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

ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA

----- ☆ -----

Số: _____ /BKĐT
Khoa: **Điện – Điện tử**
Bộ Môn: **Viễn thông**

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập – Tự do – Hạnh phúc.

----- ☆ -----

NHIỆM VỤ LUẬN VĂN TỐT NGHIỆP

1. HỌ VÀ TÊN: MSSV: LỚP :

2. NGÀNH: **ĐIỆN TỬ - VIỄN THÔNG**
3. Đề tài:
4. Nhiệm vụ (Yêu cầu về nội dung và số liệu ban đầu):

   .....
   .....
   .....
   .....
   .....
   .....
5. Ngày giao nhiệm vụ luận văn: .....
6. Ngày hoàn thành nhiệm vụ: .....
7. Họ và tên người hướng dẫn: Phần hướng dẫn

   ....
   ....

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

Tp.HCM, ngày..... tháng..... năm 20

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 gặp gỡ, thảo luận, đưa ra những vấn đề hay để rèn luyện chúng tôi khả năng giải quyết vấn đề. Thầy là người đã theo tường bước đi của chúng tôi, tận tình chỉ bảo, hướng dẫn chúng tôi từ khi làm đồ án, đề cương 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 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, chúng tôi 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 đề 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

Danh sách hình vẽ

Danh sách bảng

Danh mục viết tắt

dps	: dominant points
Dyn2S	: Dynamic two Strip
EB	: Early Break
EM	: Edge Map
FR	: Face Recognition
FBI	: Cục điều tra Liên bang Mỹ
HD	: Hausdorff Distance
LEM	: Line Edge Map
LHD	: Line Segment Hausdorff Distance
LSS	: Local Start Search
MHD	: Modified Hausdorff Distance
SURF	: Speeded Up Robust Features
SVM	: Support Vector Machine

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 bố cục 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 dường 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 [1], 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 [2]. 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 [3], 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 [4] 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 [5], và các phương pháp khác như: giải thuật khung xương [6], [7], [8]. Các bài khảo sát chi tiết có thể được tìm thấy ở [9] 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 [10]
- Nhận diện hành động con người dựa trên mô tả các đặc trưng góc 3D [11]
- Sử dụng nhiều thiết bị được gắn trên người [12]

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

1.3.1 Tìm hiểu

Để 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 Convoluntion 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.

1.3.2 Thực hiện

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 Bố cục trình bày

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 đề xuất để phân loại các từ trong ngôn ngữ ký hiệu; Cách xử lý dữ liệu đầu vào của mạng; cách huấn luyện mạng
Chương 5	Cách thức ứng dụng hoạt động.
Chương 6	Các thử nghiệm, kết quả và đánh g.iá
Chương 7	Tổng kết.

Chương 2

CƠ SỞ LÝ THUYẾT

2.1 MẠNG NEURAL NETWOK

Nguồn tham khao: https://dominhhai.github.io/vi/2018/04/nn-intro/

https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/

https://towardsdatascience.com/machine-learning-for-beginners-an-introduction-to-neural-networks-d49f22d238f9

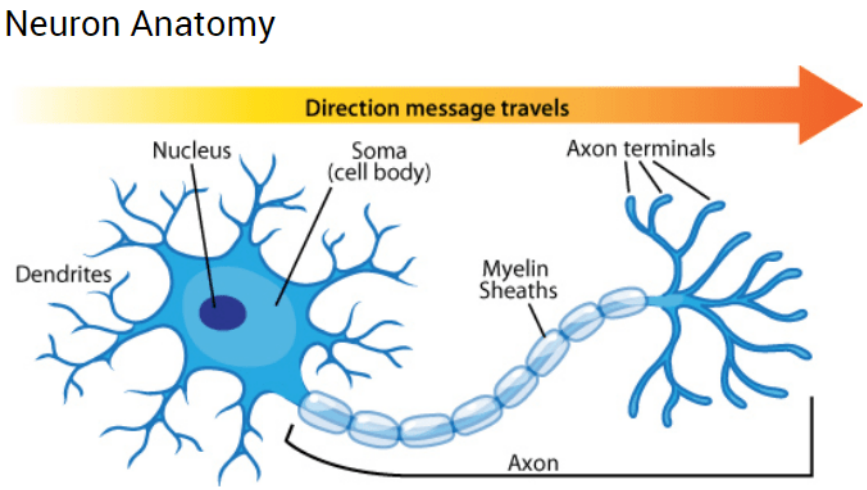
http://cs231n.github.io/neural-networks-1/ Con chó có thể phân biệt được người thân trong gia đình và người lạ hay đứa trẻ có thể phân biệt được các con vật. Những việc tưởng chừng như rất đơn giản nhưng lại cực kì khó để thực hiện bằng máy tính. Vậy sự khác biệt nằm ở đâu? Câu trả lời nằm ở bộ não với lượng lớn các nơ-ron thần kinh liên kết với nhau. Thế thì máy tính có nên mô phỏng lại mô hình ấy để giải các bài toán trên ???

Neural là tính từ của neuron (nơ-ron), network chỉ cấu trúc đồ thị nên neural network (NN) là một hệ thống tính toán lấy cảm hứng từ sự hoạt động của các nơ-ron trong hệ thần kinh.

Mạng noron nhân tạo (Neural Network - NN) là một mô hình tính toán được lấy cảm hứng từ cách mạng noron sinh học trong não người xử lý thông tin. Kết hợp với các kĩ thuật học sâu (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 thị giác máy tính và xử lý ngôn ngữ tự nhiên. Trong nội dung này, luận văn sẽ trình bày các lý thuyết cơ bản của mạng NN từ các thành phần cơ bản, kiến trúc mạng và các kỹ thuật huấn luyện (training) một mạng NN.

2.1.1 Hoạt động của các neuron sinh học



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 khoangr 10.000 neuron khác.

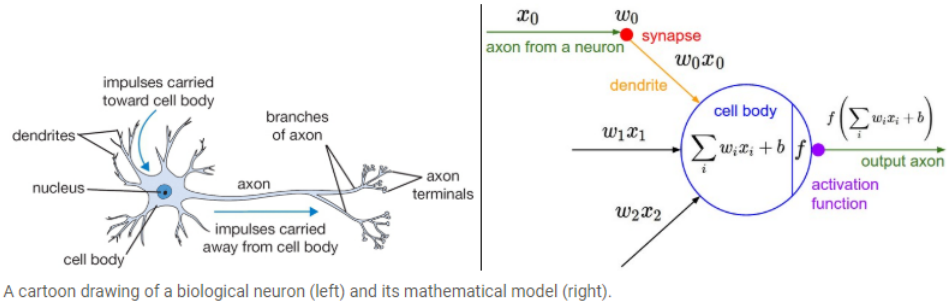
Ở mỗi neuron có phần thân (soma) chứa nhân, các tín hiệu đầu vào qua sợi nhánh (dendrites) và các tín hiệu đầu ra qua sợi trục (axon) kết nối với các neuron khác. Hiểu đơn giản mỗi neuron nhận dữ liệu đầu vào qua sợi nhánh và truyền dữ liệu đầu ra qua sợi trục, đến các sợi nhánh của các neuron khác.

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 trục đến các sợi nhánh của các neuron khác. => Ở mỗi neuron cần quyết định có kích hoạt neuron đấy hay không.

Tuy nhiên NN chỉ là lấy cảm hứng từ não bộ và cách nó hoạt động, chứ không phải

bắt chước toàn bộ các chức năng của nó. Việc chính của chúng ta là dùng mô hình đấy đi giải quyết các bài toán chúng ta cần.

