

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN - ĐIỆN TỬ
BỘ MÔN VIỄN THÔNG



LUẬN VĂN TỐT NGHIỆP ĐẠI HỌC

**NHẬN DẠNG NGÔN NGỮ KÝ HIỆU CHO
NGƯỜI KHIẾM THÍNH SỬ DỤNG KỸ
THUẬT HỌC SÂU: TÁCH VÀ PHÂN TÍCH
ĐẶC TRƯNG KHUNG XƯƠNG TRÊN
VIDEO RGB**

GVHD: PGS.TS HÀ HOÀNG KHA
SVTH: NGUYỄN THÀNH ĐẠT - 1510698

TP. HỒ CHÍ MINH, THÁNG 12 NĂM 2019

-----☆-----

-----☆-----

Số: _____ /BKĐT

Khoa: **Điện – Điện tử**

Bộ Môn: **Điện Tử**

NHIỆM VỤ LUẬN VĂN TỐT NGHIỆP

1. HỌ VÀ TÊN: **Nguyễn Thành Đạt**

MSSV: **1510698**

2. NGÀNH: **Kỹ thuật Điện tử - Viễn Thông**

LỚP : **DD15DV1**

3. Tên đề tài (Tiếng Việt và Tiếng Anh):

Nhận dạng ngôn ngữ ký hiệu cho người khiếm thính sử dụng kỹ thuật học sâu: Tách và phân tích đặc trưng khung xương trên video RGB.

(Sign Language Recognition based on Deep Learning approach: Skeleton Detection and Analysis on RGB video.)

4. Nhiệm vụ :

- Tìm hiểu về Deep Learning, mạng CNN và các nghiên cứu về nhận dạng ngôn ngữ ký hiệu.
- Tìm hiểu các thuật toán rút trích đặc trưng khung xương người từ ảnh.
- Ứng dụng các thuật toán để trích đặc trưng khung xương người từ video.
- Xây dựng mô hình nhận dạng một số từ trong ngôn ngữ ký hiệu của người khiếm thính dựa trên ứng dụng của mạng rút trích đặc trưng khung xương.
- Xây dựng chương trình nhận dạng thời gian thực dựa trên mô hình nhận dạng ngôn ngữ ký hiệu đã xây dựng.
- Đánh giá mô hình, kết luận và đề ra hướng phát triển.

5. Ngày giao nhiệm vụ luận văn: **24/08/2019**

6. Ngày hoàn thành nhiệm vụ: **18/12/2019**

7. Họ và tên người hướng dẫn: **Phản hướng dẫn**

PGS.TS Hà Hoàng Kha

100%

BM Viễn Thông, Khoa Điện - Điện Tử

Nội dung và yêu cầu LVTN đã được thông qua Bộ Môn.

Tp.HCM, ngày 18 tháng 12 năm 2019

CHỦ NHIỆM BỘ MÔN

NGƯỜI HƯỚNG DẪN CHÍNH

PHẦN DÀNH CHO KHOA, BỘ MÔN:

Người duyệt (chấm sơ bộ):.....

Đơn vị:.....

Ngày bảo vệ :

Điểm tổng kết:

Nơi lưu trữ luận văn:

Lời cảm ơn

Trong thời gian thực hiện luận án này, em đã nhận được sự hỗ trợ nhiệt tình, hướng dẫn tận tình và những lời động viên tích cực từ các giảng viên, bạn bè và gia đình. Do đó, em đã hoàn thành luận án như mục tiêu đã đặt ra. Những lời giảng dạy quý báu, không những về mặt kiến thức mà còn về đạo đức làm người của các quý thầy cô sẽ là hành trang cho con đường tương lai của các thế hệ sinh viên.

Em xin gửi đến thầy Hà Hoàng Kha, giảng viên hướng dẫn trực tiếp đề tài, lời biết ơn sâu sắc. Thầy đã dành thời gian quý báu để gấp gáp, thảo luận, đưa ra những vấn đề hay để rèn luyện chúng em khả năng giải quyết vấn đề. Thầy là người đã theo倜傥 bước đi của chúng tôi, tận tình chỉ bảo, hướng dẫn chúng tôi từ khi làm đề cương, thực tập cho đến luận văn. Ngoài những kiến thức chuyên ngành, chúng tôi còn nhận được những lời khuyên, kinh nghiệm quý giá trong học tập và nghiên cứu từ thầy.

Em cũng xin cảm ơn chân thành đến cha mẹ đã động viên và tạo điều kiện giúp đỡ chúng tôi vượt qua những khó khăn trong suốt quá trình học tập và nghiên cứu.

Mặc dù đã cố gắng trong phạm vi và khả năng cho phép, nhưng luận văn vẫn không thể tránh khỏi những thiếu sót, rất mong được sự góp ý của quý thầy cô và các bạn.

Cuối cùng, xin chân thành cảm ơn quý thầy cô và các bạn đã dành thời gian đọc luận văn này.

Lời cam đoan

Em xin cam đoan rằng, luận văn tốt nghiệp "Nhận dạng ngôn ngữ ký hiệu cho người khiếm thính sử dụng kỹ thuật học sâu: tách và phân tích đặc trưng khung xương trên video RGB" là công trình nghiên cứu của em dưới sự hướng dẫn của PGS.TS Hà Hoàng Kha. Đề tài xuất phát từ nhu cầu thực tiễn và nguyện vọng muốn thực hiện của em. Ngoại trừ những nội dung được biên soạn lại từ các công trình khác đã ghi rõ, các nội dung, kết quả kiểm tra đánh giá thực nghiệm trình bày trong luận văn này là kết quả nghiên cứu do chính em thực hiện, hoàn toàn không phải sao chép từ bất kỳ một tài liệu hoặc công trình nghiên cứu nào khác.

Nếu không thực hiện đúng các cam kết trên, em xin hoàn toàn chịu trách nhiệm trước kỷ luật của nhà trường cũng như pháp luật Nhà nước.

Sinh viên thực hiện

Tóm tắt

Trong luận văn này, em đã thực hiện viết ứng dụng realtime nhận diện được một số từ ngữ thông dụng trong ngôn ngữ ký hiệu của người khiếm thính khi giao tiếp. Ứng dụng được xây dựng với mục đích giúp cho người khiếm thính giao tiếp dễ dàng hơn với người bình thường khi họ không hiểu ngôn ngữ ký hiệu của người khiếm thính.

Bởi những phát triển trong học máy và học sâu gần đây, đã giúp con người giải quyết được những bài toán thực tế mà trước đây tưởng chừng như máy tính không thể làm được. Trong ứng dụng này em đã sử dụng một trong các phương pháp học máy đó để giải quyết bài toán nhận diện ngôn ngữ ký hiệu cho người khiếm thính Việt Nam. Phương pháp nhận diện mà em tài sử dụng chia làm 2 bước. Đầu tiên từ hình ảnh RGB có chứa hình ảnh người khiếm thính đang diễn tả một từ ngữ do camera truyền vào. Ứng dụng sử dụng một mạng CNN là mạng mobilenet để trích xuất đặc trưng khung xương là tọa độ các khớp xương trên ảnh 2D (tổng cộng có 18 khớp xương ban đầu). Sau đó các tọa độ khớp xương này được đưa vào một mạng DNN để từ đó phân loại ra các hành động diễn tả từ ngữ mà mạng đã học được. Cuối cùng ứng dụng xuất ra từ ngữ mà người đứng trước camera muốn diễn tả và xuất ra màn hình.

Phương pháp nhận diện dựa vào trích xuất đặc trưng khung xương là một phương pháp khá hay và lạ so với các nghiên cứu liên quan trước đây. Điểm hay của phương pháp này là việc trích xuất khung xương đã khái quát gần như toàn bộ tư thế và hành động mà người khiếm thị muốn diễn tả. Việc này còn làm giảm số chiều dữ liệu so với việc từ một hình ảnh RGB đưa vào để phân loại ra từ ngữ gì. Việc còn lại chỉ cần phân loại hành động dựa trên đặc trưng đã được trích xuất bằng một mạng DNN cấu trúc nhỏ. Việc này giúp tăng tốc độ xử lý của ứng dụng lên đáp ứng được việc xử lý realtime.

Các kết quả thực nghiệm thu được từ hệ thống cho thấy tỷ lệ nhận dạng cao và tốc độ đáp ứng đủ nhanh cho hoạt động chế độ thời gian thực.

ABSTRACT

Key words:

Mục lục

LỜI CẢM ƠN	i
LỜI CAM ĐOAN	ii
TÓM TẮT	iii
MỤC LỤC	v
DANH SÁCH HÌNH VẼ	viii
DANH SÁCH BẢNG	xii
DANH MỤC VIẾT TẮT	xiii
1 TỔNG QUAN	1
1.1 Đặt vấn đề	1
1.2 Những nghiên cứu liên quan	2
1.3 Mục tiêu của luận văn	3
1.4 Nội dung luận văn	4
1.5 Bố cục trình bày	6
2 CƠ SỞ LÝ THUYẾT	7
2.1 MẠNG NEURAL NETWOK	7
2.1.1 Hoạt động của các neuron sinh học	7
2.1.2 Perceptron	8
2.1.3 Kiến trúc mạng Neuron Network	10
2.1.4 Hoạt động của mạng	11
2.1.5 Quá trình huấn luyện một mạng NN	13
2.2 Mạng Convolutional Neural Network (CNN)	22
2.2.1 Phép Tính Convolution	24

2.2.2	Phép convolution trong mạng Neuron Network	27
2.2.3	Batch-Norm	31
2.2.4	Pooling layer	33
2.2.5	Drop-Out	35
2.2.6	Fully connected layer (Dense layer)	36
2.2.7	Hàm Softmax	37
2.3	Kết luận	38
3	ƯỚC TÍNH TỌA ĐỘ KHUNG XƯƠNG TỪ ẢNH RGB	40
3.1	ƯỚC TÍNH DỰA TRÊN CAMERA ĐỘ SÂU KINECT	40
3.1.1	Giới thiệu camera cảm biến độ sâu Kinect của Microsoft	40
3.1.2	Cơ chế trích xuất SJM từ Kinect	42
3.2	ƯỚC TÍNH TỪ ẢNH 2D DỰA TRÊN MẠNG NEURAL NETWORK	45
3.2.1	Tổng quan phương pháp	45
3.2.2	Giải thích các phần	47
3.2.3	Kiến trúc mạng	49
3.2.4	Phân tích nhiều người sử dụng PAFs	50
3.2.5	Kết quả	53
3.3	Kết luận	54
4	XÂY DỰNG MẠNG NEURON NHẬN DẠNG NGÔN NGỮ KÝ HIỆU	55
4.1	Tổng quan	55
4.2	Thu thập dữ liệu	56
4.3	Xử lý dữ liệu đầu vào	61
4.3.1	Loại bỏ các phần SJM dư thừa	61
4.3.2	Chuẩn hóa SJM để phân loại đặc trưng	62
4.4	Cấu trúc mạng neural network xây dựng	66
4.5	Huấn luyện mạng	68
4.5.1	Gradient Descent	68
4.5.2	Stochastic Gradient Descent	69
4.5.3	Điều kiện dừng	70
4.5.4	Áp dụng vào bài toán	71
4.6	Kết quả	73
5	Giải thuật Deep Sort theo dõi từng người trong khung hình	74
5.1	Giải thuật Sort truyền thống	75
5.2	Giải thuật Deep Sort	75

5.3	Kết luận	77
6	THỰC NGHIỆM KẾT QUẢ VÀ ĐÁNH GIÁ	78
6.1	Thực hiện phần mềm nhận dạng ngôn ngữ ký hiệu	78
6.2	Kết quả	78
6.3	Dánh giá	81
6.3.1	Accuracy	81
7	KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	83
7.1	Kết luận	83
7.2	Hướng phát triển	83
TÀI LIỆU THAM KHẢO		84
PHỤ LỤC		86

Danh sách hình vẽ

1.1	Thứ tự hoạt động mô hình nhận dạng	5
2.1	Cấu trúc một tế bào thần kinh (Nguồn: https://askabiologist.asu.edu/neuron-anatomy)	8
2.2	Perceptron	9
2.3	Một neuron cơ bản	9
2.4	Dồ thị các hàm kích hoạt	10
2.5	Mạng Neural Network cơ bản	11
2.6	Mô hình neural network trên gồm 3 layer. Input layer có 2 node ($l^{(0)} = 2$, hidden layer 1 có 3 node, hidden layer 2 có 3 node và output layer có 1 node)	12
2.7	Feedforward	14
2.8	Dường màu đỏ cho $w_{11}^{(1)}$, đường màu xanh cho x_1	20
2.9	backpropagation tác động trong lớp ẩn	21
2.10	Mô hình neural network	21
2.11	Feedforward và Backpropagation	22
2.12	Ví dụ mô hình 1 mô hình convolutional neural network (LeNet-5 [1])	24
2.13	Phép tính Convolution	24
2.14	Convolution feature map có kích thước nhỏ hơn ma trận ban đầu	25
2.15	Ma trận X có viền 0 bên ngoài	25
2.16	$stride = 1, padding = 1$	26

2.17 padding = 1, stride = 2	26
2.18 Phép tính convolution trên ảnh màu với k=3.	27
2.19 Tensor X và W 3 chiều được viết dưới dạng 3 matrix.	28
2.20 Thực hiện phép tính convolution trên ảnh màu	29
2.21 Convolutional layer đầu tiên	30
2.22 Convolutional layer tổng quát	30
2.23 Phép biến đổi back normalization [2]	32
2.24 Phép Pooling với stride = 2, padding = 0	33
2.25 Kết quả sau khi qua pooling layer 2*2. (http://cs231n.github.io/convolutional-networks)	34
2.26 Max pooling và average pooling	35
2.27 Minh họa phương pháp Drop-Out	36
2.28 Phép Flatten biến tensor về 1 vector	37
2.29 Một số ví dụ về đầu vào và đầu ra của hàm softmax Nguồn: : https://machinelearningcoban.com	38
3.1 Camera cảm biến độ sâu Kinect (Nguồn : https://www.ifixit.com)	41
3.2 Ảnh độ sâu từ Kinect (Nguồn : https://www.zonettrigger.com/)	42
3.3 Sơ đồ khối thuật toán trích xuất SJM của thiết bị Kinect (Nguồn : "Nhận dạng ngôn ngữ ký hiệu cho người câm" - B.V Phúc, H.C Thịnh)	43
3.4 Khung xương được trích xuất từ Kinect (Nguồn : https://www.zonettrigger.com/)	44
3.5 Tổng thể kiến trúc	46
3.6 Kiến trúc của CNN nhiều giai đoạn từ phiên bản tạp chí của OpenPose	48
3.7 Kiến trúc của mạng CNN nhiều bước 2 nhánh. Mỗi bước trong nhánh đầu tiên dự đoán những $CFMS^t$, và mỗi bước trong nhánh thứ 2 dự đoán $PAFsL^t$. Sau mỗi bước, những dự đoán từ 2 nhánh, cùng với những đặc trưng ảnh, được nối lại cho bước tiếp theo.	50
3.8 Kết quả ước tính tọa độ khung xương từ mạng NN	53

4.1	Sơ đồ khớp xương xuất ra từ mạng mobilenet	58
4.2	Giao diện thu thập dữ liệu (khi bắt đầu ghi dữ liệu)	59
4.3	Giao diện thu thập dữ liệu (khi đã thu thập đủ dữ liệu)	60
4.4	Sơ đồ khớp xương sau khi loại bỏ các phần không cần thiết	61
4.5	SJM sau chuẩn hóa	65
4.6	Dữ liệu khớp xương sau khi đã được chuẩn hóa	66
4.7	Các thông số của mạng DNN	67
4.8	So sánh Gradient Descent với các hiện tượng vật lý (Nguồn: "Machine learning cơ bản" - Vũ Hữu Tiệp)	69
4.9	Accuracy của tập train và validate	72
4.10	Loss của tập train và validate	72
5.1	Thuật toán Sort Nguồn : "Simple Online and Realtime Tracking with a Deep Association Metric" [3]	75
5.2	Kết quả Deep Sort	76
6.1	Giao diện phần mềm nhận dạng - Mode "Skeleton detection"	79
6.2	Giao diện phần mềm nhận dạng - Mode "Tracking"	80
6.3	Giao diện phần mềm nhận dạng - Mode "Recognize"	80
6.4	Confusion matrix của 16 lớp phân loại	82
7.1	Cấu trúc một tế bào thần kinh	86
7.2	Cấu trúc một tế bào thần kinh	86
7.3	Cấu trúc một tế bào thần kinh	87
7.4	Cấu trúc một tế bào thần kinh	87
7.5	Cấu trúc một tế bào thần kinh	88
7.6	Cấu trúc một tế bào thần kinh	88
7.7	Cấu trúc một tế bào thần kinh	89

DANH SÁCH HÌNH VẼ

7.8 Cấu trúc một tế bào thần kinh	89
7.9 Cấu trúc một tế bào thần kinh	90
7.10 Cấu trúc một tế bào thần kinh	90
7.11 Cấu trúc một tế bào thần kinh	91

Danh sách bảng

1.1	Tổng quan các chương của luận văn	6
4.1	Các khớp xương được xuất ra từ mạng	57
4.2	Các khớp xương được giữ lại	62
4.3	Một SJM đặc trưng nhận được từ quá trình thí nghiệm	63
4.4	Một SJM đặc trưng sau khi chuẩn hóa	65

Danh mục viết tắt

NN	: Neural Network
CNNs	: Convolutional Neural Networks
DNN	: Deep Neural Network
DL	: Deep Learning
MLP	: Multi Layer Perceptron
SGD	: Stochastic Gradient Descent
SJM	: Skeleton Joints Mapping
PAFs	: Part Affinity Fields
CFMs	: Confidence Maps

Chương 1

TỔNG QUAN

Nội dung chương đặt vấn đề tổng quan bài toán nhận dạng ngôn ngữ ký hiệu hiện nay và một số các công trình nghiên cứu về nhận dạng ngôn ngữ ký hiệu đã công bố trong nước và quốc tế. Từ đó đưa ra lý do chọn đề tài nhận dạng ngôn ngữ ký hiệu, giới thiệu về công cụ hỗ trợ cũng như mục tiêu, nhiệm vụ của đề tài cần tập trung giải quyết. Ngoài ra, chương mở đầu cũng giới thiệu phương pháp xử lý đề tài luận văn và trình bày bối cảnh nội dung trình bày xuyên suốt bài báo cáo.

1.1 Đặt vấn đề

Khiếm thính là trình trạng một người có thính giác kém, không nghe được những âm thanh, tiếng nói mà một người bình thường có thể nghe được. Theo số liệu thống kê năm 2014 của Trung tâm nghiên cứu Giáo dục đặc biệt (Viện Khoa học GD Việt Nam), Việt Nam có khoảng 7 triệu người khuyết tật, trong đó có hơn 1 triệu người khiếm thính. Do khả năng nghe bị suy giảm nên việc giao tiếp bằng lời nói ở cộng đồng người khiếm thính và với người bình thường thường như là không thể. Để thay thế cho việc giao tiếp bằng tiếng nói, “ngôn ngữ ký hiệu” được ra đời nhằm phục vụ việc giao tiếp trực tiếp mà không cần thông qua lời nói. Ngôn ngữ ký hiệu hay ngôn ngữ dấu hiệu, thủ ngữ là ngôn ngữ dùng những biểu hiện của bàn tay thay cho âm thanh của tiếng nói. Ngôn ngữ ký hiệu do người khiếm thính tạo ra nhằm giúp họ có thể giao tiếp với nhau trong cộng đồng của mình và tiếp thu tri thức của xã hội. Ngôn ngữ ký hiệu không như chữ viết tay hay lời nói có một cách thức và ngữ pháp cụ thể, ký hiệu rõ ràng để có thể mô hình hóa được. Để sử dụng ngôn ngữ ký hiệu, người giao tiếp cần thể hiện cử chỉ bằng

cả bàn tay và cánh tay, kết hợp với điệu bộ của cơ thể để có thể diễn tả ý nghĩa mong muốn. Tuy được cộng đồng người khiếm thính sử dụng phổ biến nhưng đối với người bình thường, hầu như đa số đều không hiểu ngôn ngữ ký hiệu. Ngoài ra, ngôn ngữ ký hiệu giữa các nước trên thế giới, thậm chí giữa các vùng miền trong một nước cũng có sự khác nhau. Việc này khiến cho việc giao tiếp của người khiếm thính gặp rất nhiều khó khăn. Vì vậy một hệ thống nhận dạng ngôn ngữ ký hiệu tự động phiên dịch sang tiếng nói giúp người khiếm thính hòa nhập cộng đồng là thật sự cần thiết. Với những lý do đó, trong đề cương này em xin trình bày một mô hình nhận dạng ngôn ngữ ký hiệu qua phân tích hình ảnh từ camera. Hệ thống này có khả năng quan sát, phát hiện con người và nhận diện hành động mà người đứng trước camera muốn diễn đạt.

1.2 Những nghiên cứu liên quan

Lĩnh vực xử lý, phân loại, nhận diện ngôn ngữ ký hiệu rất rộng và phức tạp do tính phong phú của chữ cái, câu từ và đặc tính không cấu trúc của các hành động con người. Các hướng phát triển trong lĩnh vực này rất tiềm năng và bùng nổ trong những năm gần đây.

Trên thế giới, đã có nhiều nghiên cứu phát triển các dịch vụ thông dịch ngôn ngữ ký hiệu và các sản phẩm công nghệ nhằm hỗ trợ người khiếm thính trong giao tiếp xã hội. Một số sản phẩm nổi bật như găng tay chuyển đổi ngôn ngữ ký hiệu thành giọng nói [4], các phần mềm dịch từ văn bản/ giọng nói sang ngôn ngữ ký hiệu hay các từ điển tra cứu ngôn ngữ ký hiệu online [5]. Một số tác giả cũng đã nghiên cứu sử dụng thiết bị Kinect trong việc nhận dạng các con số và các ký tự chữ cái theo ký hiệu ngôn ngữ người câm [6], tuy nhiên việc nhận dạng là dựa trên ảnh tĩnh chưa có những giải pháp nhận dạng ảnh động theo như các ký hiệu hiệu ngôn ngữ tiếng Việt.

Ở các nước, các nhà nghiên cứu đã tiếp cận bài toán nhận dạng cử chỉ bàn tay theo rất nhiều hướng khác nhau như dựa vào màu sắc bàn tay, hình dáng bàn tay hay công trình của Viola & Jones dùng các đặc trưng Haarlike.

Ngoài ra, lĩnh vực nhận diện cử chỉ của con người được quan tâm và nghiên cứu trong rất nhiều năm từ năm 1992 [7] với giải thuật HMM của Junji YAMATO cho đến những năm gần đây, sử dụng phương pháp SVM cục bộ của Christian Schudlt vào năm 2004 [8], và các phương pháp khác như: giải thuật khung xương [9], [10], [11]. Các bài khảo sát chi tiết có thể được tìm

thấy ở [12] khảo sát cách thức nhận diện hành động con người và đưa ra chi tiết nhiều bài báo có thể tham khảo.

Như đã đề cập ở các phần trên, phương pháp SVM chỉ hoạt động tốt với các ảnh tĩnh, nên các nhận diện chính xác của SVM phải nói chính xác là các cử chỉ của con người tại thời điểm tức thời đó đã được nhận diện chính xác, không phải là một hành động gồm nhiều cử chỉ. Luận văn này bên cạnh SVM thì cũng tập trung nhiều vào HMM nên các bài báo liên quan đến SVM và HMM đều được xem xét kỹ, trong đó có một số phương pháp rất hay và có thể được xem xét:

- Xây dựng mô hình 3D dựa vào nhiều camera ở các vị trí khác nhau [13]
- Nhận diện hành động con người dựa trên mô tả các đặc trưng góc 3D [14]
- Sử dụng nhiều thiết bị được gắn trên người [15]

Mặc dù đã được nghiên cứu trong thời gian dài, tuy nhiên do đặc điểm của việc nhận dạng hành động con người rất phức tạp và không có một cấu trúc rõ ràng dẫn đến việc rất khó áp dụng cho thực tế hoặc một lĩnh vực rộng rãi.

1.3 Mục tiêu của luận văn

Dể có thể thực hiện việc nhận dạng ngôn ngữ ký hiệu, cần xác định được phương pháp thực hiện, lựa chọn các giải thuật hợp lý phù hợp với điều kiện thực tế và khả năng có thể ứng dụng cao nhất. Phương án thực hiện được lựa chọn ban đầu có 3 phương án:

Phương án 1: sử dụng thiết bị kinect để trích xuất hình dáng khung xương sau đó nhận dạng ngôn ngữ ideoký hiệu bằng thuật toán Hidden Markov Model từ chuỗi ký tự khung xương được trích xuất ra.

Phương án 2: Thực hiện xây dựng mạng kết hợp Convolutional Neural Network kết hợp với Long Short Term Memory để nhận dạng hành động từ video.

Phương án 3: Trích xuất hình dáng khung xương từ từng frame ảnh 2D bằng mạng CNN để xuất ra tọa độ khung xương trên hệ tọa độ 2D. Sử dụng một mạng Deep neural network để nhận dạng ngôn ngữ ký hiệu từ chuỗi khung xương đó.

Ở phương án 1 việc sử dụng thiết bị kinect sẽ là quá cồng kềnh để có thể mang theo, và sử dụng trong đời sống hàng ngày. Mô hình HMM có thể là một mô hình máy học cổ điển, còn nhiều nhược điểm hơn so với các mô hình mới hiện nay. Ở phương án 2, để phân loại hành động, cần có tập dữ liệu lớn vì với mỗi góc nhìn khác nhau sẽ cho ra một ảnh khác nhau và với cùng một hành động sẽ cho ra những đặc trưng khác nhau, như vậy sẽ khó để có thể xây dựng mô hình này. Phương án 3 được chọn vì có nhiều ưu điểm hơn phương án 1 và 2. Vì tính khả thi có thể ứng dụng được trên các thiết bị có camera mà người sử dụng có thể mang đi như điện thoại thông minh.

Ứng dụng hướng tới hỗ trợ cộng đồng người người khiếm thính trong giao tiếp thường ngày nên việc đầu tiên cần hướng đến là tính tiện lợi và dễ sử dụng. Vì vậy thiết bị giúp hỗ trợ người khiếm thính cần có thể dễ dàng mang đi gọn nhẹ. Do đó mục tiêu luận văn hướng đến là thực hiện phần mềm để có thể hoạt động trên điện thoại thông minh.

1.4 Nội dung luận văn

Nội dung luận văn sẽ hướng đến xây dựng phần mềm nhận diện một số từ ngữ trong bộ ngôn ngữ ký hiệu của người khiếm thính ở TP.HCM. Các nội dung này sẽ sắp xếp xoay quanh theo thứ tự hoạt động của mô hình nhận diện ngôn ngữ ký hiệu được xây dựng trong luận văn. Tổng quan, phần mềm nhận diện sẽ hoạt động theo các bước trong sơ đồ hình 1.1.



Hình 1.1: Thứ tự hoạt động mô hình nhận dạng

1.5 Bố cục trình bày

Toàn bộ nội dung của luận văn sẽ đi theo bố cục chặt chẽ theo thứ tự các bước mà luận văn đã nghiên cứu và thực hiện. Các nội dung chính của luận văn sẽ được chia thành các chương cụ thể để có thể dễ dàng xem xét, và nắm bắt vấn đề. Tại mỗi chương sẽ trình bày cụ thể từ lý thuyết của chương đó cho đến cụ thể các bước thực hiện cũng như kết luận từng chương.

Bố cục của luận văn sẽ được trình bày theo trình tự và những nội dung được khái quát trong bảng 1.1.

Bảng 1.1: Tổng quan các chương của luận văn

Chương	Nội dung
Chương 1	Giới thiệu chung về ngôn ngữ ký hiệu; Các nghiên cứu về nhận dạng ngôn ngữ ký hiệu; sơ lược mục tiêu, tổng quan và cấu trúc các phần của luận văn.
Chương 2	Trình bày lý thuyết về học sâu, các thuật toán huấn luyện mạng và mạng mobile net.
Chương 3	Giới thiệu các nghiên cứu về ước lượng đặc trưng khung xương; Cách thức hoạt động của mạng ước tính đặc trưng khung xương mà luận văn sử dụng .
Chương 4	Trình bày về mạng neural network được luận văn xây dựng để phân loại các từ trong ngôn ngữ ký hiệu; Cách xử lý dữ liệu đầu vào của mạng và cách huấn luyện mạng.
Chương 5	Trình bày giải thuật Deep Sort dùng để theo dõi từng người trong khung hình.
Chương 6	Xây dựng ứng dụng nhận diện dựa trên mô hình đã huấn luyện. Các thử nghiệm, kết quả và đánh giá mô hình.
Chương 7	Trình bày kết luận và các hướng phát triển để hoàn thiện đề tài.

o

Chương 2

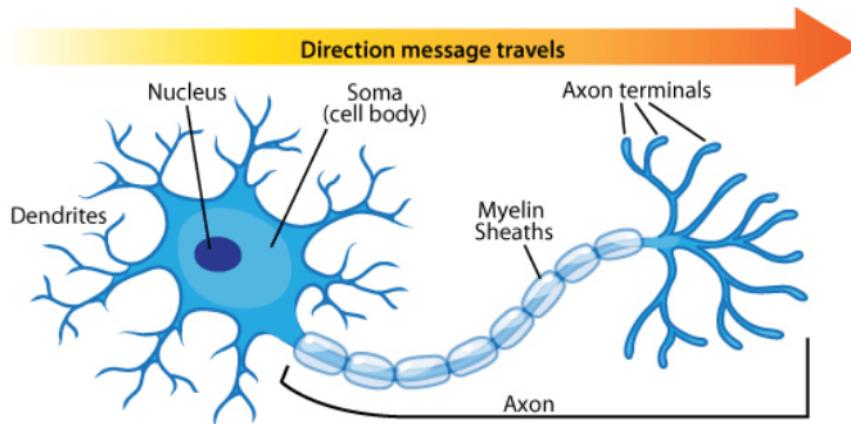
CƠ SỞ LÝ THUYẾT

2.1 MẠNG NEURAL NETWORK

Mạng nơron nhân tạo (Neural Network - NN) là một mô hình tính toán được lấy cảm hứng từ các mạng neuron sinh học trong não người để xử lý thông tin. Kết hợp với các kỹ thuật học sâu khác (Deep Learning - DL), NN trở thành một công cụ hiệu quả và có nhiều kết quả đột phá cho nhiều bài toán khó như nhận dạng ảnh, nhận dạng giọng nói trong lĩnh vực thị giác máy tính và xử lý ngôn ngữ tự nhiên. Trong chương này, luận văn sẽ trình bày các lý thuyết cơ bản của mạng NN từ các thành phần cơ bản, kiến trúc mạng đến các kỹ thuật huấn luyện (training) một mạng NN.

