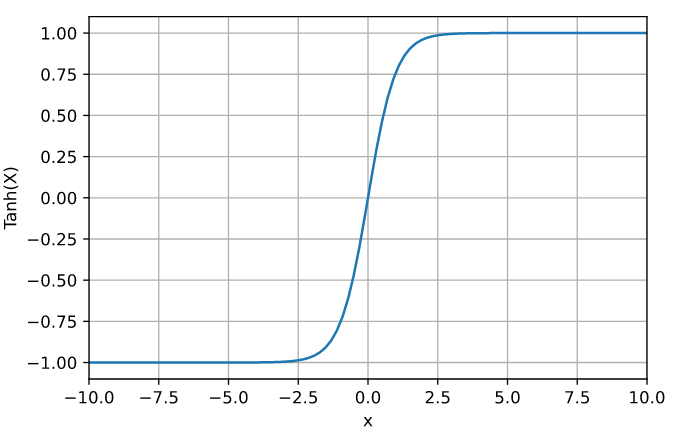


Hàm Sigmoid nhận đầu vào là một số thực và chuyển thành một giá trị trong khoảng (0;1) (xem đồ thị phía trên). Đầu vào là số thực âm rất nhỏ sẽ cho đầu ra tiệm cận với 0, ngược lại, nếu đầu vào là một số thực dương lớn sẽ cho đầu ra là một số tiệm cận với 1. Trong quá khứ hàm Sigmoid hay được dùng vì có đạo hàm rất đẹp. Tuy nhiên hiện nay hàm Sigmoid rất ít được dùng vì những nhược điểm sau: Hàm Sigmoid bão hòa và triệt tiêu gradient; Hàm Sigmoid không có trung tâm là 0 gây khó khăn cho việc hội tụ.

#### Tanh

Công thức:

Phân tích:



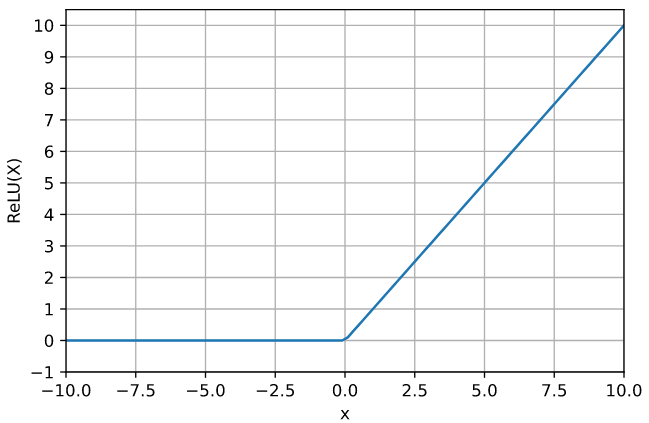
Hàm Tanh nhận đầu vào là một số thực và chuyển thành một giá trị trong khoảng (-1;1). Cũng như Sigmoid, hàm Tanh bị bão hòa ở 2 đầu (gradient thay đổi rất ít ở 2 đầu). Tuy nhiên hàm Tanh lại đối xứng qua 0 nên khắc phục được một nhược điểm của Sigmoid.

Hàm Tanh còn có thể được biểu diễn bằng hàm Sigmoid như sau:

#### ReLU

Công thức:

Phân tích:



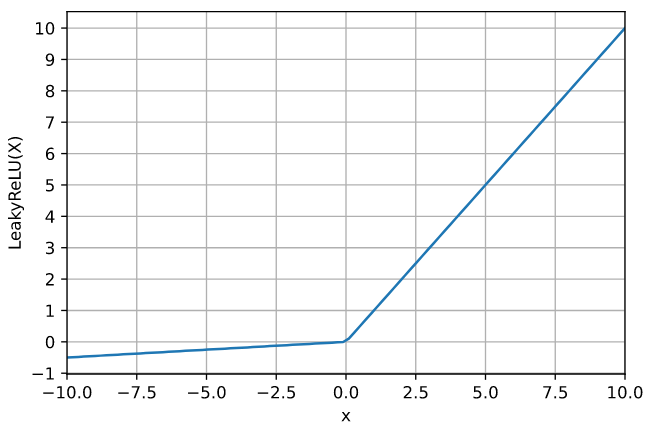
Hàm ReLU đang được sử dụng khá nhiều trong những năm gần đây khi huấn luyện các mạng nơ-ron. ReLU đơn giản lọc các giá trị < 0. Nhìn vào công thức chúng ta dễ dàng hiểu được cách hoạt động của nó. Một số ưu điểm khá vượt trội của nó so với Sigmoid và Tanh:

