

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



Nguyễn Công Hòa

**NGHIÊN CỨU HỌC SÂU TRONG NHẬN DẠNG KHUÔN MẶT
ỨNG DỤNG CHO BÀI TOÁN ĐIỂM DANH TỰ ĐỘNG HỌC SINH**

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

HÀ NỘI - 2020

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



Nguyễn Công Hòa

**NGHIÊN CỨU HỌC SÂU TRONG NHẬN DẠNG KHUÔN MẶT
ỨNG DỤNG CHO BÀI TOÁN ĐIỂM DANH TỰ ĐỘNG HỌC SINH**

CHUYÊN NGÀNH: KHOA HỌC MÁY TÍNH

MÃ SỐ: 8.48.01.01

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

NGƯỜI HƯỚNG DẪN KHOA HỌC: TS. PHAN THỊ HÀ

HÀ NỘI - 2020

LỜI CAM ĐOAN

Tôi cam đoan đây là công trình nghiên cứu của riêng tôi được TS. Phan Thị Hà - giảng viên khoa Công nghệ thông tin 1 trường Học viện công nghệ bưu chính viễn thông hướng dẫn khoa học. Nguồn tài liệu của các tác giả, cơ quan, tổ chức nếu sử dụng thì tôi đều ghi rõ trong phần tài liệu tham khảo.

Tôi xin hoàn toàn chịu trách nhiệm về nội dung luận văn của mình.

Hà nội, ngày tháng năm 2020.

Học viên Cao học.

Nguyễn Công Hòa.

LỜI CẢM ƠN

Lời đầu tiên, tôi xin bày tỏ sự biết ơn chân thành và sâu sắc nhất tới TS. Phan Thị Hà - Giáo viên hướng dẫn khoa học, người đã tận tình hướng dẫn, hỗ trợ và giúp đỡ tôi trong quá trình nghiên cứu và hoàn thiện luận văn của mình.

Tôi xin gửi lời cảm ơn chân thành tới các thầy, các cô là giảng viên khoa Công nghệ thông tin 1 của trường Học viện công nghệ bưu chính viễn thông đã tận tình truyền đạt kiến thức và hướng dẫn cho tôi trong suốt quá trình học tập tại trường.

Tôi xin gửi lời cảm ơn tới những người thân trong gia đình tôi đã chăm lo cho tôi, động viên tôi, cảm ơn cơ quan nơi tôi đang công tác - trường THPT Thanh Oai B, huyện Thanh Oai, Hà Nội đã hết sức tạo điều kiện để tôi hoàn thành khóa học này.

Trong quá trình hoàn thành luận văn do thời gian và khả năng kiến thức còn hạn chế nên khó tránh khỏi những sai sót. Kính mong nhận được sự cảm thông, góp ý của các thầy các cô.

Tôi xin chân thành cảm ơn.

Hà nội, ngày tháng năm 2020.

Người viết

Nguyễn Công Hòa

MỤC LỤC

LỜI CAM ĐOAN	i
LỜI CẢM ƠN	ii
MỤC LỤC	iii
DANH MỤC CÁC THUẬT NGỮ VIẾT TẮT	vi
DANH MỤC CÁC HÌNH VẼ.....	vii
MỞ ĐẦU	1
Chương 1. TỔNG QUAN VỀ BÀI TOÁN ĐIỂM DANH TỰ ĐỘNG.....	3
1.1. Tổng quan về xử lý ảnh	3
1.1.1. Một số khái niệm.....	3
1.1.2. Các vấn đề của xử lý ảnh.....	3
1.1.3. Ứng dụng của xử lý ảnh trong thực tế.....	4
1.2. Bài toán nhận dạng khuôn mặt	5
1.2.1. Khái niệm.	5
1.2.2. Một số trở ngại của công nghệ nhận dạng khuôn mặt.....	6
1.2.3. Tầm quan trọng của bài toán nhận dạng khuôn mặt.....	7
1.2.4. Các ứng dụng đặc trưng của bài toán nhận dạng khuôn mặt.....	7
1.2.5. Xây dựng hệ thống nhận dạng khuôn mặt.....	8
1.2.6. Một số phương pháp nhận dạng khuôn mặt	9
1.3. Vai trò và tầm quan trọng của bài toán điểm danh tự động học sinh tại trường THPT Thanh Oai B, Huyện Thanh Oai, Hà Nội.	9
1.4. Kết luận chương	10
Chương 2. TÌM HIỂU VỀ HỌC SÂU VÀ MÔ HÌNH MẠNG NƠON TÍCH CHẬP.....	11
2.1. Tổng quan về Học máy (Machine learning).....	11
2.2. Các thuật toán Học máy.....	12
2.2.1. Học có giám sát (supervised learning).....	12
2.2.2. Học không giám sát (unsupervised learning)	12
2.2.3. Học bán giám sát (Semi-Supervised Learning).....	12

2.2.4. Học củng cố (Reinforcement learning)	12
2.3. Tìm hiểu về Học sâu (Deep learning).....	12
2.3.1. Học sâu là gì?	12
2.3.2. Lịch sử Học sâu	13
2.3.3. Tổng quan về mạng nơron nhân tạo	14
2.3.4. Ứng dụng của Học sâu	21
2.4. Tìm hiểu về CNN [2].....	24
2.5. Cấu trúc của CNN.....	25
2.5.1. Lớp tích chập (Convolution)	25
2.5.2. Lớp phi tuyến Relu.....	27
2.5.3. Lớp Pooling	27
2.5.4. Lớp Fully-connected (FC)	28
2.6. Huấn luyện mô hình CNN	29
2.7. Tìm hiểu về Multi-task Cascaded Convolutional Networks.....	31
2.7.1. Multi-task Cascaded Convolutional Networks là gì?	31
2.7.2. MTCNN Workflow	31
2.7.3. Lý do lựa chọn MTCNN để detect khuôn mặt	37
2.8. Tìm hiểu về mô hình ResNet	38
2.8.1. Giới thiệu về mô hình ResNet	38
2.8.2. Điểm nổi bật của mô hình ResNet.....	38
2.8.3. Kiến trúc ResNet	39
2.8.4. Mô hình ResNet.....	39
2.9. Kết luận chương	43
Chương 3. NHẬN DẠNG KHUÔN MẶT ỨNG DỤNG CHO BÀI TOÁN	
ĐIỂM DANH TỰ ĐỘNG	44
3.1. Xây dựng hệ thống nhận dạng khuôn mặt	44
3.1.1. Công nghệ sử dụng	44
3.1.2. Xây dựng hệ thống nhận dạng khuôn mặt.....	48
3.1.3. Xây dựng dữ liệu huấn luyện	49

3.1.4. Huấn luyện mô hình nhận dạng khuôn mặt	52
3.2. Lập trình nhúng cho thiết bị điểm danh	55
3.2.1. Máy tính nhúng raspberry Pi 4:.....	56
3.2.2. Cài đặt hệ điều hành	57
3.2.3. Xây dựng giao diện cho thiết bị	58
3.2.4. Xử lý nâng cao	60
3.3. Xây dựng cơ sở dữ liệu	63
3.4. Demo và đánh giá kết quả	64
3.5. Kết luận chương	65
KẾT LUẬN	66
DANH MỤC CÁC TÀI LIỆU THAM KHẢO.	67

DANH MỤC CÁC THUẬT NGỮ VIẾT TẮT

Từ viết tắt	Tiếng Anh	Tiếng Việt
AI	Artificial Intelligence	Trí tuệ nhân tạo
ANN	Artificial neural network	Mạng nơron nhân tạo
CNN	Convolutional Neural Network	Mạng nơron tích chập
Conv	Convolution	Tích chập
DL	Deep Learning	Học sâu
ML	Machine Learning	Học máy
MTCNN	Multi-task Cascaded Convolutional Networks	Mạng chuyển đổi xếp tầng đa tác vụ
MLP	Multi layer perceptron	Mạng nơron đa lớp
NMS	Non-Maximum Suppression	
RNN	Recurrent Neural Network	Mạng nơron tái phát
ResNet	Residual Network	Mạng dư
KNN	K-nearest neighbor	K-láng giềng
SGD	Stochastic Gradient Descent	

DANH MỤC CÁC HÌNH VẼ

Hình 1.1. Quy trình xử lý ảnh [1].....	3
Hình 1.2. Các bước cơ bản trong một hệ thống xử lý ảnh [1]	3
Hình 1.3. Nền ảnh phức tạp.....	6
Hình 1.4. Hệ thống nhận dạng khuôn mặt.....	8
Hình 2.1. Mối quan hệ giữa DL, ML và AI [11]	13
Hình 2.2. Các giai đoạn phát triển Học sâu [3]	14
Hình 2.3. Mô hình mạng nơron [12]	14
Hình 2.4. Kiến trúc 3 phần của ANN.....	15
Hình 2.5. Tế bào nơron nhân tạo.....	16
Hình 2.6. Một số hàm truyền phổ biến.....	17
Hình 2.7. Huấn luyện mạng ANN sử dụng lan truyền ngược.....	18
Hình 2.8. Quá trình học của nơron.....	19
Hình 2.9. Mô hình tính toán của một nơron.....	19
Hình 2.10. Tô màu ảnh đen trắng dựa trên Học sâu	23
Hình 2.11. Cấu trúc cơ bản của mạng Nơron Tích chập (Lecun, 1989).....	25
Hình 2.12. Phép tính Convolution [4].....	26
Hình 2.13: Mô tả hàm MaxPooling với cửa sổ 2x2 mà bước trượt bằng 2.....	28
Hình 2.14: Cấu trúc MTCNN.....	32
Hình 2.15: Kim tự tháp hình ảnh.....	32
Hình 2.16: Kernel tìm kiếm khuôn mặt.....	33
Hình 2.17: P-Net.....	33
Hình 2.18: R-Net.....	35
Hình 2.19: O-Net.....	36
Hình 2.20: Ví dụ MTCNN	37
Hình 2.21: So sánh độ chính xác.....	39
Hình 2.22: Một khối xây dựng của ResNet.....	39
Hình 2.23: Kiến trúc chi tiết của ResNet.....	40
Hình 2.24: Mô hình ResNet-101	40

Hình 2.25: Code ResNet Model	41
Hình 2.26: Code ResNet Model	41
Hình 3.1: Các bước thực hiện nhận dạng khuôn mặt sử dụng Resnet-101	48
Hình 3.2: Bộ dữ liệu xây dựng	52
Hình 3.3: Mô tả phương pháp tính độ lỗi.....	53
Hình 3.4: Mô tả phương pháp tính độ lỗi dựa trên điểm neo.....	54
Hình 3.5: Biểu đồ mô tả kết quả huấn luyện.....	55
Hình 3.6: Máy tính nhúng Raspberry pi cùng màn hình.....	56
Hình 3.7: Các thành phần cơ bản cần thiết cho thiết bị	57
Hình 3.8: Thuật toán xử lý ảnh trước khi cải tiến.	61
Hình 3.9: Thuật toán xử lý ảnh sau khi cải tiến.	62
Hình 3.10: Database hệ thống điểm danh	64

MỞ ĐẦU

Đảng, Nhà Nước ta xác định “Giáo dục và đào tạo là quốc sách hàng đầu” và để thực hiện tốt mục tiêu, nhiệm vụ chiến lược này, việc ưu tiên ứng dụng công nghệ thông tin trong quản lý và trong việc hỗ trợ các hoạt động dạy - học là một vấn đề rất cấp bách, đã được thể hiện qua rất nhiều văn bản chỉ đạo của Đảng, Nhà Nước và của Bộ giáo dục và đào tạo.

Việc quản lý học sinh trong các nhà trường phổ thông hiện nay hoàn toàn dựa theo hình thức thủ công, hiện tượng học sinh bỏ cả buổi học, bỏ tiết, ngồi học không đúng lớp mình học, học sinh không phải của nhà trường ... gây ra cho công tác kiểm diện và quản lý học sinh gặp nhiều khó khăn. Hơn nữa, phụ huynh cũng muốn giám sát xem con mình có mặt ở trường, ở lớp hay không? hiện đang là một nhu cầu rất lớn.

Học sâu đã và đang rất phát triển, được ứng dụng rộng rãi trong các bài toán nhận dạng như: nhận dạng hình ảnh, nhận dạng giọng nói, xử lý ngôn ngữ tự nhiên ... và thu được những thành tựu to lớn với độ chính xác ngày càng cao. Trong đó nhận dạng khuôn mặt để xác định danh tính, giao dịch, kiểm soát an ninh ... ngày càng trở nên phổ biến.

Xuất phát từ thực tế trên, đề tài “ *nghiên cứu Học sâu trong nhận dạng khuôn mặt ứng dụng cho bài toán điểm danh tự động học sinh*” với hy vọng có thể ứng dụng thành công mô hình Học sâu hiện đại trong việc xây dựng hệ thống điểm danh tự động dựa vào nhận dạng khuôn mặt, đặc biệt là ứng dụng cụ thể vào điểm danh tự động học sinh trong mỗi lớp học của trường THPT Thanh Oai B, huyện Thanh Oai, Hà Nội.

Mục tiêu của luận văn là nghiên cứu học sâu trong nhận dạng khuôn mặt ứng dụng cho bài toán điểm danh tự động đối với quá trình quản lý học sinh trong các nhà trường phổ thông hiện nay nhằm nâng cao chất lượng quản lý học sinh trong công tác giáo dục đào tạo của nhà trường.

Đối tượng nghiên cứu của luận văn: Mô hình mạng nơron tích chập và bài toán điểm danh tự động.

Phạm vi nghiên cứu của luận văn: Xây dựng hệ thống điểm danh tự động đối với học sinh trong lớp học tại trường THPT Thanh Oai B, huyện Thanh Oai, Hà Nội.

Nội dung của luận văn được trình bày trong ba chương với nội dung chính như sau:

Chương 1: Tổng quan về bài toán điểm danh tự động

Nội dung chính của chương 1 là tìm hiểu khái quát về xử lý ảnh và bài toán nhận dạng khuôn mặt.

Chương 2: Tìm hiểu về học sâu và mô hình mạng nơron tích chập

Nội dung chính của chương 2 là tìm hiểu tổng quan về học máy, Học sâu, mô hình mạng nơron tích chập (CNN) cũng như cách hoạt động, cấu trúc và việc huấn luyện của mô hình mạng nơron tích chập.

Chương 3: Nhận dạng khuôn mặt ứng dụng cho bài toán điểm danh tự động

Nội dung chính của chương 3 là trình bày chi tiết các bước xây dựng hệ thống điểm danh tự động dựa trên nhận dạng khuôn mặt.

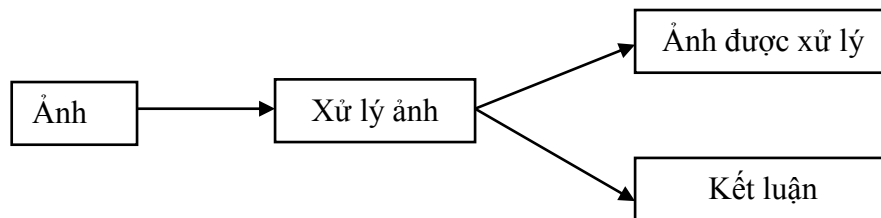
Chương 1. TỔNG QUAN VỀ BÀI TOÁN ĐIỂM DANH TỰ ĐỘNG

Chương này tập trung vào những khái niệm về xử lý hình ảnh, giới thiệu về bài toán nhận dạng khuôn mặt bao gồm các ứng dụng, tầm quan trọng và những khó khăn trở ngại hiện nay khi áp dụng bài toán về nhận dạng khuôn mặt, và cuối cùng những phương pháp nhận dạng khuôn mặt hiện nay. Chương này cũng chỉ ra vai trò và tầm quan trọng của bài toán điểm danh tự động học sinh tại trường THPT Thanh Oai B, Huyện Thanh Oai, Hà Nội.

1.1. Tổng quan về xử lý ảnh

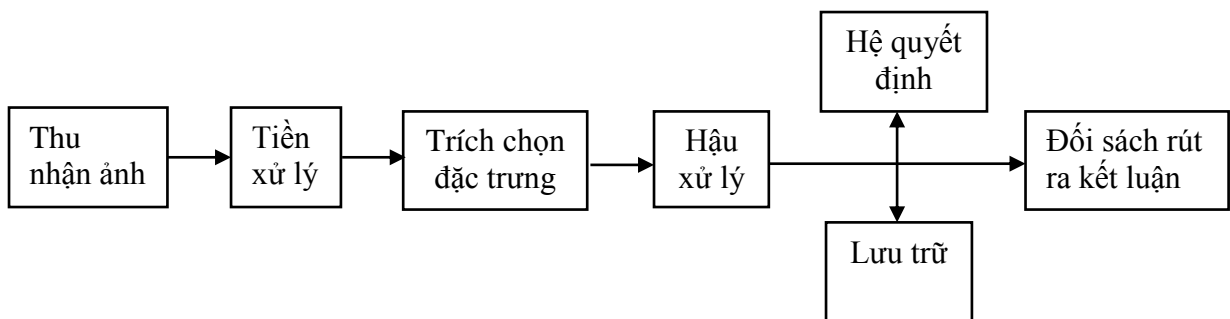
1.1.1. Một số khái niệm.

Xử lý ảnh là từ một ảnh đầu vào qua quá trình xử lý (thông qua các thuật toán) ta thu được một ảnh đã được xử lý hoặc một kết luận.



Hình 1.1. Quy trình xử lý ảnh [1]

Thông thường các ảnh tự nhiên, ảnh chụp có các tín hiệu ảnh đặc trưng bởi 2 đại lượng là biên độ và dải tần số. Nó chính là các đối tượng của xử lý ảnh.



Hình 1.2. Các bước cơ bản trong một hệ thống xử lý ảnh [1]

1.1.2. Các vấn đề của xử lý ảnh

a) Điều chỉnh mức xám của ảnh

Có 2 hướng chính là tăng số mức xám hoặc giảm số mức xám với mục đích chính là tăng cường độ mịn cho ảnh hoặc in ảnh màu ra máy in đen trắng.

b, Trích chọn đặc điểm

Tùy theo mục đích nhận dạng trong quá trình xử lý ảnh mà các đối tượng được trích chọn. Một số đặc điểm của ảnh như đặc điểm biến đổi (dựa vào lọc vùng), đặc điểm không gian (điểm uốn, phân bố xác suất, biên độ, mức xám ...) hay đặc điểm biên và đường biên.

c, Nhận dạng:

Hệ thống nhận dạng tự động bao gồm ba khâu tương ứng với ba giai đoạn chủ yếu sau đây:

1. Thu nhận dữ liệu và tiền xử lý.
2. Biểu diễn dữ liệu.
3. Nhận dạng, ra quyết định.

Bốn cách tiếp cận khác nhau trong lý thuyết nhận dạng là:

1. Đối sánh mẫu dựa trên các đặc trưng được trích chọn.
2. Phân loại thống kê.
3. Đối sánh cấu trúc.
4. Phân loại dựa trên mạng nơron nhân tạo.

d, Nén ảnh

Để giảm thiểu không gian lưu trữ ta cần nén ảnh. Có bốn kỹ thuật nén sau đây:

- Nén ảnh thống kê: Ví dụ *.TIF là mã nén theo kỹ thuật này.
- Nén ảnh không gian: Ví dụ *.PCX là mã nén theo kỹ thuật này.
- Nén ảnh sử dụng phép biến đổi: *.JPG là mã nén theo kỹ thuật này.
- Nén ảnh Fractal: Kỹ thuật nén sẽ tính toán để chỉ cần lưu trữ phần gốc

ảnh và quy luật sinh ra ảnh theo nguyên lý Fractal

1.1.3. Ứng dụng của xử lý ảnh trong thực tế.

❖ Xử lý và phục hồi hình ảnh: Ứng dụng này tương tự như photoshop: từ một hình ảnh được chụp từ máy ảnh, ta có thể chỉnh sửa, xử lý để làm ảnh đẹp hơn

hoặc phù hợp nhu cầu người dung như: làm mờ, lấy biên, chỉnh độ nét, chỉnh độ phân giải, phục hồi và nhận dạng ảnh....

❖ Lĩnh vực y tế: Các ứng dụng phổ biến của DIP trong lĩnh vực y tế là: Gamma ray imaging, PET scan, X Ray Imaging, Medical CT, UV imaging

❖ UV imaging: Lĩnh vực này liên quan nhiều đến thám hiểm, do thám. Cách hoạt động như sau: để phân tích thiệt hại của một trận động đất mà con người không thể tới được. Mặt đất nơi đó sẽ được quét bởi vệ tinh hoặc một máy bay sau đó truyền dữ liệu, hình ảnh về máy chủ để phân tích. Sẽ rất nhanh chóng so với việc chờ đợi con người tới đó. Một trận động đất có thể diện tích rất rộng mà con người không thể nào phân tích hết được.

❖ Truyền và mã hóa: Ngày nay con người sử dụng internet để truyền nhận các ảnh, video một cách nhanh chóng. Hình ảnh khi ta chụp sẽ được mã hóa và truyền theo internet. Rất nhanh sau vài giây là người bạn có thể nhận được một bức ảnh.

❖ Thị giác máy tính và robot: Hiện tại công nghệ robot đang phát triển nhanh chóng, và càng ngày càng giống con người hơn. Thị giác của máy tính cũng là một phần quan trọng. Làm thế nào để robot có thể nhìn mọi thứ, tránh vật cản, nhận dạng các vật..? Đó chính là nhờ một hệ thống quá trình xử lý ảnh phức tạp.

