BỘ GIÁO DỤC ĐÀO TẠO

TRƯỜNG ĐẠI HỌC ĐẠI NAM

--- 🕮 ---

ĐỒ ÁN TỐT NGHIỆP

NGHIÊN CỨU VÀ ÁP DỤNG MÔ HÌNH HỌC MÁY/HỌC SÂU NHẬN DẠNG CÁC LOẠI LÁ, HOA

SINH VIÊN THỰC HIỆN : ĐỖ HỒNG QUÂN

MÃ SINH VIÊN : 1451020185

KHOA : CÔNG NGHỆ THÔNG TIN

HÀ NỘI – 2024

BỘ GIÁO DỤC ĐÀO TẠO

TRƯỜNG ĐẠI HỌC ĐẠI NAM

ĐỖ HỒNG QUÂN

NGHIÊN CỨU VÀ ÁP DỤNG MÔ HÌNH HỌC MÁY/ HỌC SÂU NHẬN DẠNG CÁC LOẠI LÁ, HOA

CHUYÊN NGÀNH : **CÔNG NGHỆ THÔNG TIN**

MÃ SỐ : **74.80.201**

NGƯỜI HƯỚNG DẪN: TS. BÙI HẢI PHONG

HÀ NỘI – 2024

LỜI CAM ĐOAN

Em cam đoan rằng tất cả các thông tin được cung cấp ở đây là dựa trên kiến thức và hiểu biết của em cho đến thời điểm cập nhật gần nhất vào tháng 5 năm 2024. Em cam đoan rằng thông tin của em là hoàn toàn chính xác hay đầy đủ đối với tất cả các trường hợp và tình huống. Đối với thông tin cụ thể hoặc tư vấn chi tiết, luôn luôn nên tìm kiếm ý kiến từ các chuyên gia hoặc nguồn thông tin đáng tin cậy khác.

Ngày 17 tháng 06 năm 2024

Sinh viên ký tên

LỜI CẢM ƠN

Để hoàn thành chuyên đồ án tốt nghiệp này, trước hết em chân thành cảm ơn các cá nhân và tổ chức đã tạo điều kiện hỗ trợ, giúp đỡ em trong suốt quá trình học tập và nghiên cứu đề tài này. Trong suốt thời gian từ khi bắt đầu học tập đến nay, em đã nhận được rất nhiều sự quan tâm, giúp đỡ từ quý thầy, cô giáo trong khoa công nghệ thông tin – đại học Đại Nam đã luôn tận tình chỉ bảo, truyền đạt cho em những kiến thức quý báu trong suốt thời gian học ở trường.

Đặc biệt, em xin gửi đến thầy TS. Bùi Hải Phong người đã tận tình hướng dẫn, giúp đỡ để em có thể hoàn thành tốt đồ án tốt nghiệp trong thời gian qua.

Do bản thân còn những hạn chế nhất định về chuyên môn và thời gian nên sẽ khó tránh khỏi những sai sót trong quá trình thực hiện. Em mong nhận được sự góp ý của các thầy cô trong khoa để bài báo cáo của em được hoàn thiện tốt hơn.

Em xin trân thành cảm ơn!

LỜI NÓI ĐẦU

Trong thời đại công nghệ ngày nay, sự phát triển của trí tuệ nhân tạo và học sâu đã mở ra những cánh cửa mới cho việc nghiên cứu và ứng dụng trong nhiều lĩnh vực. Trong lĩnh vực thị giác máy tính, việc nhận dạng và phân loại đối tượng từ hình ảnh đã trở thành một trong những thách thức quan trọng. Trong ngành nông nghiệp và môi trường, việc nhận biết loài cây thông qua lá và hoa có thể mang lại nhiều lợi ích quan trọng, từ giám sát môi trường đến nâng cao hiệu suất nông nghiệp.

Trong bối cảnh này, đề tài của em nhằm mục đích nghiên cứu và phát triển một hệ thống nhận dạng hoa và lá sử dụng các phương pháp học sâu. Bằng cách kết hợp kiến thức về thị giác máy tính và các mô hình học sâu như mạng nơ-ron tích chập (CNN), em hy vọng xây dựng được một hệ thống có khả năng nhận diện và phân loại các loại hoa và lá với độ chính xác cao.

Đề tài này không chỉ mang lại những tri thức mới mẻ về ứng dụng của học sâu trong lĩnh vực thị giác máy tính mà còn có tiềm năng ứng dụng rộng rãi trong các lĩnh vực như nông nghiệp, môi trường, và y học. Em hy vọng rằng nghiên cứu này sẽ đóng góp vào sự phát triển của công nghệ thông qua việc áp dụng những phương pháp mới và tiên tiến vào thực tiễn.

Báo cáo gồm ba phần chính, được tổ chức như sau:

1. Chương 1: “Tổng quan về đề tài” cung cấp một cái nhìn tổng quan đề tài cũng như phâm vi nghiên cứu và bài toán mà em đặt ra.
2. Chương 2: “Cơ sở lý thuyết” phần này trình bày những lý thuyết mà em sẽ sử dụng để xay dựng bài toán của mình cũng như các bước làm bài và một số mô hình CNN nổi tiếng.
3. Chương 3: “Triển khai và đánh giá hệ thống” chương này em sẽ bắt đầu triển khai các bước để làm bài toán như tiền xử lí dữ liệu và xây dựng

mô hình và đánh giá xem mô hình của mình tỉ lệ dự đoán đúng là bao nhiêu phần trăm.

Sau thời gian thực hiện, các mục tiêu cơ bản được đặt ra khi thực hiện đề tài này đã đạt được. Tuy nhiên, do vốn kiến thức và kinh nghiệm còn hạn chế nên bài báo cáo còn nhiều thiếu sót. Em rất mong nhận được sự góp ý từ quý thầy cô.

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG

DANH MỤC KÍ HIỆU VÀ VIẾT TẮT

Từ viết tắt	Nghĩa tiếng Anh	Nghĩa tiếng Việt
ANN	Artificial Neural Network	Mạng nơ-ron nhân tạo
AI	Artificial Intelligence	Trí tuệ nhân tạo
AR	Augmented Reality	Thực tế tăng cường
CNN	Convolutional Neural Network	Mạng nơ-ron tích chập
LSTM	Long Short-Term Memory	Bộ nhớ dài ngắn hạn
OCR	Optical Character Recognition	Nhận dạng ký tự quang học
RNN	Recurrent Neural Network	Mạng nơ-ron hồi quy RNN
VR	Virtual Reality	Thực tế ảo

CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI

1.1. Tính cấp thiết của đề tài

Ứng dụng thực tế: Nhận dạng hoa và lá là một vấn đề quan trọng trong nhiều lĩnh vực, chẳng hạn như nghiên cứu môi trường, nông nghiệp, và sinh học. Việc xây dựng hệ thống tự động nhận dạng có thể giúp nhận biết và phân loại hoa và lá một cách nhanh chóng và chính xác, hỗ trợ trong quan trắc và quản lý tài nguyên thiên nhiên.

Tiềm năng ứng dụng: Hệ thống nhận dạng hoa và lá có thể được áp dụng rộng rãi trong các ứng dụng thực tế, bao gồm hệ thống tự động nhận dạng cây cối, hệ thống giám sát môi trường, ứng dụng di động cho việc nhận dạng hoa và lá, và nhiều ứng dụng khác liên quan đến xử lý ảnh và trí tuệ nhân tạo.

1.2. Mục đích nghiên cứu

Xây dựng mô hình nhận dạng chính xác: Một trong những mục đích chính của nghiên cứu này là tạo ra một mô hình nhận dạng hoa và lá chính xác và tin cậy. Mô hình này có thể được huấn luyện để phân loại và nhận dạng các loài hoa và lá khác nhau dựa trên hình ảnh đầu vào. Mục tiêu là cải thiện độ chính xác và hiệu suất của mô hình để đáp ứng yêu cầu của các ứng dụng thực tế.

Nghiên cứu và phát triển các phương pháp và thuật toán: Nghiên cứu về nhận dạng hoa và lá sử dụng học sâu đóng góp vào việc phát triển các phương pháp và thuật toán tiên tiến trong lĩnh vực xử lý ảnh và trí tuệ nhân tạo. Điều này có thể bao gồm việc tối ưu hóa các mô hình học sâu, tăng cường khả năng phân loại và nhận dạng, xử lý dữ liệu không đồng nhất và đa dạng, và khám phá các phương pháp mới để tăng cường hiệu suất và độ chính xác của hệ thống.

Ứng dụng trong các lĩnh vực thực tế: Mục đích của nghiên cứu này cũng là áp dụng hệ thống nhận dạng hoa và lá vào các lĩnh vực thực tế, bao gồm nghiên cứu môi trường, quản lý tài nguyên thiên nhiên, nông nghiệp, công nghệ thông

tin, và ứng dụng di động. Các ứng dụng này có thể giúp tăng cường khả năng quan trắc, giám sát và quản lý trong các lĩnh vực liên quan đến hoa và lá.

Đóng góp vào tri thức và hiểu biết: Nghiên cứu này có thể đóng góp vào tri thức và hiểu biết về các loài hoa và lá khác nhau. Việc xây dựng một cơ sở dữ liệu về nhận dạng hoa và lá và chia sẻ kết quả nghiên cứu có thể giúp mở rộng hiểu biết về sự đa dạng sinh học và các loài hoa và lá trên toàn cầu.

1.3. Phạm vi nghiên cứu

Nhận dạng loại hoa và lá: Nghiên cứu tập trung vào việc phân loại và nhận dạng các loại hoa và lá khác nhau. Điều này bao gồm việc xây dựng mô hình học sâu để phân loại các loại hoa và lá vào các nhóm, loài hoặc họ riêng biệt.

Phân biệt giữa các biến thể và đặc điểm của hoa và lá: Nghiên cứu có thể tập trung vào việc phân biệt giữa các biến thể và đặc điểm khác nhau của hoa và lá trong cùng một loại cây hoặc loài hoa. Điều này đòi hỏi mô hình học sâu có khả năng nhận dạng và phân loại các biến thể và đặc điểm riêng biệt của hoa và lá.

Xử lý ảnh và tiền xử lý dữ liệu: Nghiên cứu có thể tập trung vào các phương pháp và kỹ thuật xử lý ảnh và tiền xử lý dữ liệu để làm sạch và chuẩn bị dữ liệu huấn luyện cho mô hình học sâu. Điều này có thể bao gồm thay đổi kích thước ảnh, áp dụng các phép biến đổi hình học, loại bỏ nhiễu và xử lý dữ liệu không đồng nhất.

Tối ưu hóa mô hình học sâu: Nghiên cứu có thể tập trung vào việc tối ưu hóa mô hình học sâu để đạt được hiệu suất tốt nhất trong việc nhận dạng hoa và lá. Điều này có thể bao gồm việc điều chỉnh siêu tham số, sử dụng các kỹ thuật tăng cường dữ liệu, áp dụng kỹ thuật chuyển giao học tập và khám phá các cấu trúc mô hình mới để cải thiện kết quả.

Ứng dụng thực tiễn: Nghiên cứu có thể tập trung vào việc áp dụng hệ thống nhận dạng hoa và lá vào các ứng dụng thực tiễn như nghiên cứu môi trường, quản lý tài nguyên thiên nhiên, nông nghiệp và các ứng dụng di động. Điều này đòi hỏi

nghiên cứu liên quan đến tích hợp và triển khai hệ thống nhận dạng hoa và lá trong môi trường thực tế

1.4. Phương pháp nghiên cứu:

Tìm hiểu về lĩnh vực nghiên cứu: Trước khi bắt đầu nghiên cứu, nắm vững kiến thức cơ bản về nhận dạng hoa và lá, học sâu và thuật toán CNN. Tìm hiểu về các phương pháp và công trình nghiên cứu đã được công bố liên quan đến đề tài này để hiểu rõ hơn về tiến bộ và thách thức trong lĩnh vực này.

Thu thập dữ liệu: Tiến hành thu thập dữ liệu hình ảnh của hoa và lá từ các nguồn đáng tin cậy. Dữ liệu nên bao gồm nhiều loại hoa và lá khác nhau, có đủ biến thể và đặc điểm để đảm bảo tính đa dạng của tập dữ liệu.

Tiền xử lý dữ liệu: Thực hiện các bước tiền xử lý dữ liệu để chuẩn bị dữ liệu cho việc huấn luyện mô hình. Điều này có thể bao gồm chuyển đổi kích thước ảnh về kích thước thích hợp, chuẩn hóa giá trị pixel và xử lý nhiễu hoặc biến dạng trong ảnh.

Xây dựng mô hình CNN: Xây dựng một mô hình CNN để huấn luyện và nhận dạng hoa và lá. Mô hình CNN có thể bao gồm các lớp tích chập, lớp gộp, lớp kết nối đầy đủ và lớp đầu ra. Thiết kế kiến trúc của mô hình phải được tùy chỉnh để đáp ứng yêu cầu cụ thể của bài toán nhận dạng hoa và lá.

Huấn luyện và đánh giá mô hình: Sử dụng tập dữ liệu đã chuẩn bị, huấn luyện mô hình CNN bằng việc điều chỉnh các tham số của mô hình và sử dụng thuật toán tối ưu hóa như Gradient Descent. Đánh giá hiệu suất của mô hình trên tập dữ liệu kiểm tra bằng các độ đo như độ chính xác, độ phủ, độ F1-score.

Nâng cấp và tối ưu hóa mô hình: Dựa trên kết quả đánh giá, tiến hành nâng cấp và tối ưu hóa mô hình để cải thiện hiệu suất nhận dạng hoa và lá. Điều này có thể bao gồm thử nghiệm các kiến trúc mô hình khác nhau, điều chỉnh siêu tham số và sử dụng các kỹ thuật tăng cường dữ liệu.

1.5. Mô tả bài toán

Trong lĩnh vực thị giác máy tính, việc nhận dạng và phân loại các vật thể trong hình ảnh được xem là một trong những bài toán cơ bản nhất, mở ra cánh cửa cho nhiều ứng dụng thú vị khác nhau. Mặc dù đã tồn tại từ hàng thế kỷ, nhưng vẫn còn nhiều thách thức mà con người đối mặt khi cố gắng giải quyết bài toán này một cách toàn diện.

Sự đa dạng trong góc nhìn và vị trí của các vật thể trong hình ảnh là một trong những khó khăn lớn nhất. Cùng một vật thể có thể xuất hiện ở nhiều góc nhìn và khoảng cách khác nhau, tạo ra sự biến đổi đáng kể trong hình ảnh. Điều này đặt ra thách thức lớn trong việc huấn luyện mô hình để nhận diện các biến thể của cùng một vật thể.

Ngoài ra, sự đa dạng về kích thước và tỉ lệ của các vật thể trong hình ảnh cũng là một thách thức khác. Không có cách nào để đo lường kích thước thực tế của vật thể từ một bức ảnh hai chiều, do đó, việc đảm bảo mô hình có khả năng nhận diện vật thể ở các tỉ lệ khác nhau là cần thiết.

Sự biến đổi trong ánh sáng cũng là một yếu tố quan trọng. Ánh sáng có thể thay đổi tùy thuộc vào điều kiện môi trường và cách thiết lập của máy ảnh, gây ra sự biến đổi đáng kể trong cường độ và màu sắc của vật thể trong hình ảnh.

Cuối cùng, sự phức tạp của nền ảnh và sự chồng chéo giữa các vật thể trong hình ảnh cũng làm cho việc nhận dạng trở nên khó khăn. Đôi khi, các vật thể có thể bị che khuất hoặc phần lớn của chúng bị che lấp bởi các vật thể khác hoặc phần nền, tạo ra những thách thức đối với việc nhận dạng.

Bài toán nhận dạng và phân loại lá và hoa cũng mang trong mình những thách thức đặc biệt do sự đa dạng và biến đổi không ngừng của chúng theo từng mùa, vùng miền và điều kiện môi trường. Mỗi loại lá và hoa lại có những đặc điểm riêng biệt về hình dáng, màu sắc và kết cấu, tạo nên một thế giới thực vật phong phú và đa dạng mà hệ thống nhận dạng phải đối mặt.

Với số lượng lớn các loại lá và hoa trên thế giới, việc xây dựng một hệ thống nhận dạng chính xác và hiệu quả trở nên cực kỳ phức tạp. Mỗi loại lá và hoa lại có thể có nhiều biến thể khác nhau tùy thuộc vào điều kiện môi trường và quá trình phát triển của chúng. Từ hình dáng, màu sắc cho đến các đặc điểm về cấu trúc và vị trí trên cây, tất cả đều đóng góp vào độ phong phú và đa dạng của thế giới thực vật.

Thêm vào đó, sự biến đổi của lá và hoa theo chu kỳ phát triển và thời tiết cũng làm tăng thêm độ phức tạp cho bài toán nhận dạng. Cùng một loại lá hoặc hoa có thể có nhiều hình dạng và màu sắc khác nhau tùy thuộc vào giai đoạn phát triển của nó, đặt ra yêu cầu cao cho hệ thống nhận dạng trong việc nhận biết và phân loại các biến thể của cùng một loại thực vật.

Hình 1.1 Nhận diện các loài hoa

1.6. Tiểu kết chương 1

Trong chương này, em đã đề cập đến tính cấp thiết của việc nhận dạng hoa và lá trong ứng dụng thực tế, cũng như tiềm năng ứng dụng và mục đích nghiên cứu của đề tài. Phạm vi nghiên cứu đã được xác định rõ ràng, từ việc nhận dạng

loại hoa và lá đến việc xử lý ảnh và áp dụng vào các ứng dụng thực tiễn. Bằng cách sử dụng phương pháp nghiên cứu có hệ thống và chặt chẽ, em đã đề ra cơ sở cho việc giải quyết bài toán nhận dạng và phân loại hoa và lá một cách hiệu quả. Cuối cùng, nhận thức về những thách thức đặc biệt của bài toán đã được thể hiện, và mục tiêu cuối cùng của nghiên cứu này là xây dựng một hệ thống nhận dạng hoa và lá chính xác và đáng tin cậy, mang lại đóng góp đáng kể cho lĩnh vực thị giác máy tính và ứng dụng thực tế.

CHƯƠNG 2: CƠ SỞ LÍ THUYẾT

2.1. Phương pháp tiếp cận bài toán

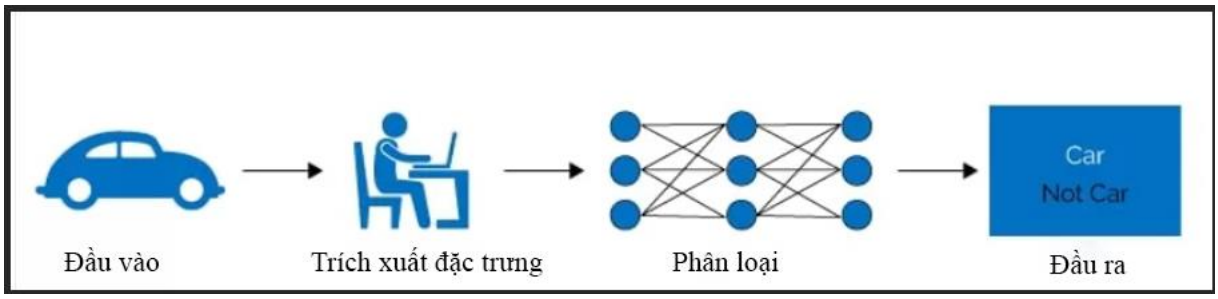
2.1.1. Phương pháp học máy truyền thống

Học máy là một lĩnh vực quan trọng trong trí tuệ nhân tạo (AI), tập trung vào việc phát triển các kỹ thuật để máy tính tự học từ dữ liệu một cách tự động, không cần phải được lập trình cụ thể. Ý tưởng cơ bản của học máy là xây dựng và huấn luyện các mô hình hoặc thuật toán dựa trên các mẫu dữ liệu để thực hiện các nhiệm vụ như dự đoán, phân loại, nhận dạng hoặc điều chỉnh hành vi của máy tính một cách tự động và liên tục dựa trên kinh nghiệm.