2.1.1 Hoạt động của các neuron sinh học

Neuron Anatomy



Hình 2.1: Cấu trúc một tế bào thần kinh

(Nguồn: <https://askabiologist.asu.edu/neuron-anatomy>)

Neuron là đơn vị cơ bản cấu tạo hệ thống thần kinh và là một phần quan trọng nhất của não. Não chúng ta gồm khoảng 10 triệu neuron và mỗi neuron liên kết với khoảng 10.000 neuron khác.

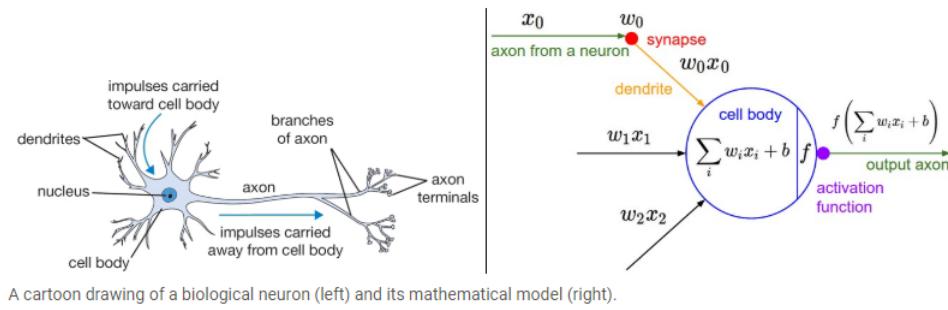
Ở mỗi neuron có phần thân (soma) chứa nhân, các tín hiệu đầu vào qua sợi nhánh (dendrites) và các tín hiệu đầu ra qua sợi trực (axon) kết nối với các neuron khác. Hiểu đơn giản mỗi neuron nhận dữ liệu đầu vào qua sợi nhánh và truyền dữ liệu đầu ra qua sợi trực, đến các sợi nhánh của các neuron khác.

Mỗi neuron nhận xung điện từ các neuron khác qua sợi nhánh. Nếu các xung điện này đủ lớn để kích hoạt neuron, thì tín hiệu này đi qua sợi trực đến các sợi nhánh của các neuron khác. Mạng neuron cần quyết định có kích hoạt neuron đấy hay không.

2.1.2 Perceptron

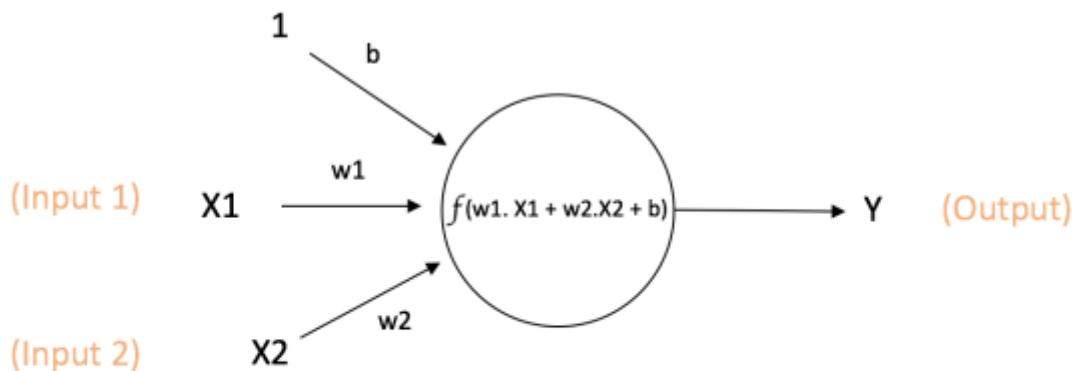
Lấy ý tưởng từ neuron sinh học, neuron nhân tạo với tên gọi perceptron cũng hoạt động theo cách gần giống với neuron sinh học để tạo thành một mạng thần kinh nhân tạo cho máy tính.

Đơn vị tính toán cơ bản trong một mạng NN được gọi là perceptron (hình 2.2) và thường được gọi là **node** hay **unit**. Một node nhận các đầu vào từ các nodes khác hoặc từ nguồn bên ngoài, sau đó tính toán tạo ra giá trị ngõ ra. Mỗi ngõ vào có một trọng số liên kết và giá trị



Hình 2.2: Perceptron

này biểu thi mức độ liên quan giữa node hiện tại và node trước nó. Node áp dụng một hàm f (được giới thiệu ở nội dung bên dưới) vào tổng các các tích ngõ vào và trọng số để tạo giá trị ngõ ra. Hình 2.3 miêu tả chi tiết một neuron và các hoạt động của nó.



$$\text{Output of neuron} = Y = f(w_1 \cdot X_1 + w_2 \cdot X_2 + b)$$

Hình 2.3: Một neuron cơ bản

Node trong hình 2.3 có hai đầu vào **Input 1** và **Input 2** có giá trị tương ứng X_1 và X_2 , trọng số là **w1** và **w2**. Ngoài ra, có một đầu vào khác với giá trị **1** và trọng số **b** (được gọi là bias). Ngõ ra của node là Y được tính toán như hình 2.3. Hàm f là hàm phi tuyến tính và được gọi là hàm kích hoạt (Activation Function). Đặc tính phi tuyến của hàm kích hoạt giúp mạng NN có thể "học" những dữ liệu thực trong tự nhiên và hầu hết chúng đều có tính chất phi tuyến.

Các hàm kích hoạt thường được sử dụng trong thực tế:

- **Sigmoid:** lấy giá trị ngõ vào thực và ép nó nằm trong giới hạn $[0, 1]$.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (2.1)$$

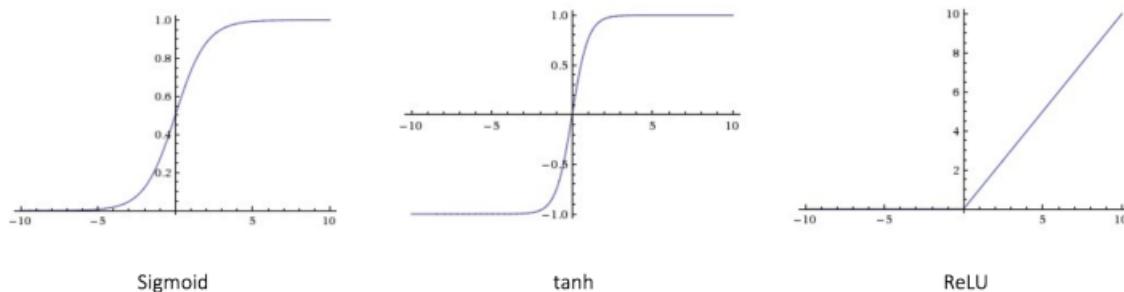
- **Tanh:** lấy giá trị ngõ vào thực và ép nó nằm trong giới hạn $[-1, 1]$.

$$\tanh(x) = 2\sigma(2x) - 1 \quad (2.2)$$

- **ReLU:** lấy giá trị ngõ vào thực và lấy ngưỡng ở 0 (thay thế các giá trị âm bằng 0 hoặc giá trị rất nhỏ).

$$f(x) = \max(0, x) \quad (2.3)$$

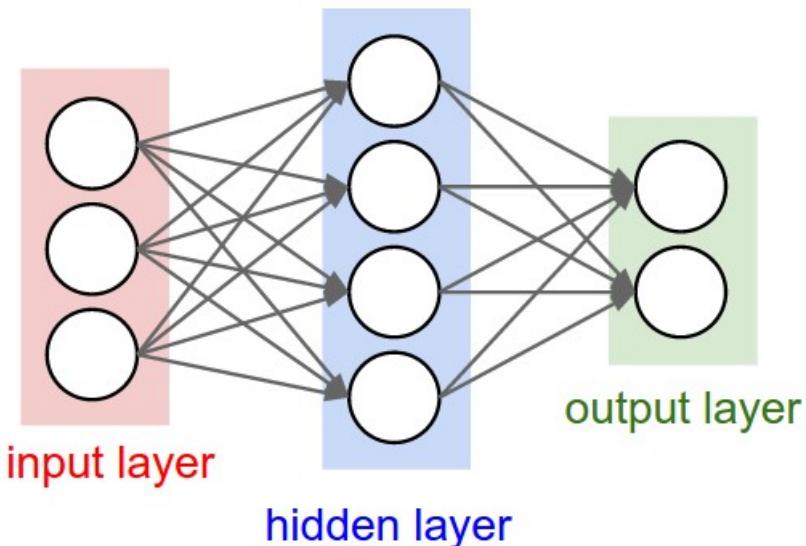
Đồ thị các hàm kích hoạt được mô tả trong hình 2.4.



Hình 2.4: Đồ thị các hàm kích hoạt

2.1.3 Kiến trúc mạng Neuron Network

Một mạng NN được xây dựng gồm nhiều lớp (layer). Mỗi lớp được cấu thành từ nhiều node cơ bản (đã trình bày trong mục 2.1.2). Ngõ ra của các node ở lớp phía trước là ngõ vào của các node lớp phía sau, chúng được gọi là các liên kết (connection) và tương ứng với các trọng số khác nhau.



Hình 2.5: Mạng Neural Network cơ bản

Một mạng NN cơ bản sẽ có 3 tầng (minh họa trong hình 2.5):

- **Tầng vào (input layer):** Là tầng bên trái cùng của mạng thể hiện cho các đầu vào của mạng.
- **Tầng ra (output layer):** Là tầng bên phải cùng của mạng thể hiện cho các đầu ra của mạng.
- **Tầng ẩn (hidden layer):** Là tầng nằm giữa tầng vào và tầng ra thể hiện cho việc suy luận logic của mạng.

Một mạng NN chỉ có một tầng vào và một tầng ra, nhưng có thể có nhiều tầng ẩn. Số lượng tầng ẩn phụ thuộc vào độ phức tạp của bài toán được mạng NN giải quyết và được thiết kế dựa trên kinh nghiệm của người xây dựng mạng.

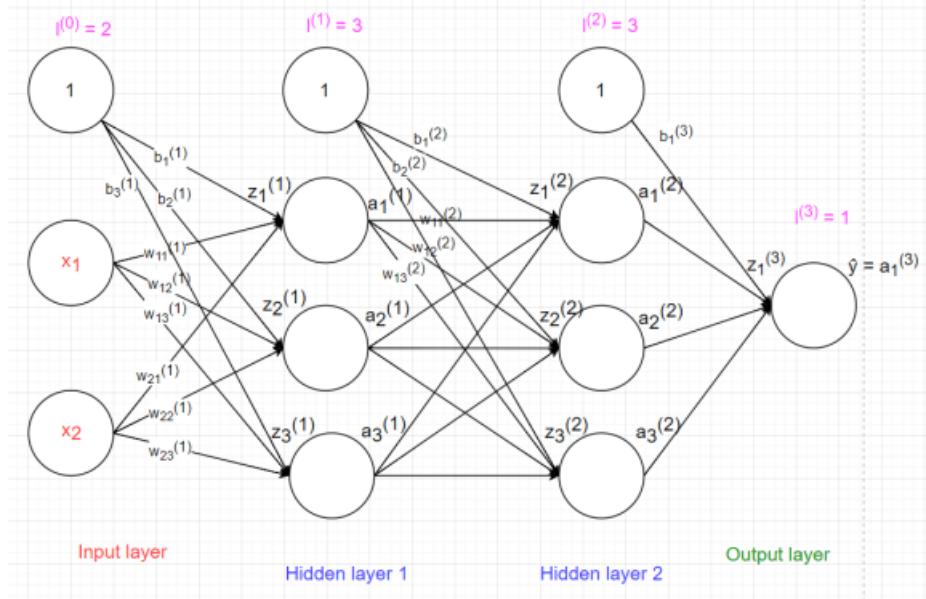
2.1.4 Hoạt động của mạng

Tín hiệu đầu vào (gồm các thông tin cần dự đoán) sẽ được truyền từ input layer. Sau đó được tính toán qua các hidden layer bởi các nodes. Cuối cùng output layer sẽ thực hiện việc dự đoán và phân loại.

Mỗi node trong hidden layer và output layer sẽ thực hiện các công việc sau:

- Liên kết với tất cả các node ở layer trước đó với các hệ số w riêng.
- Mỗi node có 1 hệ số bias b riêng.
- Diễn ra 2 bước: tính tổng linear và áp dụng activation function đưa ra output của node.

Để hiểu rõ ràng nhất, ta đi sâu vào các tính toán trong một mạng NN cụ thể như hình 2.6.



Hình 2.6: Mô hình neural network trên gồm 3 layer. Input layer có 2 node ($l^{(0)} = 2$, hidden layer 1 có 3 node, hidden layer 2 có 3 node và output layer có 1 node.

Ký hiệu:

- Số node trong hidden layer thứ i là $l^{(i)}l(i)$.
- Ma trận $W^{(k)}$ kích thước $l^{(k-1)} * l^{(k)}$ là ma trận hệ số giữa layer $(k-1)$ và layer k , trong đó $w_{ij}^{(k)}$ là hệ số kết nối từ node thứ i của layer $k-1$ đến node thứ j của layer k .
- Vector $b^{(k)}$ kích thước $l^k * 1$ là hệ số bias của các node trong layer k , trong đó $b_i^{(k)}$ là bias của node thứ i trong layer k .

Với node thứ i trong layer l có bias $b_i^{(l)}$ thực hiện 2 bước:

- Tính tổng linear: $z_i^{(l)} = \sum_{j=1}^{l^{(l-1)}} a_j^{(l-1)} * w_{ji}^{(l)} + b_i^{(l)}$ là tổng tất cả các node trong layer trước nhân với hệ số w tương ứng, rồi cộng với bias b .

- Áp dụng activation function: $a_i^{(l)} = \sigma(z_i^{(l)})$

Vector $z^{(k)}$ kích thước $l^{(k)} * 1$ là giá trị các node trong layer k sau bước tính tổng linear.

Vector $a^{(k)}$ kích thước $l^{(k)} * 1$ là giá trị của các node trong layer k sau khi áp dụng hàm activation function.

Do mỗi node trong hidden layer và output layer đều có bias nên trong input layer và hidden layer cần thêm node 1 để tính bias (nhưng không tính vào tổng số node layer có).

Tại node thứ 2 ở layer 1, ta có:

- $z_2^{(1)} = x_1 * w_{12}^{(1)} + x_2 * w_{22}^{(1)} + b_2^{(1)}$
- $a_2^{(1)} = \sigma(z_2^{(1)})$

Hay ở node thứ 3 layer 2, ta có:

- $z_3^{(2)} = a_1^{(1)} * w_{13}^{(2)} + a_2^{(1)} * w_{23}^{(2)} + a_3^{(1)} * w_{33}^{(2)} + b_3^{(2)}$
- $a_2^{(1)} = \sigma(z_2^{(1)})$

2.1.5 Quá trình huấn luyện một mạng NN

Quá trình huấn luyện một mạng NN được thể hiện qua sự lặp đi lặp lại hai bước sau:

- **Feedforward:** Lan truyền tiến. Dự đoán output \hat{y} với một input x bằng cách tính toán từ đầu đến cuối của mạng neuron.
- **Backpropagation:** Lan truyền ngược và cập nhật trọng số.

Bước 1: Lan truyền tiến Để nhất quán về mặt ký hiệu, gọi input layer là $a^{(0)} (= x)$ kích thước $2 * 1$.

$$z^{(1)} = \begin{bmatrix} z_1^{(1)} \\ z_2^{(1)} \\ z_3^{(1)} \end{bmatrix} = \begin{bmatrix} a_1^{(0)} * w_{11}^{(1)} + a_2^{(0)} * w_{21}^{(1)} + a_3^{(0)} * w_{31}^{(1)} + b_1^{(1)} \\ a_1^{(0)} * w_{12}^{(1)} + a_2^{(0)} * w_{22}^{(1)} + a_3^{(0)} * w_{32}^{(1)} + b_2^{(1)} \\ a_1^{(0)} * w_{13}^{(1)} + a_2^{(0)} * w_{23}^{(1)} + a_3^{(0)} * w_{33}^{(1)} + b_3^{(1)} \end{bmatrix} = (W^{(1)})^T * a^{(0)} + b^{(1)}$$

$$a^{(1)} = \sigma(z^{(1)})$$

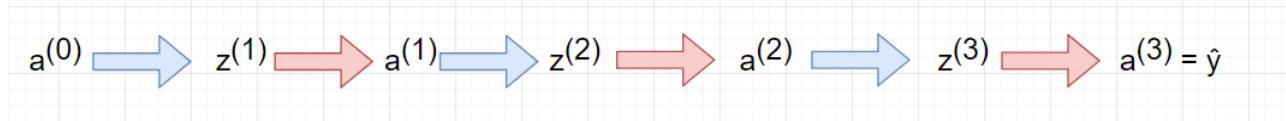
Tương tự ta có:

$$z^{(2)} = (W^{(2)})^T * a^{(1)} + b^{(2)}$$

$$a^{(2)} = \sigma(z^{(2)})$$

$$z^{(3)} = (W^{(3)})^T * a^{(2)} + b^{(3)}$$

$$\hat{y} = a^{(3)} = \sigma(z^{(3)})$$



Hình 2.7: Feedforward

□ Biểu diễn dưới dạng ma trận:

Tuy nhiên khi làm việc với dữ liệu ta cần tính dự đoán cho nhiều dữ liệu một lúc, nên gọi X là ma trận $n * d$, trong đó n là số dữ liệu và d là số trường trong mỗi dữ liệu, trong đó $x_j^{[i]}$ là giá trị trường dữ liệu thứ j của dữ liệu thứ i . Biểu diễn dạng ma trận của vector dữ liệu đầu vào như sau:

$$X = \begin{bmatrix} x_1^{[1]} & x_2^{[1]} & \dots & x_d^{[1]} \\ x_1^{[2]} & x_2^{[2]} & \dots & x_d^{[2]} \\ \dots & \dots & \dots & \dots \\ x_1^{[n]} & x_2^{[n]} & \dots & x_d^{[n]} \end{bmatrix} = \begin{bmatrix} -(x^{[1]})^T - \\ -(x^{[2]})^T - \\ \dots \\ -(x^{[n]})^T - \end{bmatrix}$$

Do $x^{[1]}$ là vector kích thước $d * 1$ tuy nhiên ở X mỗi dữ liệu được viết theo hàng nên cần transpose $x^{[1]}$ thành kích thước $1 * d$, kí hiệu: $(x^{[1]})^T$. Gọi ma trận $Z^{(i)}$ kích thước $N * l^{(i)}$ trong đó $z_j^{(i)[k]}$ là giá trị thứ j trong layer i sau bước tính tổng linear của dữ liệu thứ k trong dataset.

*** Kí hiệu (i) là layer thứ i và kí hiệu $[k]$ là dữ liệu thứ k trong dataset.

Tương tự, gọi ma trận $A^{(i)}$ kích thước $N * l^{(i)}$ trong đó $a_j^{(i)[k]}$ là giá trị thứ j trong layer i sau khi áp dụng activation function của dữ liệu thứ k trong dataset.

$$X = \begin{bmatrix} z_1^{(i)[1]} & z_2^{(i)[1]} & \dots & z_{l^{(i)}}^{(i)[1]} \\ z_1^{(i)[2]} & z_2^{(i)[2]} & \dots & z_{l^{(i)}}^{(i)[2]} \\ \dots & \dots & \dots & \dots \\ z_1^{(i)[n]} & z_2^{(i)[n]} & \dots & z_{l^{(i)}}^{(i)[n]} \end{bmatrix} = \begin{bmatrix} -(z^{(i)[1]})^T - \\ -(z^{(i)[2]})^T - \\ \dots \\ -(z^{(i)[n]})^T - \end{bmatrix}$$

Do đó:

$$Z^{(1)} = \begin{bmatrix} (z^{(1)[1]})^T \\ (z^{(1)[2]})^T \\ \dots \\ (x^{(1)[n]})^T \end{bmatrix} = \begin{bmatrix} (x^{[1]})^T * w^{[1]} + (b^{(1)})^T \\ (x^{[2]})^T * w^{(1)} + (b^{(1)})^T \\ \dots \\ (x^{[n]})^T * w^{(1)} + (b^{(1)})^T \end{bmatrix} = X * W^{(1)} + \begin{bmatrix} (b^{(1)})^T \\ (b^{(1)})^T \\ \dots \\ (b^{(1)})^T \end{bmatrix} = X * W^{(1)} + b^{(1)}$$

Như vậy:

$$A^{(1)} = \sigma(Z^{(1)})$$

$$Z^{(2)} = A^{(1)} * W^{(2)} + b^{(2)}$$

$$A^{(2)} = \sigma(Z^{(2)})$$

$$Z^{(3)} = A^{(2)} * W^{(3)} + b^{(3)}$$

$$\hat{Y} = A^{(3)} = \sigma(Z^{(3)})$$

Vậy là có thể tính được giá trị dự đoán của nhiều dữ liệu 1 lúc dưới dạng ma trận.

Giờ từ input X ta có thể tính được giá trị dự đoán \hat{Y} , tuy nhiên việc chính cần làm là đi tìm hệ số W và b . Có thể nghĩ ngay tới thuật toán gradient descent và việc quan trọng nhất trong thuật toán gradient descent là đi tìm đạo hàm của các hệ số đối với loss function. Và việc tính đạo hàm của các hệ số trong neural network được thực hiện bởi thuật toán backpropagation, sẽ được trình bày ở bước sau.

Bước 2: Backpropagation - Lan truyền ngược và cập nhật trọng số Giờ ta cần đi tìm hệ số W và b . Có thể nghĩ ngay tới thuật toán gradient descent và việc quan trọng nhất trong thuật toán gradient descent là đi tìm đạo hàm của các hệ số đối với loss function. Bước này sẽ tính đạo hàm của các hệ số trong neural network với thuật toán backpropagation.

Quá trình học vẫn là tìm lối một hàm lỗi để đánh giá và tìm cách tối ưu hàm lỗi đó để được kết quả hợp lý nhất có thể. Với mỗi điểm $(x^{[i]}, y_i)$ ta có hàm loss function được tính theo công thức:

$$L = -(y_i * \log(\hat{y}_i) + (1 - y_i) * \log(1 - \hat{y}_i))$$

Hàm loss function trên toàn bộ dữ liệu:

$$J = -\sum_{i=1}^N (y_i * \log(\hat{y}_i) + (1 - y_i) * \log(1 - \hat{y}_i))$$

■ Gradient Descent

Để áp dụng gradient descent ta cần tính được đạo hàm của các hệ số W và bias b với hàm loss function. *** Kí hiệu chuẩn về đạo hàm

- Khi hàm $f(x)$ là hàm 1 biến x , ví dụ: $f(x) = 2 * x + 1$. Đạo hàm của f đối với biến x kí hiệu là $\frac{df}{dx}$
- Khi hàm $f(x, y)$ là hàm nhiều biến, ví dụ $f(x, y) = x^2 + y^2$. Đạo hàm f với biến x kí hiệu là $\frac{\partial f}{\partial x}$

Với mỗi điểm $(x^{[i]}, y_i)$, hàm loss function sẽ là:

$$L = -(y_i * \log(\hat{y}_i) + (1 - y_i) * \log(1 - \hat{y}_i))$$

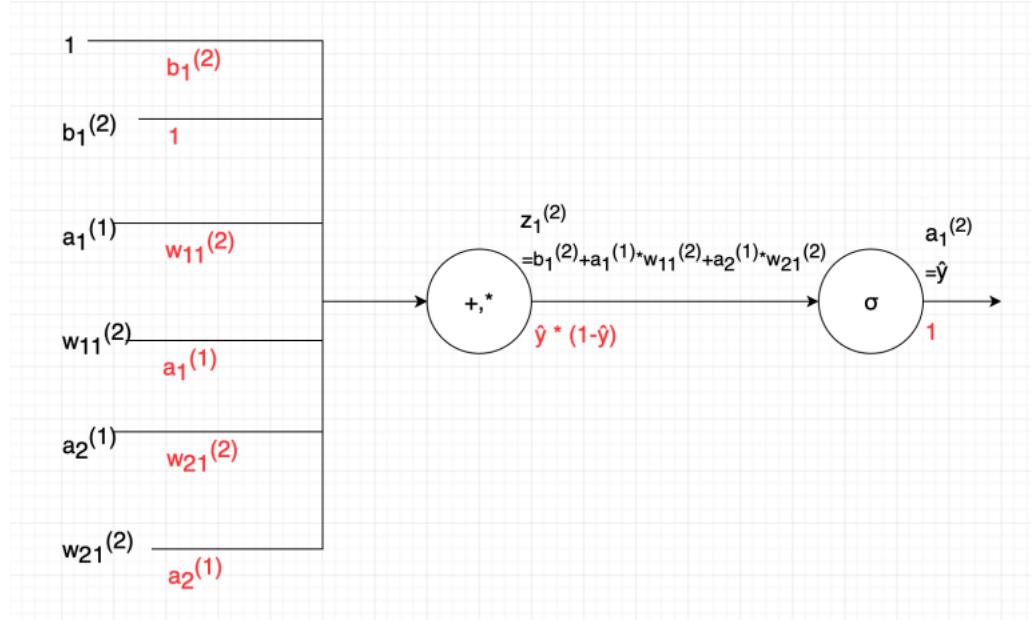
trong đó: $\hat{y}_i = a_1^{(2)} = \sigma(a_1^{(1)} * w_{11}^{(2)} + a_2^{(1)} * w_{21}^{(2)} + b_1^{(2)})$ là giá trị mà model dự đoán, còn y_i là giá trị thật của dữ liệu.

$$\frac{\partial L}{\partial \hat{y}_i} = -\frac{\partial(y_i * \log(\hat{y}_i) + (1 - y_i) * \log(1 - \hat{y}_i))}{\partial \hat{y}_i} = -\left(\frac{y_i}{\hat{y}_i} - \frac{1 - y_i}{1 - \hat{y}_i}\right)$$

Tính đạo hàm L với $W^{(2)}, b^{(2)}$

Áp dụng chain rule ta có:

$$\frac{\partial L}{\partial b_1^{(2)}} = \frac{dL}{d\hat{y}_i} * \frac{\partial \hat{y}_i}{\partial b_1^{(2)}}$$



Từ đó ta thấy:

$$\frac{\partial \hat{y}_i}{\partial b_1^{(2)}} = \hat{y}_i * (1 - \hat{y}_i)$$

$$\frac{\partial \hat{y}_i}{\partial w_{11}^{(2)}} = a_1^{(1)} * \hat{y}_i * (1 - \hat{y}_i)$$

$$\frac{\partial \hat{y}_i}{\partial w_{21}^{(2)}} = a_2^{(1)} * \hat{y}_i * (1 - \hat{y}_i)$$

$$\frac{\partial \hat{y}_i}{\partial a_1^{(1)}} = w_{11}^{(2)} * \hat{y}_i * (1 - \hat{y}_i)$$

$$\frac{\partial \hat{y}_i}{\partial a_2^{(1)}} = w_{21}^{(2)} * \hat{y}_i * (1 - \hat{y}_i)$$

Do đó:

$$\frac{\partial L}{\partial b_1^{(2)}} = \frac{\partial L}{\partial \hat{y}_i} * \frac{\partial \hat{y}_i}{\partial b_1^{(2)}} = -\left(\frac{y_i}{\hat{y}_i} - \frac{1 - y_i}{1 - \hat{y}_i}\right) * \hat{y}_i * (1 - \hat{y}_i) = -(y_i * (1 - \hat{y}_i) - (1 - y_i) * \hat{y}_i) = \hat{y}_i - y_i$$

Tương tự:

$$\frac{\partial L}{\partial w_{11}^{(2)}} = a_1^{(1)} * (\hat{y}_i - y_i)$$

$$\frac{\partial L}{\partial w_{21}^{(2)}} = a_2^{(1)} * (\hat{y}_i - y_i)$$

$$\frac{\partial L}{\partial a_1^{(1)}} = w_{11}^{(2)} * (\hat{y}_i - y_i)$$

$$\frac{\partial L}{\partial a_2^{(1)}} = w_{21}^{(2)} * (\hat{y}_i - y_i)$$

■ Biểu diễn dưới dạng ma trận:

*** Lưu ý: đạo hàm của L đối với ma trận W kích thước $m*n$ cũng là một ma trận cùng kích thước $m * n$.

$$\frac{\partial L}{\partial W} = \begin{bmatrix} \frac{\partial L}{\partial w_{11}} & \dots & \frac{\partial L}{\partial w_{1n}} \\ \frac{\partial L}{\partial w_{21}} & \dots & \frac{\partial L}{\partial w_{2n}} \\ \dots & \dots & \dots \\ \frac{\partial L}{\partial w_{m1}} & \dots & \frac{\partial L}{\partial w_{mn}} \end{bmatrix}$$

Do đó:

$$\frac{\partial J}{\partial W^{(2)}} = (A^{(1)})^T * (\hat{Y} - Y), \frac{\partial J}{\partial b^{(2)}} = (sum(\hat{Y} - Y))^T, \frac{\partial J}{\partial A^{(1)}} = (\hat{Y} - Y) * (W^{(2)})^T$$

là phép tính sum tính tổng các cột của ma trận.

$$W = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix} \Rightarrow sum(W) = (w_{11} + w_{21} + w_{31}, w_{12} + w_{22} + w_{32})$$

$$(sum(W))^T = \begin{bmatrix} w_{11} + w_{21} + w_{31} \\ w_{12} + w_{22} + w_{32} \end{bmatrix}$$

Vậy là đã tính xong đạo hàm của L với hệ số $W^{(2)}, b^{(2)}$. Giờ sẽ đi tính đạo hàm của L với hệ số $W^{(1)}, b^{(1)}$ để khi tính đạo hàm của hệ số và bias trong layer trước đây sẽ cần dùng đến.

