ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA TP. HỒ CHÍ MINH
KHOA ĐIỆN – ĐIỆN TỬ

TIỂU LUẬN MÔN HỌC

THỊ GIÁC MÁY TÍNH

Phân Loại Hoa Sử Dụng Mô hình Mạng Neuron

GVHD: TS. TRƯƠNG QUANG VINH

HV: Tiến Hoàng Trí Nghĩa - 1870048

TP.Hồ Chí Minh, tháng 06 năm 2019

LỜI CẢM ƠN

Để hoàn thành đề tài tiểu luận này, chúng em xin gửi lời cảm ơn chân thành tới Thầy TS. Trương Quang Vinh đã tận tình hướng dẫn, giảng dạy, cung cấp tài liệu để chúng em có thể hoàn thành đề tài tiểu luận này.

Chúng em xin chân thành cảm ơn quý Thầy Cô của Bộ môn Điện Tử nói riêng, Khoa Điện điện tử nói chung đã truyền đạt cho chúng em những kiến thức nền tảng thiết yếu để chúng em có thể hoàn thành tiểu luận thuận lợi.

Một lần nữa, chúng em xin chân thành cảm ơn.

Tp. Hồ Chí Minh, ngày 25 tháng 06 năm 2019

MỤC LỤC

CHƯƠNG 1 – GIỚI THIỆU TỔNG QUAN 1
1.1. TỔNG QUAN VỀ NHẬN DẠNG VÀ PHÂN LOẠI HÌNH ẢNH 1
1.2. CÁC HƯỚNG TIẾP CẬN VÀ GIẢI QUYẾT BÀI TOÁN 3
1.2.1. Phương pháp Machine Learning truyền thống 4
1.2.2. Phương pháp Deep Learning 6
CHƯƠNG 2 – CNN - Convolutional Neural Network 9
2.1. GIỚI THIỆU 9
2.2. Kiến trúc Mạng nơ-ron tích chập 9
2.2.1. Lớp tích chập - Convolutional 10
2.2.2. Lớp kích hoạt phi tuyến ReLU - Rectified Linear Unit 12
2.2.3. Lớp lấy mẫu – Pooling 12
2.2.4. Lớp kết nối đầy đủ – Fully Connected 13
CHƯƠNG 3 – GIỚI THIỆU TENSORFLOW VÀ KERAS 14
3.1. TENSORFLOW 14
3.2. KERAS 15
CHƯƠNG 4 – ỨNG DỤNG NHẬN DẠNG HOA DÙNG TENSORFLOW VÀ KERAS 17
4.1. MÔ TẢ BÀI TOÁN VÀ DỮ LIỆU 17
4.2. HUẤN LUYỆN MÔ HÌNH 17
4.2.1. Chuẩn bị tập dữ liệu 17
4.2.2. Chuyển tập dữ liệu thành mảng 18
4.2.3. Phân chia tập huấn luyện và tập dữ liệu 19
4.2.4. Xây dựng cấu trúc mạng neuron 20
4.2.5. Tăng kích thước tập huấn luyện 23
4.2.6. Huấn luyện mô hình 24
4.2.7. Lưu mô hình 25
4.3. Kết quả 25
4.3.1. Độ chính xác 25
4.3.2. Confusion Matrix 25
4.3.3. Kiểm tra với ảnh ngoài tập dữ liệu 26
CHƯƠNG 5 – KẾT LUẬN 28
TÀI LIỆU THAM KHẢO 29

DANH MỤC HÌNH ẢNH

Hình 1.1 - Các khó khăn trong bài toán nhận dạng vật thể trong ảnh 2
Hình 1.2 - Sự đa dạng về chủng loại của hoa hồng 2
Hình 1.3 – Các bước cơ bản thực hiện xử lý ảnh về nhận dạng hoa 3
Hình 1.4 – Mô hình hoạt động chung của các phương pháp Machine Learning 4
Hình 1.5 – Mối quan hệ của Deep Learning với các lĩnh vực liên quan 7
Hình 1.6 – Mức độ trừu tượng tăng dần qua các tầng học của Deep Learning 7
Hình 2.1 – Kiến trúc cơ bản của một mạng tích chập 10
Hình 2.2 – Bộ lọc tích chập được sử dụng trên ma trận điểm ảnh 10
Hình 2.3 – Trường hợp thêm/không thêm viền trắng vào ảnh khi tích chập 11
Hình 2.4 – Phương thức Avarage Pooling và Max Pooling 13
Hình 3.1 – Tensorflow framework 14
Hình 3.2 – Cấu trúc của Keras 15
Hình 4.1 – Confusion matrix sau khi huấn luyện 26
Hình 4.2 – Kết quả phân loại đúng loại hoa 27
Hình 4.2 – Kết quả phân loại sai loại hoa 27

CHƯƠNG 1 – GIỚI THIỆU TỔNG QUAN

1.1. TỔNG QUAN VỀ NHẬN DẠNG VÀ PHÂN LOẠI HÌNH ẢNH

Nhận dạng vật thể trong ảnh được coi là bài toán cơ bản nhất trong lĩnh vực Thị giác máy tính, là nền tảng cho rất nhiều bài toán mở rộng khác như bài toán phân lớp, định vị, tách biệt vật thể.

Tuy bài toán cơ bản này đã tồn tại hàng thế kỷ nhưng con người vẫn chưa thể giải quyết nó một cách triệt để, do tồn tại rất nhiều khó khăn để máy tính có thể hiểu được các thông tin trong một bức ảnh. Trong đó, những khó khăn tiêu biểu phải kể đến:

- **Sự đa dạng trong điểm nhìn – Viewpoint**: Cùng một vật thể nhưng có thể có rất nhiều vị trí và góc nhìn khác nhau, dẫn đến các hình ảnh thu được về vật thể đó sẽ không giống nhau. Việc huấn luyện để máy tính có thể hiểu được điều này thực sự là một thách thức khó khăn.
- **Sự đa dạng trong kích thước**: Các bức ảnh không có cách nào thể hiện trường thông tin về kích thước của vật thể trong đời thực, và máy tính cũng chỉ có thể tính toán được tỉ lệ tương đối của vật thể so với bức ảnh bằng cách đếm theo số lượng các điểm ảnh vật thể đó chiếm trong ảnh.
- **Các điều kiện khác nhau của chiếu sáng**: Ánh sáng có ảnh hưởng mạnh mẽ đến thông tin thể hiện trong một bức ảnh, đặc biệt là ở mức độ thấp như mức độ điểm ảnh.
- **Sự ẩn giấu một phần của vật thể sau các đối tượng khác trong ảnh**: Trong các bức ảnh, vật thể không nhất định phải xuất hiện với đầy đủ hình dạng mà có thể bị che lấp một phần nào đó bởi nền hoặc các vật thể xung quanh. Sự không đầy đủ về hình dạng của vật thể sẽ dẫn đến việc thiếu thông tin, đặc trưng và càng làm bài toán nhận dạng khó khăn hơn.
- **Sự lộn xộn phức tạp của nền**: Trong nhiều trường hợp, vật thể cần nhận dạng bị lẫn gần như hoàn toàn vào nền của bức ảnh, sự lẫn lộn về màu sắc, họa tiết giữa vật thể và nền khiến cho việc nhận dạng trở nên vô cùng khó khăn, kể cả với thị giác con người.
- **Sự đa dạng về chủng loại vật thể**: Vật thể cần nhận dạng có thể bao gồm nhiều chủng loại khác nhau, với hình dạng, màu sắc, kết cấu vô cùng khác biệt.

Đây chính là một thách thức nữa với bài toán nhận dạng, đó là làm thế nào để các mô hình nhận dạng của máy tính có thể nhận biết được các biến thể về chủng loại của vật thể.