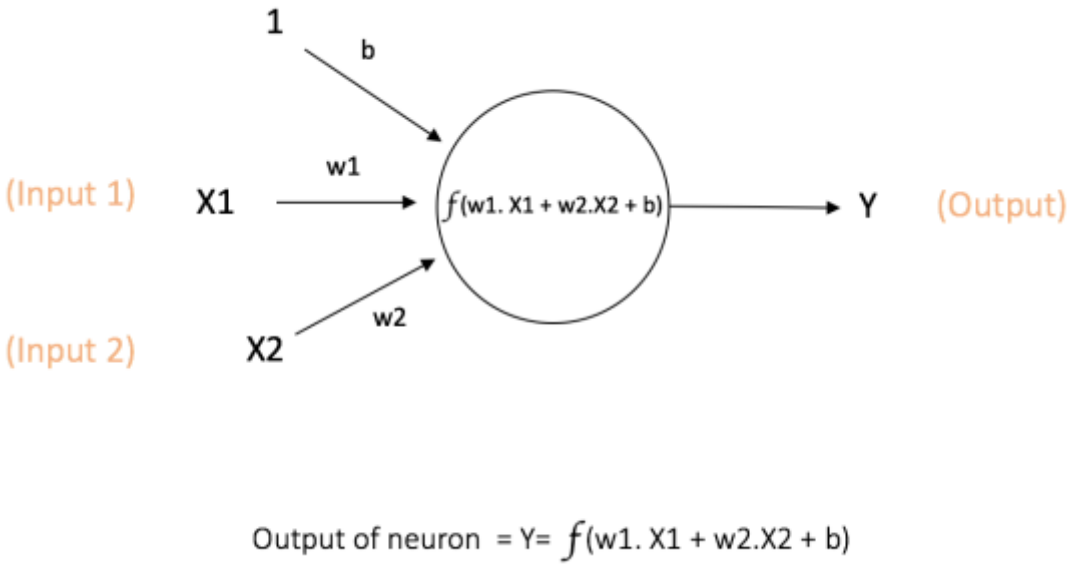
2.1.2 Perceptron



Hình 2.2: 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 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 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.



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 $X1$ và $X2$, 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 trọng 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].

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

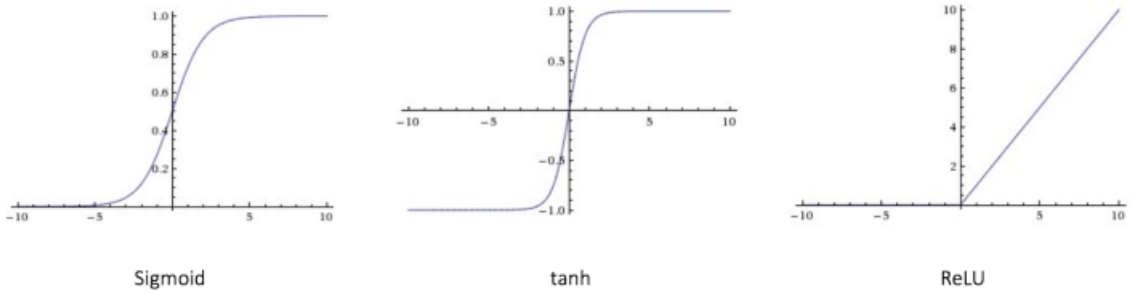
- **Tanh:** lấy giá trị ngõ vào thực và ép nó nằm trong giới hạn [-1, 1].

$$\tanh(x) = 2\sigma(2x) - 1 \tag{2.2}$$

- **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).

$$f(x) = \max(0, x) \tag{2.3}$$

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

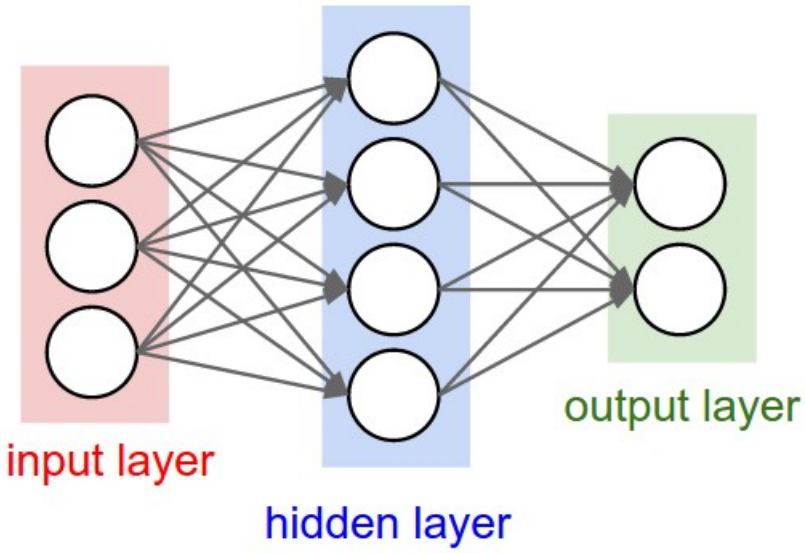


Hình 2.4: Đồ thị các hàm kích hoạt

Tầm quan trọng của bias:

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.

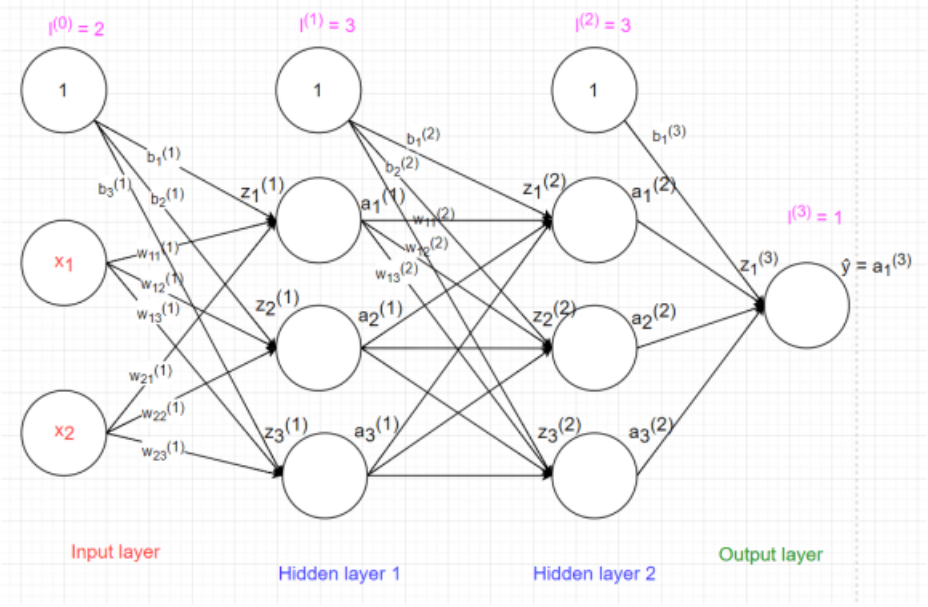
2.1.4 Hoạt động của mạng

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 $2 * 1$.

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

$$= (W^{(1)})^T * a^{(0)} + b^{(1)}$$

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

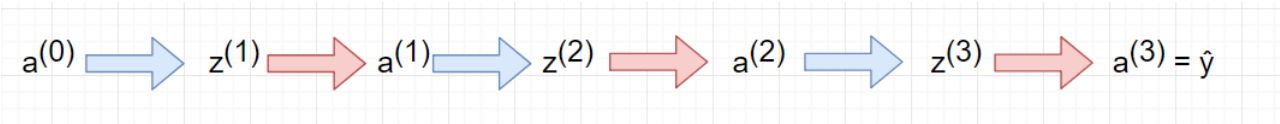
Tương tự ta có:

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

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

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

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



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.

