HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

SOULINSOMPHOU OUPALA

NGHIÊN CỨU PHÂN LOẠI ĐỘ TUỔI CỦA NGƯỜI BẰNG HÌNH ẢNH SỬ DỤNG MẠNG NƠ RON TÍCH CHẬP

LUẬN VĂN THẠC SĨ KỸ THUẬT
(Theo định hướng ứng dụng)

Hà Nội-Năm 2020

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

SOULINSOMPHOU OUPALA

NGHIÊN CỨU PHÂN LOẠI ĐỘ TUỔI CỦA NGƯỜI BẰNG HÌNH ẢNH SỬ DỤNG MẠNG NƠ RON TÍCH CHẬP

CHUYÊN NGÀNH: KHOA HỌC MÁY TÍNH
MÃ SỐ: 8.48.01.04

LUẬN VĂN THẠC SĨ KỸ THUẬT
(Theo định hướng ứng dụng)

NGƯỜI HƯỚNG DẪN KHOA HỌC

TS. NGUYỄN ĐÌNH HÓA

Hà Nội-Năm 2020

LỜI CAM ĐOAN

Tôi cam đoan rằng những nghiên cứu phân loại độ tuổi của người bằng hình ảnh sử dụng mạng nơ ron tích chập là công trình nghiên cứu của riêng tôi và chưa từng được nộp như một khóa luận, luận văn hay luận án tại Học viện Công Nghệ Bưu Chính Viễn Thông hoặc bất kỳ trường đại học khác. Những gì tôi viết ra không sao chép từ các tài liệu, không sử dụng các kết quả của người khác mà không trích dẫn cụ thể. Nếu sai tôi hoàn toàn chịu trách nhiệm theo quy định của Học viện Công Nghệ Bưu Chính Viễn Thông.

Tác giả luận văn

SOULINSOMPHOU OUPALA

LỜI CẢM ƠN

Học viên xin chân thành cảm ơn các Thầy Cô trong Khoa Đào tạo Sau Đại học và Khoa Công nghệ thông tin 1, Học Viện Công Nghệ Bưu Chính Viễn Thông đã tạo điều kiện thuận lợi cho học viên trong quá trình học tập và nghiên cứu. Học viên đặc biệt xin chân thành cảm ơn TS. Nguyễn Đình Hóa là người đã trực tiếp tận tình hướng dẫn học viên hoàn thành luận văn này.

Trong quá trình nghiên cứu và thực hiện đề tài với quyết tâm cao nhưng do hạn chế về kinh nghiệm và kiến thức cũng như vốn tiếng việt chưa được phong phú nên luận văn của em chắc chắn sẽ không tránh khỏi những thiếu xót. Em rất mong nhận được ý kiến đóng góp từ quý Thầy Cô và các bạn để đề tài được hoàn thiện hơn

Học viên xin chân thành cảm ơn các bạn bè đã sát cánh giúp học viên có được những kết quả như ngày hôm nay.

Xin chân thành cảm ơn!

Tác giả luận văn

SOULINSOMPHOU OUPALA

MỤC LỤC

DANH MỤC CÁC TỪ VIẾT TẮT

Từ viết tắt	Tiếng Anh	Tiếng Việt
AI	Artificial Intelegent	Trí tuệ nhân tạo
CNN	Convolutional Neural Network	Mạng nơ ron tích chập
CNTT	Information Technology	Công nghệ thông tin
Conv	Convolutional layer	Lớp tích chập
DL	Deep Learning	Học sâu
IMFDB	Indian Movies Face Database	Bộ dữ liệu ảnh khuôn mặt trong phim ấn độ
Lr	Learning rate	Chỉ số học
ML	Machine Learning	Học máy
OCR	Optical Character Recognition	Nhận dạng ký tự quang học

DANH MỤC CÁC BẢNG

DANH MỤC CÁC HÌNH

MỞ ĐẦU

Với sự phát triển phần cứng mạnh mẽ cho phép tính toán song song hàng tỉ phép tính, tạo tiền đề cho Mạng nơ-ron tích chập trở nên phổ biến và đóng vai trò quan trọng trong sự phát triển của trí tuệ nhân tạo nói chung và xử lý ảnh nói riêng. Một trong các ứng dụng quan trọng của mạng nơ-ron tích chập đó là cho phép các máy tính có khả năng "nhìn" và "phân tích" hình ảnh. Phân tích đặc điểm khuôn mặt người luôn là một chủ đề được quan tâm chủ yếu do tính ứng dụng của nó. Hiện nay kỹ thuật Deep Learning là kỹ thuật hiệu quả giúp phân tích những đặc điểm dựa trên khuôn mặt của con người. Trong đó tuổi tác và giới tính, hai trong số các đặc điểm quan trọng, đóng một vai trò rất cơ bản trong các tương tác xã hội. Mặc dù các vai trò cơ bản mà các thuộc tính này đóng góp trong cuộc sống hàng ngày, song khả năng tự động ước tính độ tuổi chính xác và đáng tin cậy từ hình ảnh khuôn mặt người vẫn chưa đáp ứng được nhu cầu của các ứng dụng thương mại.

Phần lớn các doanh nghiệp đều đang gặp vấn đề chung đó là không tìm được đáp án cho các câu hỏi: Làm thế nào để theo dõi hành vi của khách hàng theo độ tuổi?

Từ đó việc ước tính độ tuổi từ một hình ảnh khuôn mặt người là một nhiệm vụ quan trọng trong các ứng dụng thông minh, như kiểm soát truy cập, tương tác giữa người với máy tính, thực thi pháp luật, trí thông minh tiếp thị và giám sát trực quan.

Trong luận văn này em đề xuất xây dựng mô hình kiến trúc mạng nơ-ron tích chập để phân lớp dữ liệu hình ảnh mặt người để dự đoán ra độ tuổi của người đó.

Dựa vào thực trạng như trên kết hợp với các kỹ thuật khai phá dữ liệu đã được học hỏi và nghiên cứu để đưa ra đề tài "***Nghiên cứu phân loại độ tuổi của người bằng ảnh mặt người sử dụng mạng nơ ron tích chập***".

Nội dung của Luận văn được xây dựng thành 3 chương như sau:

Chương 1. Giới thiệu Tổng quan về Bài toán phân loại độ tuổi người bằng hình ảnh, bao gồm tổng quan về về bài toán phân loại ảnh mặt người, các nghiện cứu liên quan và một số ứng dụng thực tế của bài toán phân loại độ tuổi bằng ảnh mặt

người. Chương này cũng trình bày về hướng tiếp cận và giải quyết bài toán theo phương pháp học máy và học sâu, đưa ra những ưu nhược điểm trong từng phương pháp. Từ những cơ sở lý thuyết đó sẽ xác định rõ hướng giải quyết của luận án.

Chương 2. Phân loại độ tuổi của người bằng hình ảnh sử dụng mạng nơ ron tích chập. Trên cơ sở xác định được hướng giải quyết của luận án ở Chương 1, Chương 2 sẽ giới thiệu về mạng nơ ron tích chập và kiến trúc của mạng này trong phương pháp học sâu. Chương này cũng trình bày về các kỹ thuật tiền xử lý dữ liệu đầu vào và việc xây dựng mô hình huấn luyện cho bài toán.

Chương 3. Cài đặt và thử nghiệm. Chương này giới thiệu về bộ dữ liệu được sử dụng trong bài toán, môi trường thực hiện và áp dụng mô hình tốt nhất được xây dựng ở chương 2 vào bộ dữ liệu và đánh giá kết quả phân loại độ tuổi.

CHƯƠNG 1: TỔNG QUAN VỀ BÀI TOÁN PHÂN LOẠI ĐỘ TUỔI CỦA NGƯỜI BẰNG HÌNH ẢNH

1.1. Giới thiệu bài toán phân loại độ tuổi người qua hình ảnh

1.1.1. Tổng quan

Việc phân tích và trích xuất các thông tin có thể có từ các ảnh mặt người đã được các nhà khoa học nghiên cứu từ đầu những năm 90 của thế kỷ trước. Điều này là do có rất nhiều các thông tin có ích có thể khai thác từ một bức ảnh khuôn mặt, ví dụ như danh tính, giới tính, độ tuổi, cảm xúc, cử chỉ tương tác, dân tộc, tình trạng sức khỏe,… Trong số các thông tin có thể suy ra từ ảnh mặt người, độ tuổi là một thuộc tính quan trọng vì nó có khá nhiều ứng dụng trong thực tế, ví dụ như trong tương tác người máy, trong quảng cáo có định hướng, trong thống kê dân số.

Khuôn mặt là một đối tượng trong cơ thể con người và hình ảnh khuôn mặt mang rất nhiều thông tin quan trọng như: tuổi tác, giới tính, trạng thái cảm xúc, dân tộc,... Trong đó, việc xác định tuổi tác và giới tính là hết sức quan trọng, đặc biệt trong giao tiếp, chúng ta cần sử dụng những từ ngữ phù hợp với giới tính của người nghe ví dụ trong tiếng Việt chúng ta có: anh/chị, chú/cô... Hay với nhiều ngôn ngữ khác nhau trên thế giới, chẳng hạn như tiếng Việt thì lời chào hỏi dành cho người lớn tuổi khác với người trẻ tuổi. Do đó, việc xác định tuổi và giới tính dựa trên khuôn mặt là một bài toán hết sức quan trọng, có ý nghĩa thực tế to lớn [1].

Bài toán ước lượng tuổi đã được quan tâm nhiều trong suốt 20 năm gần đây, đã có rất nhiều các công trình được công bố với nhiều kỹ thuật khác nhau chẳng hạn như: Aging pattErn Subspace (AGES), Gaussian Mixture Models (GMM), Hidden-Markov-Model (HMM), Support Vector Machines (SVM). Từ khi các mô hình học sâu được áp dụng cho bài toán này đã cải thiện đáng kể kết quả về mặt hiệu suất cũng như tốc độ. Độ chính xác của mô hình khi ước lượng tuổi đạt 62,8% và đối với giới tính đạt 92,6% [1].

1.1.2. Các nghiên cứu liên quan

Nhiệm vụ của bài toán này là đưa ra ước lượng tuổi của một người từ bức ảnh chụp khuôn mặt của họ. Bài toán này được giới thiệu lần đầu tiên bởi Kwon và Lobo trong đó, họ sử dụng phương pháp phát hiện và tính toán tỷ lệ của các nếp nhăn trên khuôn mặt để có thể dự doán độ tuổi và sau đó nó được cải tiến bởi Ramanathan và Chellappa [4]. Tuy nhiên, phương pháp này có thể phân biệt được độ tuổi giữa người lớn và trẻ em, nhưng rất khó có thể phân biệt được độ tuổi giữa những người lớn với nhau. Một cách tiếp cận khác do Geng cùng các cộng sự [7] trình bày là sử dụng AGES cho hiệu quả cao hơn nhưng thuật toán này cần một lượng lớn hình ảnh khuôn mặt của từng người và đặc biệt hình ảnh đầu vào này cần phải ở chính giữa, mặt hướng thẳng và được căn chỉnh đúng kích thước. Tuy nhiên, trên thực tế thì các bức ảnh chụp lại rất ít khi thỏa mãn điều kiện như vậy do đó cách tiếp cận này không được phù hợp với nhiều ứng dụng thực tế. Một cách tiếp cận khác dựa trên các thuật toán thống kê đã được sử dụng như GMM và HMM, super-vectors được sử dụng để làm đại diện cho từng phần của khuôn mặt. Trong thập kỷ qua, khi các thuật toán học máy dần được cải tiến và đạt được thành tựu to lớn đặc biệt là học sâu, thì một loạt các công trình nghiên cứu về phân lớp tuổi được công bố cho kết quả khả quan, có thể kể đến như: Eidinger cùng các cộng sự đã sử dụng SVM kết hợp với dropout cho bài toán nhận diện tuổi và nhận diện giới tính. Năm 2015, GilLevi và Tal Hassner đã đưa ra mô hình Deep Neural Network đầu tiên cho bài toán phân lớp tuổi và giới tính. Sau đó, Zhu cùng các cộng sự [7] đã xây dựng một mô hình đa nhiệm vụ cho phép chia sẻ và tìm hiểu các tính năng tối ưu để cải thiện hiệu suất nhận dạng.

Năm 2015, dựa trên công nghệ nhận diện khuôn mặt, website how-old.net do Microsoft xây dựng sẽ đưa ra dự đoán về tuổi và giới tính của người trong bức hình. Face API của Microsoft sử dụng phương pháp học máy truyền thống để đoán độ tuổi. Đầu tiên là xác định khu vực khuôn mặt. Quá trình này chủ yếu dựa vào việc có tìm ra được vị trí đôi mắt. Sau khi quá trình nhận diện hoàn tất, Face API mới chia tách các vị trí trên khuôn mặt ra để xác định độ tuổi. Ví dụ tách ra vị trí của tròng đen,

đuôi mắt, chân mày, bờ môi, các nếp nhăn... Chương trình cũng chạy thử một số phép giả lập "tuổi già" bằng cách thay đổi màu sắc hoặc các đường nét trên khuôn mặt. Từ đó sẽ đưa ra con số tuổi được đặt phía trên mỗi khuôn mặt.

Dịch vụ đoán tuổi của Microsoft hoạt động khá tốt trong nhiều trường hợp nhất định nhưng cũng không ít lần cho ra sai số rất lớn. Hầu hết kết quả trả về độ tuổi không hoàn toàn chính xác. Lý do thứ nhất là vì những ảnh mà người dùng tải lên không hiển thị đầy đủ khuôn mặt, chỉ một phần hoặc không chụp ở góc chính diện. Cũng có thể do điều kiện ánh sáng khi chụp không thể hiện chính xác màu da thực tế của người đó. Chẳng hạn tấm ảnh chụp trong điều kiện đủ sáng, tươi cười thì kết quả có thể đúng hoặc trẻ hơn vài tuổi và ngược lại.

1.2. Khó khăn và thách thức

Để nhận biết một vật thể, động vật hoặc một khuôn măt là một việc tương đối dễ dàng với con người, những chúng ta hãy xem xét với góc nhìn của một máy tính hoặc một thuật toán nó là vấn đề khá phức tạp. Khó khăn và thách thức đối với bài toán phân loại độ tuổi gồm có:

Chất lượng và sự đa dạng của dữ liệu

Dữ liệu hình ảnh khuôn mặt người có thể bao gồm rất nhiều loại hình ảnh khác nhau phụ thuộc vào nhiều điều kiện như : nguồn dữ liệu, phương pháp thu thập dữ liệu, công cụ sử dụng trong việc thu thập v.v. Nên có thể dẫn đến chất lượng của bộ dữ liệu không được cao, dữ liệu bị thiếu sót hoặc không đồng đều, đặc biệt là với dữ liệu hình ảnh có thể sẽ có những trường hợp như ảnh bị nhiễu, mờ, thiếu ánh sáng hoặc ánh sang quá mức, đối tượng trong hình ảnh bị che hoặc không chụp đúng góc nhìn. Những vấn đề này đều ảnh hưởng đến chất lượng và khả năng phân lớp của mô hình. Ví dụ các vấn đề có thể gặp phải trong các bài toán phân loại ảnh [3]:

Góc nhìn đa dạng: Một vật thể có thể được chụp lại với vị trí chụp khác nhau dẫn đến có nhiều góc nhìn khác nhau.

Biến đổi về tỷ lệ: Tỷ lệ, kích thước của đối tượng có thể thay đổi tùy theo góc chụp gần hay xa, kích thước của đối tượng trong bức ảnh cũng nhỏ hơn rất nhiều kích thước của đối tượng trong thế giới thực.

Biến dạng: Một số đối tượng có những tư thế, hành động đặc biệt mà bằng mắt thường chúng ta cũng khó có thể nhận dạng đối tượng đó. Do đó khi có những bức ảnh biến dạng của đối tượng làm đầu vào trong quá trình huấn luyện mô hình cũng sẽ gặp nhiều khó khăn. Ngoài ra, có những bức ảnh chúng ta chỉ nhìn thấy một phần nhỏ của đối tượng, điều này cũng làm cho khả năng nhận dạng đối tượng bị hạn chế.

Điều kiện chiếu sáng: Các đối tượng trong bức ảnh cũng bị ảnh hưởng bởi điều kiện chiếu sáng. Các đối tượng cần quan tâm có thể bị hòa trộn vào môi trường của chúng khiến chúng khó xác định.

Dữ liệu mất cân bằng (Imbalanced Data)

Bộ dữ liệu mất cân bằng (Imbalanced dataset) là tập dữ liệu có tỷ lệ của số mẫu của từng nhãn phân loại không bằng nhau. Ví dụ, một tập dữ liệu với các bộ dữ liệu y tế mà phải phát hiện một số bệnh thường sẽ có nhiều mẫu âm tính hơn mẫu dương tính, ví dụ: 98% hình ảnh không có bệnh và 2% hình ảnh bị bệnh.

Mô hình được tạo bằng cách sử dụng dữ liệu mất cân bằng có thể nguy hiểm. Hãy tưởng tượng dữ liệu đào tạo của chúng ta là dữ liệu được minh họa trong biểu đồ ở trên. Nếu độ chính xác được sử dụng để đo lường độ tốt của mô hình, mô hình phân loại tất cả các mẫu thử thành "0" sẽ có độ chính xác tuyệt vời (99,8%), nhưng rõ ràng, mô hình này sẽ không cung cấp bất kỳ thông tin giá trị nào cho chúng ta.

Hiệu năng máy tính sử dụng trong quá trình huấn luyện mô hình

Với một số bài toán để đưa ra được mô hình có khả năng phân loại hoặc dự đoán có độ chính xác cao, đưa ra kết quả dự đoán nhanh chóng, Tuy nhiên, để đạt được hiệu suất cao thì các mô hình thường được xây dựng càng phức tạp với số lượng

tham số lớn (từ 10 triệu đến hơn 100 triệu tham số), do đó gây khó khăn trong vấn đề nhận dạng trong thời gian thực và tại đó dẫn đến vấn đề hiệu năng máy tính và chi phí tính toán của máy tính (Computational cost) khi xây dựng và áp dụng mô hình.

1.3. Hướng tiếp cận và giải quyết bài toán

Phương pháp giải quyết bài toán này có thể được phân làm hai loại phương pháp học, là Phương pháp học máy truyền thống và Phương pháp học sâu.

1.3.1. Phương pháp học máy truyền thống

Machine Learning là một lĩnh vực nhỏ của Khoa Học Máy Tính, nó có khả năng tự học hỏi dựa trên dữ liệu đưa vào mà không cần phải được lập trình cụ thể. Có thể học hỏi và tự sửa đổi mà không cần sự can thiệp của con người để tạo ra đầu ra mong muốn - bằng cách tự cung cấp thông qua dữ liệu có cấu trúc.

Machine learning theo định nghĩa cơ bản là ứng dụng các thuật toán để phân tích cú pháp dữ liệu, học hỏi từ nó, và sau đó thực hiện một quyết định hoặc dự đoán về các vấn đề có liên quan. Vì vậy, thay vì code phần mềm bằng cách thức thủ công với một bộ hướng dẫn cụ thể để hoàn thành một nhiệm vụ cụ thể, máy được "đào tạo" bằng cách sử dụng một lượng lớn dữ liệu và các thuật toán cho phép nó học cách thực hiện các tác vụ.