Hình 1.1 - Các khó khăn trong bài toán nhận dạng vật thể trong ảnh

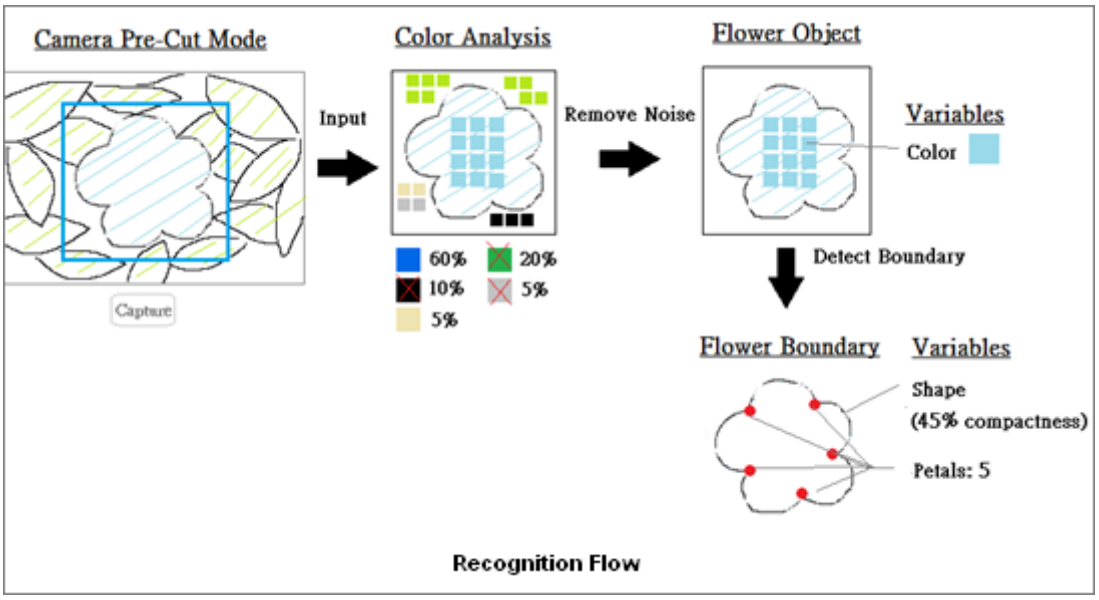
Là một trường hợp cụ thể của bài toán nhận dạng và phân lớp, bài toán nhận dạng hoa thừa các khó khăn vốn có của bài toán gốc, và kèm theo là các khó khăn riêng của chính nó, như: số lượng khổng lồ về chủng loại hoa theo mùa, vùng miền, địa hình… với vô số loại hoa có hình dáng, màu sắc, kết cấu giống nhau, dải biến thiên màu sắc theo chu kỳ phát triển của hoa từ lúc còn là búp cho tới lúc tàn héo, hay sự đa dạng về hình dạng của cùng một loại hoa do ảnh hưởng của thời tiết, điều kiện thổ nhưỡng và chế độ dinh dưỡng…

Hình 1.2 - Sự đa dạng về chủng loại của hoa hồng

1.2. CÁC HƯỚNG TIẾP CẬN VÀ GIẢI QUYẾT BÀI TOÁN

Bài toán tự động nhận dạng hoa quả 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 các 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 các tham số về màu sắc, hình dạng, kết cấu, kích thước…, từ bức ảnh đầu vào để nhận dạng hoa.

Do chỉ đơn thuần xử lý trên một vài ảnh đầu vào trong khi sự biến thiên về màu sắc, hình dạng, kích thước… của hoa quả phức tạp, kết quả đạt được của các phương pháp này không được cao và phạm vi áp dụng trên số lượng loại hoa quả cũng bị hạn chế.



Hình 1.3 – Các bước cơ bản thực hiện xử lý ảnh về nhận dạng hoa

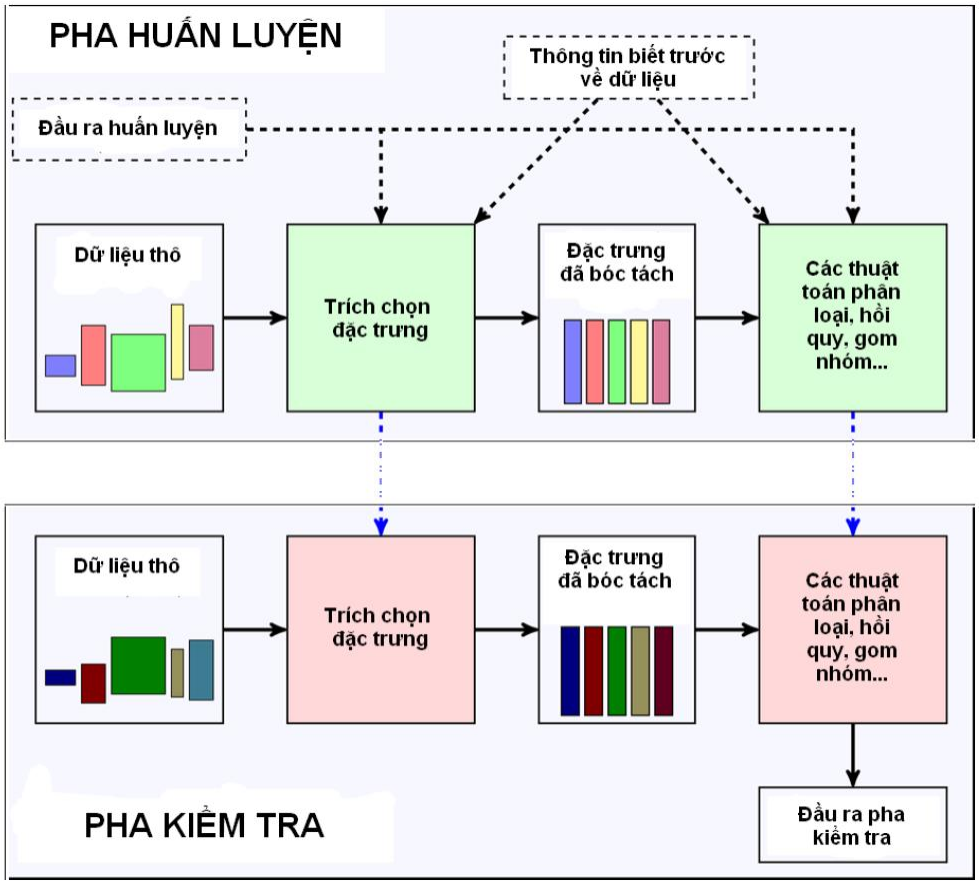
Bắt đầu từ những năm 2000s, sau khi xuất hiện một bài bài báo khoa học đề xuất áp dụng phương pháp Học máy - Machine Learning - vào bài toán nhận dạng hoa quả với độ chính xác cao, hướng giải quyết bài toán đã tập trung vào ứng dụng và cải tiến các thuật toán Machine Learning, cụ thể là nghiên cứu, thử nghiệm trích chọn các đặc trưng phù hợp nhất để đưa vào huấn luyện bộ nhận dạng tự động.

Kết quả thu được tương đối khả quan, khả năng nhận dạng hoa tự động đã được cải thiện với số lượng loại hoa được mở rộng và độ chính xác của nhận dạng cao hơn nhiều so với các phương pháp thuần Xử lý ảnh ban đầu.

Nối tiếp sự phát triển của Machine Learning, trong những năm gần đây, nhờ sự phát triển vượt bậc về sức mạnh tính toán của các máy tính cũng như sự bùng nổ dữ liệu trên Internet, một nhánh đặc biệt trong Machine Learning là Học sâu - Deep Learning đã đạt được nhiều thành tựu đáng kể, đặc biệt là trong lĩnh vực Xử lý ảnh và ngôn ngữ tự nhiên. Deep Learning cũng đã được áp dụng rất thành công vào bài toán nhận dạng hoa, trong các thử nghiệm với phạm vi hạn chế về số lượng loại hoa cần nhận dạng, phương pháp này đã đạt được kết quả rất cao.

1.2.1. Phương pháp Machine Learning truyền thống

Mô hình hoạt động chung của các phương pháp Machine Learning truyền thống được thể hiện trong Hình 1.4 dưới đây:



Hình 1.4 – Mô hình hoạt động chung của các phương pháp Machine Learning

Từ hình ta có thể thấy Học máy gồm hai giai đoạn chính là Huấn luyện – Training và Thử nghiệm – Testing, trong mỗi giai đoạn đều sử dụng hai thành phần quan trọng nhất do người xử lý bài toán thiết kế, đó là Trích chọn đặc trưng – Feature Engineering (hay còn gọi là Feature Extraction) và Thuật toán phân loại, nhận dạng… - Algorithms. Hai thành phần này có ảnh hưởng trực tiếp đến kết quả bài toán, vì thế được thiết kế rất cẩn thận, tốn nhiều thời gian, đòi hỏi người thiết kế phải có kiến thức chuyên môn và nắm rõ đặc điểm của bài toán cần xử lý.

1.2.1.1. Trích chọn đặt trưng

Trong các bài toán thực tế, ta chỉ có được những dữ liệu thô chưa qua chọn lọc xử lý, và để có thể đưa các dữ liệu này vào huấn luyện ta cần có những phép biến đổi để biến các dữ liệu thô thành dữ liệu chuẩn, với khả năng biểu diễn dữ liệu tốt hơn. Các phép biến đổi bao gồm loại bỏ dữ liệu nhiễu và tính toán để lưu lại các thông tin đặc trưng, có ý nghĩa từ dữ liệu thô ban đầu. Các thông tin đặc trưng này là khác nhau với từng loại dữ liệu và bài toán cụ thể, vì thế trong từng trường hợp phép biến đổi này cần phải được tùy biến một cách thích hợp để cải thiện độ chính xác của mô hình dự đoán. Quá trình này được gọi là Trích chọn đặc trưng – Feature Engineering, là một thành phần rất quan trọng trong các phương pháp Machine Learning truyền thống.