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

Do đó:

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

$$= X * W^{(1)} + b^{(1)}$$

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 lấy 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:

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

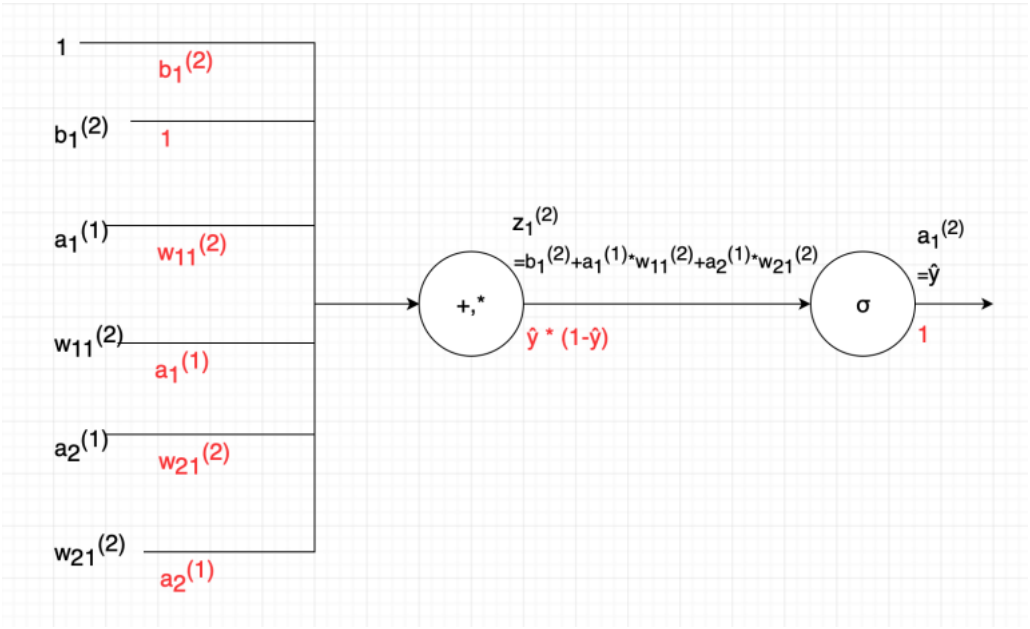
trong đó: $\hat{y}_i = a_1^{(2)} = \sigma(a_1^{(1)} * w_{11}^{(2)} + a_2^{(1)} * w_{21}^{(2)} + b_1^{(2)})$ là giá trị mà model dự đoán, còn y_i là giá trị thật của dữ liệu.

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

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 đồ thị 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:

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

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 $m * n$.

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

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.

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

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

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 đấy 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 $f(x) = a * x + b => \frac{df}{dx} = a$.

Kết hợp tất cả lại ta được:

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

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

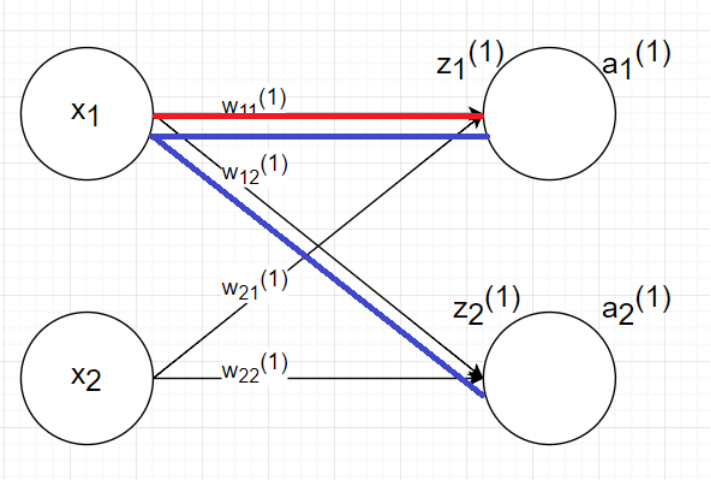
- Khi tính đạo hàm ngược lại qua bước activation thì dùng ($\otimes$).
- Khi có phép tính nhân ma trận thì dùng ($*$), nhưng đặc biệt chú ý đến **kích thước ma trận** và dùng **transpose** nếu cần thiết. Ví dụ: ma trận X kích thước $N * 3$, W kích thước $3 * 4$, $Z = X * W$ sẽ có kích thước $N * 4$ thì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

Tương tự:

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

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.

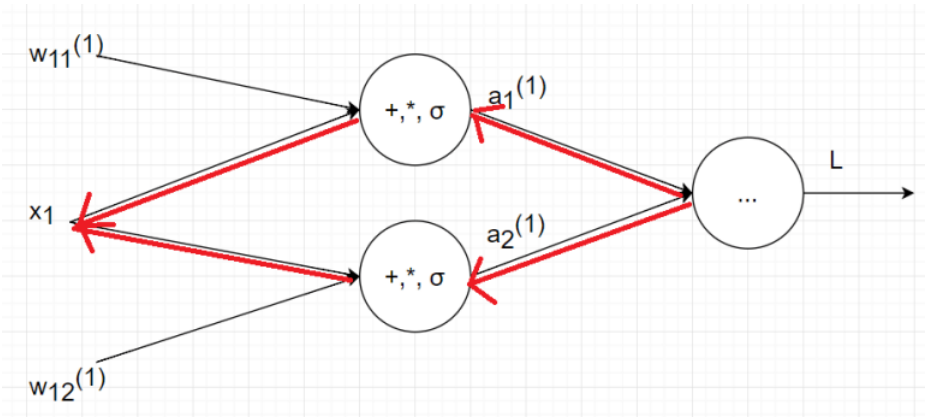
Giờ thử tính $\frac{\partial L}{\partial x_1}$, ở bài này thì không cần vì chỉ có 1 hidden layer, nhưng nếu nhiều hơn 1 hidden layer thì cần phải tính bước này để tính đạo hàm với các hệ số trước đó.



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 $a_2^{(1)}$.

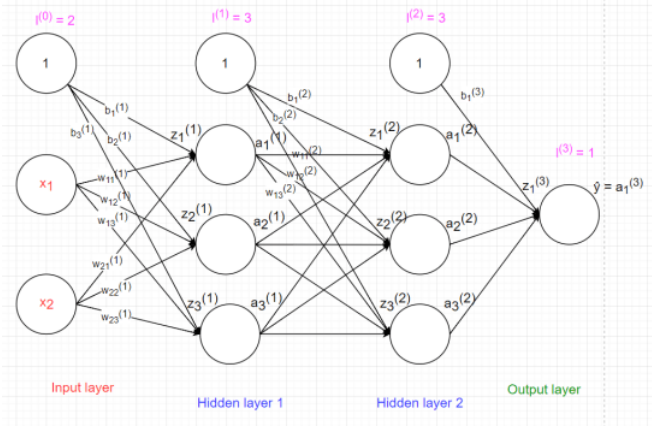


Hình 2.9: backpropagation tác động trong lớp ẩn

Do đó:

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

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

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

- **Bước 3:** Tính

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

  và tính

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

- **Bước 4:** Tính

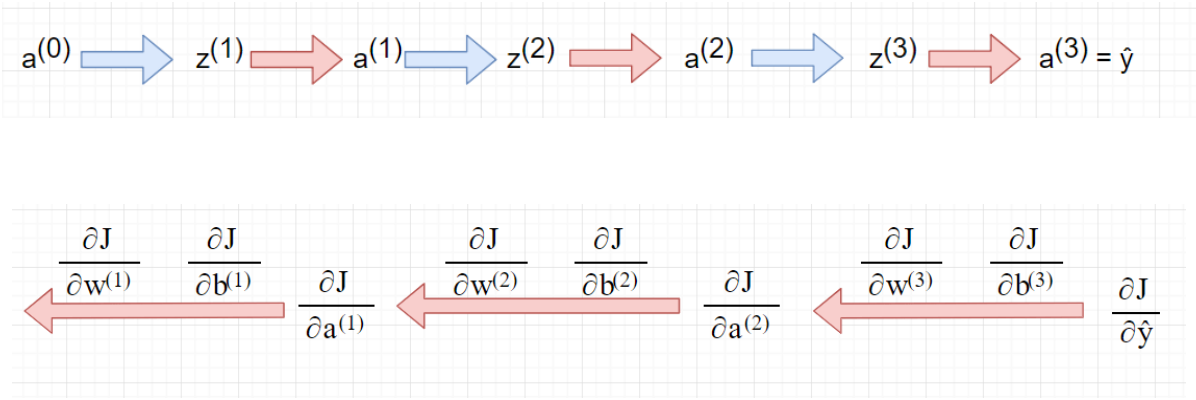
$$\frac{\partial J}{\partial \hat{W^{(1)}}} = (A^{(0)})^T * (\frac{\partial J}{\partial A^{(1)}} \otimes \frac{\partial A^{(1)}}{\partial Z^{(1)}}), \frac{\partial J}{\partial \hat{b^{(1)}}} = (sum(\frac{\partial J}{\partial A^{(1)}} \otimes \frac{\partial A^{(1)}}{\partial Z^{(1)}}))^T$$

  , 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 Feedfoward và Backpropagation sẽ diễn ra lần lượt như sau:



Hình 2.11: Feedforward và Backpropagation

(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 => 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 thướcây 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.

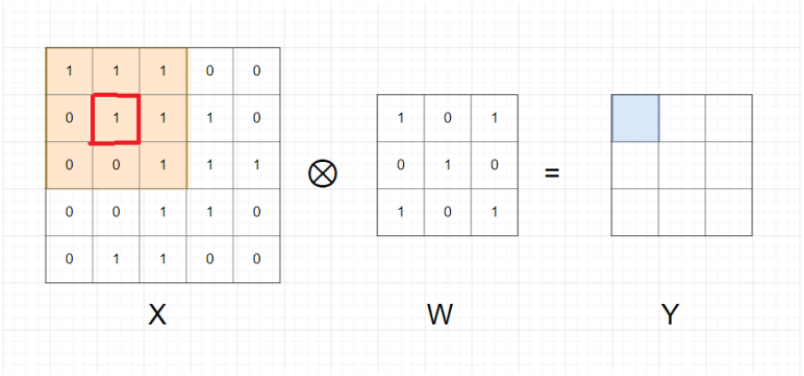
=> Do vậy ý tưởng sử dụng mạng Convolutional Neural Network ra đời. Áp dụng phép tính convolution vào layer trong neural network ta có thể giải quyết được vấn đề lượng lớn parameter mà vẫn lấy ra được các đặc trưng của ảnh.

2.2.1 Phép Tính Convolution

Để cho dễ hình dung mình sẽ lấy ví dụ trên ảnh xám, tức là 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 $1, 3, 5, 7, 9, \ldots$ Ví dụ kernel kích thước $3 * 3$.

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

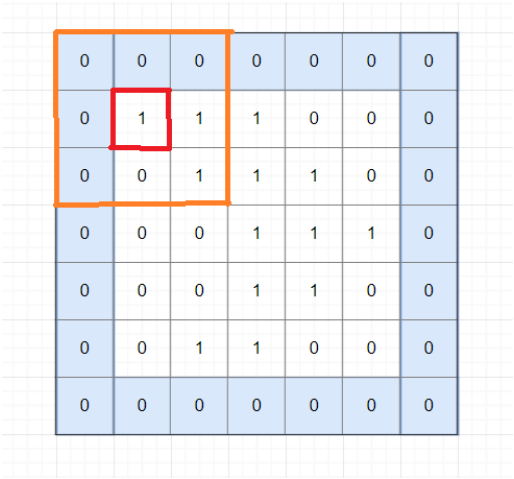
Với mỗi phần tử x_{ij} trong ma trận X lấy ra một ma trận có kích thước bằng kích thước của kernel W có phần tử x_{ij} làm trung tâm (đây là vì sao kích thước của kernel thường lẻ) gọi là ma trận A. Sau đó tính tổng các phần tử của phép tính element-wise của ma trận A và ma trận W, rồi viết vào ma trận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



Hình 2.12: Phép tính Convolution

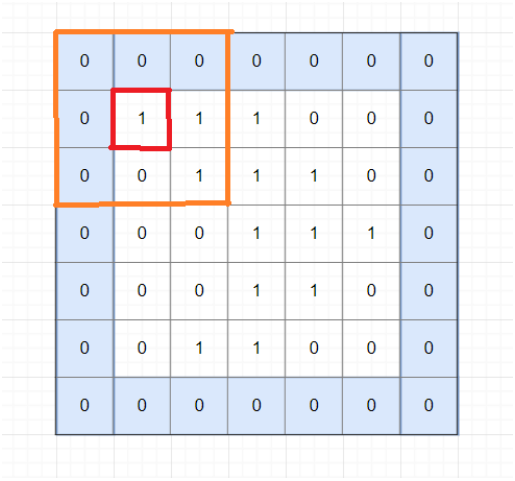
Ví dụ khi tính tại x_{22} (ô khoanh đỏ trong hình 2.12), ma trận A cùng kích thước với W, có x_{22} làm trung tâm có màu nền da cam như trong hình. Sau đó tính $y_{11} = sum(A \otimes W) = x_{11}*w_{11}+x_{12}*w_{12}+x_{13}*w_{13}+x_{21}*w_{21}+x_{22}*w_{22}+x_{23}*w_{23}+x_{31}*w_{31}+x_{32}*w_{32}+x_{33}*w_{33} = 4$. 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 2.13: Convolution feature có kích thước nhỏ hơn ảnh ban đầu

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

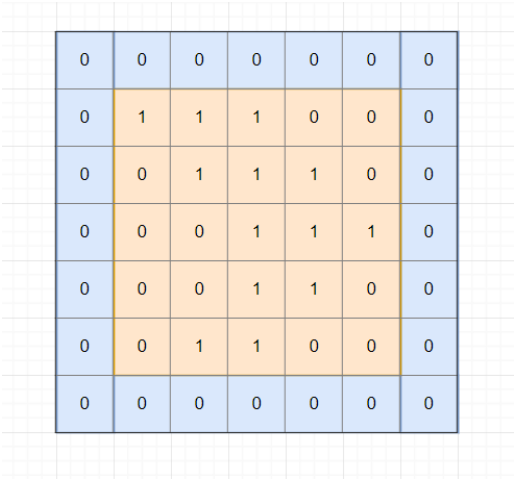


Hình 2.14: Ma trận X có viền 0 bên ngoài

Rõ ràng là giờ 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.