❖ Phát hiện vật cản: Phát hiện vật cản cũng là một lĩnh vực mới và được thực hiện bởi xử lý ảnh: tính toán khoảng cách từ robot tới vật cản bằng cách xác định được các đối tượng khác nhau trong hình ảnh sau đó xử lý và tính toán chúng.

❖ Công nghệ nhận dạng: Xử lý ảnh dùng để xác định, nhận dạng các đối tượng, các mối nguy hiểm, nhận dạng vân tay, khuôn mặt, hoặc các loại bệnh trong lĩnh vực y tế.

1.2. Bài toán nhận dạng khuôn mặt

1.2.1. Khái niệm.

Nhận dạng khuôn mặt là một loại phần mềm sinh trắc học ánh xạ các đặc điểm khuôn mặt của một cá nhân về mặt toán học và lưu trữ dữ liệu dưới dạng dấu khuôn mặt (faceprint). Công nghệ AI nhận dạng khuôn mặt là phần mềm sử dụng

các thuật toán Học sâu để so sánh ảnh chụp trực tiếp hoặc hình ảnh kỹ thuật số với ảnh được lưu trữ trong cơ sở dữ liệu để xác minh danh tính của một cá nhân.

1.2.2. Một số trở ngại của công nghệ nhận dạng khuôn mặt

❖ Góc chụp khuôn mặt: Chụp thẳng, chụp nghiêng, chụp hất lên ... Làm cho các thành phần trên khuôn mặt như mắt, mũi, miệng có thể bị khuất một phần hoặc thậm chí khuất hết, đều là những khó khăn rất lớn trong bài toán nhận dạng mặt người.

❖ Một số thành phần xuất hiện thêm hoặc không xuất hiện trên khuôn mặt như: đeo kính, đeo khẩu trang, trang điểm, mọc râu ... làm cho việc nhận dạng khuôn mặt thiếu chính xác.

❖ Khi con người thể hiện sự biểu cảm như: cười, khóc, nhăn mặt ... cũng ảnh hưởng đến kết quả nhận dạng.

❖ Ngoài ra một số tác nhân khác cũng gây ảnh hưởng đến kết quả như: Ảnh quá sáng, quá mờ, chất lượng ảnh ...

❖ Nền ảnh phức tạp: Nền của ảnh phức tạp là một trong những khó khăn nhất trong bài toán nhận dạng khuôn mặt người trong ảnh, khuôn mặt người sẽ dễ bị nhầm lẫn với nhiều khung cảnh phức tạp xung quanh và ảnh hưởng rất nhiều đến quá trình phân tích và rút trích các đặc trưng của khuôn mặt trong ảnh, có thể dẫn đến không nhận ra khuôn mặt hoặc là nhận nhầm các khung cảnh xung quanh thành khuôn mặt người.



Hình 1.3. Nền ảnh phức tạp

❖ **Màu sắc của da mặt:** Màu sắc của da mặt cũng đóng vai trò quan trọng trong nhận dạng khuôn mặt. Nếu màu sắc của da người quá tối hoặc gần với màu sắc của khung cảnh môi trường thì thuật toán sẽ gặp khó khăn trong việc nhận dạng các đặc trưng và có thể không tìm ra được khuôn mặt người.

1.2.3. Tầm quan trọng của bài toán nhận dạng khuôn mặt

Sinh trắc học được sử dụng để kiểm tra - xác thực danh tính con người thông qua một tập hợp các dữ liệu để nhận biết và kiểm chứng các đặc điểm cá biệt của người đó thông qua 2 bước “nhận dạng – Bạn là ai?” và “xác thực – Bạn thực sự là người bạn nói bạn là?”.

Hiện nay có các công nghệ sinh trắc học khác như: dấu vân tay, nhận dạng giọng nói, nhận dạng mống mắt, số hóa tổng thể lòng bàn tay và đo lường hành vi. Đây là các công nghệ sinh trắc tiên tiến được sử dụng để đảm bảo tính bảo mật cá nhân. Tuy nhiên các công nghệ trên cũng có những bất cập ví dụ công nghệ nhận dạng giọng nói có nhược điểm là tiếng ồn; công nghệ nhận dạng chữ ký cũng gây nhiều phiền phức cho người sử dụng vì khó duy trì được chữ ký giống nhau ngay trong cùng một thời điểm, công nghệ nhận dạng mống mắt lại bị tác động bởi nhiều yếu tố khác như độ rộng của mắt, lông mi, kính đeo và khó triển khai phổ biến trên diện rộng do độ phức tạp của các thiết bị.

Công nghệ nhận dạng khuôn mặt từ khi được phát minh vào năm 1970 đến nay đã có những bước tiến vượt bậc. Và ngày nay, nhận dạng khuôn mặt được xem là công nghệ đo sinh trắc học của con người tự nhiên nhất. Công nghệ nhận dạng khuôn mặt sẽ dễ triển khai và không giới hạn ứng dụng cũng như phạm vi triển khai của nó. Không có sự tương tác vật lý nào được yêu cầu bởi người dùng cuối. Hơn nữa, việc phát hiện khuôn mặt và các quy trình đối sánh khuôn mặt để xác minh/nhận dạng rất nhanh. Đây cũng là ưu điểm nổi trội của nhận dạng mặt người mà các công nghệ nhận dạng khác khó có thể có được.

1.2.4. Các ứng dụng đặc trưng của bài toán nhận dạng khuôn mặt

➤ Ứng dụng trong giám sát an ninh: các giải pháp kiểm soát an ninh (kiểm soát ra vào), nhận dạng khách lạ, khách VIP và đối tượng trong danh sách đen, tình nghi xuất hiện trong khu vực giám sát...

➤ Ứng dụng trong các ngành bán lẻ, dịch vụ: theo dõi lượng khách vào ra, nhận dạng khách hàng thân thiết, khách VIP và đối tượng xấu...

➤ Ứng dụng trong doanh nghiệp, công sở: chấm công khuôn mặt, bảo mật máy tính, quản lý ra/vào, phát hiện hành vi (cầm dao, đeo mặt nạ, đeo khẩu trang, đeo kính đen, để râu ở những nơi quan trọng).

➤ Ứng dụng trong chính phủ: giám sát giao thông thông minh, phát hiện các hành vi vi phạm giao thông.

➤ Ứng dụng trong trường học: điểm danh khuôn mặt, đăng ký, kiểm soát an ninh các khu vực cần theo dõi... là các giải pháp nhận dạng khuôn mặt cho trường học.

➤ Ứng dụng trong lĩnh vực Y tế - sức khỏe: Theo dõi việc sử dụng thuốc của bệnh nhân chính xác hơn, Phát hiện các bệnh di truyền như hội chứng DiGeorge với tỷ lệ thành công cao, hỗ trợ các thủ tục quản lý bệnh án.

➤ Ứng dụng trong các thiết bị IOT: thiết bị kiểm soát ra vào bằng khuôn mặt, thiết bị đọc giấy tờ tùy thân...

➤ Phân tích cảm xúc: Nhận biết cảm xúc trên khuôn mặt người.

1.2.5. Xây dựng hệ thống nhận dạng khuôn mặt.

Một hệ thống nhận dạng khuôn mặt có thể khái quát chung gồm có 3 bước cơ bản sau:



Hình 1.4. Hệ thống nhận dạng khuôn mặt

- Phát hiện khuôn mặt: Hệ thống nhận vào một ảnh tĩnh (từ bức hình hay một đoạn video), sau đó có thể xử lý ảnh cho chất lượng tốt hơn, như chỉnh lại độ sáng, giảm độ nhiễu ...

- Trích rút đặc trưng: Chính là việc phân tích và rút ra đặc điểm của khuôn mặt trong ảnh vì mỗi khuôn mặt có đặc điểm khác nhau (trừ các trường hợp sinh đôi cùng trứng).

- So sánh: Hệ thống sẽ so sánh các đặc điểm được trích rút với cơ sở dữ liệu khuôn mặt và sẽ quyết định kết quả so sánh có phù hợp hay không.

1.2.6. Một số phương pháp nhận dạng khuôn mặt

Dựa vào các tiêu chí mà người ta chia ra thành nhiều phương pháp nhận dạng khuôn mặt nhưng phổ biến hiện nay là các loại sau:

- + Phương pháp tiếp cận toàn cục.
- + Phương pháp tiếp cận dựa trên các đặc điểm cục bộ.
- + Phương pháp lai.

Khi làm việc trong điều kiện không có kiểm soát thì phương pháp tiếp cận dựa trên các đặc điểm cục bộ (trích chọn đặc trưng) tỏ ra thích hợp hơn hai phương pháp kia. Đó chính là các hệ thống phát hiện khuôn mặt người dựa trên tính năng (feature based).

Người ta cũng có thể chia thành hai hướng nhận dạng như làm với dữ liệu video (từ camera) hoặc làm với dữ liệu ảnh.

1.3. Vai trò và tầm quan trọng của bài toán điểm danh tự động học sinh tại trường THPT Thanh Oai B, Huyện Thanh Oai, Hà Nội.

Điểm danh là công việc được tiến hành hàng ngày và thường xuyên trong các buổi học tại các nhà trường phổ thông hiện nay trong đó có trường THPT Thanh Oai B, huyện Thanh Oai, Hà Nội.

❖ Thực trạng:

+ Học sinh thường đến lớp muộn, nghỉ học hoặc bỏ học cả buổi hoặc bỏ tiết học.

+ Hình thức điểm danh thủ công: Giáo viên đầu tiết học đếm số lớp ghi tên những học sinh vắng tiết học đó vào Sổ đầu bài.

+ Cuối tuần, cuối tháng, cuối học kỳ, cuối năm học giáo viên chủ nhiệm phải tổng hợp ngày nghỉ của học sinh để xếp thi đua và xét lên lớp cho học sinh. Công việc này rất mất thời gian, thiếu khách quan, thiếu chính xác.

❖ Giải pháp:

Cần có một hệ thống điểm danh tự động, mỗi phòng lắp một thiết bị điểm danh tại mỗi cửa phòng học. Giáo vụ, Ban giám hiệu có thể theo dõi kết quả điểm

danh học sinh vào bất kể thời gian nào trong buổi học. Cuối tháng, cuối kỳ tổng hợp và gửi danh sách cho giáo viên chủ nhiệm.

1.4. Kết luận chương

Trong chương 1, luận văn đã trình bày khái quát về xử lý ảnh và đặc biệt đã trình bày tương đối chi tiết về nhận dạng khuôn mặt ứng dụng cho bài toán của luận văn “*Điểm danh tự động học sinh*”. Trong chương tiếp theo, luận văn sẽ trình bày sơ lược về Học máy và hướng người đọc đến phần quan trọng là Học sâu.

Chương 2. TÌM HIỂU VỀ HỌC SÂU VÀ MÔ HÌNH MẠNG NƠON TÍCH CHẬP

Kỹ thuật Học sâu là một phạm trù nhỏ của lĩnh vực Học máy, Học sâu tập trung giải quyết các vấn đề liên quan đến mạng thần kinh nhân tạo (Artificial Neural Network - ANN) nhằm nâng cấp các công nghệ như nhận dạng giọng nói, thị giác máy tính và xử lý ngôn ngữ tự nhiên. Bởi vậy trong chương này, luận văn sẽ trình bày khái quát về Học máy, đi sâu vào kỹ thuật Học sâu cùng một số thuật toán và ứng dụng của nó trong thực tế, đồng thời cũng trình bày chi tiết về mô hình mạng nơon tích chập (CNN) cũng như cách hoạt động, cấu trúc và việc huấn luyện mô hình CNN.

2.1. Tổng quan về Học máy (Machine learning)

Học máy là một công nghệ phát triển từ lĩnh vực trí tuệ nhân tạo. Các thuật toán Học máy là các chương trình máy tính có khả năng học hỏi về cách hoàn thành các nhiệm vụ và cách cải thiện hiệu suất theo thời gian.

Học máy ra đời làm giảm bớt những hạn chế vốn có của AI khi nó mang lại cho máy tính khả năng có thể tìm ra mọi thứ mà không được lập trình rõ ràng. Học máy vẫn đòi hỏi sự đánh giá của con người trong việc tìm hiểu dữ liệu cơ sở và lựa chọn các kỹ thuật phù hợp để phân tích dữ liệu. Đồng thời, trước khi sử dụng, dữ liệu phải sạch, không có sai lệch và không có dữ liệu giả.

Các mô hình Học máy yêu cầu lượng dữ liệu đủ lớn để "huấn luyện" và đánh giá mô hình. Trước đây, các thuật toán Học máy thiếu quyền truy cập vào một lượng lớn dữ liệu cần thiết để mô hình hóa các mối quan hệ giữa các dữ liệu. Sự tăng trưởng trong dữ liệu lớn (big data) đã cung cấp các thuật toán Học máy với đủ dữ liệu để cải thiện độ chính xác của mô hình và dự đoán.

2.2. Các thuật toán Học máy

2.2.1. Học có giám sát (*supervised learning*)

Trong học có giám sát, máy tính học cách mô hình hóa các mối quan hệ dựa trên dữ liệu được gán nhãn (labeled data). Sau khi tìm hiểu cách tốt nhất để mô hình hóa các mối quan hệ cho dữ liệu được gán nhãn, các thuật toán được huấn luyện được sử dụng cho các bộ dữ liệu mới. Ví dụ xác định tín hiệu hay biến số tốt nhất để dự báo lợi nhuận trong tương lai của cổ phiếu hoặc dự đoán xu hướng thị trường chứng khoán.

2.2.2. Học không giám sát (*unsupervised learning*)

Trong học không giám sát, máy tính không được cung cấp dữ liệu được dán nhãn mà thay vào đó chỉ được cung cấp dữ liệu mà thuật toán tìm cách mô tả dữ liệu và cấu trúc của chúng. Ví dụ phân loại các công ty thành các nhóm công ty tương đồng dựa trên đặc điểm của chúng thay vì sử dụng tiêu chuẩn của các nhóm ngành hoặc các quốc gia.

2.2.3. Học bán giám sát (*Semi-Supervised Learning*)

Trong học bán giám sát, máy tính được cung cấp dữ liệu nhưng dữ liệu mới được dán nhãn một phần. Do đó phải sử dụng học không có giám sát (để khám phá và tìm hiểu cấu trúc dữ liệu đầu vào) và học có giám sát (để dự đoán cho dữ liệu không được gán nhãn) để giải quyết.

2.2.4. Học củng cố (*Reinforcement learning*)

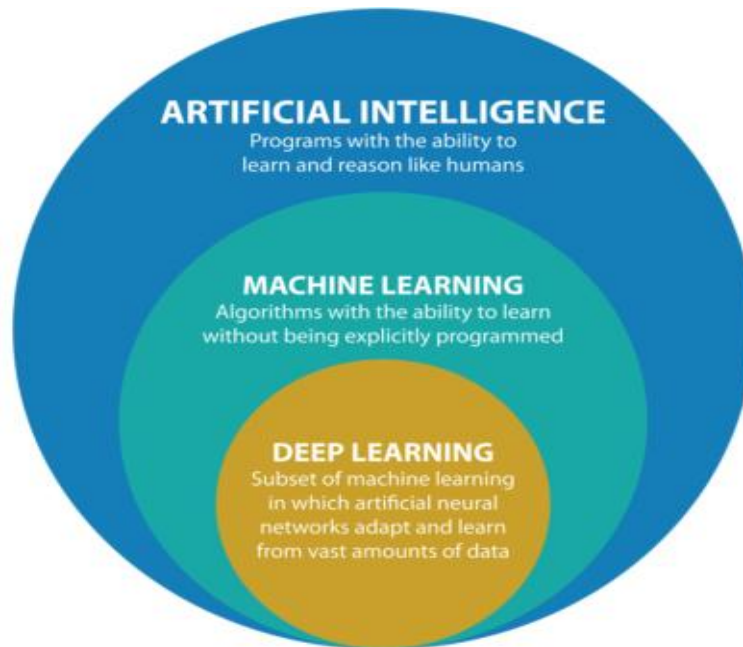
Học củng cố còn gọi là học tăng cường, thường ứng dụng vào Lý thuyết trò chơi thông qua một quá trình thử lỗi, hệ thống tự động xác định hành vi dựa trên hoàn cảnh để đạt được điểm số cao nhất. ví dụ như AlphaGo chơi cờ vây của Google [3]

2.3. Tìm hiểu về Học sâu (Deep learning)

2.3.1. Học sâu là gì?

Nếu xét về thời gian xuất hiện thì AI chính là ý tưởng xuất hiện sớm nhất tiếp theo đó là sự ra đời của học máy và cuối cùng gần đây nhất chính là Học sâu. Mặc dù xuất hiện muộn nhất nhưng Học sâu lại chính là thứ đang thúc đẩy sự bùng

phát của AI hiện nay. Nói một cách đơn giản nhất Học sâu là một tập hợp con của Học máy (Hình 2.1) và đề cập đến các mạng thần kinh có khả năng học dữ liệu đầu vào với các biểu diễn trừu tượng. Học sâu vượt trội hơn so với học máy truyền thống trong các vấn đề phức tạp như nhận dạng giọng nói, xử lý ngôn ngữ tự nhiên, phân loại hình ảnh ...



Hình 2.1. Mối quan hệ giữa DL, ML và AI [11]

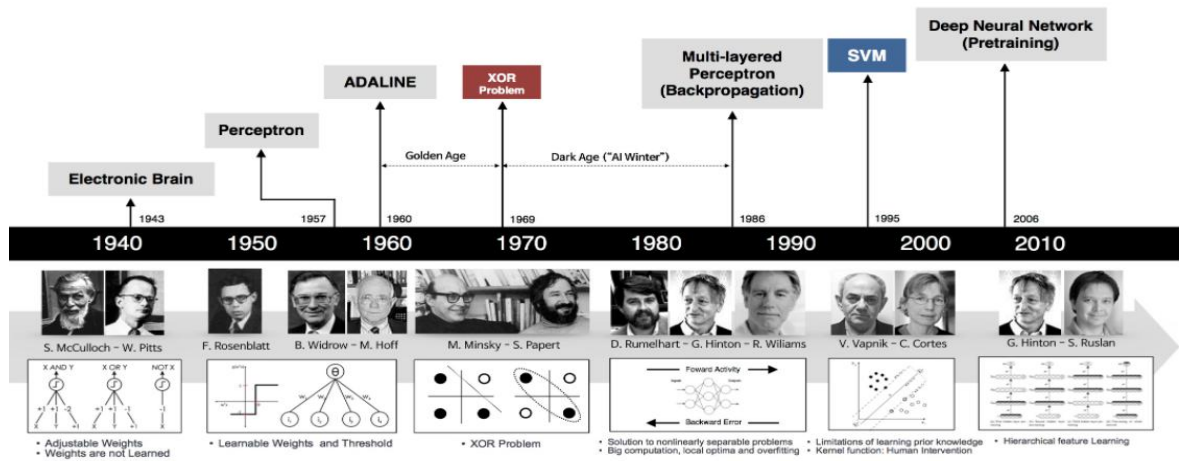
Học sâu khác với Học máy là thuật toán học sâu cần được đào tạo trong một thời gian dài vì có một số lượng lớn các tham số trong khi các thuật toán Học máy truyền thống có thể được đào tạo chỉ trong vòng vài giờ.

Học sâu đóng vai trò ngày càng lớn trong công nghệ hiện đại. Về mặt lý thuyết, nó có thể vượt quá khả năng của bộ não con người, vì chúng có các thuật toán nền tảng rộng lớn, được gọi là mạng lưới thần kinh. Ngày nay, các kỹ thuật học sâu có thể được tìm thấy ở một mật độ khá cao, từ xe tự lái đến các nghiên cứu học thuật.

2.3.2. Lịch sử Học sâu

Học sâu đã xuất hiện từ rất lâu, nhưng chỉ đến những năm gần đây mới thu hút và được nghiên cứu ứng dụng trong nhiều lĩnh vực.

Quá trình phát triển của Học sâu được thể hiện qua hình sau:



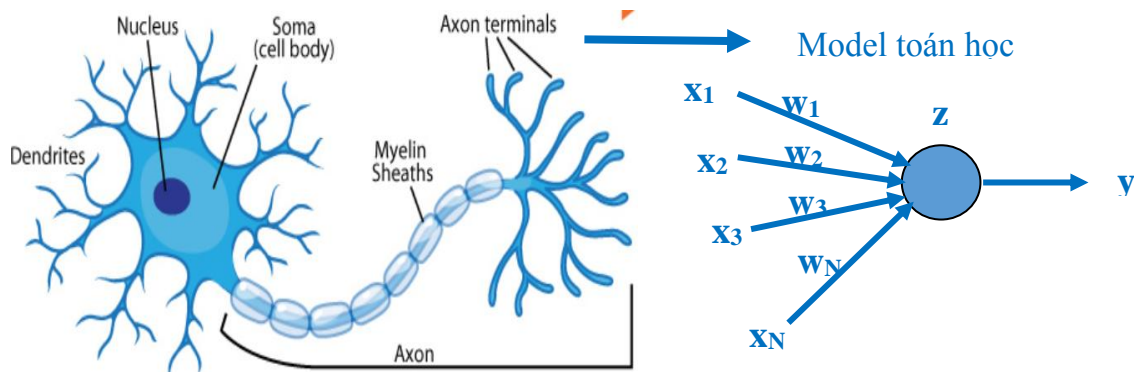
Hình 2.2. Các giai đoạn phát triển Học sâu [3]

2.3.3. Tổng quan về mạng nơron nhân tạo

a) Tìm hiểu về mạng nơron nhân tạo.