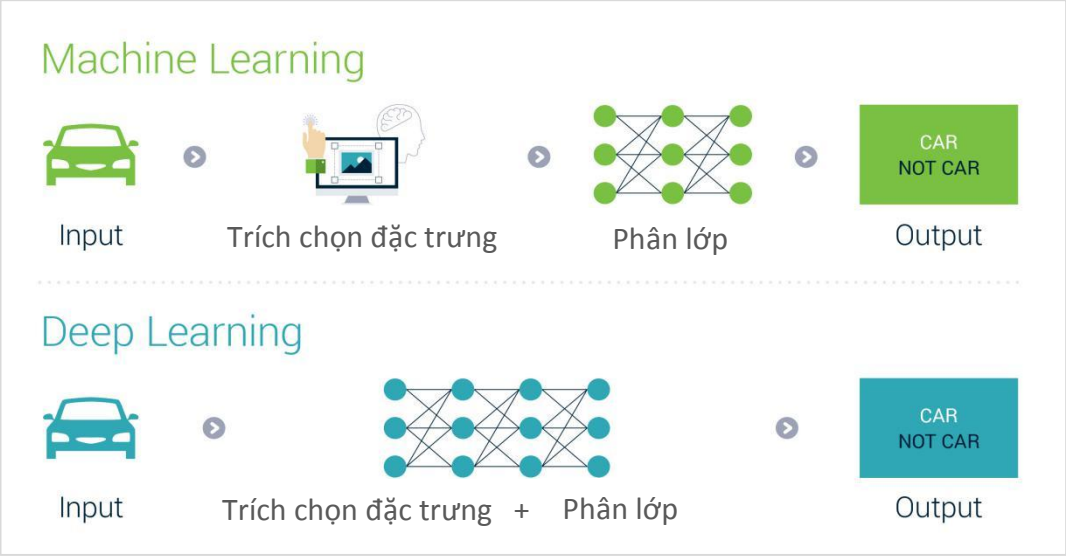
Machine learning bắt nguồn từ các định nghĩa về AI ban đầu, và các phương pháp tiếp cận thuật toán qua nhiều năm bao gồm: logic programming, clustering, reinforcement learning, and Bayesian networks. Như chúng ta đã biết, không ai đạt được mục tiêu cuối cùng của General AI, và thậm chí cả Narrow AI hầu hết là ngoài tầm với những phương pháp tiếp cận Machine learning sơ khai.

Trong mô hình học máy truyền thống bước trích xuất đặc trưng của dữ liệu ảnh hưởng lớn đến độ chính xác của mô hình phân lớp, để trích xuất được đặc trưng tốt chúng ta cần phải phân tích dữ liệu khá chi tiết và cần cả những kiến thức chuyên gia trong từng miền ứng dụng cụ thể.

1.3.2. Phương pháp học sâu

Học sâu (Deep Learning) hay viết tắt DL là một thuật toán dựa trên một số ý tưởng từ não bộ tới việc tiếp thu nhiều tầng biểu đạt, cả cụ thể lẫn trừu tượng, qua đó làm rõ nghĩa của các loại dữ liệu. DL được ứng dụng trong nhận diện hình ảnh, nhận diện giọng nói, xử lý ngôn ngữ tự nhiên Hiện nay rất nhiều các bài toán nhận dạng sử dụng DL để giải quyết do DL có thể giải quyết các bài toán với số lượng lớn, kích thước đầu vào lớn với hiệu năng cũng như độ chính xác vượt trội so với các phương pháp phân lớp truyền thống.

Những năm gần đây, khi mà khả năng tính toán của các máy tính được nâng lên một tầm cao mới và lượng dữ liệu khổng lồ được thu thập bởi các hãng công nghệ lớn, Machine Learning đã tiến thêm một bước dài và một lĩnh vực mới được ra đời gọi là DL (Học Sâu). DL đã giúp máy tính thực thi những việc tưởng chừng như không thể vào 10 năm trước: phân loại cả ngàn vật thể khác nhau trong các bức ảnh, tự tạo chú thích cho ảnh, bắt chước giọng nói và chữ viết của con người, giao tiếp với con người, hay thậm chí cả sáng tác văn hay âm nhạc [17].



Hình 1.1 So sánh phương pháp học máy với phương pháp học sâu [17]

Hình 1.1 cho thấy sự tương quan giữa học sâu với các hệ thống học cổ điển và dựa trên luật. Với các hệ thống học dựa trên luật thì các luật và đặc trưng được rút trích thủ công. Với các hệ thống học cổ điển, ví dụ như học cây quyết định, các đặc

trưng được trích thủ công, mô hình học là tự động. Học sâu được xem là một phần của học biểu diễn (representation learning) với đặc trưng và mô hình học đều tự động, nhưng các đặc trưng được học bằng nhiều tầng học khác nhau.

Hiện nay các mô hình học sâu (Deep Learning) tiêu biểu như mô hình mạng nơ-ron tích chập (Convolutional Neural Networks - CNN) được ứng dụng thành công trong bài toán phân lớp ảnh, văn bản, nhận dạng tiếng nói. Ưu điểm của các mô hình học sâu là tự động học các đặc trưng của dữ kiệu để thiết lập các đặc trưng mới và phân lớp dữ liệu.

Convolutional Neural Network (CNN – Mạng nơ-ron tích chập) là một trong những mô hình DL tiên tiến giúp cho chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao như hiện nay. Các số lượng tham số được sử dụng trong mạng CNN được chứng minh là nhỏ hơn rất nhiều so với mạng nơ ron nhân tạo thông thường và hiểu quả mạng lại thường là cao hơn nhiều so với các phương pháp trước đó. Trong luận văn cao học này, em đi vào nghiên cứu về mạng nơ ron tích chập cũng như ý tưởng của mô hình CNN trong phân lớp ảnh (Image Classification), và áp dụng trong việc xây dựng mô hình phân loại độ tuổi người bằng hình ảnh.

1.4. Kết chương

Trong chương I, luận văn đã trình bày tổng quan về bài toán phân loại độ tuổi qua ảnh mặt người, những ứng dụng của bài toán trong thực tế và hướng tiếp cận giải quyết bài toán dựa trên phương pháp học sâu sử dụng mạng nơ ron tích chập CNN.

CHƯƠNG 2: PHÂN LOẠI ĐỘ TUỔI CỦA NGƯỜI BẰNG HÌNH ẢNH SỬ DỤNG MẠNG NƠ RON TÍCH CHẬP

2.1. Giới thiệu về mạng nơ ron tích chập

CNN là một trong những mô hình DL tiên tiến giúp cho chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao như hiện nay. CNN được lấy cảm hứng từ vỏ não thị giác của con người, mỗi khi chúng ta nhìn thấy một vật nào đó, một loại các lớp tế bào thần kinh được kích hoạt, và mỗi lớp sẽ phát hiện ra một đặc trưng của đồ vật đó (hình dạng, kích thước, màu sắc,…). Lớp thần kinh mà nhận dạng được càng nhiều đặc điểm của đồ vật thì việc nhận dạng hoặc phân loại đồ vật đó đối với con người sẽ trở nên dễ dàng hơn [10].

Ý tưởng đằng sau của mạng nơ ron tích chập là nó thực hiện quá trình trích lọc hình ảnh trước khi đưa vào quá trình huấn luyện, sau quá trình trích lọc thì chúng ta sẽ nhận được các đặc trưng trong hình ảnh đó, và từ các đặc trưng đó chúng ta có thể phát hiện ra những gì mình muốn trong hình ảnh đó.

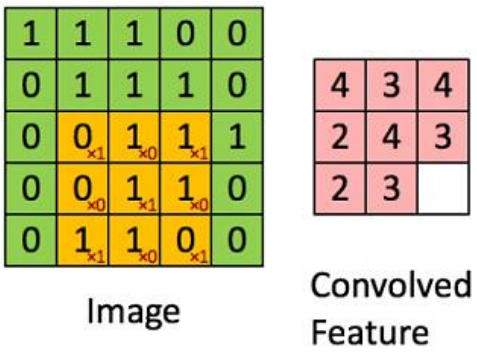
So với các thuật toán phân loại hình ảnh khác, mạng nơ ron tích chập sử dụng quá trình tiền xử lý tối thiểu, nghĩa là mạng học các bộ lọc thường được thiết kế bằng tay trong các hệ thống khác. Bởi vì CNN hoạt động với sự độc lập như vậy khỏi nỗ lực của con người, chúng mang lại nhiều lợi thế hơn các thuật toán khác.

Mục đích của CNN là giảm hình ảnh thành một hình thức dễ xử lý hơn và không mất đi các chi tiết hoặc tính năng quan trọng để hỗ trợ trong việc đưa ra các dự đoán. Điều này rất quan trọng khi chúng ta thiết kế mô hình không chỉ giỏi về các tính năng học tập mà còn xử lý được bộ dữ liệu lớn.

Trước khi tìm hiểu về kiến trúc, mô hình của mạng nơ ron tích chập CNN em sẽ trình bày những khái niệm thường được sử dụng khi làm việc với mạng nơ ron CNN.

a. Tích chập (Convolutional)

Tích chập được sử dụng đầu tiên trong xử lý tín hiệu số (Signal processing). Nhờ vào nguyên lý biến đổi thông tin, các nhà khoa học đã áp dụng kĩ thuật này vào xử lý ảnh và video số. Để dễ hình dung, ta có thể xem tích chập như một cửa sổ trượt (sliding window) áp đặt lên một ma trận. Bạn có thể theo dõi cơ chế của tích chập qua hình minh họa bên dưới.

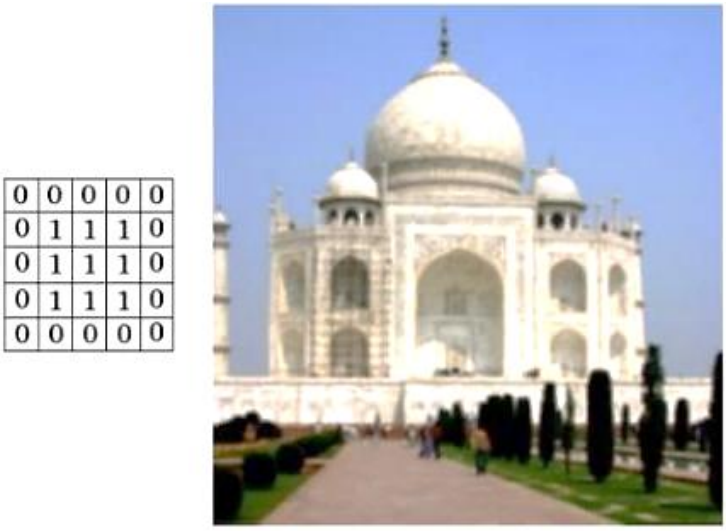


Hình 2.1 Minh học phép toán tích chập [10]

Ma trận bên trái là một bức ảnh đen trắng. Mỗi giá trị của ma trận tương đương với một điểm ảnh (pixel), 0 là màu đen, 1 là màu trắng (nếu là ảnh grayscale thì giá trị biến thiên từ 0 đến 255). Sliding window còn có tên gọi là kernel, filter hay feature detector. Ở đây, ta dùng một ma trận filter 3×3 nhân từng thành phần tương ứng (element-wise) với ma trận ảnh bên trái. Gía trị đầu ra do tích của các thành phần này cộng lại. Kết quả của tích chập là một ma trận (convoled feature) sinh ra từ việc trượt ma trận filter và thực hiện tích chập cùng lúc lên toàn bộ ma trận ảnh bên trái.

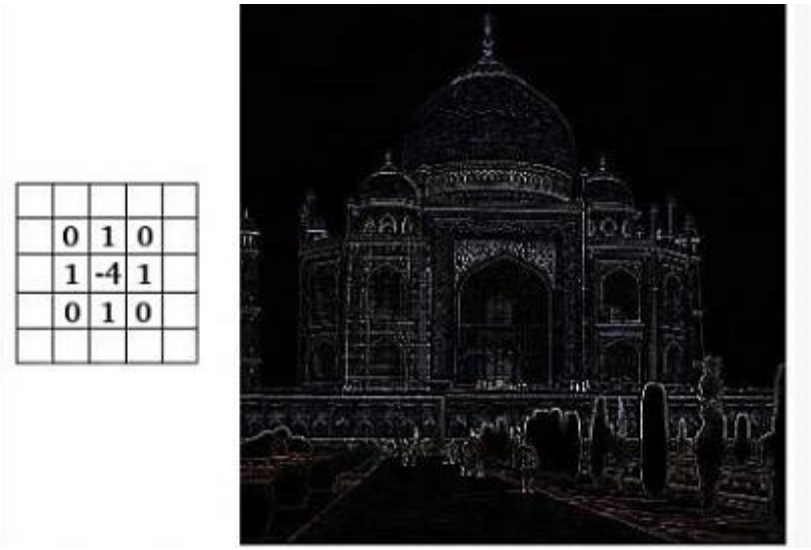
Dưới đây là một vài ví dụ của phép toán tích chập.

Ta có thể làm mờ bức ảnh ban đầu bằng cách lấy giá trị trung bình của các điểm ảnh xung quanh cho vị trí điểm ảnh trung tâm.



Hình 2.2 Minh họa phép tích chập với bộ lọc [14]

Ngoài ra, ta có thể phát hiện biên cạnh bằng cách tính vi phân (độ dị biệt) giữa các điểm ảnh lân cận.



Hình 2.3 Minh họa phép tích chập với bộ lọc cạnh [14]

b. Lớp tích chập (Convolutional Layers)

Lớp tích chập được dung để phát hiện và trích xuất đặc trưng – chi tiết của ảnh. Giống như các lớp ẩn khác, lớp tích chập lấy dữ liệu đầu vào, thực hiện các phép chuyển đổi để tạo ra dữ liệu đầu vào cho lớp kế tiếp (đầu ra của lớp này là đầu vào của lớp sau). Phép biến đổi được sử dụng là phép tính tích chập. Mỗi lớp tích chập

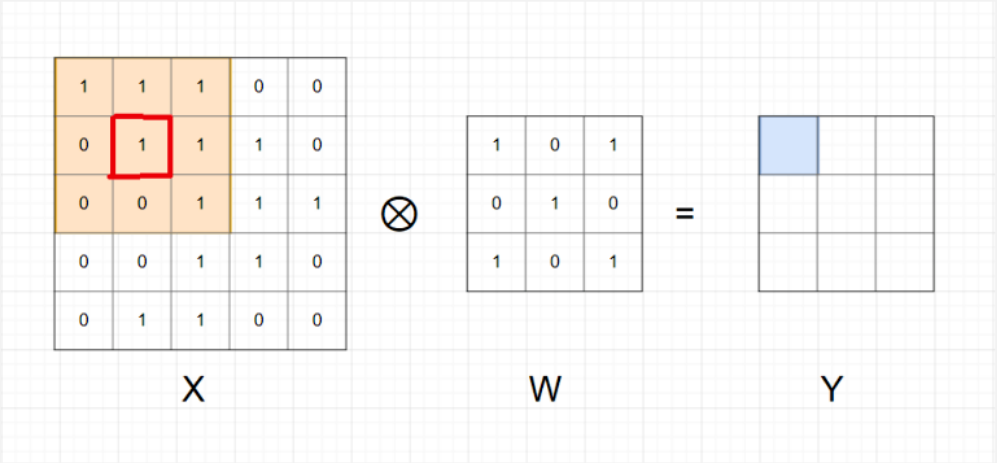
chứa một hoặc nhiều bộ lọc - bộ phát hiện đặc trưng (filter - feature detector) cho phép phát hiện và trích xuất những đặc trưng khác nhau của ảnh.

Đặc trưng ảnh là những chi tiết xuất hiện trong ảnh, từ đơn giản như cạnh, hình khối, hình tam giác, chữ viết tới phức tạp như mắt, mặt, chó, mèo, bàn, ghế, xe, đèn giao thông. Bộ lọc phát hiện đặc trưng của ảnh là bộ lọc giúp phát hiện và trích xuất các đặc trưng của ảnh, có thể là bộ lọc góc, cạnh, đường chéo, hình tròn, hình vuông, v.v.

c. Bộ lọc (Kernel/Filter)

Độ phức tạp của đặc trưng được phát hiện bởi bộ lọc tỉ lệ thuận với độ sâu của lớp tích chập mà nó thuộc về. Nghĩa là bộ lọc ở lớp tích chập càng sâu thì phát hiện các đặc trưng càng phức tạp. Trong mạng CNN, những lớp tích chập đầu tiên sử dụng bộ lọc hình học (geometric filters) để phát hiện những đặc trưng đơn giản như cạnh ngang, dọc, chéo của bức ảnh. Những lớp tích chập sau đó được dùng để phát hiện đối tượng nhỏ, bán hoàn chỉnh như mắt, mũi, tóc, v.v. Những lớp tích chập sâu nhất dùng để phát hiện đối tượng hoàn hỉnh như: chó, mèo, chim, ô tô, đèn giao thông, v.v.

Mục đích của việc tích chập (Convolutional) là để lấy ra được các hình dạng (pattern) trong hình ảnh bằng cách sử dụng các bộ lọc (Filter/Kernel) [2]. Kernel có thể được coi là tham số của mô hình CNN và được sử dụng để tính toán tích chập (convolve) trên ảnh. Chúng ta có thể thấy thao tác tích chập được mô tả trong hình dưới (Hình 2.1).



Hình 2.4 Bộ lọc W (kernel) [10]

Ta định nghĩa kernel là một ma trận vuông kích thước k*k trong đó k là số lẻ. k có thể bằng 1, 3, 5, 7, 9,… Ví dụ kernel kích thước 3*3

$$W = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

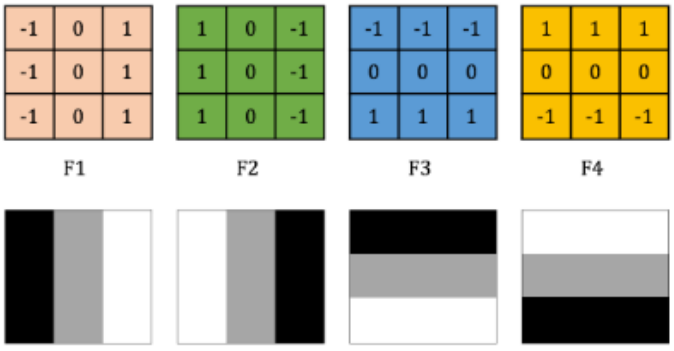
Kí hiệu phép tính tích chập (⊗), kí hiệu Y = X ⊗W, Y = X ⊗W. Với mỗi phần tử $x_{\{ij\}}x_{ij}$ trong ma trận **X** lấy ra một ma trận có kích thước bằng kích thước của bộ lọc ***W*** có phần tử $x_{\{ij\}}$ x_{ij} làm trung tâm (đây là vì sao kích thước của kernel thường lẻ) gọi là ma trận **A**. Sau đó tính tổng các phần tử của phép tính element-wise của ma trận **A** và ma trận ***W***, rồi viết vào ma trận kết quả **Y**.