* Tốc độ hội tụ nhanh hơn hẳn. ReLU có tốc độ hội tụ nhanh gấp 6 lần Tanh. Điều này có thể do ReLU không bị bão hòa ở 2 đầu như Sigmoid và Tanh.
* Tính toán nhanh hơn. Tanh và Sigmoid sử dụng hàm exp và công thức phức tạp hơn ReLU rất nhiều do vậy sẽ tốn nhiều chi phí hơn để tính toán.
* Tuy nhiên, ReLU cũng có một nhược điểm: Với các node có giá trị nhỏ hơn 0, qua ReLU activation sẽ thành 0, hiện tượng đấy gọi là “Dying ReLU”. Nếu các node bị chuyển thành 0 thì sẽ không có ý nghĩa với bước linear activation ở lớp tiếp theo và các hệ số tương ứng từ node đấy cũng không được cập nhật với gradient descent. 🡺 Leaky ReLU ra đời.
* Khi learning rate lớn, các trọng số (weights) có thể thay đổi theo các làm tất cả các nơ-ron dừng việc cập nhật.

#### Leaky ReLU

Công thức:

Phân tích:



Leaky ReLU là một cố gắng trong việc loại bỏ “dying ReLU”. Thay vì trả về giá trị 0 với các đầu vào < 0 thì Leaky ReLU tạo ra một đường xiên có độ dốc nhỏ. Có nhiều báo cáo về việc Leaky ReLU có hiệu quả tốt hơn ReLU, nhưng hiệu quả này vẫn chưa rõ ràng và nhất quán.

#### Maxout

Chúng ta sẽ không sử dụng công thức dạng nữa. Một dạng khá phổ biến là Maxout nơ-ron được tính bằng công thức: . Leaky ReLU và ReLU là các dạng đặc biệt của công thức này (với ReLU, áp dụng công thức với ). Maxout có tất cả các ưu điểm của Leaky ReLU, ReLU. Tuy vậy nó khiến mạng phải sử dụng gấp đôi tham số (parameter) cho mỗi nơ-ron, vì thế làm tăng đáng kể chi phí cả về bộ nhớ và tính toán – một điều cần suy xét khi huấn luyện mạng deep learning ở hiện tại.

## **Visual Geometry Group** (**VGG)**

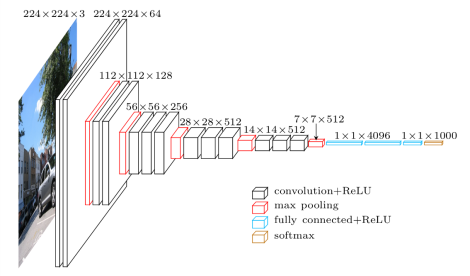
VGG là viết tắt của Visual Geometry Group; nó là một kiến trúc mạng nơ rơn tích chập (CNN) với nhiều lớp. "Deep đề cập đến số lượng các lớp với VGG16 bao gồm 16 lớp phức hợp hoặc VGG-19 bao gồm 19 lớp phức hợp.

VGG16 là mạng CNN được đề xuất bởi K.Simonyan và A.Zisserman của đại học Oxford. Các Model sau khi được tạo bởi mạng VGG16 sẽ cho độ chính xác lên đến 92.7% top-5 test trong dữ liệu ảnh mạng gồm 14 triệu hình ảnh khác nhau thuộc 1000 lớp.

Cấu trúc của VGG16 bao gồm:

* 13 lớp tích chập (Conv2 hoặc Conv3) đều có Kernel 3x3.
* Sau các layer Conv là Maxpooling giảm kích thước xuống một nửa và ba lớp Fully-Connected.