- **Đầu vào**: Toàn bộ thông tin của dữ liệu, không có quy chuẩn về dạng thông tin (véc tơ, ma trận…) hay kích thước các chiều thông tin. Đồng thời, do chứa toàn bộ thông tin, gồm cả thông tin nhiễu và không có giá trị nên kích thước lưu trữ thường lớn và không có lợi cho tính toán sau này.
- **Đầu ra**: Các thông tin hữu ích đã được tính toán, rút ra từ dữ liệu đầu vào, trong đó không còn các thành phần nhiễu hay vô nghĩa. Kích thước dữ liệu đầu ra đã được rút gọn rất nhiều so với kích thước dữ liệu đầu vào, giúp cho việc tính toán về sau trở nên nhanh gọn, thuận tiện hơn rất nhiều.
- **Thông tin biết trước về dữ liệu**: Đây là thành phần tùy chọn, không bắt buộc với mọi bài toán, mà chỉ xuất hiện trong một số trường hợp cụ thể với những thông tin rõ ràng về đặc trưng hữu ích với mô hình dự đoán. Các thông tin biết trước này giúp người thiết kế có thể lựa chọn được những đặc trưng tốt nhất và

các phương pháp tính toán phù hợp nhất để ra được mô hình dự đoán với độ chính xác cao.

1.2.1.2. Thuật toán

Sau quá trình trích chọn đặc trưng ở bước trước, ta có được các đặc trưng, được lưu trữ ở định dạng chuẩn về kiểu dữ liệu, kích thước dữ liệu…, và các thông tin đặc trưng này có thể được sử dụng cùng với các thông tin biết trước về dữ liệu (nếu có) để xây dựng ra các mô hình dự đoán phù hợp bằng các thuật toán khác nhau. Các thuật toán trong Machine Learning thường được phân loại theo hai cách phổ biến là theo phương thức học hoặc theo chức năng của thuật toán, ví dụ như:

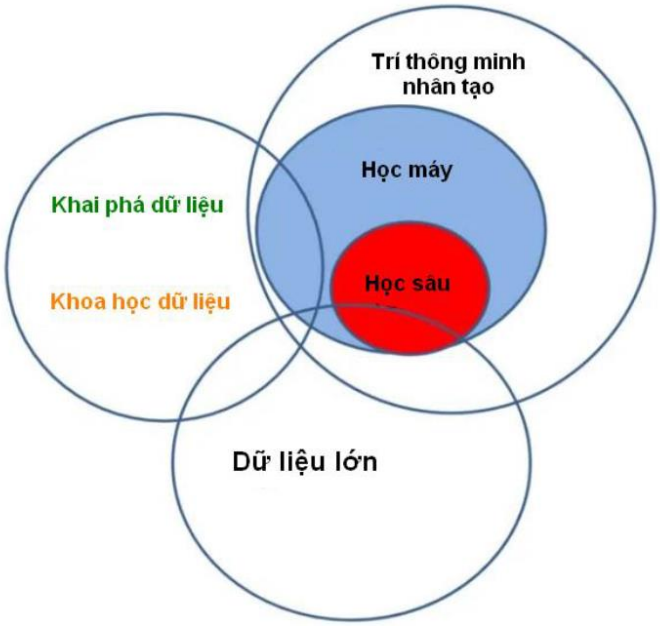
- **Phân nhóm theo phương thức học**: Học giám sát và Học không giám sát (Supervised và Unsupervised Learning)
- **Phân nhóm theo chức năng**: Các thuật toán hồi quy, phân loại, gom nhóm…

Một đặc điểm nổi bật của các phương pháp Machine Learning truyền thống là độ chính xác của mô hình dự đoán phụ thuộc rất nhiều vào chất lượng các đặc trưng được lựa chọn, các đặc trưng này càng phù hợp với bài toán đưa ra thì kết quả thu được càng tốt. Đây là điểm mạnh, và cũng là điểm yếu của các phương pháp này, bởi việc trích chọn đặc trưng chính là sự đóng góp của bản tay con người trong việc cải tiến các mô hình, nó yêu cầu sự hiểu biết thấu đáo về bài toán cần giải quyết, các thuật toán sử dụng và các thông số trong mô hình huấn luyện. Các đặc trưng được thiết kế riêng cho từng bài toán khác biệt, do vậy hiếm khi chúng có thể được tái sử dụng với các bài toán mới mà cần phải được cải thiện hay thay thế bởi các đặc trưng khác.

1.2.2. Phương pháp Deep Learning

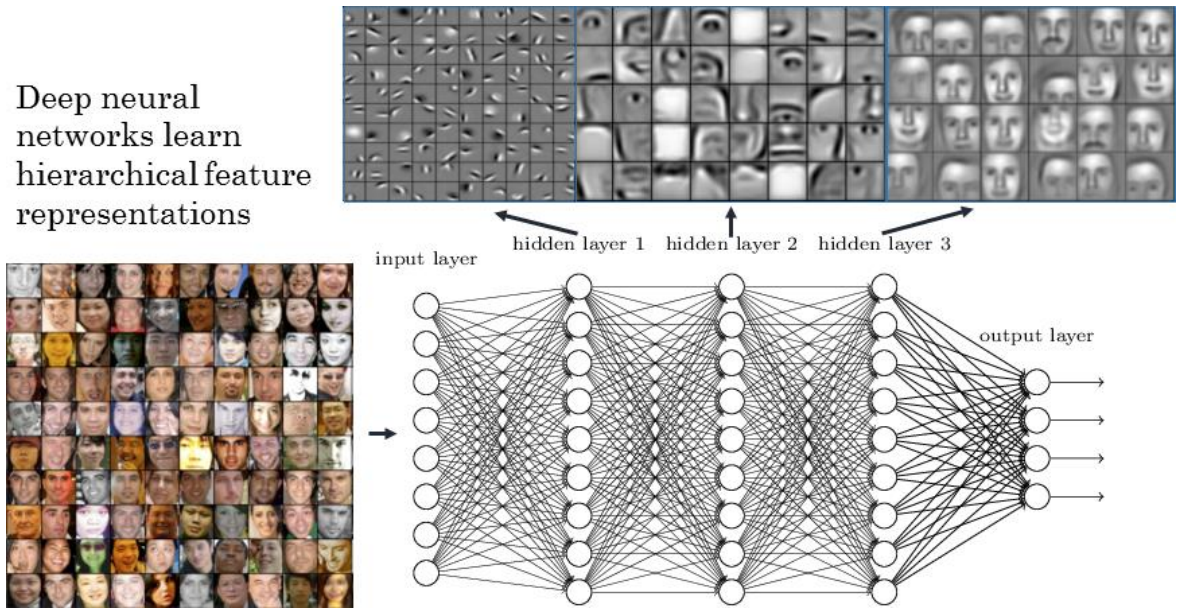
Học sâu - Deep Learning là một nhánh đặc biệt của ngành Machine Learning, và bắt đầu trở nên phổ biến trong thập kỷ gần đây do các nhà khoa học đã có thể tận dụng khả năng tính toán mạnh mẽ của các máy tính hiện đại cũng như khối lượng dữ liệu khổng lồ (hình ảnh, âm thanh, văn bản,…) trên Internet.

Ta có thể thấy rõ mối quan hệ giữa Deep Learning với Machine Learning cũng như các lĩnh vực liên quan khác qua hình ảnh mô tả bên dưới (Hình 1.5):



Hình 1.5 – Mối quan hệ của Deep Learning với các lĩnh vực liên quan

Các mạng huấn luyện theo phương pháp Deep Learning còn được gọi với cái tên khác là **Mạng nơ-ron sâu (Deep Neural Network)** do cách thức hoạt động của chúng. Về cơ bản, các mạng này bao gồm rất nhiều lớp khác nhau, mỗi lớp sẽ phân tích dữ liệu đầu vào theo các khía cạnh khác nhau và theo mức độ trừu tượng nâng cao dần



Hình 1.6 – Mức độ trừu tượng tăng dần qua các tầng học của Deep Learning

Cụ thể, với một mạng Học sâu cho nhận dạng ảnh, các lớp đầu tiên trong mạng chỉ làm nhiệm vụ rất đơn giản là tìm kiếm các đường thẳng, đường cong, hoặc đốm màu trong ảnh đầu vào. Các thông tin này sẽ được sử dụng làm đầu vào cho các lớp tiếp theo, với nhiệm vụ khó hơn là từ các đường, các cạnh đó tìm ra các thành phần của vật thể trong ảnh. Cuối cùng, các lớp cao nhất trong mạng huấn luyện sẽ nhận nhiệm vụ phát hiện ra vật thể trong ảnh.