Tức mỗi phần của image sẽ được nhân tích chập với kernel để tạo thành một ma trận mới - làm đầu vào cho lớp tiếp theo. Một kernel có hai tham số cần quan tâm đến đó là stride và size. Size là kích thước của một kernel (có thể là kích thước của một hình chữ nhật bất kì) và stride là số bước nhảy của kernel. Nếu stride bằng 1 thì gần như toàn bộ pixel trên ảnh sẽ được trượt qua và tính tích chập. Nếu stride bằng 2 chúng ta cứ cách 2 pixel lại tính tích chập một lần và như vậy số lượng pixel của ảnh đầu ra bị giảm đi một nửa so với stride = 1.

Để hiểu cách thức hoạt động của lớp tích chập cũng như phép tính tích chập, hãy cùng xem ví dụ về bộ lọc phát hiện cạnh (edge filters/ detectors) dưới đây.

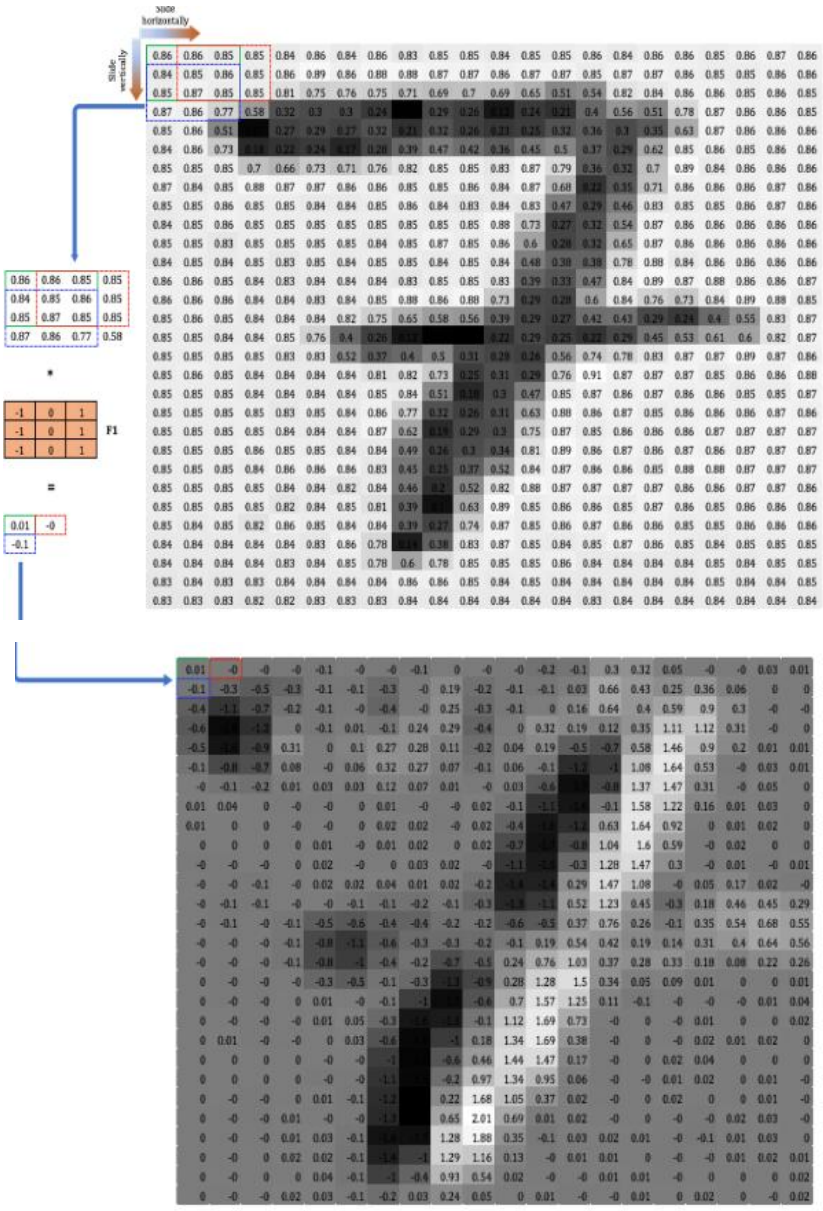
Ví dụ về bộ lọc cạnh

Trong ví dụ này, CNN được sử dụng để phân loại tập các ảnh viết tay của các số từ 00 tới 99. Đầu vào là những bức ảnh trắng đen (Gray Scale) và được biểu diễn bởi một ma trận các điểm ảnh với kích thước cố định h × wh × w. Lớp tích chập đầu tiên của CNN sử dụng 44 bộ lọc kích thước 3×33×3: F1F1, F2F2, F3F3, F4F4 với giá trị tương ứng như trong hình 1. Các giá trị tại mỗi ô của các bộ lọc có thể được biểu diễn bởi màu sắc tương ứng với Đen (−1−1), Xám (00), Trắng (11) như trong hình dưới đây.



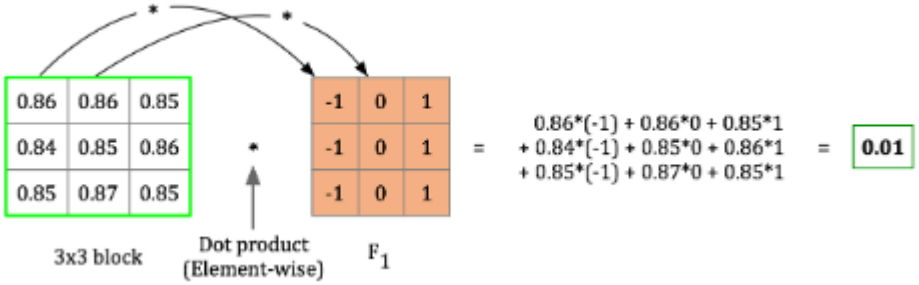
Hình 2.5 Các Bộ lọc cạnh với kích thước 3 x 3 [10]

Để minh hoạ cho phép nhân chập, chúng ta sử dụng đầu vào là một bức ảnh viết tay của số 77, biểu diễn dưới dạng ma trận 30×2230×22 và áp dụng riêng biệt từng bộ lọc ở trên. Phép nhân tích chập được thực hiện bằng cách trượt ma trận lọc 3×33×3 trên ma trận ảnh đầu vào 32×2232×22 (bộ lọc dịch sang phải/ xuống dưới 11 cột/ hàng mỗi một lần trượt) cho đến khi nó đi qua hết tất cả các vùng kích thước 3×33×3. Việc trượt ma trận lọc trên ma trận đầu vào được gọi là "chập" (convoling). Như minh hoạ trong hình 2, ma trận F1F1 được chập với từng vùng (block - region) điểm ảnh kích thước 3×33×3 của ảnh đầu vào. Tại mỗi vị trí di chuyển của ma trận F1F1, giá trị đầu ra được tính bằng tích chập (dot-product) của ma trận F1F1 với vùng bao phủ tương ứng.



Hình 2.6. Minh họa phép nhân chập với bộ lọc cạnh

Ô đầu tiên (0, 0) (0, 0) của ma trận đầu ra (giá trị 0.010.01) ra là kết quả của phép nhân chập giữa ma trận F1F1 với góc trái trên cùng của ma trận đầu vào và được tính như sau:



Hình 2.7. Kết quả của phép tích chập với bộ lọc cạnh [10]

Từ các ma trận đầu ra kích thước 28×2028×20, chúng ta thấy được cả bốn bộ lọc F1F1, F2F2, F3F3 và F4F4 dều được sử dụng để phát hiện cạnh trong bức ảnh (thể hiện bởi những điểm ảnh sáng hơn) (Hình 2.8):

F1: Phát hiện cạnh đứng phải.

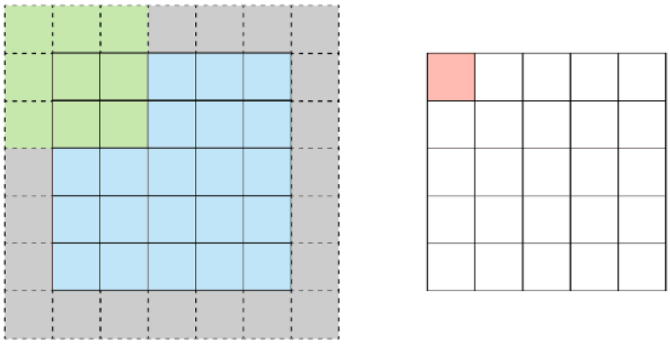
F2: Phát hiện cạnh đứng trái.

F3: Phát hiện cạnh ngang dưới.

F4: Phát hiện cạnh ngang trên.

Hình 2.8. Ví dụ về bộ lọc cạnh

Trong bước thực hiện tích chập của màng còn có thêm hai thuộc tính đấy là stride và padding. Stride là khoảng cách giữa 2 kernel khi thực hiện quét. Với stride = 1, kernel sẽ quét 2 ô ngay cạnh nhau, nhưng với stride = 2, kernel sẽ quét ô số 1 và ô số 3. Bỏ qua ô ở giữa. Điều này nhằm tránh việc lặp lại giá trị ở các ô đã quét. Chúng ta chọn thông số của stride và của kernel càng lớn thì size của feature map càng nhỏ, một phần lý do đó là bởi kernel phải nằm hoàn toàn trong input. Có một cách để giữ nguyên kích cỡ của feature map so với ban đầu. Đấy là Padding. Khi ta điều chỉnh padding = 1, là thêm 1 vùng điểm ảnh xung quanh viền của hình ảnh đầu vào (Hình 2.9), muốn phần viền xung quanh càng dày thì ta cần phải tăng giá trị padding lên.



Hình 2.9 Phép tích chập với giá trị padding bằng 1 [11]

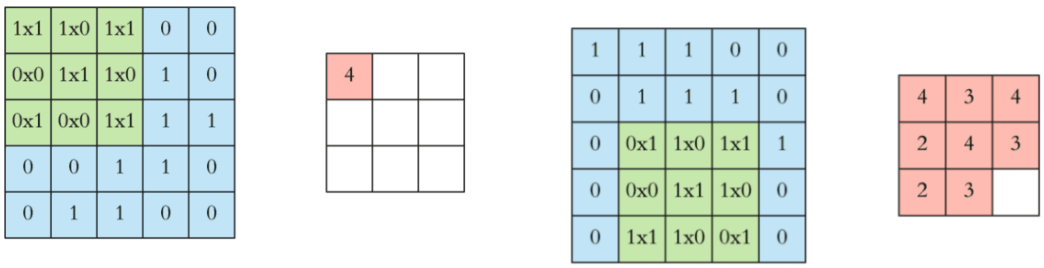
d. Feature map

Tích chập là một khối quan trọng trong CNN. Thuật ngữ tích chập được dựa trên một phép hợp nhất toán học của hai hàm tạo thành hàm thứ ba. Phép toán này kết hợp hai tập thông tin khác nhau.

Trong trường hợp CNN, tích chập được thực hiện trên giá trị đầu vào của dữ liệu và bộ lọc (Kernel/ filter thuật ngữ này được sử dụng khác nhau tùy tình huống) để tạo ra một bản đồ đặc trưng (feature map) [2].

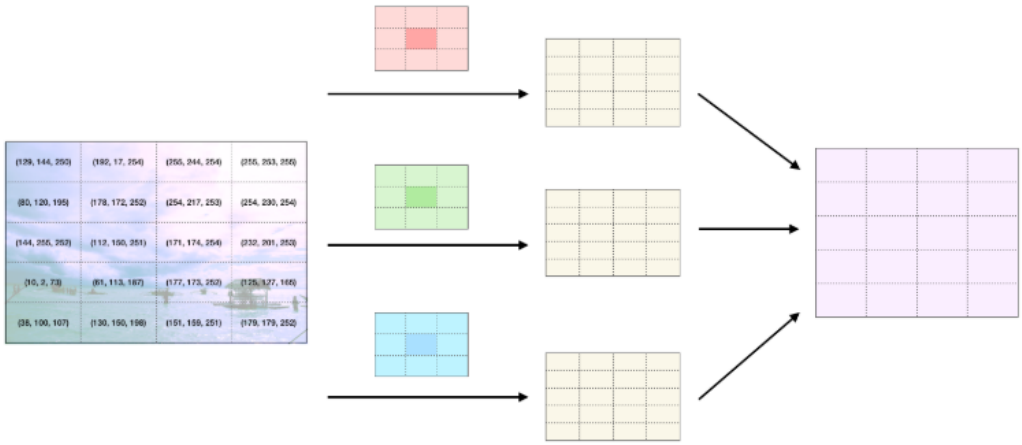
Ta thực hiện phép tích chập bằng cách trượt bộ lọc theo dữ liệu đầu vào. Tại mỗi vị trí, ta tiến hành phép nhân ma trận và tính tổng các giá trị để đưa vào bản đồ đặc trưng.

Trong hình dưới đây, thành phần bộ lọc (màu xanh lá) trượt trên đầu vào (màu xanh dương) và kết quả được trả về bản đồ đặc trưng (màu đỏ). Bộ lọc có kích thước là 3×3 trong ví dụ này.



Hình 2.10 Phép tích chập trên hình ảnh với một giải màu [11]

Đấy là trong trường hợp hình ảnh với một giải màu hoặc là ảnh xám, còn trường hợp quan trọng cần xem xét là cách mà phép tích chập được thực hiện trên hình ảnh màu. Điểm ảnh trong ảnh màu có ba giá trị tương ứng với ba giải màu - giá trị đỏ, lục và lam. Do đó, nếu chúng ta muốn chạy một phép tích chập trên một hình ảnh màu, trước tiên nó phải chia thành các thành phần màu đỏ, xanh lục và xanh lam và thực hiện chạy một bộ lọc trên từng giải dữ liệu đỏ, một trên màu xanh lục và một trên màu xanh lam và tổng hợp tất cả các kết quả.



Hình 2.11 Phép tích chập trên hình ảnh màu [11]

Chúng ta thực hiện phép tích chập trên đầu vào nhiều lần khác nhau. Mỗi lần sử dụng một bộ lọc khác nhau. Kết quả ta sẽ thu được những bản đồ đặc trưng khác nhau. Cuối cùng, ta kết hợp toàn bộ bản đồ đặc trưng này thành kết quả cuối cùng của tầng tích chập. Từ đó phát hiện ra bộ lọc nào cho ra kết quả tương ứng với lớp phân loại hiệu quả nhất. Đối với bài toán tương tụ chúng ta thường gọi kết quả của quá trình tích chập là feature map, trọng số xác định các đặc trưng là shared weight và độ lệch xác định một feature map là shared bias.

2.2. Cấu trúc mạng nơ ron tích chập cùng một số mô hình mạng thông dụng trên thực tế

Cấu trúc của một mạng nơ ron tích chập thường sẽ bao gồm các thành phần như:

- Convolution layer
- Activaion layer

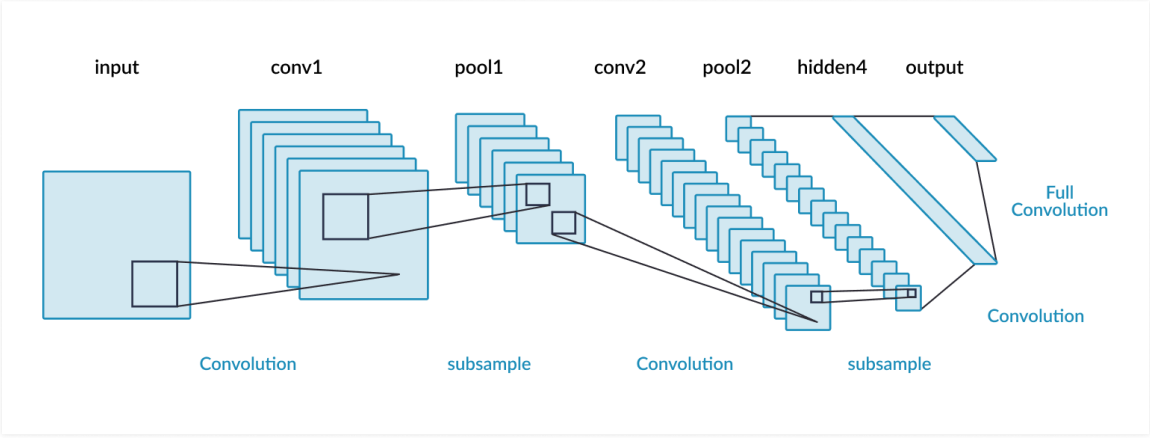
- Poolling layer
- Flatten layer
- Fully connected layer

Nếu chia theo các loại tầng thì CNN gồm hai thành phần:

Phần tầng ẩn hay phần rút trích đặc trưng: trong phần này, mạng sẽ tiến hành tính toán hàng loạt phép **tích chập** và phép **hợp nhất** (pooling) để phát hiện các đặc trưng. Ví dụ: nếu ta có hình ảnh con ngựa vằn, thì trong phần này mạng sẽ nhận diện các sọc vằn, hai tai, và bốn chân của nó.

Mỗi tầng trong các tầng ẩn tăng cường độ chi tiết và độ phức tạp trong quá trình nhận diện đặc trưng của hình ảnh ví dụ như tầng đầu tiên huấn luyện để phát hiện biên hoặc cạnh của hình ảnh và tầng cuối cùng huấn luyện để phát hiện hình dạng phức tạp hơn như hình tam giác, hình tròn, đôi mắt, mũi, lốp xe. v.v. Các nơ ron trong tầng cuối cùng của tầng ẩn kết nối đến tất cả các nơ ron của tầng đầu ra.

Phần phân lớp: tại phần này, một lớp với các liên kết đầy đủ sẽ đóng vai trò như một bộ phân lớp các đặc trưng đã rút trích được trước đó. Tầng này sẽ đưa ra xác suất của một đối tượng trong hình.



Hình 2.12 Mô phỏng cấu trúc mạng nơ ron tích chập [10]

Cấu trúc mạng CNN là một tập hợp các lớp tích chập (Convolution) chồng lên nhau và sử dụng các hàm kích hoạt như ReLU hoặc tanh để kích hoạt các trọng số trong các nơ ron. Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo. Mô hình CNN thì các tầng liên kết được với

nhau thông qua cơ chế gọi là tầng tích chập. Lớp tiếp theo là kết quả tích chập từ tầng trước đó, nhờ vậy mà ta có được các kết nối cục bộ. Như vậy mỗi neuron ở lớp kế tiếp sinh ra từ kết quả tính toán của Kernel hoặc Filter áp đặt lên một vùng ảnh đầu vào của nơ ron trước đó.

Trong mô hình CNN thì ngược lại. Các layer liên kết được với nhau thông qua cơ chế convolution. Layer tiếp theo là kết quả convolution từ layer trước đó, nhờ vậy mà ta có đươọc các kết nối cục bộ. Nghĩa là mỗi nơ-ron ở layer tiếp theo sinh ra từ filter áp đặt lên một vùng ảnh cục bộ của nơ-ron layer trước đó.