Hình 2: Kiến trúc mạng thần kinh VGG16:[8]



* Input: VGGNet có kích thước đầu vào hình ảnh là 224 × 224. Đối với cuộc thi ImageNet, những người tạo ra mô hình đã cắt bỏ mảng trung tâm 224 × 224 trong mỗi hình ảnh để giữ cho kích thước đầu vào của hình ảnh nhất quán.
* Convolutional layer: Các lớp tích tụ của VGG tận dụng một trường tiếp nhận tối thiểu, tức là 3 × 3, kích thước nhỏ nhất có thể mà vẫn chụp lên / xuống và trái / phải. Hơn nữa, cũng có các bộ lọc tích chập 1 × 1 hoạt động như một phép biến đổi tuyến tính của đầu vào. Tiếp theo là đơn vị ReLU, đây là một sự đổi mới rất lớn từ AlexNet giúp giảm thời gian huấn luyện. ReLU là viết tắt của chức năng kích hoạt đơn vị tuyến tính được chỉnh lưu; nó là một hàm tuyến tính từng mảnh sẽ xuất ra đầu vào nếu dương; nếu không, đầu ra bằng không. Sải tích chập được cố định ở 1pixel để giữ nguyên độ phân giải không gian sau khi tích chập (sải chân là số pixel dịch chuyển trên ma trận đầu vào).
* Hidden layer: Tất cả các lớp ẩn trong mạng VGG đều sử dụng ReLU. VGG thường không tận dụng Chuẩn hóa phản hồi cục bộ (LRN) vì nó làm tăng mức tiêu thụ bộ nhớ và thời gian huấn luyện. Hơn nữa, nó không cải thiện độ chính xác tổng thể.
* Fully-connected layer: VGGNet có ba lớp được kết nối đầy đủ. Trong số ba lớp, hai lớp đầu tiên có 4096 kênh mỗi lớp và lớp thứ ba có 1000 kênh, mỗi lớp 1 kênh.

## **Thư viện Tensorflow**

Để thực hiện mục đích huấn luyện và thử nghiệm mô hình CNN, chùng tôi sử dụng thư viện tensorflow. Đây là một thư viện mã nguồn mở cho machine learning được google cho phép tính toán bằng biểu đồ data flow. Nút trong biểu đồ biểu thị cho các thao tác mathe – matical, trong khi đó các cạnh biểu đồ biểu thị các mảng dữ liệu đa chiều được gọi là tensor.

Các thành phần chính trong tensorflow là client và một hoặc nhiều tiến trình xử lý. Mỗi quy trình chịu trách nhiệm cho một hay nhiều thiết bị tính toán (ví dụ như nhân CPU hay card GPU) và thực hiện các nút đồ thị trên các thiết bị đố theo hướng dẫn của người thực hiện.

Tensorflow có một số tính năng mạnh như: nó cho phép ánh xạ tính toán trên nhiều thiết bị, không giống như hầu hết các framework khác, nó được xây dựng để hỗ trợ tính toán gradient tự động, nó có thể thực hiện một phần subgraph của toàn bộ đồ thị và có thể thêm ràng buộc với các thiết bị, ….

Từ phiên bản 2.0, tensorflow đã bao gồm các tính năng của keras framework. Keras cung cấp wrapper trong các hoạt động được thực hiện ở tensorflow, đơn giản hoá quá trình gọi hàm và làm giảm tổng thể số code cần thiết để huấn luyện và kiểm tra mô hình.

Chúng tôi sẽ trình bày các phương pháp sử dụng quan trọng nhất và các loại dữ liệu có thể kết hợp với nhau với một mô tả ngắn gọn.

Lớp Convolutional layer:

Conv2D (

no \_filters,

fi lter\_ size,

strides,

padding,

name=None

)

Tính toán một convolution 2D với input [batch, in\_height, in\_width, in\_channels] và một kernel tensor [filter\_height, filter\_width]. Nhiệm vụ này thực hiện:

* Xử lý ma trận 2D [in\_height\*in\_width\*in\_channels, output\_channels]
* Chiết xuất hình ảnh từ input để tạo thành một virtual tensor [batch, out\_height, out\_width, filter\_height\*filter\_width\*in\_channels]
* Với mỗi patch, nhân ma trận filter và patch vector của hình ảnh.
* Nếu padding được set thành “same”, input là 0-padded thì output giữ nguyên height và width; ngược lại, nếu padding được set thành “valid”, input không phải 0-padded, thì output có thể có height và width nhỏ hơn.