Các phương pháp trong học máy có thể chia thành hai loại chính: học có giám sát và học không giám sát. Trong học có giám sát, mô hình được huấn luyện trên các cặp dữ liệu đầu vào và đầu ra đã được gán nhãn. Trong khi đó, trong học không giám sát, mô hình phải tự động tìm ra cấu trúc hoặc thông tin bổ sung từ dữ liệu không có nhãn.

Học máy có ứng dụng rộng rãi trong nhiều lĩnh vực như nhận dạng hình ảnh, xử lý ngôn ngữ tự nhiên, dự đoán tài chính, y học, tự lái ô tô và nhiều lĩnh vực khác. Đối với nhiều ứng dụng hiện đại, học máy đã trở thành công cụ quan trọng để trích xuất thông tin từ dữ liệu phức tạp và đưa ra các quyết định thông minh.

Mô hình hoạt động



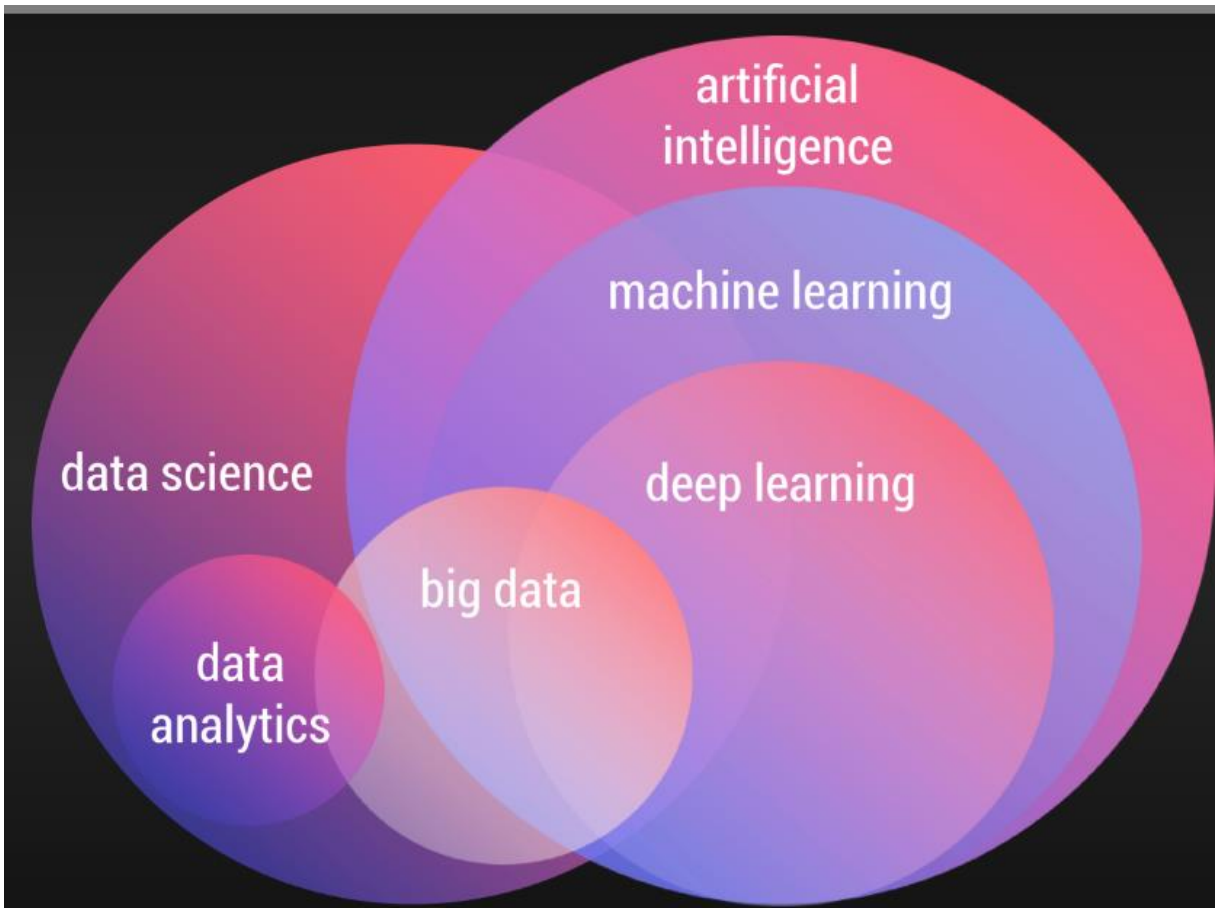
Hình 2.1 Mô hình hoạt động học máy truyền thống [10]

2.1.2 Phương pháp học sâu

Học sâu (Deep Learning) là một phương pháp trong lĩnh vực trí tuệ nhân tạo (AI) mà các mô hình máy tính sử dụng nhiều lớp (hay còn gọi là "độ sâu") của các đơn vị xử lý để học và hiểu dữ liệu. Các đơn vị xử lý này thường được tổ chức thành các kiến trúc mạng nơ-ron nhân tạo (Artificial Neural Networks - ANN) với hàng trăm hoặc thậm chí hàng ngàn lớp, cho phép mô hình học được các biểu diễn phức tạp và trích xuất các đặc trưng sâu trong dữ liệu.

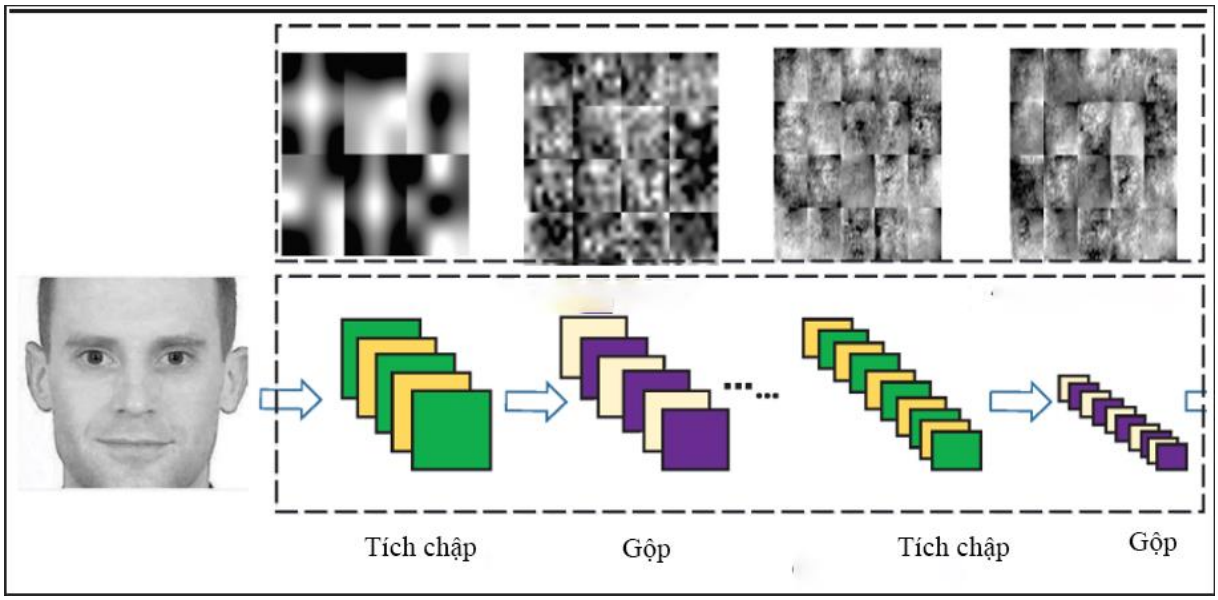
Trong học sâu, việc huấn luyện mô hình thường được thực hiện bằng cách sử dụng một phương pháp gọi là "lan truyền ngược" (backpropagation), trong đó mô hình cố gắng điều chỉnh các trọng số của các liên kết giữa các nơ-ron dựa trên sự khác biệt giữa dự đoán và kết quả thực tế của mỗi lần huấn luyện.

Học sâu đã mang lại nhiều tiến bộ đáng kể trong nhiều lĩnh vực như xử lý ngôn ngữ tự nhiên, nhận dạng hình ảnh, nhận dạng giọng nói, tự động lái xe và nhiều ứng dụng khác. Các kiến trúc mạng sâu như mạng nơ-ron con người (Convolutional Neural Networks - CNNs) và mạng nơ-ron dài hạn khớp (Long Short-Term Memory - LSTM) đã chứng minh khả năng tuyệt vời trong việc giải quyết các bài toán phức tạp và xử lý dữ liệu lớn.



Hình 2.2 Mối quan hệ học sâu và các lĩnh vực có liên quan [9]

Các mạng huấn luyện theo phương pháp học sâu 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 2.3 Mức đồ trừu tượng tăng dần qua các tầng học sâu [11]

Trong một mạng học sâu được áp dụng cho việc nhận dạng ảnh, các lớp ban đầu thường thực hiện các nhiệm vụ cơ bản như tìm kiếm các đặc điểm đơn giản như đường thẳng, đường cong hoặc đốm màu trong ảnh gốc. Những thông tin này sau đó sẽ được chuyển tiếp làm đầu vào cho các lớp sau, nơi chúng sẽ được sử dụng để nhận diện các thành phần của vật thể trong ảnh, điều này là một nhiệm vụ khó hơn.

Các lớp ở giữa của mạng sẽ tiếp tục xử lý thông tin từ những đặc điểm cơ bản này và dần dần học cách kết hợp chúng lại với nhau để nhận diện ra các phần của vật thể. Cuối cùng, các lớp cao nhất trong mạng sẽ phải thực hiện nhiệm vụ phát hiện và định vị vật thể trong ảnh, bằng cách kết hợp thông tin từ các thành phần đã được nhận dạng trước đó. Quá trình này giúp mạng học sâu hiểu được cấu trúc và đặc điểm của các đối tượng trong ảnh, từ những đặc trưng cơ bản nhất cho đến những yếu tố phức tạp hơn.

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.

Tuy có nhiều điểm ưu việt trong khả năng huấn luyện máy tính cho các bài toán phức tạp. Học sâu 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 học sâu 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. Tuy nhiên, nhờ các phương pháp học tập chuyển giao, hiện nay điểm hạn chế lớn nhất này đã không còn là vấn đề quá nghiêm trọng như trước, điều này sẽ được trình bày cụ thể trong các chương sau.

Ngoài giới hạn về kích thước dữ liệu đầu vào, học sâu cũng chưa đủ thông minh để nhận biết và hiểu được các tình huống phức tạp như con người.

Như đã được trình bày trong phần mở đầu, mục tiêu của đồ án tốt nghiệp là nghiên cứu và áp dụng một mô hình học sâu vào bài toán nhận dạng và phân loại hoa và lá. Lựa chọn học sâu làm phương pháp chính được đề xuất do khả năng mạnh mẽ vượt trội của nó so với các phương pháp học máy truyền thống, đặc biệt là khi áp dụng vào các bài toán nhận dạng vật thể, trong trường hợp này là nhận dạng hoa và lá.

Phương pháp deep learning:

Những mô hình deep learning được tạo ra bằng nhiều phương pháp khác nhau. Có thể kể đến như Learning rate decay, Transfer learning, Training from scratch, Dropout.

❖ Learning rate decay

Learning rate decay là một siêu tham số quan trọng nhất của quá trình huấn luyện. Đây là một phương pháp điều chỉnh learning rate qua mỗi bước update các tham số của mô hình. Tỷ lệ learning rate quá cao sẽ dẫn tới quá trình huấn luyện

không ổn định. Tỷ lệ learning rate quá thấp lại khiến quá trình huấn luyện kéo dài và tiềm ẩn những khó khăn.

❖ Transfer learning

Transfer learning là một kĩ thuật chuyển giao tri thức giữa các mô hình. Tức là một mô hình có khả năng tận dụng lại những tri thức được huấn luyện trước đó để thực hiện các tác vụ mới với khả năng phân loại cụ thể hơn. Phương pháp này sẽ yêu cầu ít dữ liệu hơn và giảm thời gian tính toán so với những phương pháp khác.

❖ Training from scratch

Training from scratch yêu cầu các nhà phát triển cần thu thập một lượng dữ liệu lớn có gắn nhãn. Bên cạnh đó cần phải thiết lập và định cấu hình cho mạng để nó có thể tìm hiểu các tính năng và mô hình. Với các ứng dụng mới hay ứng dụng có danh mục đầu ra lớn rất thích hợp với training from scratch. Phương pháp này chưa phổ biến bởi quá trình huấn luyện kéo dài hơn những phương pháp khác.

❖ Dropout

Dropout là một phương pháp loại bỏ các nút mạng một cách ngẫu nhiên trong quá trình huấn luyện. Bằng cách thức này đã giúp cải thiện hiệu suất của mạng nơ-ron khi thực hiện các nhiệm vụ học tập có giám sát như phân loại tài liệu, nhận dạng giọng nói, tính toán…

2.1.3. Lợi thế của Deep learning so vơi Machine learning

❖ Tự Động Hóa Các Tính Năng

Deep learning có khả năng tự động hóa việc tạo ra các tính năng từ dữ liệu, mà không cần sự can thiệp của con người. Các thuật toán deep learning có thể học và tạo ra các đặc trưng mới từ dữ liệu đầu vào, giúp giảm thiểu công đoạn tiền xử lý và tăng cường khả năng phân loại. Điều này làm cho việc triển khai các ứng dụng và công nghệ dựa trên học sâu trở nên nhanh chóng và ổn định hơn, với độ chính xác cao.

❖ Tương Thích Tốt Với Dữ Liệu Phi Cấu Trúc

Deep learning thích hợp với việc xử lý dữ liệu phi cấu trúc như hình ảnh, văn bản và âm thanh - các loại dữ liệu phổ biến trong doanh nghiệp ngày nay. So với machine learning cổ điển, deep learning có khả năng phân tích và rút trích thông tin từ dữ liệu phi cấu trúc hiệu quả hơn, mang lại các lợi ích to lớn cho việc tối ưu hóa chức năng kinh doanh từ bán hàng đến tiếp thị và tài chính.

❖ Khả Năng Tự Học Tốt Hơn

Deep learning có khả năng học và thích ứng tốt với các tác vụ tính toán phức tạp và tạo ra dự đoán chính xác. Với sự linh hoạt của các lớp nơ-ron và khả năng tự học từ dữ liệu, mạng neural sâu có thể thực hiện các hoạt động đồng thời và học từ các lỗi của chính mình. Điều này làm cho deep learning trở thành công cụ mạnh mẽ trong việc xử lý hình ảnh, âm thanh và video, đặc biệt là khi liên quan đến dữ liệu phi cấu trúc.

2.2. Mạng nơ-ron tích chập

Mạng nơ-ron tích chập (Convolutional Neural Network - CNN) đã ra đời như một phần của sự phát triển đồng thời của lĩnh vực trí tuệ nhân tạo (AI) và lĩnh vực thị giác máy tính. CNN được phát triển để giải quyết các bài toán nhận dạng hình ảnh và xử lý ảnh một cách hiệu quả.

Sự ra đời của CNN có thể được liên kết chặt chẽ với nhu cầu của thị trường và sự phát triển công nghệ. Trong những năm đầu của thế kỷ 21, với sự gia tăng về khả năng tính toán của máy tính và sự phát triển mạnh mẽ của Internet, có một nhu cầu ngày càng tăng về việc tự động hóa xử lý và hiểu thông tin từ hình ảnh. Điều này đã tạo ra động lực mạnh mẽ cho việc phát triển các mô hình AI mạnh mẽ hơn, đặc biệt là trong lĩnh vực nhận dạng hình ảnh.

CNN được đưa ra vào những năm đầu của thế kỷ 21 và nhanh chóng trở thành một trong những công cụ quan trọng nhất trong lĩnh vực thị giác máy tính. Các kiến trúc CNN sử dụng các lớp tích chập để trích xuất các đặc trưng từ hình

ảnh, kết hợp với các lớp tổng hợp để giảm kích thước của dữ liệu và các lớp kết nối đầy đủ để phân loại hình ảnh.

CNN đã mang lại nhiều tiến bộ đáng kể trong nhận dạng hình ảnh, từ việc nhận dạng khuôn mặt đến nhận dạng vật thể và phân loại hình ảnh tự nhiên, và hiện đang được sử dụng rộng rãi trong nhiều ứng dụng thực tế từ công nghiệp đến y học và đến xe tự lái.

❖ **Kiến trúc của mạng Convolutional Neural Network**

Mạng CNN là tập hợp những Convolutional layer xếp chồng lên nhau, đồng thời mạng sử dụng những hàm như ReLU và Tanh để kích hoạt các trọng số trong các node. Các lớp này sau khi qua các hàm activation sẽ có trọng số trong những node và có thể tạo ra những thông tin trừu tượng hơn đến với các lớp kế tiếp trong mạng.

Mạng này có tính kết hợp cà tính bất biến. Tức là, nếu cùng một đối tượng mà sử dụng chiếu theo các góc độ khác nhau thì sẽ có ảnh hưởng đến độ chính xác. Với dịch chuyển, co giãn hay quay ma trận ảnh thì lớp Pooling sẽ được dùng để hỗ trợ làm bất biến các tính chất này. Chính vì vậy mà mạng nơ ron này sẽ đưa ra những kết quả có độ chính xác tương ứng với từng mô hình.

Trong đó, lớp Pooling sẽ có khả năng tạo tính bất biến với phép dịch chuyển, co giãn và quay. Còn tính kết hợp cục bộ sẽ cho thấy những cấp độ biểu diễn, dữ liệu từ thấp đến cao với mức trừu tượng thông qua Convolution từ filter. Mạng CNN có những lớp liên kết nhau dựa vào cơ chế Convolution.

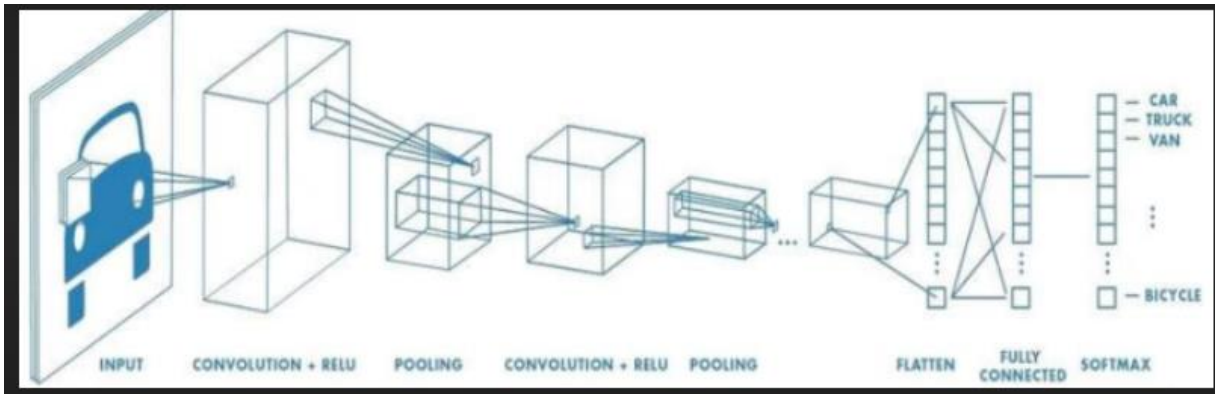
Các lớp tiếp theo sẽ là kết quả từ những lớp trước đó, vì vậy mà bạn sẽ có những liên kết cục bộ phù hợp nhất. Trong quá trình huấn luyện mạng, mạng nơ ron này sẽ tự học hỏi những giá trị thông qua filter layer dựa theo cách thức mà bạn thực hiện.