Mỗi lớp như vậy được áp đặt các bộ lọc khác nhau, thông thường có vài trăm đến vài nghìn filter như vậy. Một số lớp khác như hợp nhất dùng để chắt lọc lại các thông tin hữu ích hơn (loại bỏ các thông tin nhiễu). Tuy nhiên, em không đi sâu vào khái niệm của các lớp này.

Trong suốt quá trình huấn luyện, CNN sẽ tự động học được các thông số cho các filter. Ví dụ trong tác vụ phân lớp ảnh, CNN sẽ cố gắng tìm ra thông số tối ưu cho các filter tương ứng theo thứ tự điểm ảnh > biên (edges) > hình dạng (shapes) > mặt (facial) > các đặc trưng cao hơn (high-level features). Lớp cuối cùng được dùng để phân lớp ảnh.

2.2.1. Convolutional

Đây thường là tầng đầu tiên của mạng nơ ron tích chập, giống như các lớp ẩn khác, lớp tích chập lấy dữ liệu đầu vào, thực hiện các phép chuyển đổi để tạo ra dữ liệu đầu vào cho lớp kế tiếp (đầu ra của lớp này là đầu vào của lớp sau). Phép biến đổi được sử dụng trong lớp tích chập này là phép tính tích chập. Mỗi lớp tích chập chứa một hoặc nhiều bộ lọc - bộ phát hiện đặc trưng (Kernel/Filter) cho phép phát hiện và trích xuất những đặc trưng khác nhau của ảnh.

Với mô hình mạng CNN, lớp tích chập cũng chính là lớp ẩn (Hidden layer), khác ở chỗ lớp tích chập là một tập các bản đồ đặc trưng, và mỗi bản đồ đặc trưng là một bản scan của dữ liệu đầu vào ban đầu, như được trích xuất ra các đặc tính (Feature) cụ thể. Trong tầng này ta sẽ có một ma trận gọi là convolution filter hay kernel thực hiện quét hoặc dịch qua ma trận đầu vào, từ trái qua phải, từ trên xuống

dưới và nhân tương ứng với từng giá trị của ma trận đầu vào, rồi cộng lại đưa qua hàm kích hoạt (Sigmoid, ReLU, Elu…), kết quả nhận được là một con số cụ thể và tập hợp lại thành một ma trận đầu ra của tầng này, ma trận này chính là bản đồ đặc trưng [5].

Giả sử ma trận đầu vào là ***I***, ma trận của bộ lọc là ***K*** có kích thước là ***h* x *w,*** ta có ma trận ***I* x *K*** sẽ được tính bởi công thức dưới :

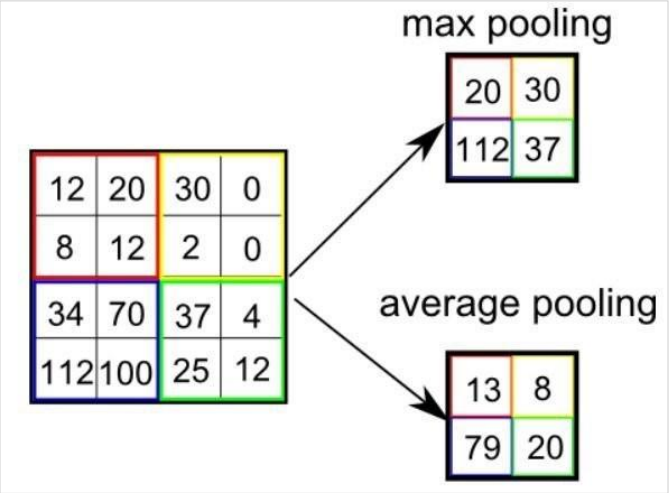
$$\left(I * K\right)_{xy} = \sum_{i=1}^{h} \sum_{j=1}^{w} K_{ij} \times I_{x+i-1, y+j-1}$$

2.2.2. Poolling

Là một lớp được thêm vào giữa các lớp tích chập với mục đích là giảm kích thước của dữ liệu thông qua việc lấy mẫu (sampling) [5], để đơn giản hóa thông tin đầu ra để giảm bớt số lượng neuron. Việc lấy mẫu này thực hiện bằng cách lấy giá trị lớn nhất hoặc giá trị trung bình của tất cả các giá trị trong cửa sổ pooling được chọn. Pooling là một cách là giảm kích thước ma trận mà vẫn giữ được những thông tin quan trọng nhất của ma trận. Nhiệm vụ của nó là duyệt một ô cửa sổ nhỏ dọc trên một ma trận hình ảnh và lấy giá trị đặc trưng của cửa sổ từ mỗi bước. Trên thực tế cửa sổ được dùng thường có kích thước 2x2 hoặc 3x3. Pooling được xem là một trong những kĩ thuật giúp giảm hiện tượng overfitting trong CNN Chúng ta có thê hình dung hoạt động của nó trong hình sau (Hình 2.13).

Lớp Pooling được sử dụng trong CNN để giảm kích thước đầu vào, tăng tốc độ tính toán và hiệu năng trong việc phát hiện các đặc trưng. Có nhiều hướng Pooling được sử dung, trong đó phổ biến nhất là pooling theo giá trị cực đại (max pooling) và pooling theo giá trị trung bình (average pooling).

- Max Pooling trả về giá trị tối đa từ cửa sổ trượt được bao phủ bởi bộ lọc (Kernel/feature).
- Average Pooling trung bình trả về mức trung bình của tất cả các giá trị từ cửa sổ trượt được bao phủ bởi bộ lọc.



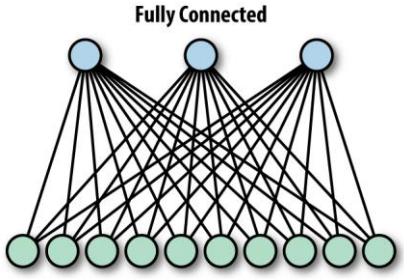
Hình 2.13 Việc thực hiện lấy mẫu trong tầng Pooling [10]

Lớp pooling thường được sử dụng ngay sau lớp tích chập (convulational), với tính chất của lớp pooling, nó làm giảm đáng kể tính chất của ma trận, giúp giảm chi phí tính toán đi đáng kể.

Lớp tích chập và Lớp Pooling, cùng nhau tạo thành lớp thứ i của mạng. Tùy thuộc vào độ phức tạp trong ảnh, số lượng các lớp như vậy có thể được tăng lên để lấy ra được chi tiết ở mức độ sâu hơn, nhưng nó yêu cầu về hiệu năng tính toán của máy tính nhiều hơn. Sau khi trải qua các bước tích chập thì mô hình có thể hiểu và phân lớp được dữ liệu. Bước tiếp theo là đưa dữu liệu đầu ra của lớp tích chập vào một mảng nơ ron bình thường để thực hiejn quá trình phân lớp.

2.2.3. Lớp kết nối đầy đủ (Fully connected layer)

Tên tiếng viết là Mạng liên kết đầy đủ. Tại lớp mạng này, mỗi một nơ-ron của layer này sẽ liên kết tới mọi nơ-ron của lớp khác. Để đưa ảnh từ các layer trước vào mạng này, buộc phải dàn phẳng bức ảnh ra thành 1 vector thay vì là mảng nhiều chiều như trước. Tại layer cuối cùng sẽ sử dụng 1 hàm kinh điển trong học máy mà bất kì ai cũng từng sử dụng đó là softmax để phân loại đối tượng dựa vào vector đặc trưng đã được tính toán của các lớp trước đó.



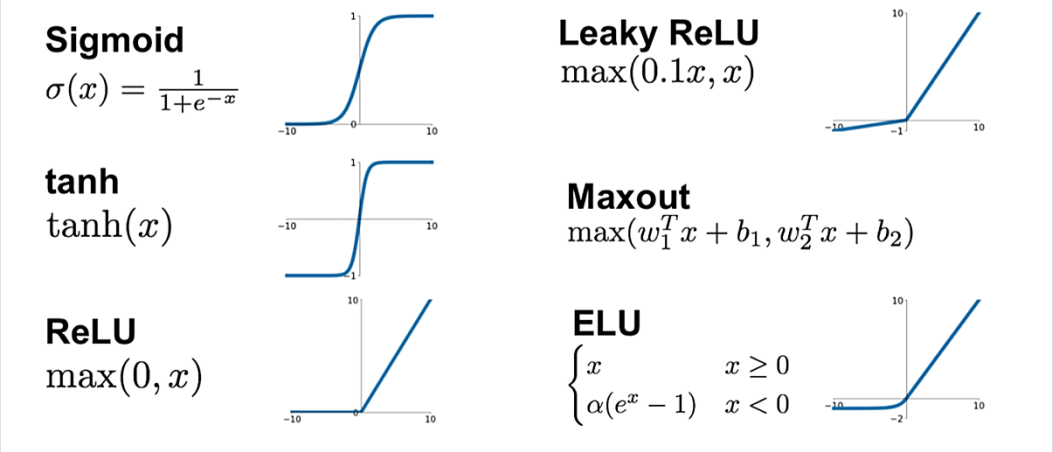
Hình 2.14 Minh họa lớp kết nối đầy đủ

Một lớp được kết nối đầy đủ với đầu ra của lớp trước đó, mỗi nơ ron tại lớp này được kết nối đến tất cả các nơ ron tại lớp tiếp theo được gọi là lớp kết nối đầy đủ. Lớp này sẽ nhận giá trị đầu vào từ lớp pooling và xác định kết quả đầu ra là gì. Đầu ra của lớp này sẽ thực hiện cuộc bầu chọn xem những đặc trưng của mình giống với kết quả hoặc nhãn đầu ra nào nhất, từ đó sẽ xác định được nhãn của dữ liệu đầu vào này là gì.

Thường thì sau các lớp Convolution và lớp Pooling thì sẽ là 2 lớp Fully connected, 1 lớp để tập hợp các feature layer mà ta đã tìm ra, chuyển đổi dữ liệu từ 3D, hoặc 2D thành 1D, tức chỉ còn là 1 vector. Còn 1 lớp nữa là kết quả đầu ra, số nơ ron của lớp này phụ thuộc vào số kết quả đầu ra hoặc nhãn phần loại mà ta muốn tìm ra. Đối với bài toán phân lớp đa nhãn (Multiclass classification) sẽ sử dụng hàm Softmax để thực hiện tính toán để xem xác suất nhãn nào lớp nhấn và dựa vào đó để thực hiện dự đoán.

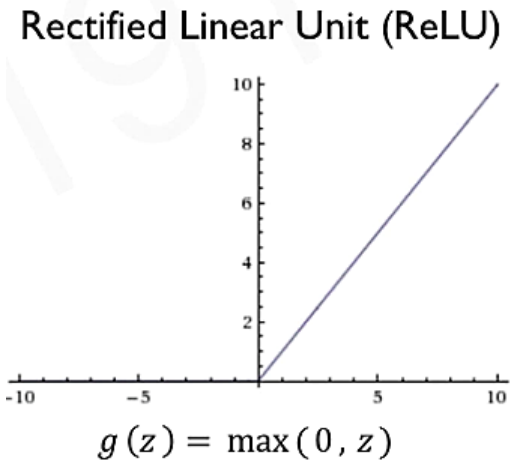
2.2.4. Hàm Kích hoạt (Activation Function)

Hàm kích hoạt là một nút được đặt ở cuối hoặc ở giữa của cấu trúc mạng nơ ron, có rất nhiều loại hàm kích hoạt khác nhau như hàm Sigmoid, Maxout, ReLU… (Hình 2.1). Việc lựa chọn hàm kích hoạt đôi khi là kinh nghiệm của người xây dựng mạng và nó còn phụ thuộc khá nhiều ở bài toán mà chúng ta đang giải quyết. Tuy nhiên hàm ReLU hoạt động khá tốt cho phần lớn các bài toán trong DL.



Hình 2.15 Các hàm kích hoạt phổ biết trong mô hình mạng nơ ron. [12]

- **ReLU** (Rectified Linear Unit) được dựa trên tư tưởng của việc loại bỏ bớt những tham số không quan trọng trong quá trình training và điều đó là cho mạng của chúng ta trở nên nhẹ hơn và việc training cũng nhanh chóng và có hiệu quả hơn. Hàm này thực hiện một việc rất đơn giản như sau: giữ nguyên những giá trị đầu vào lớn hơn 0, nếu giá trị đầu vào nhỏ hơn 0 thì coi là 0. Chúng ta có thể hình dung kĩ hơn trong hình sau (Hình 2.2):



Hình 2.16 Hàm kích hoạt ReLU [12]

Bởi vì hàm ReLU trả về dữ liệu khác 0 trong mọi trường hợp nên điều đó làm mạng chúng ta không phải training những dữ liệu không cần thiết và hơn nữa công thức của ReLU rất đơn giản khiến cho việc tính toán cũng

trở nên dễ dàng hơn. Điều này làm cho hàm kích hoạt ReLU đang được sử dụng phổ biến trong nhiều ứng dụng của mô hình mạng nơ ron.

- **Softmax** là một loại của hàm kích hoat - activation function. Nó rất hữu ích trong bài toán phân loại đa lớp. Softmax nhận đầu vào là một mảng số thực và đầu ra là một phân phối xác suất với mỗi phần tử nằm trong khoảng [0, 1] và tổng các phần tử là 1 (tương ứng với 100%). Để làm được điều này hàm softmax sẽ chuyển đổi giá trị đầu ra của mạng nơ ron bằng cách chia cho tổng giá trị. Lúc này đầu ra có thể coi là một vector của xác suất dự đoán của các class. Chúng ta có thể thấy rõ hơn trong công thức sau

$$softmax_i(a) = \frac{\exp a_i}{\sum \exp a_i}$$

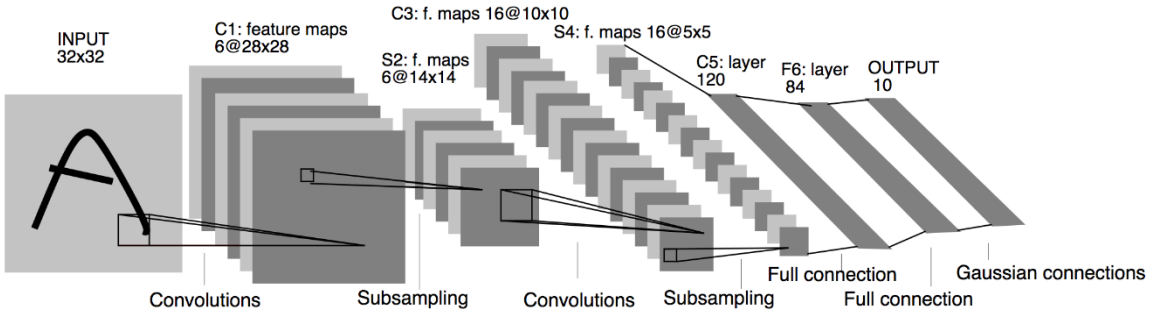
Chúng ta có thể sử dụng khoảng các Euclid để so sánh khoảng cách giữa one-hot encoding và softmax nhằm phục vụ cho việc xây dựng hàm loss và tối ưu các tham số của mạng nơ ron.

2.2.5. Một số mô hình mạng thông dụng trên thực tế

Trên thực tế mô hình mạng nơ ron được sử dụng phổ biến với các kiến trúc mạng sử dụng lớp tích chập nhiều tầng với kích thước của feature map của từng lớp tăng dần, nhưng có nhiều mô hình với kiến trúc mạng mới đây đã thiết kế sáng tạo hơn và cho kết quả hiệu quả hơn. Dưới đây là các ví dụ của một số kiến trúc mạng nơ ron tích chập thông dụng:

LeNet

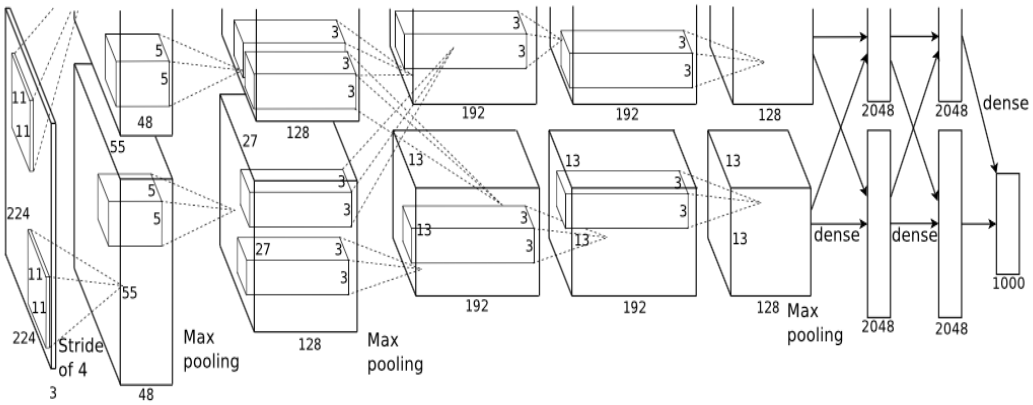
LeNet là một trong những mạng CNN lâu đời nổi tiếng nhất được Yann LeCUn phát triển vào những năm 1998s. Cấu trúc của LeNet gồm 2 layer (Convolution + maxpooling) và 2 layer fully connected layer và output là softmax layer. Chúng ta cùng tìm hiểu chi tiết architect của LeNet đối với dữ liệu mnist (accuracy lên đến 99%) [8].



Hình 2.17 Kiến trúc mạng LeNet [8]

Alexnet

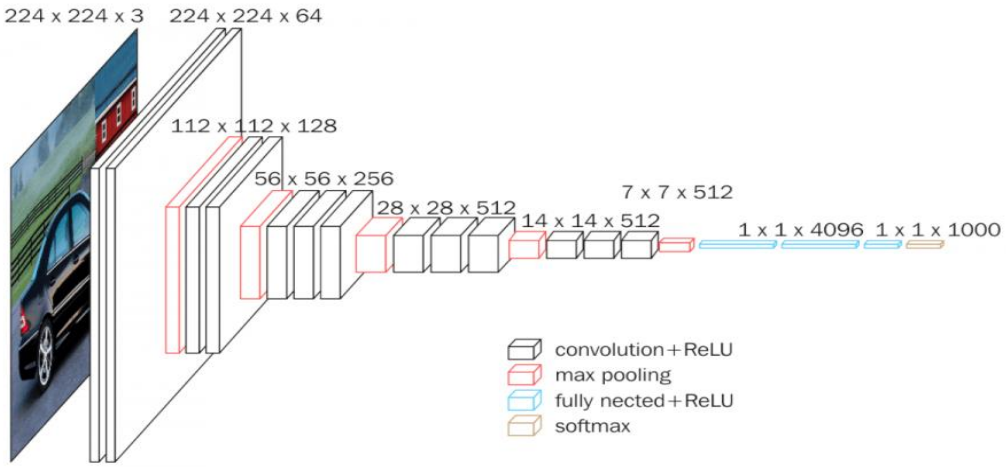
AlexNet là một mạng CNN đã dành chiến thắng trong cuộc thi ImageNet LSVRC-2012 năm 2012 với large margin (15.3% VS 26.2% error rates). AlexNet là một mạng CNN traning với một số lượng parameter rất lớn (60 million) so với LeNet. Một số đặc điểm như: Sử dụng relu thay cho sigmoid(or tanh) để xử lý với 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, Sử dụng kỹ thuật data augmentation để tạo them data training bằng cách translations, horizontal reflections [8].