MaxPooling2D (

filter\_size,

strides,

padding,

name=NONE,

)

Max pooling thực hiện hoạt động trên input. filter\_size đại diện cho kích thước của cửa sổ trong đó hàm max được áp dụng. strides đại diện cho số bước của của sổ trượt cho mỗi kích thước của input tensor. Tương tự Conv2D, padding parameter có thể lả “valid” hoặc “same”.

Activation (

operation,

name=NONE,

)

Hàm activation được tính toán đặc biệt bằng operation.

Drop (

prop,

name=NONE,

)

Đặt ngẫu nhiên các giá trị input thành 0 với xác xuất prop.

# **MÔ HÌNH ĐỀ XUẤT**

## **Bộ dữ liệu trái cây**

Bài báo cáo sử dụng bộ dữ liệu hình ảnh trái cây kích thước 100x100 gồm 131 loại trái cây khác nhau, mỗi hình ảnh chỉ chứa 1 loại trái cây/ hình bao gốm các góc cạnh khác nhau của trái cây, trong đó có 67692 hình ảnh dùng để huấn luyện mô hình, 22688 hình ảnh dùng để thử nghiệm mô hình (Bảng 1).

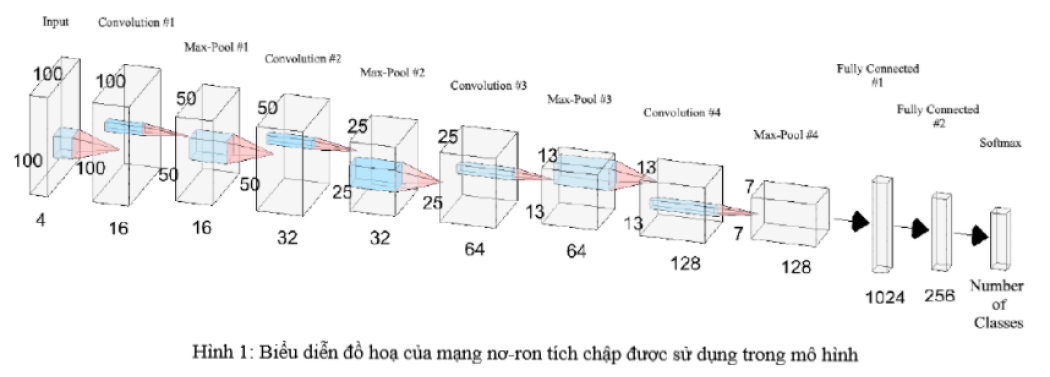
Bảng 1: Số hình ảnh trên mỗi loại trái cây (Một số loại trái cây khác nhau nhưng không thể phân biệt tên được đánh số thứ tự. Ví dụ: Apple Red 1, Apple Red 2, …)