Cấu trúc cơ bản của một mô hình mạng CNN thường bao gồm 3 phần chính bao gồm:

Trường cục bộ/ Local receptive field: Lớp này sử dụng để tách lọc dữ liệu, thông tin hình ảnh để từ đó có thể lựa chọn các vùng có giá trị sử dụng hiệu quả cao nhất.

Trọng số chia sẻ/ Shared weights and bias: Lớp này hỗ trợ làm giảm các tham số đến mức tối thiểu trong mạng CNN. Trong từng lớp convolution sẽ chứa các feature map riêng và từng feature thì sẽ có khả năng phát hiện một vài feature trong hình ảnh.

Lớp tổng hợp/ Pooling layer: Đây là lớp cuối cùng và sử dụng để làm đơn giản các thông tin output. Tức là, sau khi tính toán xong và quét qua các layer trong mạng thì pooling layer sẽ được dùng để lược bỏ các thông tin không hữu ích. Từ đó cho ra kết quả theo kỳ vọng người dùng



Hình 2.4 Mô hình hoạt động của thuật toán CNN [7]

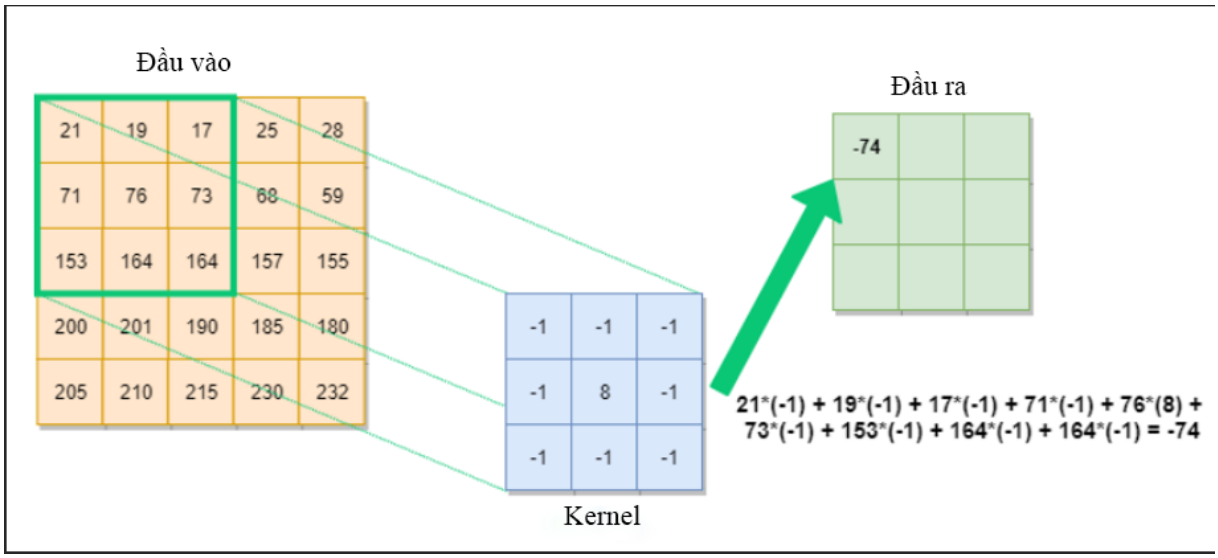
❖ **Lớp tích chập(Convolutional layer)**

Lớp tích chập là một thành phần quan trọng trong mạng nơ-ron tích chập (CNN). Lớp này chịu trách nhiệm chính trong việc trích xuất các đặc trưng từ dữ liệu hình ảnh. Mục tiêu của lớp tích chập là chạy qua ảnh đầu vào và tạo ra các "bản đồ đặc trưng" bằng cách áp dụng các bộ lọc (hay còn gọi là "kernel" hoặc "filter") lên từng phần của ảnh.

Mỗi bộ lọc sẽ quét qua từng phần của ảnh và tính tổng các tích của các giá trị pixel trong phần đó, tạo ra một giá trị mới trên bản đồ đặc trưng. Bản đồ đặc

trưng này giúp chúng ta biết được sự hiện diện của các đặc trưng cụ thể như cạnh, góc, hoặc các đặc điểm phức tạp hơn trong ảnh.

Lớp tích chập không chỉ giúp giảm chiều dữ liệu mà còn giúp mạng nơ-ron học được các đặc trưng cấp cao từ dữ liệu đầu vào, giúp cải thiện khả năng nhận dạng và phân loại của mạng. Đặc biệt, việc chia sẻ trọng số giữa các vùng của ảnh giúp giảm lượng tham số cần được học, làm cho mô hình trở nên hiệu quả và dễ huấn luyện hơn.



Hình 2.5 Ví dụ bộ lọc tích chập được sử dụng trên ma trận điểm ảnh [6]

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: $o = (i+2*p-k\ s) + 1$

Trong đó:

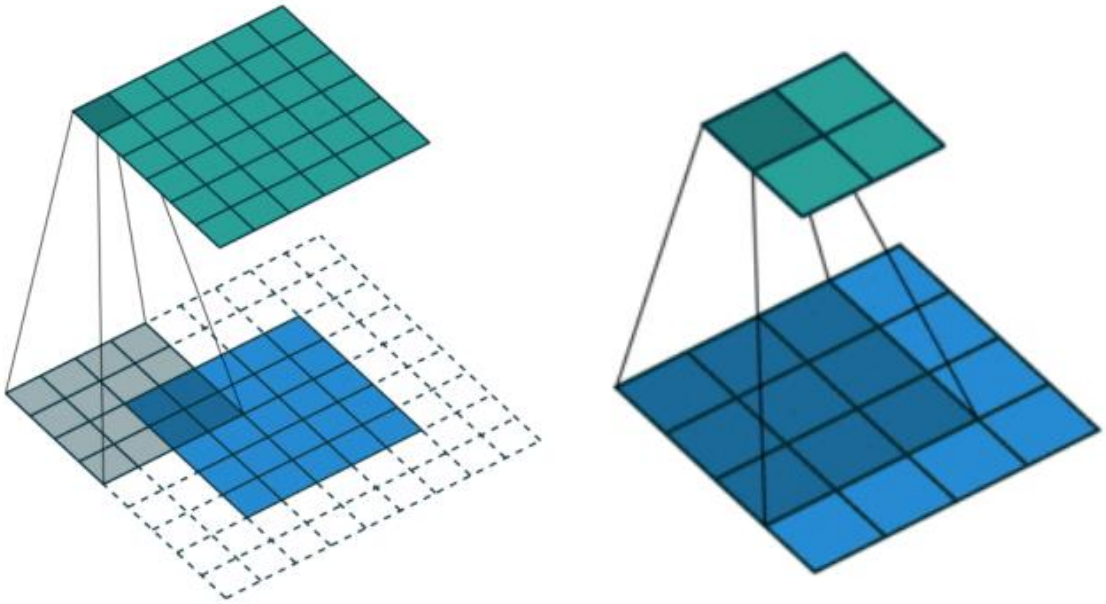
- o: kích thước ảnh đầu ra

- 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 2.6 Trường hợp thêm/không thêm viền trắng vào ảnh khi tích chập [2]

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.

❖ **Lớp kích hoạt phi tuyến ReLU**

Lớp kích hoạt phi tuyến ReLU (Rectified Linear Unit) là một trong những lớp kích hoạt phổ biến được sử dụng trong mạng nơ-ron tích chập (CNN) và các mô hình học sâu khác. ReLU là một hàm phi tuyến tính đơn giản nhưng rất hiệu quả, được biểu diễn bởi công thức:

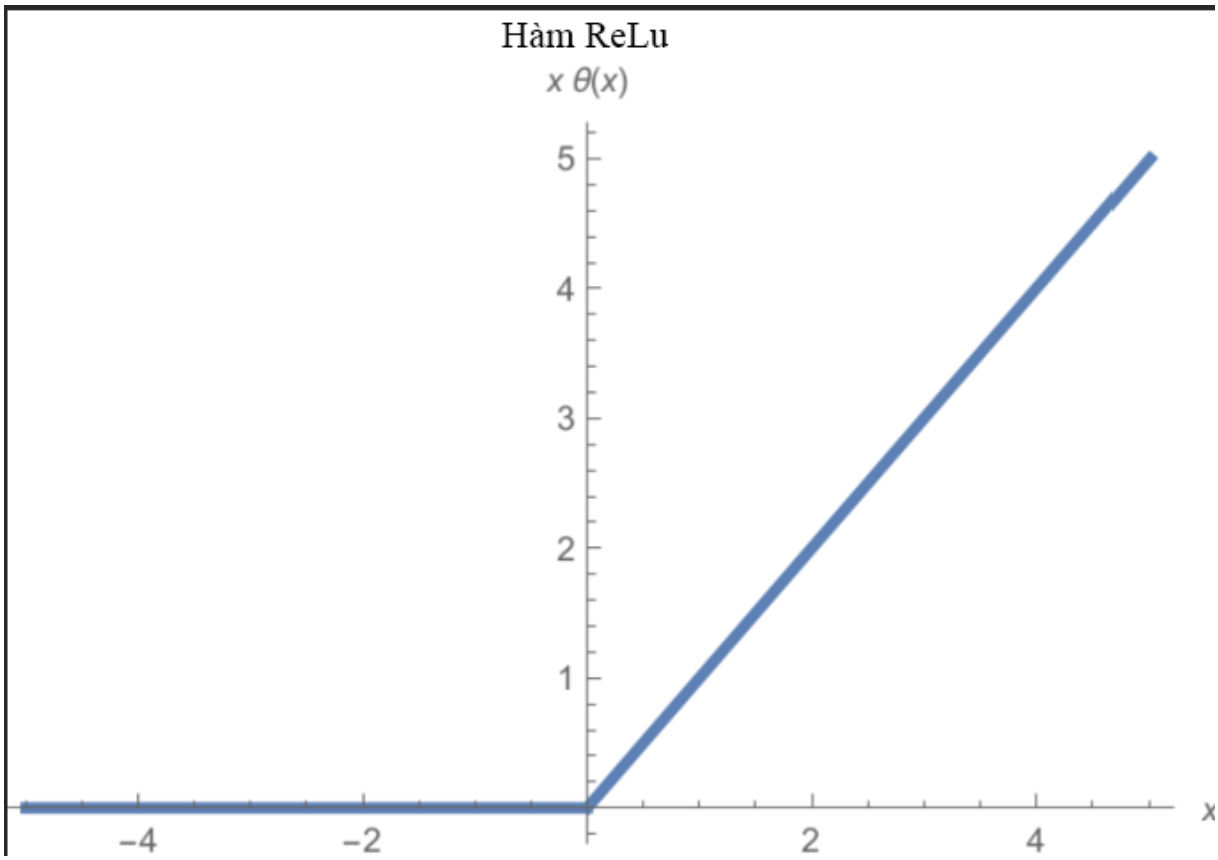
$f(x)=\max(0,x)$f(x)=max(0,x)

Trong đó:

- x là giá trị đầu vào.
- f(x) là giá trị đầu ra sau khi áp dụng hàm ReLU.

ReLu thực hiện chức năng chuyển đổi phi tuyến tính bằng cách giữ nguyên giá trị dương và chuyển đổi tất cả các giá trị âm thành 0. Điều này giúp cải thiện khả năng học của mạng bằng cách tạo ra sự không tuyến tính trong dữ liệu, giúp mô hình có khả năng học được các biểu diễn phức tạp hơn.

Sự đơn giản và tính hiệu quả của ReLU đã khiến nó trở thành lựa chọn phổ biến cho các lớp kích hoạt trong các mạng nơ-ron sâu. Đồng thời, việc tính toán ReLU cũng nhanh chóng và không yêu cầu nhiều tài nguyên tính toán, giúp tăng tốc độ huấn luyện của mạng.



Hình 2.7 Hàm Relu [3]

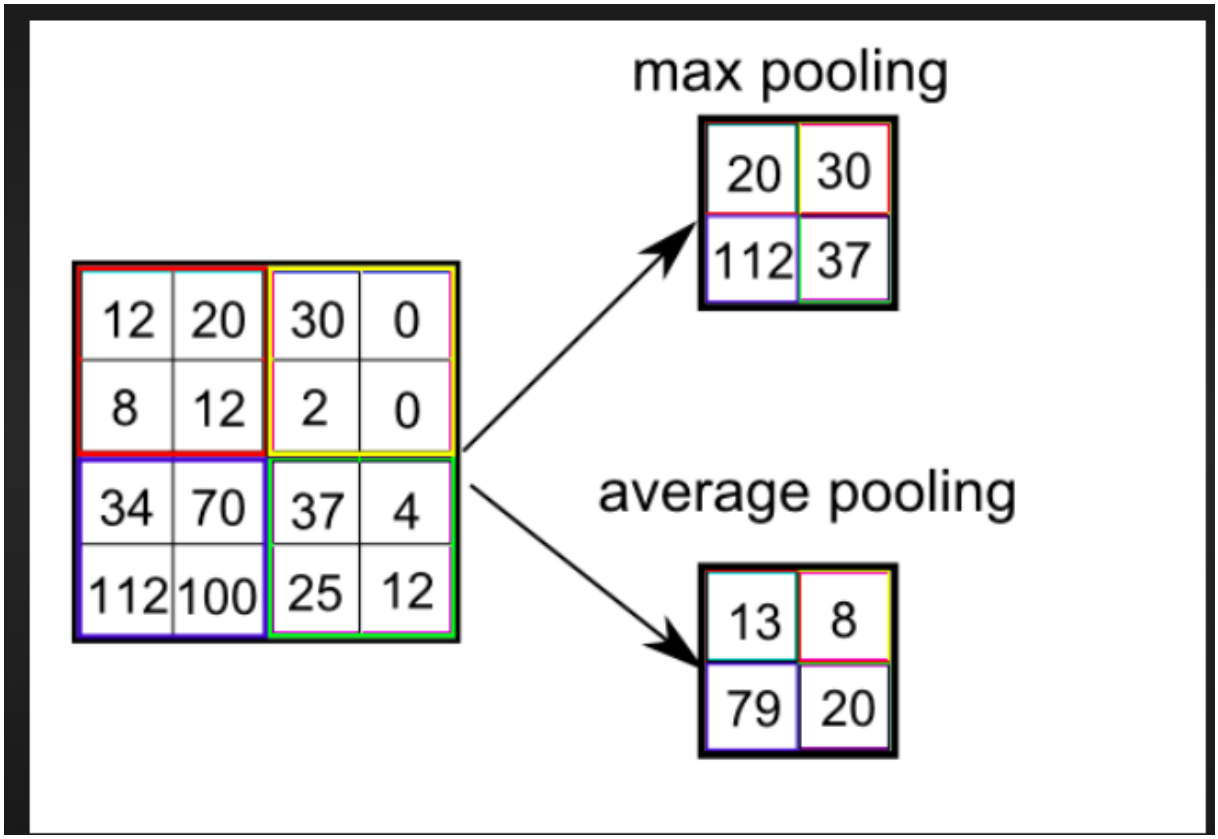
- **Lớp lấy mẫu MaxPooling và AveragePooling**

Là một thành phần quan trọng trong mạng nơ-ron tích chập (CNN). Chức năng chính của lớp lấy mẫu là giảm kích thước của dữ liệu đầu vào thông qua việc lấy mẫu từ các vùng dữ liệu.

Có hai loại lớp lấy mẫu phổ biến trong CNN:

- Lớp lấy mẫu cực đại (Max pooling layer): Trong lớp này, mỗi vùng dữ liệu được chia thành các ô không gian (ví dụ: 2x2 hoặc 3x3) và giá trị lớn nhất trong mỗi ô được chọn làm giá trị biểu diễn cho vùng đó. Việc chọn giá trị lớn nhất giúp tạo ra một biểu diễn tốt về sự tồn tại của các đặc trưng quan trọng trong vùng dữ liệu.
- Lớp lấy mẫu trung bình (Average pooling layer): Tương tự như lớp cực đại, lớp lấy mẫu trung bình chia dữ liệu thành các ô không gian và tính giá trị trung bình trong mỗi ô. Tuy nhiên, thay vì chọn giá trị lớn nhất, nó chọn giá trị trung bình. Lớp này thường được sử dụng ít hơn vì nó có thể làm mất đi thông tin quan trọng trong quá trình lấy mẫu.

Việc sử dụng lớp lấy mẫu giúp giảm kích thước của dữ liệu, làm giảm chi phí tính toán và giảm overfitting. Nó cũng giúp tăng cường tính tổng quát hóa của mô hình bằng cách tạo ra các biểu diễn bao quát hơn từ dữ liệu đầu vào.



Hình 2.8 Phương thức Avarage Pooling và Max Pooling [8]

❖ **Lớp kết nối đầy đủ(Fully Connected Layer)**

Lớp kết nối đầy đủ (Fully Connected Layer), còn được gọi là lớp hoàn toàn kết nối, là một loại lớp trong mạng nơ-ron nhân tạo (ANN) trong đó mỗi nơ-ron trong lớp này kết nối với tất cả các nơ-ron trong lớp trước đó. Điều này có nghĩa là mỗi đầu ra của lớp trước đó là đầu vào cho mỗi nơ-ron trong lớp kết nối đầy đủ.

Công việc chính của lớp kết nối đầy đủ là học cách kết hợp thông tin từ các đặc trưng được trích xuất từ các lớp trước đó và tạo ra các dự đoán hoặc đầu ra cuối cùng cho mô hình. Trong mạng nơ-ron tích chập (CNN), lớp kết nối đầy đủ thường được sử dụng ở cuối mạng để thực hiện các tác vụ như phân loại.

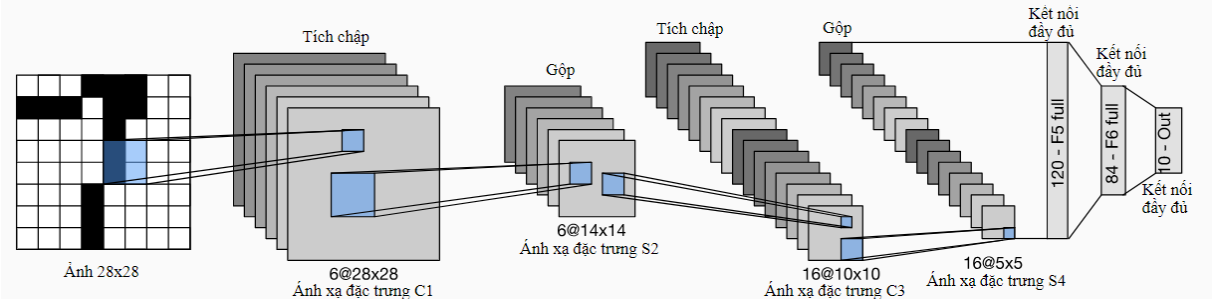
Một số điểm cần lưu ý về lớp kết nối đầy đủ bao gồm:

- Số lượng tham số trong lớp này có thể rất lớn, đặc biệt khi số lượng nơ-ron trong lớp trước đó lớn.
- Lớp kết nối đầy đủ có thể dẫn đến hiện tượng overfitting, đặc biệt khi sử dụng trong các mô hình có kích thước lớn.
- Việc sử dụng các kỹ thuật như dropout có thể giúp giảm overfitting trong lớp kết nối đầy đủ.

2.3. Một số mô hình mạng CNN nổi tiếng

2.3.1. Mạng Lenet

LeNet là một trong những mạng CNN lâu đời nổi tiếng nhất được Yann LeCUn phát triển vào những năm 1998. Cấu trúc của LeNet gồm 2 layer (Convolution + maxpooling) và 2 layer fully connected layer và output là softmax layer. Chúng ta cùng tìm hiểu chi tiết architect của LeNet đối với dữ liệu mnist (accuracy lên đến 99%).