Với cách thức học thông tin từ ảnh lần lượt qua rất nhiều lớp, nhiều tầng khác nhau như vậy, các phương pháp này có thể giúp cho máy tính hiểu được những dữ liệu phức tạp bằng nhiều lớp thông tin đơn giản qua từng bước phân tích. Đó cũng là lý do chúng được gọi là các phương pháp Học sâu - Deep Learning.

Tuy có nhiều điểm ưu việt trong khả năng huấn luyện máy tính cho các bài toán phức tạp, Deep Learning vẫn còn rất nhiều giới hạn khiến nó chưa thể được áp dụng vào giải quyết mọi vấn đề. Điểm hạn chế lớn nhất của phương pháp này là yêu cầu về kích thước dữ liệu huấn luyện, mô hình huấn luyện Deep Learning đòi hỏi phải có một lượng khổng lồ dữ liệu đầu vào để có thể thực hiện việc học qua nhiều lớp với một số lượng lớn nơ-ron và tham số. Đồng thời, việc tính toán trên quy mô dữ liệu và tham số lớn như vậy cũng yêu cầu đến sức mạnh xử lý của các máy tính server cỡ lớn.

Quy trình chọn lọc dữ liệu cũng như huấn luyện mô hình đều tốn nhiều thời gian và công sức, dẫn đến việc thử nghiệm các tham số mới cho mô hình là công việc xa xỉ, khó thực hiện.

Ngoài hạn chế về kích thước dữ liệu đầu vào, Deep Learning còn chưa đủ thông minh để nhận biết và hiểu được các logic phức tạp như con người, các tác vụ do chúng thực hiện vẫn tương đối máy móc và cần cải thiện để “thông minh” hơn nữa.

CHƯƠNG 2 – CNN - Convolutional Neural Network

2.1. GIỚI THIỆU

Mạng nơ-ron tích chập (CNN - Convolutional Neural Network) là một trong những mô hình mạng Deep Learning phổ biến nhất hiện nay, có khả năng nhận dạng và phân loại hình ảnh với độ chính xác rất cao, thậm chí còn tốt hơn con người trong nhiều trường hợp. Mô hình này đã và đang được phát triển, ứng dụng vào các hệ thống xử lý ảnh lớn của Facebook, Google hay Amazon… cho các mục đích khác nhau như các thuật toán tagging tự động, tìm kiếm ảnh hoặc gợi ý sản phẩm cho người tiêu dùng.

Sự ra đời của mạng CNN là dựa trên ý tưởng cải tiến cách thức các mạng nơ-ron nhân tạo truyền thống học thông tin trong ảnh. Do sử dụng các liên kết đầy đủ giữa các điểm ảnh vào node, các mạng nơ-ron nhân tạo truyền thẳng (Feedforward Neural Network) bị hạn chế rất nhiều bởi kích thước của ảnh, ảnh càng lớn thì số lượng liên kết càng tăng nhanh và kéo theo sự bùng nổ khối lượng tính toán. Ngoài ra sự liên kết đầy đủ này cũng là sự dư thừa khi với mỗi bức ảnh, các thông tin chủ yếu thể hiện qua sự phụ thuộc giữa các điểm ảnh với những điểm xung quanh nó mà không quan tâm nhiều đến các điểm ảnh ở cách xa nhau.

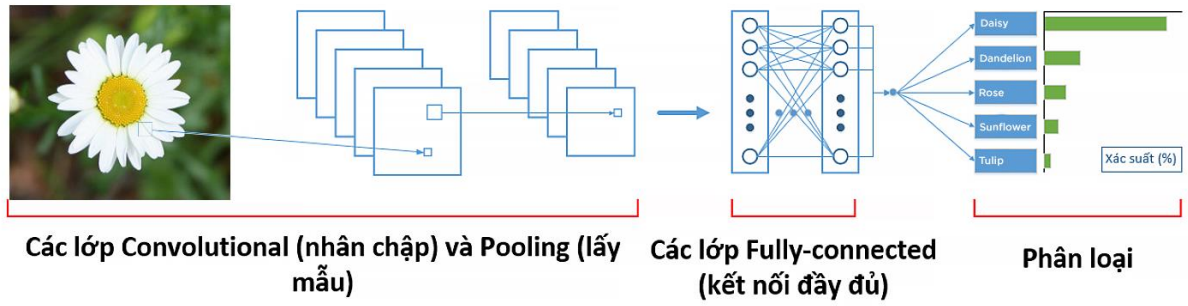
Mạng CNN ra đời với kiến trúc thay đổi, có khả năng xây dựng liên kết chỉ sử dụng một phần cục bộ trong ảnh kết nối đến node trong lớp tiếp theo thay vì toàn bộ ảnh như trong mạng nơ-ron truyền thẳng.

2.2. Kiến trúc Mạng nơ-ron tích chập

Các lớp cơ bản trong một mạng CNN bao gồm:

- ❖ Lớp tích chập (Convolutional)
- ❖ Lớp kích hoạt phi tuyến ReLU (Rectified Linear Unit)
- ❖ Lớp lấy mẫu (Pooling)
- ❖ Lớp kết nối đầy đủ (Fully - connected)

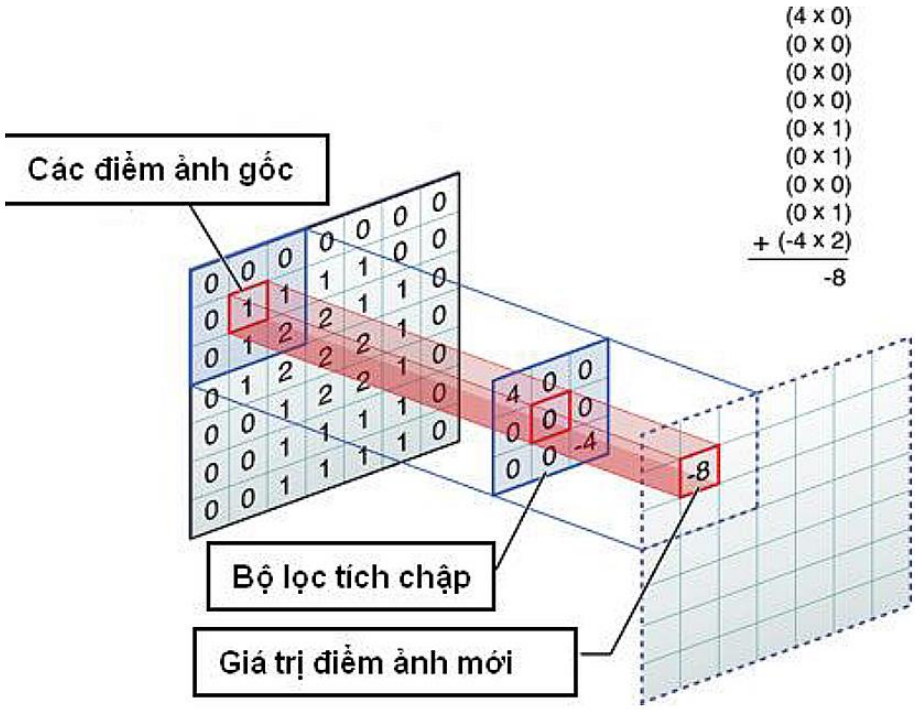
Các lớp được thay đổi về số lượng và cách sắp xếp để tạo ra các mô hình huấn luyện phù hợp cho từng bài toán khác nhau.



Hình 2.1 – Kiến trúc cơ bản của một mạng tích chập

2.2.1. Lớp tích chập - Convolutional

Đây là thành phần quan trọng nhất trong mạng CNN, cũng là nơi thể hiện tư tưởng xây dựng sự liên kết cục bộ thay vì kết nối toàn bộ các điểm ảnh. Các liên kết cục bộ này được tính toán bằng phép tích chập giữa các giá trị điểm ảnh trong một vùng ảnh cục bộ với các bộ lọc – filters – có kích thước nhỏ.