|  |  |  |
| --- | --- | --- |
| Label | Training Images | Test Images |
| Apple Braeburn | 492 | 164 |
| Apple Crimson Snow | 444 | 148 |
| Apple Golden 1 | 480 | 160 |
| Apple Golden 2 | 492 | 164 |
| Apple Golden 3 | 481 | 161 |
| Apple Granny Smith | 492 | 164 |
| Apple Pink Lady | 456 | 152 |
| Apple Red 1 | 492 | 164 |
| Apple Red 2 | 492 | 164 |
| Apple Red 3 | 429 | 144 |
| Apple Red Delicious | 490 | 166 |
| Apple Red Yellow 1 | 492 | 164 |
| Apple Red Yellow 2 | 672 | 219 |
| Apricot | 492 | 164 |
| Avocado | 427 | 143 |
| Avocado ripe | 491 | 166 |
| Banana | 490 | 166 |
| Banana Lady Finger | 450 | 152 |
| Banana Red | 490 | 166 |
| Beetroot | 450 | 150 |
| Blueberry | 462 | 154 |
| Cactus fruit | 490 | 166 |
| Cantaloupe 1 | 492 | 164 |
| Cantaloupe 2 | 492 | 164 |
| Carambula | 490 | 166 |
| Cauliï¬‚ower | 702 | 234 |
| Cherry 1 | 492 | 164 |
| Cherry 2 | 738 | 246 |
| Cherry Rainier | 738 | 246 |
| Cherry Wax Black | 492 | 164 |
| Cherry Wax Red | 492 | 164 |
| Cherry Wax Yellow | 492 | 164 |
| Chestnut | 450 | 153 |
| Clementine | 490 | 166 |
| Cocos | 490 | 166 |
| Corn | 450 | 150 |
| Corn Husk | 462 | 154 |
| Cucumber Ripe | 392 | 130 |
| Cucumber Ripe 2 | 468 | 156 |
| Dates | 490 | 166 |
| Eggplant | 468 | 156 |
| Fig | 702 | 234 |
| Ginger Root | 297 | 99 |
| Granadilla | 490 | 166 |
| Grape Blue | 984 | 328 |
| Grape Pink | 492 | 164 |
| Grape White | 490 | 166 |
| Grape White 2 | 490 | 166 |
| Grape White 3 | 492 | 164 |
| Grape White 4 | 471 | 158 |
| Grapefruit Pink | 490 | 166 |
| Grapefruit White | 492 | 164 |
| Guava | 490 | 166 |
| Hazelnut | 464 | 157 |
| Huckleberry | 490 | 166 |
| Kaki | 490 | 166 |
| Kiwi | 466 | 156 |
| Kohlrabi | 471 | 157 |
| Kumquats | 490 | 166 |
| Lemon | 492 | 164 |
| Lemon Meyer | 490 | 166 |
| Limes | 490 | 166 |
| Lychee | 490 | 166 |
| Mandarine | 490 | 166 |
| Mango | 490 | 166 |
| Mango Red | 426 | 142 |
| Mangostan | 300 | 102 |
| Maracuja | 490 | 166 |
| Melon Piel de Sapo | 738 | 246 |
| Mulberry | 492 | 164 |
| Nectarine | 492 | 164 |
| Nectarine Flat | 480 | 160 |
| Nut Forest | 654 | 218 |
| Nut Pecan | 534 | 178 |
| Onion Red | 450 | 150 |
| Onion Red Peeled | 445 | 155 |
| Onion White | 438 | 146 |
| Orange | 479 | 160 |
| Papaya | 492 | 164 |
| Passion Fruit | 490 | 166 |
| Peach | 492 | 164 |
| Peach 2 | 738 | 246 |
| Peach Flat | 492 | 164 |
| Pear | 492 | 164 |
| Pear 2 | 696 | 232 |
| Pear Abate | 490 | 166 |
| Pear Forelle | 702 | 234 |
| Pear Kaiser | 300 | 102 |
| Pear Monster | 490 | 166 |
| Pear Red | 666 | 222 |
| Pear Stone | 711 | 237 |
| Pear Williams | 490 | 166 |
| Pepino | 490 | 166 |
| Pepper Green | 444 | 148 |
| Pepper Orange | 702 | 234 |
| Pepper Red | 666 | 222 |
| Pepper Yellow | 666 | 222 |
| Physalis | 492 | 164 |
| Physalis with Husk | 492 | 164 |
| Pineapple | 490 | 166 |
| Pineapple Mini | 493 | 163 |
| Pitahaya Red | 490 | 166 |
| Plum 447 151Plum 2 | 420 | 142 |
| Plum 3 | 900 | 304 |
| Pomegranate | 492 | 164 |
| Pomelo Sweetie | 450 | 153 |
| Potato Red | 450 | 150 |
| Potato Red Washed | 453 | 151 |
| Potato Sweet | 450 | 150 |
| Potato White | 450 | 150 |
| Quince | 490 | 166 |
| Rambutan | 492 | 164 |
| Raspberry | 490 | 166 |
| Redcurrant | 492 | 164 |
| Salak | 490 | 162 |
| Strawberry | 492 | 164 |
| Strawberry Wedge | 738 | 246 |
| Tamarillo | 490 | 166 |
| Tangelo | 490 | 166 |
| Tomato 1 | 738 | 246 |
| Tomato 2 | 672 | 225 |
| Tomato 3 | 738 | 246 |
| Tomato 4 | 479 | 160 |
| Tomato Cherry Red | 492 | 164 |
| Tomato Heart | 684 | 228 |
| Tomato Maroon | 367 | 127 |
| Tomato not Ripened | 474 | 158 |
| Tomato Yellow | 459 | 153 |
| Walnut | 735 | 249 |
| Watermelon | 475 | 157 |

## **Cấu trúc mạng nơ-ron được sử dụng trong mô hình**

Trong bài báo cáo này, chúng tôi sử dụng mô hình CNN kết hợp cùng thư viện Tensorflow. Như đã mô tả trước đây, mạng này sử dụng các lớp convolutional layer, pooling layer, ReLU layer, fully connected layer và loss layer. Trong một kiến trúc CNN điển hình, mỗi convolutional layer được theo sau bởi một ReLU layer, sau đó là một pooling layer tiếp tục cùng một hoặc nhiều convolutional layer và cuối cùng là một hoặc nhiều fully connected layer.