Hình 2.9 Cách thức hoạt động của mạng Lenet [12]

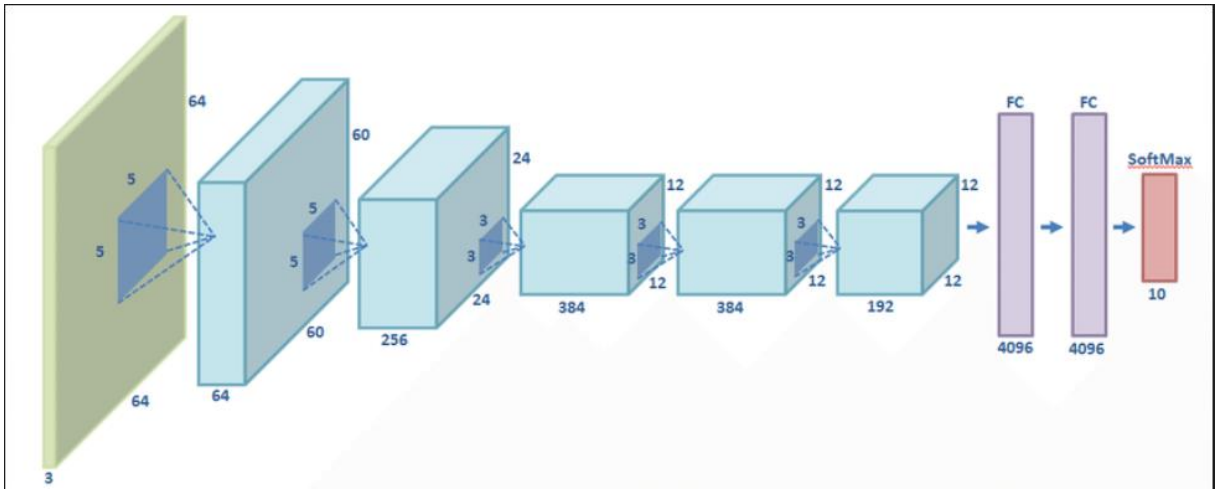
- Input shape 28x28x3
- Layer 1 :
- Convolution layer 1 : Kernel 5x5x3 , stride = 1,no padding, number filter = 6 ,output = 28x28x6.
- Maxpooling layer : pooling size 2x2,stride = 2,padding = “same”,output = 14x14x6.
- Layer 2 :
- Convolution layer 2 : kernel 5x5x6,stride = 1, no padding, number filter = 16,output = 10x10x16.
- Maxpooling layer : pooling size = 2x2, stride = 2, padding =”same”,output = 5x5x16.
- Flatten output = 5x5x16 = 400
- Fully connected 1 : output = 120
- Fully connected 2 : output = 84
- Softmax layer, output = 10 (10 digits).

Nhược điểm của LeNet là mạng còn rất đơn giản và sử dụng sigmoid (or tanh) ở mỗi convolution layer mạng tính toán rất chậm.

2.3.2 Mạng Alexnet

AlexNet là một mạng CNN đã dành chiến thắng trong cuộc thi ImageNet LSVRC-2012 năm 2012 với large margin (15.3% VS 26.2% error rates). AlexNet là một mạng CNN traning với một số lượng parameter rất lớn (60 million) so với LeNet. Một số đặc điểm:

- Sử dụng relu thay cho sigmoid(or tanh) để xử lý với non-linearity. Tăng tốc độ tính toán lên 6 lần.
- Sử dụng dropout như một phương pháp regularization mới cho CNN. Dropout không những giúp mô hình tránh được overfitting mà còn làm giảm thời gian huấn luyện mô hình
- Overlap pooling để giảm size của network (Traditionally pooling regions không overlap).
- Sử dụng local response normalization để chuẩn hóa ở mỗi layer.
- Sử dụng kỹ thuật data augmentation để tạo them data training bằng cách translations, horizontal reflections.
- Alexnet training với 90 epochs trong 5 đến 6 ngày với 2 GTX 580 GPUs. Sử dụng SGD với learning rate 0.01, momentum 0.9 và weight decay 0.0005.



Hình 2.10 Cách thức hoạt động của mạng Alexnet [14]

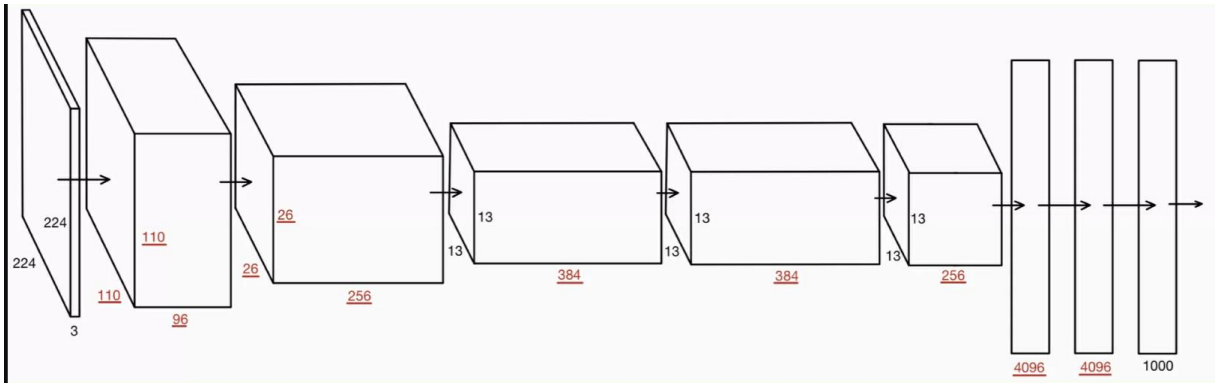
- Architect của Alexnet gồm 5 convolutional layer và 3 fully connection layer. Activation Relu được sử dụng sau mỗi convolution và fully connection layer. Detail architecter với dataset là imagenet size là 227x227x3 với 1000 class (khác với trong hình trên size là 224x224):
- Input shape: 227x227x3.
- Layer 1 :
- Conv 1: kernel : 11x11x3,stride = 4,no padding, number = 96,activation = relu,output = 55x55x96.
- Maxpooling layer: poolingsize = 3x3, stride = 2, padding =”same”, output = 27x27x96.
- Normalize layer.
- Layer 2 :
- Conv 2: kernel: 3x3x96, stride = 1, padding = “same”, number filter = 256, activation = relu, output = 27x27x256.
- Maxpooling layer: pooling size = 3x3, stride=2, padding =”same”, output = 13x13x256.
- Normalize layer.
- Layer 3:
- Conv 3: kernel :3x3x256, stride = 1, padding=”same”, number filter = 384, activation = relu, output = 13x13x384.
- Layer 4:
- Conv 4 : kernel : 3x3x384 , stride = 1, padding = “same”, number filter = 384, activation= relu, output = 13x13x384
- Layer 5 :

- Conv 5 : kernel 3x3x384, stride = 1, padding = “same”, number filter = 256, activation = relu, output = 13x13x256.
- Pooling layer : pooling size = 3x3,stride =2,padding =”same”,output = 6x6x256.
- Flatten 256x6x6 = 9216
- Fully connected layer 1 : activation = relu , output = 4096 + dropout(0.5).
- Fully connected layer 2 : activation = relu , output = 4096 + dropout(0.5).
- Fully connected layer 3 : activation = softmax , output = 1000 (number class)

2.3.3. Mạng ZFNet

ZFNet là một mạng cnn thắng trong ILSVRC 2013 với top-5 error rate của 14.8% . ZFNet có cấu trúc rất giống với AlexNet với 5 layer convolution, 2 fully connected layer và 1 output softmax layer. Khác biệt ở chỗ kernel size ở mỗi Conv layer. Một số đặc điểm chính:

- Tương tự AlexNet nhưng có một số điều chỉnh nhỏ.
- Alexnet training trên 15m image trong khi ZF training chỉ có 1.3m image.
- Sử dụng kernel 7x7 ở first layer (alexnet 11x11). Lý do là sử dụng kernel nhỏ hơn để giữ lại nhiều thông tin trên image hơn.
- Tăng số lượng filter nhiều hơn so với alexnet
- Training trên GTX 580 GPU trong 20 ngày



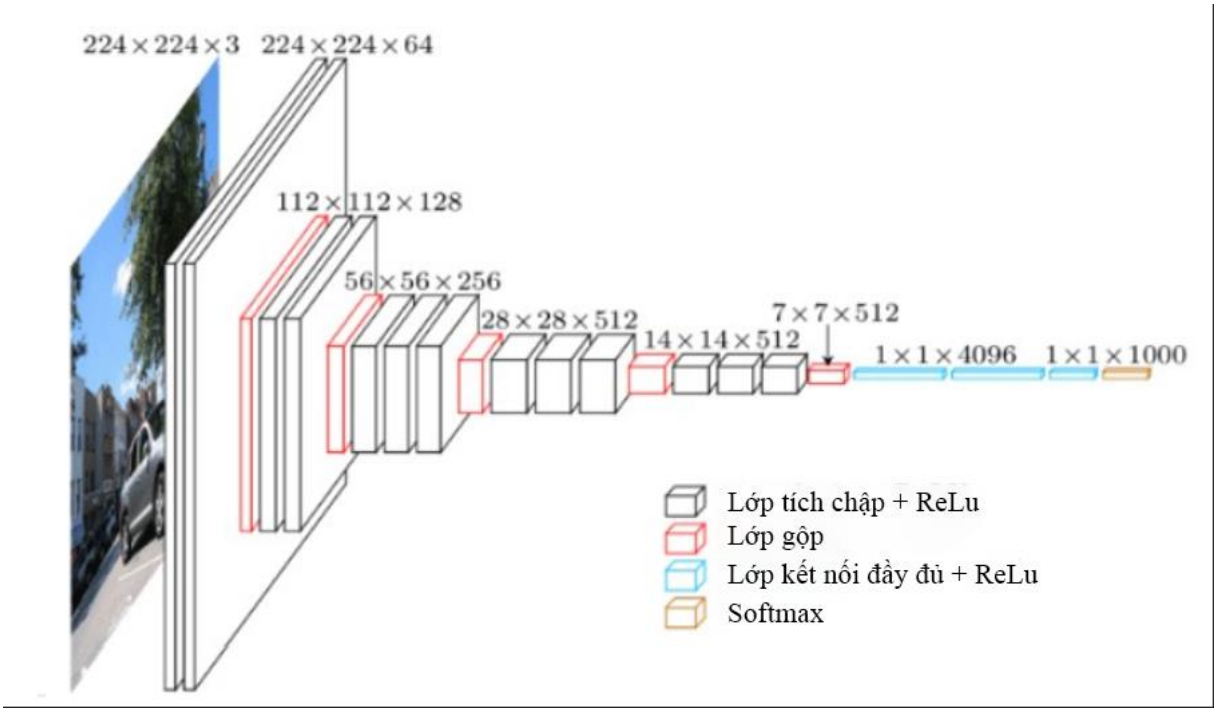
Hình 2.11 Cách thức hoạt động của mạng ZFNet [13]

- Input shape 224x224x3 .
- Layer 1 :
- Conv 1 : kernel = 7x7x3, stride = 2, no padding, number filter = 96, output = 110x110x96.
- Maxpooling 1 : pooling size = 3x3,stride=2, padding = “same”,output = 55x55x96
- Normalize layer.
- Layer 2 :
- Conv 2 : kernel = 5x5x96, stride = 2, no padding, number filter = 256, output = 26x26x256.
- Maxpooling 2 : pooling size = 3x3, stride=2, padding = “same”,output = 13x13x256
- Normalize layer.
- Layer 3:
- Conv 3 : kernel = 3x3x256, stride=1, padding=”same”, number filter = 384,output = 13x13x384.
- Layer 4 :
- Conv 4 : kernel = 3x3x384, stride=1, padding=”same”, number filter = 384,output = 13x13x384.

- Layer 5 :
- Conv 5 : kernel = 3x3x384, stride=1, padding=”same”, number filter = 256,output = 13x13x256.
- Maxpooling : pooling size = 3x3,stride =2,padding =”same”,output = 6x6x256.
- Flatten 6x6x256 = 9216
- Fully connected 1 : activation = relu,output =4096
- Fully connected 2 : activation = relu,output =4096
- Softmax layer for classifier ouput = 1000

2.3.4. Mạng VGG

Sau AlexNet thì VGG ra đời với một số cải thiện hơn , trước tiên là model VGG sẽ deeper hơn, tiếp theo là thay đổi trong thứ tự conv. Từ LeNet đến AlexNet đều sử dụng Conv-maxpooling còn VGG thì sử dụng 1 chuỗi Conv liên tiếp Conv-Conv-Conv ở middle và end của architect VGG. Việc này sẽ làm cho việc tính toán trở nên lâu hơn nhưng những feature sẽ vẫn được giữ lại nhiều hơn so với việc sử dụng maxpooling sau mỗi Conv. Hơn nữa hiện nay với sự ra đời của GPU giúp tốc độ tính toán trở nên nhanh hơn rất nhiều lần thì vấn đề này không còn đáng lo ngại. VGG cho small error hơn AlexNet trong ImageNet Large Scale Visual Recognition Challenge (ILSVRC) năm 2014. VGG có 2 phiên bản là VGG16 và VGG19



Hình 2.12 Cách thức hoạt động của mạng VGG [5]

- Architect của VGG16 bao gồm 16 layer :13 layer Conv (2 layer conv-conv,3 layer conv-conv-conv) đều có kernel 3x3, sau mỗi layer conv là maxpooling downsize xuống 0.5, và 3 layer fully connection. VGG19 tương tự như VGG16 nhưng có thêm 3 layer convolution ở 3 layer conv cuối (thành 4 conv stack với nhau).
- Detail parameter VGG16

Bảng 2.1 Bảng chi tiết các tham số mạng VGG16

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

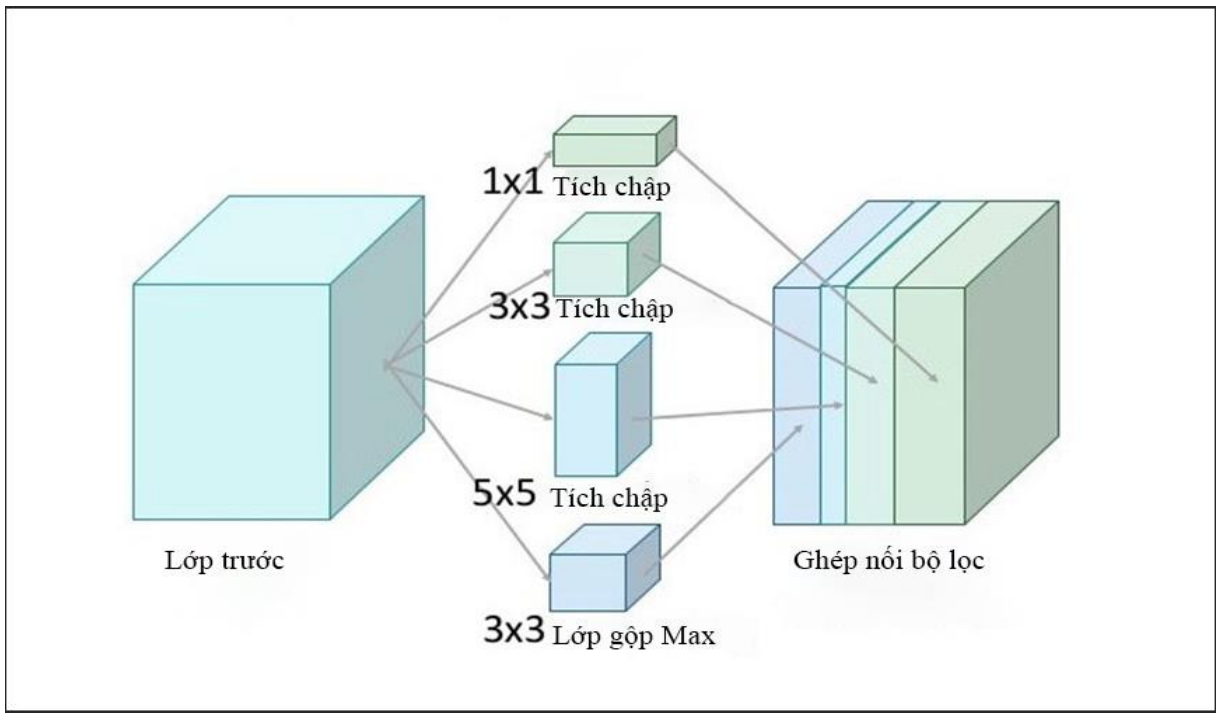
- Sử dụng kernel 3x3 thay vì 11x11 ở alexnet(7x7 ZFNet). Kết hợp 2 conv 3x3 có hiểu quả hơn 1 cov 5x5 về receptive field giúp mạng deeper hơn lại giảm tham số tính toán cho model.
- 3 Conv 3x3 có receptive field same 1 conv 7x7.
- Input size giảm dần qua các conv nhưng tăng số chiều sâu.
- Làm việc rất tốt cho task classifier và localizer (rất hay được sử dụng trong object detection).
- Sử dụng relu sau mỗi conv và training bằng batch gradient descent.
- Có sử dụng data augmentation technique trong quá trình training.
- Training với 4 Nvidia Titan Black GPUs trong 2-3 tuần.

2.3.5. Mạng GoogleNet

- Năm 2014, google publish một mạng neural do nhóm research của họ phát triển có tên là googleNet. Nó performance tốt hơn VGG, googleNet 6.7% error

rate trong khi VGG là 7.3% Ý tưởng chính là họ tạo ra một module mới có tên là inception giúp mạng traning sâu và nhanh hơn, chỉ có 5m tham số so với alexnet là 60m nhanh hơn gấp 12 lần.

- Inception module là một mạng CNN giúp training wider(thay vì them nhiều layer hơn vì rất dễ xảy ra overfitting + tăng parameter người ta nghĩ ra tăng deeper ở mỗi tầng layer) so với mạng CNN bình thường. Mỗi layer trong CNN truyền thống sẽ extract các thông tin khác nhau. Output của 5x5 conv kernel sẽ khác với 3x3 kernel. Vậy để lấy những thông tin cần thiết cho bài toán của chúng ta thì nên dùng kernel size như thế nào ? Tại sao chúng sử dụng tất cả ta và sau đó để model tự chọn. Đó chính là ý tưởng của Inception module, nó tính toán các kernel size khác nhau từ một input sau đó concatenate nó lại thành output.