Tính đạo hàm L với $W^{(1)}, b^{(1)}$ Do $a_1^{(1)} = \sigma(b_1^{(1)} + x_1 * w_{11}^{(1)} + x_2 * w_{21}^{(1)})$

Áp dụng chain rule ta có:

$$\frac{\partial L}{\partial b_1^{(1)}} = \frac{\partial L}{\partial a_1^{(1)}} * \frac{\partial a_1^{(1)}}{\partial b_1^{(1)}}$$

Ta có:

$$\frac{\partial a_1^{(1)}}{\partial b_1^{(1)}} = \frac{\partial a_1^{(1)}}{z_1^{(1)}} * \frac{z_1^{(1)}}{\partial b_1^{(1)}} = a_1^{(1)} * (1 - a_1^{(1)})$$

Do đó:

$$\frac{\partial L}{\partial b_1^{(1)}} = a_1^{(1)} * (1 - a_1^{(1)}) * w_{11}^{(2)} * (\hat{y}_i - y_i)$$

Tương tự:

$$\frac{\partial L}{\partial w_{11}^{(1)}} = x_1 * a_1^{(1)} * (1 - a_1^{(1)}) * w_{11}^{(2)} * (\hat{y}_i - y_i)$$

$$\frac{\partial L}{\partial w_{12}^{(1)}} = x_1 * a_2^{(1)} * (1 - a_2^{(1)}) * w_{11}^{(2)} * (\hat{y}_i - y_i)$$

$$\frac{\partial L}{\partial w_{21}^{(1)}} = x_2 * a_1^{(1)} * (1 - a_1^{(1)}) * w_{21}^{(2)} * (\hat{y}_i - y_i)$$

$$\frac{\partial L}{\partial w_{22}^{(1)}} = x_2 * a_2^{(1)} * (1 - a_2^{(1)}) * w_{21}^{(2)} * (\hat{y}_i - y_i)$$

Có thể tạm viết dưới dạng chain rule là:

$$\frac{\partial J}{\partial W^{(1)}} = \frac{\partial J}{\partial A^{(1)}} * \frac{\partial A^{(1)}}{\partial Z^{(1)}} * \frac{\partial Z^{(1)}}{\partial W^{(1)}}(1)$$

Từ trên đã tính được:

$$\frac{\partial J}{\partial A^{(1)}} = (\hat{Y} - Y) * (W^{(2)})^T$$

Đạo hàm của hàm sigmoid: $\frac{d\sigma(x)}{dx} = \sigma(x) * (1 - \sigma(x))$ và $A^{(1)} = \sigma(Z^{(1)})$, nên trong (1) có thể hiểu là $\frac{\partial A^{(1)}}{\partial Z^{(1)}} = A^{(1)} * (1 - A^{(1)})$

Cuối cùng, $Z^{(1)} = X * W^{(1)} + b^{(1)}$ nên có thể tạm hiểu $\frac{\partial Z^{(1)}}{\partial W^{(1)}} = X$, nó giống như $f(x) = a * x + b \Rightarrow \frac{df}{dx} = a$.

Kết hợp tất cả lại ta được:

$$\frac{\partial J}{\partial W^{(1)}} = X^T * (((\hat{Y} - Y) * (W^{(2)})^T) \otimes A^{(1)} \otimes (1 - A^{(1)}))$$

Vậy khi nào cần dùng element-wise (\otimes), khi nào dùng nhân ma trận (*)?

- Khi tính đạo hàm ngược lại qua bước activation thì dùng (\otimes).

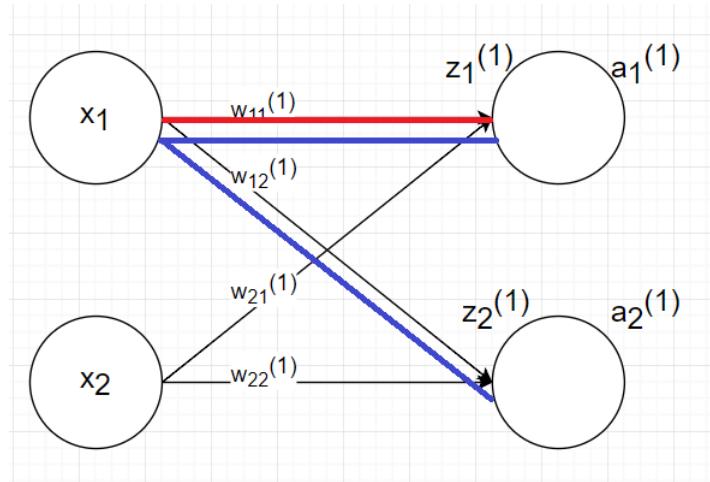
- Khi có phép tính nhân ma trận thì dùng (*), nhưng đặc biệt chú ý đến **kích thước ma trận** và dùng transpose nếu cần thiết. Ví dụ: ma trận X kích thước $N * 3$, W kích thước $3 * 4$, $Z = X * W$ sẽ có kích thước $N * 4$ thì $\frac{\partial J}{\partial W} = X^T * (\frac{\partial J}{\partial Z})$ và $\frac{\partial J}{\partial X} = (\frac{\partial J}{\partial Z}) * W^T$.

Tương tự:

$$\frac{\partial L}{\partial b^{(1)}} = \text{sum}(((\hat{Y} - Y) * (W^{(2)})^T) \otimes A^{(1)})^T$$

Vậy là đã tính xong hết đạo hàm của loss function với các hệ số W và bias b , giờ có thể áp dụng gradient descent để giải bài toán.

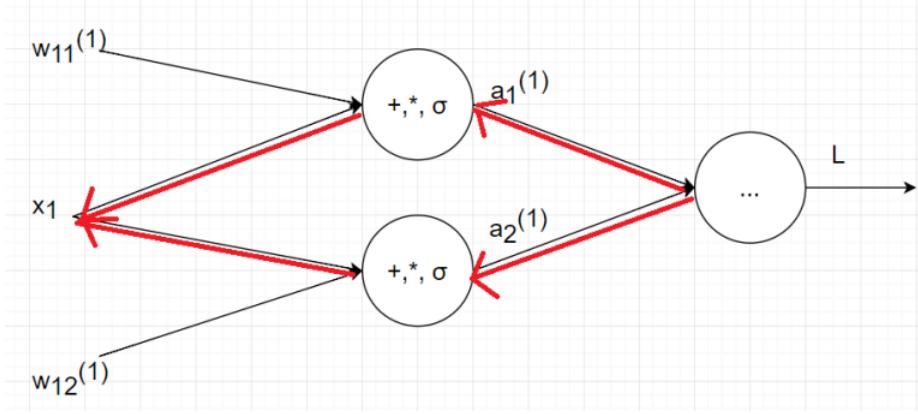
Giờ thử tính $\frac{\partial L}{\partial x_1}$, ở bài này thì không cần vì chỉ có 1 hidden layer, nhưng nếu nhiều hơn 1 hidden layer thì cần phải tính bước này để tính đạo hàm với các hệ số trước đó.



Hình 2.8: Đường màu đỏ cho $w_{11}^{(1)}$, đường màu xanh cho x_1

Ta thấy $w_{11}^{(1)}$ chỉ tác động đến $a_1^{(1)}$, cụ thể là $a_1^{(1)} = \sigma(b_1^{(1)} + x_1 * w_{11}^{(1)} + x_2 * w_{21}^{(1)})$

Tuy nhiên x_1 không những tác động đến $a_1^{(1)}$ mà còn tác động đến $a_2^{(1)}$, nên khi áp dụng chain rule tính đạo hàm của L với x_1 cần tính tổng đạo hàm qua cả $a_1^{(1)}$ và $a_2^{(1)}$.

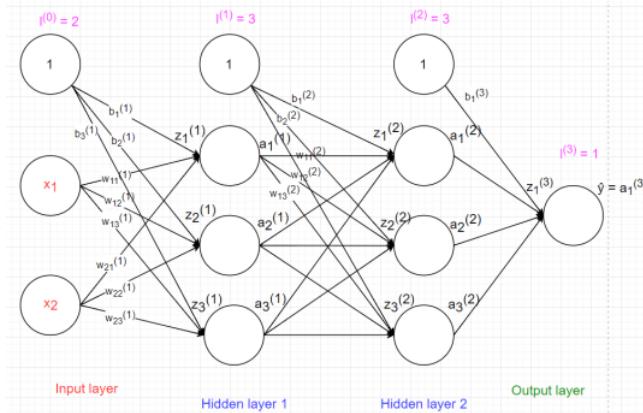


Hình 2.9: backpropagation tác động trong lớp ẩn

Do đó:

$$\frac{\partial L}{\partial x_1} = \frac{\partial L}{\partial a_1^{(1)}} * \frac{\partial a_1^{(1)}}{\partial x_1} + \frac{\partial L}{\partial a_2^{(1)}} * \frac{\partial a_2^{(1)}}{\partial x_1} = w_{11}^{(1)} * a_1^{(1)} * (1 - a_1^{(1)}) * w_{11}^{(2)} * (y_i - \hat{y}_i) + w_{12}^{(1)} * a_2^{(1)} * (1 - a_2^{(1)}) * w_{21}^{(2)} * (y_i - \hat{y}_i)$$

Sau tất cả, mô hình tổng quát sẽ bao gồm các bước như sau:



Hình 2.10: Mô hình neural network

- **Bước 1:** Tính $\frac{\partial J}{\partial \hat{Y}}$, trong đó $\hat{Y} = A^{(3)}$

- **Bước 2:** Tính

$$\frac{\partial J}{\partial \hat{W}^{(3)}} = (A^{(2)})^T * \left(\frac{\partial J}{\partial \hat{Y}} \otimes \frac{\partial A^{(3)}}{\partial Z^{(3)}} \right), \frac{\partial J}{\partial \hat{b}^{(3)}} = \left(\text{sum} \left(\frac{\partial J}{\partial \hat{Y}} \otimes \frac{\partial A^{(3)}}{\partial Z^{(3)}} \right) \right)^T$$

- **Bước 3:** Tính

$$\frac{\partial J}{\partial \hat{W}^{(2)}} = (A^{(1)})^T * \left(\frac{\partial J}{\partial A^{(2)}} \otimes \frac{\partial A^{(2)}}{\partial Z^{(2)}} \right), \frac{\partial J}{\partial \hat{b}^{(2)}} = \left(\text{sum} \left(\frac{\partial J}{\partial A^{(2)}} \otimes \frac{\partial A^{(2)}}{\partial Z^{(2)}} \right) \right)^T$$

và tính

$$\frac{\partial J}{\partial \hat{A}^{(1)}} = \left(\frac{\partial J}{\partial A^{(2)}} \otimes \frac{\partial A^{(2)}}{\partial Z^{(2)}} \right) * (W^{(2)})^T$$

- **Bước 4:** Tính

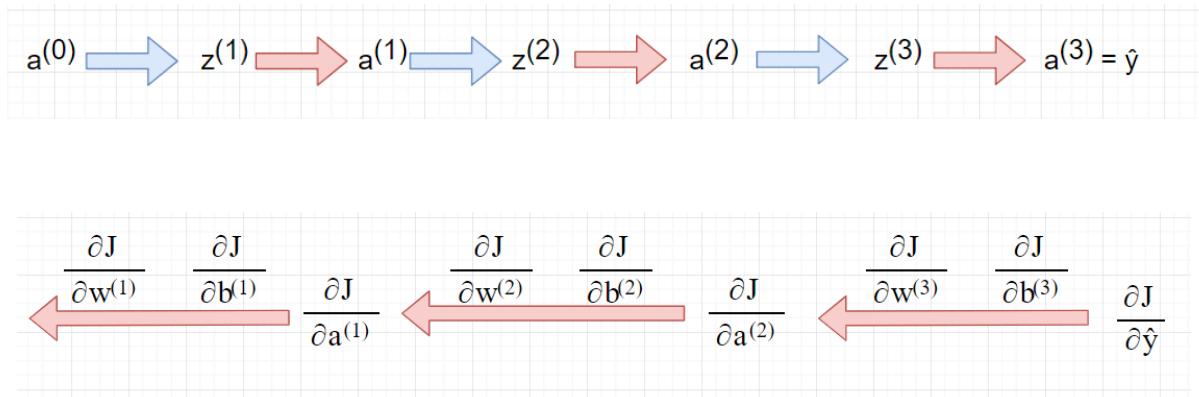
$$\frac{\partial J}{\partial \hat{W}^{(1)}} = (A^{(0)})^T * \left(\frac{\partial J}{\partial A^{(1)}} \otimes \frac{\partial A^{(1)}}{\partial Z^{(1)}} \right), \frac{\partial J}{\partial \hat{b}^{(1)}} = (\text{sum}(\frac{\partial J}{\partial A^{(1)}} \otimes \frac{\partial A^{(1)}}{\partial Z^{(1)}}))^T$$

, trong đó $A^{(0)} = X$

Nếu network có nhiều layer hơn thì cứ tiếp tục cho đến khi tính được đạo hàm của loss function J với tất cả các hệ số W và bias b .

Nếu hàm activation là sigmoid thì $\frac{\partial A^{(i)}}{\partial Z^{(i)}} = A^{(i)} \otimes (1 - A^{(i)})$

Tổng kết lại, 2 quá trình Feedforward và Backpropagation sẽ diễn ra lần lượt như sau:



Hình 2.11: Feedforward và Backpropagation

(Nội dung dựa theo "Sách Deep Learning cơ bản" - tác giả: Nguyễn Thanh Tuấn)

2.2 Mạng Convolutional Neural Network (CNN)

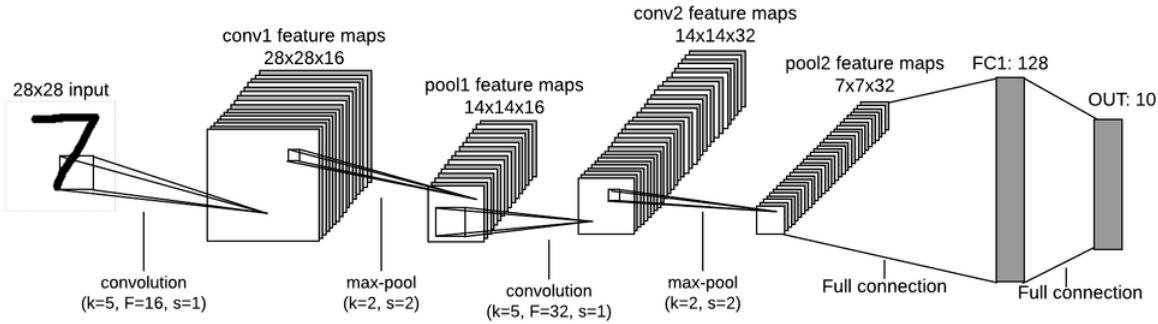
Convolutional Neural Network (CNNs – Mạng nơ-ron tích chập) là một trong những mô hình Deep Learning tiên tiến giúp cho chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao như hiện nay. Trong luận văn này, sẽ trình bày về Convolution (tích chập) đi từ những khái niệm cơ bản nhất đến ứng dụng của nó cũng như ý tưởng của mô hình CNNs trong phát hiện và trích xuất đặc trưng khung xương từ ảnh RGB.

Mạng Neural Network truyền thống tuy đã giải quyết được một số vấn đề lớn lúc bấy giờ nhưng lại gặp một số khó khăn khi giải quyết bài toán xử lý, phân loại hình ảnh. Đối với mạng Neural Network truyền thống khi xử lý ảnh màu $64*64$ được biểu diễn dưới dạng 1 tensor $64*64*3$. Việc để biểu thị hết nội dung của bức ảnh thì cần truyền vào input layer tất cả các pixel ($64*64*3 = 12288$). Nghĩa là input layer giờ có 12288 nodes. Giả sử số lượng node trong hidden layer 1 là 1000. Số lượng weight W giữa input layer và hidden layer 1 là $12288*1000 = 12288000$, số lượng bias là 1000 \Rightarrow tổng số parameter là: 12289000. Đây mới chỉ là số parameter giữa input layer và hidden layer 1, trong model còn nhiều layer nữa, và nếu kích thước ảnh tăng, ví dụ $512*512$ thì số lượng parameter tăng cực kì nhanh. Điều này khiến cho việc tính toán của máy tính cần rất nhiều công sức nhưng lại không mang lại hiệu quả cao. Do vậy ta cần có giải pháp tốt hơn.

Nhận xét:

- Trong ảnh các pixel ở cạnh nhau thường có liên kết với nhau hơn là những pixel ở xa. Ví dụ để thể hiện một vật thể trên ảnh cần các pixel gần nhau và có màu sắc tương tự nhau.
- Ngoài ra để so sánh các đối tượng là giống hay khác nhau cần phải so sánh giữa khu vực này với khu vực kia của bức ảnh. Do vậy cần phải có một bộ hệ số tính toán với các pixel quét hết toàn bộ bức ảnh để so sánh các vùng. Hay nói cách khác là các pixel ảnh chia sẻ hệ số với nhau.

Do vậy ý tưởng sử dụng mạng CNNs ra đời. Áp dụng phép tính convolution vào các layer trong neural network ta có thể giải quyết được vấn đề giảm thiểu lượng lớn parameter mà vẫn lấy ra được các đặc trưng của ảnh. Một mô hình mạng CNNs sẽ có cấu trúc chung gồm các lớp convolution và các lớp khác như pooling, fully connected, softmax,... (ví dụ mô hình mạng CNN - LeNet-5 trong hình 2.12).



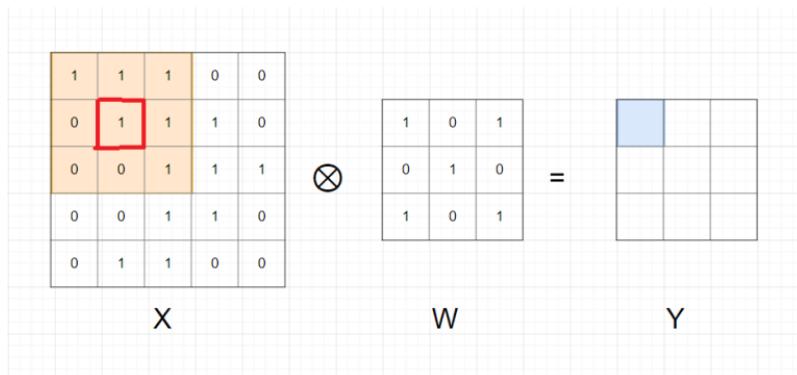
Hình 2.12: Ví dụ mô hình 1 mô hình convolutional neural network (LeNet-5 [1])

2.2.1 Phép Tính Convolution

Lấy ví dụ trên ảnh xám, ảnh được biểu diễn dưới dạng ma trận A kích thước $m * n$. Ta định nghĩa kernel là một ma trận vuông kích thước $k * k$ trong đó k là số lẻ. k có thể bằng 1, 3, 5, 7, 9, ... Ví dụ kernel kích thước $3 * 3$.

Kí hiệu phép tính convolution là (\otimes) , kí hiệu $Y = X \otimes W$.

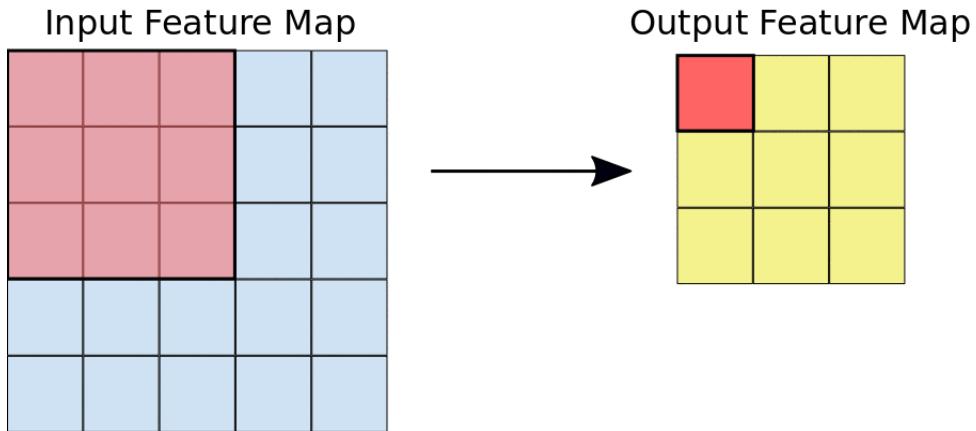
Với mỗi phần tử x_{ij} trong ma trận X lấy ra một ma trận có kích thước bằng kích thước của kernel W có phần tử x_{ij} làm trung tâm gọi là ma trận A . Sau đó tính tổng các phần tử của phép tính element-wise của ma trận A và ma trận W , rồi viết vào ma trận kết quả Y .



Hình 2.13: Phép tính Convolution

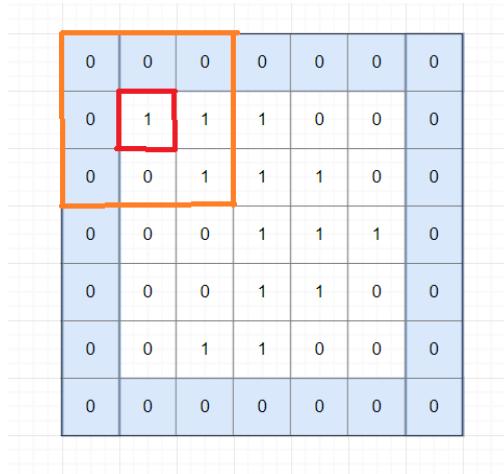
Ví dụ khi tính tại x_{22} (ô khoanh đỏ trong hình 2.13), ma trận A cùng kích thước với W , có x_{22} làm trung tâm có màu nền da cam như trong hình. Sau đó tính $y_{11} = \text{sum}(A \otimes W) = x_{11}*w_{11} + x_{12}*w_{12} + x_{13}*w_{13} + x_{21}*w_{21} + x_{22}*w_{22} + x_{23}*w_{23} + x_{31}*w_{31} + x_{32}*w_{32} + x_{33}*w_{33} = 4$. Và làm tương tự với các phần tử còn lại trong ma trận.

Vì tâm của kernel W không thể lướt hết ma trận X nên Y sẽ có kích thước nhỏ hơn ma trận X . Kích thước của ma trận Y là $(m-k+1) * (n-k+1)$.



Hình 2.14: Convolution feature map có kích thước nhỏ hơn ma trận ban đầu

■ **Padding** Mỗi lần thực hiện phép tính convolution xong thì kích thước ma trận Y đều nhỏ hơn X . Tuy nhiên nếu ta muốn ma trận Y thu được có kích thước bằng ma trận X ta cần tìm cách giải quyết cho các phần tử ở viền bằng cách thêm giá trị 0 ở viền ngoài ma trận X .



Hình 2.15: Ma trận X có viền 0 bên ngoài

Vậy ta đã giải quyết được vấn đề tìm A cho phần tử x_{11} , và ma trận Y thu được sẽ bằng kích thước ma trận X ban đầu.

Phép tính này gọi là convolution với $padding = 1$. $Padding = k$ nghĩa là thêm k vector 0 vào mỗi phía của ma trận.

■ **Stride** Như ở trên ta thực hiện tuần tự các phần tử trong ma trận X , thu được ma trận Y cùng kích thước ma trận X , ta gọi là $stride = 1$.

0	0	0	0	0	0	0
0	1	1	1	0	0	0
0	0	1	1	1	0	0
0	0	0	1	1	1	0
0	0	0	1	1	0	0
0	0	1	1	0	0	0
0	0	0	0	0	0	0

Hình 2.16: $stride = 1, padding = 1$

Tuy nhiên nếu $stride = k (k > 1)$ thì ta chỉ thực hiện phép tính convolution trên các phần tử $x_{1+i*k, 1+j*k}$. Ví dụ $k = 2$.

0	0	0	0	0	0	0
0	1	1	1	0	0	0
0	0	1	1	1	0	0
0	0	0	1	1	1	0
0	0	0	1	1	0	0
0	0	1	1	0	0	0
0	0	0	0	0	0	0

Hình 2.17: $padding = 1, stride = 2$

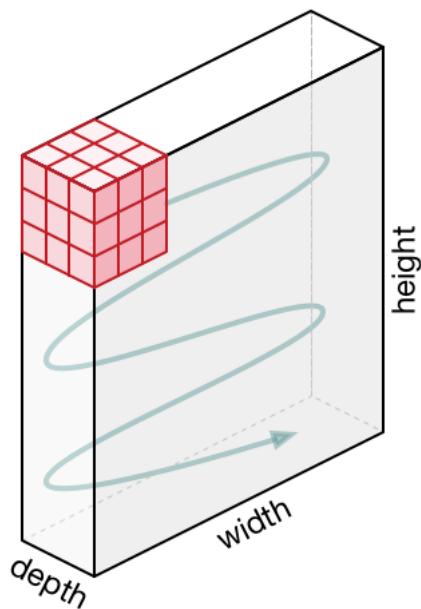
Bắt đầu từ vị trí x_{11} , sau đó cho kernel một lần nhảy k bước theo chiều dọc và ngang cho đến hết ma trận X . Ta thấy kích thước của ma trận Y là $3 * 3$ đã giảm đi đáng kể so với ma trận X . Công thức tổng quát cho phép tính convolution của ma trận X kích thước $m * n$ với kernel kích thước $k * k$, $stride = s$, $padding = p$ ra ma trận Y kích thước là

$$\left(\frac{m - k + 2p}{s} + 1 \right) * \left(\frac{n - k + 2p}{s} + 1 \right)$$

Stride thường dùng để giảm kích thước của ma trận sau phép tính convolution. Ý nghĩa của phép tính convolution: Mục đích của phép tính convolution trên ảnh là làm mờ, làm nét ảnh; xác định các đường;... Với mỗi kernel khác nhau ta sẽ phép tính được convolution có ý nghĩa khác nhau.

2.2.2 Phép convolution trong mạng Neuron Network

Với ảnh màu có tối 3 channels red, green, blue nên khi biểu diễn ảnh sẽ dưới dạng tensor 3 chiều. Vì vậy kernel cũng sẽ là 1 tensor 3 chiều kích thước $k * k * 3$.



Hình 2.18: Phép tính convolution trên ảnh màu với $k=3$.