Lưu ý một lần nữa ràng đặc điểm làm tách CNN khỏi mạng nơ-ron thông thường đang tính đến cấu trúc của hình ảnh trong khi xử lý chúng. Một mạng nơ-ron thông thường chuyển đổi input trong mảng một chiều làm cho bộ huấn luyện ý nhạy cảm hơn với những thay đổi về vị trị.

 Input mà chúng tôi sử dụng bao gồm hình ảnh RGB tiêu chuẩn có kích thước 100x100 pixels.

Mạng nơ-ron mà chúng tôi sử dụng trong mô hình có cấu trúc như bảng 2.

Bàng 2: Cấu trúc mạng nơ-ron sử dụng trong mô hình

|  |  |  |
| --- | --- | --- |
| Lớp | Kích thước | Output |
| Convolutional | 5 x 5 x 4 | 16 |
| Max pooling | 2 x 2 - stride: 2 | - |
| Convolutional | 5 x 5 x 16 | 32 |
| Max pooling | 2 x 2 – stride: 2 | - |
| Convolutional | 5 x 5 x 32 | 64 |
| Max pooling | 2 x 2 - stride: 2 | - |
| Convolutional | 5 x 5 x 64 | 128 |
| Max pooling | 2 x 2 - stride: 2 | - |
| Fully connected | 5 x 5 x 128 | 1024 |
| Fully connected | 1024 | 256 |
| Softmax | 256 | 131 |

Một biểu diễn trực quan hơn của mạng nơ-ron được sử dụng được đưa ra trong hình 1:

* Lớp đầu tiên (Convolution #1) là một convolutional layer áp dụng 16 bộ lọc 5 x 5. Trong lớp này, chúng tôi áp dụng max pooling với bộ lọc 2 x 2 cùng stride = 2 để xác định rằng các vùng được gộp không bị chồng chéo nhau (Max pooling #1). Điều này cũng làm giảm height và width xuống 50 pixels mỗi hình.
* Lớp thứ 2 (Convolutional #2) áp dụng 32 bộ lọc 5 x 5 cùng 32 output activation map. Chúng tôi áp dụng trên lớp này cùng một loại max pooling (Max pooling #2) như ở lớp đầu tiên.
* Lớp thứ 3 (Convolutional #3) áp dụng 64 bộ lọc 5 x 5. Max pooling (Max pooling #3) cũng tương tự như trên với bộ lọc 2 x 2 cùng stride = 2.
* Lớp thứ 4 (Convolutional #4) áp dụng 128 bộ lọc 5 x 5 và cũng có max pooling (Max pooling #4) tương tự.
* Do 4 lớp max pooling, kích thước biểu diễn đã giảm theo hệ số 16, do đó, lớp thứ 5, là một lớp fully connected (Fully connected #1) có 16 input 7 x 7.
* Lớp này được nạp vào một lớp fully connected khác (Fully connected #2) với 1024 input và 256 output.
* Lớp cuối cùng là một lớp softmax loss layer (Softmax) với 256 input. Số lượng output bằng với số class.

Chúng tôi trình bày một đoạn ngắn của quá trình huấn luyện như sau:

epochs = 100,

read\_images(images),

apply\_random\_vertical\_horizontal\_flips(images )

apply\_random\_hue\_saturation\_changes(images)

convert\_to\_hsv(image s )

add\_grayscale\_layer(images )

define\_network\_structure(network)

for i in range(1, epochs ) :

train\_network(network)

test\_network(network )

## **Mô hình huấn luyện**

Chúng tôi sử dụng mô hình như phần trước đã trình bày để tiến hành huấn luyện mạng.

Sử dụng ImagesDataGenerator từ gói tensorflow.keras.preprocessing.image, chúng tôi chọn input ngẫu nhiên cho mạng. Đối tượng này quản lý tải hình ảnh, tạo hàng loạt và có thể thực hiện các chức năng tăng cường như xoay ảnh ngang và dọc.