Hình 2.18 Kiến trúc mạng Alexnet [8]

VGGNet

Sau AlexNet thì VGG ra đời với một số cải thiện hơn, trước tiên là model VGG sẽ có độ sâu hơn, tiếp theo là thay đổi trong thứ tự lớp tích chập. Từ LeNet đến AlexNet đều sử dụng tích chập rồi maxpooling còn VGG thì sử dụng 1 chuỗi tích chập liên tiếp (Conv-Conv-Conv) ở giữa và cuối của kiến trúc mạng VGG. Việc này sẽ làm cho việc tính toán trở nên lâu hơn nhưng những đặc trưng sẽ vẫn được giữ lại nhiều hơn so với việc sử dụng maxpooling sau mỗi lớp tích chập. 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 [8].



Hình 2.19 Kiến trúc mạng VGGNet [8]

2.3. Ứng dụng mạng nơ ron tích chập trong các bài toán thực tế về xử lý và phân loại ảnh

CNN được ứng dụng nhiều trong thị giác máy tính, hệ thống hỗ trợ ra quyết định, các hệ thống phân tích ngôn ngữ tự nhiên và nhiều hệ thống khác như:

Nhận dạng văn bản: Bởi vì CNN giỏi trong việc giải mã hình ảnh trực quan, chúng đã được chứng minh là hữu ích trong phân tích văn bản. Sức mạnh của CNN giúp xử lý ngôn ngữ tự nhiên trên các tài liệu tương tự hoặc viết tay, trong đó hình

ảnh là biểu tượng cần phiên mã. Chức năng này của CNN được gọi là nhận dạng ký tự quang học hoặc viết tắt là OCR (Optical Character Recognition).

Nhận diện khuôn mặt: Vì nhận dạng hình ảnh và phân loại hình ảnh là một cách sử dụng phổ biến của CNN, nhận dạng khuôn mặt cũng là một ứng dụng nổi tiếng. Gần đây như năm 2015, CNN đã thể hiện khả năng nhận diện khuôn mặt từ nhiều góc độ khác nhau, ngay cả với tầm nhìn hạn chế. Các thuật toán thay thế tốt nhất đấu tranh với việc nhận ra các số liệu trong một hình ảnh có thể nhỏ và mỏng, nhưng CNN có thể hoàn thành được công việc.

Nghiên cứu chế tạo thuốc: CNN đã được áp dụng đối với khám phá thuốc. Các mạng lưới thần kinh xử lý dữ liệu để hiểu sự tương tác giữa các phân tử và protein sinh học. Giống như cách CNN xử lý hình ảnh phân tích các đặc điểm của hình ảnh, CNN đã được huấn luyện để hiểu cấu trúc hóa học trong việc khám phá thuốc. Thông tin mà các CNN này cung cấp dẫn đến các phương pháp điều trị y tế tiềm năng tốt hơn. Đáng chú ý, CNN AtomNet đã được sử dụng trong việc phát triển các phương pháp điều trị cho virus Ebola.

2.4. Xây dựng tập dữ liệu cho bài toán

2.4.1. Giới thệu về bộ dữ liệu sử dụng trong bài toán

Trong bài toán phân loại độ tuổi của người bằng hình ảnh sử dụng mạng nơ ron tích chập chúng ta sử dụng tập dữ liệu khuôn mặt các diễn viên trong phim của Ấn Độ (Idian Movies Face Database) hay viết tắt là IMFDB, được lấy từ nguồn [15]. IMFDB là một bộ dữ liệu khuôn mặt lớn bao gồm 26742 hình ảnh của 100 diễn viên Ấn Độ được thu thập từ hơn 100 video. Tất cả các hình ảnh được lựa chọn và cắt xén thủ công từ các khung hình video dẫn đến rất nhiều kích thước, tư thế, biểu hiện, độ sáng, độ tuổi và độ phân giải. Bộ dữ liệu IMFDB là cơ sở dữ liệu khuôn mặt đầu tiên cung cấp nhãn chi tiết cho mọi hình ảnh về độ tuổi, tư thế, giới tính và biểu hiện có thể giúp nhiều ứng dụng khác nhau liên quan đến phân tích khuôn mặt.

Ảnh khuôn mặt người trong bộ dữ liệu IMFDB được thu thập từ các video, vì thế khuôn mặt trong các video phim được cho là có rất đa dạng ảnh khác nhau về độ

chiếu sáng, góc nhìn, độ phân giải, độ mờ, v.v. Video được thu thập từ hai thập kỷ trước nên nó chứa rất nhiều ảnh khuôn mặt khác nhau về độ tuổi so với hình ảnh được thu thập từ Internet thông qua truy vấn tìm kiếm trong hiện nay[6]. Bộ dữ liệu IMFDB được xây dựng bằng cách chọn thủ công và cắt xén các khung hình video dẫn đến mức độ đa dạng của các biểu cảm của khuôn mặt người và có thể được sử dụng để phát triển các thuật toán để phân tích biểu cảm của mặt người [15].

Cắt xén khuôn mặt: Khuôn mặt được cắt bằng một khung hình vừa với một khuôn mặt. Để duy trì tính nhất quán trên các hình ảnh, tôi đã theo dõi một cách cắt xén khuôn mặt từ trán đến cằm.

Hình 2.20 Một số hình ảnh ví dụ của bộ dữ liệu IMFDB [15]

Trong bộ dữ liệu kèm theo tệp bảng dữ liệu tương ứng với thông tin mã hình ảnh, các thông tin khác và nhãn phân loại của từng hình ảnh [6]. Bao gồm các trường dữ liệu như sau :

ID : Image id

Expressions : Anger, Happiness, Sadness, Surprise, Fear, Disgust

Illumination : Bad, Medium, High

Pose : Frontal, Left, Right, Up, Down

Occlusion : Glasses, Beard, Ornaments, Hair, Hand, None, Others

Makeup : Partial makeup, Over-makeup

Gender : Male, Female

Age : Young, Middle, Old

Bảng 2.1 Mẫu bộ dữ liệu IMFDB

ID	Expression	Illumination	Pose	Occlusion	Makeup	Gender	Age
088.jpg	SURPRISE	MEDIUM	FRONTAL	GLASSES	OVER	Male	Young
177.jpg	NEUTRAL	BAD	RIGHT	NONE	PARTIAL	Male	Old
634.jpg	HAPPINESS	MEDIUM	FRONTAL	OTHERS	PARTIAL	Male	Young
671.jpg	SADNESS	BAD	LEFT	OTHERS	PARTIAL	Female	Young
799.jpg	HAPPINESS	MEDIUM	DOWN	OTHERS	PARTIAL	Female	Middle
807.jpg	HAPPINESS	MEDIUM	UP	OTHERS	PARTIAL	Male	Middle
908.jpg	DISGUST	BAD	DOWN	OTHERS	PARTIAL	Male	Middle
938.jpg	SADNESS	BAD	LEFT	NONE	PARTIAL	Male	Young
033.jpg	HAPPINESS	MEDIUM	FRONTAL	NONE	PARTIAL	Male	Young
183.jpg	SURPRISE	MEDIUM	FRONTAL	NONE	OVER	Female	Young

2.4.2. Tiền xử lý và chuẩn bị dữ liệu

a. Loại bỏ đặc trưng không cần thiết

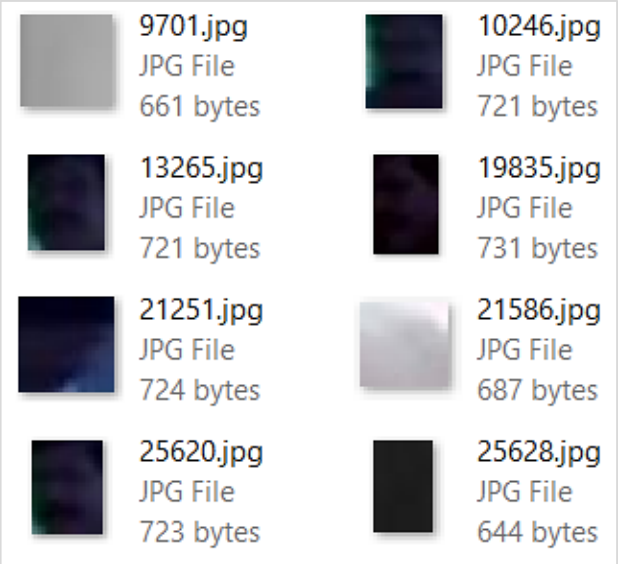
Để nhận được kết quả tốt trong việc thực hiện huấn luyện mô hình chúng ta phải phân biệt và loại bỏ những trường trong bộ dữ liệu mà không cần thiết trong bài toán phân loại độ tuổi. Dựa vào yêu cầu thực tế của bài toán phân loại độ tuổi thì chúng ta loại bỏ các trường không cần thiết như biểu cảm (Expressions), tư thế (Pose), trang điểm (Makeup) và giới tính (Gender) còn lại hai thuộc tính là mã hình ảnh (ID) và độ tuổi (Age), với nhãn độ tuổi sẽ bao gồm có ba nhãn là "Young" tương đương với độ tuổi từ 13 đến 30 tuổi, "Middle" tương đương với 31 đến 50 tuổi và "Old" là tương đương với trên 50 tuổi [6].

	A	B
1	ID	Class
2	377.jpg	MIDDLE
3	17814.jpg	YOUNG
4	21283.jpg	MIDDLE
5	16496.jpg	YOUNG
6	4487.jpg	MIDDLE
7	6283.jpg	MIDDLE
8	23495.jpg	YOUNG
9	7100.jpg	YOUNG
10	6028.jpg	YOUNG
11	22617.jpg	OLD
12	11177.jpg	YOUNG
13	2462.jpg	MIDDLE

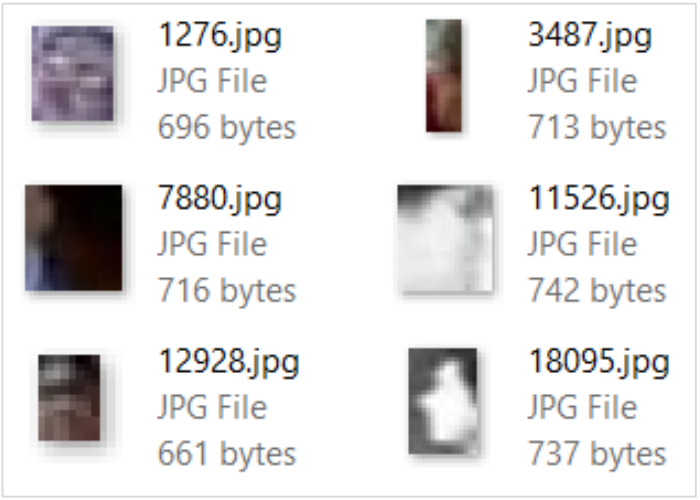
Hình 2.21 Dữ liệu sau khi loại bỏ các thuộc tính không cần thiết

b. Loại bỏ nhiễu trong dữ liệu

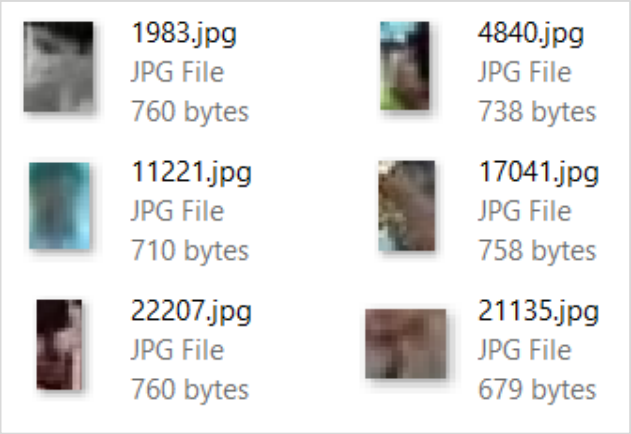
Do bộ dữ liệu được thực hiện thu thập bằng cách tự động cắt xén khuôn mặt người tự động từ video nên có thể tồn tại các nhiễu về dữ liệu như: thiếu độ sáng hoặc hình ảnh sáng chói (Hình 2.22), bị chênh lệch khuôn mặt, khuôn mặt bị che (Hình 2.24) hoặc quá mờ và kích thước quá nhỏ (Hình 2.23). để khôi phục được vấn đề này chúng ta sẽ phải duyệt qua các tập dữ liệu và loại bỏ đi các dữ liệu nhiễu trong tập dữ liệu. Dưới đây là các hình ảnh của các dữ liệu bị nhiễu:



Hình 2.22 Ví dụ hình ảnh bị thiếu sáng và sáng chói



Hình 2.23 Ví dụ hình ảnh có kích thước quá bé và quá mờ



Hình 2.24 Một số hình ảnh bị lệch khuôn mặt và bị che khuôn mặt

c. Chỉnh kích thước ảnh sang kích thước phù hợp

Các dữ liệu hình ảnh trong bộ dữ liệu có rất nhiều kích thước khác nhau giao động từ 543 x 616 pixel (diểm ảnh) đến 13 x 22 pixel, nên đầu tiên chúng ta phải điều chỉnh kích thước của bức ảnh, để cho hình ảnh có kích thước bằng nhau trước khi sử dụng làm dữ liệu đầu vào của mô hình**.** Trong bài toán này chúng ta sử dụng ảnh với kích thước là 128 x 128 pixel. Đây là một kích thước phù hợp để dữ lại các chi tiết của khuôn mặt người và đồng thời kích thước không quá to mà ảnh hưởng đến thời gian huấn luyện mô hình. Dưới đây là hình ảnh sau khi chỉnh sửa (Hình 2.25):

Hình 2.25 Hình ảnh sau khi chỉnh sửa kích thước

d. Phân chia dữ liệu

Chúng ta sẽ chia bộ dữ liệu ra thành ba tập bao gồm tập dữ liệu huấn luyện, tập kiểm tra và tập dữ liệu Kiểm chứng

Bộ dữ liệu bao gồm tất cả 26742 bản ghi và được chia ra thành hai phần là tập huấn luyện 19906 bản ghi tương đương với 75% và tập kiểm chứng 6636 bản ghi tương đương với 25%.

Bảng 2.2 Phân chia dữ liệu thành hai tập

Tập dữ liệu	Tỷ lệ	Số bản ghi
Tập huấn luyện	**75%**	**19906**
Tập kiểm chứng	**25%**	**6636**
Tổng	**100%**	**26742**

Từ tập dữ liệu huấn luyện đã được phân chia chúng ta tiếp tục chia ra thành tập huấn luyện 85% và tập kiểm tra (validation) 15% như sau

Bảng 2.3 Phân chia dữ liệu thành hai tập

Tập dữ liệu	Tỷ lệ	Số bản ghi
Tập huấn luyện	**85%**	**16920**
Tập kiểm chứng	**15%**	**2986**
Tổng	**100%**	**19906**

2.5. Xây dựng mô hình mạng nơ ron tích chập để giải quyết bài toán phân loại độ tuổi của người bằng hình ảnh.

2.5.1. Cấu trúc mô hình

Việc xây dựng mô hình để đạt được hiệu quả cao phụ thuộc vào các yếu tố như cấu trúc cảu mô hình mạng, lựa chọn thuật toán, xác định các biến dữ liệu phù hợp và điều chỉnh các tham số để cho phù hợp dựa trên bộ dữ liệu sử dụng để huấn luyện mô hình. Đối với bài toán này chúng ta sẽ sử dụng mô hình mạng nơ ron tích chập CNN để phân loại độ tuổi bằng hình ảnh với tập dữ liệu dã giới thiệu ở mục trên.

Cấu trúc mô hình mạng nơ ron tích chập được sử dụng trong bài toán dựa vào một mô hình mạng LeNet, có cấu trúc bao gồm ba lớp tích chập (Convolution) với mỗi lớp sẽ có các lớp Pooling ở giữa của từng lớp, tiếp theo đến ba lớp kết nối đầy đủ (Fullyconnected) với lớp kết nối đầy đủ cuối cùng là lớp giá trị đầu ra với sô nơ ron bằng sô nhãn phân loại. Mô hình bao gồm các chi tiết cụ thể như sau:

❖ **Lớp tích chập (Convolution)**

Ba lớp đầu tiên của mô hình mạng chúng ta đều là lớp tích chập, với lớp tích chập đầu tiên nhận dữ liệu đàu vào là mảng cẩu các hình ảnh có kích thước 128 x 128 pixel. Tại lớp này chúng ta khai báo với số bộ lọc (Kernel) sử dụng là 25 với kích thước của từng bộ lọc mà 3 x 3. Tiếp theo là khai báo hàm kích hoạt cho lớp này, chúng ta sử dụng hàm "ReLu" đã giới thiệu ở mục trên.

Tương tự với lớp tích chập đầu tiên, lớp tích chập thứ hai và thứ ba chúng ta sẽ khai báo tương ứng nhưng với số dặc trưng sử dụng khác nhau là 50 và 75 theo lần lượt. Còn hàm kích hoạt thì chúng ta vẫn khai báo hàm "ReLu" tương tự như trên.

Giữa các lớp kích hoạt chúng ta sẽ khai báo một lớp hợp nhất (Pooling), ở đây chúng ta sử dụng phép hợp nhất tối đa (Max pooling) với giá trị kích thước là 2 x 2.

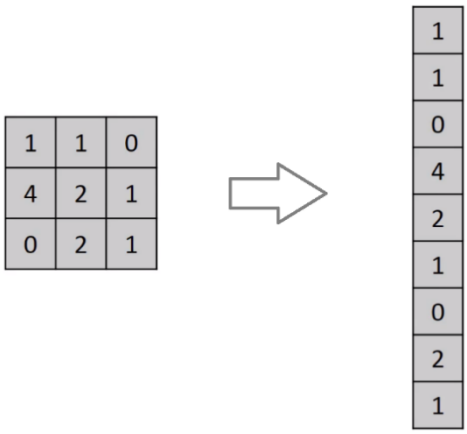
❖ **Hàm kích hoạt sử dụng (Activation function)**

Hàm kích hoạt sử dụng trong bài toán này gồm có hai hàm là "ReLu" (Rectified Linear Unit) và hàm "Softmax".

Tại các lớp tích chập chúng ta sử dụng hàm "ReLu" là hàm kích hoạt. Hàm này có công thức dễ thực hiện tính toán và hiệu quả với nhiều loại bài toán, với tốc độ thữ hiện nhanh đãn đến thời gian huấn luyện mô hình tương đối nhanh so vưới hàm kích hoạt khác. Tại tầng liên kết đầy đủ cuối cùng, chung ta sử dụng hàm "Softmax". Hàm "Softmax" thường được sử dụng ở tầng đầu ra, nhằm đánh giá sắc xuất nhãn phân loại của dữ liệu đầu vào của tầng đấy.

❖ **Lớp làm phẳng (Flatten)**

Lớp này có nhiệm vụ chuyển đổi kết quả đầu ra từ lớp tích chập là mảng nhiều chiều và chuyển đổi thành vec tơ một chiều trước khi được vào tầng kết nối đầy đủ để thực hiện quá trình phân loại [14].