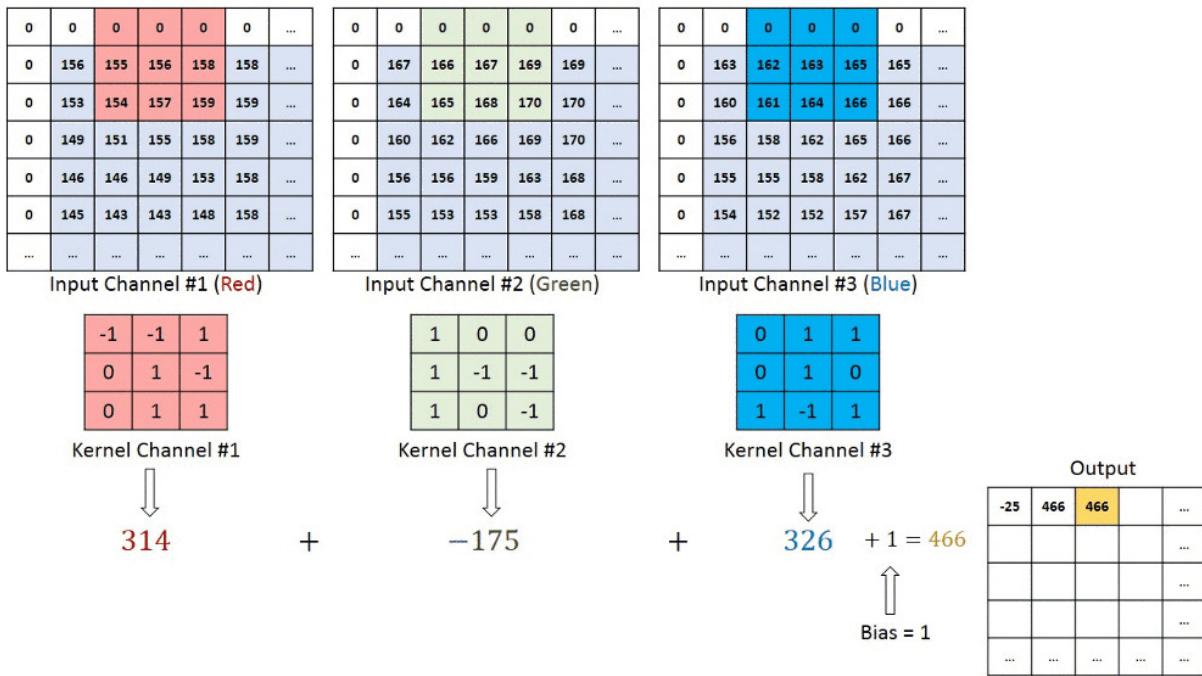
Ta định nghĩa kernel có cùng độ sâu (depth) với biểu diễn ảnh, rồi sau đó thực hiện di chuyển khối kernel tương tự như khi thực hiện trên ảnh xám.

$$\begin{array}{c}
 \begin{array}{|c|c|c|c|c|c|} \hline
 0 & 0 & 0 & 0 & 0 & 0 \\ \hline
 0 & 156 & 155 & 156 & 158 & 0 \\ \hline
 0 & 153 & 154 & 157 & 159 & 0 \\ \hline
 0 & 149 & 151 & 155 & 159 & 0 \\ \hline
 0 & 146 & 146 & 149 & 153 & 0 \\ \hline
 0 & 0 & 0 & 0 & 0 & 0 \\ \hline
 \end{array} \quad \begin{array}{|c|c|c|} \hline
 -1 & -1 & 1 \\ \hline
 0 & 1 & -1 \\ \hline
 0 & 1 & 1 \\ \hline
 \end{array} \\
 \otimes \quad = \quad \begin{array}{|c|c|c|} \hline
 \textcolor{purple}{\boxed{}} & & \\ \hline
 \end{array}
 \end{array} \\
 \begin{array}{c}
 \begin{array}{|c|c|c|c|c|c|} \hline
 0 & 0 & 0 & 0 & 0 & 0 \\ \hline
 0 & 167 & 166 & 167 & 158 & 0 \\ \hline
 0 & 164 & 165 & 168 & 159 & 0 \\ \hline
 0 & 160 & 162 & 166 & 159 & 0 \\ \hline
 0 & 146 & 146 & 149 & 153 & 0 \\ \hline
 0 & 0 & 0 & 0 & 0 & 0 \\ \hline
 \end{array} \quad \begin{array}{|c|c|c|} \hline
 1 & 0 & 0 \\ \hline
 1 & -1 & -1 \\ \hline
 1 & 0 & -1 \\ \hline
 \end{array} \\
 \times \quad \begin{array}{|c|c|c|} \hline
 0 & 1 & 1 \\ \hline
 0 & 1 & 0 \\ \hline
 1 & -1 & 1 \\ \hline
 \end{array} \quad \textbf{Y}
 \end{array}$$

Hình 2.19: Tensor X và W 3 chiều được viết dưới dạng 3 matrix.

Khi biểu diễn ma trận ta cần 2 chỉ số hàng và cột: i và j , thì khi biểu diễn ở dạng tensor 3 chiều cần thêm chỉ số độ sâu k . Nên chỉ số mỗi phần tử trong tensor là x_{ijk} .

$$\begin{aligned}
 y_{11} = b + & (x_{111} * w_{111} + x_{121} * w_{121} + x_{131} * w_{131} + x_{211} * w_{211} + x_{221} * w_{221} + x_{231} * w_{231} + x_{311} * w_{311} + \\
 & x_{321} * w_{321} + x_{331} * w_{331}) + (x_{112} * w_{112} + x_{122} * w_{122} + x_{132} * w_{132} + x_{212} * w_{212} + x_{222} * w_{222} + \\
 & x_{232} * w_{232} + x_{312} * w_{312} + x_{322} * w_{322} + x_{332} * w_{332}) + (x_{113} * w_{113} + x_{123} * w_{123} + x_{133} * w_{133} + x_{213} * w_{213} + \\
 & x_{223} * w_{223} + x_{233} * w_{233} + x_{313} * w_{313} + x_{323} * w_{323} + x_{333} * w_{333}) = -25
 \end{aligned}$$

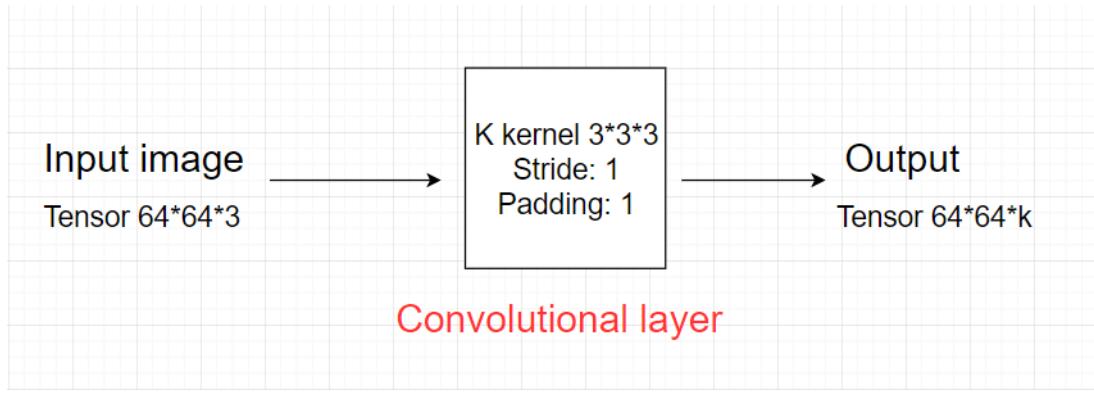


Hình 2.20: Thực hiện phép tính convolution trên ảnh màu

Nhận xét:

- Output Y của phép tính convolution trên ảnh màu là 1 matrix.
- Có 1 hệ số bias được cộng vào sau bước tính tổng các phần tử của phép tính element-wise.

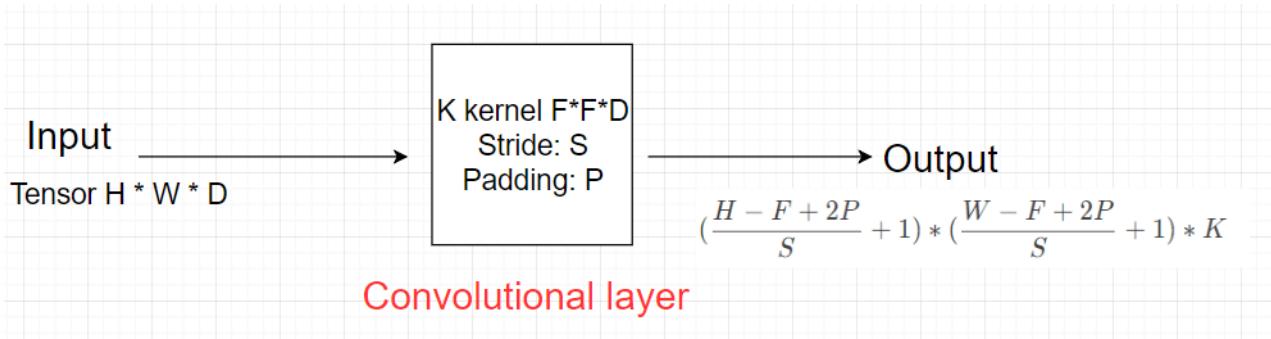
Với mỗi kernel khác nhau ta sẽ học được những đặc trưng khác nhau của ảnh, nên trong mỗi convolutional layer ta sẽ dùng nhiều kernel để học được nhiều thuộc tính của ảnh. Vì mỗi kernel cho ra output là 1 matrix nên k kernel sẽ cho ra k output matrix. Ta kết hợp k output matrix này lại thành 1 tensor 3 chiều có chiều sâu k. Output của convolutional layer đầu tiên sẽ thành input của convolutional layer tiếp theo.



Hình 2.21: Convolutional layer đầu tiên

■ Convolutional layer tổng quát:

Giả sử input của 1 convolutional layer tổng quát là tensor kích thước $H * W * D$. Kernel có kích thước $F * F * D$ (kernel luôn có depth bằng depth của input và F là số lẻ), stride: S , padding: P . Convolutional layer áp dụng K kernel. Lúc này, output của layer là tensor 3 chiều có kích thước: $(\frac{H-F+2P}{S} + 1) * (\frac{W-F+2P}{S} + 1) * K$ (hình 2.22)



Hình 2.22: Convolutional layer tổng quát

Ta nhận thấy:

- Output của convolutional layer sẽ qua hàm activation function trước khi trở thành input của convolutional layer tiếp theo.
- Tổng số parameter của layer: Mỗi kernel có kích thước $F * F * D$ và có 1 hệ số bias, nên tổng parameter của 1 kernel là $F * F * D + 1$. Mà convolutional layer áp dụng K kernel \Rightarrow Tổng số parameter trong layer này là $K * (F * F * D + 1)$.

Mạng Convolution Neural Network, ngoài các lớp Convolution ra còn có các lớp Pooling, Dropout, Dense, và Backnomalization,... để làm cho chúng trở nên "dễ học" hơn.

2.2.3 Batch-Norm

Việc chuẩn hóa (normalize) dữ liệu đầu vào giúp cho quá trình huấn luyện nhanh hơn (hàm chi phí hội tụ nhanh hơn) so với việc không chuẩn hóa. Nhiệm vụ của lớp chuẩn hóa cũng tương tự như vậy ở các lớp ẩn (hidden layer). Ngoài ra lớp chuẩn hóa còn giúp cho thuật toán có thể học được các dữ liệu có phân phối (distribution) khác nhau giúp cho mô hình tổng quan hơn.

Batch Normalization là một phương pháp hiệu quả khi training một mô hình mạng nơ ron. Mục tiêu của phương pháp này chính là việc muôn chuẩn hóa các feature (đầu ra của mỗi layer sau khi đi qua các activation) về trạng thái "zero-mean" với độ lệch chuẩn 1. Trong mạng CNN, hidden layer sử dụng phương pháp này gọi là lớp Batch-Norm. Ý tưởng của Batch Normalization là thêm một công đoạn (operation) ngay trước activation function cho mỗi layer. Operation đó có nhiệm vụ chuẩn hóa (normalizing) và zero-centering (mean subtracting) các inputs (mean của inputs sẽ là 0). Kết quả sau đó sẽ được scaling và shifting sử dụng hai parameters cho mỗi layer. Để thực hiện normalizing và zero-centering, Batch-Norm sẽ tính độ lệch chuẩn và phương sai của các inputs trên các mini-batches, sau đó sử dụng hai parameter là γ và β để thực hiện việc scaling.

Thuật toán chuẩn hóa cụ thể sẽ hoạt động như sau:

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_1 \dots m\}$;
 Parameters to be learned: γ, β

Output: $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{normalize}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // \text{scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Hình 2.23: Phép biến đổi back normalization [2]

■ Trong đó:

- μ_B là giá trị kỳ vọng thực nghiệm được tính toán trên mini-batch B.
- σ_B là độ lệch chuẩn thực nghiệm cũng được tính toán trên mini-batch B.
- m là số lượng instances trong mini-batch B.
- \hat{x}_i là giá trị của input thứ i trong mini-batch sau khi đã được chuẩn hóa và zero-centered.
- y là scaling factor hay scaling parameter của layer.
- β là shifting factor hay offset của layer.
- ϵ là smoothing factor dùng để tránh việc chia cho 0, thường có giá trị rất nhỏ.
- y_i là đầu ra của Batch Normalization operation.

Mỗi Batch-Norm layer sẽ có 4 parameter cần phải học: y (scale), β (shift), μ (mean) và σ (standard deviation).

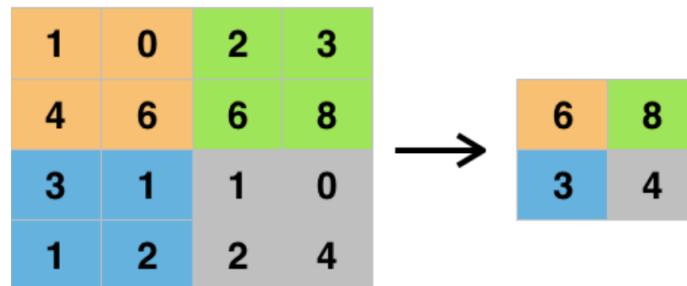
Một số lợi ích của Batch Normalization:

- Giảm thiểu đến mức tối thiểu hiện tượng Vanishing / Exploding gradients và chúng ta có thể quay lại sử dụng các activation function như sigmoid hay tanh.
- Giảm thiểu sự phụ thuộc vào quá trình weight initialization.
- Có thể sử dụng learning rate lớn hơn để tăng tốc quá trình training.
- Batch-Norm có thể được sử dụng như một regularize giúp giảm overfitting.

2.2.4 Pooling layer

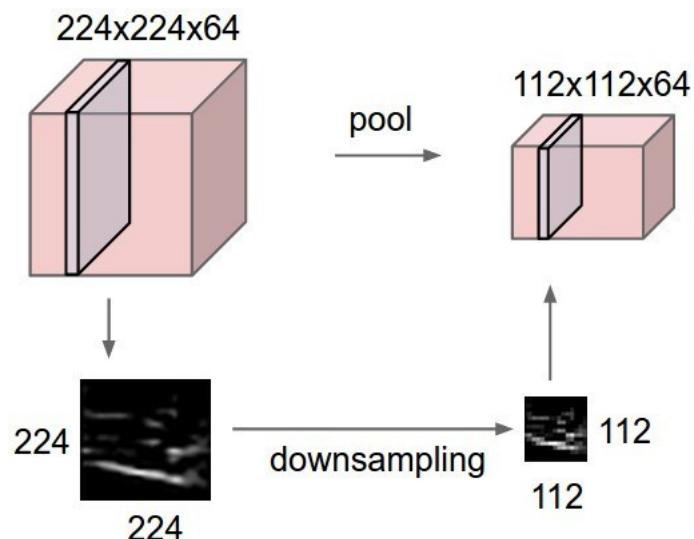
Pooling layer thường được dùng giữa các convolutional layer, để giảm kích thước dữ liệu nhưng vẫn giữ được các thuộc tính quan trọng. Kích thước dữ liệu giảm giúp giảm việc tính toán trong model.

Gọi pooling size kích thước $K * K$. Input của pooling layer có kích thước $H * W * D$, ta tách ra làm D ma trận kích thước $H * W$. Với mỗi ma trận, trên vùng kích thước $K * K$ trên ma trận ta tìm maximum hoặc average của dữ liệu rồi viết vào ma trận kết quả. Quy tắc về stride và padding áp dụng như phép tính convolution trên ảnh.



Hình 2.24: Phép Pooling với $stride = 2, padding = 0$

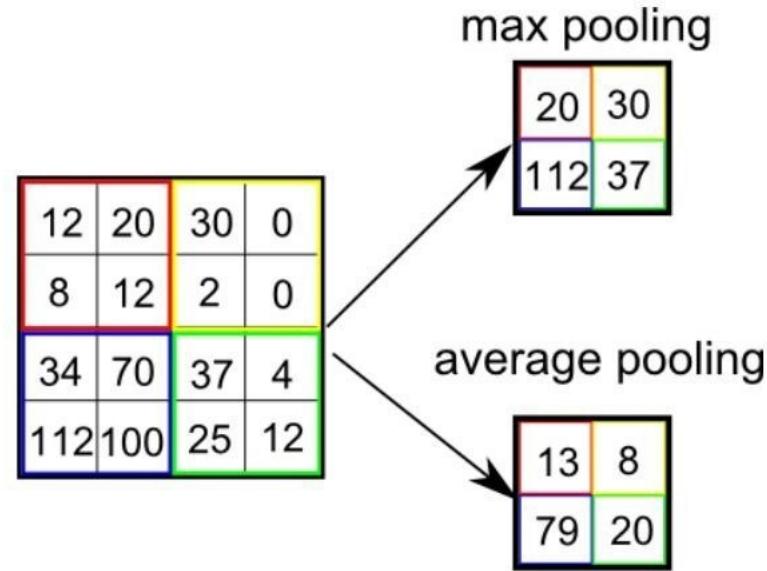
Nhưng hầu hết khi dùng pooling layer thì sẽ dùng $size = (2, 2)$, $stride = 2$, $padding = 0$. Khi đó output width và height của dữ liệu giảm đi một nửa, depth thì được giữ nguyên.



Hình 2.25: Kết quả sau khi qua pooling layer 2×2 .

(<http://cs231n.github.io/convolutional-networks>)

Có 2 loại pooling layer phổ biến là: max pooling và average pooling.

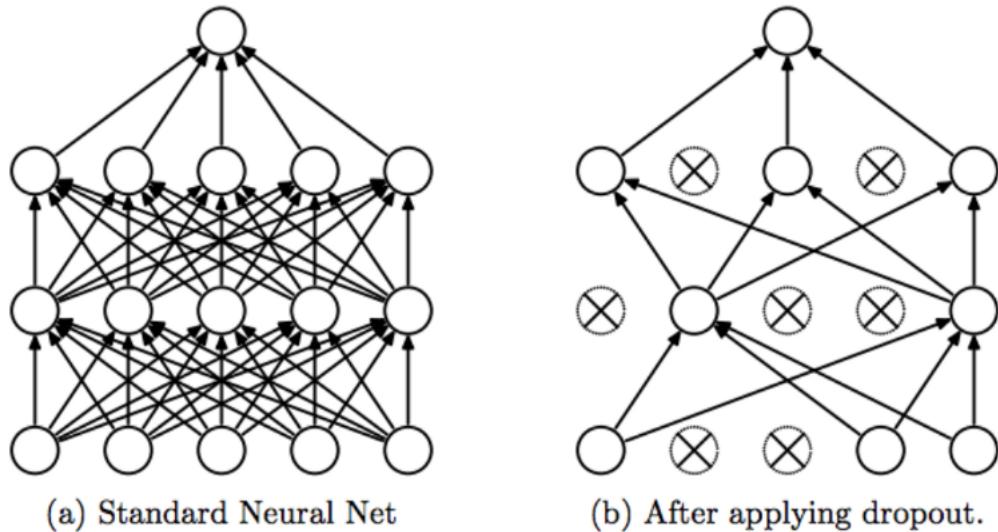


Hình 2.26: Max pooling và average pooling

Ngoài ra còn có thể dùng convolutional layer với stride > 1 để giảm kích thước dữ liệu thay cho pooling layer.

2.2.5 Drop-Out

Trong neural network, việc cuối cùng là tối ưu các tham số để làm cho giảm loss function, nhưng đôi khi có unit thay đổi theo cách sửa lại lỗi của các unit khác dẫn đến việc hòa trộn làm giảm tính dự đoán của model, hay còn gọi là overfitting. Dropout là cách thức mà chúng ta giả định một phần các unit bị ẩn đi trong quá trình training, qua đó làm giảm tích hòa trộn (hay nói cách khác là 1 hidden unit không thể dựa vào 1 unit khác để sửa lỗi làm của nó, để cho chúng ta thấy các hidden unit không đáng tin cậy). Tại mỗi step trong quá trình training, khi thực hiện Forward Propagation (Lan truyền xuôi) đến layer sử dụng Drop-Out, thay vì tính toán tất cả unit có trên layer, tại mỗi layer ta chọn ngẫu nhiên một số unit có được tính hay không theo tỉ lệ p .



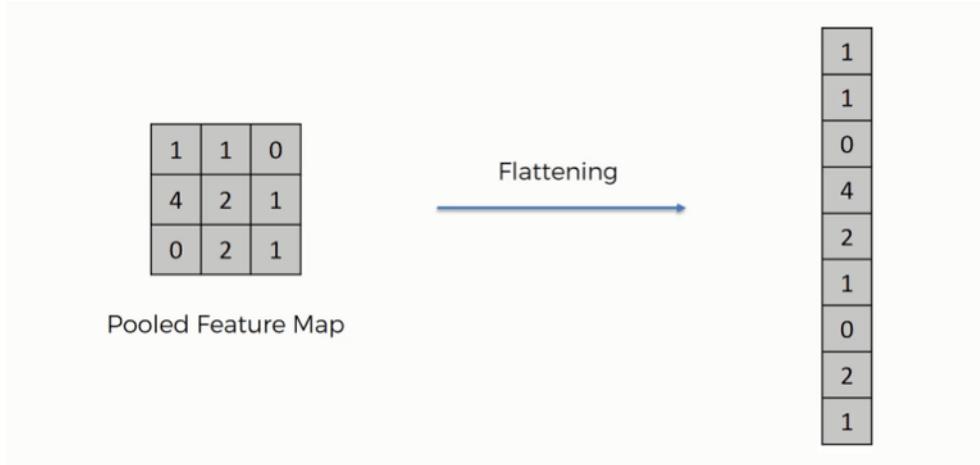
Hình 2.27: Minh họa phương pháp Drop-Out

■ Một số đặc điểm của Drop-Out:

- Dropout ép mạng neural phải tìm ra nhiều robust features hơn, với đặc điểm là chúng phải hữu ích hơn, tốt hơn, ngon hơn khi kết hợp với nhiều neuron khác.
- Dropout đòi hỏi phải gấp đôi quá trình huấn luyện để đạt được sự hội tụ. Tuy nhiên, thời gian huấn luyện cho mỗi epoch sẽ ít hơn.
- Với H unit trong mô hình, mỗi unit đều có xác suất bị bỏ qua hoặc được chọn, chúng ta sẽ có 2^H mô hình có thể có. Trong pha test, toàn bộ network được sử dụng và mỗi hàm activation được giảm đi với hệ số p .
- Một số nghiên cứu chỉ ra rằng, khi sử dụng Dropout và Batch Normalization cùng nhau thì kết quả rất tệ, trong cả lý thuyết và thực nghiệm [?].

2.2.6 Fully connected layer (Dense layer)

Sau khi ảnh được truyền qua nhiều convolutional layer và pooling layer thì model đã học được tương đối các đặc điểm của ảnh (ví dụ mắt, mũi, khung mặt,...) thì tensor của output của layer cuối cùng, kích thước $H * W * D$, sẽ được chuyển về 1 vector kích thước $(H * W * D)$.



Hình 2.28: Phép Flatten biến tensor về 1 vector

Sau đó, mỗi điểm của vector sẽ được liên kết với toàn bộ output của mode giống như 1 lớp của mạng Neural Network truyền thống. Và cuối cùng của mạng sẽ có nhiệm vụ phân loại theo như yêu cầu của từng bài toán. Thường sẽ sử dụng hàm softmax để tính đầu ra cho lớp này.

2.2.7 Hàm Softmax

Sau khi mạng CNNs được học qua các lớp phía trước, ta thu được một vector đặc trưng. Việc cần làm tiếp theo là dựa vào vector đặc trưng này để phân loại vào các lớp theo yêu cầu bài toán. Chúng ta cần một mô hình xác suất sao cho với mỗi input x , sẽ tính được output a_i thể hiện xác suất để input đó rơi vào lớp thứ i . Sử dụng hàm softmax ta tính được điều này. Hàm softmax nhận đầu vào là một vector và cho đầu ra là 1 vector có cùng số chiều $a(z) : \mathbb{R}^n \mapsto \mathbb{R}^n$. Công thức của hàm như sau:

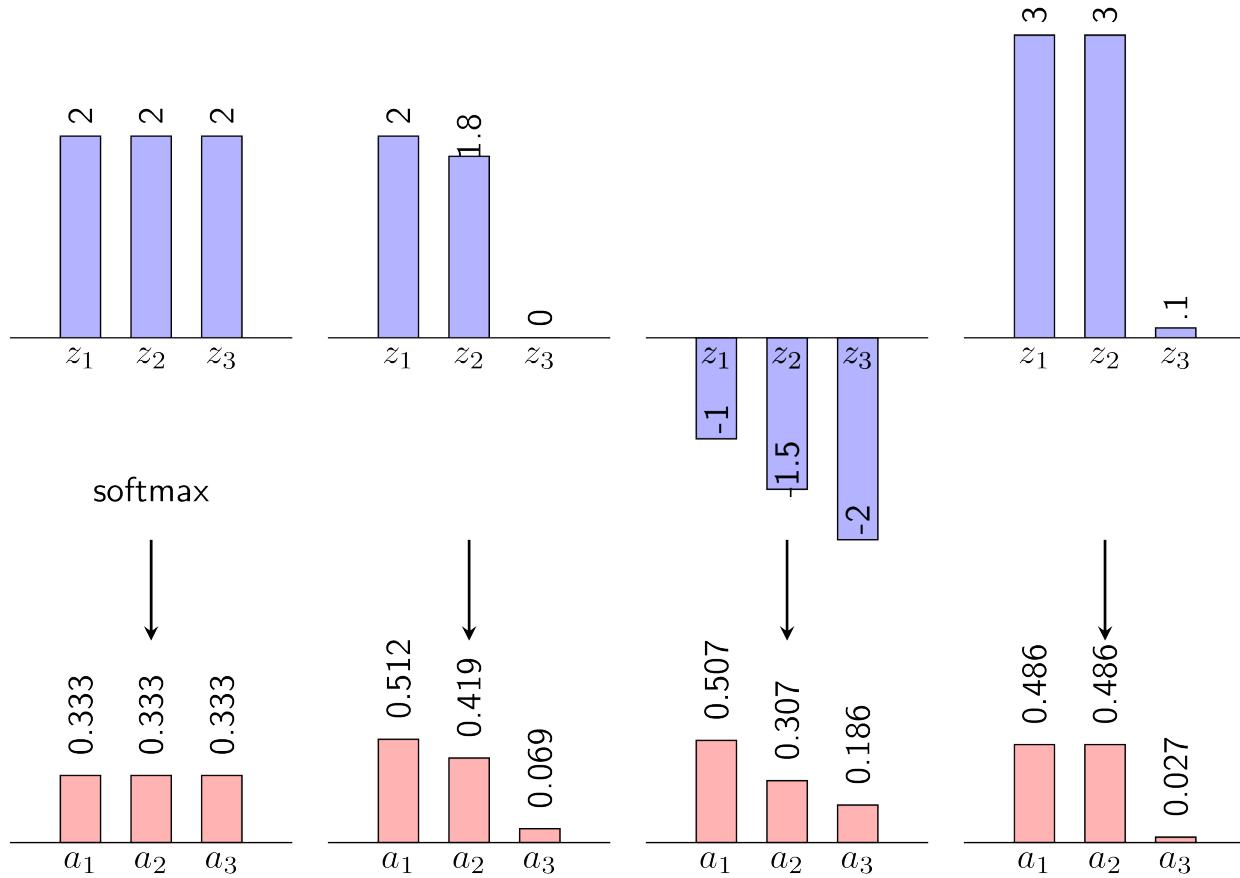
$$a_i = \frac{\exp(z_i)}{\sum_{j=1}^C \exp(z_j)}, \quad \forall i = 1, 2, \dots, C$$

■ Ở đây, ta có nhận xét:

- Giá trị $z_i = \mathbf{w}_i^T \mathbf{x}$ càng lớn thì xác suất dữ liệu rơi vào class i càng cao.
- Các a_i lớn hơn 0 và có tổng bằng 1.

Như vậy hàm softmax đã tính tất cả các a_i dựa vào tất cả các z_i , thỏa mãn tất cả các điều kiện: dương, tổng bằng 1, giữ được thứ tự của z_i . Giá trị xuất ra của hàm softmax tại mỗi class

thường được sử dụng làm độ tin cậy của class đó. Dưới đây là một vài ví dụ về mối quan hệ giữa đầu vào và đầu ra của hàm softmax (hình 2.29). Hàng trên màu xanh nhạt thể hiện các scores z_i với giả sử rằng số classes là 3. Hàng dưới màu đỏ nhạt thể hiện các giá trị đầu ra a_i của hàm softmax.



Hình 2.29: Một số ví dụ về đầu vào và đầu ra của hàm softmax

Nguồn: :<https://machinelearningcoban.com>

Softmax đặc biệt được sử dụng nhiều làm hàm kích hoạt ở lớp output trong bài toán phân loại qua mạng CNNs. Những lớp phía trước có thể được coi như một bộ Feature Extractor, lớp cuối cùng của DNN cho bài toán classification thường là Softmax Regression.

2.3 Kết luận

Mạng CNNs cùng với các thuật toán machine learning khác đã trở nên phổ biến và đóng vai trò quan trọng trong sự phát triển của trí tuệ nhân tạo nói chung và thị giác máy tính nói

riêng. Sau khi trình bày cơ bản về các lý thuyết của mạng NN trong chương này, chương tiếp theo luận văn xin trình bày về ứng dụng của mạng CNNs trong việc ước tính tọa độ hay rút trích đặc trưng khung xương mà luận văn áp dụng.