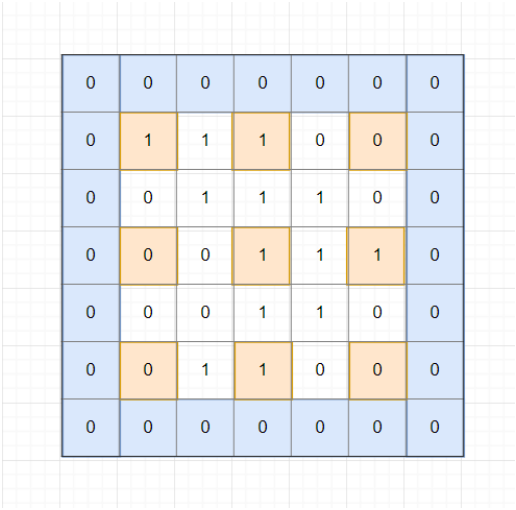
Phép tính này gọi là convolution với $padding = 1$. $Padding = k$ nghĩa là thêm k vector 0 vào mỗi phía của ma trận.

- **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 $stride = 1$.



Hình 2.15: $stride = 1, padding = 1$

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 $k = 2$.



Hình 2.16: $padding = 1, stride = 2$

Hiểu đơn giản là bắt đầu từ vị trí x_{11} sau đó nhảy k bước theo chiều dọc và ngang cho đến hết ma trận X.

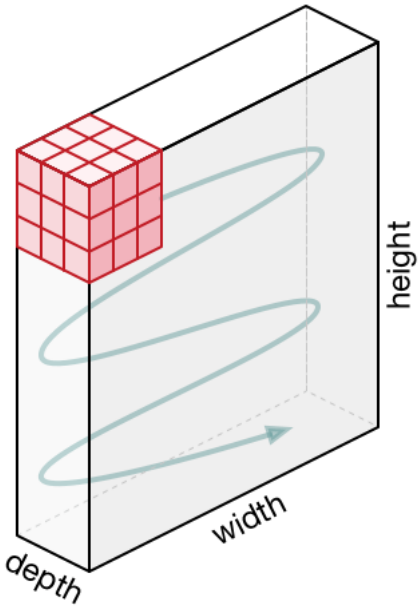
Kích thước của ma trận Y là $3 * 3$ đã giảm đi đáng kể so với ma trận X. Công thức tổng quát cho phép tính convolution của ma trận X kích thước $m * n$ với kernel kích thước $k * k$, $stride = s$, $padding = p$ ra ma trận Y kích thước

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

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 mở, làm nét ảnh; xác định các đường;... Mỗi kernel khác nhau thì sẽ phép tính convolution sẽ có ý nghĩa khác nhau.

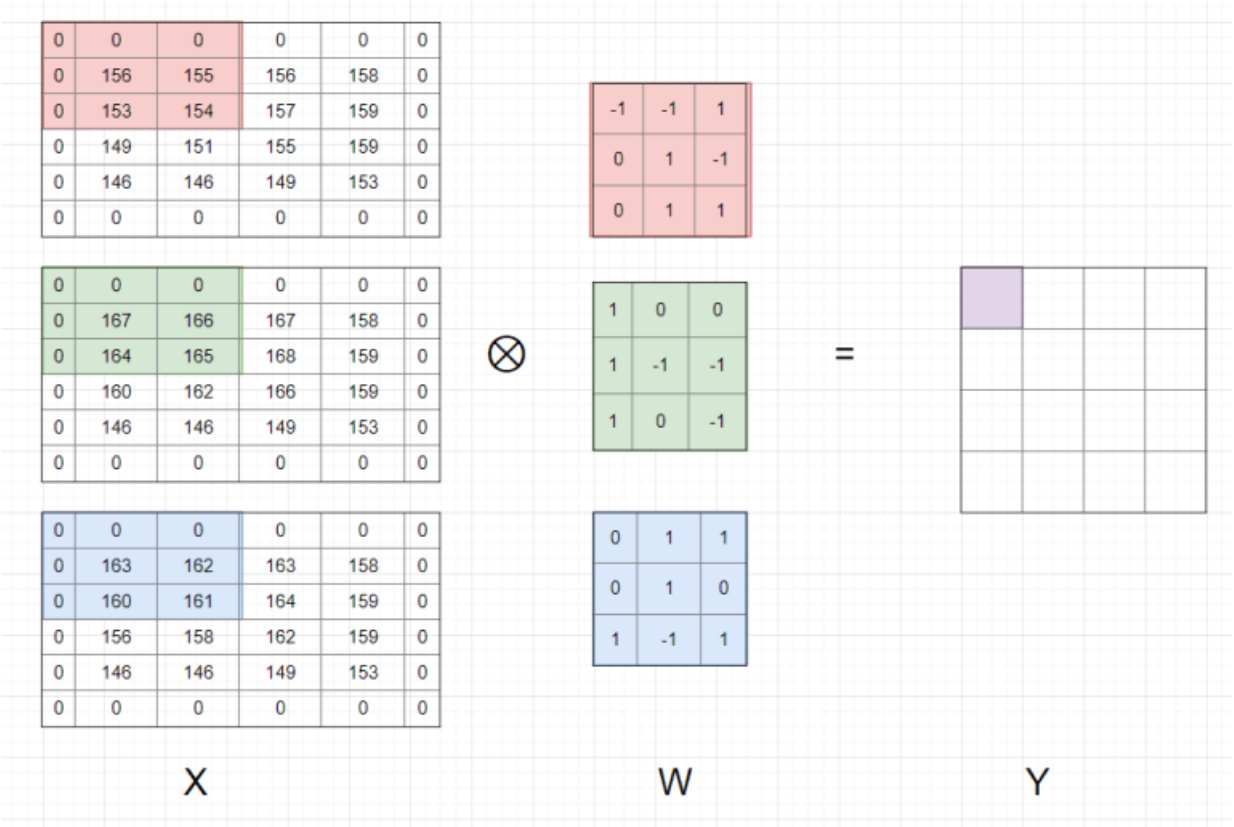
2.2.2 Phép convolution trong mạng Neuron Network

Với ảnh màu có tới 3 channels red, green, blue nên khi biểu diễn ảnh sẽ dưới dạng tensor 3 chiều. Nên ta cũng sẽ định nghĩa kernel là 1 tensor 3 chiều kích thước $k * k * 3$.



Hình 2.17: Phép tính convolution trên ảnh màu với k=3.

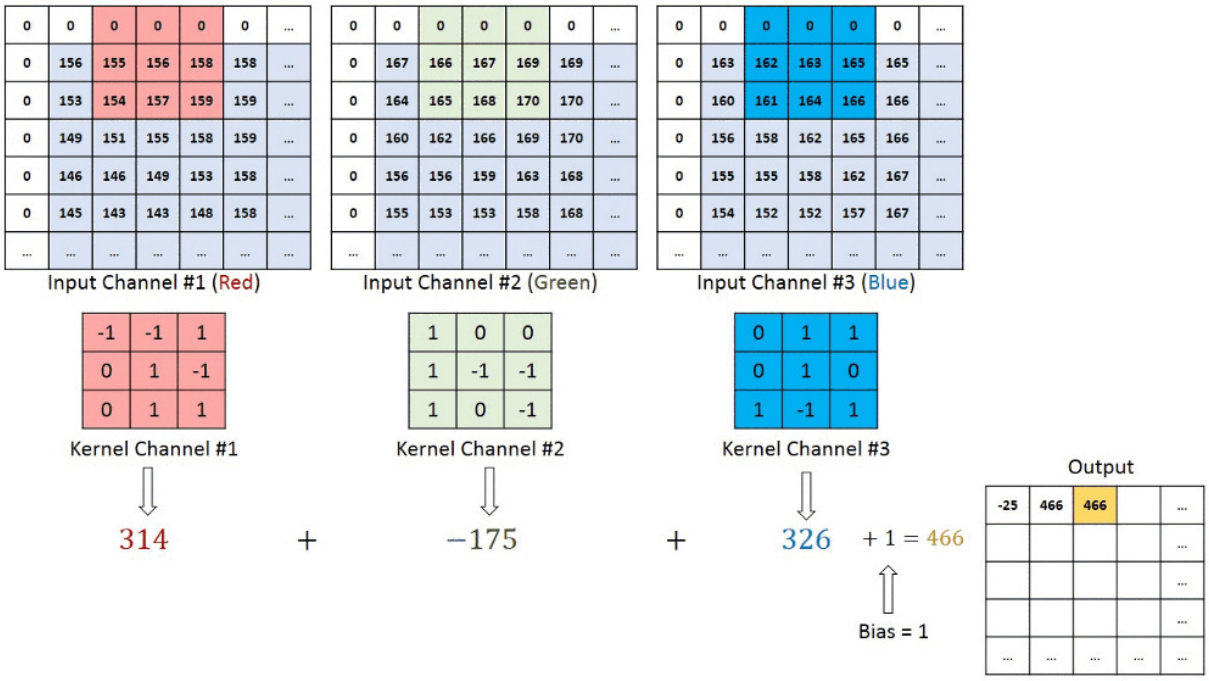
Ta định nghĩa kernel có cùng độ sâu (depth) với biểu diễn ảnh, rồi sau đó thực hiện di chuyển khối kernel tương tự như khi thực hiện trên ảnh xám.