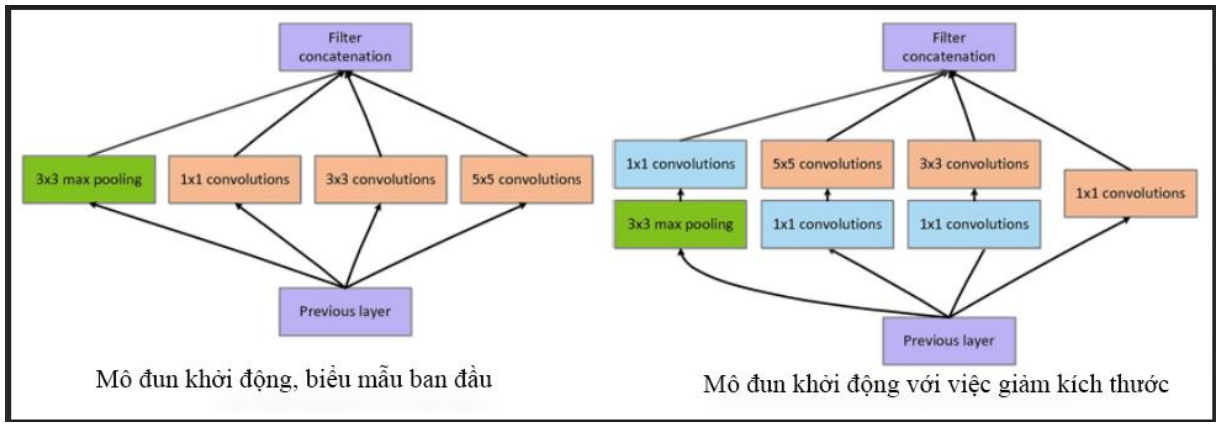


Hình 2.13 Cách thức hoạt động của mạng GoogleNet [5]

- Trong inception người ta dùng conv kernel 1x1 với 2 mục đích là giảm tham số tính toán và dimensionality reduction . Dimensionality reduction có thể hiểu làm giảm depth của input (vd iput 28x28x100 qua kernel 1x1 với filter = 10 sẽ

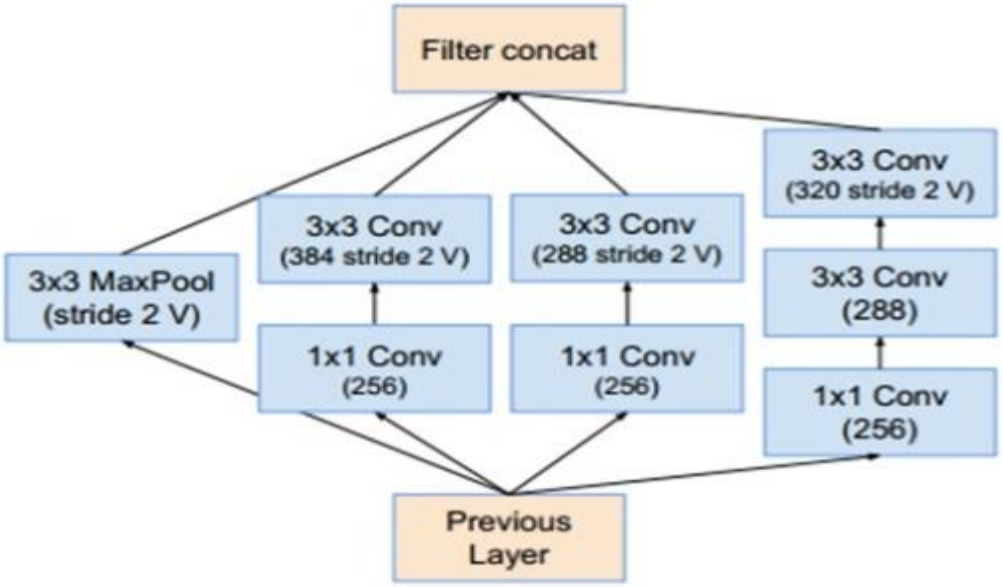
giảm depth về còn 28x28x10). Giảm chi phí tính toán có thể hiểu qua ví dụ sau:

- Input shape 28x28x192 qua kernel 5x5 với 32 thì ouput là 28x28x32(padding same) thì tham số tính toán là (5x5x192)*(28x28x32)=120 million
- Input shape 28x28x192 qua kernel 1x1x192 filter = 16 , output = 28x28x16 tiếp tục với kernel 5x5x32 filter = 16 đươc output = 28x28x32. Tổng tham số tính toán : (28x28x16)192 + (28x28x32)(5x5x16) = 2.4 + 10 = 12.4 million. Ta thấy với cùng output là 28x28x32 thì nếu dùng kernel 5x5x192 với 32 filter thì sẽ có tham số gấp 10 lần so với sử dụng kernel 1x1x192 sau đó dùng tiếp 1 kernel 5x5x16 với filter 32. Inception hiện giờ có 4 version , ta sẽ cùng tìm hiểu sơ qua các version:
- Inception v1: có 2 dạng là naive và dimension reduction. Khác biệt chính đó là version dimension reduction nó dùng conv 1x1 ở mỗi layer để giảm depth của input giúp model có ít tham số hơn. Inception naïve có architect gồm 1x1 conv,3x3 conv, 5x5 c 5x5 conv và 3x3 maxpooling.



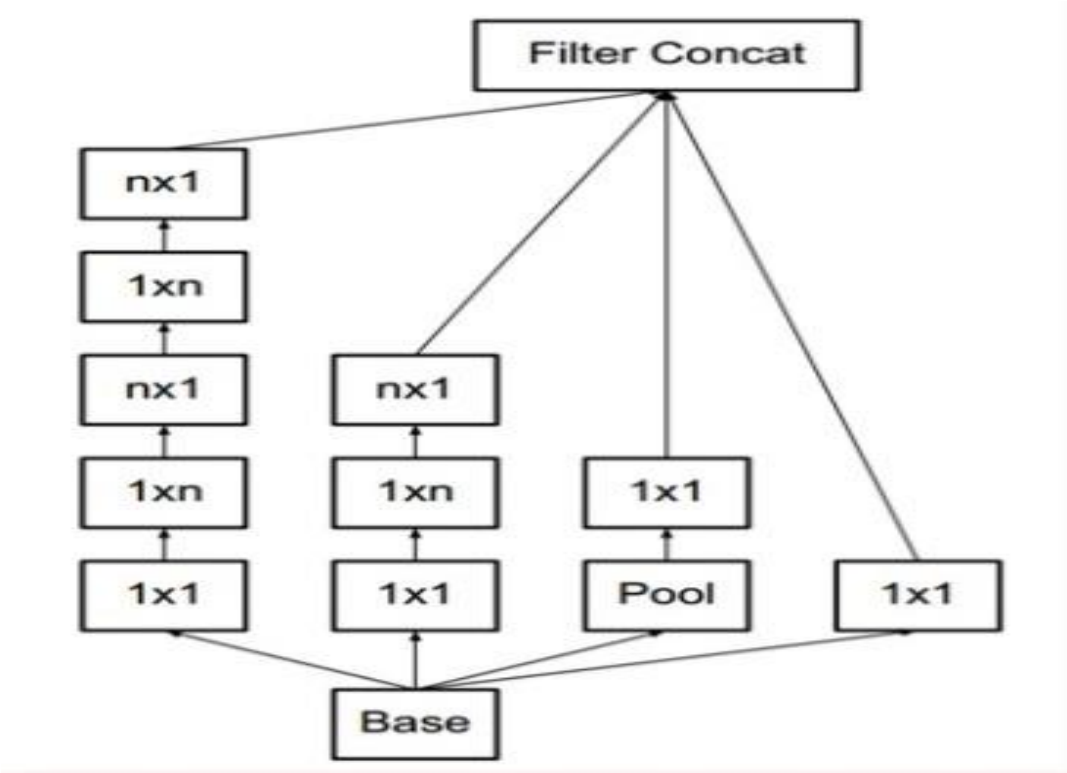
Hình 2.14 Inception v1 [5]

- Inception v2: Cải thiện version 1, thêm layer batchnormalize và giảm Internal Covariate Shift. Ouput của mỗi layer sẽ được normalize về Gaussian N(0,1). Conv 5x5 sẽ được thay thế bằng 2 conv 3x3 để giảm computation cost.



Hình 2.15 Inception v2[5]

- Inception v3: Điểm đáng chú ý ở version này là Factorization. Conv 7x7 sẽ được giảm về conv 1 dimesion là (1x7),(7x1). Tương tự conv 3x3 (3x1,1x3). Tăng tốc độ tính toán. Khi tách ra 2 conv thì làm model deeper hơn.



Hình 2.16 Inception v3 [5]

- Inception v4: là sự kết hợp inception và resnet. Detail googleNet architect

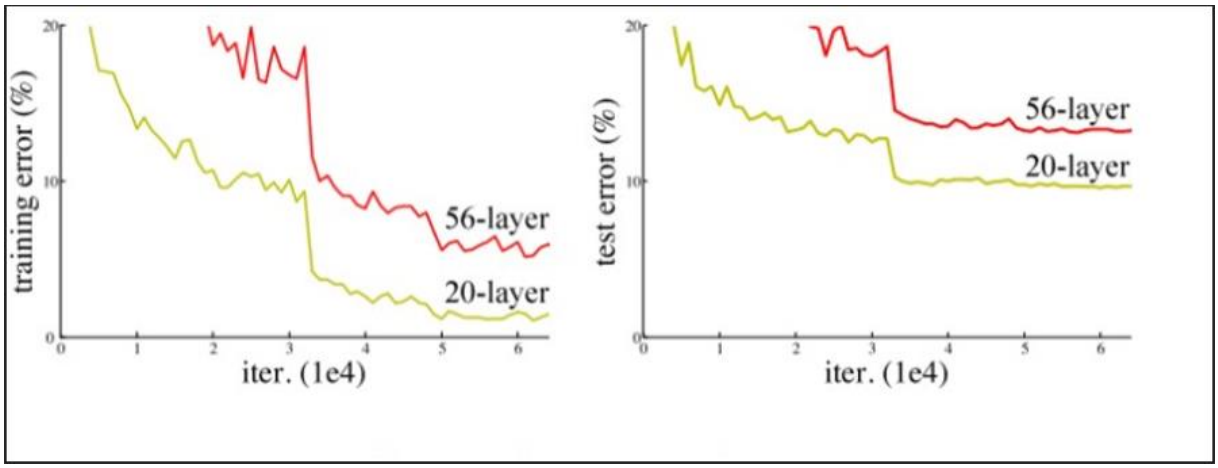
Bảng 2.2 Bảng tham số chi tiết của inception v4

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

- GoogleNet gồm 22 layer, khởi đầu vẫn là những simple convolution layer, tiếp theo là những block của inception module với maxpooling theo sau mỗi block. Một số đặc điểm chính.
- Sử dụng 9 Inception module trên toàn bộ architect. Làm model deeper hơn rất nhiều.
- Không sử dụng fully connection layer mà thay vào đó là average pooling từ 7x7x1024 volume thành 1x1x1024 volume giảm thiểu được rất nhiều parameter.
- Ít hơn 12x parameter so với Alexnet.
- Auxiliary Loss được add vào total loss(weight =0.3). Nhưng được loại bỏ khi test.

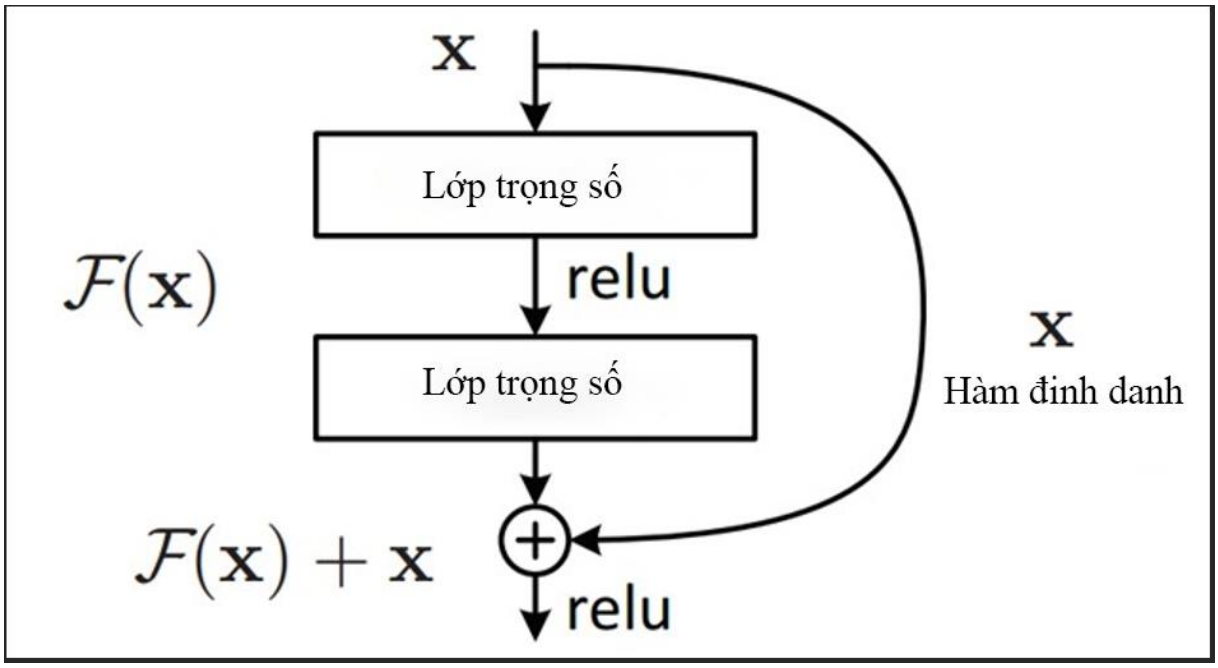
2.3.6. Mạng ResNets

ResNet được phát triển bởi microsoft năm 2015 với paper " Deep residual learning for image recognition". ResNet winer ImageNet ILSVRC competition 2015 với error rate 3.57% ,ResNet có cấu trúc gần giống VGG với nhiều stack layer làm cho model deeper hơn. Không giống VGG, resNet có depth sâu hơn như 34,55,101 và 151 . Resnet giải quyết được vấn đề của deep learning truyền thống , nó có thể dễ dàng training model với hàng trăm layer. Để hiểu ResNet chúng ta cần hiểu vấn đề khi stack nhiều layer khi training, vấn đề đầu tiên khi tăng model deeper hơn gradient sẽ bị vanishing/explodes. Vấn đề này có thể giải quyết bằng cách thêm Batch Normalization nó giúp normalize output giúp các hệ số trở nên cân bằng hơn không quá nhỏ hoặc quá lớn nên sẽ giúp model dễ hội tụ hơn. Vấn đề thứ 2 là degradation, Khi model deeper accuracy bắt đầu bão hòa(saturated) thậm chí là giảm. Như hình vẽ bên dưới khi stack nhiều layer hơn thì training error lại cao hơn ít layer như vậy vấn đề không phải là do overfitting. Vấn đề này là do model không dễ training khó học hơn, thử tượng tượng một training một shallow model, sau đó chúng ta stack thêm nhiều layer , các layer sau khi thêm vào sẽ không học thêm được gì cả (identity mapping) nên accuracy sẽ tương tự như shallow model mà không tăng. Resnet được ra đời để giải quyết vấn đề degradation này.



Hình 2.17 So sánh độ chính xác của trainning và test [5]

ResNet có architecture gồm nhiều residual block, ý tưởng chính là skip layer bằng cách add connection với layer trước. Ý tưởng của residual block là feed foword x(input) qua một số layer conv-max-conv, ta thu được F(x) sau đó add thêm x vào H(x) = F(x) + x . Model sẽ dễ học hơn khi chúng ta thêm feature từ layer trước vào.

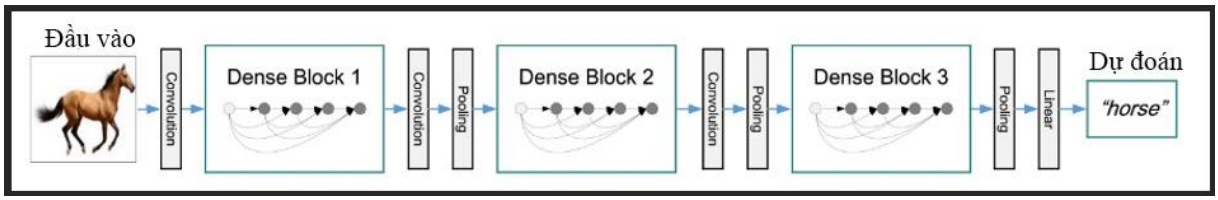


Hình 2.18 Khối ResNets [5]

- Sử dụng batch Normalization sau mỗi Conv layer.
- Initialization Xavier/2
- Training với SGD + momentum(0.9)
- Learning rate 0.1, giảm 10 lần nếu error ko giảm
- Mini batch size 256
- Weight decay 10^-5
- Không sử dụng dropout

2.3.7. Mạng Densenet

Densenet(Dense connected convolutional network) là một trong những netwok mới nhất cho visual object recognition. Nó cũng gần giống Resnet nhưng có một vài điểm khác biệt. Densenet có cấu trúc gồm các dense block và các transition layers. Được stack dense block- transition layers-dense block- transition layers như hình vẽ. Với CNN truyền thống nếu chúng ta có L layer thì sẽ có L connection, còn trong densenet sẽ có L(L+1)/2 connection.



Hình 2.19 Mô hình hoạt động của mạng Denseset [5]

Hãy tưởng tượng ban đầu ta có 1 image size (28,28,3). Đầu tiên ta khởi tạo feature layer bằng Conv tạo ra 1 layer size (28,28,24). Sau mỗi layer tiếp theo (Trong dense block) nó sẽ tạo thêm K = 12 feature giữa nguyên width và height. Khi đó output tiếp theo sẽ là (28,28,24 +12),(28,28,24 +12+12). Ở mỗi dense block sẽ có normalization, nonlinearity và dropout. Để giảm size và depth của feature thì transition layer được đặt giữa các dense block, nó gồm Conv kernel size =1, average pooling (2x2) với stride = 2 nó sẽ giảm output thành (14,14,48)

- Detail Parameter

Bảng 2.3 Bảng tham số chi tiết cảu mạng Densenet

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv, 3 × 3 conv] × 24	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48	[1 × 1 conv, 3 × 3 conv] × 64
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv, 3 × 3 conv] × 16	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Một số ưu điểm của Densenet:

- Accuracy : Densenet training tham số ít hơn 1 nửa so với Resnet nhưng có same accuracy so trên ImageNet classification dataset.
- Overfitting : DenseNet resistance overfitting rất hiệu quả.
- Giảm được vashing gradient.
- Sử dụng lại feature hiệu quả hơn.

2.4. Các phương pháp tiền xử lí dữ liệu trong CNN

Trước khi dữ liệu hình ảnh được đưa vào mô hình Convolutional Neural Network (CNN), quá trình tiền xử lý dữ liệu đóng vai trò quan trọng để chuẩn bị dữ liệu đầu vào một cách hiệu quả và tối ưu. Các phương pháp tiền xử lý này không chỉ giúp cải thiện chất lượng của dữ liệu, mà còn giúp mô hình học sâu hội tụ nhanh hơn và đạt được kết quả tốt hơn trong việc nhận dạng và phân loại ảnh.

Một trong những bước đầu tiên trong quá trình tiền xử lý dữ liệu là thay đổi kích thước của ảnh. Đối với một tập dữ liệu hình ảnh, kích thước của các ảnh thường không đồng nhất. Do đó, trước khi đưa vào mô hình, các ảnh thường được

thay đổi kích thước về một kích thước chuẩn nhất định. Thông thường, kích thước này là các kích thước vuông hoặc có tỷ lệ, chẳng hạn như 16x16, 32x32, 64x64, vv. Thay đổi kích thước giúp đồng nhất kích thước của tất cả các ảnh trong tập dữ liệu, từ đó giảm chi phí tính toán và tăng hiệu suất của mô hình.

Tiếp theo, sau khi thay đổi kích thước, dữ liệu được chuẩn hóa để đảm bảo rằng các giá trị pixel của ảnh nằm trong phạm vi từ 0 đến 1. Việc này giúp mô hình học sâu hội tụ nhanh hơn và tránh tình trạng số hóa không tốt. Phương pháp chuẩn hóa phổ biến nhất là chia mỗi giá trị pixel cho giá trị lớn nhất có thể (thường là 255), làm cho các giá trị pixel nằm trong khoảng từ 0 đến 1.

Ngoài ra, một phương pháp quan trọng khác để tăng cường dữ liệu là tăng cường dữ liệu (Data Augmentation). Tăng cường dữ liệu là quá trình mở rộng tập dữ liệu huấn luyện bằng cách tạo ra các biến thể của các mẫu dữ liệu gốc. Các biến thể này có thể bao gồm cắt tỉa (cropping), phóng to (zooming), quay ảnh (rotation), lật ảnh (flipping), và điều chỉnh độ sáng tối (brightness adjustment). Tăng cường dữ liệu giúp mô hình học sâu trở nên tổng quát hơn và chống lại hiện tượng quá mức học.