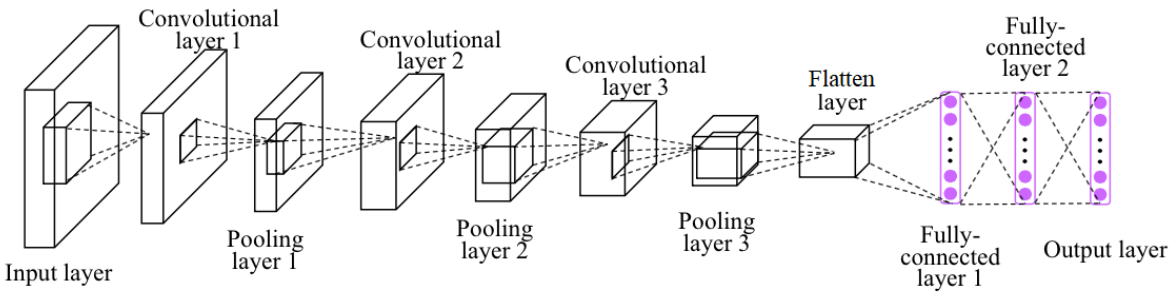
Hình 2.26 Minh họa phương thức làm phẳng (Flatten) [14]

❖ **Lớp kết nối đầy đủ (Fully connected layer)**

Tại lớp này chúng ta sử dụng tất cả 3 lớp, hai lớp đầu tiên là một lớp kết nối đầy đủ với số nơ ron bằng 32 và sử dụng hàm "ReLU" là hàm kích hoạt.

Với lớp phân nhãn cuối cùng với số nơ ron bằng 3 tương ứng với số nhãn phân loại của tập dữ liệu là ("Middle", "Old", "Young").

Cuối cùng chúng ta nhận được mô hình mạng được minh họa ở hình dưới (Hình 2.27):



Hình 2.27 Minh họa mô hình mạng sử dụng trong bài toán [14]

2.5.2. Các hàm và kỹ thuật sử dụng

❖ **Thuật toán tối ưu (Optimizer)**

Thuật toán tối ưu chính là trung tâm cơ sở để xây dựng các mô hình neural network với mục tiêu là "học" được các đặc điểm (features hay patterns) từ dữ liệu đầu vào, đây là cách mô hình được cập nhật dựa trên dữ liệu huấn luyện được cung cấp và hàm thiệt hại. Từ đó có thể tìm một tập các trọng số **W** (weights) và độ lệch **b** (bias) để tối ưu hóa độ chính xác của models hoặc tối ưu giá trị của hàm mất mát.

Trong bài toán này chúng ta sử dụng thuật toán tối ưu Adam (Adaptive moment Estimation). Adam là một thuật toán tối ưu có thể biến đổi thích nghi tốc độ học (learning rate), và được thiết kế đặc trưng cho các mô hình học sâu. Thuật toán Adam có thể được xem như là sự kết hợp của RMSprop và SDG (Sochastic Gradient Descent). Nó sử dụng giá trị bình phương của gradient để chia tỷ lệ học tập như thuật toán RMSprop và nó tận dụng động lượng (momentum) bằng cách sử dụng trung bình di chuyển của giá trị gradient thay vì gradient như SGD. Hãy cùng xem xét kỹ hơn về cách thức hoạt động của nó [3].

❖ **Hàm lỗi (Loss function)**

Hàm lỗi dùng để đo lường mức độ chính xác của mô hình trong quá trình huấn luyện. Chúng ta cần giảm thiểu giá trị của hạm này "điều khiển" mô hình đi đúng hướng (chỉ số lỗi càng ít, chính xác càng cao). Hàm lỗi sử dụng trong bài toán này là **Cross Entropy** tính toán độ chênh lệch giữa 2 phân phối xác suất của dự đoán và của

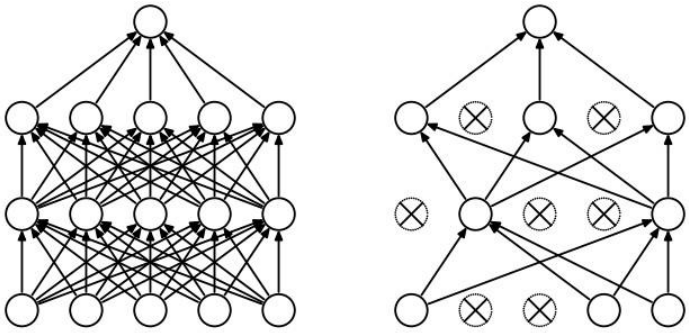
nhãn. Nhãn được biến đổi thành một vector one-hot với tất cả phần tử trong vector nhận giá trị 0 trừ một vị trí biểu diễn nhãn nhận giá trị 1. Đầu ra của mô hình là một phân phối xác suất (p1, p2, ..., pn) với n là số lượng các lớp. Phân phối xác suất của nhãn là 1 vector one-hot (0, 0, ... , 1, 0, ..., 0) với vị trí thứ **i** nhận giá trị 1 tương ứng lớp thứ **i**.

$$CE = -\sum_{i}^{C} t_i log(s_i)$$

Tuy nhiên hàm cross entropy được bàn luận sau đây là một trong những hàm loss được sử dụng rất phổ biến và chứng minh được hiệu quả rõ rệt trong những bài toán phân loại đa lớp.

❖ **Kỹ thuật Drop-out để giảm Over-fitting**

Khi chúng ta sử dụng full connected layer, các nơ ron sẽ phụ thuộc "mạnh" lẫn nhau trong suốt quá trình huấn luyện, điều này làm giảm sức mạng cho mỗi neural và dẫn đến trường hợp over-fitting trên tập dữ liệu huấn luyện, và để cải thiện được vấn đề này chúng ta sử dụng ký thuật dropout. Kỹ thuật drop-out là cách thức mà chúng ta giả định một phần các unit (nơ ron) bị ẩn đi trong quá trình huấn luyện, qua đó làm giảm tích hòa trộn mà dẫn đến hiện tượng over-fitting. Tại mỗi bước trong quá trình huấn luyện, khi thực hiện quá trình lan truyền xuôi (Forward propagation) đến tầng sử dụng drop-out, thay vì tính toán tất cả unit có trên tầng đó, tại mỗi unit ta thực hiện lựa chọn ngẫu nhiên một số unit được thực hiện tính toán dựa trên xác suất p [13].



Hình 2.28 Minh họa trước và sau khi sử dụng drop-out [13]

❖ **Kỹ thuật tăng cường dữ liệu (Data augmentation)**

Hiện nay trong deep learning thì vấn đề dữ liệu có vai trò rất quan trọng. Chính vì vậy có những lĩnh vực có ít dữ liệu để cho việc train model thì rất khó để tạo ra được kết quả tốt trong việc dự đoán. Do đó chúng ta cần đến một kỹ thuật gọi là tăng cường dữ liệu (data augmentation) để phục vụ cho trường hợp số lượng dữ liệu hạn chế, thì bạn vẫn có thể tạo ra được nhiều dữ liệu hơn dựa trên những dữ liệu bạn đã có. Có một số trường hợp sử dụng kỹ thuật này để cải thiện về vấn đề dữ liệu mất cân bằng [16].

Cách thức thực hiện của kỹ thuật này là nó thực hiện các phép biến đổi trên dữ liệu đầu vào và sinh ra dữ liệu mới là kết quả sau khi biến đổi. Có thể xem ví dụ tại hình dưới, đó là các hình được tạo ra thêm từ một ảnh gốc ban đầu.

Hình 2.29 Kỹ thuật tăng cường dữ liệu [16]

Trong kỹ thuật tăng cường dữ liệu có rất nhiều phương thức để sinh ra dữ liệu khác nhau từ một dữ liệu gốc như : rotation, flip, random crop, width shift, height shift, color shift, contrast change, v.v.

Cách lựa chọn sử dụng phương thức phù hợp để đưa ra kết quả tốt thì tùy thuộc vào dữ liệu, số lượng mẫu, tính balance/imbalance của mẫu và kiến trúc mạng.

❖ **Cân bằng trọng số (weight balancing)**

Kỹ thuật này được sử dụng phổ biến với các mô hình mà sử dụng bộ dữ liệu không cân bằng (Imbalanced dataset). Dữ liệu mất cân bằng có ảnh hưởng rất lớn tới độ chính xác của mô hình. Nhưng hiện tượng mất cân bằng này lại là một hiện tượng rất hay xảy ra trong các bài toán học máy và học sâu, khi yêu cầu của bài toán là một

số lượng lớn dữ liệu thì dễ dàng xảy ra trường hợp các dữ liệu của từng nhãn có số lượng chênh lệch nhau.

Với bộ dữ liệu sử dụng trong thuật toán chúng ta phát hiện ra tỷ lệ phân chia của số lượng mẫu không cân bằng giữa các nhãn. Cụ thể như sau:

Bảng 2.4 Tỷ lệ số mẫu dữ liệu trên các nhãn

Nhãn	Tỷ lệ
Middle	0.542
Young	0.336
Old	0.120

Bằng kỹ thuật này chúng ta sẽ tang giá trị của trọng số (weight) cho phần dữ liệu thiếu, mô hình sẽ dựa vào giá trị trọng số trên mỗi mẫu huấn luyện để thực hiện việc điều chỉnh hàm lỗi. Với giá trị mặc định thì mỗi mẫu dữ liệu đều bằng nhau với giá trị là 1. Việc tăng giá trị trọng số của một mẫu hoặc một nhãn nào đấy sẽ tăng mức độ quan trọng của nhãn phân loại đấy lên.

2.5.3. Định nghĩa mô hình

Trong luận văn này chúng ta sử dụng Keras framework hỗ trợ xây dựng mô hình và huấn luyện mô hình trên ngôn ngữ python. Loại mô hình mà chúng ta sẽ sử dụng là mô hình Tuần tự (Sequential). Mô hình tuần tự có thể dễ để xây dựng một mô hình trong Keras. Cho phép chúng ta xây dựng một mô hình theo từng lớp. Bằng cách sử dụng hàm “Add()” để thêm các tầng vào mô hình mạng.

Đầu tiên chúng ta thực hiện import các thư viện cần thiết sửu dụng trong bài toán bao gồm :

OS : thư viện hỗ trợ việc thao tác với các thư mục trong máy

Pandas : Thư viện sử dụng để thao tác và phân tích dữ liệu

Tensorflow: Thư viện hỗ trợ về các phương thức học máy

Keras: Thư viện hỗ trợ xây dựng và huấn luyện mô hình mạng, gồm các hàm như "model", "layer", "conv", "maxpooling".

Sklearn : Hỗ trợ các thao tác đọc ghi, thay đổi kích thước ảnh

OpenCV : Thư viện xử lý ảnh

Mathplotlib : Thư viện biểu diễn dữ liệu, vẽ đồ thị

Numpy : Các thao tác thực hiện với dữ liệu kiểu mảng

```
import os
import pandas as pd # some array operation
import tensorflow
import keras
from keras.models import Sequential, load_model
from keras.layers import Dense, Flatten, InputLayer, Dropout, Conv2D, Activation, MaxPooling2D
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
import pickle # Load saved data
import numpy as np # Manipulating with array
import cv2 # image operation
```

Tiếp tục khai báo các biến sử dụng trong mô hình bao gồm có:

"image_size" : Kích thước dữ liệu đầu vào của mô hình

"channel_num" : Số kênh của dữ liệu

"class_num" : Số nhãn phân loại của bộ dữ liệu

```
# Define number of parameter that we going to use in our nn model
img_size = 128
chanel_num = 3
class_num = 3
```

Trong mô hình này chúng ta dung sữ liệu đầu vào là hình ảnh với kích thước 128 x 128 pixel, và là hình ảnh màu nên chúng ta khai bái biến "img_size" là 128 và "channel" là 3 tương ứng với ba giải màu RGB của ảnh màu. Với biến "class_num" ta khai báo là 3 tương ứng với ba nhãn phân loại độ tuổi.

Bắt đầu khai báo và định nghĩa mô hình mạng CNN. Đầu tiên chúng ta khai báo kiểu mô hình sử dụng là mô hình kiểu tuần tự.

```
# Define model type
model = Sequential()
```

Tiếp theo chúng ta thiết lập các lớp trong mô hình bằng cách sử dụng hàm "Add()" để thêm các tầng của mô hình tương tụ như các tham số của tầng đấy:

```
# Add 1th layer, convolution 32 neurons with relu activation func
model.add(Conv2D(25,kernel_size=3, strides=1, input_shape=(img_size,img_size,chanel_num)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
```

Với tầng đầu tiên chúng ta khai báo một lớp tích chập với số bộ lọc là "filters" 25, kích thước của bộ lọc "kernel_size" là 3 tương ứng với 3 x 3, số bước dịch chuyển của bộ lọc "Strides" là 1 và kích thước dữ liệu đầu vào "input_shape" là bằng kích thước hình ảnh mà chúng ta đã chuẩn bị tại mục trên là 128 x 128 x 3 với 3 là số giải màu của hình ảnh mà chúng ta sử dụng làm dữ liệu đầu vào, nếu là ảnh xám thì giá trị này sẽ bằng 128 x 128 x 1. Một thành phần quan trọng mà chúng ta phải khai báo trong các tầng đấy là hàm kích hoạt sử dụng tại tầng đó, ở đây chúng ta dung hàm "ReLU".

Trong mô hình này chúng ta sẽ sử dụng lớp maxpooling mỗi khi thực hiện lớp tích chập. Tham số khai báo cho lớp này là (2,2), là kích thước của cửa sổ bằng 2x2.

Tương tự như tầng đầu tiên, chúng ta khai báo lớp tiếp theo là một lớp tích chập với số bộ lọc là 50, và cá tham số còn lại là giống nhau. Tương tụ như lớp maxpooling.

```
# Add convolution layer with ReLU activation fuction
model.add(Conv2D(50,kernel_size=3, strides=1))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
```

Tại tầng thứ ba chúng ta khai báo lớp tích chập tương tự như tầng trên và tăng số bộ lọc sử dụng lên thành 75, và các tham số còn lại không thay đổi. Tương tự như lớp maxpooling với kích thước là 2 x 2.

```
# Add 3nd convolution layer
model.add(Conv2D(75,kernel_size=3, strides=1))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
```

Trước khi truyền dữ liệu vào tầng kết nối đầy đủ chúng ta phải thực hiện làm phẳng dữ liệu bằng cách sử dụng lớp "Flatten()", khai báo bằng phương thức "Add(Flatten())":

```
# Add flatten layer
model.add(Flatten())
```

Tầng tiếp theo chúng ta sử dụng là 2 tầng kết nối đầy đủ với số nơ ron là 32 và hàm kích hoạt sử dụng là "relu":

```
# Add normal dense layer with 32 neurons
model.add(Dense(32))
model.add(Activation('relu'))
```

```
model.add(Dense(32))
model.add(Activation('relu'))
```

Trước khi thêm tầng đầu ra cuối cùng chúng ta thực hiện qua lớp "Drop-out" để tránh xáy ra trường hợp Overfitting trong quá trình huấn luyện:

```
model.add(Dropout(0.5))
```

Với sự hỗ trợ của thư viện Keras thì chúng ta khai báo hàm drop-out chỉ cần gọi đến hàm "Add(dropout(0.5))", với giá trị 0.5 là tỉ lệ thành phần trong mảng mà mình muốn huấn luyện tại tầng đó.

Tại tầng cuối cùng là tầng đưa ra kết quả đầu ra chúng ta khai báo một lớp kết nối đầy đủ với số nơ ron bằng 3 tương ứng với số nhãn phân loại của bộ dữ liệu sử dụng, và hàm kích hoạt sử dụng tại tầng này là hàm "Softmax":

```
# Final dense layer are 3 neurons means 3 classes that we have (young, middle, old)
model.add(Dense(3))
# Add final layer activation fuc using softmax func
model.add(Activation('softmax'))
```

Sau khi chúng ta khai báo hết tất cả các tầng thì chúng ta có thể xem tổng quát cấu trúc mô hình mạng và số lượng các tham số tổng quát hoặc trên các tầng cụ thể bằng cách gọi hàm "summary()" như sau :

```
model.summary()
```

Kết quả hiển thị là:

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 126, 126, 25)      700
_________________________________________________________________
activation_1 (Activation)    (None, 126, 126, 25)      0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 63, 63, 25)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 61, 61, 50)        11300
_________________________________________________________________
activation_2 (Activation)    (None, 61, 61, 50)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 30, 30, 50)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 28, 28, 75)        33825
_________________________________________________________________
activation_3 (Activation)    (None, 28, 28, 75)        0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 14, 14, 75)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 14700)             0
_________________________________________________________________
dense_1 (Dense)              (None, 32)                470432
_________________________________________________________________
activation_4 (Activation)    (None, 32)                0
_________________________________________________________________
dense_2 (Dense)              (None, 32)                1056
_________________________________________________________________
activation_5 (Activation)    (None, 32)                0
_________________________________________________________________
dropout_1 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_3 (Dense)              (None, 3)                 99
_________________________________________________________________
activation_6 (Activation)    (None, 3)                 0
=================================================================
Total params: 517,412
Trainable params: 517,412
Non-trainable params: 0
```

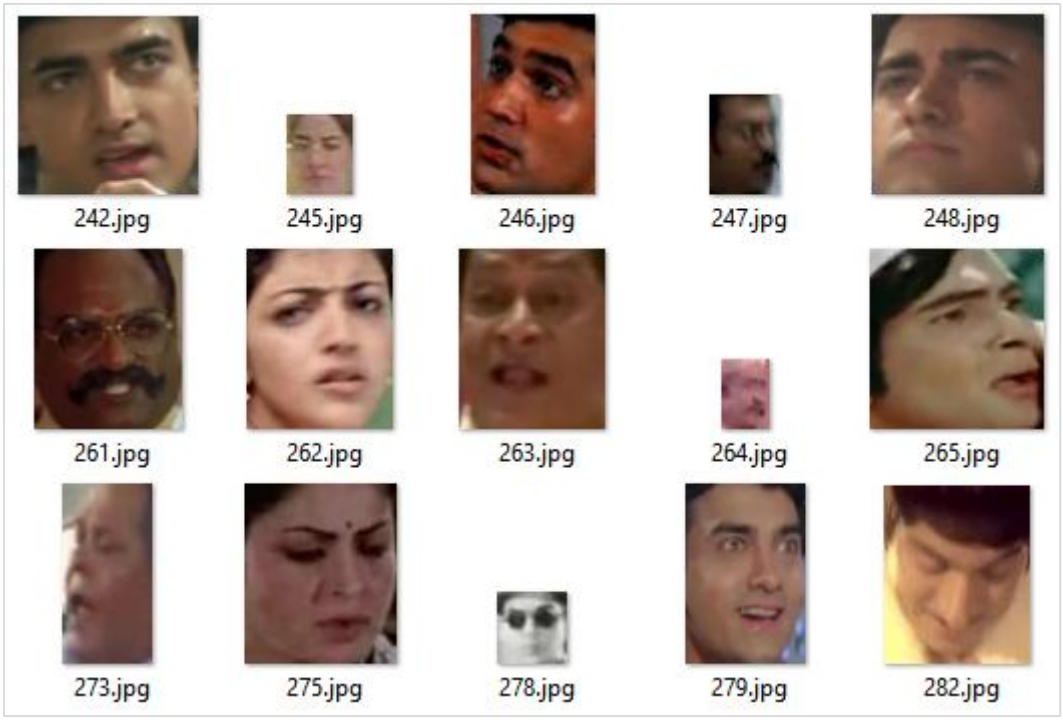
Hình 2.30 Mô hình mạng tổng quát

Tại hình trên (Hình 2.30) thể hiện các thông tin như sau. Cột thứ nhất là tên các lớp theo thứ tự từ trên xuống dưới là từ tầng đầu tiên nhận dữ liệu đầu vào đến tầng kết quả đầu ra cuối cùng, cột thứ hai biểu diễn kích thước cho mỗi dữ liệu đầu ra của từng lớp đó, cột thứ ba là số lượng trọng số (weights) cho cho từng lớp. Hàng cuối cùng là tổng hợp số lượng tham số cho cả mô hình.

2.5.4. Chuẩn bị dữ liệu

a. Nhập dữ liệu

Chúng ta sẽ dùng tập dữ liệu hình khuôn mặt người từ tập dữ liệu IMFDB, chứa khoảng 27642 ảnh màu, mỗi ảnh có mã ảnh riêng, có cùng định dạng tệp ".jpg" và được phân thành 3 độ tuổi khác nhau, trong đó nhãn phân loại đi kèm trong một bảng dữ liệu được lưu dưới dạng tệp ".csv". Mỗi một ảnh là một khuôn mặt của một người và có độ phân giải khác nhau theo từng ảnh, như hình minh hoạ bên dưới (Hình 2.31):



Hình 2.31 Bộ dữ liệu được sử dụng trong bài toán

Dữ liệu được lưu trữ trên một thư mục trong máy tính. Đầu tiên chúng ta sẽ khai báo các biến lưu trữ đường dẫn đến thư mục dữ liệu, bao gồm hai biến như:

"Train_data_dir" : Lưu đường dẫn đến dữ liệu ảnh của tập huấn luyện

"Test_data_dir" : Lưu đường dẫn đến dữ liệu ảnh của tập kiểm chứng

Được thực hiện trong ngôn ngữ python như sau:

```
# Define data directory
Train_data_dir = "C:/Users/Kin/Documents/Drive_D/Study and Learning/LuậnVăn_TN/Data/Face picture/train_DETg9GD"
Test_data_dir = "C:/Users/Kin/Documents/Drive_D/Study and Learning/LuậnVăn_TN/Data/Face picture/test_Bh8pGW3"
```

Sau đó nhập bảng dữ liệu lưu trữ thông tin nhãn phân loại dưới dạng file “csv”, tương tự như trên chúng ta khai báo hai biến :

“train” : Lưu bảng thông tin nhãn phân loại của tập huấn luyện

“test” : Lưu bảng thông tin nhãn phân loại của tập kiểm chứng

Được thực hiện trong ngôn ngữ python như sau:

```
# Import lable files
train = pd.read_csv(os.path.join(Train_data_dir, "train.csv"))
test = pd.read_csv(os.path.join(Test_data_dir, "test.csv"))
```

Sau khi khai báo và nhập bảng dữ liệu chúng ta sẽ thữ hiện tạo hai tập dữ liệu, tập huấn luyện và tập kiểm chúng. Vì mô hình mạng CNN mà chúng ta xây dựng yêu cầu dữ liệu dầu vào là bộ hình ảnh có kích thước bằng nhau và có kích thước hình vuông, có nghĩa là chiều dài bằng chiều rộng. Nên chúng ta sẽ thực hiện nhập dữ liệu ảnh lần lượt và chỉnh sửa kích thước của nó. Đầu tiên chúng ta khai báo các biến như sau:

“img_path” : Lưu đường dẫn đến mỗi hình ảnh

“img” : Lưu hình ảnh đã nhập vào

“temp” : Mảng để lưu chũ các hình ảnh

Chúng ta thực hiện bằng cách tạo hàm “Create_train_data()” như sau:

```
# Create the training data
temp = []
def create_train_data():
    for img_name in train.ID:
        img_path = os.path.join(Train_data_dir, img_name) # define image dir path and name
        img = io.imread(img_path) # read in the image
        img = cv2.resize(img, (IMG_SIZE, IMG_SIZE)) # resize image
        img = img.astype('float32')
        temp.append(img)
```

Tương tụ như dữ liệu kiểm chứng :

```
# Create the test data
temp1 = []
def create_test_data():
    for img_name in test.ID:
        img_path = os.path.join(Test_data_dir, img_name) # define image dir path and name
        img = io.imread(img_path) # read in the image
        img = cv2.resize(img, (IMG_SIZE, IMG_SIZE)) # resize image
        img = img.astype('float32')
        temp1.append(img)
```

Đoạn mã python trên thực hiện tạo hàm “Create_train_data()” trong đó nó thực hiện vòng lặp đọc lần lượt từng hình ảnh vào theo thứ tự trong bảng dữ liệu (được

lưu tại biến "train") và lưu trên biến "img" sao đó thực hiện chỉnh sửa kích thước bằng phương thức "resize" và cuối cùng gán vào mảng "temp". Tương tự với mã của tập kiểm chứng.

b. Định dạng dữ liệu

Trước khi cho dữ liệu vào mô hình mạng CNN chúng ta phải chuyển tập dữ liệu sang dạng mảng (Array) với kích thước theo yêu cầu của mô hình CNN. Ở đây chúng ta sẽ sử dụng thư viện hỗ trợ Pandas để chuyển sang mảng và chuyển dữ liệu sang dạng mảng như sau. Đầu tiên khai báo hai biến như:

"train_x" : Lưu dữ liệu mảng hai chiều của tập huấn luyện

"test_x" : Lưu dữ liệu mảng hai chiều của tập kiểm chứng

Và thực hiện đoạn mã sau:

```
create_train_data()
train_x = np.stack(temp)
train_x = np.array(train_x).reshape(-1, IMG_SIZE,IMG_SIZE, 3) # Reshape the image to input into CNN
train_x = train_x / 255. # Normalize image
```

Thực hiện tương tự với dữ liệu kiểm chứng:

```
create_test_data()
test_x = np.stack(temp1)
test_x = np.array(test_x).reshape(-1, IMG_SIZE,IMG_SIZE, 3) # Reshape the image to input into CNN
test_x = test_x / 255. # Normalize image
```

Đoạn mã trên thực hiện gọi đến hàm "Create_train_data()" đã khai báo trước đó để lấy dữ liệu trả về là mảng của các hình ảnh lưu trữ tại biến "temp", sau đó thực hiện chuyển sang dạng mảng numpy bằng cách gán vào biến "train_x" bằng phương thức "np.stack()" và áp dụng phương thức "np.array()" để gán dữ liệu sang dạng mảng numpy.

Kết quả chúng ta nhận được tại hình sau:

```
[[145.,  73.,  61.],
 [144.,  72.,  60.],
 [142.,  70.,  59.],
 ...,
 [ 82.,  32.,  22.],
 [ 81.,  32.,  21.],
 [ 81.,  32.,  21.]],

[[144.,  69.,  58.],
 [143.,  68.,  57.],
 [141.,  66.,  55.],
 ...,
 [ 96.,  39.,  29.],
 [ 94.,  38.,  28.],
 [ 94.,  38.,  28.]],

[[151.,  74.,  64.],
 [149.,  72.,  62.],
 [146.,  69.,  59.],
 ...,
 [103.,  41.,  31.],
 [100.,  40.,  30.],
 [100.,  40.,  30.]]],
```

Hình 2.32 Mảng biểu diễn dữ liệu hình ảnh

Kết quả nhận được là dữ liệu được định dạng theo mảng, Mỗi ảnh là một mảng NumPy 2 chiều, 128x128 pixel, với mỗi pixel có giá trị từ 0 đến 255, tương ứng với giá trị giải màu của từng điểm ảnh trong hình ảnh. Để hỗ trợ mô hình mảng thực hiện các phép toán nhanh hơn chúng ta sẽ thực hiện Normalize các giá trị điểm ảnh bằng cách chia mỗi giá trị điểm ảnh với 255. Kết quả nhận được sẽ là mảng với giá trị các điểm ảnh trong khoảng từ 0 đến 1 như hình sau:

```
[[[0.18431373 0.11764706 0.14117648]
  [0.18431373 0.11764706 0.14117648]
  [0.18431373 0.12156863 0.13725491]
  ...
  [0.13333334 0.08627451 0.05490196]
  [0.13725491 0.09411765 0.05882353]
  [0.13725491 0.09803922 0.05882353]]

 [[0.1764706  0.10980392 0.13333334]
  [0.1764706  0.11372549 0.12941177]
  [0.1764706  0.11372549 0.1254902 ]
  ...
  [0.13725491 0.08627451 0.05490196]
  [0.14117648 0.09803922 0.05882353]
  [0.14117648 0.10196079 0.0627451 ]]

 [[0.16470589 0.10588235 0.11764706]
  [0.16470589 0.10588235 0.11764706]
  [0.16470589 0.10588235 0.11764706]
  ...
  [0.14509805 0.09019608 0.05490196]
  [0.14117648 0.09019608 0.05490196]
  [0.14117648 0.09019608 0.05490196]]
```

Hình 2.33 Mảng biểu diễn dữ liệu hình ảnh sau khi thực hiện Normalize

c. Phân chia dữ liệu

Để thực hiện phân chia dữ liệu thành hai tập dữ liệu là tập huấn luyện và tập kiểm tra, chúng ta khai báo các biến như :

"train_x" để lưu giá trị dữ liệu của tập huấn luyện

"train_y" để lưu giá trị nhãn của tập huấn luyện

"valid_x" để lưu giá trị dữ liệu của tập kiểm tra

"valid_y" để lưu giá trị nhãn của tập kiểm tra

Sử dụng hàm hỗ trợ phân tập dữ liệu "train_test_split()" như sau :

```
# Split off the validation data
train_x, valid_x, train_y, valid_y = train_test_split(train_x, train_y, test_size=0.15, random_state=42)
```

Nhãn là một mảng của các số nguyên từ 0 đến 2, tương ứng với mỗi lớp độ tuổi "Young", "Old" và "Middle". Từ mảng số nguyên trên chúng ta phải thực hiện định dạng bằng kỹ thuật one-hot encoding trước khi đưa vào mô hình mạng. One-hot encoding là quá trình biến đổi từng giá trị thành các đặc trưng nhị phân chỉ chứa giá trị 1 hoặc 0. Mỗi mẫu trong đặc trưng phân loại sẽ được biến đổi thành một véc tơ có kích thước bằng số nhãn phân loại và chỉ với một trong các giá trị đó là 1. Chúng ta thực hiện định dạng cho hai mảng là "train_y" và "valid_y" như sau :

```
# Change lable to one-hot vector
lb = LabelEncoder()
train_y = lb.fit_transform(train_y.Class)
train_y = keras.utils.np_utils.to_categorical(train_y)
```

```
valid_y = lb.fit_transform(valid_y.Class)
valid_y = keras.utils.np_utils.to_categorical(valid_y)
```

Từ trên chúng ta sẽ nhận được kết quả biểu diễn dưới bảng dưới đây:

Bảng 2.5 Định dạng nhãn phân loại thành véc tơ

Nhãn	Số nguyên	Mảng đặc trưng nhị phân
Young	0	[0, 0, 1]
Old	1	[0, 1, 0]
Middle	2	[1, 0, 0]

Kết quả nhận được cuối cùng sau khi thực hiện các phương pháp chuẩn bị dữ liệu trên chúng ta sẽ có :

- 2 mảng "train_x" và "train_y" là tập huấn luyện. Mô hình sẽ học từ dữ liệu của 2 mảng này.
- 2 mảng "valid_x" vả "valid_y" là tập kiểm thử. Sau khi mô hình được huấn luyện xong, chúng ta sẽ chạy thử mô hình với dữ liệu đầu vào từ valid_x để lấy kết quả, và so sánh kết quả đó với dữ liệu đối ứng từ valid_y để kiểm thử chất lượng của mạng neuron.

2.5.5. Huấn luyện mô hình

Sau khi chúng ta đã chuẩn bị dữ liệu và định nghĩa được mô hình mạng CNN sử dụng để phân loại độ tuổi xong tại mục trên. Nhưng trước khi chúng ta đưa dữ liệu vào mô hình để thực hiện huấn luyện chúng ta sẽ áp dụng các kỹ thuật để hỗ trợ mô hình huấn luyện hiệu quả và đưa ra được kết quả phân loại với độ chính xác cao hơn, và trong một số trường hợp để tránh Overfitting dữ liệu huấn luyện. Các kỹ thuật sử dụng gồm có: tăng cường dữ liệu (Data Augmentation) và cân bằng trọng số (Weight balancing).

- Tăng cường dữ liệu:

Đầu tiên là khai báo hàm "ImageDataGenerator()" để thực hiện sinh dữ liệu từ tập dữ liệu huấn luyện mà chúng ta dụng. Để sử dụng được hàm này chúng ta phải nhập phương thức từ thư viện "Keras":

```
from keras.preprocessing.image import ImageDataGenerator

data_aug = ImageDataGenerator(
    featurewise_center = False,
    samplewise_center = False,
    featurewise_std_normalization = False,
    samplewise_std_normalization = False,
    zca_whitening = False,
    rotation_range = 45,
    width_shift_range = 0.2,
    height_shift_range = 0.2,
    horizontal_flip = True,
    vertical_flip = False
)

data_aug.fit(train_x)
```

Trong hàm sinh dữ liệu này bao gồm nhiều tham số để thao tác với dữ liệu của chúng ta, nhưng ở đây chúng ta chỉ thực hiện phép quay hình ảnh với số "rotation_rage" bằng 30 tương đương với quay với góc 30 độ, phép dịch chiều ngang "width_shift" 20% và phép dịch chiều dọc "height_shift" 20%.

- Cân bằng trọng số:

Chúng ta khai báo một biến "weights" để lưu giá trị trọng số của từng nhãn mà chúng ta thay đổi:

```
# Define weights
weights = {
    0:.21,
    1:.50,
    2:.29
}
```

Bước tiếp theo chúng ta sẽ định nghĩa biến và các hàm sử dụng trong quá trình huấn luyện, bao gồm các biến như số vòng chạy huấn luyện mô hình và số batch, hàm lỗi và hàm tối ưu. Đầu tiên chúng ta khai báo biến:

"epoch_num" : Số vòng chạy của mô hình

"batch_size" : Số lượng mẫu dữ liệu trong mỗi batch

```
epoch_num = 30
batch_size = 64
```

Với số vòng chạy "epoch" chúng ta khai báo với 30 vòng và "batch_size" là 64.

Tiếp theo chúng ta khai báo hai hàm sử dụng trong quá trình huấn luyện bằng cách gọi hàm "compile()" với các tham số của hàm là "loss" là hàm lỗi mà chúng ta sử dụng, "optimizer" là hàm tối ưu và "metrics" là độ đo.

```
# Compile model
model.compile(loss='categorical_crossentropy',
              optimizer= Adam(lr=0.0002),
              metrics=['accuracy'])
```

Trong mô hình này chúng ta khai báo hàm lỗi là "categorical_crossentropy", với thuật toán tối ưu sử dụng là "Adam" với tham số "lr" hay "learning rate" bằng 0,0002, và độ đo chúng ta khai báo Accuracy là độ chính xác của mô hình.

Trong thư viện keras chúng ta thực hiện quá trình huấn luyện mô hình bằng phương thức "Fit()", với các tham số là "x" là dữ liệu huấn luyện, "y" nhãn của dữ liệu huấn luyện, "batch size" số mẫu dữ liệu trong một batch, "epoch" số vòng huấn luyện của mô hình, "validation data" tập dữ liệu kiểm thử và "class weight" là mảng giá trị trọng số. chúng ta có thể thấy cụ thể như sau:

```
# Train the model
history = model.fit_generator(
                    train_x,
                    train_y,
                    batch_size = batch_num,
                    epochs = 4,
                    validation_data=(valid_x,valid_y),
                    verbose = 1,
                    class_weight = weights)
```

Thực hiện huấn luyện mô hình, quá trình huấn luyện được minh họa với hình ảnh bên dưới (Hình 2.34).