Hình 2.18: 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}.

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

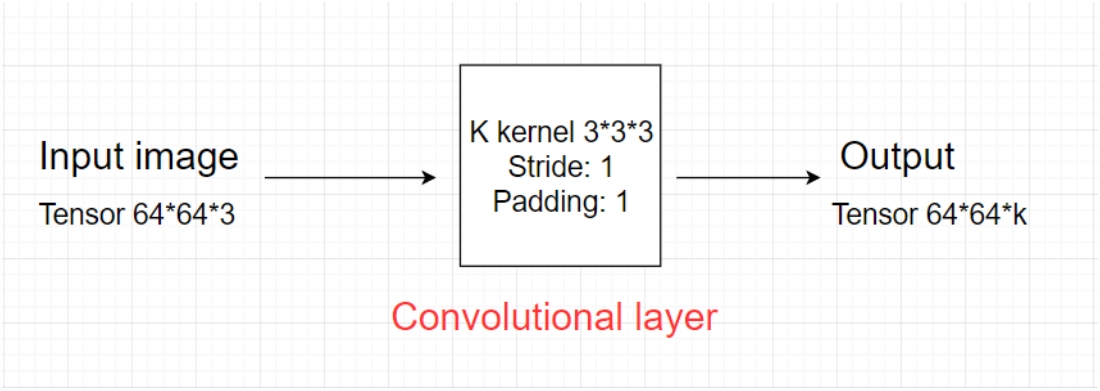


Hình 2.19: Thực hiện phép tính convolution trên ảnh màu

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.



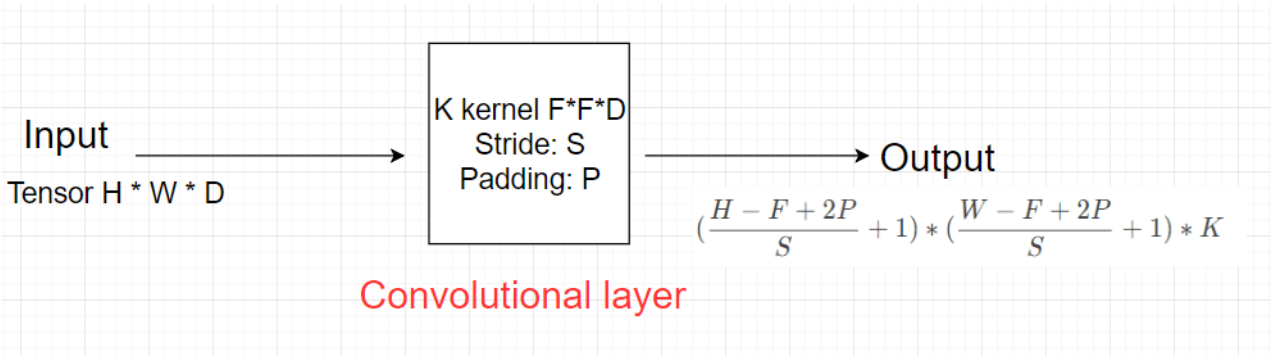
Hình 2.20: Convolutional layer đầu tiên

Output của convolutional layer đầu tiên sẽ thành input của convolutional layer tiếp theo.

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. => Output của layer là tensor 3 chiều có kích thước: $(\frac{H-F+2P}{S}+1) * (\frac{W-F+2P}{S}+1) * K$



Hình 2.21: Convolutional layer tổng quát

Lưu ý:

- Output của convolutional layer sẽ qua hàm activation function trước khi trở thành input của convolutional layer tiếp theo.

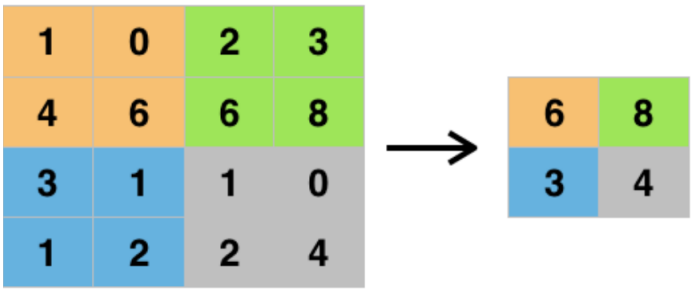
- Tổng số parameter của layer: Mỗi kernel có kích thước $F * F * D$ và có 1 hệ số bias, nên tổng parameter của 1 kernel là $F * F * D + 1$. Mà convolutional layer áp dụng K kernel => Tổng số parameter trong layer này là $K * (F * F * D + 1)$.

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 Pooling layer

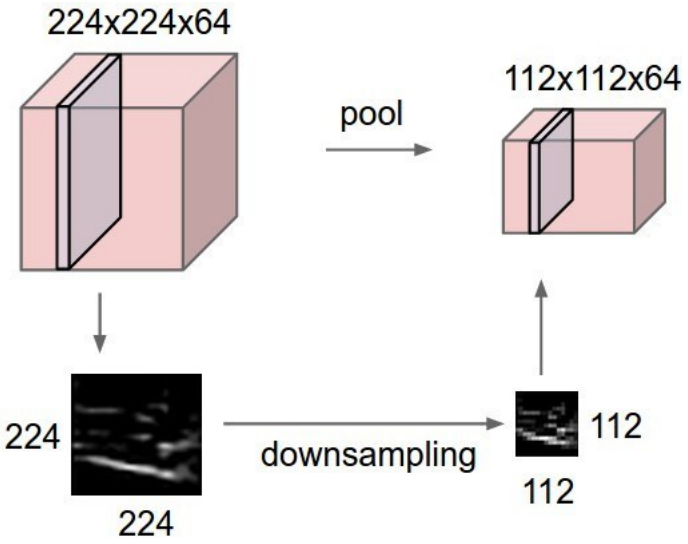
Pooling layer thường được dùng giữa các convolutional layer, để giảm kích thước dữ liệu nhưng vẫn giữ được các thuộc tính quan trọng. Kích thước dữ liệu giảm giúp giảm việc tính toán trong model.