Hình 2.2 – Bộ lọc tích chập được sử dụng trên ma trận điểm ảnh

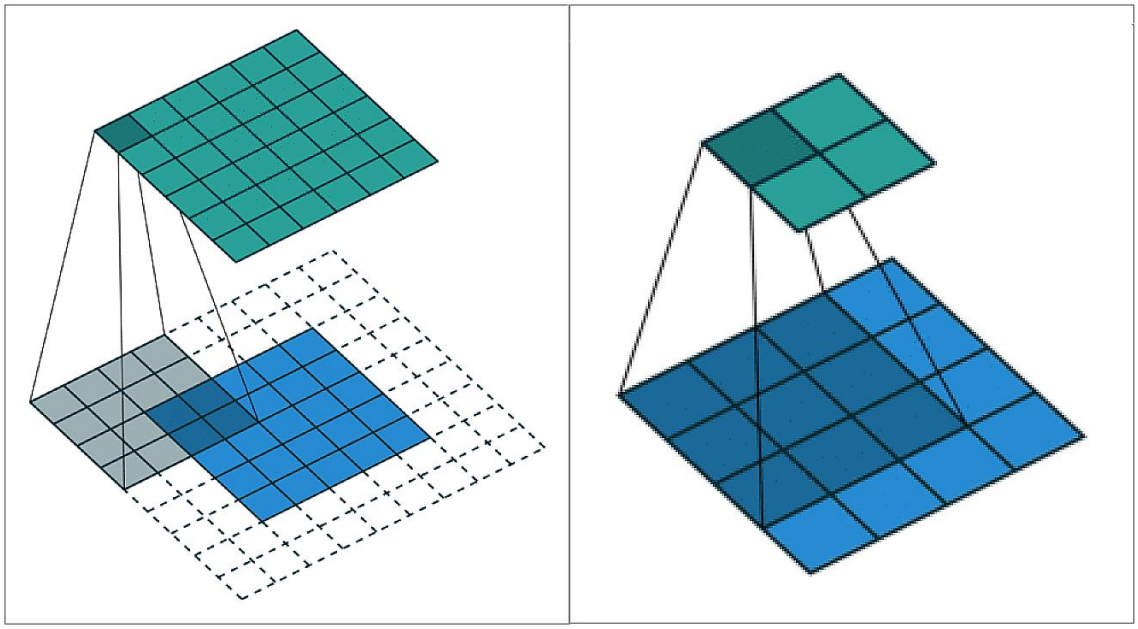
Trong ví dụ ở Hình 2.2, ta thấy bộ lọc được sử dụng là một ma trận có kích thước 3x3. Bộ lọc này được dịch chuyển lần lượt qua từng vùng ảnh đến khi hoàn thành

quét toàn bộ bức ảnh, tạo ra một bức ảnh mới có kích thước nhỏ hơn hoặc bằng với kích thước ảnh đầu vào. Kích thước này được quyết định tùy theo kích thước các khoảng trắng được thêm ở viền bức ảnh gốc và được tính theo công thức:

$$o = \frac{i + 2 * p - k}{s} + 1$$

Trong đó:

- o: kích thước ảnh đầu ra
- i: kích thước ảnh đầu vào
- p: kích thước khoảng trắng phía ngoài viền của ảnh gốc - k: kích thước bộ lọc
- s: bước trượt của bộ lọc



Hình 2.3 – Trường hợp thêm/không thêm viền trắng vào ảnh khi tích chập

Như vậy, sau khi đưa một bức ảnh đầu vào cho lớp Tích chập ta nhận được kết quả đầu ra là một loạt ảnh tương ứng với các bộ lọc đã được sử dụng để thực hiện phép tích chập. Các trọng số của các bộ lọc này được khởi tạo ngẫu nhiên trong lần đầu tiên và sẽ được cải thiện dần xuyên suốt quá trình huấn luyện.

2.2.2. Lớp kích hoạt phi tuyến ReLU - Rectified Linear Unit

Lớp này được xây dựng với ý nghĩa đảm bảo tính phi tuyến của mô hình huấn luyện sau khi đã thực hiện một loạt các phép tính toán tuyến tính qua các lớp Tích chập. Lớp Kích hoạt phi tuyến nói chung sử dụng các hàm kích hoạt phi tuyến như ReLU hoặc sigmoid, tanh… để giới hạn phạm vi biên độ cho phép của giá trị đầu ra. Trong số các hàm kích hoạt này, hàm ReLU được chọn do cài đặt đơn giản, tốc độ xử lý nhanh mà vẫn đảm bảo được tính toán hiệu quả. Cụ thể, phép tính toán của hàm ReLU chỉ đơn giản là chuyển tất cả các giá trị âm thành giá trị 0.

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

Thông thường, lớp ReLU được áp dụng ngay phía sau lớp Tích chập, với đầu ra là một ảnh mới có kích thước giống với ảnh đầu vào, các giá trị điểm ảnh cũng hoàn toàn tương tự trừ các giá trị âm đã bị loại bỏ.

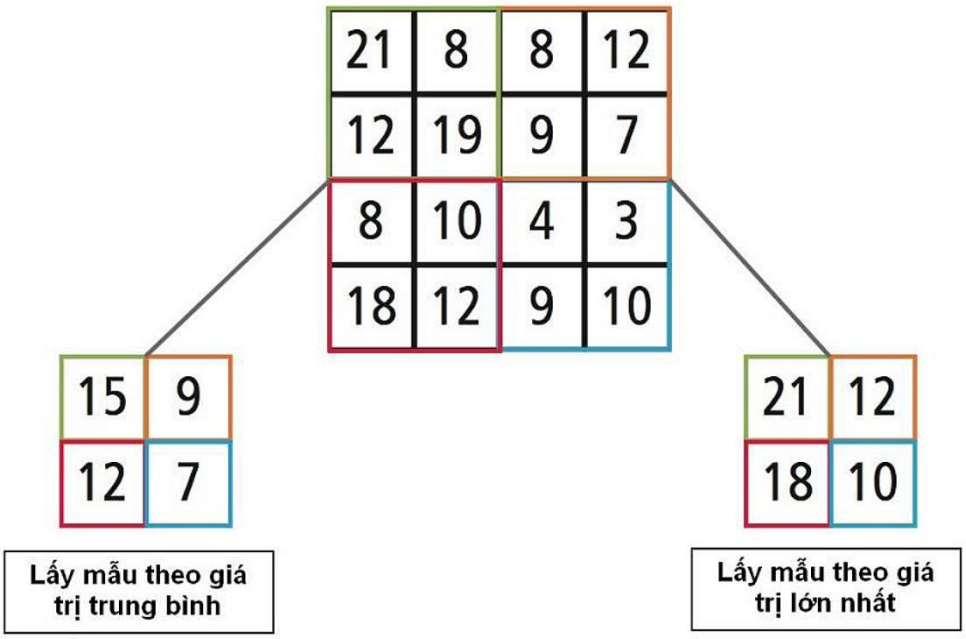
2.2.3. Lớp lấy mẫu – Pooling

Một thành phần tính toán chính khác trong mạng CNN là lấy mẫu (Pooling), thường được đặt sau lớp Tích chập và lớp ReLU để làm giảm kích thước kích thước ảnh đầu ra trong khi vẫn giữ được các thông tin quan trọng của ảnh đầu vào. Việc giảm kích thước dữ liệu có tác dụng làm giảm được số lượng tham số cũng như tăng hiệu quả tính toán.

Lớp lấy mẫu cũng sử dụng một cửa sổ trượt để quét toàn bộ các vùng trong ảnh tương tự như lớp Tích chập, và thực hiện phép lấy mẫu thay vì phép tích chập – tức là ta sẽ chọn lưu lại một giá trị duy nhất đại diện cho toàn bộ thông tin của vùng ảnh đó.

Hình 2.4 thể hiện các phương thức lấy mẫu thường được sử dụng nhất hiện nay, đó là Max Pooling (lấy giá trị điểm ảnh lớn nhất) và Avarage Pooling (lấy giá trị trung bình của các điểm ảnh trong vùng ảnh cục bộ).

Như vậy, với mỗi ảnh đầu vào được đưa qua lấy mẫu ta thu được một ảnh đầu ra tương ứng, có kích thước giảm xuống đáng kể nhưng vẫn giữ được các đặc trưng cần thiết cho quá trình tính toán sau này.



Hình 2.4 – Phương thức Avarage Pooling và Max Pooling

2.2.4. Lớp kết nối đầy đủ – Fully Connected

Lớp kết nối đầy đủ này được thiết kế hoàn toàn tương tự như trong mạng nơ-ron truyền thống, tức là tất cả các điểm ảnh được kết nối đầy đủ với node trong lớp tiếp theo.