Bên cạnh các phương pháp trên, có thể sử dụng các kỹ thuật khác như loại bỏ nhiễu, phân đoạn ảnh (image segmentation), điều chỉnh độ tương phản (contrast adjustment), và làm mịn ảnh (smoothing) để cải thiện chất lượng và độ phức tạp của dữ liệu.

Tóm lại, quá trình tiền xử lý dữ liệu trong thuật toán CNN không chỉ là một bước cần thiết mà còn là một phần quan trọng trong việc chuẩn bị dữ liệu cho mô hình. Bằng cách kết hợp các phương pháp tiền xử lý này, chúng ta có thể tối ưu hóa hiệu suất và độ chính xác của mô hình CNN trong việc nhận dạng và phân loại ảnh.

2.5. Ứng dụng của thuật toán CNN

Nhận diện hình ảnh (Image Recognition):

- Nhận diện đối tượng (Object Recognition): CNN có khả năng nhận diện và phân loại các đối tượng trong ảnh với độ chính xác cao, ví dụ như phân biệt giữa người đi bộ, xe hơi, xe đạp trong hệ thống xe tự lái.
- Nhận diện khuôn mặt (Facial Recognition): Công nghệ nhận diện khuôn mặt được sử dụng trong nhiều ứng dụng bảo mật như mở khóa điện thoại, kiểm tra an ninh tại sân bay, và giám sát an ninh công cộng.

Phân loại hình ảnh (Image Classification):

- Ứng dụng y tế: CNN được sử dụng để phân loại các loại bệnh từ hình ảnh y tế như X-quang, MRI, và CT, ví dụ như xác định bệnh lao từ hình ảnh X-quang phổi.
- Phân loại thực vật và động vật: Trong nghiên cứu sinh học, CNN giúp phân loại các loài thực vật và động vật từ hình ảnh, hỗ trợ các nhà khoa học trong việc nghiên cứu và bảo tồn.

Phát hiện đối tượng (Object Detection):

- An ninh và giám sát: Hệ thống giám sát an ninh sử dụng CNN để phát hiện các hành vi đáng ngờ hoặc nhận diện đối tượng trong video thời gian thực.
- Ứng dụng công nghiệp: Trong các nhà máy, CNN được sử dụng để phát hiện các sản phẩm lỗi hoặc theo dõi quy trình sản xuất.

Phân đoạn ảnh (Image Segmentation):

- Ứng dụng y tế: Phân đoạn ảnh giúp xác định vùng bị tổn thương trong các hình ảnh MRI hoặc CT, như xác định khối u trong não.
- Định vị đường: Trong xe tự lái, phân đoạn ảnh giúp xe xác định làn đường, biển báo giao thông, và các đối tượng xung quanh để di chuyển an toàn.

Xử lý văn bản và hình ảnh (Text and Image Processing):

- Nhận dạng ký tự quang học (OCR): Công nghệ OCR sử dụng CNN để chuyển đổi hình ảnh chứa văn bản thành văn bản dạng kỹ thuật số, hữu ích trong việc số hóa tài liệu.
- Tìm kiếm hình ảnh (Image Search): Các công cụ tìm kiếm sử dụng CNN để phân tích và tìm kiếm hình ảnh tương tự dựa trên nội dung hình ảnh.

Xử lý hình ảnh trong thực tế ảo và tăng cường (AR/VR):

- Thực tế tăng cường (AR): CNN giúp nhận diện và theo dõi các đối tượng trong môi trường thực để hiển thị thông tin hoặc đồ họa bổ sung trong thời gian thực.
- Thực tế ảo (VR): Trong môi trường VR, CNN giúp cải thiện tương tác và trải nghiệm người dùng bằng cách nhận diện và theo dõi các hành động của người dùng.

Chuyển đổi phong cách hình ảnh (Image Style Transfer):

- Nghệ thuật kỹ thuật số: Các nghệ sĩ kỹ thuật số sử dụng CNN để tạo ra các tác phẩm nghệ thuật bằng cách áp dụng phong cách của một bức tranh nổi tiếng lên ảnh chụp.
- Ứng dụng thương mại: Các ứng dụng di động và phần mềm chỉnh sửa ảnh sử dụng công nghệ này để cung cấp các bộ lọc phong cách nghệ thuật cho người dùng.

Siêu phân giải ảnh (Image Super-Resolution):

- Nâng cao chất lượng ảnh: CNN được sử dụng để cải thiện chất lượng của các bức ảnh có độ phân giải thấp, giúp tăng độ nét và chi tiết của ảnh.
- Ứng dụng truyền hình và phim: Các hãng truyền hình và sản xuất phim sử dụng CNN để nâng cao chất lượng video và phim ảnh.

2.6. Tiểu kết chương 2

Trong chương 2, em đã tìm hiểu về cơ sở lý thuyết của mạng nơ-ron tích chập (CNN) và các phương pháp tiền xử lý dữ liệu trong học sâu. Em đã khám

phá sâu hơn về kiến trúc của mạng CNN, từ cách hoạt động của các lớp convolutional, pooling đến lợi ích của fully connected layers. Cùng với đó, em đã điểm qua một số mô hình CNN nổi tiếng như LeNet, AlexNet, VGG và ResNets, hiểu rõ về đặc điểm và ứng dụng của mỗi mô hình. Ngoài ra, em cũng đã được giới thiệu với các phương pháp tiền xử lý dữ liệu như chuẩn hóa, chuyển đổi kích thước ảnh và loại bỏ nhiễu, nhằm chuẩn bị dữ liệu cho quá trình huấn luyện mô hình. Cuối cùng, em đã nhìn nhận về ứng dụng của CNN trong nhiều lĩnh vực, từ nhận dạng vật thể đến xử lý ảnh y khoa, mở ra một cái nhìn rộng lớn về tiềm năng của công nghệ này trong thực tế.

CHƯƠNG 3: TRIỂN KHAI VÀ ĐÁNH GIÁ HỆ THỐNG

3.1. Thiết kế hệ hệ thống

3.1.1. Mục tiêu của hệ thống

Nhận dạng chính xác các loài thực vật: Hệ thống được thiết kế để nhận dạng và phân loại các loài hoa, lá, cây, hoặc các bộ phận khác của thực vật từ dữ liệu hình ảnh. Điều này đòi hỏi mô hình học sâu phải có khả năng phát hiện các đặc điểm đặc trưng và sử dụng chúng để xác định loài.

Hỗ trợ người dùng không chuyên về thực vật: Một trong những mục tiêu quan trọng là giúp những người không chuyên về thực vật có thể xác định tên và thông tin về các loài hoa và lá một cách dễ dàng. Điều này giúp phổ biến kiến thức về thực vật tới cộng đồng rộng rãi hơn.

Tăng cường giáo dục và nhận thức về hệ thực vật: Hệ thống này có thể được sử dụng trong giáo dục để giúp học sinh, sinh viên, và những người quan tâm đến thực vật học có thể học hỏi và nhận biết các loài thực vật dễ dàng hơn. Điều này có thể góp phần vào việc nâng cao ý thức bảo vệ môi trường và đa dạng sinh học.

Hỗ trợ các ứng dụng trong nông nghiệp và lâm nghiệp: Hệ thống nhận dạng hoa lá có thể có ứng dụng trong ngành nông nghiệp và lâm nghiệp. Nó có thể giúp các nhà nông nghiệp và lâm nghiệp xác định các loại cây trồng, phát hiện sớm bệnh tật trên lá, và theo dõi sức khỏe của cây trồng, từ đó nâng cao năng suất và chất lượng.

Cung cấp thông tin bổ sung về các loài thực vật: Ngoài việc nhận dạng, hệ thống có thể cung cấp thông tin bổ sung về các loài thực vật, chẳng hạn như điều kiện sinh trưởng, môi trường sống, và cách chăm sóc. Điều này làm tăng giá trị của chương trình như một công cụ tham khảo và học hỏi.

3.1.2. Lựa chọn công nghệ

Python là một trong những ngôn ngữ lập trình phổ biến nhất trong lĩnh vực học máy và học sâu. Nó được ưa chuộng bởi cộng đồng phát triển lớn, cú pháp dễ

hiểu, và khả năng tích hợp với nhiều thư viện mạnh mẽ. Khi chọn Python, em có thể tận dụng lợi ích từ một loạt các thư viện chuyên dụng cho khoa học dữ liệu và học sâu.

❖ **Thư viện học Sâu: TensorFlow và Keras**

- TensorFlow là một nền tảng học sâu mã nguồn mở do Google phát triển. Nó cung cấp một cơ sở hạ tầng linh hoạt cho việc xây dựng, huấn luyện, và triển khai các mô hình học máy và học sâu. TensorFlow hỗ trợ cả CPU và GPU, giúp tăng tốc quá trình huấn luyện mô hình.

- Keras là một API cấp cao trên TensorFlow, giúp đơn giản hóa quá trình xây dựng mô hình học sâu. Keras cho phép ta nhanh chóng xây dựng các mạng nơron với cú pháp dễ đọc, giúp quá trình triển khai và thử nghiệm trở nên dễ dàng hơn.

Lựa chọn TensorFlow và Keras

- Dễ dàng xây dựng mô hình: Với Keras, em có thể dễ dàng xây dựng các mô hình học sâu như Convolutional Neural Networks (CNN) hoặc Recurrent Neural Networks (RNN) bằng cú pháp ngắn gọn.

- Hỗ trợ trên nhiều nền tảng: TensorFlow có thể chạy trên nhiều nền tảng, bao gồm máy tính cá nhân, máy chủ, và môi trường đám mây. Nó cũng hỗ trợ triển khai mô hình trên các thiết bị di động hoặc IoT.

- Cộng đồng và tài liệu: Cả TensorFlow và Keras đều có tài liệu phong phú và cộng đồng hỗ trợ lớn, giúp ta dễ dàng tìm kiếm giải pháp và học hỏi từ các nhà phát triển khác.

❖ **Thư viện hình ảnh: Matplotlib**

Matplotlib là một thư viện vẽ đồ thị và hình ảnh cho Python. Trong bối cảnh hệ thống nhận dạng hoa lá, Matplotlib được sử dụng để:

- Hiển thị hình ảnh: Em có thể sử dụng Matplotlib để hiển thị hình ảnh hoa lá trong quá trình tiền xử lý và kiểm thử mô hình.

- Trực quan hóa kết quả: Matplotlib cũng hữu ích trong việc trực quan hóa kết quả đánh giá mô hình, chẳng hạn như biểu đồ hiển thị độ chính xác, biểu đồ confusion matrix, hoặc biểu đồ ROC.

❖ **Thư viện tạo giao diện: Streamlit**

Streamlit là một thư viện mã nguồn mở và miễn phí trong Python, được sử dụng để tạo giao diện người dùng web cho các ứng dụng dữ liệu và máy học (machine learning) một cách dễ dàng và nhanh chóng. Thư viện này cho phép ta tạo các ứng dụng tương tác một cách nhanh chóng chỉ bằng việc sử dụng các đoạn mã Python.

Một số điểm nổi bật của Streamlit:

- Dễ sử dụng: Streamlit được thiết kế để đơn giản và dễ sử dụng, đặc biệt là cho những người có kiến thức cơ bản về Python.
- Tự động cập nhật: Khi thay đổi mã Python, giao diện của ứng dụng Streamlit sẽ tự động cập nhật mà không cần phải làm mới trình duyệt.
- Hỗ trợ nhanh chóng cho dữ liệu: Streamlit hỗ trợ nhanh chóng cho việc hiển thị dữ liệu trong các định dạng phổ biến như DataFrame, biểu đồ Matplotlib hoặc Plotly.
- Tùy biến cao: Mặc dù Streamlit cung cấp một loạt các thành phần giao diện người dùng sẵn có, nhưng bạn cũng có thể tùy chỉnh giao diện theo ý muốn bằng cách sử dụng các thành phần HTML và CSS.
- Hỗ trợ tích hợp: Streamlit tích hợp tốt với các thư viện và công cụ phổ biến khác trong hệ sinh thái Python, bao gồm Pandas, NumPy, Matplotlib, Plotly, và nhiều thư viện máy học khác.

❖ **Thư viện pickle**

Thư viện pickle trong Python là một công cụ mạnh mẽ được sử dụng để chuyển đổi các cấu trúc dữ liệu Python thành một chuỗi byte và ngược lại. Nó cho

phép ta lưu trữ các đối tượng Python (như danh sách, từ điển, đối tượng lớp) vào tệp và khôi phục chúng một cách dễ dàng sau này

Lưu trữ dữ liệu Python:

- Thư viện pickle cho phép lưu trữ các đối tượng Python (như danh sách, từ điển, đối tượng lớp) vào tệp với định dạng nhị phân.
- Điều này rất hữu ích khi bạn muốn lưu trữ dữ liệu trạng thái của chương trình để sử dụng sau này hoặc chia sẻ dữ liệu với các chương trình khác.

Chuyển đổi dữ liệu Python thành byte stream:

- Pickle cho phép chuyển đổi các cấu trúc dữ liệu Python thành chuỗi byte, gọi là pickle object.
- Pickle object có thể lưu trữ dễ dàng trong tệp hoặc truyền qua mạng mà không cần phải thực hiện bất kỳ xử lý nào khác.

Khôi phục dữ liệu từ pickle object:

- Bằng cách sử dụng pickle, có thể khôi phục lại các cấu trúc dữ liệu Python từ pickle object một cách dễ dàng.
- Điều này cho phép tái tạo lại dữ liệu ban đầu và sử dụng nó trong chương trình của bạn mà không cần phải viết mã phức tạp.
- Khả năng chuyển đổi mọi loại dữ liệu Python:
- Pickle có thể chuyển đổi mọi loại dữ liệu Python, bao gồm cả các lớp và đối tượng tùy chỉnh mà bạn đã định nghĩa.

3.1.3. Đính dạng dữ liệu và cấu trúc dữ liệu

- Định dạng dữ liệu

Trong đồ án của em, em đã quyết định sử dụng định dạng hình ảnh PNG để làm việc với dữ liệu vì nó cung cấp một số lợi ích đáng kể. PNG (Portable Network

Graphics) là một định dạng hình ảnh nén không mất mát, được sử dụng rộng rãi trong các ứng dụng liên quan đến đồ họa và hình ảnh số.

Một trong những ưu điểm quan trọng của PNG là khả năng bảo tồn độ trong suốt và màu nền trong suốt. Điều này cho phép em làm việc với các hình ảnh có nền trong suốt hoặc tích hợp vào các tài liệu, ứng dụng hoặc trang web mà không làm mất đi tính thẩm mỹ. Điều này rất hữu ích khi em muốn hiển thị hình ảnh trên nền khác nhau mà không gây ra hiệu ứng nổi bật hoặc gây phiền hà cho người xem.

PNG cũng hỗ trợ màu sắc chính xác và hình ảnh chất lượng cao. Định dạng này sử dụng một thuật toán nén không mất mát, cho phép em lưu trữ hình ảnh mà không làm giảm chất lượng. Điều này rất hữu ích khi em cần làm việc với các hình ảnh chứa thông tin chi tiết quan trọng, chẳng hạn như trong các ứng dụng y tế, khoa học hoặc nghệ thuật.

Ngoài ra, PNG cũng hỗ trợ độ phân giải cao và khả năng nén tốt cho hình ảnh có sự đa dạng màu sắc. Điều này cho phép em lưu trữ và truyền tải dữ liệu hình ảnh mà không làm giảm chất lượng hình ảnh hoặc tạo ra hiện tượng nhiễu không mong muốn.

- Cấu trúc dữ liệu

Em đăt tên cho ảnh theo tên của 1 folder. Điều này giúp tổ chức và phân loại dữ liệu hình ảnh liên quan đến hoa lá một cách dễ dàng

- Đầu tiên, em đã tạo một thư mục gốc để lưu trữ toàn bộ dữ liệu hình ảnh về hoa lá. Thư mục gốc có thể được đặt tên tùy theo nhu cầu và mục đích của đồ án.
- Tiếp theo, em đã tạo các thư mục con bên trong thư mục gốc, mỗi thư mục con đại diện cho một loại hoa lá cụ thể. Ví dụ, em có thể tạo các thư mục con như "Hoa hồng", "Lá cây thông", "Hoa cúc", v.v.

3.2. Mô hình học sâu

Trong đồ án của mình, em đã sử dụng thuật toán CNN để nhận dạng các loài hoa và lá.

Một trong những điểm mạnh của thuật toán CNN là khả năng của nó trong việc tự động học các đặc trưng từ dữ liệu hình ảnh. Thay vì phải thiết kế và cài đặt các thuật toán phức tạp từ đầu, chúng ta có thể sử dụng CNN để tự động trích xuất các đặc điểm quan trọng từ dữ liệu đầu vào, giúp tiết kiệm thời gian và công sức đáng kể.

Thêm vào đó, một ưu điểm khác của CNN là khả năng học và điều chỉnh từ dữ liệu lớn. Điều này có nghĩa là với đủ dữ liệu huấn luyện, thuật toán có thể tự động cải thiện hiệu suất của mình, giúp chúng ta đạt được kết quả chính xác hơn trong việc phân loại, nhận dạng hoặc phân tích ảnh.

Cuối cùng, CNN còn có thể kết hợp với các kỹ thuật tiền xử lý và tiên tiến như tăng cường ảnh, trích xuất đặc trưng, để cải thiện hiệu suất của chúng. Sự kết hợp này tạo ra những hệ thống xử lý ảnh mạnh mẽ và linh hoạt hơn, phục vụ nhu cầu ngày càng đa dạng của xã hội hiện đại.