Chương 3

ƯỚC TÍNH TỌA ĐỘ KHUNG XUỐNG TỪ ẢNH RGB

Trong nội dung này, em sẽ trình bày hai phương pháp để xuất để trích xuất đặc trưng khung xương. Đó là trích xuất đặc trưng từ thiết bị Kinect và trích xuất đặc trưng từ mạng neuron network. Phương pháp trích xuất đặc trưng khung xương từ ảnh RGB qua mạng NN dựa trên mạng mobilenet v2 được áp dụng trong đề tài vì tính khả thi, có thể ứng dụng được trong đời sống cao hơn so với phương pháp sử dụng camera Kinect.

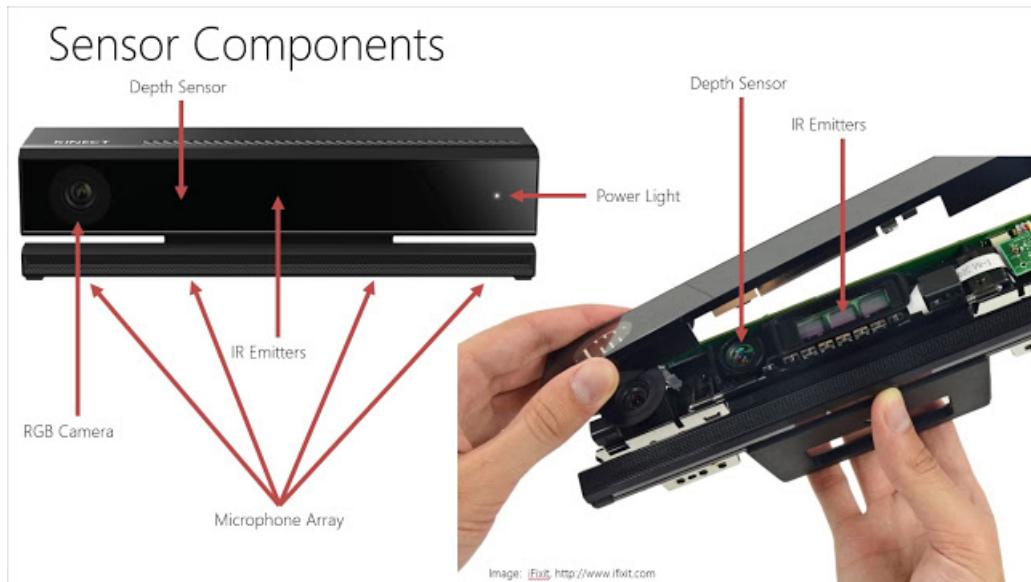
3.1 ƯỚC TÍNH DỰA TRÊN CAMERA ĐỘ SÂU KINECT

3.1.1 Giới thiệu camera cảm biến độ sâu Kinect của Microsoft

Kinect là một thiết bị đầu vào, là cảm biến chuyển động do hãng Microsoft sản xuất dành cho Xbox 360 và máy tính Windows. Dựa trên một webcam kiểu add-on ngoại vi cho Xbox 360, nó cho phép người dùng điều khiển và tương tác với Xbox 360 mà không cần phải dùng đến một bộ điều khiển tay cầm, thông qua một giao diện người dùng tự nhiên bằng cử chỉ và lệnh nói. Thiết bị được giới thiệu vào tháng 11 năm 2010 như một phụ kiện của Xbox 360. Cảm biến chiều sâu (depth sensor) được sử dụng trong Kinect được lấy từ việc trích xuất camera hồng ngoại.

Chức năng chính của Kinect là một công cụ để người dùng tương tác với Xbox 360 bằng cử chỉ và lệnh nói. Vì lý do này, các bộ cảm biến có khả năng thu thập dữ liệu ở độ phân giải

640x480 điểm ảnh. Với các dữ liệu chiều sâu, có thể lấy được "các vector đặc trưng mang hình dáng của khung xương con người(SJM)" của người đứng phía trước của cảm biến. Và với các SJM đó, nó có thể nhận biết được cử chỉ của người sử dụng.



Hình 3.1: Camera cảm biến độ sâu Kinect

(Nguồn : <https://www.ifixit.com>)

Các thông số cơ bản của cảm biến như sau:

- Ảnh màu : 1920x1080 @30Hz (15Hz ánh sáng yếu)
- Ảnh độ sâu : 512x424 @30Hz
- Tầm xa : 0.5 ~ 4.5m
- Góc nhìn (Dọc/Ngang) : 70 / 60 độ
- Số lượng SJM (phát hiện/theo dõi) : 6 / 6 SJMs
- Số lượng SJM-J : 25 SJM-Js
- Hệ điều hành : Windows 8/10
- Cổng tín hiệu : USB 3.0

Cơ chế hoạt động: Ban đầu, bộ phận phát tia hồng ngoại sẽ phát ra tia hồng ngoại trong vùng hoạt động của nó. Thông qua phản chiếu các tia hồng ngoại về camera hồng ngoại sẽ thu

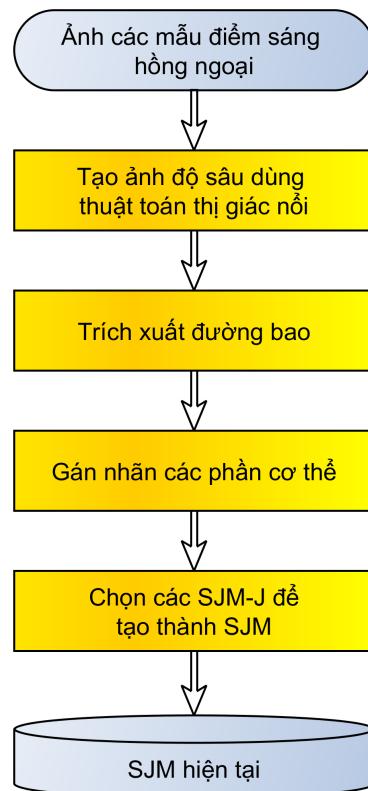
nhanh được các tia phản xạ về. Dựa vào thời gian trễ để đo khoảng cách tới các điểm trong vùng quan sát. Kết quả thu về sẽ là hình ảnh vùng quan sát với những chiều sâu khác nhau.



Hình 3.2: Ảnh độ sâu từ Kinect
(Nguồn : <https://www.zonettrigger.com/>)

3.1.2 Cơ chế trích xuất SJM từ Kinect

Ưu điểm của việc sử dụng Kinect là các thiết kế phần cứng đã tối ưu cho việc tạo ra SJM, không yêu cầu phải thực hiện lại hoặc tạo ra thuật toán mới cho phần trích xuất đặc trưng. Sơ đồ khái trich xuất SJM của Kinect v2 minh họa bằng hình 3.3.



Hình 3.3: Sơ đồ khái niệm thuật toán trích xuất SJM của thiết bị Kinect

(Nguồn : "Nhận dạng ngôn ngữ ký hiệu cho người câm" - B.V Phúc, H.C Thịnh)

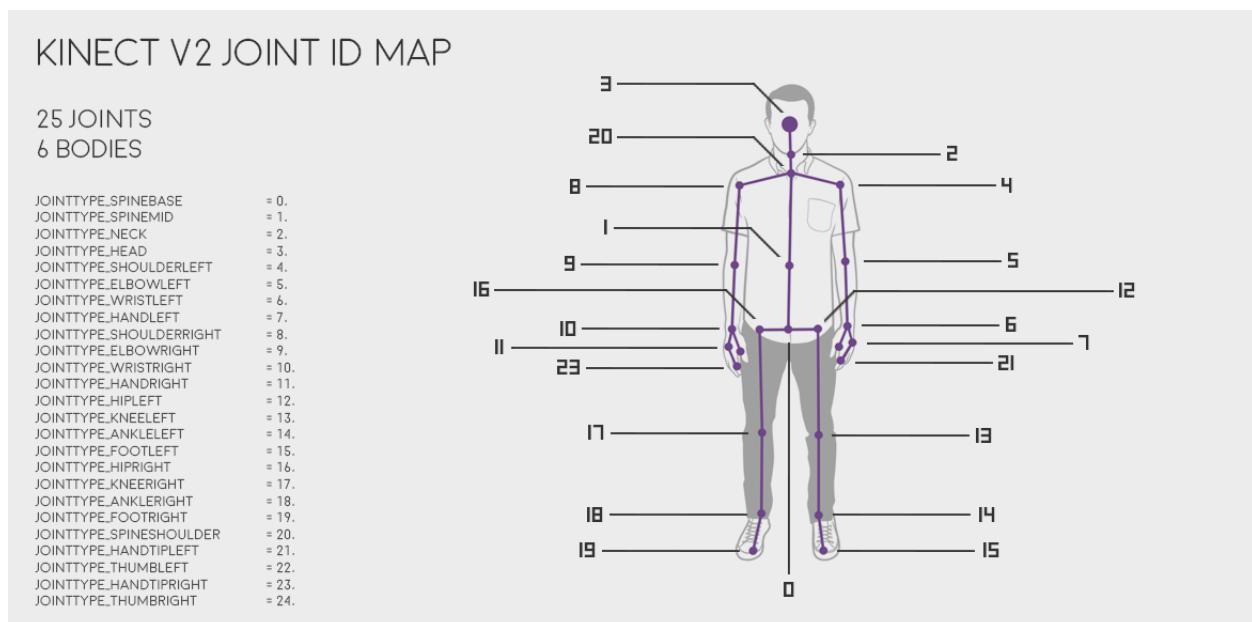
Dữ liệu ngõ vào

Hành động di chuyển của người dùng trong không gian nhận diện được của thiết bị Kinect v2.

Dữ liệu ngõ ra

- Ảnh màu kích thước 1920x1080 pxs.
- Ảnh biểu diễn độ sâu 512x424 pxs.
- SJM tương ứng với ảnh hiện tại biểu diễn tọa độ bằng meter (đo tương đối bằng ảnh độ sâu) hoặc px (tương ứng với vị trí của ảnh độ sâu).

Sau khi ước tính tọa độ khung xương bằng Kinect, ta thu được tọa độ các vị trí của 25 khớp xương trong không gian 3 chiều (hình 3.4)



Hình 3.4: Khung xương được trích xuất từ Kinect

(Nguồn : <https://www.zonetigger.com/>)

Việc ước tính tọa độ khung xương dựa trên camera Kinect mang nhiều ưu điểm như tốc độ ước tính nhanh do phần cứng đã tối ưu việc xử lý khớp xương. Kết quả ước tính trên không gian 3 chiều mang độ chính xác cao giúp cho các ứng dụng dựa trên ước tính tọa độ khung xương thực hiện tốt. Tuy nhiên luận văn áp dụng mô hình ước tính tọa độ khung xương dựa

vào mạng NN giúp áp dụng vào các thiết bị camera RGB bình thường dễ dàng hơn. Phần tiếp theo luận văn xin trình bày về mạng Neural network này.

3.2 ƯỚC TÍNH TỪ ẢNH 2D DỰA TRÊN MẠNG NEURAL NETWORK

3.2.1 Tổng quan phương pháp

Phương pháp trích xuất hình dáng khung xương được áp dụng từ bài báo "Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields" [16] sử dụng phương pháp bottom-up ước tính tọa độ khung xương từ ảnh sang không gian 2D. Tuy có rất nhiều mạng pose estimate cả 2D lẫn 3D và cả dense pose(ước tính toàn bộ hình dáng cơ thể người) nhưng phương pháp ước tính 2D được chọn vì tốc độ xử lý có thể đáp ứng realtime và hoạt động trên các thiết bị cấu hình thấp. Ngoài ra phương pháp này có thể ước tính tọa độ khung xương nhiều người cùng một lúc

Mạng sử dụng detect tư thế 2D song song của nhiều người trong một ảnh. Phương pháp sử dụng một đại diện không có thông số (PAFs) được tham khảo để học cách liên kết các bộ phận cơ thể với mỗi cá nhân trong ảnh. Mô hình mã hóa toàn bộ bối cảnh, cho phép một bước phân tích từ dưới lên trên (bước này có độ chính xác cao, realtime và thực hiện song song nhiều người). Mô hình được thiết kế để kết hợp tìm vị trí các phần và liên kết giữa chúng thông qua 2 nhánh của quá trình dự đoán chuỗi giống nhau. Phương pháp này đã đạt giải nhất tại cuộc thi COCO 2016 keypoints challenge và vượt trội hơn so với kết quả trước đó trong MPII Multi-Person benchmark về performance và sự hiệu quả.

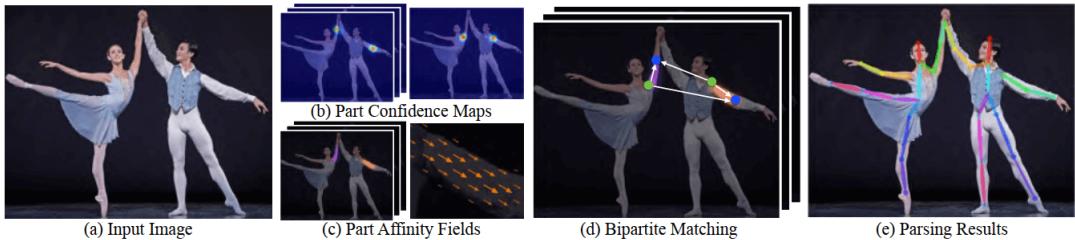


Fig. 2: Overall pipeline. (a) Our method takes the entire image as the input for a CNN to jointly predict (b) confidence maps for body part detection and (c) PAFs for part association. (d) The parsing step performs a set of bipartite matchings to associate body part candidates. (e) We finally assemble them into full body poses for all people in the image.

Hình 3.5: Tổng thể kiến trúc

Hình 3.5 minh họa toàn bộ nội dung phương pháp. Hệ thống lấy đầu vào, một ảnh màu có kích thước $w * h$ (hình 3.5a) và tạo ra ngõ ra, tọa độ của những keypoints cho mỗi cá nhân trong ảnh (hình 3.5e). Đầu tiên, một mạng CNN đồng thời dự đoán một loạt những confidence maps (cfm) S của những vị trí bộ phận cơ thể (hình 3.5b) và một loạt những miền vector 2D (vf) L của part affinities, cái mà mã hóa độ liên kết giữa các phần cơ thể (hình 3.5c). Tập hợp có J cfm, một map cho mỗi bộ phận, trong đó . Tập hợp có C vf, một cho mỗi chi, trong đó , mỗi vị trí ảnh trong Lc mã hóa một vector 2D (được show trong hình 1). Cuối cùng, cfm và affinity fields được phân tích bởi suy luận tham lam (hình 3.5d) để tạo ra các keypoints 2D cho tất cả những người trong ảnh.

Phương pháp này đưa toàn bộ ảnh đầu vào qua một mạng CNN 2 nhánh để đồng thời dự đoán những confidence map cho sự detect phần cơ thể, thể hiện trong hình 3.5b, và part affinity fields cho sự liên kết các phần, thể hiện trong hình 3.5c. Bước phân tích thể hiện một loạt những liên kết giữa hai điểm (liên kết lưỡng cực) để liên kết những phần cơ thể (3.5d). Cuối cùng, chúng được lắp ráp chúng lại với nhau tạo thành những tư thế cơ thể hoàn chỉnh cho tất cả những người trong ảnh (3.5e).

- Đầu tiên, hình ảnh được truyền qua mạng cơ sở để trích xuất các bản đồ đặc trưng. Trong bài báo, tác giả sử dụng 10 lớp đầu tiên của mô hình VGG-19. Tuy nhiên model được luận văn áp dụng sử dụng 10 lớp đầu tiên của mô hình mobilenet_v2 model này gọn nhẹ hơn so với VGG-19 nên có thể đáp ứng realtime đối với các máy tính cấu hình thấp.
- Sau đó, các bản đồ tính năng được xử lý với nhiều giai đoạn CNN để tạo: một bộ Bản đồ tin cậy một phần và một bộ các trường có mối quan hệ một phần (PAF)
 - **Confidence Maps**: một bộ bản đồ độ tin cậy 2D S cho các vị trí phần cơ thể. Mỗi

vị trí chung có một bản đồ.

- **Part Affinity Fields** : một tập hợp các trường vectơ 2D L mã hóa mức độ liên kết giữa các phần.
- Cuối cùng, **Confidence Maps** và **Part Affinity Fields** được xử lý bằng thuật toán tham lam để có được tư thế cho mỗi người trong ảnh.

3.2.2 Giải thích các phần

- **Confidence Maps (CFMs)** là một đại diện 2D cho niềm tin rằng một bộ phận cơ thể cụ thể có thể được đặt trong bất kỳ pixel nào. Với J là số lượng vị trí bộ phận cơ thể (khớp). Sau đó, **Confidence Maps** $S = (S_1, S_2, \dots, S_J)$ với $S_j \in R^{w \times h}, j \in (1 \dots J)$ Tóm lại, mỗi bản đồ tương ứng với một khớp và có cùng kích thước với hình ảnh đầu vào .
- **Part Affinity Fields(PAF)** Trường quan hệ một phần (**PAF**) là một tập hợp các trường dòng mã hóa các mối quan hệ cặp đôi không cấu trúc giữa các bộ phận cơ thể. Mỗi cặp bộ phận cơ thể có một **PAF**, tức là cổ, mũi, khuỷu tay, v.v.
Cho C là số lượng các cặp phần trên cơ thể. Sau đó **PAFs** là các thiết lập $L = (L_1, L_2, \dots, L_c)$ với $L_c \in R^{w \times h \times 2}, c \in (1 \dots C)$
Nếu một pixel nằm trên một chi (phần cơ thể), giá trị trong L_c tại pixel đó là một vectơ đơn vị 2D từ khớp bắt đầu đến khớp cuối.
- **CNN nhiều giai đoạn**

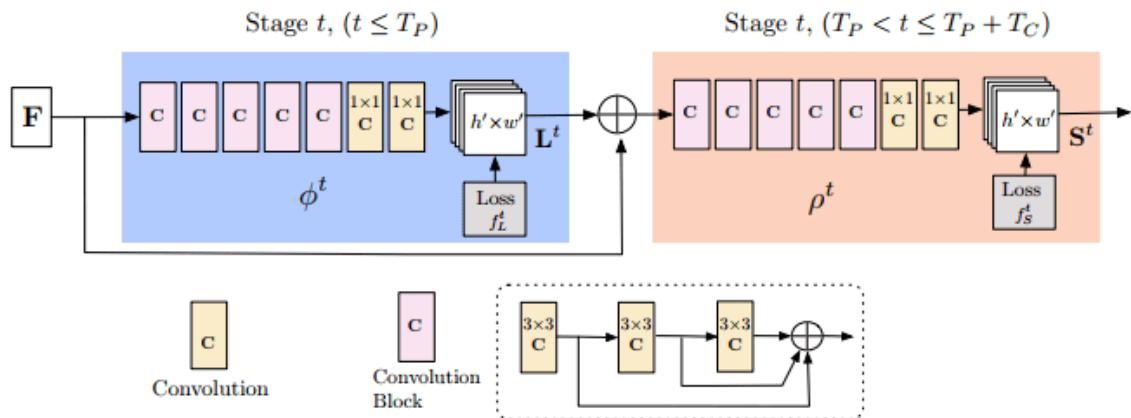


Fig. 3: Architecture of the multi-stage CNN. The first set of stages predicts PAFs L^t , while the last set predicts confidence maps S^t . The predictions of each stage and their corresponding image features are concatenated for each subsequent stage. Convolutions of kernel size 7 from the original approach [3] are replaced with 3 layers of convolutions of kernel 3 which are concatenated at their end.

Hình 3.6: Kiến trúc của CNN nhiều giai đoạn từ phiên bản tạp chí của OpenPose

CNN nhiều giai đoạn gồm các bước như sau:

- Tính toán các **part affinity fields (PAFs)**, L^1 từ feature maps của mạng cơ sở F . Cho ϕ^1 là mạng CNN mạng CNN tại bước 1.

$$L^1 = \phi^1(F)$$

- Giai đoạn t đến giai đoạn T_P : Tinh chỉnh dự đoán của **PAF** từ giai đoạn trước bằng cách sử dụng bản đồ tính năng F và các **PAF** trước đó (L^{t-1}). Với ϕ^t là CNN ở giai đoạn t .

$$L^t = \phi^t(F, L^{t-1}), \forall 2 \leq t \leq T_P$$

- Sau khi T_P lặp đi lặp lại, quá trình được lặp lại việc phát hiện **confidence maps**, bắt đầu trong dự đoán PAF cập nhật mới nhất . ρ^t là CNN ở giai đoạn t . Quá trình được lặp lại cho T_C .

$$S^{T_P} = \rho^t(F, L^{T_P}), \forall t = T_P$$

$$S^t = \rho^t(F, L^{T_P}, S^{t-1}), \forall T_P \leq t \leq T_P + T_C$$

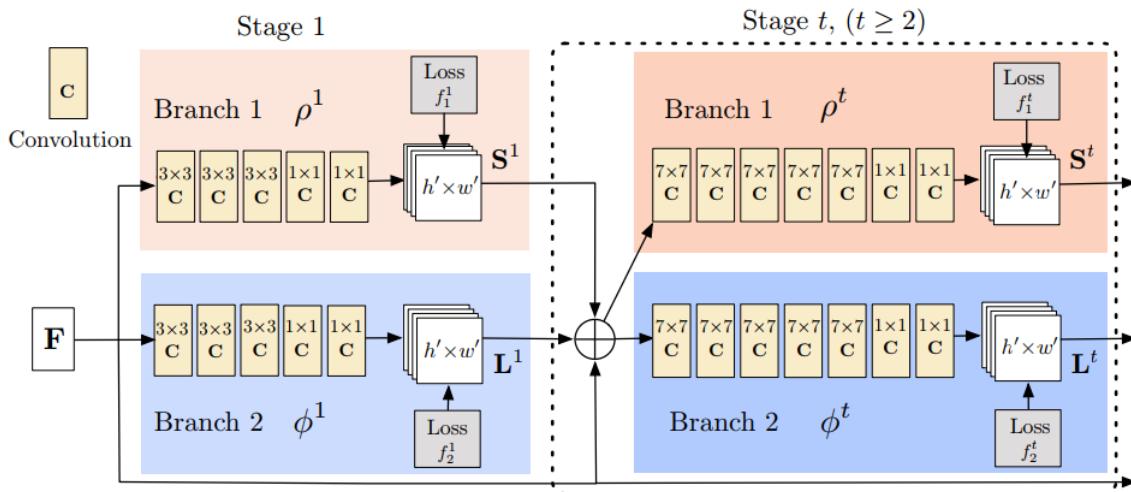
- Ma trận S và L cuối cùng là **Confidence Maps** và **part affinity fields (PAFs)** sẽ được xử lý thêm bằng thuật toán tham lam.

Chú thích: CNN nhiều giai đoạn này là từ phiên bản tạp chí 2018. Trong phiên bản CVPR 2017 gốc, họ đã tinh chỉnh cả bản đồ độ tin cậy và các trường mối quan hệ một phần (PAF) ở mỗi giai đoạn. Do đó, họ đòi hỏi nhiều tính toán và thời gian hơn ở mỗi giai đoạn. Trong cách tiếp cận mới, tác giả nhận thấy rằng cách tiếp cận mới làm tăng cả tốc độ và độ chính xác tương ứng 200% và 7%.

3.2.3 Kiến trúc mạng

3.2.3.1 Phát hiện và liên kết đồng thời

Kiến trúc của mạng, được show trong hình 3.7, dự đoán đồng thời những cfm và affinity fields (**af**) cái mà mã hóa liên kết giữa các phần. Mạng được tách thành 2 nhánh: Nhánh trên, màu hồng da, dự đoán những **cfm**, và nhánh dưới, màu xanh, dự đoán những **af**. Mỗi nhánh là một kiến trúc dự đoán lặp lại, theo như ??, nó tinh chỉnh những dự đoán qua các giai đoạn liên tiếp $t \in 1 \dots T$, với giám sát trung gian ở từng giai đoạn.



Hình 3.7: Kiến trúc của mạng CNN nhiều bước 2 nhánh. Mỗi bước trong nhánh đầu tiên dự đoán những $CFMS^t$, và mỗi bước trong nhánh thứ 2 dự đoán $PAFsL^t$. Sau mỗi bước, những dự đoán từ 2 nhánh, cùng với những đặc trưng ảnh, được nối lại cho bước tiếp theo.

3.2.4 Phân tích nhiều người sử dụng PAFs

Trong phần này, sẽ nói về tổng quan về thuật toán tham lam được sử dụng để phân tích tư thế của nhiều người từ PAFs và CFMs.

Quá trình phân tích được tóm tắt thành 3 bước sau:

- Bước 1: Tìm tất cả những vị trí khớp sử dụng confidence maps.
- Bước 2: Tìm những khớp cùng nhau tạo các chi (bộ phận cơ thể) sử dụng PAFs và những khớp trong bước 1.
- Bước 3: Kết nối các chi thuộc cùng một người và tạo danh sách những tư thế cơ thể.

3.2.4.1 Bước 1 - Tìm tất cả những vị trí khớp sử dụng confidence maps.

• Đầu vào:

- Confidence maps, $S = (S_1, S_2, \dots, S_J)$ trong đó $S_j \in R^{w \times h}, j \in (1 \dots J)$.
- Up-sampling scale: Sự khác biệt giữa dài/rộng giữa ảnh đầu vào và confidence maps.

• Đầu ra:

- Danh sách khớp(joints _ list): một danh sách những vị trí khớp của kích thước J, mỗi phần tử là một danh sách các đỉnh (x, y, xác suất).
- Ví dụ, kích thước của joints _ list là 18 cho 18 vị trí khớp (mũi, cổ,...) và những phần tử trong joints _ list là những danh sách có độ dài khác nhau. Những phần tử này lưu trữ thông tin đỉnh (vị trí x, y và điểm xác suất) cho mỗi vị trí khớp.

- **Xử lý: Cho mỗi khớp từ 1 đến J:**

- Lấy heatmap 2D tương ứng cho khớp trong confidence maps.
- Tìm những đỉnh bằng cách lấy ngưỡng heatmap 2D.
- Đối với mỗi đỉnh:
 - * Lấy một đốm xung quanh đỉnh trong heap.
 - * Phóng to đốm sử dụng up-sampling scale.
 - * Lấy vị trí đỉnh lớn nhất trong đốm được phóng to.
 - * Thêm thông tin đỉnh vào danh sách các đỉnh của khớp.

3.2.4.2 Bước 2 - Tìm những khớp cùng nhau tạo các chi (bộ phận cơ thể) sử dụng PAFs và những khớp trong bước 1.

- **Đầu vào:**

- joints-list ngõ ra của bước 1.
- **PAFs:** $L = (L_1, L_2, \dots, L_c)$ trong đó $L_c \in R^{w \times h \times 2}$, $c \in (1 \dots C)$
- Up-sampling scale: sự khác nhau giữa dài rộng của ảnh đầu vào và PAFs maps.
- Số lượng những điểm trung gian: số lượng những điểm trung gian giữa một khớp nguồn và những khớp đích để có được giá trị PAFs.

- **Đầu ra:**

- Những chi được kết nối (connected-limbs): một danh sách những chi được kết nối có kích thước C. Trong đó, mỗi phần tử là một danh sách tất cả các chi của loại đó.
- Mỗi thông tin chi chứa: id của chi nguồn, id của chi đích và điểm số thể hiện độ tốt của kết nối.

- **Xử lý:**

- Phóng to PAFs đến kích thước của ngõ vào sử dụng up-sampling scale.
- Đối với mỗi loại chi, ví dụ cổ tay-khuỷu tay trái:
 - * Lấy tất cả những đỉnh khớp nguồn và những đỉnh khớp đích, ví dụ: tất cả những đỉnh cổ tay trái và tất cả những đỉnh khuỷu tay trái.
 - * Nếu kích thước của những đỉnh nguồn hoặc đỉnh đích bằng 0, thì bỏ qua chi đó.
 - * Tạo một danh sách để lưu trữ tất cả những candidates kết nối chi.
 - * Đối với mỗi đỉnh nguồn và mỗi đỉnh đích:
 - Lấy vector trực tiếp bằng cách trừ những vị trí nguồn và vị trí đích.
 - Chuẩn hóa vector trực tiếp thành vector đơn vị.
 - Lấy giá trị PAFs tại mỗi điểm trung gian giữa những đỉnh nguồn và đỉnh đích.
 - Tính điểm số của kết nối chi hiện tại bằng cách lấy trung bình những giá trị PAFs.
 - Thêm điểm số để xác định khoảng cách chi phù hợp:
 $\min(0.5 * \text{paf_height} / \text{limb_dist} - 1, 0)$.
 - Thêm những kết nối chi hiện tại vào những candidates kết nối chi:
 - + Sắp xếp những candidates kết nối chi.
 - + Đối với mỗi candidate kết nối chi: Thêm sự kết nối vào danh sách cuối cùng nếu nguồn và đích không được chọn cho bất kỳ sự kết nối nào.