# **ĐÁNH GIÁ THỬ NGHIỆM**

## **Môi trường thử nghiệm**

Quá trình thử nghiệm thực hiện trên hệ thống máy ảo thuộc kaggle với CPU 8 nhân, 16GB ram, TPU v3-10, HDD 80GB. Hệ thống này đã được cài đặt đầy đủ môi trường Python 3.7, các thư viện vào framework dành cho machine learning. Theo đó, chúng tôi thử nghiệm mô hình trên môi trường Python và sử dụng framework của tensorflow với giao diện thư viện keras, đây là thư viện cung cấp tính năng mạnh mẽ cho khả năng xử lý hình ảnh và cho mô hình CNN. Ngoài ra, chúng tôi sử dụng jupyter notebook để thực hiện debug chương trình.

Chúng tôi sử dụng mô hình đã nêu để huấn luyện và đánh giá lại quá trình huấn luyện. Quá trình đánh giá sẽ trải qua thử nghiệm mô hình trên bộ dữ liệu huấn luyện và bộ dữ liệu thử nghiệm.

## **Kết quả**

Sau thử nghiệm, chúng tôi thu được các kết quả của mô hình CNN đã huấn luyện trên bộ dữ liệu thử nghiệm có độ chính xác là 85.2% (19330/22688), mô hình VGG16 đã huấn luyện trên bộ dữ liệu thử nghiệm có độ chính xác là 95.78% (21731/22688). Sai lệch khi thử nghiệm trên bộ dữ liệu thử nghiệm có thể do màu sắc, kích thước, hình dạng của mội số loại trái cây tương đồng nhau dẫn đến nhận diện nhầm đối tượng.

# **KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

Chúng tôi đánh giá mô hình huấn luyện có độ chính xác rất cao với bộ dữ liệu đã được xử lý. Tuy nhiên, mô hình vẫn chưa được tối ưu cho nhận diện với hình ảnh chưa được xử lý hay nhận diện nhiều loại trái cây trong một hình ảnh.

Theo quan điểm của chúng tôi, một trong những mục tiêu chính trong tương lai là cải thiện mô hình mạng CNN (VGG16 là một mô hình CNN đã được nghiên cứu và cải tiến với độ chính xác rất cao). Điều này liên quan đến thử nghiệm sâu hơn với cấu trúc của mạng nơ-ron.

# **TÀI LIỆU THAM KHẢO**

[1] K. Y. Sè, “C l Fe C l C l,” pp. 65–78, 2018.

[2] H. N. Patel, R. K. Jain, A. D. Patel, and M. V Joshi, “Fruit Detection using Improved Multiple Features based Algorithm,” 2011.

[3] K. Yamamoto, W. Guo, Y. Yoshioka, and S. Ninomiya, “On plant detection of intact tomato fruits using image analysis and machine learning methods,” *Sensors (Switzerland)*, vol. 14, no. 7, pp. 12191–12206, Jul. 2014, doi: 10.3390/s140712191.

[4] I. Sa, Z. Ge, F. Dayoub, B. Upcroft, T. Perez, and C. McCool, “Deepfruits: A fruit detection system using deep neural networks,” *Sensors (Switzerland)*, vol. 16, no. 8, p. 1222, Aug. 2016, doi: 10.3390/s16081222.

[5] S. Bargoti and J. P. Underwood, “Image Segmentation for Fruit Detection and Yield Estimation in Apple Orchards,” *J. F. Robot.*, vol. 34, no. 6, pp. 1039–1060, Sep. 2017, doi: 10.1002/rob.21699.

[6] E. H. Yossy, J. Pranata, T. Wijaya, H. Hermawan, and W. Budiharto, “Mango Fruit Sortation System using Neural Network and Computer Vision,” in *Procedia Computer Science*, Jan. 2017, vol. 116, pp. 596–603, doi: 10.1016/j.procs.2017.10.013.

[7] Y. Zhang *et al.*, “Understanding of Object Detection Based on CNN Family and YOLO Related content Real-time vehicle detection and tracking in video based on faster R-CNN A Real-Time Method to Estimate Speed of Object Based on Object Detection and Optical Flow Calculation Un,” *J. Phys*, p. 12029, 2018, doi: 10.1088/1742-6596/1004/1/012029.

[8] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, Sep. 2014, doi: 10.48550/arxiv.1409.1556.