3.3. Dữ liệu và tiền xử lí dữ liệu

Dữ liệu gồm có 10345 ảnh bao gồm:

- Apple Folder có: 1645 ảnh
- daisy Folder có: 764 ảnh
- dandelion Folder có: 1052 ảnh
- Maize Folder có: 1162 ảnh
- Pepper Folder có: 1478 ảnh
- Potato Folder có: 152 ảnh
- rose Folder có: 784 ảnh
- sunflower Folder có: 733 ảnh

- Tomato Folder có: 1591 ảnh
- tulip Folder có: 984 ảnh

Trong đó thì 90% của mỗi ảnh dùng để huấn luyện và 10% dùng để test

1 số hàm dùng để gán nhãn label và xây dựng bộ dữ liệu

```
def make_data():
    for category in categories:
        path = os.path.join(data_dir,category)
        label = categories.index(category)
        for img_name in os.listdir(path):
            img_path = os.path.join(path,img_name)
            image = cv2.imread(img_path)
            try:
                image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
                image = cv2.resize(image, (200,200))
                image = np.array(image,dtype=np.float32)
                data.append([image,label])
            except Exception as e:
                pass
    print(len(data))
    pik = open('data.pickle','wb')
    pickle.dump(data,pik)
    pik.close()
def load_data():
    pick = open('data.pickle','rb')
```

```
    data = pickle.load(pick)
    pick.close()
    np.random.shuffle(data)
    feature=[]
    labels =[]
    for img, label in data:
        feature.append(img)
        labels.append(label)
    feature = np.array(feature,dtype=np.float32)
    labels = np.array(labels)
    feature = feature/255.0
    return [feature,labels]
```

❖ Xây dựng mô hình CNN tự xây dựng mô hình

```
input_layer = layers.Input([150,150,3])
conv1 = layers.Conv2D(filters=32,kernel_size=(5,5),padding='same',activation='relu')(input_layer)
pool1 = layers.MaxPool2D(pool_size=(2,2))(conv1)
conv2 = layers.Conv2D(filters=64,kernel_size=(3,3),padding='same',activation='relu')(pool1)
pool2 = layers.MaxPool2D(pool_size=(2,2),strides=(2,2))(conv2)
conv3 = layers.Conv2D(filters=96,kernel_size=(3,3),padding='same',activation='relu')(pool2)
pool3 = layers.MaxPooling2D(pool_size=(2,2),strides=(2,2))(conv3)
```

```
conv4                                                                    =
layers.Conv2D(filters=96,kernel_size=(3,3),padding='same',activation='relu')(p
ool3)

pool4 = layers.MaxPooling2D(pool_size=(2,2),strides=(2,2))(conv4)


flt1 = layers.Flatten()(pool4)

dn1 = layers.Dense(512,activation='relu')(flt1)

out = layers.Dense(7,activation='softmax')(dn1)
```

accuracy: 0.9680 - loss: 0.0851

- Lớp đầu vào (Input Layer):
- Hình ảnh đầu vào có kích thước [224, 224, 3]. Không có thay đổi nào xảy ra ở bước này.
- Lớp tích chập (Convolutional Layer 1):
- Áp dụng tích chập với 32 bộ lọc và kích thước kernel (5, 5). Kích thước không gian của đầu ra là [224, 224, 32]. Hình ảnh được biểu diễn bởi 32 kênh tích chập.
- Lớp pooling (Max Pooling 1):
- Áp dụng pooling với kích thước cửa sổ (2, 2). Kích thước không gian của đầu ra giảm xuống là [112, 112, 32]. Kích thước không gian của hình ảnh bị thu nhỏ đi một nửa.
- Lớp tích chập (Convolutional Layer 2):
- Áp dụng tích chập với 64 bộ lọc và kích thước kernel (3, 3). Kích thước không gian của đầu ra là [112, 112, 64]. Số lượng kênh tích chập tăng lên.
- Lớp pooling (Max Pooling 2):

- Áp dụng pooling với kích thước cửa sổ (2, 2) và bước nhảy (2, 2). Kích thước không gian của đầu ra giảm xuống là [56, 56, 64]. Hình ảnh lại bị thu nhỏ đi một nửa.
- Lớp tích chập (Convolutional Layer 3):
- Áp dụng tích chập với 96 bộ lọc và kích thước kernel (3, 3). Kích thước không gian của đầu ra là [56, 56, 96]. Số lượng kênh tích chập tăng lên.
- Lớp pooling (Max Pooling 3):
- Áp dụng pooling với kích thước cửa sổ (2, 2) và bước nhảy (2, 2). Kích thước không gian của đầu ra giảm xuống là [28, 28, 96]. Hình ảnh lại bị thu nhỏ đi một nửa.
- Lớp tích chập (Convolutional Layer 4):
- Áp dụng tích chập với 96 bộ lọc và kích thước kernel (3, 3). Kích thước không gian của đầu ra là [28, 28, 96]. Số lượng kênh tích chập không thay đổi.
- Lớp pooling (Max Pooling 4):
- Áp dụng pooling với kích thước cửa sổ (2, 2) và bước nhảy (2, 2). Kích thước không gian của đầu ra giảm xuống là [14, 14, 96]. Hình ảnh lại bị thu nhỏ đi một nửa.

❖ Xây dựng mô hình CNN của mạng AlexNet

```
input_layer = layers.Input([200,200, 3])

# Block 1

conv1 = layers.Conv2D(filters=96, kernel_size=(11, 11), strides=(4, 4),
activation='relu')(input_layer)

pool1 = layers.MaxPool2D(pool_size=(3, 3), strides=(2, 2))(conv1)
```

```
# Block 2
conv2 = layers.Conv2D(filters=256, kernel_size=(5, 5), padding='same',
activation='relu')(pool1)
pool2 = layers.MaxPool2D(pool_size=(3, 3), strides=(2, 2))(conv2)


# Block 3
conv3 = layers.Conv2D(filters=384, kernel_size=(3, 3), padding='same',
activation='relu')(pool2)
conv4 = layers.Conv2D(filters=384, kernel_size=(3, 3), padding='same',
activation='relu')(conv3)
conv5 = layers.Conv2D(filters=256, kernel_size=(3, 3), padding='same',
activation='relu')(conv4)
pool3 = layers.MaxPool2D(pool_size=(3, 3), strides=(2, 2))(conv5)


# Fully Connected Layers
flat1 = layers.Flatten()(pool3)
fc1 = layers.Dense(4096, activation='relu')(flat1)
drop1 = layers.Dropout(0.5)(fc1)
fc2 = layers.Dense(4096, activation='relu')(drop1)
drop2 = layers.Dropout(0.5)(fc2)
out = layers.Dense(10, activation='softmax')(drop2)
```

accuracy: 0.8964 - loss: 0.2847 của mạng Alexnet

❖ Xây dựng mô hình CNN theo mạng Lenet

```
input_layer = layers.Input([200,200, 3])


conv1 = layers.Conv2D(filters=6, kernel_size=(5, 5),
activation='tanh')(input_layer)
pool1 = layers.AveragePooling2D(pool_size=(2, 2), strides=(2, 2))(conv1)
# Layer 2: Convolutional Layer C3
conv2 = layers.Conv2D(filters=16, kernel_size=(5, 5),
activation='tanh')(pool1)
pool2 = layers.AveragePooling2D(pool_size=(2, 2), strides=(2, 2))(conv2)
# Flatten the output for the fully connected layers
flt1 = layers.Flatten()(pool2)
# Layer 3: Fully Connected Layer C5
fc1 = layers.Dense(120, activation='tanh')(flt1)
# Layer 4: Fully Connected Layer F6
fc2 = layers.Dense(84, activation='tanh')(fc1)
# Layer 5: Output Layer
out = layers.Dense(10, activation='softmax')(fc2)
```

accuracy: 0.9990 - loss: 0.0042

Công thức tính đầu ra sau mỗi bước conv:

- Kích thước đầu ra sau khi conv = (kích thước đầu vào – kernel + 1) / stride

Công thức tính đầu ra sau mõi bước maxpooling:

- Kích thước đầu ra sau khi maxpooling = ((kích thước đầu vào – kích thước maxpooling)/stride) +1

Tổng số trọng số 512*(n+1) trong đó n là số phần tử trong vector flatten

❖ Code giao diện người dùng

```
import os
```

```
import cv2
import streamlit as st
from keras.models import load_model
from keras.preprocessing import image
import numpy as np

# Tải mô hình đã train
# model = load_model('mymodel.h5')
# model = load_model('leaf.h5')
# model = load_model('leafflower.h5')
model = load_model('modeltest.h5')
# model = load_model('modeltesttwo.h5')
# model = load_model('Flowerleaf_Recog_Model.h5')

# Biến kiểm soát trạng thái camera
camera_on = False

# Dự đoán từ hình ảnh
def predict_from_image(img):
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    img_array = img_array / 255.0  # Chuẩn hóa dữ liệu
    prediction = model.predict(img_array)
    return prediction

st.set_page_config(page_title="Dự đoán loại hoa và lá từ hình ảnh",
page_icon="🌸")
st.title("Dự đoán loại hoa và lá từ hình ảnh")
st.write("Ứng dụng này sẽ giúp bạn dự đoán loại hoa và lá từ hình ảnh đã
cho.")
```

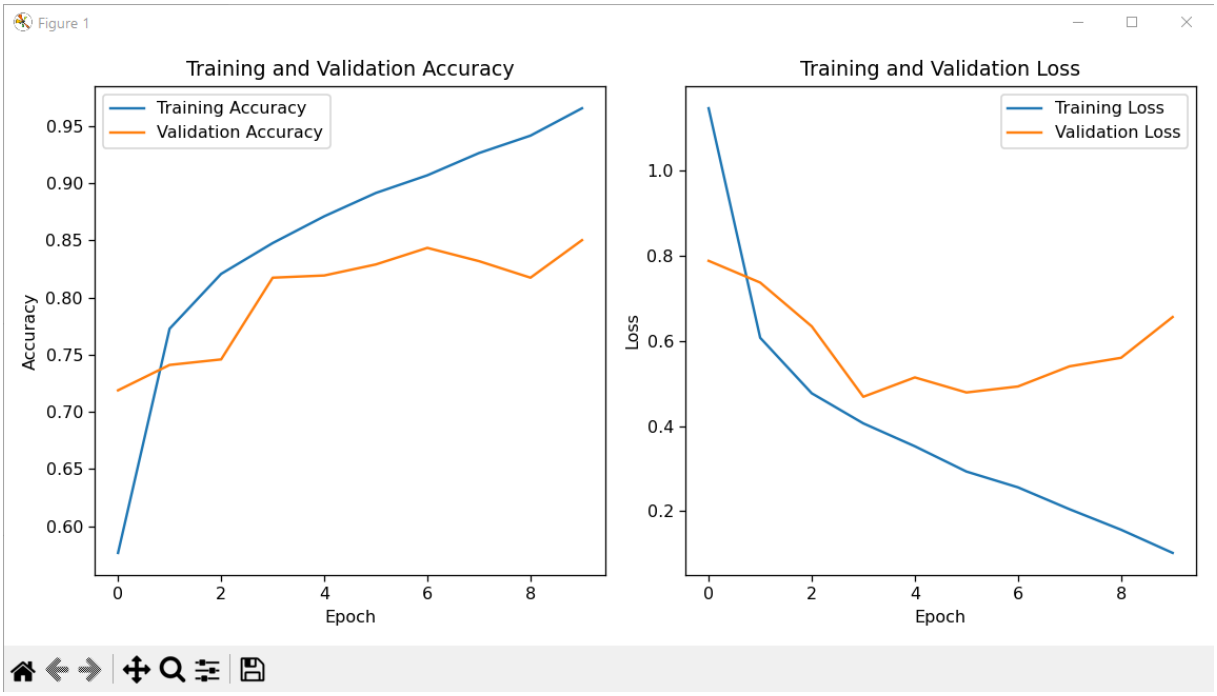
```
# Giao diện Streamlit
option = st.radio("Chọn phương pháp nhập ảnh:", ('Tải ảnh từ file', 'Chụp ảnh từ camera'))
if option == 'Tải ảnh từ file':
    uploaded_file = st.file_uploader("Tải lên hình ảnh", type=["jpg", "jpeg", "png"])
    if uploaded_file is not None:
        st.image(uploaded_file, caption='Hình ảnh đã tải lên', use_column_width=True)
        st.write("")
        st.write("Dự đoán:")

        # Tiến hành dự đoán khi có hình ảnh được tải lên
        if st.button('Dự đoán'):
            with st.spinner('Đang dự đoán...'):
                img = image.load_img(uploaded_file, target_size=(200,200))
                prediction = predict_from_image(img)
                # flower_name = ['daisy', 'dandelion', 'rose', 'sunflower', 'tulip']
                flower_name = ["Apple","daisy","dandelion","Maize","Pepper","Potato", "rose", "sunflower","Tomato", "tulip"]

                predicted_class = np.argmax(prediction)
                predicted_label = flower_name[predicted_class]
                # st.success('Loại hoa dự đoán: {}'.format(predicted_label))
                confidence = np.max(prediction) * 100
                threshold = 80
                # st.write("Tỉ lệ dự đoán đúng từ file: {:.2f}%".format(confidence))
                if confidence < threshold:  # Điều kiện ngưỡng confidence
```

```
                    st.warning("Không phải loại nào trong các loại trên.")
                else:
                    st.success('Loại hoa dự đoán: {}'.format(predicted_label))
                    st.write("Tỉ lệ dự đoán đúng từ file:
{:.2f}%".format(confidence))

elif option == 'Chụp ảnh từ camera':
    st.write("Nhấn nút bên dưới để chụp ảnh:")
    if not camera_on:
        if st.button('Chụp ảnh'):  # Nút này để kích hoạt camera
            camera_on = True
            cap = cv2.VideoCapture(0)
            while camera_on:
                ret, frame = cap.read()
                if ret:
                    frame = cv2.resize(frame, (640, 480))
                    cv2.imshow('Camera', frame)
                    key = cv2.waitKey(1)
                    if key & 0xFF == ord('q'):  # Nhấn 'q' để chụp ảnh
                        img = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
                        img = cv2.resize(img, (150,150))
                        img = np.array(img)
                        img = np.expand_dims(img, axis=0)
                        img = img / 255.0  # Chuẩn hóa dữ liệu

                        # Tiến hành dự đoán khi có ảnh từ camera
                        with st.spinner('Đang dự đoán...'):
                            prediction = model.predict(img)
                            # flower_name = ['daisy', 'dandelion', 'rose',
'sunflower', 'tulip']
```

```
                            flower_name =
["Apple","daisy","dandelion","Maize","Pepper","Potato", "rose",
"sunflower","Tomato", "tulip"]
                            predicted_class = np.argmax(prediction)
                            predicted_label = flower_name[predicted_class]

                        # Vẽ kết quả dự đoán lên ảnh đã chụp
                        cv2.putText(frame, predicted_label, (50, 50),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2)
                        st.image(frame, caption='Hình ảnh đã chụp và dự đoán',
use_column_width=True)
                        camera_on = False  # Dừng camera sau khi chụp ảnh và
dự đoán
                        confidence = np.max(prediction) * 100
                        # st.write("Tỉ lệ dự đoán đúng từ ảnh:
{:.2f}%".format(confidence))
                        threshold = 80
                        # st.write("Tỉ lệ dự đoán đúng từ file:
{:.2f}%".format(confidence))
                        if confidence < threshold:  # Điều kiện ngưỡng
confidence
                            st.warning("Không phải loại nào trong các loại
trên.")
                        else:
                            st.success('Loại hoa dự đoán:
{}'.format(predicted_label))
                            st.write("Tỉ lệ dự đoán đúng từ file:
{:.2f}%".format(confidence))
            cap.release()
            cv2.destroyAllWindows()
```

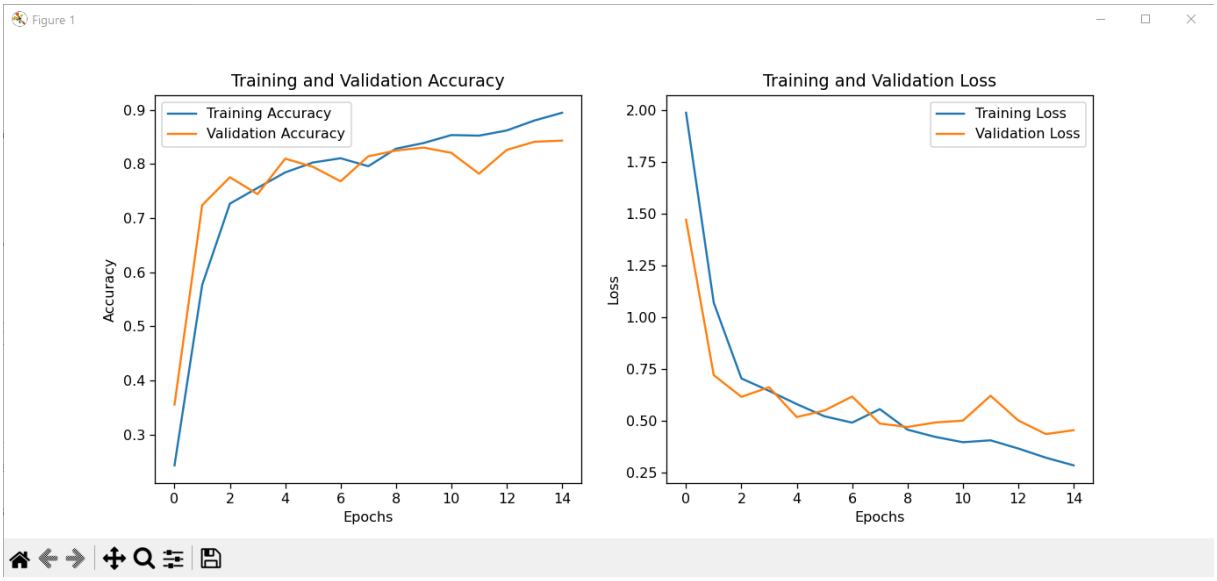
❖ Đồ thị về accuraru và loss của mạng lenet

```
history = model.fit(x_train, y_train, batch_size=100, epochs=10,
validation_data=(x_test, y_test))

plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(['Train', 'Validation'], loc='upper left')

# Vẽ biểu đồ loss
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['Train', 'Validation'], loc='upper right')

plt.tight_layout()
plt.show()
```

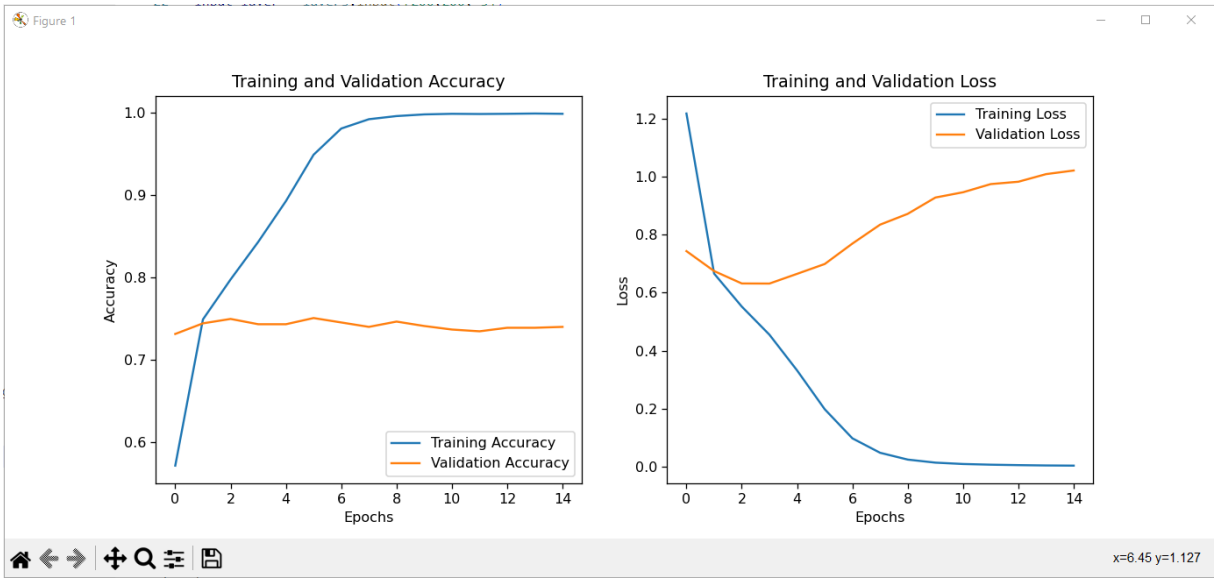


Hình 3.1 Đồ thị về accurary và loss của mạng tự làm

Đồ thị accurary và loss của mạng Alexnet



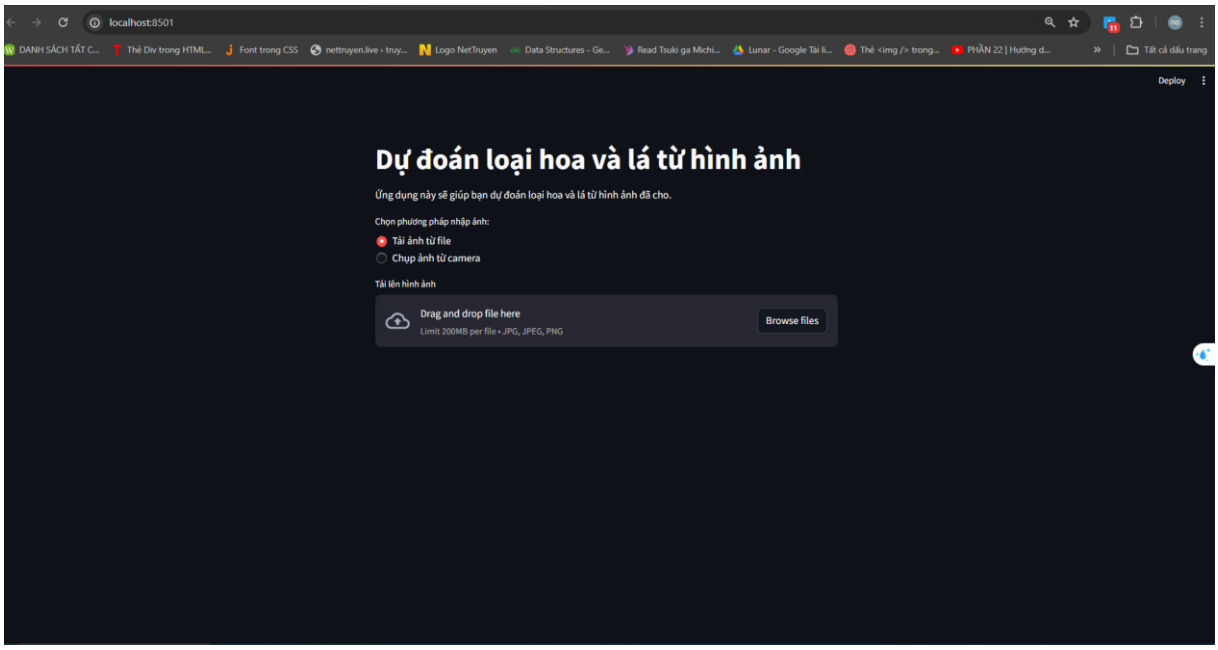
Hình 3.2 Đồ thị về accurary và loss của mạng Alexnet