3.2.4.3 Bước 3 - Kết nối các chi thuộc cùng một người và tạo danh sách những tư thế cơ thể.

- **Đầu vào:**

- Danh sách khớp (**joint_list**) từ bước 1.
- Những chi được kết nối (**connected_limbs**) từ bước 2

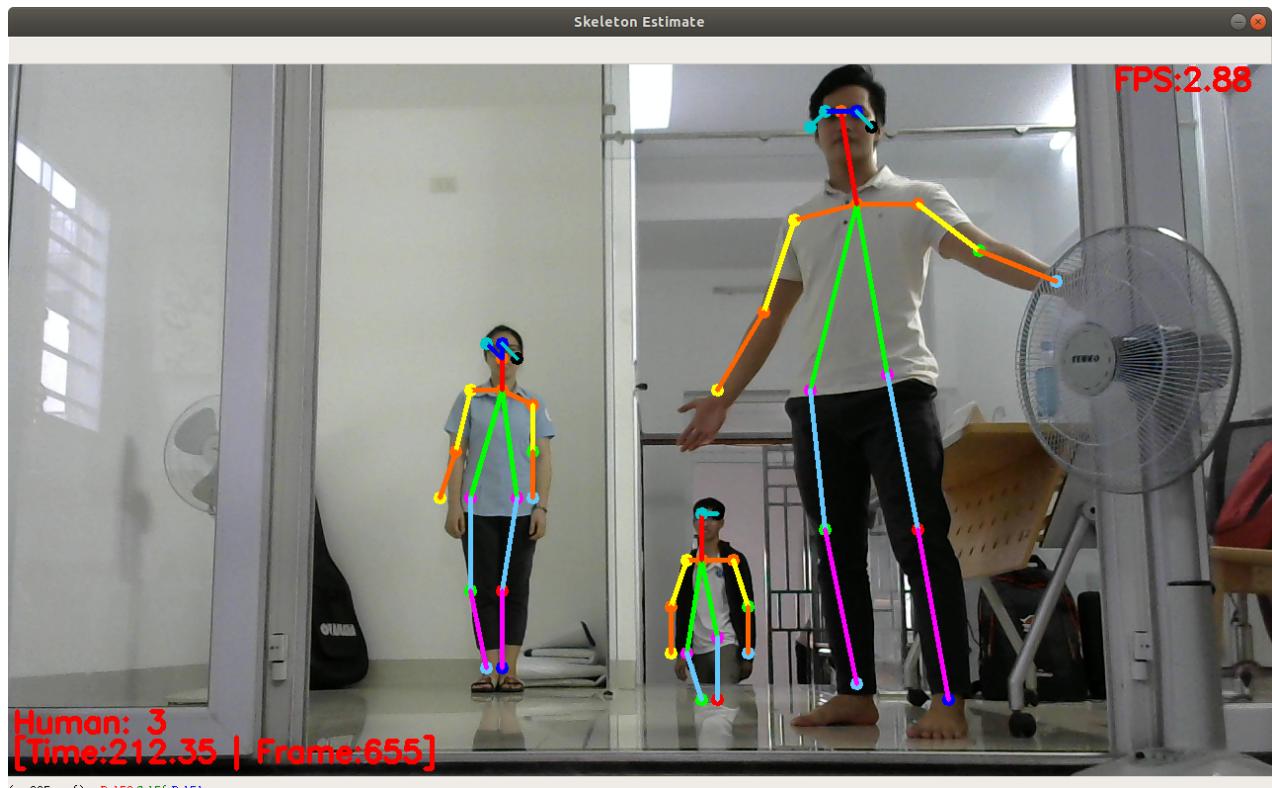
- **Đầu ra:** Những tư thế: một danh sách những tư thế người cho mỗi các nhân trong ảnh. Mỗi phần tử chứa những vị trí khớp cho người đó.

- **Xử lý:**

- Đối với mỗi loại chi và đối với mỗi kết nối giữa các chi (connected _ limbs) của loại đó:
 - * Tìm những cá nhân liên kết với khớp của kết nối hiện tại.
 - * Nếu không có người nào: Tạo một người mới với kết nối hiện tại.
 - * Nếu có một người: Thêm kết nối hiện tại vào người đó.
 - * Nếu có 2 người: Xác nhập 2 người này thành một người.
- Loại bỏ những người có quá ít khớp.

3.2.5 Kết quả

Ảnh RGB đầu vào chứa nhiều người sau khi xử lý qua mạng ta nhận được tọa độ 18 khớp xương của từng người được vẽ ra trên ảnh đầu ra. Kết quả ước lượng có thể đáp ứng được xử lý thời gian thực (realtime) trên máy tính cấu hình thấp với FPS đạt được khoảng 2.5 -> 4(FPS). Tốc độ nhận dạng tùy thuộc vào số người trong khung hình, tốc độ xử lý sẽ càng nhanh khi số lượng người trong khung hình càng ít và ngược lại.



Hình 3.8: Kết quả ước tính tọa độ khung xương từ mạng NN

3.3 Kết luận

Ước lượng khung xương hay ước tính tư thế con người là một hướng nghiên cứu quan trọng và đầy thách thức của thị giác máy tính. Phương pháp này được khá nhiều nghiên cứu áp dụng để giải quyết nhiều bài toán khó như nhận dạng dáng đi, tư thế con người, nhận dạng hành động hay giúp robot học theo chuyển động của con người... Chương tiếp theo, luận văn sẽ trình bày về ứng dụng của ước lượng tọa độ khung xương trong nhận dạng ngôn ngữ ký hiệu, là phần nghiên cứu chính của luận văn.

Chương 4

XÂY DỰNG MẠNG NEURON NHẬN DẠNG NGÔN NGỮ KÝ HIỆU

Trong chương 3, thuật toán ước tính tư thế khung xương đã được trình bày chi tiết về lý thuyết và phương pháp xử lý áp dụng thuật toán. Ta thấy rằng phương pháp này hỗ trợ rất tốt cho việc ước tính nhau tư thế khung xương trong xử lý thời gian thực. Trong chương 4 này, luận văn sẽ trình bày về đề xuất mô hình mạng DNN lấy đầu vào là tọa độ khung xương được ước tính và nhận dạng cử chỉ thời gian thực. Nội dung chương trình bao gồm cấu trúc mạng neural network đề xuất, cách thu thập và xử lý dữ liệu cũng như sơ đồ hoạt động chương trình.

4.1 Tổng quan

Một hành động của con người được đánh giá, xem xét bằng một loạt các cử chỉ theo thời gian. Tuy nhiên, khi xem xét các hành động của con người nhằm tìm ra một cấu trúc nhất quán và có thể tạo thành mô hình thì gặp các vấn đề phức tạp sau:

- Nếu chỉ xem xét đường bao của con người, các phần cơ thể của con người quá gần nhau để có thể xác định được chính xác phần cơ thể cần thiết.
- Hình dạng của cử chỉ (đường bao con người, màu sắc các phần cơ thể), vị trí, loại và kiểu của cử chỉ rất phức tạp. Việc xem xét một mô hình có thể biểu diễn toàn bộ các hành động ngôn ngữ ký hiệu hầu như không thể xem xét nêu trong phạm vi luận văn này chỉ xem xét đến việc một mô hình có thể biểu diễn 16 cử chỉ cần sự phối hợp cả hai tay và các cử chỉ tương

đối khác nhau (**Xin chào, Tôi, thành phố, vui vẻ, ấm em, Sài Gòn, Vĩnh Long, đi bộ, mùa màng, đói bụng, yêu, ăn, biểu quyết, đứng yên, hép, rộng**).

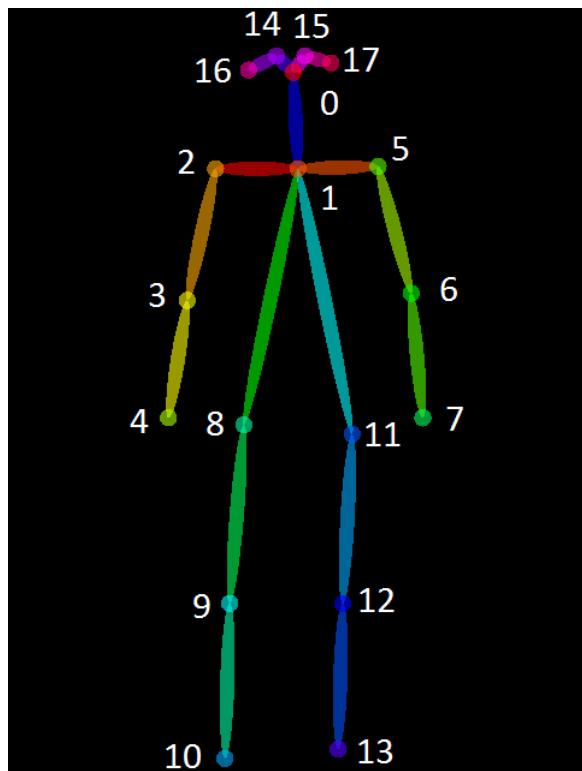
- Các hành động của con người có thể giống nhau, tuy nhiên nếu việc quan sát hoặc camera quan sát nằm ở vị trí khác nhau, hướng, độ cao,... đều ảnh hưởng đến khả năng nhận diện hành động của con người. Luận văn đã nêu được phương pháp để có thể phát triển cho việc xác định hành động của con người khi vị trí của camera thay đổi. Tuy nhiên việc kiểm chứng khả năng hoạt động ở các vị trí camera khác nhau sẽ được xem xét ở tương lai.
- Nếu một phần cơ thể bị che khuất bởi các vật thể thì việc xác định hành động của con người sẽ gặp khó khăn hơn rất nhiều so với trường hợp không bị che khuất, phạm vi luận văn không xem xét đến vấn đề này. Tuy nhiên đây là một điểm cần xem xét đến để có thể hoàn thiện hệ thống nhận diện cử chỉ trong tương lai.

4.2 Thu thập dữ liệu

Dữ liệu đầu vào là tọa độ của 18 khớp xương được detect từ mạng mobilenet. Các khớp xương được xuất ra từ mạng được đánh số thứ tự từ 0 tới 17. Các khớp xương cụ thể được thể hiện trong bảng 4.1 và trong hình 4.4.

Bảng 4.1: Các khớp xương được xuất ra từ mạng

Số thứ tự khớp xương	Vị trí
0	Mũi
1	Cổ
2	Vai phải
3	Khủy tay phải
4	Cổ tay phải
5	Vai trái
6	Khủy trái
7	Cổ tay trái
8	Hông phải
9	Dầu gối phải
10	Cổ chân phải
11	Hông trái
12	Dầu gối trái
13	Cổ chân trái
14	Mắt phải
15	Mắt trái
16	Tai phải
17	Tai trái

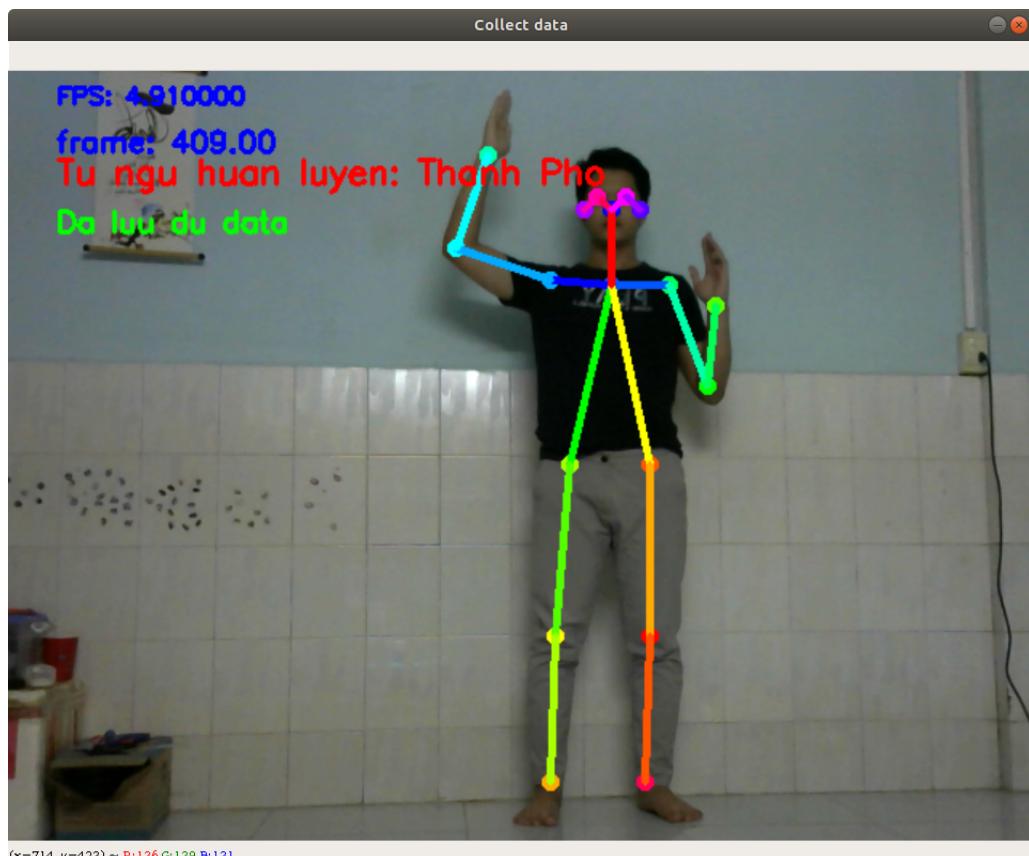


Hình 4.1: Sơ đồ khớp xương xuất ra từ mạng mobilenet

Dầu tiên người thu thập dữ liệu đứng trước camera, thực hiện hành động thể hiện từ cần huấn luyện. Chương trình thu thập dữ liệu sẽ xuất ra tọa độ khung xương sau khi xử lý qua mạng pose estimate. Sau đó lưu các tọa độ khớp xương cùng với nhãn của hành động vào file "data.csv" để sau đó đưa vào train. Hình 4.2 và 4.3 thể hiện quá trình khi thu thập dữ liệu.



Hình 4.2: Giao diện thu thập dữ liệu (khi bắt đầu ghi dữ liệu)



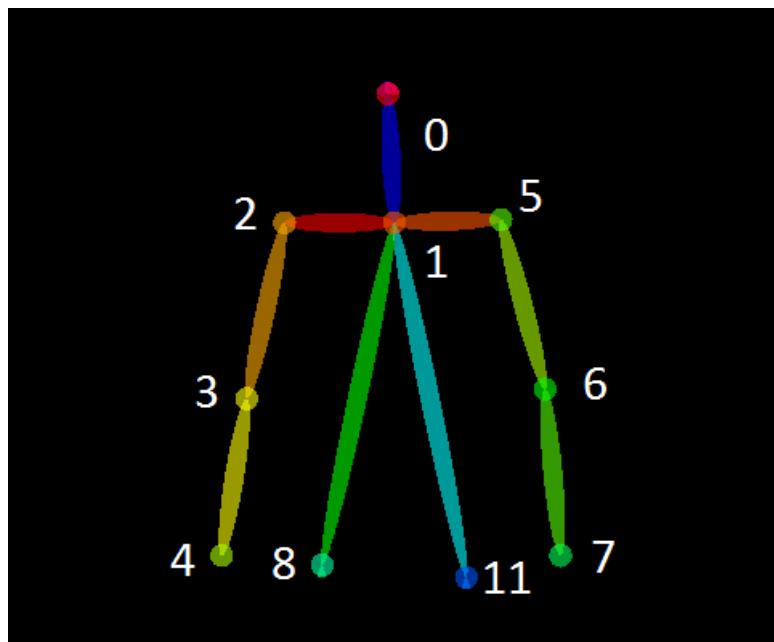
Hình 4.3: Giao diện thu thập dữ liệu (khi đã thu thập đủ dữ liệu)

4.3 Xử lý dữ liệu đầu vào

Dữ liệu sau khi thu thập ban đầu bao gồm tất cả tọa độ các khớp xương ứng các hành động cần nhận dạng. Tuy nhiên khi huấn luyện bộ nhận diện ngôn ngữ ký hiệu ta cần phải xử lý lại dữ liệu khớp xương đã thu thập được. Quá trình xử lý trải qua 2 phần: Loại bỏ các phần khớp xương dư thừa, sau đó chuẩn hóa lại các vector SJM.

4.3.1 Loại bỏ các phần SJM dư thừa

Ngôn ngữ ký hiệu với đặc trưng là sự phối hợp phần trên cơ thể với hai tay để diễn đạt từ ngữ mong muốn. Do đó việc xem xét đến các khớp xương phần dưới cơ thể là không cần thiết. Phần xử lý này, đề tài đã loại bỏ các tọa độ SJM dư thừa, chỉ giữ lại 10 điểm khớp xương cần thiết cho việc nhận diện: **Mũi**, **Cổ**, **Vai trái**, **khuỷu tay trái**, **Cổ tay trái**, **Vai phải**, **Khuỷu tay phải**, **Cổ tay phải**, **hông phải** và **hông trái** như thể hiện trong hình 4.4 và bảng 4.2



Hình 4.4: Sơ đồ khớp xương sau khi loại bỏ các phần không cần thiết

Bảng 4.2: Các khớp xương được giữ lại

Số thứ tự khớp xương	Vị trí
0	Mũi
1	Cổ
2	Vai phải
3	Khủyu tay phải
4	Cổ tay phải
5	Vai trái
6	Khủyu tay trái
7	Cổ tay trái
8	Hông phải
11	Hông trái

4.3.2 Chuẩn hóa SJM để phân loại đặc trưng

Dựa vào dữ liệu thu thập được, ta có thể nhận thấy là tùy vào vị trí camera, khoảng cách từ camera đến người được quan sát sẽ làm thay đổi giá trị của các vector khung xương SJM. Hình 3.8 từ kết quả chương 3 có thể thấy khoảng cách các SJM của người ở xa camera sẽ gần nhau hơn so với người ở gần. Do đó cần phải sử dụng phương pháp chuẩn hóa dữ liệu các vector, đưa các vector đặc trưng SJM về cùng kích cỡ thì dữ liệu đưa vào train mới có thể có cùng những đặc trưng giống nhau.

Phương pháp chuẩn hóa được áp dụng như sau:

Đầu tiên, ta sử dụng các dữ liệu ban đầu - raw data cho việc gán đặc trưng. Xem xét bảng vị trí của các SJM-J trong một SJM đặc trưng từ bảng 4.3.

Các vector của SJM có thể được biểu diễn:

$$v_n = (x_n; y_n; z_n) \quad (4.1)$$

Trong đó: $(x_n; y_n; z_n)$: tọa độ các SJM-J đang xét tới.

Với các tính toán vector như trên, từ 7 SJM-J có được 6 vectors biểu diễn SJM. Vấn đề xảy ra khi sử dụng 4 vectors này để tính toán đặc trưng, 6 vectors này có tọa độ trong không gian

Bảng 4.3: Một SJM đặc trưng nhận được từ quá trình thí nghiệm

Điểm SJM-J	Tên SJM-J	$(x_i; y_i)$
A	Mũi	(992.574; 411.311)
B	Vai trái	(987.327; 519.09)
C	Vai phải	(987.795; 783.878)
D	Cổ tay trái	(870.784; 666.671)
E	Cổ tay phải	(826.394; 825.466)
F	Khuỷu tay trái	(1114.381; 662.63)
G	Khuỷu tay phải	(1174.36; 825.96)

ba chiều sẽ khiến cho việc mô hình hóa khó khăn hơn. Do đó, ta chuyển đổi 6 vectors này từ không gian ba chiều xuống không gian hai chiều. Xem xét việc trong quá trình hoạt động thể hiện hành động ta luôn đứng hướng thẳng mặt vào camera, do đó khoảng cách tối đa trên trục z (khoảng cách đến anten) giữa các SJM-J là không lớn nên tất cả các SJM-J được chiếu lên trục z.

Chiếu các vector khung xương lên trục z ta có được các vector mới:

$$v'_n = (x_n; y_n) \quad (4.2)$$

Để có thể tính toán tổng bình phương khoảng cách chuẩn hóa thì ta phải xoay và thay đổi độ dài của các vector sao cho vector khung xương được tạo ra giữa hai SJM-J cột sống trở thành vector đơn vị - unit vector:

$$v_{spin-shouder} = (0; 1) \quad (4.3)$$

Ta có tọa độ vị trí các SJM-J sau khi chuẩn hóa:

$$\begin{aligned} A(992.574; 411.311) &\Rightarrow \dots \\ B(987.327; 519.09) &\Rightarrow B(0; 1) \\ C(987.795; 783.878) &\Rightarrow C(0; 0) \\ D(870.784; 666.671) &\Rightarrow \dots \\ E(826.394; 825.466) &\Rightarrow \dots \\ F(1114.381; 662.63) &\Rightarrow \dots \\ G(1174.36; 825.96) &\Rightarrow \dots \end{aligned}$$

Công thức để tính toán vị trí mới, chuẩn hóa của A, B, C, D, E, F, G:

$$y_{A'} = \frac{1}{2} \left(\left(\frac{AC}{BC} \right)^2 - \left(\frac{AB}{BC} \right)^2 + 1 \right) = \frac{AC^2 - AB^2 + BC^2}{2BC^2} \quad (4.4)$$

Một hàm tính toán được thêm vào để dễ dàng lập trình hơn:

$$f_{BC}(x, y) = (y_B - y_C)x - (x_B - x_C)y - (y_B - y_C)x_C + (x_B - x_C)y_C \quad (4.5)$$

với đặc điểm:

- $f_{BC}(x_A, y_A) > 0$: điểm A nằm phía trên đoạn thẳng BC
- $f_{BC}(x_A, y_A) < 0$: điểm A nằm phía dưới đoạn thẳng BC
- $f_{BC}(x_A, y_A) = 0$: điểm A nằm trên đường thẳng chứa đoạn thẳng BC

Hàm BC không có ý nghĩa hình học tuy nhiên lại có ý nghĩa về mặt đại số, dựa vào hàm kết quả vị trí tương đối giữa điểm A và BC, ta có thể tìm được giá trị của $x_{A'}$:

$$x_{A'} = \text{sgn}(f_{BC}(x_A, y_A)) \sqrt{\left(\frac{AC}{BC} \right)^2 - y_{A'}^2} \quad (4.6)$$

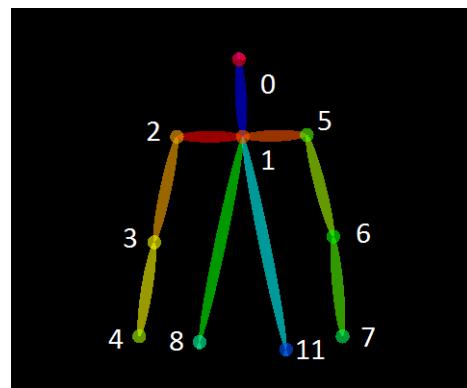
Tương tự với vị trí D_0, E_0, F_0, G_0 với điểm gốc là B,C. Sau khi tìm được $A_0, B_0, C_0, D_0, E_0, F_0, G_0$ ta có được 6 vectors đặc trưng tương ứng với SJM_i như sau:

$$\begin{aligned} \overrightarrow{v_{i,1}} &= \overrightarrow{C'A'} = (x_{A'}; y_{A'}) \\ \overrightarrow{v_{i,0}} &= \overrightarrow{C'B'} = (0; 1) \\ \overrightarrow{v_{i,2}} &= \overrightarrow{C'D'} = (x_{D'}; y_{D'}) \\ \overrightarrow{v_{i,3}} &= \overrightarrow{C'E'} = (x_{E'}; y_{E'}) \\ \overrightarrow{v_{i,4}} &= \overrightarrow{C'F'} = (x_{F'}; y_{F'}) \\ \overrightarrow{v_{i,5}} &= \overrightarrow{C'G'} = (x_{G'}; y_{G'}) \end{aligned}$$

Áp dụng cách tính toán trên vào bảng 4.3, ta được tọa độ 6 vector đặc trưng của các SJM_i tương ứng và minh họa trên hình 4.5:

Bảng 4.4: Một SJM đặc trưng sau khi chuẩn hóa

Điểm SJM-J	Tên SJM-J	$(x_i; y_i)$
A	Dầu	(0.0205352100186477; -0.407002543626264)
B	Cột sống vai	(0; 0)
C	Cột sống gốc	(0; 1)
D	Khuỷu tay trái	(-0.441120735865053; 0.55657565862356)
E	Bàn tay trái	(-0.60982370936332; 1.15598366430513)
F	Khuỷu tay phải	(0.478873301943448; 0.542940438030838)
G	Bàn tay phải	(0.704299437754533; 1.16017195694997)

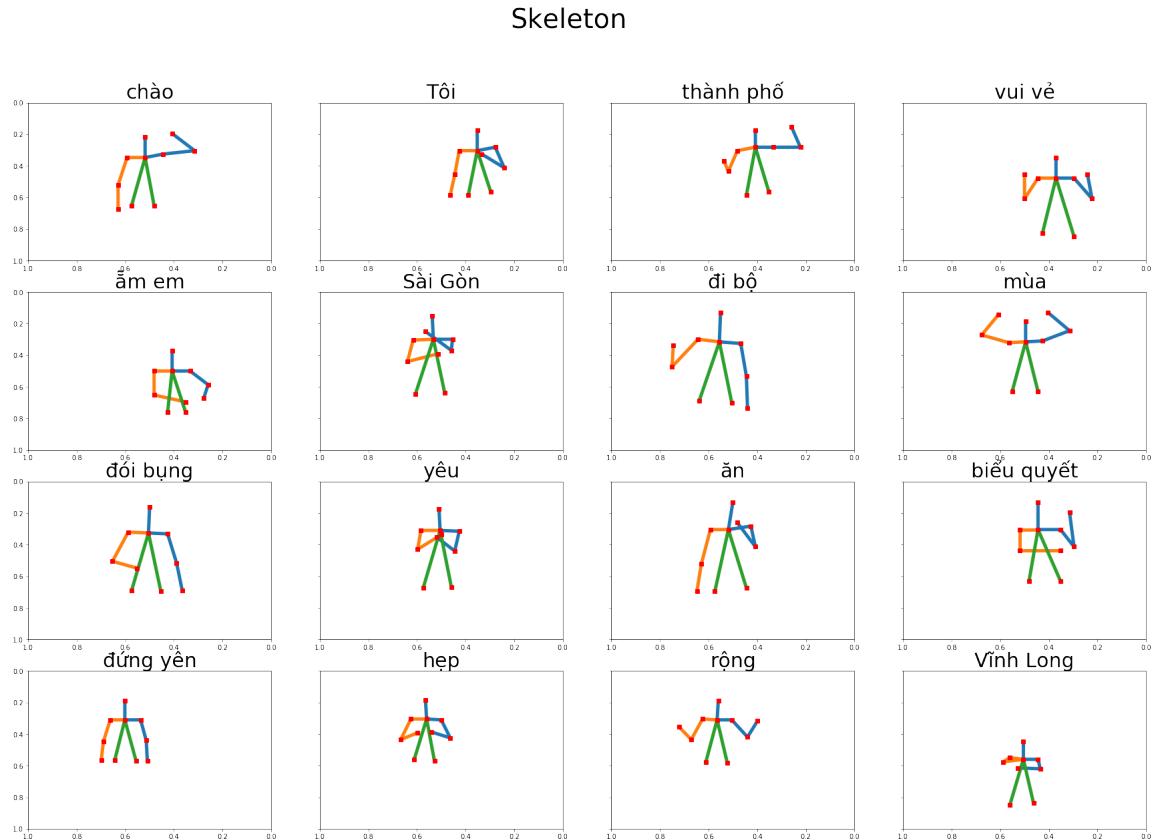


Hình 4.5: SJM sau chuẩn hóa

Giá trị bình phương sai lệch khoảng cách giữa hai SJM_i và SJM_j đã chuẩn hóa mang ý nghĩa là giá trị khác biệt giữa hai SJMs:

$$\Delta_{ij} = \sum_{k=1}^5 \| \vec{v}_{i,k} - \vec{v}_{j,k} \| \quad (4.7)$$

Sau khi loại bỏ các phần thừa và chuẩn hóa hết bộ dữ liệu các SJM, ta được các SJM với tọa độ các điểm khớp xương được chuẩn hóa nằm trong khoảng $0 \rightarrow 1$ thể hiện trong hình 4.6



Hình 4.6: Dữ liệu khớp xương sau khi đã được chuẩn hóa

4.4 Cấu trúc mạng neural network xây dựng

Từ dữ liệu đầu vào là một ảnh RGB, qua bước phát hiện khung xương với mạng CNN và các bước chuẩn hóa, loại bỏ khung xương thừa, dữ liệu cần phân loại gần như đã lấy ra hết đặc trưng cần tìm và giảm số chiều đến tối đa. Do đó mạng DNN cần dùng để phân loại các từ ngôn ngữ ký hiệu chỉ cần một cấu trúc vừa phải và giảm thiểu số trọng số đến tối đa.

- Input: Mạng với đầu vào là 20 nodes ứng với tọa độ của 10 khớp xương (mỗi khớp là 2 tọa độ).
- Hidden layer: Là các lớp Fully connected (Dense) và Normalization xen kẽ nhau.
- Output: Với softmax với 16 nodes ứng với 16 từ ngữ nhận diện được.

Cụ thể các thông số và cấu trúc được thể hiện trong hình 4.7.

```
model.summary()
```

```
Model: "sequential_4"
```

Layer (type)	Output Shape	Param #
<hr/>		
dense_13 (Dense)	(None, 128)	2432
batch_normalization_10 (Batch Normalization)	(None, 128)	512
dense_14 (Dense)	(None, 64)	8256
batch_normalization_11 (Batch Normalization)	(None, 64)	256
dense_15 (Dense)	(None, 16)	1040
batch_normalization_12 (Batch Normalization)	(None, 16)	64
dense_16 (Dense)	(None, 16)	272
<hr/>		
Total params: 12,832		
Trainable params: 12,416		
Non-trainable params: 416		

Hình 4.7: Các thông số của mạng DNN

Model sử dụng khá nhiều lớp Batch Normalization. Về lý thuyết toán thì layer này có tác dụng trong 2 việc đó là:

- Tránh gradient tiến về 0 sẽ chẳng biết là đi lên hay xuống.
- Gradient quá lớn khiến các tham số không thể tối ưu được (loss sẽ nhảy lung tung).