Mạng neural nhân tạo (Artificial Neural Network: ANN), gọi tắt neural network là mô hình xử lý thông tin mô phỏng hoạt động của các hệ neuron sinh học mà cụ thể hơn ở đây là bộ não con người. Trong hình 2.3 đầu tiên là tính chất truyền đi của thông tin trên neuron, khi neuron nhận tín hiệu đầu vào từ các dendrite, khi tín hiệu vượt qua một ngưỡng (threshold) thì tín hiệu sẽ được truyền đi sang neuron khác (Neurons Fire) theo sợi trục (axon).

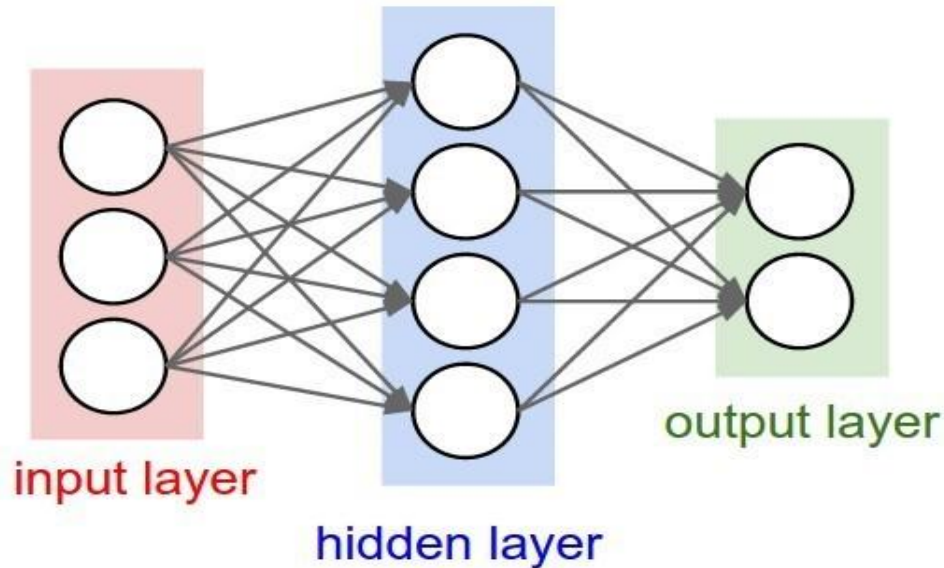
Neuron của model toán học ở đây cũng được mô phỏng tương tự như vậy. Trong đó, thành phần cơ bản của ANN là neuron nhân tạo có cách thức hoạt động và xử lý tương tự neuron sinh học. ANN được hình thành từ số lượng lớn các neuron được liên kết với nhau theo cấu trúc từng tầng (layer), các neuron kết nối với nhau giữa các tầng thông qua trọng số liên kết (weight).



Hình 2.3. Mô hình mạng nơron [12]

b) Kiến trúc ANN

Kiến trúc chung của ANN được mô tả trong hình 2.4 gồm 3 lớp: Lớp đầu vào (Input Layer), lớp ẩn (Hidden Layer) và lớp đầu ra (Output Layer).



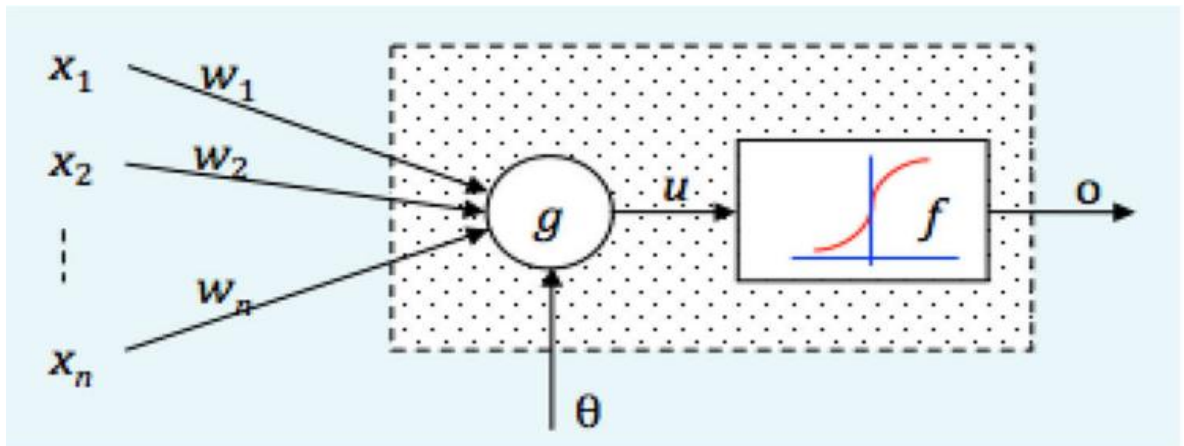
Hình 2.4. Kiến trúc 3 phần của ANN

Lớp đầu vào là một chuỗi các nơron kỹ thuật số có thể nhìn thấy thông tin mà máy tính được cung cấp. Ví dụ, một tế bào thần kinh có thể phát sáng khi màu xanh lục hiện dạng trong một hình ảnh, trong khi một tế bào khác có thể phát sáng khi có hình dạng cụ thể, một đặc tính cụ thể trong dữ liệu.

Lớp đầu ra cho máy tính biết phải làm gì để đáp ứng với dữ liệu đầu vào. Trong một chiếc xe tự lái, đây sẽ là những tế bào thần kinh kỹ thuật số cuối cùng bảo máy tính quyết định tăng tốc, phanh hoặc quay đầu.

Lớp ẩn chính là “Phép thuật” của mạng ANN. Lớp này lấy các nơron thần kinh từ lớp đầu vào và chuyển hướng để các nơron lớp đầu ra lấy được. Lớp ẩn bao gồm hàng ngàn hoặc hàng triệu nơron riêng lẻ, mỗi hàng được kết nối với nhau trong mạng lưới.

Hoạt động của ANN được mô tả như hình 2.5.



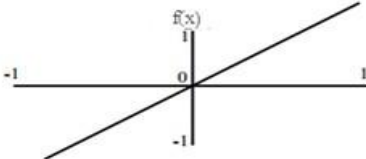
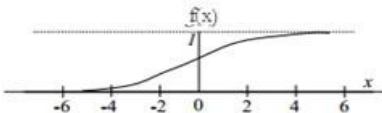
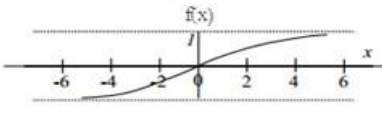
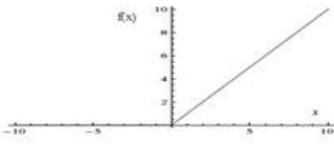
Hình 2.5. Tế bào nơron nhân tạo

➤ Các giá trị đầu vào x_1, x_2, \dots, x_n . Trong quá trình thực nghiệm dữ liệu này thường được đưa vào dưới dạng một véc tơ N chiều.

➤ Tập các liên kết: Có chức năng truyền thông tin trong đó ứng với một liên kết có một trọng số (hay trọng số liên kết là số thực biểu thị mức độ quan trọng của liên kết với đầu ra - weight) W_1, W_2, \dots, W_n . Thông thường, các trọng số này được khởi tạo một cách ngẫu nhiên ở thời điểm khởi tạo mạng và được cập nhật liên tục trong quá trình học mạng.

➤ Hàm kết hợp g (combination function): Thường dùng để tính tổng của tích các giá trị đầu vào với trọng số liên kết tương ứng của nó (vì thế một số tài liệu là hàm tổng – summing function).

➤ Hàm truyền f (transfer function) hay hàm kích hoạt (active function)}: Hàm này được dùng để giới hạn phạm vi đầu ra của mỗi neuron. Có đầu vào là kết quả của hàm kết hợp và ngưỡng đã cho. Thông thường, phạm vi đầu ra của mỗi neuron được giới hạn trong đoạn $[0,1]$ hoặc $[-1, 1]$. Các hàm truyền rất đa dạng, có thể là các hàm tuyến tính hoặc phi tuyến. Việc lựa chọn hàm truyền nào là tùy thuộc vào từng bài toán và kinh nghiệm của người thiết kế mạng. Bảng 2.6 dưới đây là các hàm truyền phổ biến:

Hàm Truyền	Công thức	Đồ thị
Linear	$f(x) = x$	
Sigmoid	$f(x) = \frac{1}{1 + e^{-x}}$	
Tan-Sigmoid	$f(x) = \frac{1 - e^x}{1 + e^x}$	
ReLU	$f(x) = \max(0, x)$	

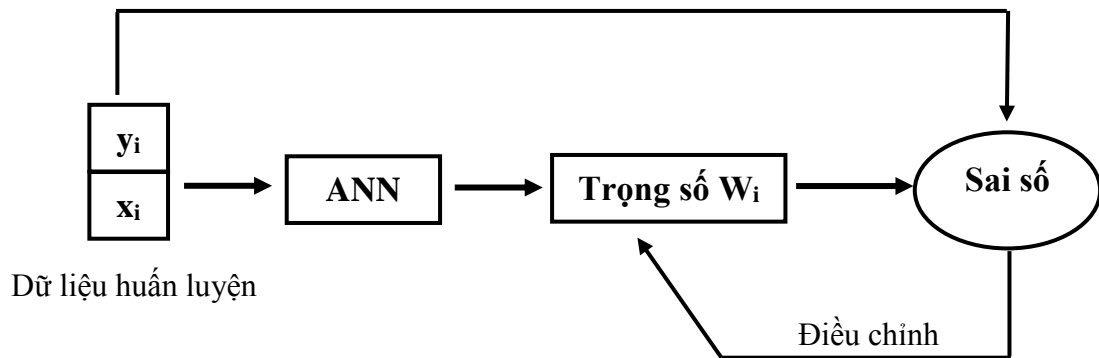
Hình 2.6. Một số hàm truyền phổ biến

c) Huấn luyện ANN

Chức năng của một mạng ANN được quyết định bởi các nhân tố như sau: hình trạng hay cấu trúc của mạng (số tầng, số neural trong mỗi tầng và cách các tầng liên kết với nhau) và các trọng số của các liên kết bên trong mạng. Kiến trúc mạng thường là cố định tương ứng với mỗi bài toán và các trọng số liên kết được quyết định bởi một thuật toán huấn luyện (training algorithm). Quá trình điều chỉnh các trọng số để mạng có thể nhận biết được mối quan hệ giữa đầu vào và đích mong muốn được gọi là học hay huấn luyện (learning - training). Có nhiều thuật toán dùng để huấn luyện mạng ANN trong đó thuật toán lan truyền ngược (back-propagation) được sử dụng phổ biến.

- Lan truyền ngược (back-propagation) là phương pháp huấn luyện mạng ANN với mục tiêu xác định trọng số tối ưu cho mạng thông qua việc lặp đi lặp lại 2 quá trình: lan truyền tiền (tính giá trị đầu ra của mạng từ đó tính sai số giữa giá trị này với giá trị mong muốn). Tiếp theo là quá trình lan truyền ngược sai số (dựa vào sai số sẽ cập nhật lại các trọng số). Mạng ANN có kiến trúc được tổ chức theo từng

tầng bao gồm tầng nhập, tầng xuất và một số tầng ẩn. Trong mỗi tầng có nhiều neural. Các neural tầng này liên kết với neural tầng kia thông qua cung liên kết chứa trọng số liên kết. Quá trình huấn luyện mạng bắt đầu bằng việc khởi tạo giá trị trọng số của mạng một cách ngẫu nhiên.

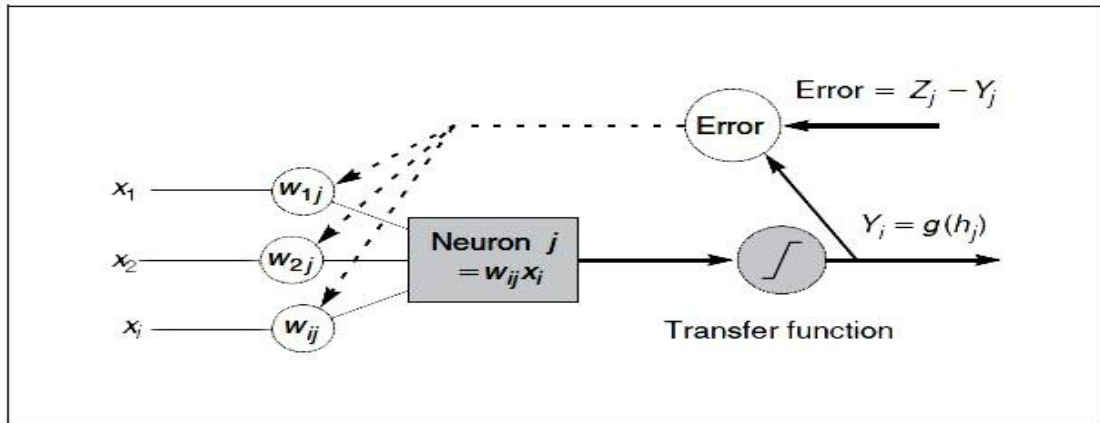


Hình 2.7. Huấn luyện mạng ANN sử dụng lan truyền ngược

+) Lan truyền tiến: Dữ liệu từ tập huấn luyện thông qua tầng nhập sẽ được chuyển vào tầng tiếp theo. Tại mỗi neural của mỗi tầng sẽ tiến hành thực hiện việc tính toán thông qua các hàm kết hợp, giá trị này sau khi truyền qua hàm kích hoạt là đầu ra mỗi neural. Việc tính toán sẽ thực hiện trên tất cả các neural của mạng và từ tầng nhập cho ra tới giá trị của tầng xuất. Sai số được tính bằng cách so sánh giá trị thực xuất ra của mạng với giá trị mong muốn, trong đó sai số của quá trình huấn luyện thường được lấy bằng tổng bình phương tất cả sai số thành phần.

+) Lan truyền ngược sai số: Dựa trên sai số được tính từ quá trình lan truyền tiến, mạng sẽ cập nhật lại các trọng số theo nguyên tắc lan truyền ngược sai số. Trong đó kỹ thuật cơ bản được áp dụng trong quá trình cập nhật trọng số đó là gradient descent.

Như vậy, để huấn luyện hay để một mạng học từ dữ liệu thì mạng thực hiện 2 bước lan truyền tiến và lan truyền ngược sai số. Quá trình này thực hiện cho tới khi sai số đạt được một ngưỡng nào đó hoặc thực hiện qua số bước lặp được người huấn luyện mạng đặt ra.



Hình 2.8. Quá trình học của nơron

d) Thuật toán lan truyền ngược (Back – Propagation)

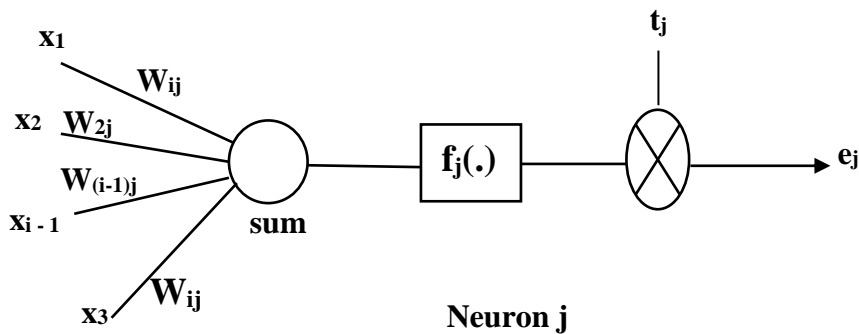
Thuật toán này [5] được sử dụng để điều chỉnh các trọng số kết nối sao cho tổng sai số e nhỏ nhất.

$$E = \sum_{i=1}^n (t(x_i, w) - y(x_i))^2 \quad (2-1)$$

Trong công thức trên thì giá trị tập mẫu là $t(x_i, w)$ và giá trị kết xuất của mạng là $y(x_i)$. Mỗi nơron đều có giá trị vào và ra và đều một trọng số, từ trọng số này để xác định mức độ ảnh hưởng của giá trị đó. Vấn đề đặt ra là làm thế nào để $e_j = t_j - y_j$ là nhỏ nhất

Kí hiệu:

- +) W_{ij} : là vector trọng số.
- +) u_j : là vector giá trị kết xuất.



Hình 2.9. Mô hình tính toán của một nơron

Khi đó ta lần lượt có công thức tính giá trị sai số, tổng bình phương sai số, tổng trọng số, giá trị kết xuất của nơon j theo các công thức sau:

$$e_j(n) = t_j(n) - y_j(n) \quad (2-2)$$

$$E(n) = \frac{1}{2} \sum_{j=1}^k e_j^2(n) \quad (2-3)$$

$$u_j(n) = \sum_{i=0}^p w_{ij} x_i(n) \quad (2-4)$$

$$y_j(n) = f_j(u_j(n)) \quad (2-5)$$

Tính toán giá trị đạo hàm sai số cho mỗi nơon w_{ij} :

$$\frac{\partial E(n)}{\partial w_{ij}(n)} = \frac{\partial E(n) \partial e_j(n) \partial y_j(n) \partial u_j(n)}{\partial e_j(n) \partial y_j(n) \partial u_j(n) \partial w_{ij}(n)} \quad (2-6)$$

Trong đó:

$$\frac{\partial E(n)}{\partial e_j(n)} = \frac{\frac{1}{2} \sum_{j=1}^k e_j^2(n)}{\partial e_j(n)} = \partial e_j(n); \quad (2-7)$$

$$\frac{\partial e_j(n)}{\partial y_j(n)} = \frac{\partial (t_j(n) - y_j(n))}{\partial y_j(n)} = -1; \quad (2-8)$$

$$\frac{\partial y(n)}{\partial u_j(n)} = f_j'(u_j(n)); \quad (2-9)$$

$$\frac{\partial u_j(n)}{\partial w_{ij}(n)} = \frac{\partial (\sum_{i=0}^p w_{ij} x_i(n))}{\partial w_{ij}(n)} = x_i(n) \quad (2-10)$$

$$\Rightarrow \frac{\partial E(n)}{\partial w_{ij}(n)} = -e_j(n) \cdot f_j'(u_j(n)) \cdot x_i(n) \quad (2-11)$$

Giá trị điều chỉnh trọng số được tính theo công thức:

$$\Delta w_{ij} = -\eta \frac{\partial E(n)}{\partial w_{ij}(n)} = -\eta e_j(n) \cdot f_j'(u_j(n)) \cdot x_i(n); \quad (2-12)$$

Đặt:

$$\partial_j = \frac{\partial E(n)}{\partial w_{ij}(n)} = \frac{\partial E(n) \partial e_j(n) \partial y_j(n)}{\partial e_j(n) \partial y_j(n) \partial u_j(n)} = e_j(n) \cdot f_j'(u_j(n)) \quad (2-13)$$

Suy ra:

$$\Delta w_{ij} = -\eta \delta_j(n) \cdot x_i(n); \quad (2-14)$$

ta điều chỉnh trọng số theo công thức sau:

$$w_{ij}(n+1) = w_{ij}(n) + \Delta w_{ij}(n) \quad (2-15)$$

Qua công thức điều chỉnh trọng số ta thấy việc xác định vị trí của noron thuộc lớp nào là rất quan trọng (lớp ẩn, lớp xuất).

e) Giảm lỗi cho mạng

❖ **Underfitting**: Là trường hợp mà mô hình huấn luyện của ta quá đơn giản, không thể giảm được loss function (hàm mất mát) nên không thể mô hình hóa được dữ liệu đào tạo cũng như dữ liệu mới. Đối với trường hợp này, ta cần phải phức tạp hóa mô hình của mình bằng cách thêm các lớp ẩn hoặc thêm noron trong mỗi lớp. Tuy nhiên việc làm phức tạp mô hình huấn luyện cũng cần phải cân nhắc bởi nếu mô hình trở nên quá phức tạp thì ta lại gặp phải một tình huống khác, đó là Overfitting.

❖ **Overfitting** (quá khớp): Là trường hợp mà mô hình huấn luyện của chúng ta “quá tốt” với bộ dữ liệu huấn luyện (training). Tức là, với mô hình đó, loss function gần như xấp xỉ bằng 0. Đáng lẽ ra, khi loss function càng nhỏ thì mô hình phải càng tốt. Nhưng đúng là cái gì quá cũng không tốt, khi trường hợp này xảy ra, mô hình trở nên quá cố gắng khớp với dữ liệu training và dẫn tới khi thực nghiệm với dữ liệu kiểm tra (testing) thì lại cho kết quả không cao.

Cách đơn giản để hạn chế vấn đề Overfitting là lại làm giảm độ phức tạp của mô hình đi nhưng với mức độ vừa đủ. Để làm được điều này, người thiết kế mô hình phải có kinh nghiệm cũng như phải thử nghiệm nhiều mô hình để có thể đánh giá và đưa ra được mô hình cuối cùng. Ngoài ra, việc dừng học sớm cũng là giải pháp để khắc phục tình trạng này.