Hình 3.3 Đồ thị về accurary và loss của mạng Lenet

Từ đồ thị trên thì ta có thể thấy được tỉ lệ dự đoạn chính xác sẽ tăng dần qua mỗi vòng lặp và tỉ lệ dự đoán sai cũng sẽ giảm dần qua mỗi vòng lặp

3.3. Giao diện chương trình



Hình 3.4 Giao diện hệ thống

Chương trình sẽ cho người sẽ có 2 lựa chọn cho người dùng. Đó là nhận diện thông qua camera và nhận diện ảnh thông qua ảnh:

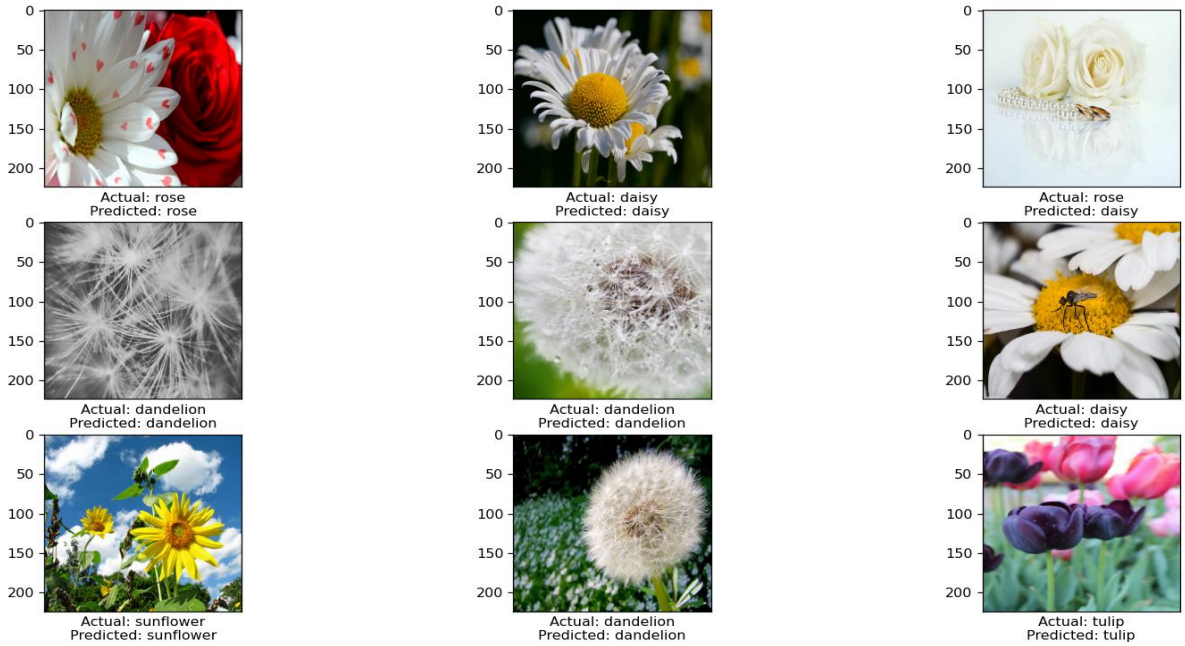
Về lựa chọn “chụp ảnh từ camera”:

1. Người dùng tích vào chế độ cần chọn
2. Sau đó bấm vào chữ chụp ảnh trên giao diện và nó sẽ hiển thị một cửa sổ hình ảnh lên để người nhìn và chụp ảnh
3. Khi muốn chụp ảnh thì người nhấn phím q để chụp lại ảnh và chương trình sẽ tự động tắt camrera và hiển thị ảnh mà người dùng chụp lên chương trình
4. Sau đó người dùng bấm vào nút dự đoán trên giao diện chương trình và chương trình sẽ hiển thị ra kết quả dự đoán lên màn hình

Về lựa chọn “tải ảnh từ file”:

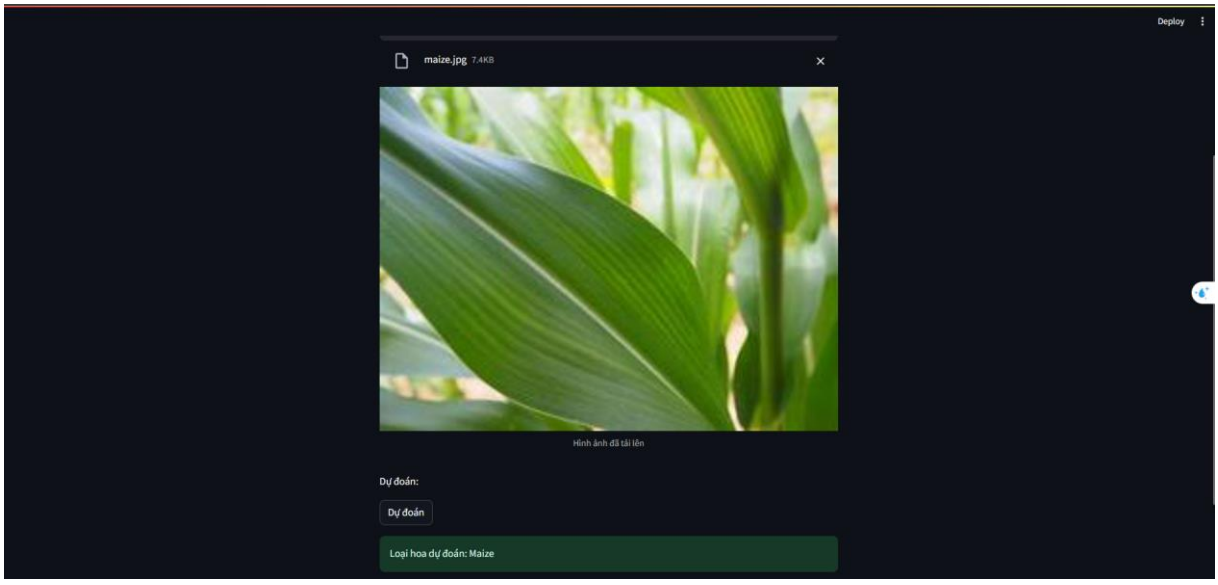
1. Người dùng cần tích vào chế độ cần chọn
2. Sau đó bấm vào nút Browser File trên chương trình nó sẽ hiển thị ra một các thư mục trong máy tính của người dùng
3. Sau khi tìm được ảnh muốn dự đoán thì bấm vào nút dự đoán trên màn hình và chương trình sẽ hiển thị kết quả dự đoán

Đây là hình ảnh kết quả dự đoán khi train

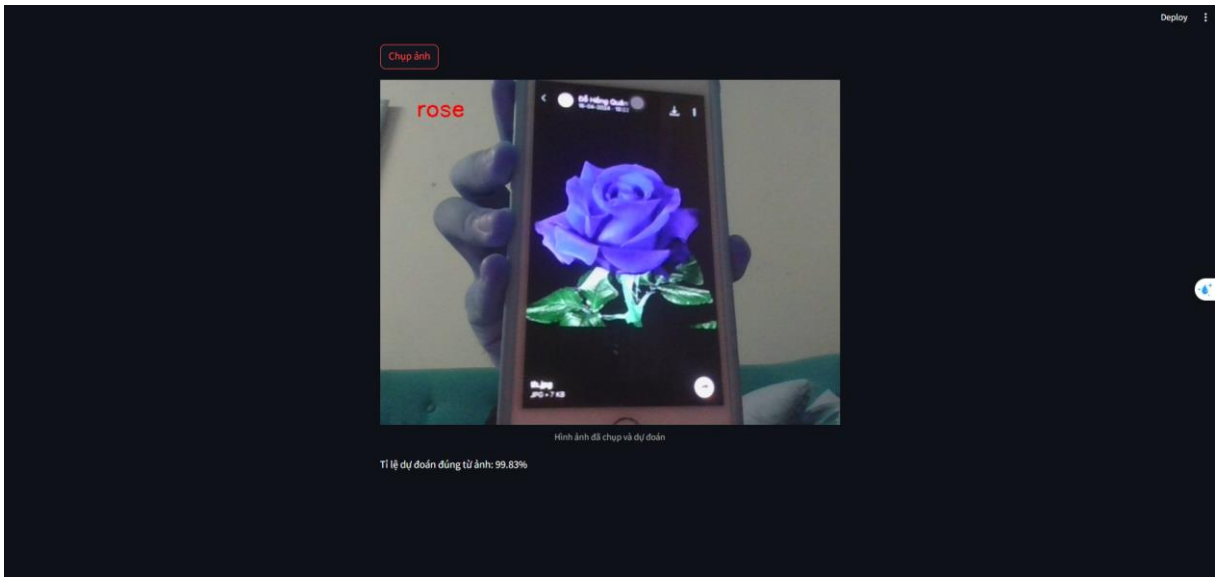


Hình 3.5 Kết quả train

Đây là kết quả của dự đoán trên chương trình với 2 lựa chọn trên



Hình 3.6 Kết quả dự đoán khi dùng file



Hình 3.7 Kết quả dự đoán khi dùng camera

3.4. Đánh giá chương trình

Chương trình trên đã sử dụng một bộ dữ liệu gồm 10345 ảnh bao gồm 5 loài hoa và 5 loài lá. Tỉ lệ dự đoán của chương trình là trên 80%. Tuy nhiện thì khi dự đoán kết quả dự đoán từ camera sẽ không cao bằng dự đoán so với việc lấy ảnh từ file.

Bài cũng dùng 3 mô hình mạng CNN với cùng một input là (220,220,3): một là mạng do em tự xây dựng, hai là mạng Lenet, ba là mạng Alexnet. Từ kết quả đồ thị thì ta có thể thấy là khi trainning thì kết quả của ba mạng sau mỗi lần lặp thì đều sẽ tốt đều lên nhưng khi vào thực tế và dự đoán thì ta có thể thấy được ưu thế rõ rệt của Alexnet qua mỗi lần lặp thì tỉ lệ loss của mạng Alexnet là thấp nhất. Cho nên ta có thể thấy được Alexnet có chiều sâu hơn mạng Lenet và mạng tự làm và nó phù hợp với các loại hình ảnh phức tạp hơn. Alexnet đã sử dụng dropout để Dropout buộc mạng phải học các đặc trưng quan trọng hơn, thay vì chỉ nhớ các mẫu huấn luyện. Điều này giúp tăng tốc độ hội tụ và hiệu quả của quá trình huấn luyện.

Khả năng về việc dự đoán ảnh từ camrera còn chưa tốt một phần là do chất lượng camera còn chưa tốt cũng như các yếu tố về ánh sáng, góc khuất của ảnh, và một số lí do khác.

Nói chung thì chương trình vẫn còn nhiều hạn chế. Đặc biết hạn chế về phạm vi của ảnh khá ít so với thực tế nên một số ảnh còn dự đoán chưa chính xác

3.5. Tiểu kết chương 3

Trong chương 3, em đã tiến hành quá trình triển khai và đánh giá hệ thống một cách chi tiết. Bằng cách đặt ra mục tiêu cụ thể và lựa chọn công nghệ phù hợp, em đã xây dựng hệ thống một cách cẩn thận để đáp ứng yêu cầu. Việc triển khai ba mô hình CNN khác nhau, bao gồm mạng Lenet, một mô hình tự xây dựng và mạng AlexNet, đã giúp em đánh giá sâu hơn về hiệu suất và tính linh hoạt của hệ thống. Đồng thời, việc thiết kế giao diện chương trình dễ sử dụng và việc đánh giá kết quả nhận dạng đã làm nổi bật sự tiện ích và đáng tin cậy của hệ thống trong thực tế. Cuối cùng, việc đánh giá hiệu suất đã cung cấp cái nhìn tổng quan về khả năng hoạt động và ổn định của hệ thống, đóng góp vào việc đảm bảo rằng nó đáp ứng được các tiêu chuẩn và yêu cầu cuối của người dùng.

KẾT LUẬN

Đồ án đã nghiên cứu về thuật toán CNN(Convolutional Neural Network) và đã áp dụng được nó trong bài toán nhận dạng hoa và lá. Mặc dù kết quả đạt được còn chưa cao nhưng cũng đã tìm hiểu được về phương án giải quyết bài toán dựa trên sự tìm kiếm và tiếp cận với một số công trình nghiên cứu và các bài báo về bài toán nhận dạng hình ảnh. Đây là kết quả mà đồ án đã đạt được với những mục tiêu ban đầu như sau:

Hoàn thiện được bộ cơ sở dữ liệu ảnh phục vụ huấn luyện và đã nhận dạng được 5 loại hoa và 5 loại lá, với số lượng hình ảnh trùng bình cho mỗi loại ảnh từ 150- 1000 ảnh

Cài đặt và hoàn thành được 1 mạng nơron tích chập đã được huấn luyện trước, và đã ứng dụng được vào bài toán nhận dạng hoa quả

Thực hiện với bộ dữ liệu test và trong thực tế đã cho kết quả khá tốt và còn một số hạn chế như phạm vi của hoa và lá còn quá ít so với Việt Nam nói riêng cũng như thế giới nói chung. Chương trình tự động nhận dạng hoa và lá còn cần rất nheiuè điều pahri cải thiện đặc biệt là khả năng mở rộng phạm vi các loại hoa và lá.

Đặc biệt với mô hình Alexnet đã cho thấy sự vượt trội trong quá trình huấn luyện. Mô hình Lenet có độ chính xác cao hơn nhưng trong đồ thị thì ta thấy kết quả dự đoán của có cùng với mô hình tự làm có tỉ lệ sai cao chỉ có mô hình Alexnet là thấp cho nên mô hình tự làm vẫn chưa thể thay thế được mà cần phải điều chỉnh thêm.

Các mô hình CNN có độ sâu tốt như Alexnet đã chứng tỏ khả năng vượt trội của nó so vơi mô hình tự làm và Lenet

Hướng phát triển của đề tài

Tăng cường độ chính xác và hiệu suất: Tiếp tục nghiên cứu và phát triển các kiến trúc mạng nơ-ron sâu mới, tối ưu hóa siêu tham số, kết hợp các kỹ thuật regularization để nâng cao độ chính xác trong nhận dạng hoa lá.

Xử lý dữ liệu đa dạng và khó xử lý: Tập trung vào việc xử lý các loại ảnh hoa lá có chất lượng kém, góc chụp khó khăn, điều kiện ánh sáng thay đổi, v.v. Sử dụng kỹ thuật tăng cường dữ liệu, chuyển giao học tập để cải thiện khả năng đối phó với dữ liệu đa dạng.

Ứng dụng trong thực tế: Tích hợp các giải pháp nhận dạng hoa lá dựa trên học sâu vào các ứng dụng thực tế như quản lý vườn ươm, chẩn đoán bệnh cây, phân loại cây trồng, v.v. Điều này đòi hỏi phải có sự tối ưu hóa về hiệu suất, bộ nhớ và thời gian phản hồi.

Kết hợp với các cảm biến và công nghệ khác: Kết hợp các giải pháp nhận dạng hoa lá dựa trên học sâu với các cảm biến như camera, cảm biến nhiệt, ẩm độ, v.v. để tạo ra các hệ thống thông minh hơn và chính xác hơn.

Hiểu biết sinh học và tự động hóa: Nghiên cứu sâu hơn về đặc điểm sinh học của hoa lá để bổ sung vào các mô hình học sâu, từ đó tạo ra các hệ thống tự động hóa hiệu quả hơn.

Ứng dụng các kỹ thuật học sâu tiên tiến: Áp dụng các kỹ thuật học sâu mới như transfer learning, meta-learning, reinforcement learning để cải thiện hiệu suất và khả năng ứng dụng.

Tích hợp vào các nền tảng và hệ thống thông minh: Kết hợp các giải pháp nhận dạng hoa lá dựa trên học sâu vào các nền tảng IoT, hệ thống nông nghiệp thông minh, v.v. để tạo ra các ứng dụng toàn diện hơn.

TÀI LIỆU THAM KHẢO

Danh mục các tài liệu Tiếng Anh:

[1] Aaron Courville, Goodfellow and Yoshua Bengio - Deep Learning released in 2016

[2] Andrew Zisserman and Simonyan - Very deep convolutional networks for large-scale image recognition presented at ILCR2015

[3] Francois Chollet - Deep Learning in Python: Holistic Data Science and Machine Learning with Modern Deep Learning in Python, Theano, and TensorFlow release in 2016

Danh mục các Website tham khảo:

[1] https://stanford.edu/~shervine/l/vi/teaching/cs-230/cheatsheet-convolutional-neural-networks ngày truy cập: tháng 5/2024

[2] https://blog.csdn.net/weixin_41028208/article/details/82805126 ngày truy cập: tháng 5/2024

[3] https://phamdinhkhanh.github.io/2020/05/31/CNNHistory.html ngày truy cập: tháng 5/2024

[4] https://jst-haui.vn/media/30/uffile-upload-no-title30832.pdf ngày truy cập: tháng 5/2024

[5] https://dlapplications.github.io/2018-07-06-CNN/ ngày truy cập: tháng 5/2024

[6] https://pengfeinie.github.io/convolutional-neural-network/ ngày truy cập: tháng 5/2024

[7] https://viblo.asia/p/deep-learning-tim-hieu-ve-mang-tich-chap-cnn-maGK73bOKj2 ngày truy cập: tháng 5/2024

[8] https://www.researchgate.net/figure/Illustration-of-max-pooling-and-average-pooling-11_fig13_355428014 ngày truy cập: tháng 5/2024

[9] https://juejin.cn/post/7117538324398473224 ngày truy cập:tháng 5/2024

[10] https://itgtechnology.vn/machine-learning-la-gi/ ngày truy cập: tháng 5/2024

[11] https://www.mdpi.com/2078-2489/10/12/375 ngày truy cập: tháng 5/2024

[12] https://d2l.aivivn.com/chapter_convolutional-neural-networks/lenet_vn.html ngày truy cập: tháng 5/2024

[13] https://developers.agirobots.com/jp/deep-learning-2020-zfnet/ ngày truy cập: tháng 5/2024

[14] https://mmlab.uit.edu.vn/tutorials/ml/deep-learning/mo_hinh_cnn_cai_tien ngày truy cập: tháng 5/2024