Cụ thể là nó giúp chuẩn hóa dữ liệu về 1 khoảng (0,1).

4.5 Huấn luyện mạng

Mạng được huấn luyện theo thuật toán Stochastic Gradient Descent (SGD) là một biến thể của Gradient Descent. Cụ thể ta sẽ tìm hiểu về nó qua các phần dưới đây.

4.5.1 Gradient Descent

Trong Machine Learning nói riêng và Toán Tối Ưu nói chung, chúng ta thường xuyên phải tìm giá trị nhỏ nhất (hoặc đôi khi là lớn nhất) của một hàm số nào đó. Ví dụ như các hàm mất mát trong hai bài Linear Regression và K-means Clustering. Nhìn chung, việc tìm global minimum của các hàm mất mát trong Machine Learning là rất phức tạp, thậm chí là bất khả thi. Thay vào đó, người ta thường cố gắng tìm các điểm local minimum, và ở một mức độ nào đó, coi đó là nghiệm cần tìm của bài toán.

Các điểm local minimum là nghiệm của phương trình đạo hàm bằng 0. Nếu bằng một cách nào đó có thể tìm được toàn bộ (hữu hạn) các điểm cực tiểu, ta chỉ cần thay từng điểm local minimum đó vào hàm số rồi tìm điểm làm cho hàm có giá trị nhỏ nhất. Tuy nhiên, trong hầu hết các trường hợp, việc giải phương trình đạo hàm bằng 0 là bất khả thi. Nguyên nhân có thể đến từ sự phức tạp của dạng của đạo hàm, từ việc các điểm dữ liệu có số chiều lớn, hoặc từ việc có quá nhiều điểm dữ liệu.

Hướng tiếp cận phổ biến nhất là xuất phát từ một điểm mà chúng ta coi là gần với nghiệm của bài toán, sau đó dùng một phép toán lặp để tiến dần đến điểm cần tìm, tức đến khi đạo hàm gần với 0. Gradient Descent (viết gọn là GD) và các biến thể của nó là một trong những phương pháp được dùng nhiều nhất.

Các bước quan trọng của thuật toán Gradient Descent như sau:

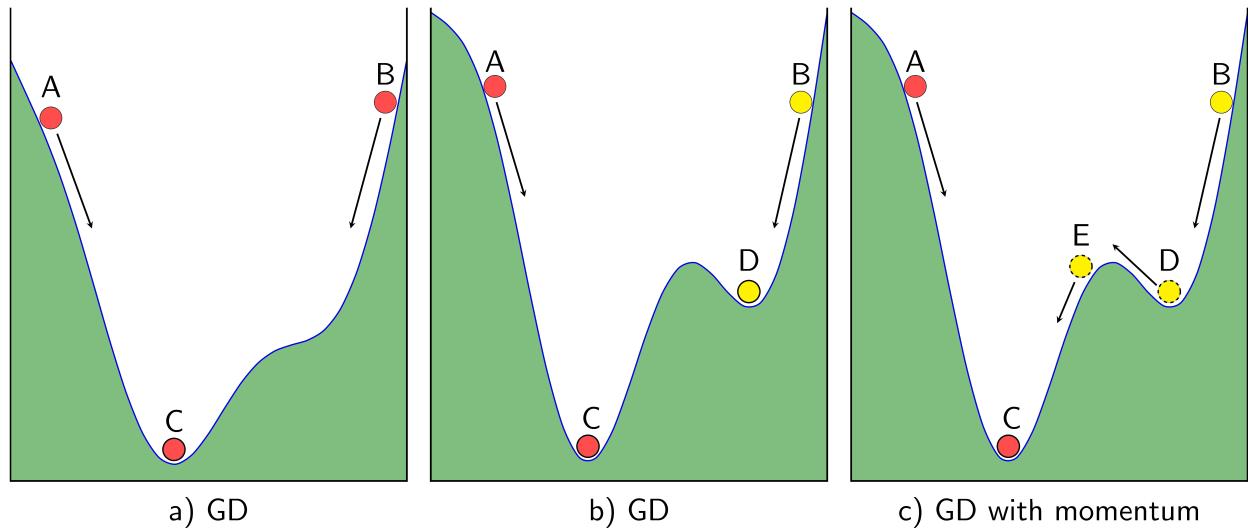
- Dự đoán một điểm khởi tạo $\theta = \theta_0$.
- Cập nhật θ đến khi đạt được kết quả chấp nhận được:

$$\theta = \theta - \eta \nabla_{\theta} J(\theta)$$

với $\nabla_{\theta} J(\theta)$ là đạo hàm của hàm mất mát tại θ

Gradient dưới góc nhìn vật lý:

Thuật toán GD thường được ví với tác dụng của trọng lực lên một hòn bi đặt trên một mặt có dạng như hình một thung lũng giống như hình 4.8 dưới đây. Bất kể ta đặt hòn bi ở A hay B thì cuối cùng hòn bi cũng sẽ lăn xuống và kết thúc ở vị trí C.



Hình 4.8: So sánh Gradient Descent với các hiện tượng vật lý (Nguồn: "Machine learning cơ bản" - Vũ Hữu Tiệp)

Tuy nhiên, nếu như bề mặt có hai đáy thung lũng như Hình 4.8b thì tùy vào việc đặt bi ở A hay B, vị trí cuối cùng của bi sẽ ở C hoặc D. Điểm D là một điểm local minimum chúng ta không mong muốn.

Nếu suy nghĩ một cách vật lý hơn, vẫn trong Hình 4.8b, nếu vận tốc ban đầu của bi khi ở điểm B đủ lớn, khi bi lăn đến điểm D, theo đó, bi có thể tiếp tục di chuyển lên dốc phía bên trái của D. Và nếu giả sử vận tốc ban đầu lớn hơn nữa, bi có thể vượt dốc tới điểm E rồi lăn xuống C như trong Hình 4.8c. Đây chính là điều chúng ta mong muốn.

4.5.2 Stochastic Gradient Descent

Trong thuật toán này, tại 1 thời điểm, ta chỉ tính đạo hàm của hàm mất mát dựa trên chỉ một điểm dữ liệu x_i rồi cập nhật θ dựa trên đạo hàm này. Việc này được thực hiện với từng điểm trên toàn bộ dữ liệu, sau đó lặp lại quá trình trên. Thuật toán rất đơn giản này trên thực tế lại làm việc rất hiệu quả.

Mỗi lần duyệt một lượt qua tất cả các điểm trên toàn bộ dữ liệu được gọi là một epoch. Với GD thông thường thì mỗi epoch ứng với 1 lần cập nhật θ , với SGD thì mỗi epoch ứng với

N lần cập nhật θ với N là số điểm dữ liệu. Nhìn vào một mặt, việc cập nhật từng điểm một như thế này có thể làm giảm đi tốc độ thực hiện 1 epoch. Nhưng nhìn vào một mặt khác, SGD chỉ yêu cầu một lượng epoch rất nhỏ (thường là 10 cho lần đầu tiên, sau đó khi có dữ liệu mới thì chỉ cần chạy dưới một epoch là đã có nghiệm tốt). Vì vậy SGD phù hợp với các bài toán có lượng cơ sở dữ liệu lớn (chủ yếu là Deep Learning mà chúng ta sẽ thấy trong phần sau của blog) và các bài toán yêu cầu mô hình thay đổi liên tục, tức online learning.

Thứ tự lựa chọn điểm dữ liệu: Một điểm cần lưu ý đó là: sau mỗi epoch, chúng ta cần shuffle (xáo trộn) thứ tự của các dữ liệu để đảm bảo tính ngẫu nhiên. Việc này cũng ảnh hưởng tới hiệu năng của SGD.

Một cách toán học, quy tắc cập nhật của SGD là:

$$\theta = \theta - \eta \nabla_{\theta} J(\theta; \mathbf{x}_i; \mathbf{y}_i)$$

trong đó $J(\theta; \mathbf{x}_i; \mathbf{y}_i)$ là hàm mất mát với chỉ 1 cặp điểm dữ liệu (input, label) là $(\mathbf{x}_i, \mathbf{y}_i)$. Ngoài ra, chúng ta hoàn toàn có thể áp dụng các thuật toán tăng tốc GD như Momentum, AdaGrad,... vào SGD.

4.5.3 Điều kiện dừng

Trong thực nghiệm, có một vài phương pháp như dưới đây để xác định khi nào cần dừng số vòng lặp khi huấn luyện:

- Giới hạn số vòng lặp: đây là phương pháp phổ biến nhất và cũng để đảm bảo rằng chương trình chạy không quá lâu. Tuy nhiên, một nhược điểm của cách làm này là có thể thuật toán dừng lại trước khi đủ gần với nghiệm.
- So sánh gradient của nghiệm tại hai lần cập nhật liên tiếp, khi nào giá trị này đủ nhỏ thì dừng lại. Phương pháp này cũng có một nhược điểm lớn là việc tính đạo hàm đôi khi trở nên quá phức tạp (ví dụ như khi có quá nhiều dữ liệu), nếu áp dụng phương pháp này thì coi như ta không được lợi khi sử dụng SGD và mini-batch GD.
- So sánh giá trị của hàm mất mát của nghiệm tại hai lần cập nhật liên tiếp, khi nào giá trị này đủ nhỏ thì dừng lại. Nhược điểm của phương pháp này là nếu tại một thời điểm, đồ thị hàm số có dạng bằng phẳng tại một khu vực nhưng khu vực đó không chứa điểm

local minimum (khu vực này thường được gọi là saddle points), thuật toán cũng dừng lại trước khi đạt giá trị mong muốn.

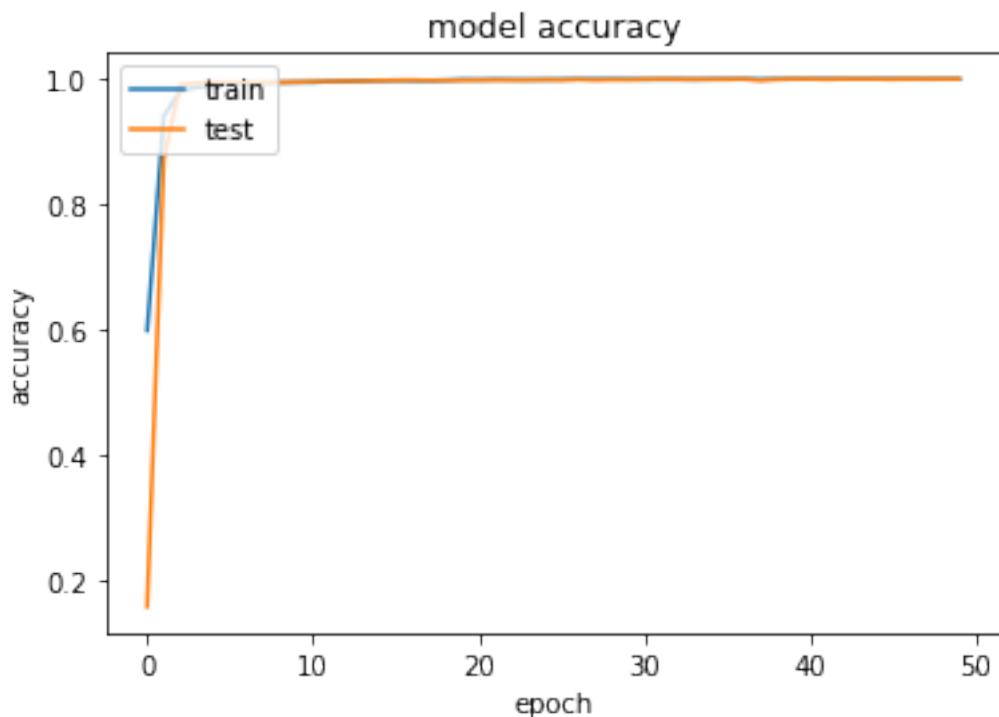
- Trong SGD và mini-batch GD, cách thường dùng là so sánh nghiệm sau một vài lần cập nhật.

4.5.4 Áp dụng vào bài toán

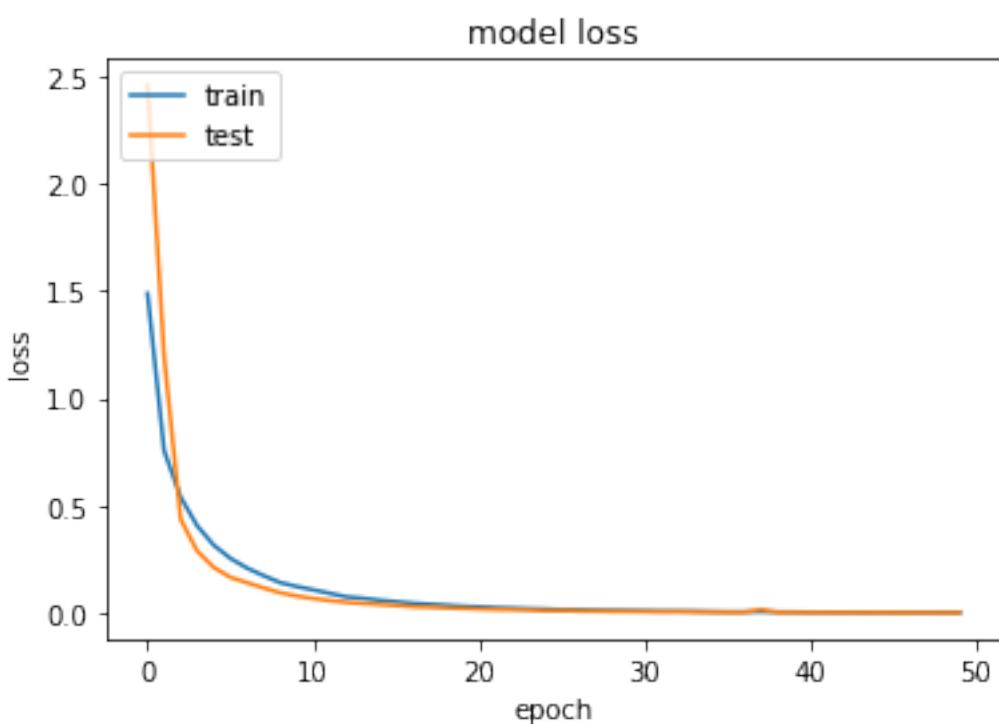
Với bộ dữ liệu của bài toán nhận dạng ngôn ngữ ký hiệu, dữ liệu là các SJM được thu thập từ trước. Dữ liệu sẽ được chia ra thành 2 tập gồm: train data (dữ liệu để huấn luyện) và test data (dữ liệu đánh giá) với tỷ lệ 80 và 20. Nghĩa là với mỗi hành động (data gồm 800 SJM) sẽ có 640 SJM để train và 160 SJM để đánh giá. Áp dụng thuật toán SGD, với các thông số ban đầu được khởi tạo trước khi huấn luyện như sau:

- epoch: 50
- learning rate: 0.0001
- batch_size : 32

Trong quá trình huấn luyện, luận văn sử dụng phương pháp lưu lại checkpoint. Cách thức hoạt động của phương pháp này là sau mỗi epoch, accuracy trên tập test của model hiện tại được so sánh với accuracy của model trước đó. Nếu nó lớn hơn thì trọng số của mô hình sẽ được cập nhật theo trọng số mới nhất. Nếu nhỏ hơn, trọng số của mô hình trước đó vẫn được giữ lại cho đến khi accuracy khác cao hơn. Phương pháp này giúp tìm được model tối ưu nhất với accuracy cao nhất có thể. Sau 50 epoch, ta có kết quả về độ chính xác(accuracy) và sai số (loss) trên tập train và tập test như sau:



Hình 4.9: Accuracy của tập train và validate



Hình 4.10: Loss của tập train và validate

Từ biểu đồ accuracy và loss của mô hình, có thể thấy mô hình đã học được rất tốt. Kể từ epoch thứ 10, mô hình đã đạt được accuracy cao và loss giảm xuống thấp. Đường biểu diễn của 2 tập đối với accuracy và loss đi sát nhau có thể phần nào dự đoán được mô hình không bị overfitting.

4.6 Kết quả

Nhận thấy kết quả mô hình đã học được rất tốt với thuật toán này. Với kết quả cuối cùng:

- Train Accuracy: 1.0000
- Test Accuracy: 0.9996
- Train Loss: 0.0153
- Test Loss : 0.0123

Dựa vào những thông số trên có thể thấy kết quả sau khi huấn luyện mô hình khá tốt. Nhưng để áp dụng mô hình vào thực tế được, ta cần phải thử nghiệm và đánh giá bằng nhiều phương pháp trên tập dữ liệu mới. Trong chương 5 luận văn sẽ trình bày về cách thử nghiệm, đánh giá mô hình cũng như xây dựng phần mềm ứng dụng từ mô hình. Chương 4 tiếp theo luận văn sẽ trình bày giải thuật Deep Sort dùng để theo dõi những người trong khung ảnh, từ đó có thể áp dụng nhận diện dc chuỗi các ký tự của ngôn ngữ ký hiệu mà từng người diễn đạt.

Chương 5

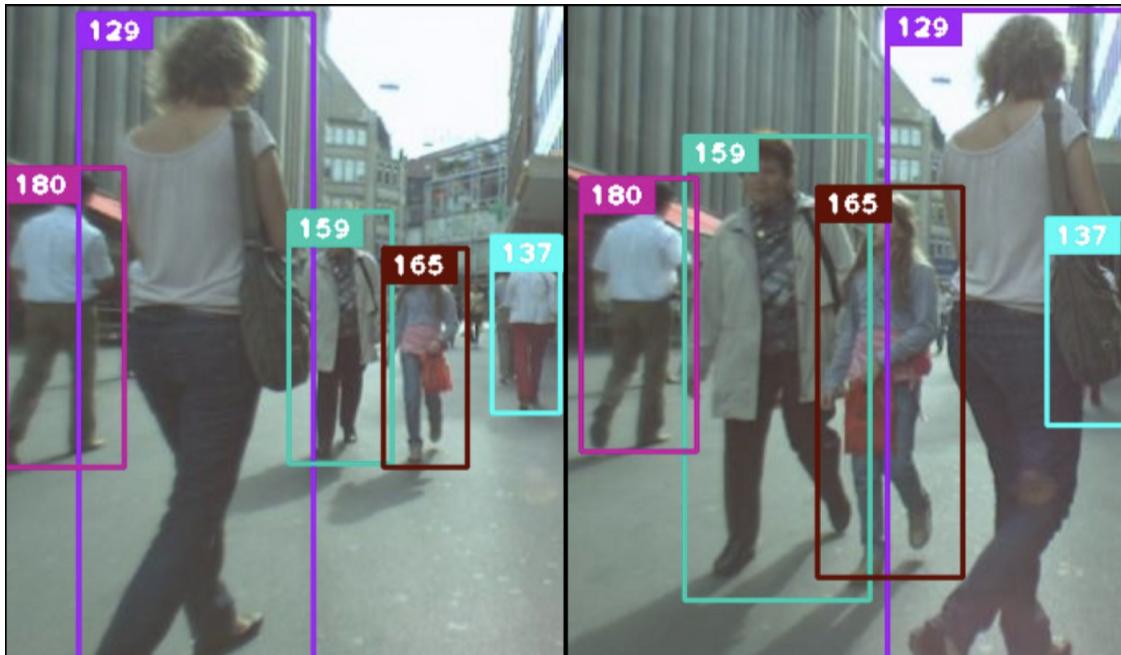
Giải thuật Deep Sort theo dõi từng người trong khung hình

Qua các chương trước, luận văn đã trình bày thuật toán, lý thuyết cũng như cách thức hoạt động của chương trình nhận dạng ngôn ngữ ký hiệu. Ở chương này, luận văn sẽ trình bày giải thuật Deep Sort dùng để theo dõi từng người trong khung hình, đánh số thứ tự từng người, cùng với ngôn ngữ ký hiệu của họ muốn diễn đạt.

Phát hiện đối tượng và theo dõi đối tượng là một trong những chủ đề được thị giác máy tính nghiên cứu từ rất lâu. Các thành tựu của chúng đã đạt đến những thành công rất cao cũng như được ứng dụng rộng rãi vào đời sống. Phát hiện đối tượng (objects detection) chỉ tập trung vào việc phát hiện từng đối tượng, đặt chúng trong từng khung hình riêng lẻ và sau đó phân loại đối tượng đó. Việc phát hiện đối tượng chỉ dừng lại ở đây, như vậy đối với cùng một đối tượng nhưng với các khung hình tiếp nhau, máy tính sẽ không thể biết được 2 khung hình này chứa cùng một đối tượng . Khác với các thuật toán phát hiện đối tượng, các thuật toán theo dõi đối tượng đều hoạt động theo cách thức khóa từng đối tượng trong khung hình, xác định duy nhất từng đối tượng và theo dõi tất cả chúng cho đến khi chúng rời khỏi khung hình. Ví dụ, nếu máy tính phát hiện được 3 ôtô trong một khung hình, trình theo dõi sẽ phải xác định được 3 đối tượng riêng biệt, đánh số thứ tự chúng và theo dõi qua các khung hình tiếp theo. Theo dõi đối tượng bao gồm theo dõi đối tượng đơn(single object tracking) và theo dõi đa đối tượng (multiple object tracking). Trong phần này, luận văn sẽ chủ yếu đề cập đến các thuật toán theo dõi đa đối tượng cụ thể là giải thuật Sort và Deep Sort (được ứng dụng trong luận văn).

5.1 Giải thuật Sort truyền thống

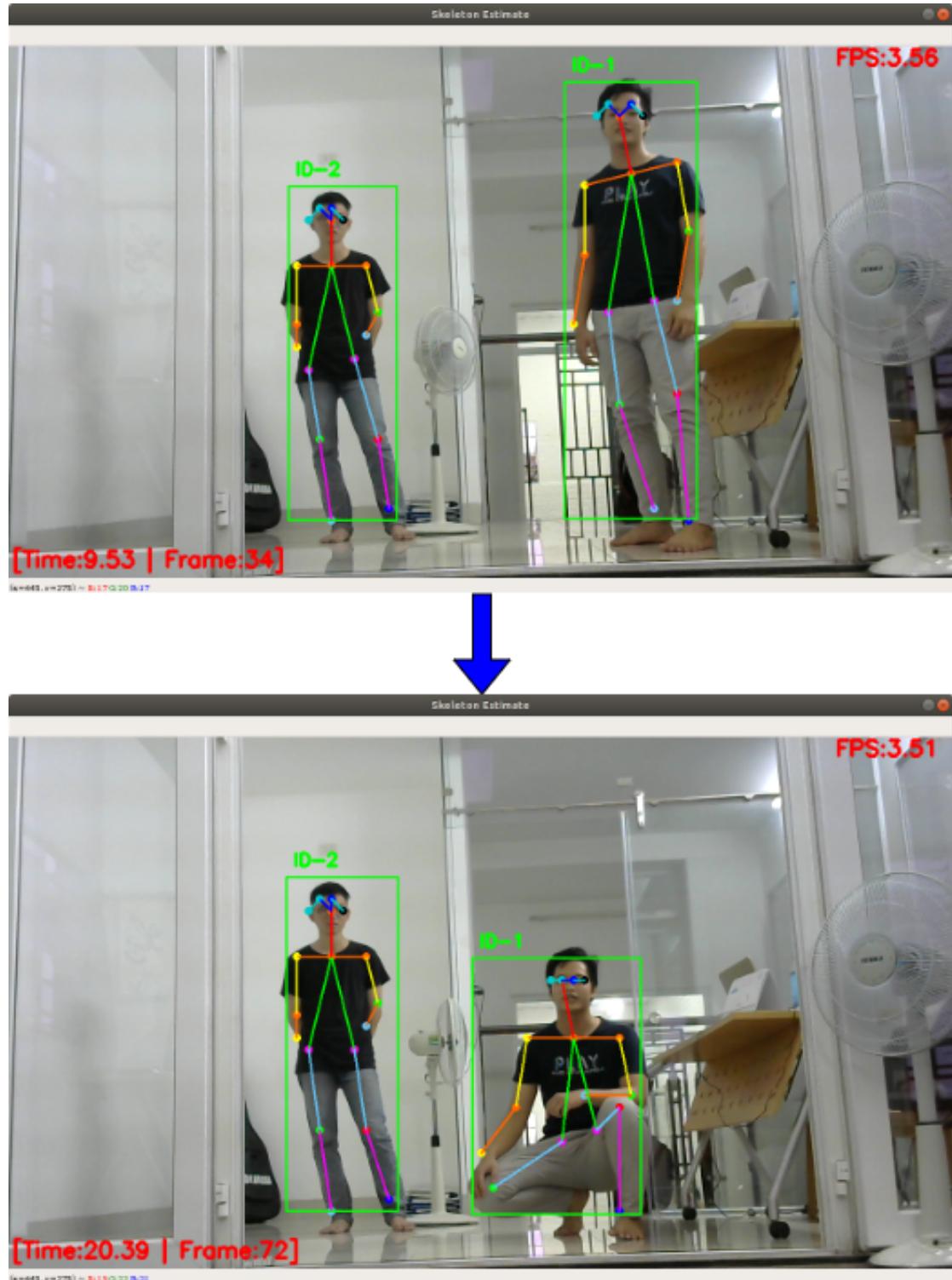
Phổ biến nhất và là một trong những thuật toán đơn giản nhất để theo dõi là SORT (Simple Online and Realtime Tracking [17]). Thuật toán này có thể theo dõi nhiều đối tượng trong thời gian thực nhưng thuật toán chỉ liên kết các đối tượng đã phát hiện trên các khung khác nhau dựa trên tọa độ của kết quả phát hiện, như hình 5.1



Hình 5.1: Thuật toán Sort

Nguồn : "Simple Online and Realtime Tracking with a Deep Association Metric" [3]

5.2 Giải thuật Deep Sort



Hình 5.2: Kết quả Deep Sort

5.3 Kết luận

Chương 6

THỰC NGHIỆM KẾT QUẢ VÀ ĐÁNH GIÁ

Qua các chương trước, luận văn đã trình bày thuật toán, lý thuyết cũng như cách thức hoạt động của chương trình nhận dạng ngôn ngữ ký hiệu. Ở chương này, luận văn sẽ trình bày kết quả thực nghiệm mà luận văn đã đạt được sau khi huấn luyện và các phương pháp đánh giá độ chính xác của mô hình.

6.1 Thực hiện phần mềm nhận dạng ngôn ngữ ký hiệu

Sau khi đã huấn luyện được mô hình nhận dạng ngôn ngữ ký hiệu, xây dựng thuật toán theo dõi đối tượng dựa trên Deep Sort. Tiếp theo, luận văn tiến hành xây dựng phần mềm nhận dạng thời gian thực để ứng dụng được mô hình đã huấn luyện trên ra thực tế.

- **Ngôn ngữ lập trình** Python được sử dụng làm ngôn ngữ lập trình chính để xây dựng model cũng như viết ứng dụng nhận dạng.
- **Thư viện Keras** Thư viện Keras được sử dụng trong việc xây dựng model

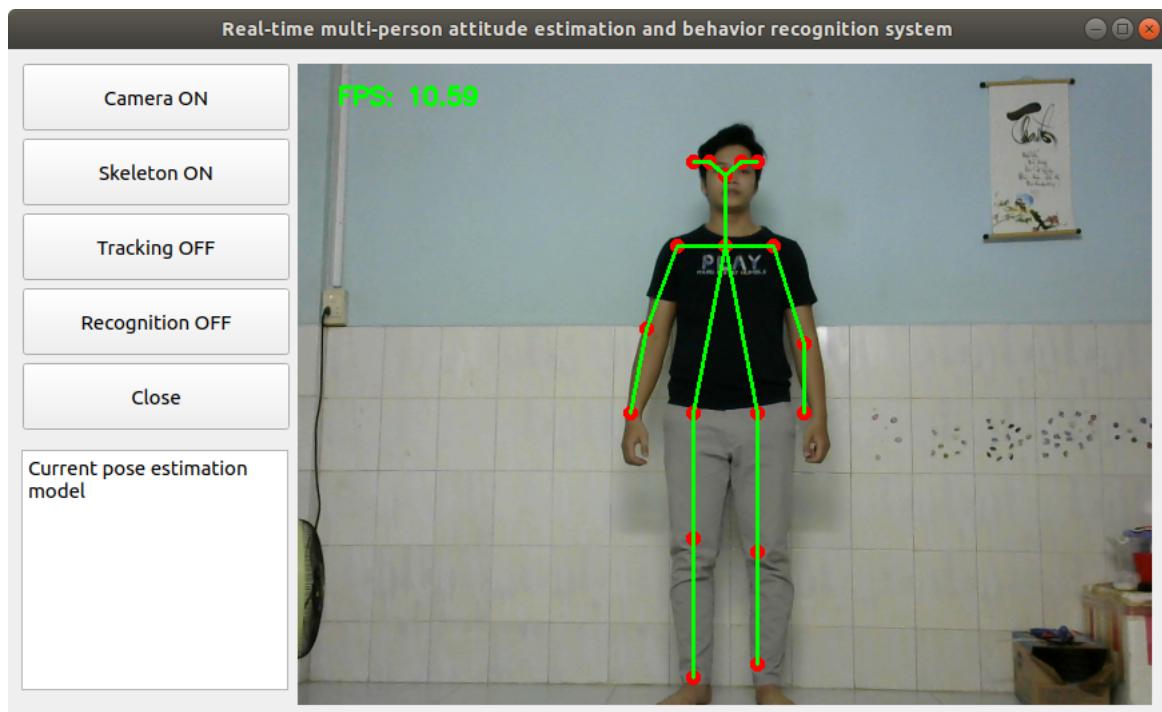
6.2 Kết quả

Chương trình thực nghiệm đã nhận dạng được khá chính xác các hành động đã huấn luyện (16 ngôn ngữ ký hiệu). Chương trình chạy trên hệ điều hành ubuntu, sử dụng camera là webcam.