2.3.4. Ứng dụng của Học sâu

Học sâu là thuật toán đòi hỏi một dữ liệu lớn để học tập, huấn luyện. Trước đây, khi internet còn chưa được sử dụng rộng rãi, Học sâu quả thực gặp rất nhiều khó khăn trong việc thu thập dữ liệu để có thể phát huy hết khả năng của nó. Nhưng hiện nay, nhờ có sự bùng nổ của internet, các mạng xã hội, mỗi ngày chúng ta tạo

ra khoảng 2.6 nghìn tỉ byte dữ liệu, đó là lượng dữ liệu rất quý báu cho việc phát triển Học sâu. Chính vì thế, Học sâu hiện đã, đang và sẽ phát triển hơn nữa trong tương lai. Sau đây là một số những ứng dụng, lợi ích mà Học sâu mang lại.

a) Nhận dạng giọng nói

Hiện nay các trợ lý ảo như Siri (Apple), Assistant (google), Alaxa (Amazon), Cortana (Microsoft) và gần đây là Bixby (Samsung) đều áp dụng Học sâu trợ giúp người sử dụng tương tác với máy tính qua giọng nói. Những trợ lý này hiện được trang bị cho những dòng loa thông minh lắp đặt trong những ngôi nhà được kết nối.

b) Dịch thuật

Dịch thuật bằng trí thông minh nhân tạo dựa trên nền tảng học sâu sử dụng mạng nơ-ron thần kinh và quá trình xử lý ngôn ngữ tự nhiên mang lại kết quả chính xác hơn so với công nghệ dịch dựa vào cụm từ trước đây của Google Translate. Theo Google, trí thông minh nhân tạo sẽ dịch toàn bộ câu văn cùng một lúc, thay vì dịch từng đoạn như trước đây, câu văn có cấu trúc ngữ pháp chính xác hơn so với phương pháp cũ, đồng thời tự hoàn thiện tốt hơn theo thời gian.

c) Ô tô tự vận hành hay máy bay không người lái.

Trong tương lai, nếu “trí tuệ nhân tạo” được nghiên cứu phát triển hơn và ứng dụng vào ngành hàng không hay công nghiệp ô tô thì có lẽ những chiếc xe không đơn thuần là những phương tiện di chuyển hằng ngày nữa mà sẽ là những sinh vật sống có tư duy, suy luận và cảm tính như con người. Và lúc đó chúng sẽ trở thành bạn đồng hành cùng sống, cùng phát triển ở một thế giới mới.

d) Tô màu, phục chế hình ảnh

Trước kia, quá trình phục chế màu ảnh thường được thực hiện thủ công trên phần mềm Photoshop. Cách này rất phức tạp vì nó đòi hỏi người phục chế phải am hiểu về màu sắc và các thang độ màu xám tương ứng, cũng như biết sử dụng thành thạo phần mềm Photoshop. Hiện nay có rất nhiều phần mềm chuyển đổi hình ảnh đen trắng thành ảnh màu, các phần mềm này dựa vào các thuật toán Học sâu, người phục chế ảnh chỉ cần một vài thao tác đơn giản để tô màu chúng và nhận được một

bức ảnh màu ấn tượng và chính xác, ngay cả khi không biết gì về màu sắc và photoshop.



Hình 2.10. Tô màu ảnh đen trắng dựa trên Học sâu

e) Nhận dạng khuôn mặt

Facebook đã phát triển một hệ thống nhận dạng khuôn mặt Deep Learning có tên là "DeepFace" để gắn thẻ mọi người đăng bài. Facebook sử dụng mạng thần kinh chín lớp có trọng lượng kết nối 120 triệu và được đào tạo trên 4 triệu hình ảnh kết nối được tải lên bởi người dùng Facebook. Hệ thống được cho là chính xác 97%.

Tương tự, FaceNet của Google được tuyên bố là một phương pháp rất chính xác để nhận dạng khuôn mặt đạt độ chính xác gần 86%. Nó có một bộ dữ liệu hình ảnh với gần 260 triệu hình ảnh từ khắp nơi trên thế giới và nó có thể đặt tên cho một khuôn mặt và trình bày hình ảnh phù hợp với tìm kiếm khuôn mặt.

f) Y học và dược phẩm

Phân tích các cuộc kiểm tra, quét CT, nhập dữ liệu và các nhiệm vụ khác có thể được thực hiện nhanh hơn và chính xác hơn bởi robot. Những ứng dụng theo dõi thuốc của bệnh nhân dựa trên Học sâu được phát triển nhanh chóng, tự động xác nhận rằng bệnh nhân đang dùng thuốc theo toa và giúp họ quản lý tình trạng của họ. Di truyền và hệ gen tìm kiếm các đột biến và liên kết với bệnh từ thông tin trong DNA. Với sự giúp đỡ của Học sâu quét cơ thể có thể phát hiện bệnh ung thư và các bệnh về máu sớm, dự đoán các vấn đề sức khỏe mà chúng ta phải đối mặt dựa trên

di truyền của chính mình. Ngoài ra các thiết bị theo dõi sức khỏe được đeo để theo dõi nhịp tim và mức độ hoạt động của con người ngày càng trở nên phổ biến.

g) Dịch vụ giải trí và mua sắm

Hiện nay các website xem film hay bán hàng trực tuyến đã ứng dụng Học sâu trong việc đưa ra các gợi ý cho khách hàng mà các đề xuất rất phù hợp với nhu cầu xem hoặc mua của khách hàng. Ví dụ như Amazon hay Netflix ... chẳng hạn.

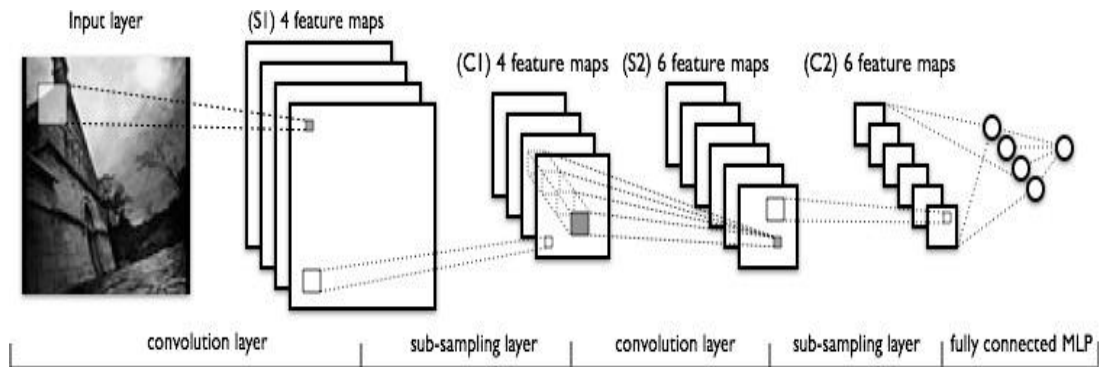
2.4. Tìm hiểu về CNN [2]

Mạng nơon tích chập (CNNs) (LeCun et al, 1989) là một mô hình học sâu có thể xây dựng được các hệ thống phân loại với độ chính xác cao. Ý tưởng của CNNs được lấy cảm hứng từ khả năng nhận biết thị giác của bộ não người. Để có thể nhận biết được các hình ảnh trong vỏ não người có hai loại tế bào là tế bào đơn giản và tế bào phức tạp (Hubel & Wiesel, 1968). Các tế bào đơn giản phản ứng với các mẫu hình dạng cơ bản ở các vùng kích thích thị giác và các tế bào phức tạp tổng hợp thông tin từ các tế bào đơn giản để xác định các mẫu hình dạng phức tạp hơn. Khả năng nhận biết các hình ảnh của não người là một hệ thống xử lý hình ảnh tự nhiên mạnh mẽ và tự nhiên nên CNNs được phát triển dựa trên ba ý tưởng chính: tính kết nối cục bộ (Local connectivity hay compositionality), tính bất biến (Location invariance) và tính bất biến đối với quá trình chuyển đổi cục bộ (Invariance to local transition) (LeCun et al., 2015). CNNs là một dạng mạng nơon chuyên dụng để xử lý các dữ liệu dạng lưới 1 chiều như dữ liệu âm thanh, dữ liệu MGE hoặc nhiều chiều như dữ liệu hình ảnh, video.

ANN được áp dụng nhiều trong các bài toán nhận dạng. Tuy nhiên, ANN không thể hiện tốt lắm đối với các dữ liệu hình ảnh. Chính sự liên kết quá đầy đủ tạo nên những hạn chế cho mô hình. CNN có kiến trúc hợp lý hơn so với mạng ANN, đặc biệt không giống một mạng thần kinh truyền thống, các lớp của một CNN sắp xếp theo 3 chiều: chiều rộng, chiều cao và chiều sâu.

Cấu trúc cơ bản của CNNs gồm các lớp tích chập (Convolution layer), lớp phi tuyến (Nonlinear layer) và lớp lọc (Pooling layer) như hình 1. Các lớp tích chập kết hợp với các lớp phi tuyến sử dụng các hàm phi tuyến như ReLU

($\text{ReLU}(x)=\max(x,0)$) [6] hay TanH để tạo ra thông tin trừu tượng hơn (Abstract/higher-level) cho các lớp tiếp theo.



Hình 2.11. Cấu trúc cơ bản của mạng Nơron Tích chập (Lecun, 1989)

Các lớp liên kết trong CNNs được với nhau thông qua cơ chế tích chập. Lớp tiếp theo là kết quả tích chập từ lớp trước đó vì vậy CNNs có được các kết nối cục bộ vì mỗi nơron ở lớp tiếp theo sinh ra từ một bộ lọc được áp đặt lên một vùng cục bộ của lớp trước đó. Nguyên tắc này được gọi là kết nối cục bộ (Local connectivity). Mỗi lớp như vậy được áp đặt các bộ lọc khác nhau. Một số lớp khác như lớp pooling/subsampling dùng để lọc lại các thông tin hữu ích hơn bằng cách loại bỏ các thông tin nhiễu. Trong suốt quá trình huấn luyện, CNNs sẽ tự động học các tham số cho các lớp. Lớp cuối cùng được gọi là lớp kết nối đầy đủ (Fully connect layer) dùng để phân lớp.

2.5. Cấu trúc của CNN

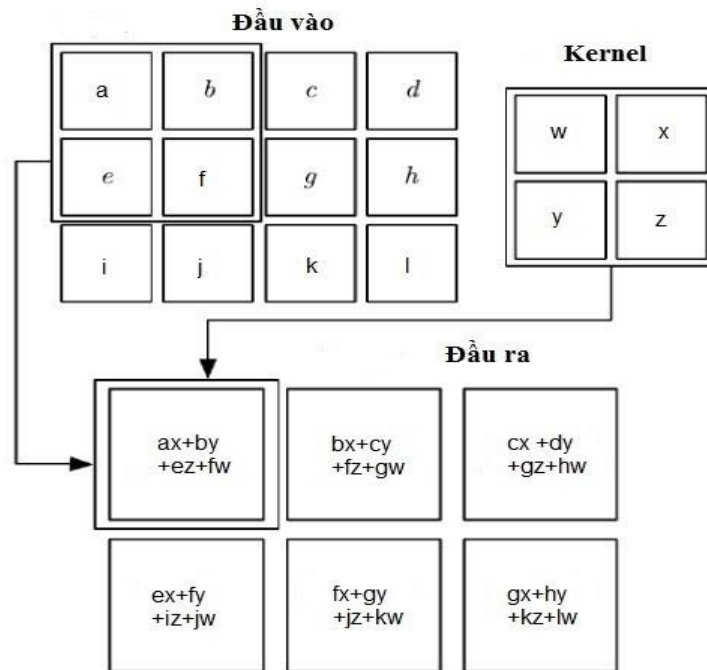
CNN là một kiểu mạng ANN truyền thẳng, trong đó kiến trúc chính gồm nhiều thành phần được ghép nối với nhau theo cấu trúc nhiều lớp đó là: Convolution, Pooling, ReLU và Fully connected.

2.5.1. Lớp tích chập (Convolution)

Lớp Convolution (Conv) là lớp quan trọng nhất trong cấu trúc của CNN. Tích chập được sử dụng đầu tiên trong xử lý tín hiệu số. Nhờ vào nguyên lý biến đổi thông tin có thể áp dụng kỹ thuật này vào xử lý ảnh và video số. Trong lớp tích chập sử dụng một bộ các bộ lọc có kích thước nhỏ hơn với ma trận đầu vào và áp lên một vùng của ma trận đầu vào và tiến hành tính tích chập giữa bộ filter và giá trị của ma trận trong vùng cục bộ đó. Các filter sẽ dịch chuyển một bước trượt (Stride) chạy

đọc theo ma trận đầu vào và quét toàn bộ ma trận. Trọng số của filter ban đầu sẽ được khởi tạo ngẫu nhiên và sẽ được học dần trong quá trình huấn luyện mô hình.

Hình 2.12 mô tả lý thuyết và cách thức Conv hoạt động trên một dữ liệu đầu vào được biểu diễn bằng một ma trận hai chiều. Ta có thể hình dung phép tính này được thực hiện bằng cách dịch chuyển một cửa sổ mà ta gọi là kernel trên ma trận đầu vào, trong đó kết quả mỗi lần dịch chuyển được tính bằng tổng tích chập (tích của các giá trị giữa 2 ma trận tại vị trí tương ứng). Khi được áp dụng phép tính Conv vào xử lý ảnh người ta thấy rằng Conv sẽ giúp biến đổi các thông tin đầu vào thành các yếu tố đặc trưng (nó tương ứng như bộ phát hiện – detector các đặc trưng về cạnh, hướng, đếm màu ...). Hình 2.12 là minh họa việc áp dụng phép tính Conv trên ảnh trong đó (a) là kết quả biến đổi hình ảnh khi thực hiện phép Conv khác nhau cho ra kết quả khác nhau, (b) là trực quan hóa các kernel dùng để detector các đặc trưng về cạnh, hướng, đếm màu



Hình 2.12. Phép tính Convolution [4]

Như vậy sử dụng Conv có những ưu điểm sau:

- Giảm số lượng tham số: Ở ANN truyền thống, các nơron ở lớp trước sẽ kết nối tới tất cả các nơron ở lớp sau (full connected) gây nên tình trạng quá nhiều tham

số cần học. Đây là nguyên nhân chính gây nên tình trạng overfitting cũng như làm tăng thời gian huấn luyện. Với việc sử dụng Conv trong đó cho phép chia sẻ trọng số liên kết (shared weights), cũng như thay vì sử dụng full connected sẽ sử dụng local receptive fields giúp giảm tham số.

- Các tham số trong quá trình sử dụng Conv hay giá trị của các filter – kernel sẽ được học trong quá trình huấn luyện. Như giới thiệu ở phần trên các thông tin này biểu thị thông tin giúp rút trích ra được các đặc trưng như góc, cạnh, đốm màu trong ảnh ... như vậy việc sử dụng Conv sẽ giúp xây dựng mô hình tự học ra đặc trưng.

2.5.2. Lớp phi tuyến Relu

Về cơ bản, convolution là một phép biến đổi tuyến tính. Nếu tất cả các neural được tổng hợp bởi các phép biến đổi tuyến tính thì một mạng neural đều có thể đưa về dưới dạng một hàm tuyến tính. Khi đó mạng ANN sẽ đưa các bài toán về logistic regression. Do đó tại mỗi neural cần có một hàm truyền dưới dạng phi tuyến. Có nhiều dạng hàm phi tuyến được sử dụng trong quá trình này như đã giới thiệu trong phần neural căn bản. Tuy nhiên, các nghiên cứu gần đây chứng minh được việc sử dụng hàm ReLu (Rectified Linear Unit) cho kết quả tốt hơn ở các khía cạnh:

- + Tính toán đơn giản
- + Tạo ra tính thưa (sparsity) ở các neural ẩn. Ví dụ như sau bước khởi tạo ngẫu nhiên các trọng số, khoảng 50% các neural ẩn được kích hoạt (có giá trị lớn hơn 0).
- + Quá trình huấn luyện nhanh hơn ngay cả khi không phải trải qua bước tiền huấn luyện.

Như vậy tầng ReLu cơ bản chỉ đơn giản là áp dụng hàm truyền ReLu.

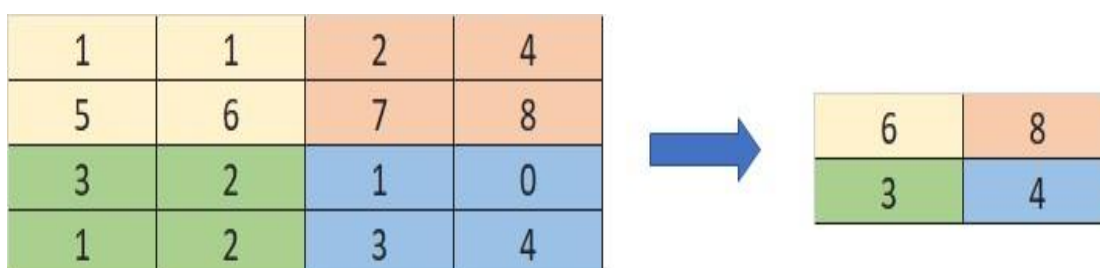
2.5.3. Lớp Pooling

Xét về mặt toán học pooling thực chất là quá trình tính toán trên ma trận trong đó mục tiêu sau khi tính toán là giảm kích thước ma trận nhưng vẫn làm nổi bật lên được đặc trưng có trong ma trận đầu vào.

Có nhiều toán tử pooling như Sum-Pooling, Max-Pooling, L2-Pooling nhưng Max-Pooling thường được sử dụng. Về mặt ý nghĩa thì Max-Pooling xác định vị trí cho tín hiệu mạnh nhất khi áp dụng một loại filter. Điều này cũng tương tự như là một bộ lọc phát hiện vị trí đối tượng bằng filter trong bài toán phát hiện đối tượng trong ảnh.

Về mặt lý thuyết với ma trận đầu vào có kích thước $W_1 \times H_1 \times D_1$ và thực hiện toán tử pooling trên ma trận con của ma trận đầu vào có kích thước $F \times F$ với bước nhảy S pixel thì ta được ma trận đầu ra $W_2 \times H_2 \times D_2$ đó:

- $W_2 = (W_1 - F)/S + 1$
- $H_2 = (H_1 - F)/S + 1$
- $D_2 = D_1$



Hình 2.13: Mô tả hàm MaxPooling với cửa sổ 2x2 mà bước trượt bằng 2

Công dụng của lớp Pooling dùng để giảm kích thước dữ liệu, các tầng trong CNNs chồng lên nhau có lớp Pooling ở cuối mỗi tầng giúp cho kích thước dữ liệu được co lại nhưng vẫn giữ được các đặc trưng để lấy mẫu. Ngoài ra giảm kích thước dữ liệu sẽ giảm số lượng tham số của mạng làm tăng tính hiệu quả và kiểm soát hiện tượng học vẹt (Overfitting).

2.5.4. Lớp Fully-connected (FC)

Lớp kết nối đầy đủ là một lớp giống như mạng nơron truyền thẳng các giá trị được tính toán từ các lớp trước sẽ được liên kết đầy đủ vào trong nơron của lớp tiếp theo. Tại lớp kết nối đầy đủ sẽ tiến hành phân lớp dữ liệu bằng cách kích hoạt hàm softmax để tính xác suất ở lớp đầu ra.

2.6. Huấn luyện mô hình CNN

Ta chọn ngẫu nhiên trọng số và giá trị bộ lọc ngay từ ban đầu, tiếp theo ta gán nhãn cho mỗi hình ảnh đi qua mô hình CNN, Tập các ảnh phải đủ lớn và đã được gán nhãn cho biết cụ thể ảnh đó là gì. CNN cũng sử dụng thuật toán lan truyền ngược tương tự như ANN để huấn luyện mô hình.

Mô tả huấn luyện một mạng CNN như sau:

Quá trình lan truyền thẳng: Truyền một hình ảnh cần đào tạo có kích thước 32x32x3 qua toàn bộ mạng. Các trọng số và bộ lọc trong trường hợp này đã được khởi tạo ngẫu nhiên, đầu ra trong trường hợp này tương tự như: [.1 .1 .1 .1 .1 .1 .1 .1 .1 .1] có thể nói với đầu ra kể trên không ưu tiên cho một đối tượng cụ thể nào do đó sẽ không có kết luận hợp lý ở đây, do vậy dẫn đến hàm mất mát được lan truyền ngược.

Hàm mất mát (loss function): Loss function có thể được định nghĩa theo nhiều cách khác nhau nhưng luận văn sẽ trình bày hàm mất mát cho Softmax Regression:

$$J(W; X, Y) = - \sum_{i=1}^N \sum_{j=1}^C y_{ji} \log(a_{ji}) = - \sum_{i=1}^N \sum_{j=1}^C y_{ji} \log \left(\frac{\exp(w_j^T x_i)}{\sum_{k=1}^C \exp(w_k^T x_i)} \right) \quad (2-16)$$

Giả sử biến L tương đương với đại lượng trên, chúng ta dễ dàng nhận thấy giá trị của hàm Loss function sẽ là rất cao với một số hình ảnh đào tạo đầu tiên. Mục tiêu của mô hình là nhận dự đoán giống nhãn đào tạo, để làm được điều này ta cần phải giảm thiểu giá trị loss function mà mô hình đang gặp phải. Để giải quyết vấn đề này ta cần tìm ra trọng số nào (weight) làm ảnh hưởng đến loss function của mô hình, tiếp theo ta cần thực hiện backward pass đi qua mô hình để xử lý vấn đề trên.

Lan truyền ngược (backward pass): Xác định trọng số ảnh hưởng nhất cho loss function và tìm cách để làm tối thiểu loss function, một khi chúng ta thực hiện được công việc này chúng ta sẽ đi đến bước cuối cùng là cập nhật trọng số. Để làm tối thiểu loss function ta phải tối ưu hàm mất mát.

Tối ưu hàm mất mát: Một lần nữa, chúng ta lại sử dụng Stochastic Gradient Descent (SGD) ở đây.