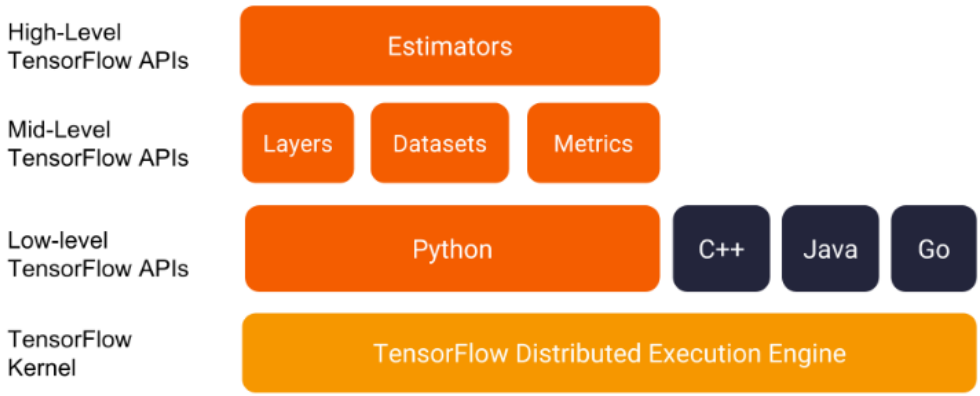
So với mạng nơ-ron truyền thống, các ảnh đầu vào của lớp này đã có kích thước được giảm bớt rất nhiều, đồng thời vẫn đảm bảo các thông tin quan trọng cho việc nhận dạng. Do vậy, việc tính toán nhận dạng sử dụng mô hình truyền thẳng đã không còn phức tạp và tốn nhiều thời gian như trong mạng nơ ron truyền thống.

CHƯƠNG 3 – GIỚI THIỆU TENSORFLOW VÀ KERAS

3.1. TENSORFLOW

TensorFlow là một thư viện mã nguồn mở được Google phát triển nhằm mục đích hỗ trợ việc nghiên và phát triển các ứng dụng AI (Machine learning/Deep learning). Phiên bản đầu tiên của Tensor Flow được công bố vào tháng 11/2015. Phần Core của TensorFlow được viết bằng C++ nhưng người dùng có thể sử dụng cả C++ hoặc Python để phát triển ứng dụng của mình (python vẫn là ngôn ngữ được sử dụng nhiều nhất để viết TensorFlow).

Tên gọi TensorFlow được bắt nguồn bằng cách kết hợp Tensor + Flow (dòng chảy). Ở đây, Tensor chính là một kiểu dữ liệu dạng mảng có nhiều chiều (ví dụ 1-d tensor hay còn gọi là vector hoặc 2-d tensor hay còn gọi là matrix). Do đó chúng ta có thể hiểu đơn giản, TensorFlow chính là các dòng/luồng dữ liệu Tensor.



Hình 3.1 – Tensorflow framework

Không những vậy, đi kèm với TensorFlow còn có một số công cụ hỗ trợ như:

- TensorBoard: công cụ giúp minh họa các đồ thị tính toán (computational graph), sự thay đổi giá trị của các hàm tính toán (loss, accuracy, …) dưới dạng biểu đồ.
- TensorFlow Serving: công cụ giúp triển khai các mô hình Machine Learning viết bằng TensorFlow thành một sản phẩm thực sự.
- Các API giúp cho việc sử dụng TensorFlow dễ dàng hơn được phát triển bởi những nhà nghiên cứu về Machine Learning trên toàn thế giới (TensorFlow High Level API, TF-Slim, TensorFlow Object Detection API)

- Tập hợp code mẫu giúp cho người dùng dễ tiếp cận hơn.

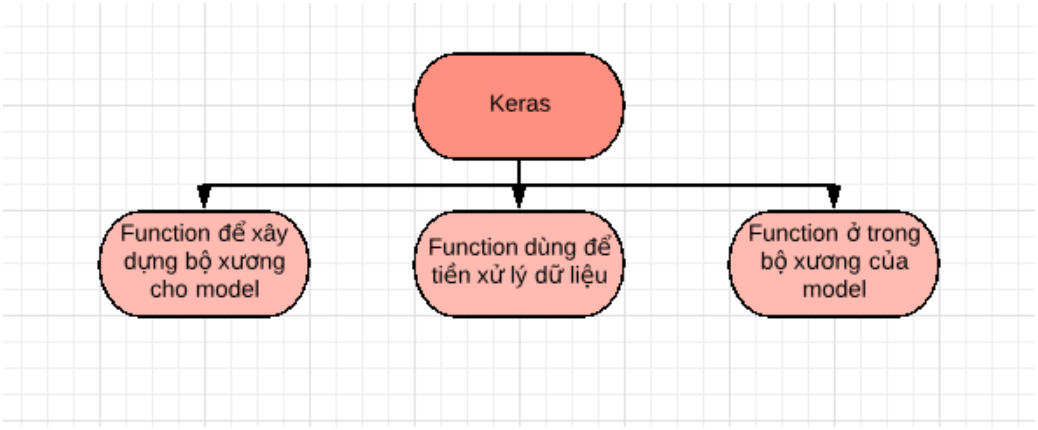
3.2. KERAS

Keras là một library được phát triển vào năm 2015 bởi François Chollet, là một kỹ sư nghiên cứu deep learning tại google. Nó là một open source cho neural network được viết bởi ngôn ngữ python.

Keras là một API bậc cao có thể sử dụng chung với các thư viện deep learning nổi tiếng như Tensorflow (được phát triển bởi Google), CNTK (được phát triển bởi Microsoft). Keras có một số ưu điểm như:

- Dễ sử dụng, xây dựng model nhanh.
- Có thể run trên cả cpu và gpu
- Hỗ trợ xây dựng CNN, RNN và có thể kết hợp cả 2.

Cấu trúc của Keras có thể chia ra thành 3 phần chính:



Hình 3.2 – Cấu trúc của Keras

Để khởi tạo một model trong Keras ta có thể dùng 2 cách:

- **Cách 1**: Thông qua Sequential.

  Khởi tạo model bằng Sequential sau đó dùng method add để thêm các layer.

- **Cách 2**: Dùng function API.

  Xem input cũng là một layer sau đó build từ input tới output sau đó kết hợp lại bằng hàm Model. Ưu điểm của phương pháp này có thể tùy biến nhiều hơn, giúp ta xây dựng các model phức tạp nhiều input và output.

Các method cần lưu ý khi khởi tạo một model là:

- **Compile**: Sau khi build model xong thì compile nó có tác dụng biên tập lại toàn bộ model của chúng ta đã build. Ở đây chúng ta có thể chọn các tham số để training model như: thuật toán training thông qua tham số optimizer, function loss của model chúng ta có thể sử dụng mặc định hoặc tự build thông qua tham số loss, chọn metrics hiện thị khi model được training.
- **Summary method**: tổng hợp lại model xem model có bao nhiêu layer, tổng số tham số bao nhiêu, shape của mỗi layer …
- **Fit**: đưa data vào training để tìm tham số model (tương tự như sklearn)
- **Predict**: dự đoán các new instance
- **Evaluate**: tính toán độ chính xác của model
- **History**: xem accuracy, loss qua từng epochs. Thường dùng với matplotlib để vẽ chart.

CHƯƠNG 4 – ỨNG DỤNG NHẬN DẠNG HOA DÙNG TENSORFLOW VÀ KERAS

4.1. MÔ TẢ BÀI TOÁN VÀ DỮ LIỆU

Mục tiêu là xây dựng một mô hình mạng neuron cho phép nhận dạng 5 loại hoa từ một tấm ảnh chụp. Danh sách 5 loại hoa này bao gồm:

- Daisy (Hoa cúc)
- Dandelion (Hoa bồ công anh)
- Rose (Hoa hồng)
- Sunflower (Hoa hướng dương)
- Tulip (Hoa uất kim hương)

Bộ dữ liệu dùng để huấn luyện mô hình được lấy về từ kaggle:

https://www.kaggle.com/alxmamaev/flowers-recognition

Tập dữ liệu này bao gồm 4242 hình cảnh của 5 loại hoa hồng (rose), hoa mặt trời (sunflower), hoa bồ công anh (dandelion), hoa cúc (daisy) và hoa tulip. Nhóm tác giả đã thu thập dữ liệu dựa trên các trang web flicr, google images, yandex.

Tập hình ảnh được chia thành 5 lớp, mỗi lớp có khoảng 800 hình, có kích thước xấp xỉ 320x320 pixel. Các hình ảnh có kích thước không đồng nhất với nhau.

Các loại hoa có thể được phân biệt nhờ: Hình dạng của chúng, (hoa hồng và hoa hướng dương), màu sắc (hoa cúc và bồ công anh), và các chi tiết phụ khác. Nguy cơ nhầm lẫn hoàn toàn có thể xảy ra giữa hoa hồng với tulip và hoa cúc với bồ công anh.