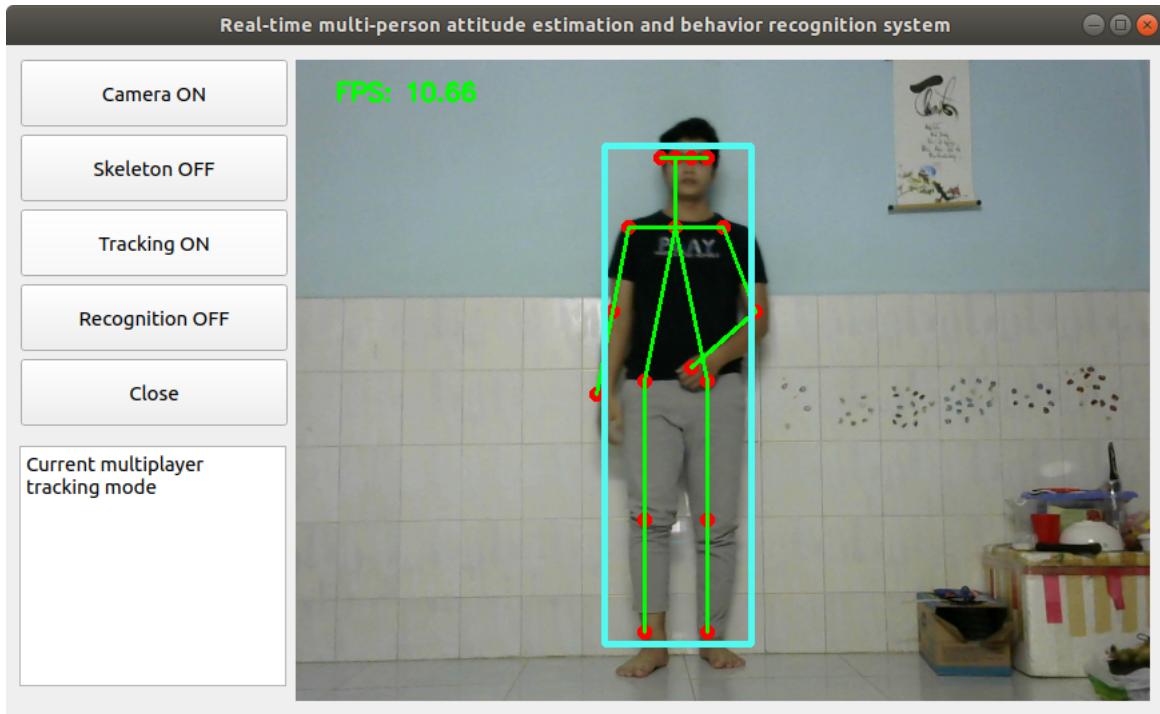
Giao diện được viết trên thư viện PyQt5, là một thư viện hỗ trợ viết giao diện ứng dụng trên ngôn ngữ Python. Các nút điều khiển trên giao diện, giúp chọn các chế độ hoạt động của chương trình. Có tất cả 3 chế độ đều có thể hoạt động khi có nhiều người trong khung hình:

- Skeleton detection: Phát hiện và và trích đặc trưng khung xương từ camera.
- Tracking: Theo dõi và đánh chỉ số của từng người trong khung hình dựa trên giải thuật Deep Sort.
- Recognize: Theo dõi từng người và nhận dạng ngôn ngữ ký hiệu.

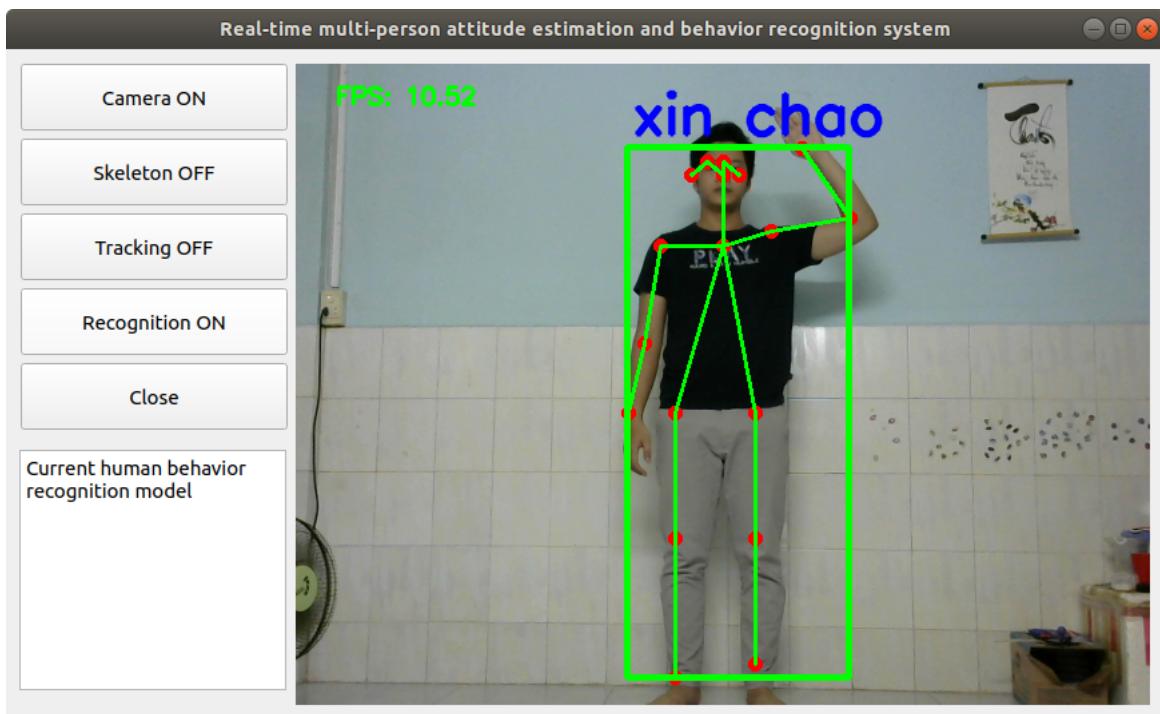
Giao diện và các mode hoạt động được trình bày như các hình 6.1, 6.2, 6.3



Hình 6.1: Giao diện phần mềm nhận dạng - Mode "Skeleton detection"



Hình 6.2: Giao diện phần mềm nhận dạng - Mode "Tracking"



Hình 6.3: Giao diện phần mềm nhận dạng - Mode "Recognize"

6.3 Đánh giá

Khi xây dựng một mô hình Machine Learning, chúng ta cần một phép đánh giá để xem mô hình sử dụng có hiệu quả không và để so sánh khả năng của các mô hình. Trong phần này, luận văn sẽ trình bày các mô hình classification.

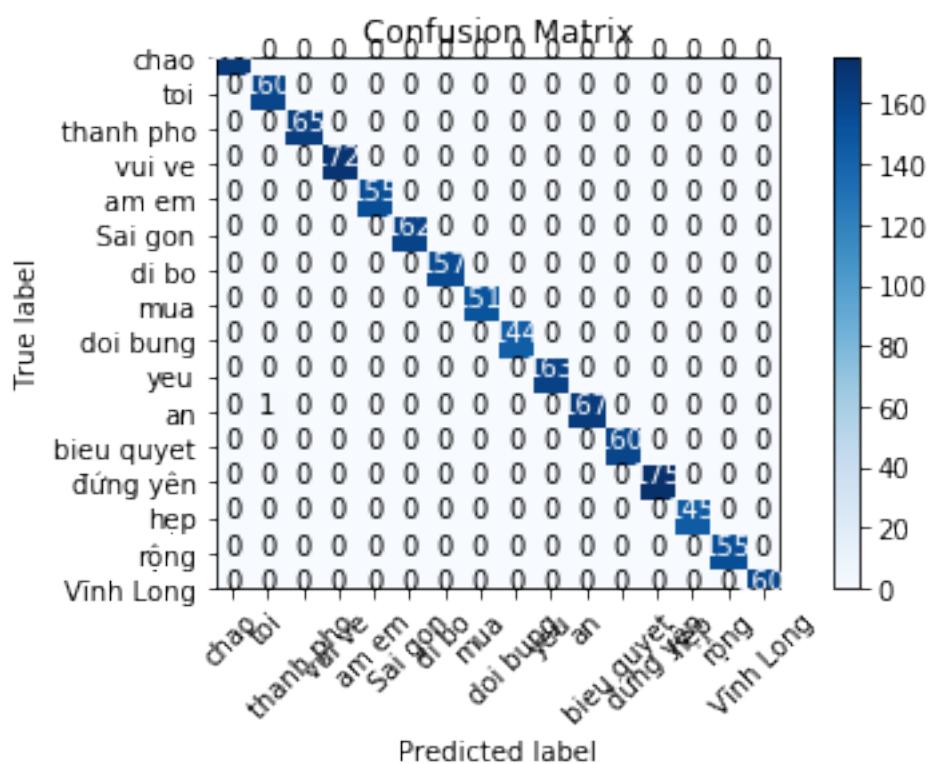
Hiệu năng của một mô hình thường được đánh giá dựa trên tập dữ liệu kiểm thử (test data). Cụ thể, giả sử đầu ra của mô hình khi đầu vào là tập kiểm thử được mô tả bởi vector $y\text{-pred}$ - là vector dự đoán đầu ra với mỗi phần tử là class được dự đoán của một điểm dữ liệu trong tập kiểm thử. Ta cần so sánh giữa vector dự đoán $y\text{-pred}$ này với vector class thật của dữ liệu, được mô tả bởi vector $y\text{-true}$. Đối với bài toán phân loại của luận văn, có 16 lớp dữ liệu được gán nhãn là tên các từ được huấn luyện.

Có rất nhiều cách đánh giá một mô hình phân lớp. Tuỳ vào những bài toán khác nhau mà chúng ta sử dụng các phương pháp khác nhau. Các phương pháp thường được sử dụng là: accuracy score, confusion matrix, ROC curve, Area Under the Curve, Precision and Recall, F1 score, Top R error, etc.

Trong phần này, luận văn sẽ trình bày về accuracy score, confusion matrix, ROC curve, và Area Under the Curve. Và đánh giá mô hình của luận văn trên tập dữ liệu test gồm 400 SJM đối với mỗi từ.

6.3.1 Accuracy

Cách đơn giản và hay được sử dụng nhất là accuracy (độ chính xác). Cách đánh giá này đơn giản tính tỉ lệ giữa số điểm được dự đoán đúng và tổng số điểm trong tập dữ liệu kiểm thử.



Hình 6.4: Confusion matrix của 16 lớp phân loại

Chương 7

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

7.1 Kết luận

7.2 Hướng phát triển

Tài liệu tham khảo

- [1] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, *et al.*, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [2] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *arXiv preprint arXiv:1502.03167*, 2015.
- [3] N. Wojke, A. Bewley, and D. Paulus, “Simple online and realtime tracking with a deep association metric,” in *2017 IEEE International Conference on Image Processing (ICIP)*, pp. 3645–3649, IEEE, 2017.
- [4] X. Chai, “Sign language recognition and translation with kinect,” *IEEE Conf. on AFGR*, 2013.
- [5] M. Boulares and M. Jemni, “Mobile sign language translation system for deaf community,” in *Proceedings of the International Cross-Disciplinary Conference on Web Accessibility*, W4A ’12, (New York, NY, USA), pp. 37:1–37:4, ACM, 2012.
- [6] T.-H. T. J.-W. H. C.-M. Tsai, “Sign language recognition system via kinect: Number and english alphabet,” *Machine Learning and Cybernetics (ICMLC), International Conference*, 2016.
- [7] J. Yamato, J. Ohya, and K. Ishii, “Recognizing human action in time-sequential images using hidden markov model,” in *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR ’92., 1992 IEEE Computer Society Conference on*, pp. 379–385, Jun 1992.
- [8] C. Schuldt, I. Laptev, and B. Caputo, “Recognizing human actions: A local svm approach,” in *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR’04) Volume 3 - Volume 03*, ICPR ’04, (Washington, DC, USA), pp. 32–36, IEEE Computer Society, 2004.

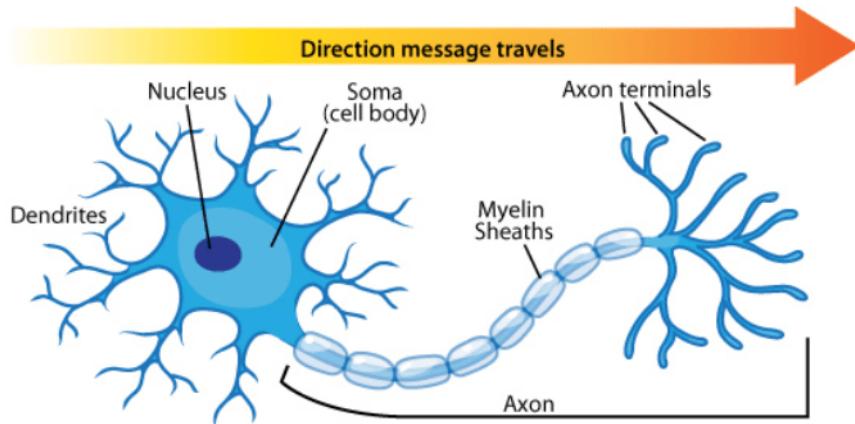
- [9] H.-S. Chen, H.-T. Chen, Y.-W. Chen, and S.-Y. Lee, “Human action recognition using star skeleton,” in *Proceedings of the 4th ACM International Workshop on Video Surveillance and Sensor Networks, VSSN ’06*, (New York, NY, USA), pp. 171–178, ACM, 2006.
- [10] W. Yan and D. Forsyth, “Learning the behavior of users in a public space through video tracking,” in *Application of Computer Vision, 2005. WACV/MOTIONS ’05 Volume 1. Seventh IEEE Workshops on*, vol. 1, pp. 370–377, Jan 2005.
- [11] W. Lao, J. Han, and P. de With, “Automatic video-based human motion analyzer for consumer surveillance system,” *Consumer Electronics, IEEE Transactions on*, vol. 55, pp. 591–598, May 2009.
- [12] M. Cristani, R. Raghavendra, A. D. Bue, and V. Murino, “Human behavior analysis in video surveillance: A social signal processing perspective,” *Neurocomputing*, vol. 100, pp. 86 – 97, 2013. Special issue: Behaviours in video.
- [13] H. L. U. Thuc, P. V. Tuan, and J.-N. Hwang, “An effective 3d geometric relational feature descriptor for human action recognition,” in *Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), 2012 IEEE RIVF International Conference on*, pp. 1–6, Feb 2012.
- [14] Y. Ming, “Human activity recognition based on 3d mesh mosift feature descriptor,” in *Social Computing (SocialCom), 2013 International Conference on*, pp. 959–962, Sept 2013.
- [15] Y. Ariki, J. Morimoto, and S. Hyon, “Behavior recognition with ground reaction force estimation and its application to imitation learning,” in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pp. 2029–2034, Sept 2008.
- [16] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, “Realtime multi-person 2d pose estimation using part affinity fields,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7291–7299, 2017.
- [17] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, “Simple online and realtime tracking,” in *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 3464–3468, IEEE, 2016.

Phụ lục

Các từ nhạy dạng được

- **Tôi**

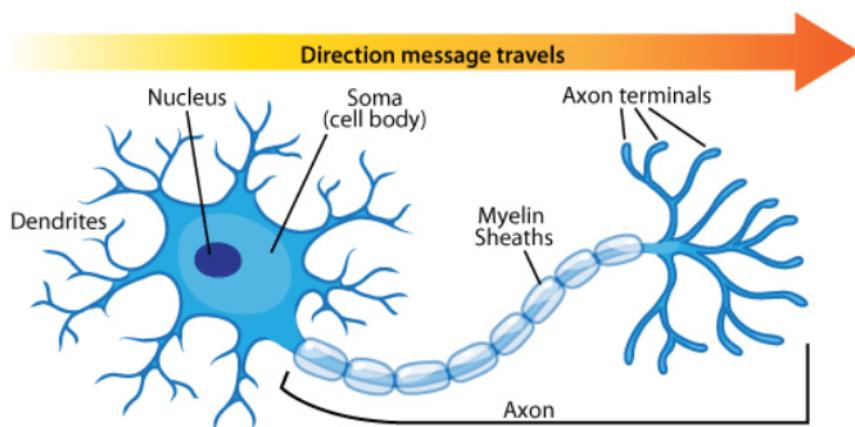
Neuron Anatomy



Hình 7.1: Cấu trúc một tế bào thần kinh

- **Đi bộ**

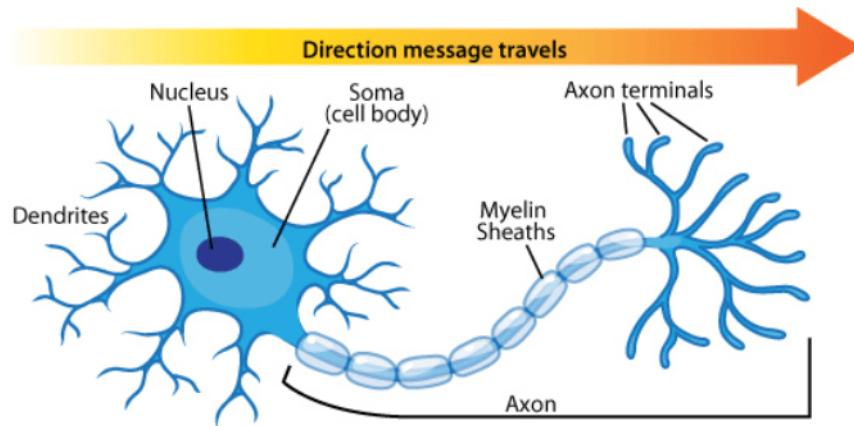
Neuron Anatomy



Hình 7.2: Cấu trúc một tế bào thần kinh

- **Đứng yên**

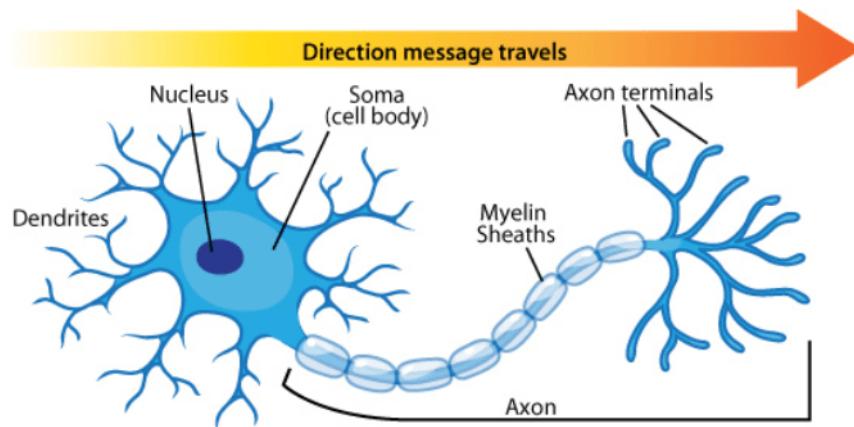
Neuron Anatomy



Hình 7.3: Cấu trúc một tế bào thần kinh

- **Vui vẻ**

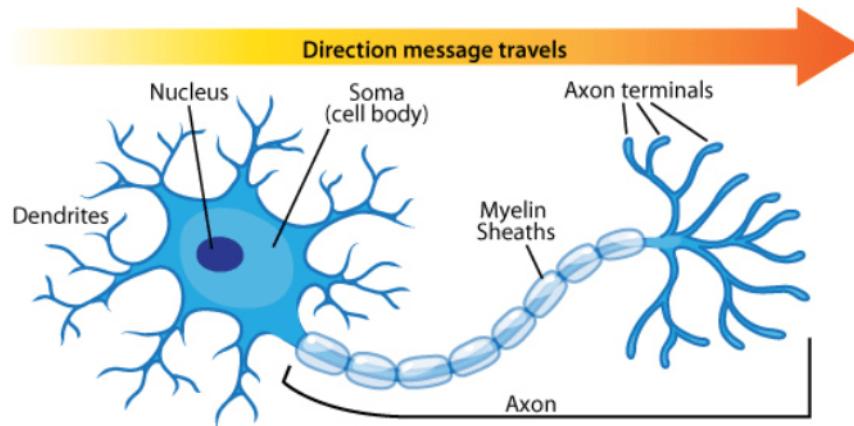
Neuron Anatomy



Hình 7.4: Cấu trúc một tế bào thần kinh

- **TP.HCM**

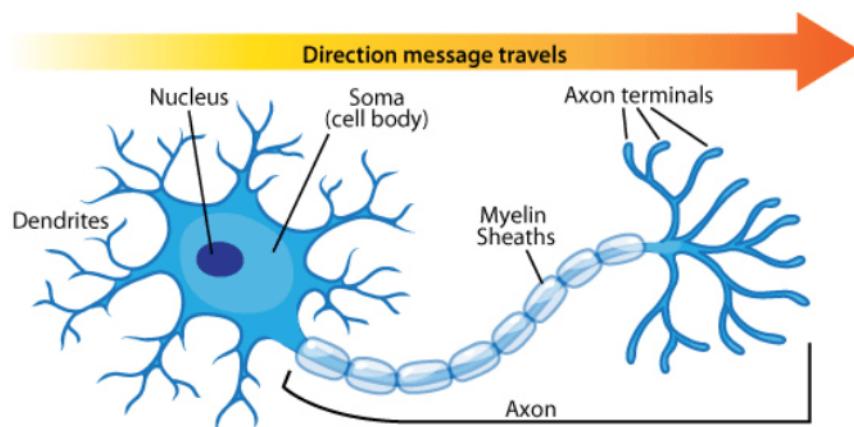
Neuron Anatomy



Hình 7.5: Cấu trúc một tế bào thần kinh

- **VĨNH LONG**

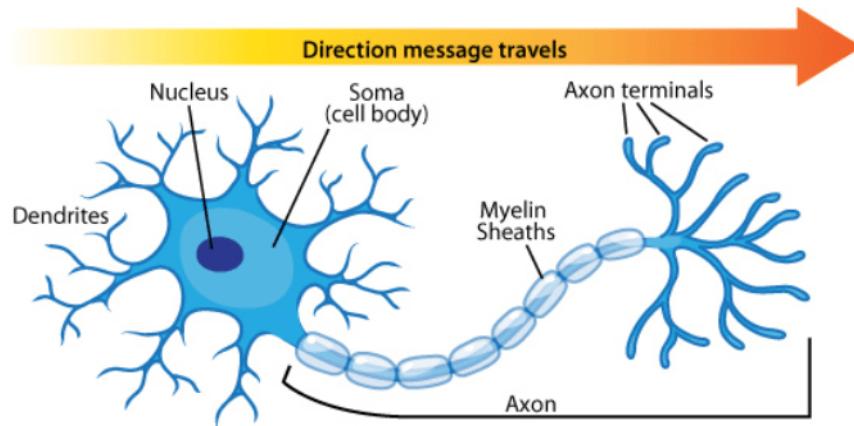
Neuron Anatomy



Hình 7.6: Cấu trúc một tế bào thần kinh

- **MÙA MÀNG**

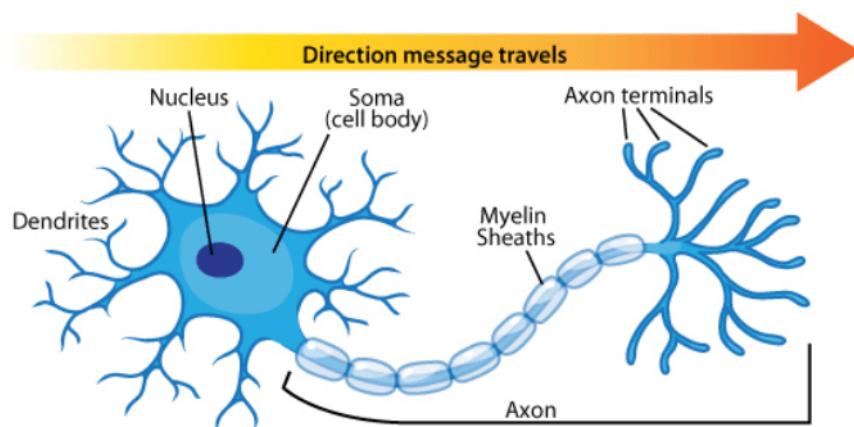
Neuron Anatomy



Hình 7.7: Cấu trúc một tế bào thần kinh

• Ăm em

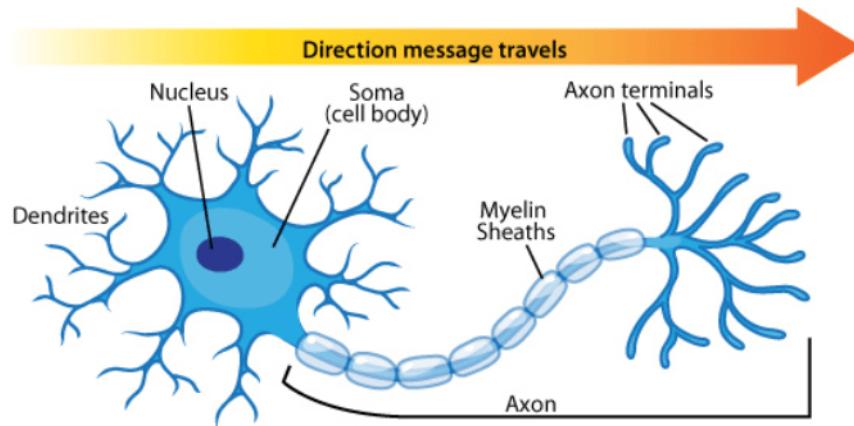
Neuron Anatomy



Hình 7.8: Cấu trúc một tế bào thần kinh

• Yêu

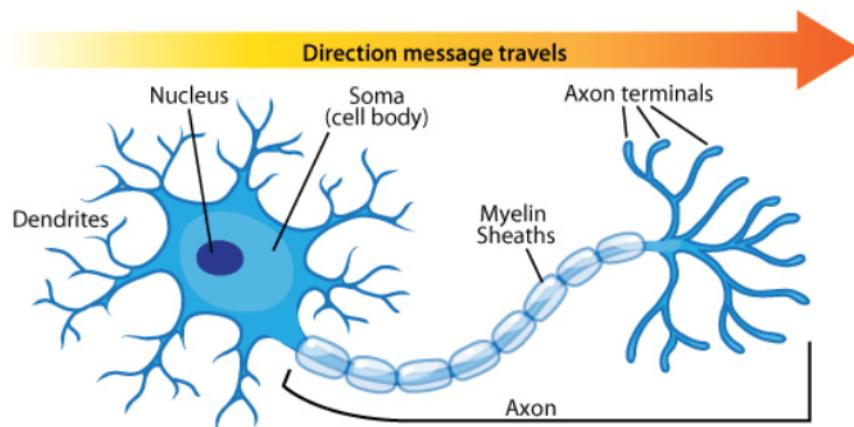
Neuron Anatomy



Hình 7.9: Cấu trúc một tế bào thần kinh

- **Hẹp**

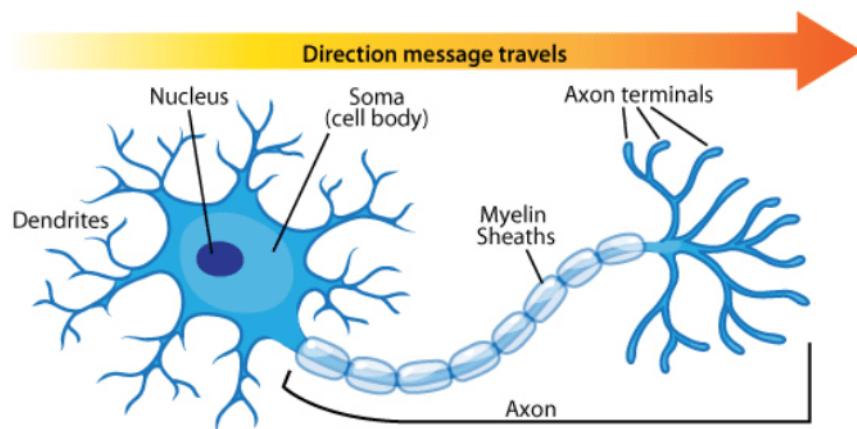
Neuron Anatomy



Hình 7.10: Cấu trúc một tế bào thần kinh

- **Rỗng**

Neuron Anatomy



Hình 7.11: Cấu trúc một tế bào thần kinh