Với chỉ một cặp dữ liệu (x_i, y_i) , ta có:

$$\begin{aligned}
 J_i(W) &\triangleq J(W; x_i, y_i) = - \sum_{j=1}^C y_{ji} \log \left(\frac{\exp(w_j^T x_i)}{\sum_{k=1}^C \exp(w_k^T x_i)} \right) \\
 &= - \sum_{j=1}^C \left(y_{ji} w_j^T x_i - y_{ji} \log \left(\sum_{k=1}^C \exp(w_k^T x_i) \right) \right) \\
 &= - \sum_{j=1}^C y_{ji} w_j^T x_i + \log \left(\sum_{k=1}^C \exp(w_k^T x_i) \right)
 \end{aligned} \tag{2-17}$$

Trong biến đổi ở dòng cuối cùng, chúng ta đã sử dụng quan sát: $\sum_{j=1}^C y_{ji} = 1$ vì nó là tổng các xác suất.

Công thức tiếp theo ta sử dụng là:

$$\frac{\partial J_i(W)}{\partial W} = \left[\frac{\partial J_i(W)}{\partial w_1}, \frac{\partial J_i(W)}{\partial w_2}, \dots, \frac{\partial J_i(W)}{\partial w_C} \right] \tag{2-18}$$

Trong đó, gradient theo từng cột có thể tính được dựa theo công thức (2-17):

$$\begin{aligned}
 \frac{\partial J_i(W)}{\partial w_j} &= -y_{ji} x_i + \frac{\exp(w_j^T x_i)}{\sum_{k=1}^C \exp(w_k^T x_i)} x_i \\
 &= -y_{ji} x_i + a_{ji} x_i = x_i (a_{ji} - y_{ji}) \\
 &= e_{ji} x_i \text{ (where } e_{ji} = a_{ji} - y_{ji} \text{)}
 \end{aligned} \tag{2-19}$$

Trong đó: $e_{ji} = a_{ji} - y_{ji}$ là sai số dự đoán.

Từ (2-18), (2-19):

$$\frac{\partial J_i(W)}{\partial W} = x_i [e_{1i}, e_{2i}, \dots, e_{Ci}] = x_i e_i^T \tag{2-20}$$

Ta dễ dàng suy ra:

$$\frac{\partial J(W)}{\partial W} = \sum_{i=1}^N x_i e_i^T = X E^T \tag{2-21}$$

với $E=A-Y$. Công thức tính gradient đơn giản thế này giúp cho cả Batch Gradient Descent, Stochastic Gradient Descent (SGD), và Mini-batch Gradient Descent đều có thể dễ dàng được áp dụng.

Giả sử rằng chúng ta sử dụng SGD, công thức cập nhật cho ma trận trọng số W sẽ là:

$$W = W + \eta x_i (y_i - a_i)^T \quad (2-22)$$

Trong đó:

- * W là trọng số lúc sau cập nhật.
- * W_i là trọng số lúc đầu.
- * η tỉ lệ học.

Tỉ lệ học (learning rate) là một tham số được khởi tạo bởi người lập trình. Nếu tỉ lệ học quá cao thì dẫn đến bước nhảy lớn và như thế thì khó các định được các điểm tối ưu cần tìm.

Quá trình trên được lặp lại liên tục trong một thời gian nhất định cho mỗi tập hình ảnh đào tạo.

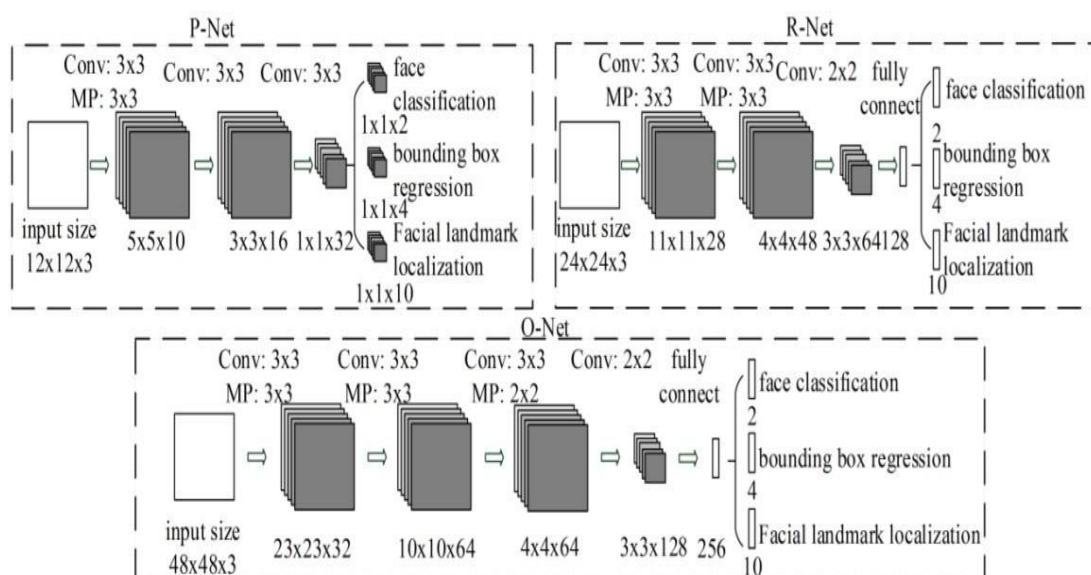
2.7. Tìm hiểu về Multi-task Cascaded Convolutional Networks

2.7.1. Multi-task Cascaded Convolutional Networks là gì?

Multi-task Cascaded Convolutional Networks được viết tắt là MTCNN. Nó bao gồm 3 mạng CNN xếp chồng và đồng thời hoạt động khi detect khuôn mặt. Mỗi mạng có cấu trúc khác nhau và đảm nhiệm vai trò khác nhau trong task. Đầu ra của MTCNN là vị trí khuôn mặt và các điểm trên mặt như: mắt, mũi, miệng...

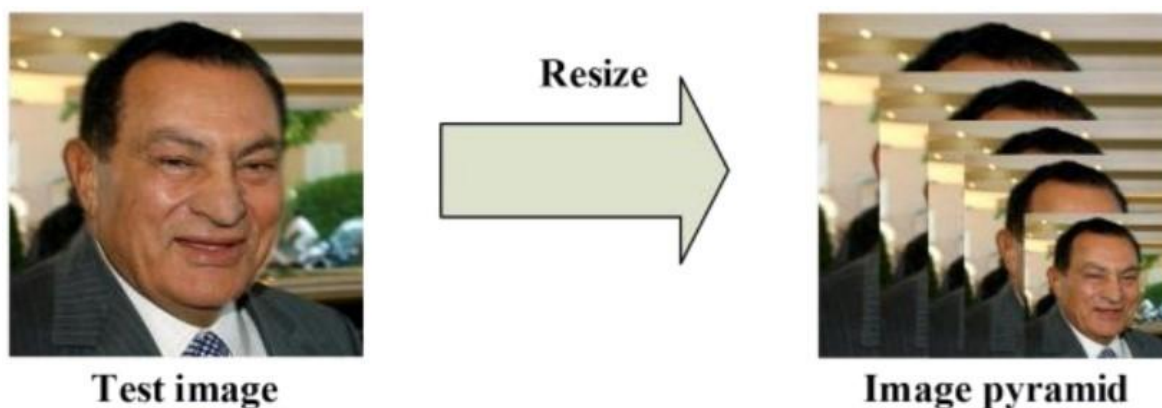
2.7.2. MTCNN Workflow

MTCNN hoạt động theo ba bước [5], mỗi bước dùng một mạng nơ-ron riêng lần lượt là: mạng đề xuất P-Net (Proposal Network) nhằm dự đoán các vùng trong ảnh ví dụ là vùng chứa khuôn mặt; mạng tinh chế R-Net (Refine Network) sử dụng đầu ra của P-Net để loại bỏ các vùng không phải khuôn mặt; và mạng đầu ra (Output Network): sử dụng đầu ra R-Net để đưa ra kết quả cuối cùng với 5 điểm đánh dấu khuôn mặt: 2 điểm mắt, 1 điểm mũi và 2 điểm khóe miệng (Hình 2.14)



Hình 2.14: Cấu trúc MTCNN

Đối với mỗi hình ảnh truyền vào, mạng tạo ra một kim tự tháp hình ảnh nghĩa là nó sẽ tạo ra nhiều bản sao của hình ảnh đó ở các kích thước khác nhau. Mục đích của việc này là để phát hiện các khuôn mặt ở tất cả các kích thước khác nhau.



Hình 2.15: Kim tự tháp hình ảnh

a. Mạng P-Net

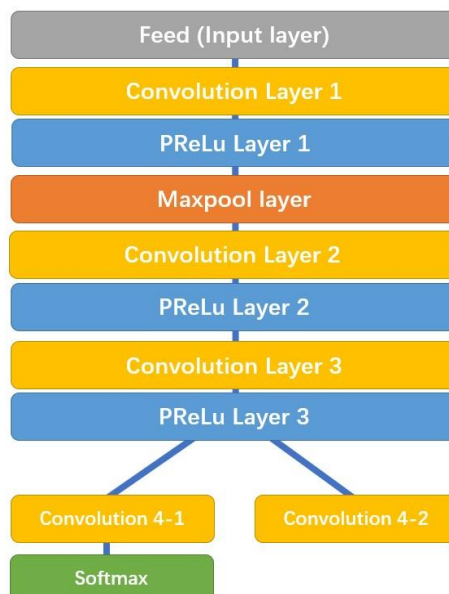
Tại P-Net, đối với mỗi hình ảnh được chia tỉ lệ, hạt nhân (kernel) 12x12 chạy qua hình ảnh đó để tìm kiếm khuôn mặt. Trong hình ảnh bên dưới, hình vuông màu đỏ đại diện cho hạt nhân từ từ di chuyển qua lại hình ảnh tìm kiếm khuôn mặt.



Hình 2.16: Kernel tìm kiếm khuôn mặt

Trong mỗi hạt nhân 12×12 này, có 3 cấu trúc được chạy qua với hạt nhân 3×3 . Sau mỗi lớp convolution, một lớp prelu được triển khai. Ngoài ra, một lớp max pool được đưa vào sau lớp prelu đầu tiên (maxpool lấy ra mọi pixel khác, chỉ để lại lớp lớn nhất trong vùng lân cận).

Sau lớp convolution thứ ba, mạng chia thành 2 lớp. Các kích hoạt từ lớp thứ ba được chuyển đến hai lớp convolution riêng biệt và một lớp softmax sau một trong các lớp convolution đó. Trong trường hợp này, nó tạo ra 2 xác suất: xác suất mà ở đó là một khuôn mặt trong khu vực và xác suất mà ở đó không phải là một gương mặt



Hình 2.17: P-Net

Convolution 4-1 đưa ra xác suất của một khuôn mặt nằm trong mỗi bounding boxes, và Convolution 4-2 cung cấp tọa độ của các bounding boxes.

Cấu trúc P-Net trong MTCNN:

```
class PNet(Network):
    def _config(self):
        layer_factory = LayerFactory(self)
        layer_factory.new_feed(name='data', layer_shape=(None, None, None, 3))
        layer_factory.new_conv(name='conv1', kernel_size=(3, 3),
channels_output=10, stride_size=(1, 1),
padding='VALID', relu=False)
        layer_factory.new_prelu(name='prelu1')
        layer_factory.new_max_pool(name='pool1', kernel_size=(2, 2),
stride_size=(2, 2))
        layer_factory.new_conv(name='conv2', kernel_size=(3, 3),
channels_output=16, stride_size=(1, 1),
padding='VALID', relu=False)
        layer_factory.new_prelu(name='prelu2')
        layer_factory.new_conv(name='conv3', kernel_size=(3, 3),
channels_output=32, stride_size=(1, 1),
padding='VALID', relu=False)
        layer_factory.new_prelu(name='prelu3')
        layer_factory.new_conv(name='conv4-1', kernel_size=(1, 1),
channels_output=2, stride_size=(1, 1), relu=False)
        layer_factory.new_softmax(name='prob1', axis=3)
        layer_factory.new_conv(name='conv4-2', kernel_size=(1, 1),
channels_output=4, stride_size=(1, 1),
input_layer_name='prelu3', relu=False)
```

b. Mạng R-Net

R-Net có cấu trúc tương tự, nhưng có nhiều lớp hơn. Nó lấy đầu ra của P-Net làm đầu vào và đưa ra tọa độ hộp giới hạn chính xác hơn.



Hình 2.18: R-Net

Cấu trúc R-Net trong MTCNN:

```

class RNet(Network):
    def _config(self):
        layer_factory = LayerFactory(self)
        layer_factory.new_feed(name='data', layer_shape=(None, 24, 24, 3))
        layer_factory.new_conv(name='conv1', kernel_size=(3, 3),
channels_output=28,
        stride_size=(1, 1),
        padding='VALID', relu=False)
        layer_factory.new_prelu(name='prelu1')
        layer_factory.new_max_pool(name='pool1', kernel_size=(3, 3),
stride_size=(2,2))
        layer_factory.new_conv(name='conv2', kernel_size=(3, 3),
channels_output=48, stride_size=(1, 1),
        padding='VALID', relu=False)
        layer_factory.new_prelu(name='prelu2')
        layer_factory.new_max_pool(name='pool2', kernel_size=(3, 3),
stride_size=(2,2),padding='VALID')
        layer_factory.new_conv(name='conv3', kernel_size=(2, 2),
channels_output=64, stride_size=(1, 1),
        padding='VALID', relu=False)
        layer_factory.new_prelu(name='prelu3')
        layer_factory.new_fully_connected(name='fc1', output_count=128,
relu=False)
        layer_factory.new_prelu(name='prelu4')
        layer_factory.new_fully_connected(name='fc2-1', output_count=2,
relu=False)
        layer_factory.new_softmax(name='prob1', axis=1)
        layer_factory.new_fully_connected(name='fc2-2', output_count=4,
relu=False,
        input_layer_name='prelu4')
  
```

c. Mạng O-Net

Cuối cùng, O-Net lấy các đầu ra của R-Net làm đầu vào và đưa ra 3 bộ dữ liệu: xác suất của một mặt nằm trong hộp, tọa độ của hộp giới hạn và tọa độ của các mốc mặt (vị trí của mắt, mũi và miệng).



Hình 2.19: O-Net

Cấu trúc O-Net trong MTCNN:

```

class ONet(Network):
    def _config(self):
        layer_factory = LayerFactory(self)
        layer_factory.new_feed(name='data', layer_shape=(None, 48, 48,
3))

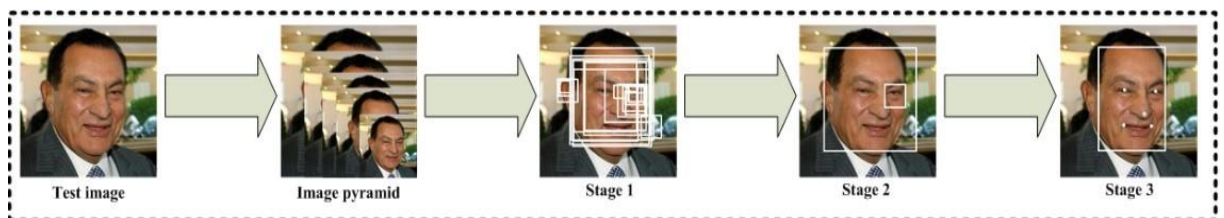
        layer_factory.new_conv(name='conv1', kernel_size=(3, 3),
channels_output=32, stride_size=(1, 1),
padding='VALID', relu=False)
        layer_factory.new_prelu(name='prelu1')
        layer_factory.new_max_pool(name='pool1', kernel_size=(3, 3),
stride_size=(2, 2))
        layer_factory.new_conv(name='conv2', kernel_size=(3, 3),
channels_output=64, stride_size=(1, 1),
padding='VALID', relu=False)
        layer_factory.new_prelu(name='prelu2')
        layer_factory.new_max_pool(name='pool2', kernel_size=(3, 3),
stride_size=(2, 2), padding='VALID')
        layer_factory.new_conv(name='conv3', kernel_size=(3, 3),
channels_output=64, stride_size=(1, 1),
padding='VALID', relu=False)
        layer_factory.new_prelu(name='prelu3')
        layer_factory.new_max_pool(name='pool3', kernel_size=(2, 2),
stride_size=(2, 2))
        layer_factory.new_conv(name='conv4', kernel_size=(2, 2),
channels_output=128, stride_size=(1, 1),

```

```
padding='VALID', relu=False)
layer_factory.new_prelu(name='prelu4')
layer_factory.new_fully_connected(name='fc1', output_count=256,
relu=False)
layer_factory.new_prelu(name='prelu5')
layer_factory.new_fully_connected(name='fc2-1', output_count=2,
relu=False)
layer_factory.new_softmax(name='prob1', axis=1)
layer_factory.new_fully_connected(name='fc2-2', output_count=4,
relu=False,
input_layer_name='prelu5')
layer_factory.new_fully_connected(name='fc2-3', output_count=10,
relu=False, input_layer_name='prelu5')
```

Tóm lại:

- P-Net: Proposal Network, dự đoán các vùng trong bức ảnh có thể là khuôn mặt (trong đó vẫn còn nhiều vùng không phải khuôn mặt).
- R-Net: Refine Network, sử dụng đầu ra của P-Net để loại bỏ các vùng không phải là khuôn mặt.
- O-Net: Output Network, sử dụng đầu ra của R-Net để đưa ra kết quả cuối cùng là vị trí khuôn mặt và các điểm mắt, mũi, miệng (facial landmark)



Hình 2.20: Ví dụ MTCNN

2.7.3. Lý do lựa chọn MTCNN để detect khuôn mặt

MTCNN phát hiện được khuôn mặt khá nhanh, chỉ đứng sau Haar Cascade và Histograms of Oriented Gradients (HOG). Tuy nhiên, nếu Haar Cascade chỉ hoạt động tốt với frontal face và dễ bị ảnh hưởng bởi ánh sáng môi trường, HOG thì không thể hoạt động tốt khi bị che lấp thì MTCNN hoạt động tốt ngay cả trường hợp mặt bị thiếu và che lấp nhiều và MTCNN rất ít bị ảnh hưởng bởi ánh sáng môi trường bên ngoài.

2.8. Tìm hiểu về mô hình ResNet

2.8.1. Giới thiệu về mô hình ResNet

ResNet (viết tắt của Residual Network), là mạng Học sâu nhận được quan tâm từ những năm 2015 sau cuộc thi LSVRC2015 và trở nên phổ biến trong lĩnh vực thị giác máy. ResNet đã được train với 18, 34, 50, 101, 152 lớp.

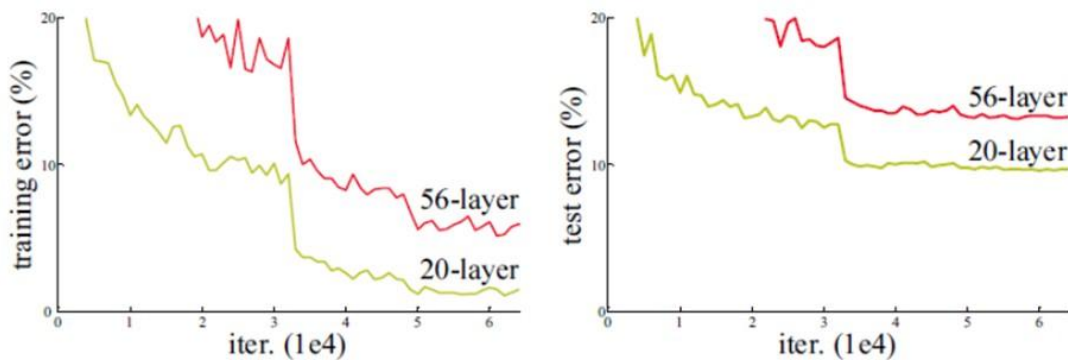
Nhờ khả năng biểu diễn mạnh mẽ của ResNet, hiệu suất của nhiều ứng dụng thị giác máy, không chỉ các ứng dụng phân loại hình ảnh được tăng cường. Một số ví dụ có thể kể đến là các ứng dụng phát hiện đồ vật và nhận dạng khuôn mặt.

Theo định lý gần đúng phổ quát, về mặt kiến trúc, một mạng nơ ron truyền thẳng có khả năng xấp xỉ mọi hàm với dữ liệu huấn luyện được cung cấp, miễn là không vượt quá sức chứa của nó. Tuy nhiên, xấp xỉ tốt dữ liệu không phải là mục tiêu duy nhất, chúng ta cần một mô hình có khả năng tổng quát hóa dữ liệu. Đó là lý do các kiến trúc sâu trở thành xu hướng của cộng đồng nghiên cứu.

2.8.2. Điểm nổi bật của mô hình ResNet

Resnet giải quyết được vấn đề của Học sâu truyền thống, nó có thể dễ dàng training model với hàng trăm layer. Để hiểu ResNet chúng ta cần hiểu vấn đề khi stack nhiều layer khi training, vấn đề đầu tiên khi tăng model deeper hơn gradient sẽ bị vanishing/explodes. Vấn đề này có thể giải quyết bằng cách thêm Batch Normalization nó giúp normalize output giúp các hệ số trở nên cân bằng hơn không quá nhỏ hoặc quá lớn nên sẽ giúp model dễ hội tụ hơn.