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 2.22: Phép Pooling với $stride = 2, padding = 0$

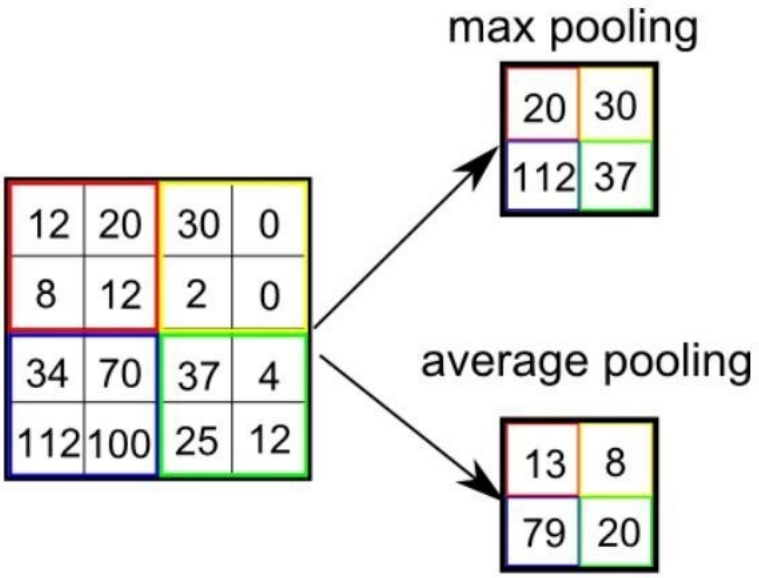
Nhưng hầu hết khi dùng pooling layer thì sẽ dùng $size = (2, 2)$, $stride = 2$, $padding = 0$. Khi đó output width và height của dữ liệu giảm đi một nửa, depth thì được giữ nguyên.



Hình 2.23: 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.

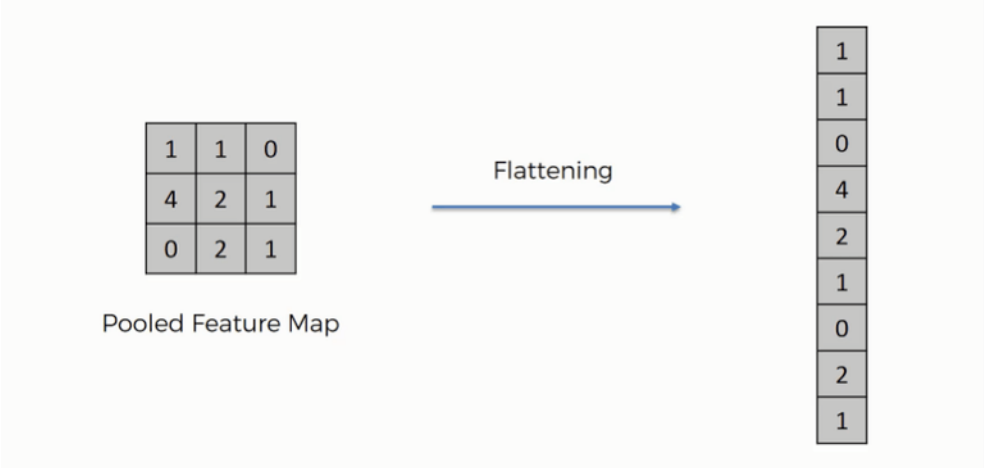


Hình 2.24: Max pooling và average pooling

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

2.2.4 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 $(H * W * D)$.

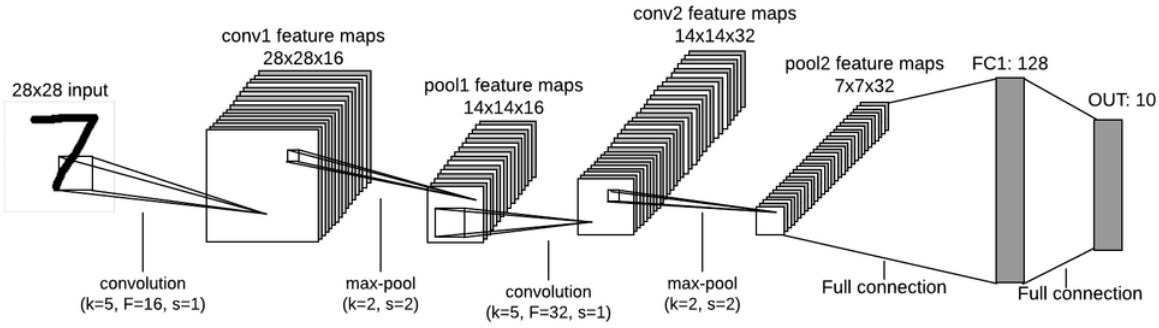


Hình 2.25: Phép Flatten biến tensor về 1 vector

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.

2.2.5 Kết luận

Tổng hợp lại, một mô hình mạng CNN sẽ có cấu trúc chung gần giống như hình **??**



Hình 2.26: Ví dụ mô hình 1 môn hình convolutional neural network

Input image -> Convolutional layer (Conv) + Pooling layer (Pool) -> Fully connected layer (FC) -> Output

Nguồn: https://www.easy-tensorflow.com/tf-tutorials/convolutional-neural-nets-cnns

Chương 3

ƯỚC TÍNH TỌA ĐỘ KHUNG XƯƠNG TRÊN ẢNH RGB BẰNG MẠNG MOBILENET-V2

Trong nội dung này, em sẽ trình bày hai phương pháp đề 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 mobilenet v2 được áp dụng trong đề tài vì tính khả thi, có thể ứng dụng được trong đời sống bằng cách tích hợp lên điện thoại thông minh.

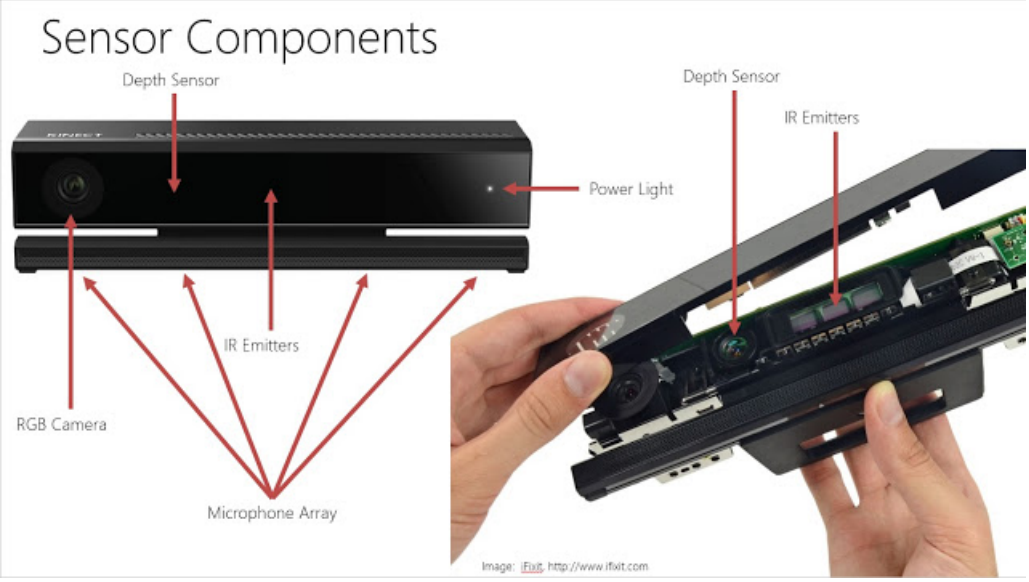
3.1 ƯỚC TÍNH DỰA TRÊN CAMERA ĐỘ SÂU KINECT