```
Train on 15924 samples, validate on 3982 samples
Epoch 1/30
15924/15924 [==============================] - 209s 13ms/step - loss: 0.7792 - accuracy: 0.6739 - val_loss: 0.7488 - val_accura
cy: 0.6768
Epoch 2/30
15924/15924 [==============================] - 165s 10ms/step - loss: 0.7651 - accuracy: 0.6814 - val_loss: 0.7444 - val_accura
cy: 0.6763
Epoch 3/30
15924/15924 [==============================] - 170s 11ms/step - loss: 0.7491 - accuracy: 0.6836 - val_loss: 0.7321 - val_accura
cy: 0.6899
Epoch 4/30
15924/15924 [==============================] - 179s 11ms/step - loss: 0.7492 - accuracy: 0.6878 - val_loss: 0.7267 - val_accura
cy: 0.6888
Epoch 5/30
15924/15924 [==============================] - 180s 11ms/step - loss: 0.7353 - accuracy: 0.6929 - val_loss: 0.7214 - val_accura
cy: 0.6755
Epoch 6/30
15924/15924 [==============================] - 180s 11ms/step - loss: 0.7224 - accuracy: 0.7003 - val_loss: 0.7036 - val_accura
cy: 0.6951
Epoch 7/30
15924/15924 [==============================] - 180s 11ms/step - loss: 0.7209 - accuracy: 0.7056 - val_loss: 0.6999 - val_accura
cy: 0.7004
Epoch 8/30
15924/15924 [==============================] - 180s 11ms/step - loss: 0.7065 - accuracy: 0.7080 - val_loss: 0.6964 - val_accura
cy: 0.6989
Epoch 9/30
15924/15924 [==============================] - 179s 11ms/step - loss: 0.7009 - accuracy: 0.7094 - val_loss: 0.6827 - val_accura
cy: 0.7017
Epoch 10/30
15924/15924 [==============================] - 179s 11ms/step - loss: 0.6966 - accuracy: 0.7077 - val_loss: 0.6820 - val_accura
cy: 0.7022
Epoch 11/30
15924/15924 [==============================] - 181s 11ms/step - loss: 0.6868 - accuracy: 0.7168 - val_loss: 0.6711 - val_accura
cy: 0.7049
Epoch 12/30
15924/15924 [==============================] - 180s 11ms/step - loss: 0.6759 - accuracy: 0.7192 - val_loss: 0.6743 - val_accura
cy: 0.7069
Epoch 13/30
15924/15924 [==============================] - 179s 11ms/step - loss: 0.6682 - accuracy: 0.7246 - val_loss: 0.6673 - val_accura
cy: 0.7104
Epoch 14/30
15924/15924 [==============================] - 182s 11ms/step - loss: 0.6676 - accuracy: 0.7264 - val_loss: 0.6567 - val_accura
cy: 0.7177
Epoch 15/30
15924/15924 [==============================] - 184s 12ms/step - loss: 0.6515 - accuracy: 0.7324 - val_loss: 0.6587 - val_accura
cy: 0.7137
Epoch 16/30
15924/15924 [==============================] - 183s 11ms/step - loss: 0.6431 - accuracy: 0.7373 - val_loss: 0.6588 - val_accura
cy: 0.7192
Epoch 17/30
15924/15924 [==============================] - 181s 11ms/step - loss: 0.6348 - accuracy: 0.7375 - val_loss: 0.6426 - val_accura
cy: 0.7250
Epoch 18/30
15924/15924 [==============================] - 180s 11ms/step - loss: 0.6239 - accuracy: 0.7452 - val_loss: 0.6406 - val_accura
cy: 0.7285
Epoch 19/30
15924/15924 [==============================] - 180s 11ms/step - loss: 0.6202 - accuracy: 0.7487 - val_loss: 0.6330 - val_accura
cy: 0.7318
Epoch 20/30
15924/15924 [==============================] - 180s 11ms/step - loss: 0.6114 - accuracy: 0.7558 - val_loss: 0.6350 - val_accura
cy: 0.7386
Epoch 21/30
15924/15924 [==============================] - 180s 11ms/step - loss: 0.6064 - accuracy: 0.7589 - val_loss: 0.6270 - val_accura
cy: 0.7464
```

Hình 2.34 Quá trình huấn luyện mô hình

2.6. Kết chương

Trong chương II, Luận văn trình bày giới thiệu về mạng nơ ron tích chập, cấu trúc mạng và các ứng dụng trên thực tế sử dụng mạng nơ ron tích chập. Giới thiệu về bộ dữ liệu sử dụng trong luận văn tiền xử lý dữ liệu và chuẩn bị cho mô hình mạng, sau đó xây dựng một mô hình mạng để giải quyết bài toán phân loại độ tuổi người và thực hiện huấn luyện mô hình. Trong chương tiếp theo tôi sẽ đánh giá kết quả huấn luyện và kiểm chứng của mô hình.

CHƯƠNG 3: CÀI ĐẶT VÀ THỬ NGHIỆM

3.1. Cài đặt môi trường thực hiện huấn luyện và thử nghiệm mạng nơ ron tích chập áp dụng trên bộ dữ liệu thực tế.

Trong quá trình triển khai và xây dựng mô hình của luận văn này tôi đã áp dụng các giải pháp phần mềm sau:

Ngôn ngữ lập trình Python

Python hiện là ngôn ngữ lập trình phổ biến nhất cho nghiên cứu và phát triển trong lĩnh vực học máy và học sâu. Python có nhiều các thư viện hỗ trợ trong việc tính toán số học, xử lý dữ liệu, xử lý hình ảnh, dễ dàng triển khai các thuật toán và xây dựng các mô hình học máy.

Trong luận văng này tôi ta sử dụng ngôn ngữ lập trình Python với phiên bản là Python 3.x cụ thể là phiên bản 3.6.1.

Công cụ và môi trường tích hợp mã nguồn Python sử dụng là Jupyter Notebook

Jupyter Notebook là một môi trường tính toán tương tác dựa trên web để tạo tài liệu sổ ghi chép Jupyter. Nó hỗ trợ một số ngôn ngữ như Python (IPython), Julia, R, v.v. và phần lớn được sử dụng để phân tích dữ liệu, trực quan hóa dữ liệu và tính toán tương tác, khám phá hơn nữa.

Các thư viện hỗ trợ Python:

- Keras: là Framework mã nguồn mở hỗ trợ xây dựng các thuật toán học máy và học sâu trên nền tảng ngôn ngữ Python. Đặc biệt tập trung vào thử nghiệm với các mô hình mạng học sâu. Thư viện này đơn giản hóa việc thực hiện các mạng thần kinh. Nó cũng có các chức năng tốt nhất cho các mô hình điện toán, đánh giá các tập dữ liệu, hiển thị biểu đồ và nhiều hơn nữa.

- Tensorflow: thư viện này được sử dụng phổ biến trong việc viết các thuật toán học máy và thực hiện các tính toán mà yêu cầu chi phí tính toán cao liên quan đến mạng nơ-ron.

- Pandas: Một thư viện khoa học dữ liệu mã nguồn mở Python mạnh mẽ, hỗ trợ việc thu thập dữ liệu vào các cấu trúc (data frame) rõ ràng, cung cấp phân tích trực quan và dễ dàng thao tác.

- OpenCV: (Open Source Computer Vision Library) là một thư viện mã nguồn mở về thị giác máy (computer vision) và học máy. Cung cấp nhiều tính năng thao tác với dữ liệu ảnh, xử lý ảnh.

- Mathplotlib: là thư viện hỗ trợ Python được xây dựng riêng cho việc biểu diễn dữ liệu và tạo các biểu đồ rất mạnh mẽ. Có rất nhiều công cụ để tạo nhãn, lưới, các biểu tượng v.v.

- Numpy: là một thư viện python được sử dụng đặc biệt để tính toán dữ liệu khoa học hoặc các phép tính toán học. cung cấp rất nhiều tính năng hữu ích cho các phần thực hiện phép toán trong n-arrays và ma trận trong. Thư viện này cung cấp khả năng vector hóa các vận hành về toán trong định dạng array NumPy, giúp cải thiện hiệu suất và theo đó là tốc độ tính toán.

- Scikit-learn: Là một thư viện phần mềm miễn phí cho học máy, bao gồm các thuật toán phân loại, hồi quy và phân cụm khác nhau. Được coi là một trong những thư viện tốt nhất để làm việc với dữ liệu phức tạp. Ngoài ra, Scikit-learn có thể được sử dụng kết hợp với NumPy và SciPy.

Mối trường cài đặt:

- Máy tính hệ điều hành Windows 10
- CPU Intel core i7 2680U 2.0GHz
- RAM 8 GB
- SSD 256 GB

3.2. Phương pháp đánh giá

Để đánh giá hiệu suất của bài toán phân loại văn bản chúng ta sử dụng các độ đo như: Accuracy, Precision, Recall, và bảng Confusion matrix. Được định nghĩa ở phần dưới. Để ước lượng các độ đo này, có thể dựa vào bảng sau:

Bảng 3.1 Bảng Confusion matrix

		Dự đoán	
		0	1
Nhãn	0	TN	FP
	1	FN	TP

Một số tiêu chí mô tả độ hiệu quả của mô hình phân loại bao gồm có:

- Accuracy : Khả năng mô hình phân loại dự báo chính xác, phân loại chính xác hay xác định đúng nhãn hoặc lớp đối với dữ liệu cần phân loại. Được tính bằng công thức:

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

- Precision : Được định nghĩa như là xác suất mà một dữ liệu phân loại là 1 là một phân loại đúng (độ chính xác của mỗi lần dự đoán). Được tính toán ước lượng như sau:

$$Precision = \frac{TP}{TP + FP}$$

- Recall : Được định nghĩa như là xác suất mà một dữ liệu với nhãn là 1 đã được phân loại đúng (độ chính xác của dự đoán cho từng nhãn). Được tính toán ước lượng như sau:

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

3.3. Đánh giá kết quả

Từ kết quả của quá trình huấn luyện trên, chúng ta thực hiện kiểm chứng mô hình sau đã huấn luyện trên với dữ liệu kiểm chứng đã chuẩn bị trước đó như sau:

```
# Python script for confusion matrix creation
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

test_pred = model.predict_classes(test_x)
test_pred = lb.inverse_transform(test_pred)

results = confusion_matrix(test_y.Class, test_pred)

print('Confusion Matrix :')
print(results)
print('Accuracy Score :',accuracy_score(test_y0.Class, test_pred))
print('Report : ')
print(classification_report(test_y.Class, test_pred))
```

Kết quả nhận được được thể hiện dưới đây:

```
Score = model.evaluate(test_x, test_y, verbose=0)
print("Accuracy: %.4f%%" % (Score[1]*100))
```

```
Accuracy: 86.3667%
```

Từ kết quả kiểm chứng với chỉ số đánh giá là độ chính xác phân lớp của mô hình chúng ta nhận được ở mức tương đương 86.37% với bộ dữ liệu kiểm chứng. Chúng ta sẽ xem xét với độ đo khác như Precision và Recall từ kết quả dưới đây:

```
Accuracy Score : 0.8636668905305575
Report :
              precision    recall  f1-score   support

      MIDDLE       0.92      0.84      0.88      1597
         OLD       0.78      0.82      0.80       374
       YOUNG       0.82      0.91      0.87      1007

    accuracy                           0.86      2978
   macro avg       0.84      0.86      0.85      2978
weighted avg       0.87      0.86      0.86      2978
```

Hình 3.1 Kết quả kiểm chứng mô hình

Kết quả nhận được từ ba độ đo trên có thể thể hiện dưới bảng sau:

Bảng 3.2 Kết quả phân loại của mô hình

Nhãn	Precision	Recall
Middle	0.92	0.84
Old	0.78	0.82
Young	0.82	0.91
Trung bình	0.84	0.86

Với số đo độ Precision cho thấy trong dữ liệu kiểm chứng thì 84% dữ liệu được phân loại đúng nhãn. Độ đo Recall cho chúng ta thấy là mẫu dữ liệu có nhãn phân loại được phân loại đúng nhãn của nó với xác suất là 86%.

Để đánh giá được mô hình cụ thể hơn chúng ta sẽ xem kết quả phân loại trên từng nhãn, xem mô hình có tỷ lệ phân nhãn của từng nhãn với độ chính xác là bao nhiêu bằng cách sinh bảng Confusion matrix dưới đây:

Bảng 3.3 Confusion matrix

		Dự đoán		
		Middle	Old	Young
Nhãn	Middle	1346	73	178
	Old	49	305	20
	Young	73	13	921

Bảng 3.3 là ma trận phân tích độ chính xác của dự đoán nhãn sau khi chạy mô hình. Chúng ta có thể nhận xét thấy có một số mẫu dữ liệu được phân loại sai đặc biệt là với mẫu có nhãn phân loại là “Middle” thường được nhận dạng sai sang Young nhiều hơn so với “Old”, và ngược lại các mẫu có nhãn phân loại là “Young” được nhận dạng sai nhiều hơn sang mẫu có nhãn “Middle” so với nhãn “Old”.Ở đây chúng ta có thể thấy lý do xảy ra lỗi này một phần là do hai độ tuổi này gần nhau, về mức tuổi cũng gần sát với nhau, vì tại nhãn phân loại “Young” sẽ phân loại cho những người có độ tuổi từ 13 đến 30, còn với nhãn “Middle” là những người có mức tuổi từ 31 đến 50. Do đó làm cho có sự tương tự nhau giữa những người có mức tuổi gần nhau như 29, 30 với 31 tuổi. Một nguyên nhân nữa của việc dự đoán sai nhãn bắt nguồn từ bộ dữ liệu ảnh đầu vào. Các bức ảnh mặt người trong bộ dữ liệu IMFDB

chưa thực sự rõ ràng để nhận ra được độ tuổi chính xác của người trong hình ảnh dù là con người như chúng ta. Có thể nhận thấy điều này với ví dụ sau (Hình 3.2):

Hình 3.2 Ví dụ hình ảnh có độ tuổi trẻ "Young" (trái) và ảnh có độ tuổi trung bình (Phải)

Từ bảng trên chúng ta có thể thấy được kết quả phân loại đúng, sai của từng nhãn như bảng 3.4 dưới đây:

Bảng 3.4 Kết quả phân loại theo từng nhãn

Nhãn phân loại	Phân loại đúng	Phân loại sai	Tổng số mẫu
Middle	1346	251	1597
Old	305	69	374
Young	921	86	1007

Tại lớp có nhãn phân loại độ tuổi là "Old" thì mô hình phân loại đúng 305 mẫu trên tất cả 374 mẫu, tương đương với 81.55%. Với lớp có nhãn phân loại là Middle, mô hình phân loại đúng 1346 mẫu trên tổng số 1597 mẫu, chiếm 84,28%. Tương tự, xác suất phân loại mẫu chính xác của nhãn "Young" là 91,46%.

Nhận xét: Với kết quả đạt được từ mô hình trên cho thấy độ chính xác phân loại của mô hình tương đối ổn định nhưng chưa đạt được kết quả tốt nhất, cũng như

độ chính xác phân lớp của nhãn độ tuổi già (Old) có độ chính xác tương đối thấp so với các lớp khác. Từ đó tôi đã nhận thấy được hai vấn đề chưa giải quyết được trong đo bài toán này là: Vấn đề về xử lý bộ dữ liệu mất cân bằng, do số tổng số mẫu của dữ liệu mang nhãn độ tuổi già chỉ chiếm 12% trong tất cả bộ dữ liệu. Vấn đề thứ hai là mô hình phân loại, chúng ta chưa xây dựng được mô hình tốt nhất để thực hiện bài toán phân loại độ tuổi với bộ dữ liệu này, do hạn chế về tài nguyên máy tính nên không khả năng xử lý mô hình mô hình với độ phức tạp cao hơn với số tham số mô hình cao hơn.

3.4. Kết chương

Trong chương này, tôi đã trình bày về môi trường cài đặt, ngôn ngữ lập trình và thư viện hỗ trợ được sử dụng và đưa ra kết quả, cũng như những phương pháp phân tích đánh giá mô hình từ đó đánh giá được những kết quả đạt được. Ngoài ra, tôi đã đưa ra những vấn đề ảnh hưởng đến độ chính xác của mô hình huấn luyện, từ đó cải thiện độ chính xác của mô hình trong những nghiên cứu sau này được tốt hơn.

KẾT LUẬN

Trong luận văn này, tôi đã đề xuất một mô hình học sâu sử dụng mạng CNN để nhận diện độ tuổi của người dựa vào hình ảnh khuôn mặt. Mô hình mới này cho phép sử dụng một số lượng nhỏ các tham số và đạt hiệu suất 86%. Trong tương lai gần, tôi đang có kế hoạch cải thiện độ chính xác của mô hình, đặc biệt là đối với ước lượng độ tuổi bằng cách thử áp dụng dữ liệu mới tự thu thập và giải quyết được vấn đề nhận dạng thời gian thực. Mặt khác, tôi sẽ áp dụng mô hình của tôi cho các bài toán khác trong lĩnh vực thị giác máy tính và học sâu.

Những kết quả hoạt động chính của luận văn:

Trong phạm vi luận văn , Tôi đã tìm hiểu được hai phương pháp tiếp cận để giải quyết bài toán phân loại độ tuổi dựa vào ảnh mặt người, đó là về phương pháp học sâu và phương pháp học máy truyền thống. Từ đó, lựa chọn phương pháp học sâu để giải quyết bài toán phân loại độ tuổi. Tiếp theo, đi sâu vào tìm hiểu và nghiên cứu một mô hình mạng phổ biến trong học sâu – đó là mạng nơ ron tích chập bao gồm các thành phần, kiến trúc mô hình mạng, chức năng và ứng dụng thực tế của nó. Cuối cùng là thực hiện xây dụng một mô hình mạng nơ ron tích chập cho bài toán phân loại độ tuổi người bằng hình ảnh. Trong đó thực hiện huấn luyện, điều chỉnh các thông số trong mạng và áp dụng các kỹ thuật để đạt được khả năng dự đoán độ tuổi với độ chính xác 86%.

Luận văn còn một số vấn đề tiếp tục phát triển. Do thời gian, kinh nghiệm, hạn chế về ngôn ngữ nên không tránh khỏi các sai sót, kính mong được sự thông cảm của các Thầy Cô, các Nhà khoa học.

Định hướng phát triển của luận văn

Hướng nghiên cứu tiếp theo của luận văn sẽ tập trung vào phần xây dựng mô hình mạng phân loại độ tuổi của người với độ chính xác cao hơn, có thể sử dụng mô hình áp dụng nhiều tầng tích chập theo kiến trúc mạng VGGNet để phân tích được các đặc trưng chi tiết hơn. Thử nghiệp áp dụng kỹ thuật over-sampling vào dữ liệu với nhãn phân loại là già ”Old”, để tăng thêm độ cân bằng của bộ dữ liệu.

Hà nội, tháng 5 năm 2020

DANH MỤC CÁC TÀI LIỆU THAM KHẢO

Tiếng Việt

[1] Phùng Thị Thu Trang, Ma Thị Hồng Thu, (2019), Một mô hình Deep learning nhẹ cho bài toán nhận dạng tuổi và giới tính sử dụng mạng CNN, Khoa ngoại ngữ – ĐH Thái Nguyên, ĐH Tân Trào.

Tiếng Anh

[2] Adit Deshpande Engineering at Forward, *A Beginner's Guide To Understanding Convolutional Neural Networks,* UCLA CS '19.

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

[4] N. Ramanathan and R. Chellappa, (2006), "Modeling age progression in young faces", in Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on, vol.1. IEEE, pp. 387–394.

[5] Tales Lima Fonseca,(Nov 3, 2017), *What's happening inside the Convolutional Neural Network? The answer is Convolution*, https://buzzrobot.com/whats-happening-inside-the-convolutional-neural-network-the-answer-is-convolution-2c22075dc68d

[6] Shankar Setty, Moula Husain, Parisa Beham, Jyothi Gudavalli, Menaka Kandasamy, Radhesyam Vaddi, Vidyagouri Hemadri, J C Karure, Raja Raju, Rajan, Vijay Kumar and C V Jawahar. *"Indian Movie Face Database: A Benchmark for Face Recognition Under Wide Variations"*

[7] L.Zhu, K.Wang, L.Lin, and L.Zhang, (2016), "Learning a light weight deep convolutional network for joint age and gender recognition",Pattern Recognition (ICPR), 2016 23rd International Conference on. IEEE, pp. 3282–3287.

Trang web

[8] https://dlapplications.github.io/2018-07-06-CNN/, truy cập ngày 23/04/2020

[9] https://www.analyticsvidhya.com/blog/2017/06/hands-on-with-deep-learning-solution-for-age-detection-practice-problem/?utm_source=practice-problem-age-detection&utm_medium=Datahack, FAIZAN SHAIKH, (JUNE 27, 2017), Hands on with Deep Learning – Solution for Age Detection Practice Problem, truy cập ngày 25/04/2020

[10] https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 , truy câjp ngày 29/06/2020

[11] https://www.freecodecamp.org/news/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050/ , truy câjp ngày 29/06/2020

[12] https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6 , truy câjp ngày 29/06/2020

[13] https://techblog.vn/dropout-trong-neural-network , truy câjp ngày 30/06/2020

[14] https://topdev.vn/blog/ung-dung-convolutional-neural-network-trong-bai-toan-phan-loai-anh/ , truy cập ngày 30/06/2020

[15] http://cvit.iiit.ac.in/projects/IMFDB/ , truy cập ngày 30/06/2020

[16] https://forum.machinelearningcoban.com/t/tong-hop-data-augmentation-trong-thi-giac-may-update-22-06-2019/5323 , truy cập ngày 30/06/2020

[17] https://quantrimang.com/su-khac-biet-giua-ai-hoc-may-va-hoc-sau-157948, truy cập ngày 30/06/2020