4.2. HUẤN LUYỆN MÔ HÌNH

4.2.1. Chuẩn bị tập dữ liệu

Giải nén tập dữ liệu và kiểm tra các thư mục.

```
import os
print(os.listdir("../input/flowers/flowers"))
```

Hiển thị ngẫu nhiên 1 tấm ảnh để kiểm tra tập dữ liệu

```
img = plt.imread("../input/flowers/flowers/daisy/100080576_f52e8ee070_n.jpg")
img = cv2.resize(img,(124,124))
```

```
plt.imshow(img)
plt.axis("off")
plt.show()
```

4.2.2. Chuyển tập dữ liệu thành mảng

Chuyển từ ảnh chụp thành dữ liệu mảng để phù hợp cho công cụ ta đang sử dụng (Keras và TensorFlow).

Thu nhỏ kích thước ảnh trong quá trình huấn luyện còn 256x256.

```
x_ = list()
y = list()
IMG_SIZE = 256
for i in os.listdir("../input/flowers/flowers/daisy"):
    try:
        path = "../input/flowers/flowers/daisy/"+i
        img = plt.imread(path)
        img = cv2.resize(img,(IMG_SIZE,IMG_SIZE))
        x_.append(img)
        y.append(0)
    except:
        None
for i in os.listdir("../input/flowers/flowers/dandelion"):
    try:
        path = "../input/flowers/flowers/dandelion/"+i
        img = plt.imread(path)
        img = cv2.resize(img,(IMG_SIZE,IMG_SIZE))
        x_.append(img)
        y.append(1)
    except:
        None
for i in os.listdir("../input/flowers/flowers/rose"):
    try:
        path = "../input/flowers/flowers/rose/"+i
```

```
            img = plt.imread(path)
            img = cv2.resize(img,(IMG_SIZE,IMG_SIZE))
            x_.append(img)
            y.append(2)
        except:
            None
    for i in os.listdir("../input/flowers/flowers/sunflower"):
        try:
            path = "../input/flowers/flowers/sunflower/"+i
            img = plt.imread(path)
            img = cv2.resize(img,(IMG_SIZE,IMG_SIZE))
            x_.append(img)
            y.append(3)
        except:
            None
    for i in os.listdir("../input/flowers/flowers/tulip"):
        try:
            path = "../input/flowers/flowers/tulip/"+i
            img = plt.imread(path)
            img = cv2.resize(img,(IMG_SIZE,IMG_SIZE))
            x_.append(img)
            y.append(4)
        except:
            None
    x_ = np.array(x_)
```

4.2.3. Phân chia tập huấn luyện và tập dữ liệu

Chuyển y_train, y_test về dạng one-hot coding, với cách mã hóa trên, ta có 5 classes tương ứng với 5 loại hoa là class 1, 2, 3, 4, 5 sẽ lần lượt được mã hóa dưới dạng nhị phân bởi 10000, 01000, 00100, 00010 hoặc 00001. One-hot vì chỉ có one bit là hot (bằng 1).

```
from keras.utils.np_utils import to_categorical
y = to_categorical(y,num_classes = 5)
```

Tiếp theo chúng ta dùng hàm train_test_split() trong thư viện klearn, chia dữ liệu thành hai phần: một tập huấn luyện và một tập kiểm tra.

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x_,y,test_size=0.15,random_state=42)
```

Hàm này tách ngẫu nhiên dữ liệu bằng cách sử dụng test_size tham số. Sau khi thực hiện xong, ta sẽ có 15% tập dữ liệu gốc cho tập kiểm tra (y_test) và 85% tập dữ liệu gốc cho tập huấn luyện (y_train). Và các nhãn tương ứng cho cả các biến x_train / x_test là y_train / y_test.

Do khi xây dựng mô hình, ta không được sử dụng tập kiểm tra nên để đánh giá chất lượng của mô hình ta tiếp tục trích từ tập tập huấn luyện ra một tập con nhỏ gọi là tập xác nhận (validation set) và thực hiện việc đánh giá mô hình trên tập con nhỏ này.

```
x_train,x_val,y_train,y_val=train_test_split(x_train,y_train,test_size=0.15,random_state = 42)
```

Sau khi thực hiện xong, ta sẽ có 15% dữ liệu từ tập huấn luyện cho tập xác nhận (y_val) và 85% dữ liệu còn lại cho tập huấn luyện (y_train).

4.2.4. Xây dựng cấu trúc mạng neuron

Xây dựng mạng neuron với cấu trúc gồm 5 lớp Convolutional, lớp Full Connected và softmax để phân lớp các loại hoa, việc huấn luyện sẽ tìm ra trọng số của lớp Full Connected cuối cùng.

Trong đó lớp convolutional đầu tiên tiếp nhận dữ liệu tensor với các thông số image width, image height, channels của ảnh trong tập huấn luyện, hàm kích hoạt là relu.

Mỗi lớp convolutional sẽ có thêm 1 lớp Pooling và 1 lớp Batch Normalization.

Tiếp theo các lớp Convolution sẽ là 1 lớp Flatten nối với 1 lớp Dense (1024 neuron) và 1 lớp Dropout sẽ chuyển giá trị đầu ra của lớp Pooling thành vector feature, sau đó tinh chỉnh.

Lớp cuối cùng: xuất kết quả, với số neuron đúng bằng số nhãn cần phân loại, với hàm kích hoạt là softmax.

Mô hình được compile với hàm loss = categorical_crossentropy và tùy chỉnh optimizer = adam, tiêu chí tối ưu hóa là accuracy.

```
model = Sequential()
# 1st Convolutional Layer
model.add(Conv2D(filters=64, kernel_size=(3,3),padding="Same",activation="relu"
, input_shape = (IMG_SIZE,IMG_SIZE,3)))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
model.add(BatchNormalization())
model.add(Dropout(0.2))
# 2nd Convolutional Layer
model.add(Conv2D(filters=128,
kernel_size=(3,3),padding="Same",activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
model.add(BatchNormalization())
model.add(Dropout(0.3))
# 3rd Convolutional Layer
model.add(Conv2D(filters=128,
kernel_size=(3,3),padding="Same",activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
model.add(BatchNormalization())
model.add(Dropout(0.3))
# 4th Convolutional Layer
model.add(Conv2D(filters=256,kernel_size =
(3,3),padding="Same",activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
model.add(BatchNormalization())
model.add(Dropout(0.2))
```

```
    # 5th Convolutional Layer
    model.add(Conv2D(filters=512,kernel_size =
(3,3),padding="Same",activation="relu"))
    model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
    model.add(BatchNormalization())
    model.add(Dropout(0.3))


    model.add(Flatten())
    # 1st Fully Connected Layer
    model.add(Dense(1024,activation="relu"))
    model.add(Dropout(0.5))
    model.add(BatchNormalization())
    # Add output layer
    model.add(Dense(5,activation="softmax"))


    model.summary() # print summary my model
    model.compile(loss='categorical_crossentropy',
optimizer='adam',metrics=['accuracy']) #compile model


    model.compile(loss='categorical_crossentropy',
            optimizer=Adam(lr=0.001),
            metrics=['accuracy'])
```

OUTPUT FORMULA FOR CONVOLUTION

$$O = \frac{W - K + 2P}{S} + 1$$

Trong đó:

- O : output height/length
- W : input height/length
- K : filter size (kernel size)
- P : same padding (non-zero)

$$P = \frac{K-1}{2}$$

- P: stride

OUTPUT FORMULA FOR POOLING

$$O = \frac{W-K}{S} + 1$$

Trong đó:

- W : input height/width
- K : filter size
- S : stride size = filter size

4.2.5. Tăng kích thước tập huấn luyện

Tiến hành dùng thư viện ImageDataGenerator để có được nhiều dữ liệu mẫu hơn và chống overfit.

Thư viện ImageDataGenerator giúp tăng kích thước của tập dữ liệu lên nhanh chóng đồng thời thay đổi, thêm mới các tính chất vào trong dữ liệu hiện tại giúp cho độ phong phú của dữ liệu trở nên dồi dào hơn. Kĩ thuật này đặc biệt hữu ích khi các tập dữ liệu của chúng ta là nhỏ. Một vài kĩ thuật trong số đó áp dụng cho dữ liệu dạng hình ảnh (image).

- **Flipping**: lật ảnh theo chiều dọc hay theo chiều ngang.
- **Random Cropping** - cắt ra ngẫu nhiên một thành phần trong hình ảnh gốc để làm dữ liệu mới.
- **Shearing** - thay đổi góc nhìn của hình ảnh, hình dạng của hình ảnh thông qua các giải thuật transform.
- **Rotation**: xoay ảnh theo một góc nào đó.
- **Whitening**: chỉ giữ lại các thành phần quan trọng trong ảnh thông qua giải thuật Principal Component Analysis.
- **Normalization** - là một phương pháp chuẩn hóa hình ảnh theo giá trị trung bình và độ lệch chuẩn.

- **Channel shifting**: thay đổi các kênh màu làm cho mô hình dễ dàng học được các đặc trưng từ ảnh hơn.