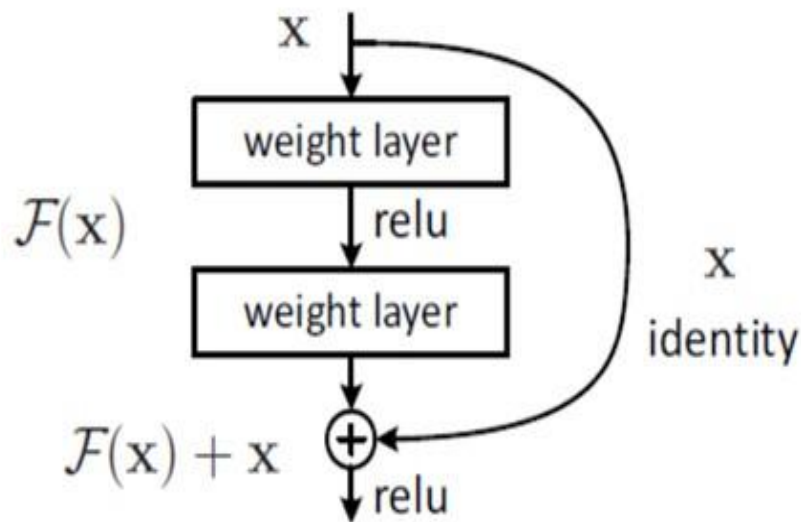
Vấn đề thứ hai là degradation. Khi model deeper accuracy bắt đầu bão hòa (saturated) thậm chí là giảm. Như hình vẽ bên dưới khi stack nhiều layer hơn thì training error lại cao hơn ít layer như vậy vấn đề không phải là do overfitting. Vấn đề này là do model không dễ training khó học hơn, thử tưởng tượng một training một shallow model, sau đó chúng ta stack thêm nhiều layer, các layer sau khi thêm vào sẽ không học thêm được gì cả (identity mapping) nên accuracy sẽ tương tự như shallow model mà không tăng. Resnet được ra đời để giải quyết vấn đề degradation này.



Hình 2.21: So sánh độ chính xác

2.8.3. Kiến trúc ResNet

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



Hình 2.22: Một khối xây dựng của ResNet

2.8.4. Mô hình ResNet

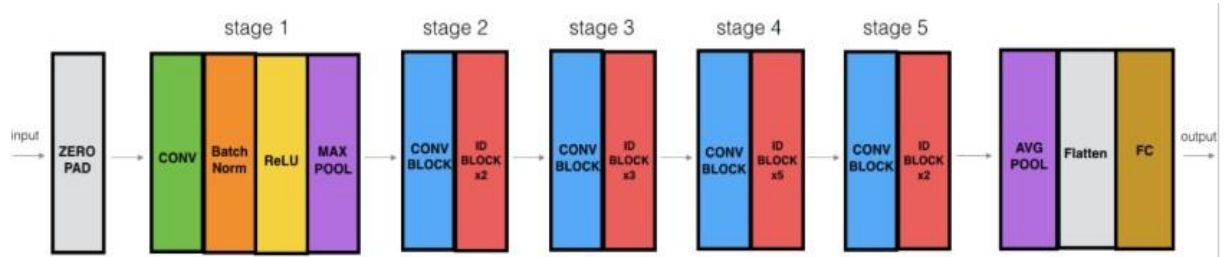
Luận văn sử dụng ResNet-100 để phân lớp dữ liệu. ResNet-100 chính là mô hình ResNet-101 sau khi bỏ đi lớp cuối cùng để thu được đầu ra là vector với

512 chiều. Hình 2.23 dưới đây là kiến trúc chi tiết của mô hình ResNet trong đó có ResNet-101.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Hình 2.23: Kiến trúc chi tiết của ResNet

Mô hình chi tiết ResNet-101.



Hình 2.24: Mô hình ResNet-101

Đầu tiên, đệm đầu vào với một miếng đệm 3×3

***Giai đoạn một:**

Hai lớp ResNet đầu tiên giống như hai lớp của GoogLeNet: layer Convolution 7×7 với 64 đầu ra và bước nhảy stride = 2, theo sau đó là layer max pooling 3×3 bước nhảy stride = 2. Điểm khác biệt là lớp chuẩn hóa hàng loạt được thêm vào sau mỗi lớp chập trong ResNet.

```
net = nn.Sequential()
net.add(nn.Conv2D(64, kernel_size=7, strides=2, padding=3),
        nn.BatchNorm(), nn.Activation('relu'),
        nn.MaxPool2D(pool_size=3, strides=2, padding=1))
```

Hình 2.25: Code ResNet Model

GoogLeNet sử dụng bốn khối được tạo thành từ các khối Inception. Tuy nhiên, ResNet sử dụng bốn mô-đun được tạo thành từ các residual blocks, mỗi khối sử dụng một số khối còn lại có cùng số kênh đầu ra. Số lượng kênh trong mô-đun đầu tiên giống với số lượng kênh đầu vào. Vì một layer max pooling với bước nhảy là 2 đã được sử dụng, nên không cần thiết phải giảm kích cỡ (width, height). Trong residual blocks đầu tiên cho mỗi mô-đun tiếp theo, số lượng kênh được nhân đôi so với mô-đun trước đó và chiều cao và chiều rộng được giảm một nửa. Lưu ý rằng xử lý đặc biệt đã được thực hiện trên mô-đun đầu tiên.

```
def resnet_block(num_channels, num_residuals, first_block=False):
    blk = nn.Sequential()
    for i in range(num_residuals):
        if i == 0 and not first_block:
            blk.add(Residual(num_channels, use_1x1conv=True, strides=2))
        else:
            blk.add(Residual(num_channels))
    return blk
```

Hình 2.26: Code ResNet Model

Kích thước đầu ra của giai đoạn này là 56x56.

***Giai đoạn hai:**

Giai đoạn hai gồm có 3 khối trong đó có 1 khối Convolution và 2 khối nhận dạng.

Khối Convolution sử dụng 3 bộ lọc có kích thước 64x64x256 với $f=3$ và bước nhảy stride $s=2$.

Tương tự, 2 khối nhận dạng cũng sử dụng 3 bộ lọc có kích thước 64x64x256 với $f=3$.

Kích thước đầu ra của giai đoạn này là 56x56.

***Giai đoạn ba:**

Giai đoạn ba gồm có 4 khối trong đó có 1 khối Convolution và 3 khối nhận dạng.

Khối Convolution sử dụng 3 bộ lọc có kích thước $128 \times 128 \times 512$ với $f = 3$ và bước nhảy stride $s = 2$.

Tương tự, 3 khối nhận dạng cũng sử dụng 3 bộ lọc có kích thước $128 \times 128 \times 512$ với $f = 3$.

Kích thước đầu ra của giai đoạn này là 28×28 .

***Giai đoạn bốn:**

Giai đoạn bốn gồm có 23 khối trong đó có 1 khối Convolution và 22 khối nhận dạng.

Khối Convolution sử dụng 3 bộ lọc có kích thước $256 \times 256 \times 1024$ với $f = 3$ và bước nhảy stride $s = 2$.

Tương tự, 22 khối nhận dạng cũng sử dụng 3 bộ lọc có kích thước $256 \times 256 \times 1024$ với $f = 3$.

Kích thước đầu ra của giai đoạn này là 14×14 .

***Giai đoạn năm:**

Giai đoạn năm gồm có 3 khối trong đó có 1 khối Convolution và 2 khối nhận dạng.

Khối Convolution sử dụng 3 bộ lọc có kích thước $512 \times 512 \times 2048$ với $f = 3$ và bước nhảy stride $s = 2$.

Tương tự, 2 khối nhận dạng cũng sử dụng 3 bộ lọc có kích thước $512 \times 512 \times 2048$ với $f = 3$.

Kích thước đầu ra của giai đoạn này là 7×7 .

Cuối cùng, giống như GoogLeNet, ResNet đã thêm một layer pooling trung bình toàn cầu, theo sau là đầu ra layer full connected.

Mặc dù kiến trúc chính của ResNet tương tự như của GoogLeNet nhưng cấu trúc của ResNet đơn giản và dễ sửa đổi hơn. Tất cả các yếu tố này đã dẫn đến việc sử dụng ResNet nhanh chóng và rộng rãi.

Trước khi đào tạo ResNet, hãy quan sát cách hình dạng đầu vào thay đổi giữa các mô-đun khác nhau trong ResNet. Như trong tất cả các kiến trúc trước đây, độ phân giải giảm trong khi số lượng kênh tăng lên cho đến khi một layer pooling trung bình toàn cầu tổng hợp tất cả các features.

2.9. Kết luận chương

Trong chương 2, luận văn đã trình bày 2 nội dung chính sau:

- + Khái quát về học máy, đặc biệt là Học sâu. Qua đó luận văn cũng đã trình bày sơ lược về một số mô hình mạng Học sâu cũng như ứng dụng của chúng.
- + Trình bày tương đối chi tiết về mô hình, hoạt động của mạng nơron tích chập CNN cũng như cách xây dựng nó. Bên cạnh đó, luận văn cũng đã trình bày về hai mô hình CNN chính sẽ sử dụng để xây dựng hệ thống nhận dạng khuôn mặt là MTCNN và ResNet.

Trong chương tiếp theo, luận văn sẽ trình bày chi tiết quá trình xây dựng hệ thống nhận dạng khuôn mặt

Chương 3. NHẬN DẠNG KHUÔN MẶT ỨNG DỤNG CHO BÀI TOÁN ĐIỂM DANH TỰ ĐỘNG

Nội dung chương này tập trung vào xây dựng mô hình nhận dạng khuôn mặt, phương pháp huấn luyện và đánh giá mô hình. Bên cạnh đó chương cũng sẽ giới thiệu về các công nghệ được sử dụng, phương pháp xây dựng bộ dữ liệu huấn luyện. Cuối cùng là trình bày về nghiên cứu và thiết kế phần cứng cho thiết bị điểm danh và cách tối ưu luồng xử lý.

3.1. Xây dựng hệ thống nhận dạng khuôn mặt

3.1.1. Công nghệ sử dụng

Để xây dựng hệ thống nhận dạng khuôn mặt, trong phạm vi luận văn này tôi sử dụng ngôn ngữ lập trình python và hệ quản trị cơ sở dữ liệu SQLite để lưu trữ dữ liệu khuôn mặt và thông tin gắn với từng khuôn mặt.

- **MXNet**

Để implement các mô hình CNN trong việc phân tách khuôn mặt (face detection) và tiến hành việc nhận dạng (face recognition) tôi sử dụng framework MXNet dựa trên ngôn ngữ python. MXNet là một framework mã nguồn mở được sử dụng để huấn luyện, triển khai các mô hình mạng nơ-ron học sâu. Ngoài ra nó có khả năng mở rộng, cho phép huấn luyện mô hình nhanh chóng và linh hoạt, hỗ trợ huấn luyện sử dụng GPU và có thể sử dụng nhiều ngôn ngữ (bao gồm C++, Python, Java, Julia, Matlab, JavaScript, Go, R, Scala, Perl, và Wolfram). Vì vậy việc sử dụng MXNet giúp việc triển khai mô hình dễ dàng hơn trên các nền tảng khác nhau. Dẫn chứng cho việc này đó là nền tảng Amazon Webservice lựa chọn MXNet làm framework để triển khai mô hình.

- **SQLite**

Để lưu trữ dữ liệu khuôn mặt của học sinh và thông tin của học sinh tôi sử dụng hệ quản trị cơ sở dữ liệu SQLite. Đây là hệ quản trị cơ sở dữ liệu nhỏ gọn, không cần máy chủ riêng biệt phức tạp, có thể chạy như một tiến trình độc lập và truy cập trực tiếp dữ liệu dựa trên các file lưu trữ. SQLite là một tập hợp các thành

phần khép kín, được đóng gói và không phụ thuộc vào các thành phần bên ngoài, điều này giúp cho hệ quản trị cơ sở dữ liệu này dễ dàng triển khai trong các ứng dụng cần lưu trữ dữ liệu và truy xuất thông tin nhưng không yêu cầu các tính năng quá sâu và thừa của một hệ quản trị cơ sở dữ liệu phức tạp. Vì vậy với việc hỗ trợ các chức năng cơ bản có thể đáp ứng được đầy đủ các yêu cầu đặt ra khi xây dựng hệ thống điểm danh.

Trước khi sử dụng hệ quản trị cơ sở dữ liệu SQLite tôi đã thử nghiệm lưu trữ dữ liệu trên tệp và thực hiện thao tác với dữ liệu bằng các tác vụ đọc/ghi file nhưng tốc độ các truy vấn dữ liệu quá chậm dẫn đến thời gian điểm danh lâu và trải nghiệm người dùng không tốt. Ngoài ra khi sử dụng file để lưu trữ dữ liệu rất khó để tổ chức được cấu trúc dữ liệu hợp lý cho ứng dụng. SQLite lưu trữ các dữ liệu có cấu trúc, hỗ trợ việc truy vấn dựa trên chỉ mục nên tốc độ truy vấn nhanh khi cơ sở dữ liệu lớn.

- **Thuật toán k-NN (K-Nearest Neighbors)**

Đầu ra của mô hình nhận dạng khuôn mặt sử dụng mạng Resnet sau khi bỏ tầng Softmax activation là một vector 128 chiều, nghĩa là mỗi khuôn mặt sẽ được trích xuất các đặc điểm và tổng hợp trong vector này. Đối với cơ sở dữ liệu khuôn mặt của học sinh, mỗi học sinh sẽ có 3-4 ảnh chụp khuôn mặt của mình trong hệ thống. Vấn đề đặt ra là sử dụng thuật toán nào để so khớp khi đưa một ảnh chụp khuôn mặt bất kỳ vào phần mềm và xác định được đó là khuôn mặt của ai. Để giải quyết vấn đề này, tôi nhận thấy thuật toán k-NN khá hiệu quả và đảm bảo được yêu cầu đề ra.

K-NN thực chất là thuật toán để phân lớp đối tượng dựa vào khoảng cách gần nhất giữa các đối tượng cần xếp lớp. Tập dữ liệu huấn luyện là các điểm dữ liệu đã được gán nhãn, khi đưa vào một điểm dữ liệu cần phân lớp, thuật toán sẽ tính độ giống giữa điểm dữ liệu này với các điểm dữ liệu trong bộ dữ liệu huấn luyện, sau đó lấy trung bình độ giống với lớp dữ liệu tương ứng. Giá trị trung bình này lớn nghĩa là điểm dữ liệu này gần với các điểm dữ liệu trong lớp đó, khả năng thuộc vào lớp đó sẽ cao. Và giá trị trung bình lớn nhất nghĩa là điểm dữ liệu đó có khả năng

thuộc vào lớp đó nhất. Tuy nhiên ta cũng cần phải xác định một ngưỡng tối thiểu (threshold) để đảm bảo một điểm dữ liệu có thể không thuộc vào lớp nào nếu nằm dưới ngưỡng tối thiểu đó.

Áp dụng k-NN vào trong bài toán này, mỗi học sinh trong cơ sở dữ liệu sẽ có 4 ảnh và mỗi học sinh sẽ được coi là một lớp dữ liệu, mỗi ảnh trong lớp dữ liệu sẽ được coi là một điểm dữ liệu. Như vậy mỗi lớp dữ liệu gồm tối đa 4 điểm dữ liệu. Lặp qua từng học sinh, ta sẽ tính độ giống vector đặc điểm của ảnh học sinh được đưa vào với vector đặc điểm của từng khuôn mặt của học sinh trong cơ sở dữ liệu, sau đó lấy giá trị trung bình [7]. Kết thúc lặp ta sẽ có giá trị các độ giống trung bình của ảnh đưa vào với khuôn mặt từng học sinh. Dựa vào giá trị này ta sẽ xác định được khuôn mặt đó là của học sinh nào.

$$d_k = \frac{1}{n} \sum_{i=1}^n \text{similar}(q, y_i) \quad (3-1)$$

Trong đó:

d_k : Là trung bình độ giống của vector q với các khuôn mặt trong lớp k (trường hợp này $n = 4$)

q : vector đặc điểm của khuôn mặt đưa vào hệ thống

y_i : Vector khuôn mặt thứ i của lớp k .

Để đo độ giống (similar) giữa 2 vector ở đây tôi sử dụng khoảng cách Euclid:

$$\text{similar}(q, y) = \sqrt{\sum_{j=1}^m (q_j - y_j)^2} \quad (3-2)$$

Viết gọn lại:

$$d_k = \frac{1}{n} \sum_{i=1}^n \left(\sqrt{\sum_{j=1}^m (q_j - y_j)^2} \right) \quad (3-3)$$

Để xác định được khuôn mặt đưa vào là của học sinh nào, ta tiến hành lấy giá trị d_k lớn nhất theo công thức sau:

$$c = \max_i(d_i) \mid i = (1, C), d_i \leq \text{threshold} \quad (3-4)$$

Để tránh trường hợp một người không có trong cơ sở dữ liệu nhưng sử dụng hệ thống điểm danh, ta không thể lấy giá trị trung bình độ giống lớn nhất để kết luận đó là ai vì sẽ dẫn đến việc nhận nhầm mà còn phải so sánh với giá trị ngưỡng (threshold) để đảm bảo giá trị về độ giống luôn phải nhỏ hơn ngưỡng đó.

- **Thiết kế giao diện thiết bị bằng QtDesigner :**

Qt là một Application framework đa nền tảng viết trên ngôn ngữ C++ , được dùng để phát triển các ứng dụng trên desktop, hệ thống nhúng và mobile. Hỗ trợ cho các platform bao gồm : Linux, OS X, Windows, VxWorks, QNX, Android, iOS, BlackBerry, Sailfish OS và một số platform khác. PyQt là Python interface của Qt, kết hợp của ngôn ngữ lập trình Python và thư viện Qt, là một thư viện bao gồm các thành phần giao diện điều khiển (widgets , graphical control elements). PyQt API bao gồm các module bao gồm số lượng lớn với các classes và functions hỗ trợ cho việc thiết kế ra các giao diện giao tiếp với người dùng của các phần mềm chức năng. Hỗ trợ với Python 2.x và 3.x. PyQt được phát triển bởi Riverbank Computing Limited, version mới nhất của PyQt có thể download tại đường link : [PyQt Riverbank Computing Limited](#) Các class của PyQt5 được chia thành các module, bao gồm :

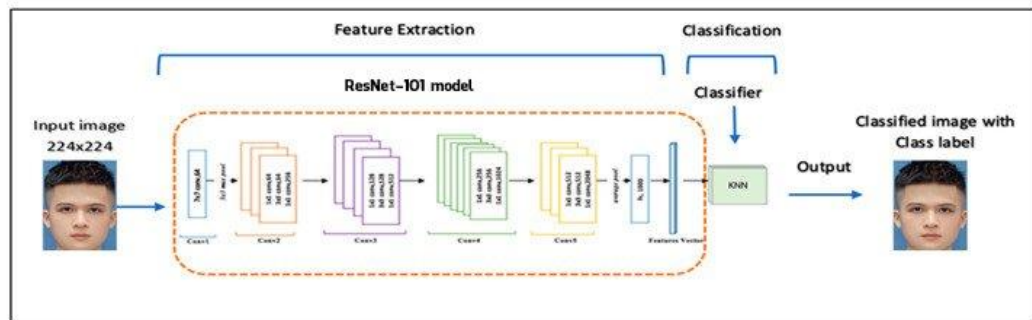
- **QtCore:** là module bao gồm phần lõi không thuộc chức năng GUI, ví dụ dùng để làm việc với thời gian, file và thư mục, các loại dữ liệu, streams, URLs, mime type, threads hoặc processes.
- **QtGui:** bao gồm các class dùng cho việc lập trình giao diện (windowing system integration), event handling, 2D graphics, basic imaging, fonts và text.
- **QtWidgets:** bao gồm các class cho widget, ví dụ : button, hộp thoại, ... được sử dụng để tạo nên giao diện người dùng cơ bản nhất.
- **QtMultimedia:** thư viện cho việc sử dụng âm thanh, hình ảnh, camera,...
- **QtBluetooth:** bao gồm các class giúp tìm kiếm và kết nối với các thiết bị có giao tiếp với phần mềm.
- **QtNetwork:** bao gồm các class dùng cho việc lập trình mạng, hỗ trợ lập trình TCP/IP và UDP client , server hỗ trợ việc lập trình mạng.

- **QtPositioning**: bao gồm các class giúp việc hỗ trợ xác định vị.
- **Enginio**: module giúp các client truy cập các Cloud Services của Qt.
- **QtWebSockets**: cung cấp các công cụ cho WebSocket protocol.
- **QtWebKit**: cung cấp các class dùng cho làm việc với các trình duyệt Web, dựa trên thư viện WebKit2.

- **QtWebKitWidgets** các widget cho WebKit.
- **QtXml**: các class dùng cho làm việc với XML file.
- **QtSvg**: dùng cho hiển thị các thành phần của SVG file.
- **QtSql**: cung cấp các class dùng cho việc làm việc với dữ liệu.
- **QtTest**: cung cấp các công cụ cho phép test các đơn vị của ứng dụng với PyQt5.

3.1.2. Xây dựng hệ thống nhận dạng khuôn mặt

Luận văn sử dụng mô hình ResNet (đã tìm hiểu ở chương 2) để trích xuất đặc trưng khuôn mặt. Sau đây tôi sẽ trình bày về cách áp dụng mô hình Resnet- 101 để thực hiện việc nhận dạng khuôn mặt:



Hình 3.1: Các bước thực hiện nhận dạng khuôn mặt sử dụng Resnet-101

Camera sẽ chụp ảnh học sinh sau đó ảnh sẽ được đưa qua mạng MTCNN để mạng này phát hiện vị trí và trích xuất khuôn mặt. Một ảnh đầu vào có thể có nhiều khuôn mặt. Ảnh khuôn mặt sẽ được chỉnh về kích thước chuẩn là 224 x 224px để giảm số chiều vector. Ảnh lúc này trở thành 1 ma trận 224 x 224 x 3 [1].