```
datagen = ImageDataGenerator (
    featurewise_center=False,  # set input mean to 0 over the dataset
    samplewise_center=False,  # set each sample mean to 0
    featurewise_std_normalization=False,  # divide inputs by std of the dataset
    samplewise_std_normalization=False,  # divide each input by its std
    rotation_range=60,  # randomly rotate images in the range (60, 0 to 180)
    zoom_range = 0.1, # Randomly zoom image
    width_shift_range=0.1,  # randomly shift images horizontally (fraction of total width)
    height_shift_range=0.1,
    shear_range=0.1,
    fill_mode = "reflect"
    )
datagen.fit(x_train)
```

4.2.6. Huấn luyện mô hình

Sau khi có cấu trúc mạng CNN, ta bắt đầu thực hiện quy trình huấn luyện cho nó. Quy trình này bắt đầu bằng việc đưa tensor data (train_image_array_gen, được chuẩn bị ở các bước trên) vào lớp tiếp nhận dữ liệu, mô hình được huấn luyện 50 lượt để tối ưu hóa accuracy.

```
history = model.fit_generator(datagen.flow(x_train,y_train,batch_size=batch_size),
                              epochs = epoch,
                              validation_data = (x_val,y_val),
                              steps_per_epoch = x_train.shape[0] // batch_size
                              )
```

4.2.7. Lưu mô hình

Sau khi kết thúc quá trình huấn luyện, chúng ta sử dụng model.save () để lưu trữ Model đã train để sử dụng cho các mục đích khác trong tương lai như dung để dự đoán ảnh mới.

```
model.save('../output/cusstom_cnn_model.h5') # Save a model as HDF5 file
```

4.3. Kết quả

4.3.1. Độ chính xác

Với mô hình đã train, ta tính tỉ lệ giữa số điểm được dự đoán đúng và tổng số điểm trong tập dữ liệu kiểm thử.

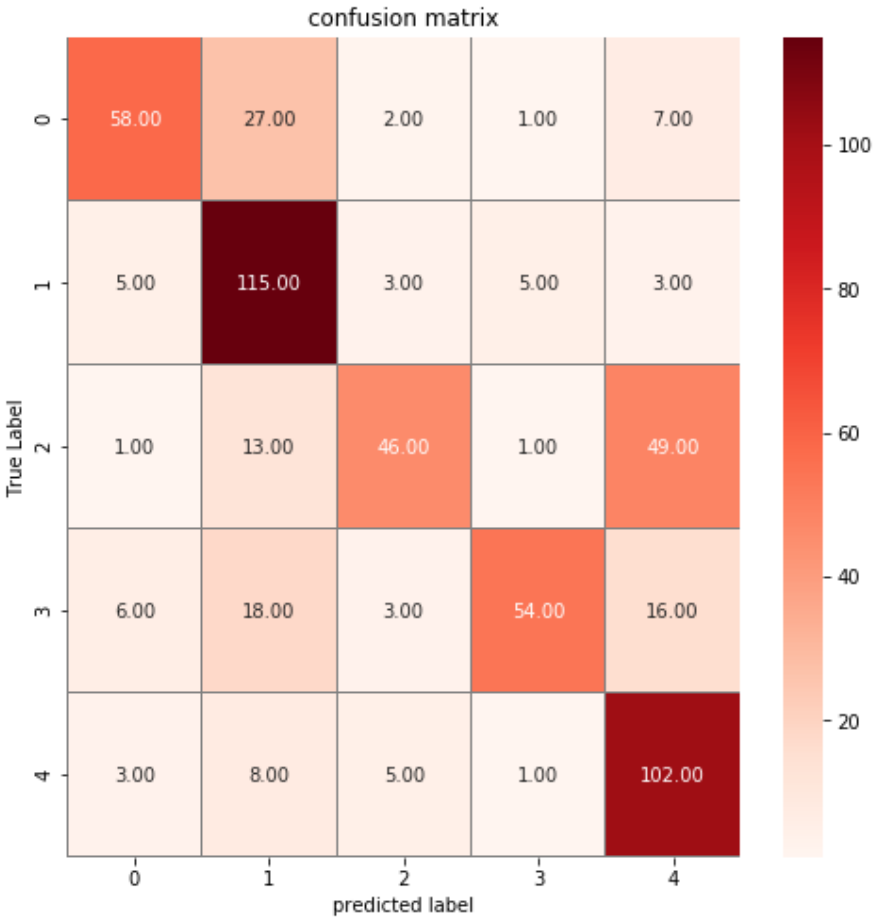
```
649/649 [==============================] - 136s 209ms/step
Test Accuracy: 71.34%
```

Với kết quả trên ta kết luận độ chính xác của mô hình là 71.34%.

4.3.2. Confusion Matrix

Cách tính sử dụng accuracy như ở trên chỉ cho chúng ta biết được bao nhiêu phần trăm lượng dữ liệu được phân loại đúng mà không chỉ ra được cụ thể mỗi loại được phân loại như thế nào, lớp nào được phân loại đúng nhiều nhất, và dữ liệu thuộc lớp nào thường bị phân loại nhầm vào lớp khác. Để có thể đánh giá được các giá trị này, chúng ta sử dụng một ma trận được gọi là confusion matrix.

Về cơ bản, confusion matrix thể hiện có bao nhiêu điểm dữ liệu thực sự thuộc vào một class, và được dự đoán là rơi vào một class. Ma trận thu được được gọi là confusion matrix. Nó là một ma trận vuông với kích thước mỗi chiều bằng số lượng lớp dữ liệu. Giá trị tại hàng thứ i, cột thứ j là số lượng điểm lẽ ra thuộc vào class i nhưng lại được dự đoán là thuộc vào class j.



Hình 4.1 – Confusion matrix sau khi huấn luyện

Chúng ta có thể suy ra ngay rằng tổng các phần tử trong toàn ma trận này chính là số điểm trong tập kiểm thử. Các phần tử trên đường chéo của ma trận là số điểm được phân loại đúng của mỗi lớp dữ liệu. Từ đây có thể suy ra accuracy chính bằng tổng các phần tử trên đường chéo chia cho tổng các phần tử của toàn ma trận. Đoạn code dưới đây mô tả cách tính confusion matrix:

4.3.3. Kiểm tra với ảnh ngoài tập dữ liệu

Sau khi chạy chương trình dự đoán, mô hình có thể dự đoán chính xác được ảnh hoa hướng dương. Nhưng trong một số trường hợp, chương trình dự đoán dễ phân biệt nhầm lẫn giữa hoa hồng và hoa tulip. Nguyên nhân là giữa 2 loại hoa này có nhiều điểm tương đồng, nên cần một thuật toán tối ưu hơn để khắc phục nhược điểm trên.

Hình 4.2 – Kết quả phân loại đúng loại hoa

Hình 4.3 – Kết quả phân loại sai loại hoa

CHƯƠNG 5 – KẾT LUẬN

Những điểm đã đạt được:

- Phân loại được sự khác nhau giữa phương pháp Machine Learning truyền thống và phương pháp deep learning.
- Hiểu được mô hình nơ ron tích chập CNN, quy trình phân tích và ứng dụng mô hình vào bài toán cụ thể là phân loại hoa.
- Ứng dụng tensorflow và keras để tiếp cận được machine learning nhanh hơn thông qua các thư viện có sẵn.
- Chương trình mô phỏng phân loại tương đối chính xác 5 loại hoa, với tỉ lệ khoảng 71%

Những mặt hạn chế:

- Quá trình huấn luyện mô hình chưa tối ưu, tập huấn luyện còn hạn chế.
- Chương trình mô phỏng chưa phân biệt tốt những loài hoa có các đặc trưng gần giống nhau.
- Chưa có giao diện GUI để tiện cho người dùng thao tác được nhanh chóng hơn.

TÀI LIỆU THAM KHẢO

[1] Trần Tuấn Linh. (2017). Ứng dụng nhận dạng hoa quả cho điện thoại thông minh dựa trên hình ảnh.

[2] François Chollet, Keras, https://github.com/fchollet/keras, 2015

[3] Vũ Hữu Tiệp. (2017). Machine Learning cơ bản http://machinelearningcoban.com/

[4] Andrej Karpathy. CS231n Convolutional Neural Networks for Visual Recognition - Image Classification. http://cs231n.github.io/classification/

[5] Các tài liệu về EmguCV tại www.emgucv.com OPenCV tại www.opencv.com.

[6] Huew Engineering. (2015). Introduction to Convolution Neural Networks – Huew Engineering.

https://engineering.huew.co/introduction-to-convolution-neural-networks-18981d1cd09a

[7] Naskar, S. (2015). A Fruit Recognition Technique using Multiple Features and Artificial Neural Network, 116(20), 23–28.