Sau đó tiếp tục đưa ma trận này vào mạng Resnet để trích xuất đặc điểm khuôn mặt. Sau khi qua mô hình ta sẽ nhận được một features vector 128 chiều. Với vector 128

chiều này ta sẽ sử dụng để làm căn cứ để xác định độ tương đồng giữa 2 khuôn mặt và đưa vào thuật toán phân lớp KNN để gán nhãn cho khuôn mặt [9].

Để xây dựng hệ thống này, tôi đã xây dựng các thành phần của hệ thống sau:

- Lựa chọn phần cứng để xây dựng bộ thiết bị điểm danh tại chỗ và tiến hành lập trình nhúng cho thiết bị.
- Xây dựng web service để lưu trữ cơ sở dữ liệu hệ thống và thông tin về việc điểm danh
- Xây dựng phần mềm để ghi nhận, quản lý, lưu trữ và xuất kết quả điểm danh

3.1.3. Xây dựng dữ liệu huấn luyện

Dữ liệu huấn luyện được chia ra làm 2 loại: Dữ liệu thu thập từ internet và dữ liệu tự xây dựng.

Trong phạm vi luận văn này tôi sử dụng các bộ dữ liệu sau đây:

ORL: Cơ sở dữ liệu được sử dụng trong các thử nghiệm nhận dạng. Nó chứa 10 hình ảnh của 40 cá nhân, thêm vào tổng số 400 hình ảnh có các góc mặt, nét mặt và các chi tiết trên khuôn mặt khác nhau. Bộ dữ liệu được thu thập tại Phòng thí nghiệm Nghiên cứu Olivetti tại Đại học Cambridge cho một số cá nhân.

• **Cơ sở dữ liệu khuôn mặt GTAV:** Cơ sở dữ liệu chứa hình ảnh của 44 cá nhân, được chụp ở các chế độ xem tư thế khác nhau (0° , $\pm 30^\circ$, $\pm 45^\circ$, $\pm 60^\circ$ và 90°) cho ba lần chiếu sáng (môi trường hoặc ánh sáng tự nhiên, nguồn sáng mạnh từ một góc 45° và nguồn sáng gần như trực diện trung bình mạnh với môi trường hoặc ánh sáng tự nhiên). Trong nghiên cứu của chúng tôi, 34 hình ảnh cho mỗi người trong tập dữ liệu đã được chọn.

• **Cơ sở dữ liệu khuôn mặt Georgia Tech:** Cơ sở dữ liệu này chứa các bộ hình ảnh cho 50 cá nhân và có 15 hình ảnh màu cho mỗi người. Hầu hết các bức ảnh được chụp trong hai phiên khác nhau để xem xét sự khác nhau về điều kiện ánh sáng, ngoại hình và nét mặt. Ngoài ra, các hình ảnh trong bộ dữ liệu được chụp ở các hướng và tỷ lệ khác nhau.

Face FEI: Cơ sở dữ liệu có 14 bộ ảnh cho mỗi cá nhân trong số 200 người, tổng cộng lên đến 2800 ảnh. Trong nghiên cứu của chúng tôi, chúng tôi đã chọn hình ảnh

trực diện cho mỗi cá nhân. Tổng số hình ảnh được chọn trong nghiên cứu là 400 hình ảnh. Trong thử nghiệm của mình, chúng tôi chọn hình ảnh cho 50 cá nhân trong tổng số 700 hình ảnh.

- **Các khuôn mặt được gắn nhãn trong tự nhiên (LFW):** Bộ dữ liệu này được thiết kế để nghiên cứu vấn đề nhận dạng khuôn mặt không bị giới hạn. Bộ dữ liệu chứa hơn 13.000 hình ảnh về khuôn mặt được thu thập từ web. Mỗi khuôn mặt đã được dán nhãn với tên của người trong hình. Tổng số 1680 người trong số những người được chụp có hai hoặc nhiều bức ảnh khác biệt trong tập dữ liệu.

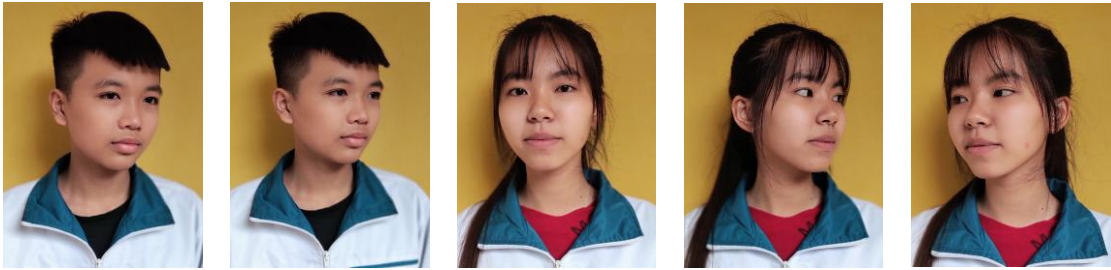
- **Các khuôn mặt được dán nhãn được chuẩn hóa chính diện trong tự nhiên (F_LFW):** Tập dữ liệu này chứa phiên bản được chuẩn hóa của các hình ảnh được thu thập trong tập dữ liệu LFW. Bộ dữ liệu được thiết kế để nghiên cứu tính năng nhận dạng khuôn mặt không bị giới hạn. Nó đã được tạo ra trong nghiên cứu.

Bên cạnh bộ dữ liệu sẵn có, tôi cũng xây dựng bộ dữ liệu thêm khuôn mặt học sinh để tăng độ chính xác mô hình và phù hợp với mục tiêu bài toán. Bộ dữ liệu này được xây dựng bằng cách thu thập ảnh chụp học sinh từ điện thoại với nhiều góc chụp.

- **Dữ liệu tự xây dựng (hình 3.2):** Dữ liệu tự xây dựng gồm 300 bức ảnh chụp học sinh khối 11 trường THPT Thanh Oai B với các góc chụp khác nhau như thẳng, nghiêng ($\pm 30^\circ$, $\pm 45^\circ$, $\pm 60^\circ$)







Hình 3.2: Bộ dữ liệu xây dựng

3.1.4. Huấn luyện mô hình nhận dạng khuôn mặt

a) Hàm mất mát

Để phân tích sự giống nhau giữa hai hình ảnh, chúng ta cần biến đổi hình ảnh đầu vào của mình thành một hình biểu diễn nhỏ hơn, chẳng hạn như một vector duy nhất. Biểu diễn này thường được gọi là embedding. Chúng ta cần xây dựng các embedding vector để chúng có các thuộc tính sau:

Hai hình ảnh giống nhau tạo ra hai embedding vector và khoảng cách toán học giữa chúng là nhỏ.









Hai hình ảnh rất khác nhau tạo ra hai embedding vector và khoảng cách toán học giữa chúng lớn.

Để làm được điều đó, chúng ta cần huấn luyện một mạng nơ-ron để tạo ra các embedding vector tốt chứa các thuộc tính này. Để so sánh khuôn mặt, 'hai hình ảnh giống nhau' mà tôi tham chiếu ở trên có thể là cùng một khuôn mặt trong hai ảnh khác nhau (tạo ra các embedding vector có khoảng cách gần) và 'hai hình ảnh rất khác nhau' có thể là hai khuôn mặt khác nhau, tạo ra các embedding vector khoảng cách lớn.

Như vậy số chiều của vector nhúng là bao nhiêu thì đủ? Ý tưởng chính của việc encode hình ảnh khuôn mặt là vector đầu ra phải có khả năng biểu diễn các thuộc tính đủ để phân biệt khuôn mặt này với khuôn mặt khác. Để giải quyết vấn đề này, chiều dài của vector được coi là một tham số cần tìm. Với mạng Resnet-101, tôi lựa chọn 128 số để biểu diễn các thuộc tính của khuôn mặt tương ứng với vector đầu ra là 128 chiều.

Có hai cách chính để tìm ra các tham số cho mạng CNN:

Thứ nhất, chúng ta có thể đơn giản coi phần bên phải của hệ thống như một bộ phân loại nhị phân với $Y = 1$ nếu các ảnh đầu vào thuộc cùng một lớp và $Y = 0$ nếu ảnh đầu vào không thuộc cùng một lớp.

X		Y
		1
		0
		1
		1

Hình 3.3: Mô tả phương pháp tính độ lỗi

Cách thứ 2, vì chúng ta muốn so sánh 2 hình ảnh và có khoảng cách giữa 2 embedding vector là nhỏ nếu 2 bức ảnh giống nhau và khoảng cách là lớn nếu 2 bức ảnh khác nhau [10]. Như vậy:

- Một bức ảnh đầu tiên, được gọi là Anchor (neo)

- Một hình ảnh từ cùng lớp với Anchor, được gọi là Positive
- Một hình ảnh từ một lớp khác với Anchor, được gọi là Negative



Hình 3.4: Mô tả phương pháp tính độ lỗi dựa trên điểm neo

Với bộ ba của ba hình ảnh này, (gọi các vector embedding của các ảnh này là A, P và N), như vậy mô hình cần phải đưa ra:

$$\text{Khoảng cách}(A, P) < \text{khoảng cách}(A, N)$$

Hay viết lại theo cách khác:

$$\text{Khoảng cách}(A, P) - \text{khoảng cách}(A, N) < 0$$

Để tránh trường hợp khoảng cách = 0 ta thêm tham số margin vào như sau:

$$\text{distance}(A, P) - \text{distance}(A, N) + \text{margin} < 0$$

Như vậy hàm mất mát sẽ có dạng [5]:

$$L = \max(d(A, P) - d(A, N) + \text{margin}, 0)$$

b) Phân chia tập dữ liệu huấn luyện

Để huấn luyện mô hình nhận dạng khuôn mặt, tôi chia tập dữ liệu huấn luyện thành 2 phần: 70% tập dữ liệu dùng để huấn luyện và 30% tập dữ liệu để kiểm thử mô hình.

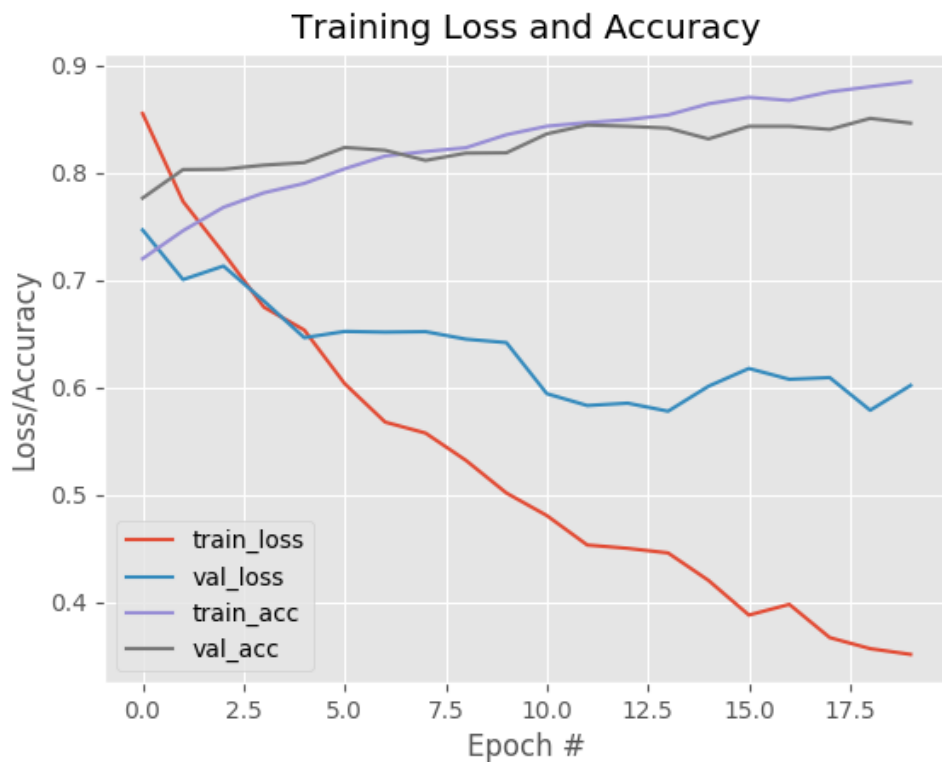
c) Kết quả huấn luyện

Mô hình được huấn luyện bằng dịch vụ Google Colab có sử dụng TPU. Kết quả huấn luyện được mô tả bằng biểu đồ. Tôi sử dụng phương pháp đánh giá bằng độ chính xác theo phần trăm [8].

Do mô hình sử dụng pretrained từ mô hình resnet nên bắt đầu quá trình huấn luyện, độ chính xác của mô hình bắt đầu ở 77% đối với tập validate (pretrained

được huấn luyện trên tập dữ liệu gồm 1.3 triệu khuôn mặt), sau đó tăng dần trong quá trình huấn luyện và đạt 85% ở gần 18 epochs. Với tập dữ liệu đã thu thập, mô hình được cải thiện đáng kể với độ chính xác tăng dần.

Loss đối với tập huấn luyện giảm dần từ 0.85 về 0.2. . Đối với tập validate, loss giảm ở 0.75 về 0.6 trong suốt quá trình huấn luyện



Hình 3.5: Biểu đồ mô tả kết quả huấn luyện

3.2. Lập trình nhúng cho thiết bị điểm danh

Để triển khai việc điểm danh bằng nhận dạng khuôn mặt cần phải tạo ra thiết bị đặt ở mỗi lớp học để thực hiện việc điểm danh. Thiết bị này có gắn camera để thực hiện nhận dạng học sinh và kết nối internet để gửi thông tin người điểm danh về hệ thống theo dõi để thống kê và báo cáo. Vì vậy yêu cầu thiết bị phải có tốc độ xử lý nhanh, chính xác.

Trong quá trình nghiên cứu tôi đã đưa ra một số giải pháp cùng với các ưu/nhược điểm của các giải pháp như sau:

- **Sử dụng máy tính PC kèm webcam:** Giải pháp này công kênh và chi phí cao (cả chi phí ban đầu và chi phí vận hành), trung bình mỗi lớp học sẽ cần một bộ thiết bị điểm danh sẽ khiến giải pháp không khả thi.

- **Sử dụng máy tính nhúng và camera tích hợp dạng module:** Máy tính nhúng là thiết bị nhỏ gọn, tiết kiệm năng lượng khi sử dụng và có khả năng xử lý tính toán như máy tính PC. Ngoài ra máy tính nhúng còn có thể tích hợp thêm màn hình thông qua kết nối dạng cáp CSI.

Như vậy với ưu điểm vượt trội của máy tính nhúng, tôi lựa chọn giải pháp này để thực hiện triển khai mô hình điểm danh học sinh.



Hình 3.6: Máy tính nhúng Raspberry pi cùng màn hình

3.2.1. Máy tính nhúng raspberry Pi 4:

Raspberry Pi là một máy tính rất nhỏ gọn, kích thước hai cạnh như bằng khoảng một cái thẻ ATM và chạy hệ điều hành Linux. Raspberry Pi được phát triển bởi Raspberry Pi Foundation - một tổ chức phi lợi nhuận.

Chúng ta có thể sử dụng Raspberry Pi như một máy vi tính bởi nhà sản xuất đã tích hợp mọi thứ cần thiết trong đó. Bộ xử lý SoC Broadcom BCM2835 của nó bao gồm CPU, GPU, RAM, khe cắm thẻ microSD, Wi-Fi, Bluetooth và 4 cổng USB 2.0.

Với Raspberry Pi, chỉ cần cài hệ điều hành, gắn chuột, bàn phím và màn hình là có thể sử dụng như một máy vi tính. Raspberry Pi không hoàn toàn có thể thay thế

được máy tính để bàn hoặc laptop nhưng nó là một thiết bị đa năng có thể được sử dụng cho những hệ thống điện tử, thiết lập hệ thống tính toán, những dự án DIY... với chi phí rẻ.



1. Nguồn cho Raspberry 5V-2A
2. Dây HDMI cabos 1.5m
3. Vỏ hộp Pi2/B+
4. Raspberry Pi Camera Board
5. Kết nối wifi cho Pi : Wireless USB EP-N8508GS
6. Màn hình cảm ứng 7 inch
7. Thẻ nhớ

Hình 3.7: Các thành phần cơ bản cần thiết cho thiết bị

Máy tính nhúng Raspberry Pi 4 hỗ trợ nhiều hệ điều hành, trong ứng dụng này tôi sử dụng hệ điều hành linux giúp triển khai các mô hình được dễ dàng:

3.2.2. Cài đặt hệ điều hành

Để cài đặt hệ điều hành, cần tải về file IMG chứa hệ điều hành cần cài đặt. File này được cung cấp sẵn

Các bước cài đặt hệ điều hành cho Raspberry Pi:

Bước 1: Chèn thẻ MicroSD của bạn vào đầu đọc thẻ ở máy tính và kiểm tra tên ổ được gán cho thẻ nhớ (ví dụ ổ H:), tránh nhầm ổ dẫn đến mất dữ liệu vì phần mềm sẽ format thẻ nhớ.

Bước 2: Mở phần mềm Win32DiskImager, phần mềm này chỉ cần download về rồi chạy mà không cần cài đặt.

Bước 3: Lựa chọn file hệ điều hành vừa tải về. (Lưu ý, hệ điều hành cần phải ở định dạng .img. Thông thường, hệ điều hành của RPI được nén dưới dạng .zip hoặc

.tar.gz,... Khi tải về cần giải nén nó ra để có file hệ điều hành dạng .img). Sau đó lựa chọn ổ thẻ nhớ cần ghi.

Sau khi cài đặt xong hệ điều hành cho Raspberry Pi, cần cài đặt một số môi trường để cấu hình cho mô hình:

Cài đặt môi trường: Raspberry sẽ cần cài đặt môi trường và một số thư viện sau:

- Python 3
- OpenCV
- Tensorflow

3.2.3. Xây dựng giao diện cho thiết bị

Như đã đề cập trong phần giới thiệu công nghệ sử dụng, để xây dựng giao diện cho thiết bị điểm danh tôi đã sử dụng PyQt để lập trình đồ họa dựa trên ngôn ngữ lập trình python.

Sau đây là hình ảnh của máy điểm danh mà tôi đã xây dựng:

Code chương trình:

```
class Ui_Form(QMainWindow):
    def setupUi(self, Form):
        Form.setObjectName("Form")
        Form.resize(793, 455)
        Form.setMaximumSize(QtCore.QSize(800, 480))
        self.graphicsView = ImageWidget(Form)
        self.graphicsView.setGeometry(QtCore.QRect(10, 50,
1561, 331))
        self.graphicsView.setObjectName("graphicsView")
        self.graphicsView.setAutoFillBackground(True)
        self.dateEdit = QtWidgets.QDateEdit(Form)
        self.dateEdit.setGeometry(QtCore.QRect(640, 20,
141, 24))
        self.dateEdit.setObjectName("dateEdit")
        self.label = QtWidgets.QLabel(Form)
        self.label.setGeometry(QtCore.QRect(540, 20, 111,
20))
        self.label.setObjectName("label")
        self.pushButton = QtWidgets.QPushButton(Form)
        self.pushButton.setGeometry(QtCore.QRect(408, 390,
161, 51))
        icon = QtGui.QIcon()
        icon.addPixmap(QtGui.QPixmap("camera-
1724286_960_720.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)
```

```

        self.pushButton.setIcon(icon)
        self.pushButton.setObjectName("pushButton")
        self.textBrowser = QtWidgets.QTextBrowser(Form)
        self.textBrowser.setGeometry(QtCore.QRect(580, 280,
211, 111))
        self.textBrowser.setObjectName("textBrowser")
        self.pushButton_2 = QtWidgets.QPushButton(Form)
        self.pushButton_2.setGeometry(QtCore.QRect(10, 390,
181, 51))
        icon1 = QtGui.QIcon()
        icon1.addPixmap(QtGui.QPixmap("Download-Icon-
2.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)
        self.pushButton_2.setIcon(icon1)
        self.pushButton_2.setObjectName("pushButton_2")
        self.progressBar = QtWidgets.QProgressBar(Form)
        self.progressBar.setGeometry(QtCore.QRect(670, 420,
118, 20))
        self.progressBar.setProperty("value", 24)
        self.progressBar.setObjectName("progressBar")
        self.listView = QtWidgets.QListView(Form)
        self.listView.setGeometry(QtCore.QRect(580, 50,
211, 221))
        self.listView.setObjectName("listView")
        self.label_2 = QtWidgets.QLabel(Form)
        self.label_2.setGeometry(QtCore.QRect(10, 9, 521,
31))
        font = QtGui.QFont()
        font.setPointSize(18)
        self.label_2.setFont(font)
        self.label_2.setObjectName("label_2")

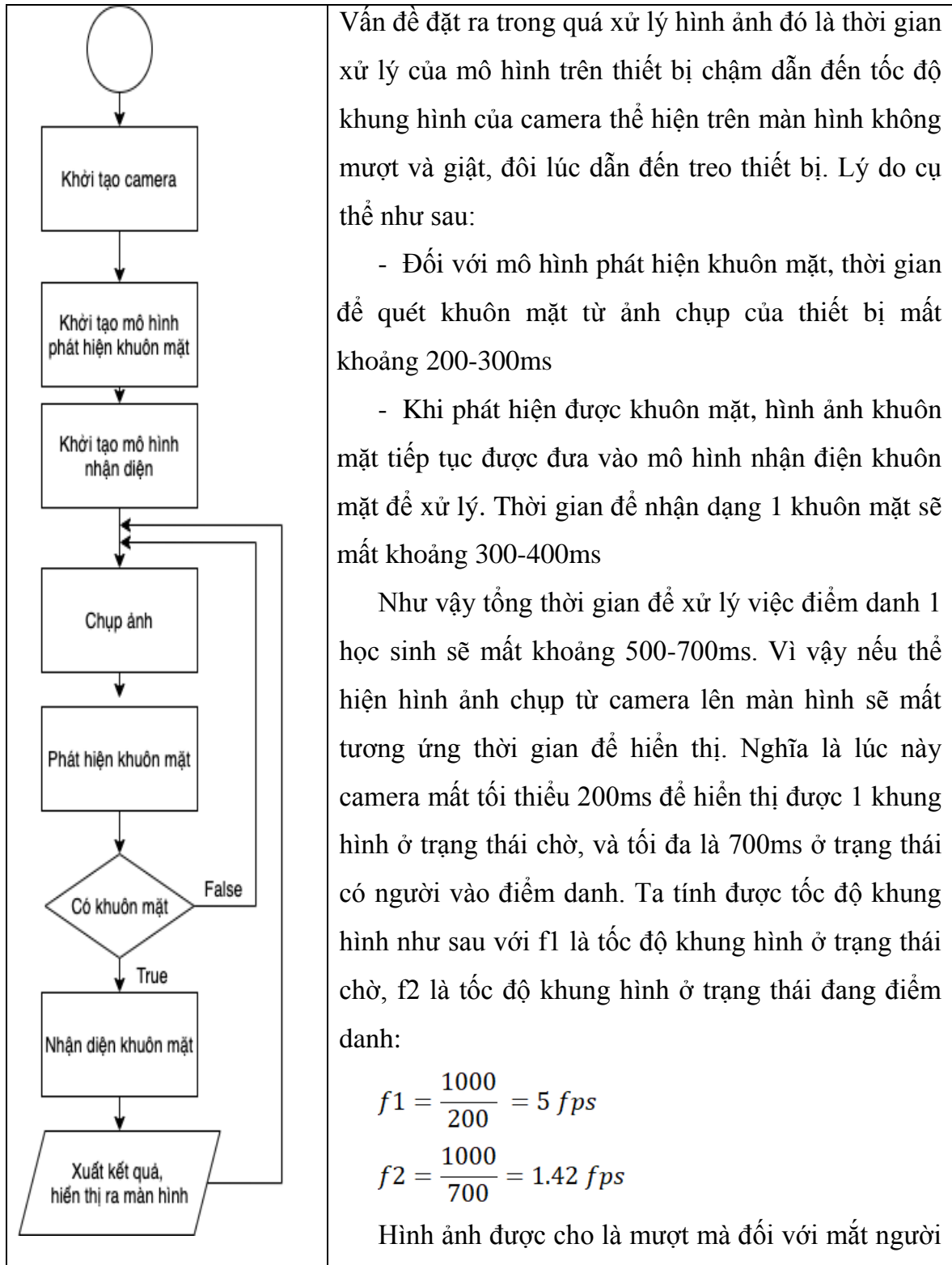
        self.retranslateUi(Form)
        QtCore.QMetaObject.connectSlotsByName(Form)

    def retranslateUi(self, Form):
        _translate = QtCore.QCoreApplication.translate
        Form.setWindowTitle(_translate("Form", "Máy điểm
danh"))
        self.label.setText(_translate("Form", "Ngày hiện
tại"))
        self.pushButton.setText(_translate("Form", "CHỤP
ẢNH"))
        self.pushButton_2.setText(_translate("Form", "CẬP
NHẬT CSDL"))

```

3.2.4. Xử lý nâng cao

Trong quá trình thực hiện sản phẩm, ban đầu tôi sử dụng thuật toán tuần tự như sau để thực hiện việc xử lý hình ảnh:

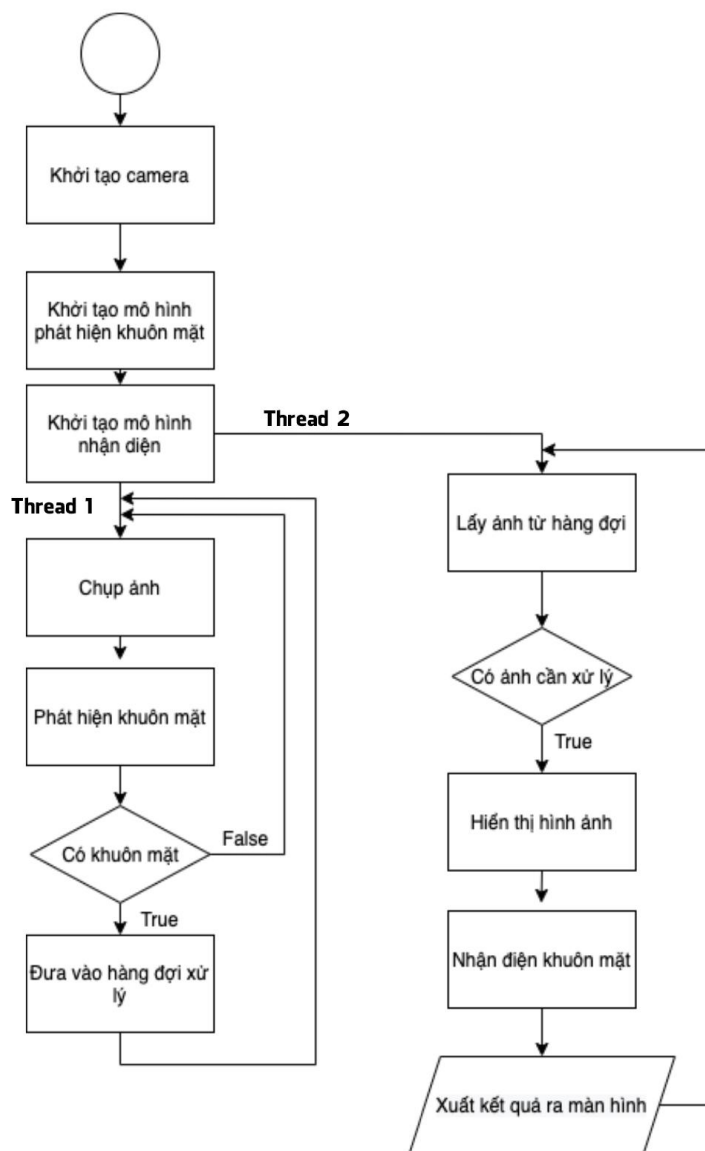


	là 24fps. Như vậy với 5fps và 1.42fps sẽ khiến hình ảnh hiển thị từ máy điểm danh bị giật lag
--	---

Hình 3.8: Thuật toán xử lý ảnh trước khi cải tiến.

Ngoài ra có một vấn đề nữa nếu sử dụng thuật toán này, đó là việc khi mô hình phát hiện khuôn mặt tìm ra nhiều hơn 1 hình ảnh khuôn mặt (có thể do nhiều người điểm danh cùng lúc) thì máy chấm công sẽ dẫn đến bị treo, xảy ra hiện tượng thất cổ chai về mặt xử lý do mô hình nhận dạng khuôn mặt không xử lý kịp.

Để khắc phục điều này, tôi đã sử dụng hàng đợi để xử lý. Hàng đợi sẽ được xây dựng dựa trên thuật toán sau:



Hình 3.9: Thuật toán xử lý ảnh sau khi cải tiến.

Để thực hiện xử lý hàng đợi, cần tạo ra thêm 1 luồng xử lý song song với xử lý chính để giúp việc xử lý được nhanh hơn. Ngoài ra tôi cũng thay đổi thời gian nhận dạng từ liên tục thành chụp ảnh sao mỗi 500ms, điều này ảnh hưởng ít đến tốc độ mà giúp tăng trải nghiệm của người dùng khi sử dụng máy điểm danh.

Như vậy việc sử dụng hàng đợi có kết quả tốt hơn rõ rệt. Sau đây là đoạn code tôi đã thực hiện để đẩy ảnh vào hàng đợi xử lý:

```

IMG_SIZE      = 1280,720                # 640,480 or 1280,720 or
1920,1080
IMG_FORMAT    = QImage.Format_RGB888
DISP_SCALE    = 3                      # Scaling factor for display
image
DISP_MSEC     = 50                     # Delay between display cycles
CAP_API       = cv2.CAP_ANY            # API: CAP_ANY or CAP_DSHOW
etc...
EXPOSURE      = 0                     # Zero for automatic exposure
TEXT_FONT     = QFont("Courier", 10)

camera_num    = 1                      # Default camera (first in list)
image_queue   = Queue.Queue()          # Queue to hold images
capturing     = True                   # Flag to indicate capturing

# Grab images from the camera (separate thread)
def grab_images(cam_num, queue):
    cap = cv2.VideoCapture(cam_num-1 + CAP_API)
    cap.set(cv2.CAP_PROP_FRAME_WIDTH, IMG_SIZE[0])
    cap.set(cv2.CAP_PROP_FRAME_HEIGHT, IMG_SIZE[1])
    if EXPOSURE:
        cap.set(cv2.CAP_PROP_AUTO_EXPOSURE, 0)
        cap.set(cv2.CAP_PROP_EXPOSURE, EXPOSURE)
    else:
        cap.set(cv2.CAP_PROP_AUTO_EXPOSURE, 1)
    while capturing:
        if cap.grab():
            retval, image = cap.retrieve(0)
            if image is not None and queue.qsize() < 2:
                queue.put(image)
            else:
                time.sleep(DISP_MSEC / 1000.0)
        else:
            print("Error: can't grab camera image")
            break

```



```
cap.release()
```

Thiết lập tham số `queue.qsize() < 2` để giới hạn số lượng cần xử lý trong hàng đợi, nếu hàng đợi đang có dữ liệu cần xử lý thì ta sẽ không đẩy thêm vào mà chờ cho hàng đợi xử lý xong.

❖ Cách thức triển khai mô hình lên Raspberry Pi như sau:

Bước 1: Đầu tiên tôi load model đã được train sẵn. Tôi xuất trọng số của mô hình ra 2 file là `mtcnn_model_trained.model` và `resnet_face_identity.model`, sau đó cài đặt mô hình và load 2 file model này vào bằng python.

Bước 2: Tiến hành đọc dữ liệu từ video file hoặc webcam của người dùng sau đó capture ảnh, đưa qua model để detect các khuôn mặt trong đó.

Bước 3: Xử lý dữ liệu đầu ra và tiến hành hiển thị lên màn hình của Pi các thông tin detect được

Do Pi là một máy có cấu hình khá yếu nên tôi đã thực hiện:

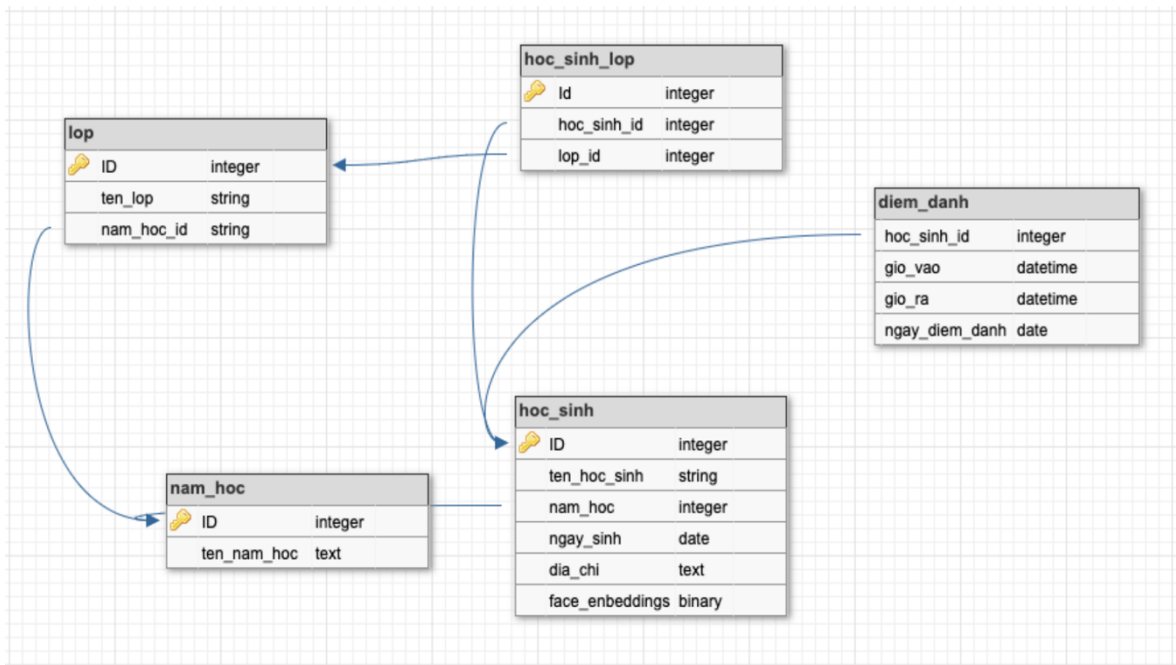
- Resize ảnh đọc từ camera của Pi về size nhỏ hơn trước khi xử lý để tăng tốc độ xử lý
- Thay vì việc đọc all các frame và nhận diện thì tôi thêm 1 biến đếm và chỉ xử lý mỗi 10 frame. Nghĩa là cứ mỗi 10 frame mới detect object một lần.

3.3. Xây dựng cơ sở dữ liệu

Để xây dựng cơ sở dữ liệu ứng dụng điểm danh, tôi sử dụng cơ sở dữ liệu dạng quan hệ được thiết kế như sau:

- Bảng “lop” lưu thông tin lớp học, mỗi lớp học sẽ được tham chiếu đến năm học
- Bảng “nam_hoc” lưu thông tin về từng năm học. Khi bắt đầu năm học mới, người quản trị sẽ phải thêm thông tin vào bảng này
- Bảng “hoc_sinh” lưu thông tin của từng học sinh, ở bảng này có thêm trường “face_embeddings” chứa một danh sách các mảng các đặc trưng của khuôn mặt. Mỗi phần tử của danh sách là trích xuất đặc trưng của một góc chụp khuôn mặt của học sinh. Mỗi học sinh có thể có một hoặc nhiều góc chụp của một khuôn mặt. Nên dữ liệu của trường này là dạng mảng.

- Bảng “hoc_sinh_lop” lưu tham chiếu học sinh thuộc vào một lớp nào đó.
- Bảng “diem_danh” lưu thông tin điểm danh của từng học sinh trong từng buổi học. Bảng này liên kết với bảng ‘hoc_sinh’ thể hiện việc ghi nhận sự có mặt của học sinh nào đó tại thời điểm nhất định. Cụ thể là ghi nhận ngày điểm danh, các thời gian vào/ra của học sinh. Dựa vào bảng này ta sẽ biết được kết quả điểm danh của từng lớp và từng cá nhân trong lớp dựa vào liên kết của bảng “hoc_sinh_lop”.



Hình 3.10: Database hệ thống điểm danh

3.4. Demo và đánh giá kết quả

Khi mỗi khuôn mặt được đưa vào để nhận dạng thì mô hình Resnet sẽ phân lớp và lấy ra vector đặc trưng, sau đó hệ thống sẽ nhận dạng với thuật toán k-NN. Để đưa các thông tin nhận dạng được ra màn hình ở đây tôi sử dụng thư viện OpenCV.

Ban đầu, người dùng sẽ phải khai báo thông tin về khuôn mặt bằng cách chụp ảnh trên máy điểm danh. Sau đó giáo viên chủ nhiệm sẽ tiến hành khớp thông tin của học sinh với dữ liệu hình ảnh từ máy điểm danh gửi về phần mềm, mỗi học sinh có thể có nhiều hình ảnh khuôn mặt với nhiều góc độ chụp khác nhau để tăng độ chính xác.

Sau đó, người dùng với vai trò là học sinh sẽ thực hiện điểm danh, học sinh sẽ phải đứng trước máy điểm danh, sau đó đợi hệ thống nhận dạng để xác định danh tính. Sau khi xác định xong danh tính, hệ thống sẽ gửi dữ liệu lên phần mềm và lưu trữ dữ liệu trên máy chủ cloud để cung cấp số liệu xuất ra báo cáo về kết quả điểm danh như sĩ số, tỉ lệ chuyên cần của từng lớp và gửi đến giáo viên chủ nhiệm .vv...

❖ **Đánh giá độ chính xác của hệ thống**

Sau khi triển khai hệ thống tại trường THPT Thanh Oai B, hệ thống hoạt động ổn định và đạt độ chính xác cao, tỉ lệ nhận dạng sai khuôn mặt rất nhỏ. Để khắc phục những trường hợp nhận dạng không chính xác, khuyến cáo đến người dùng là thử chụp lại ảnh, thay đổi các góc chụp chính xác và sau đó cập nhật lại cho hệ thống. Hầu hết các trường hợp bị nhận dạng sai đều được khắc phục. Cụ thể như sau:

- Trong quá trình triển khai thực tế, thiết bị vẫn xảy ra hiện tượng nhận nhầm học sinh. Trên thiết bị có báo cáo trường hợp thiết bị nhận sai danh tính, tỉ lệ này theo báo cáo tại các lớp học thì chiếm khoảng 5,4% trong tất cả các học sinh điểm danh.

- Các trường hợp bị nhận nhầm khuôn mặt được khắc phục bằng cách chụp lại các ảnh có thể gây nhầm lẫn, cần chụp lại ảnh của cả 2 người bị nhầm lẫn nhau.

3.5. Kết luận chương

Kết thúc chương cuối cùng này, luận văn đã hoàn thành hệ thống nhận dạng khuôn mặt. Kết quả thu được cũng tương đối khả quan so với mong muốn. Bên cạnh đó, trong tương lai khi có tập dữ liệu tốt hơn, mô hình phức tạp hơn nhờ có khả năng xử lý của máy móc tốt hơn thì mô hình CNN sẽ có thể mang lại một kết quả hài lòng hơn nữa.

KẾT LUẬN

Kết quả đạt được của luận văn

Qua nghiên cứu và thực nghiệm, luận văn đã đạt được những kết quả chính sau:

- Nghiên cứu tổng quan về xử lí ảnh.
- Nghiên cứu chi tiết mô hình học sâu tiêu biểu đó là mô hình CNN.
- Áp dụng mô hình CNN cho bài toán nhận dạng khuôn mặt.
- Ứng dụng các thuật toán đã tìm hiểu để giải quyết bài toán phân lớp thông qua mô hình huấn luyện bằng dữ liệu.
- Xây dựng tập dữ liệu huấn luyện và kiểm tra: Bước đầu xây dựng được quy trình thu thập dữ liệu từ thực tế, xử lý dữ liệu thô để đưa vào huấn luyện cho mô hình.
- Xây dựng hệ thống nhận dạng khuôn mặt.
- Triển khai được mô hình trên thiết bị phần cứng thực tế và nền tảng lưu trữ trên máy chủ cloud.

Bước đầu cho thấy hiệu quả của CNN trong việc nhận dạng ảnh đạt được kết quả tương đối khả quan.

Hướng phát triển của luận văn

Luận văn này đạt được một số kết quả nêu trên, nhưng luận văn còn nhiều hạn chế trong việc xây dựng mô hình và xử lí ảnh. Vì vậy, hướng nghiên cứu tiếp theo của luận văn sẽ là:

- Nghiên cứu thêm về mô hình CNN để có thể tăng độ chính xác cho việc nhận dạng trên thực tế.
- Nghiên cứu về mô hình phân lớp để phân loại được không chỉ là khuôn mặt người mà còn có thể dự đoán được tuổi tác và giới tính.
- Có thể phát triển ứng dụng trên bộ dữ liệu đầy đủ và chi tiết hơn.

DANH MỤC CÁC TÀI LIỆU THAM KHẢO.

TÀI LIỆU TIẾNG VIỆT

- [1] Phạm Việt Bình, Đỗ Năng Toàn - Giáo trình môn học xử lý ảnh, Thái Nguyên, tháng 11/2007
- [2] Huỳnh Phước Hải, Nguyễn Văn Hòa, Đỗ Thanh Nghị (2017), So sánh mô hình học sâu với các phương pháp học tự động khác trong phân lớp dữ liệu biểu hiện gen microarray, *Kỷ yếu Hội nghị Quốc gia lần thứ 10 về Nghiên cứu cơ bản và ứng dụng Công Nghệ thông tin (FAIR)*; Đà Nẵng
- [3] Vũ Hữu Tiệp, Machine Learning cơ bản, 2017

TÀI LIỆU TIẾNG ANH

- [4] C. H. R. K. & R. Samer, Image Recognition Using Convolutional Neural Networks., Cadence Whitepaper, 2015.
- [5] D. E. Rumelhart, G. E. Hinton and R. J. Williams, Learning representations by back-propagating errors, *Nature*, Volume 323, Issue
- [6] H. E. Geoffrey, I. Sutskever and K. Alex, ImageNet Classification with Deep Convolutional, 2012.
- [7] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In *BMVC*, 2011.
- [8] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional neural networks. In *ECCV*, 2014.
- [9] O. M. Parkhi, A. Vedaldi, A. Zisserman et al., “Deep face recognition.” in *BMVC*, vol. 1, no. 3, 2015, p. 6
- [10] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv:1408.5093*, 2014.

Trang WEB

- [11] *Convolutional Neural Networks (CNNs / ConvNets)*.
URL: <https://www.qubole.com/blog/deep-learning-the-latest-trend-in-ai-and-ml>
(cited on page 3).
- [12] *Neurons and Nerves*. URL: <https://askabiologist.asu.edu/neuron-anatomy> (cited on page 72).