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

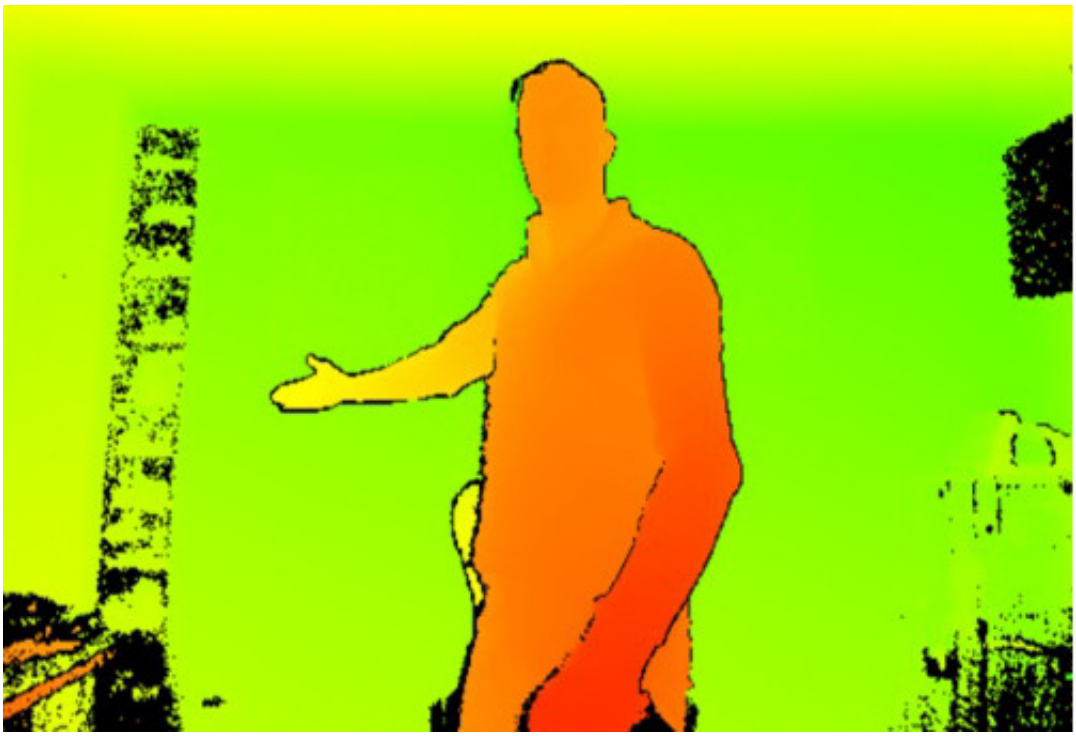
Nguồn : `https://www.ifixit.com`

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 $\sim$ 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 phần 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 nhận 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.zonetrigger.com/`)

3.1.2 Cơ chế trích xuất SJM từ Kinect

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" [13] 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.

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 phần 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 của chúng tối đạt giải nhất 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.

3.2.2 Kiến trúc mạng

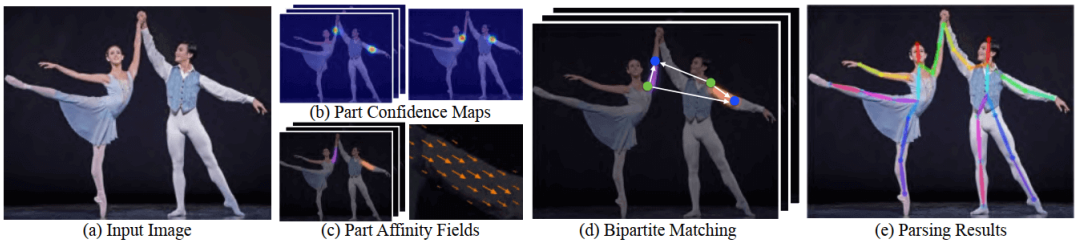


Fig. 2: Overall pipeline. (a) Our method takes the entire image as the input for a CNN to jointly predict (b) confidence maps for body part detection and (c) PAFs for part association. (d) The parsing step performs a set of bipartite matchings to associate body part candidates. (e) We finally assemble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

Hình 3.3: Tổng thể kiến trúc

Hình 2 mình họa toàn bộ nội dung phương pháp. Hệ thống lấy đầu vào, một ảnh màu có kích thước wxh (hình 2a) và tạo ra ngõ ra, tọa độ của những keyponts cho mỗi cá nhân trong ảnh (hình 2e). 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 2b) 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 2C). 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 2d) để tạo ra các keypoints 2D cho tất cả 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 b, và part affinity fields cho sự liên kết các phần, thể hiện trong hình c. Bước phân tích thể hiện một loạt những liên kết giữa hai điểm (liên kết lưỡng cực) để liên kết những phần cơ thể (d). Cuối cùng, chúng tôi lắp ráp chúng lại với nhau tạo thành những tư thế cơ thể hoàn chỉnh cho tất cả những người trong ảnh (e).

- 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.

- 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.

3.2.3 Các phần cụ thể

□ **Confidence Maps** Confidence Maps 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** $S = (S_1, S_2, .., S_J)$ 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 $L = (L_1, L_2, ..., L_c)$ 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**

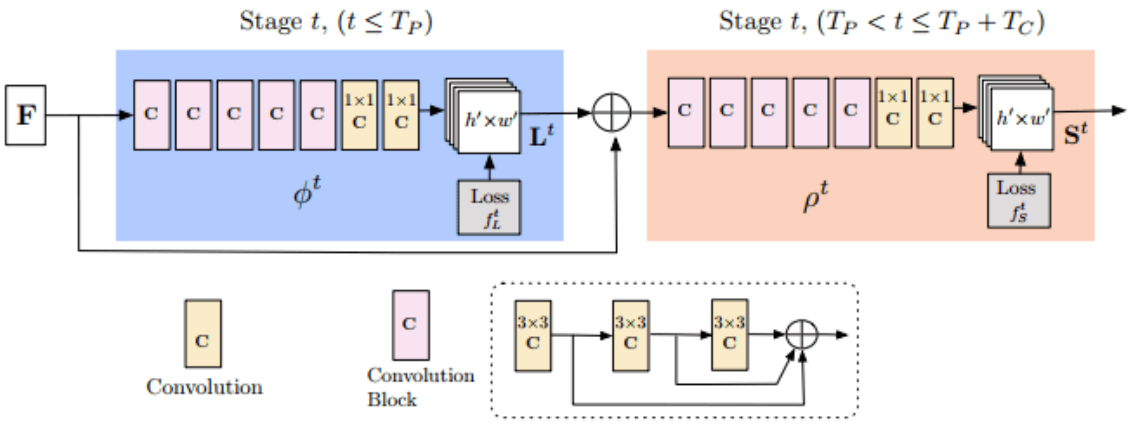


Fig. 3: Architecture of the multi-stage CNN. The first set of stages predicts PAFs $\mathbf{L}^t$, while the last set predicts confidence maps $\mathbf{S}^t$. The predictions of each stage and their corresponding image features are concatenated for each subsequent stage. Convolutions of kernel size 7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

Hình 3.4: 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%.

3.2.4 Phương pháp hồi quy

Chương 4

ĐỀ XUẤT: NHẬN DẠNG NGÔN NGỮ KÝ HIỆU TỪ TỌA ĐỘ KHUNG XƯƠNG BẰNG MẠNG DNN

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 nh anhtư 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 toạ độ khung xương được ước tính và nhận dạng cử chỉ thời gian thực. Nội dụng chương trình bày cấu trúc mạng neural network đề xuất, cách thu thập và xử lý dữ liệu cũng như sơ đồ hoạt động chương trinh.

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 nên 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, ẵm em, Sài Gòn, Vĩnh Long, đi bộ, mùa màng, đói bụng, yêu, ăn, biểu quyết, đứng yên, hẹp, 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.

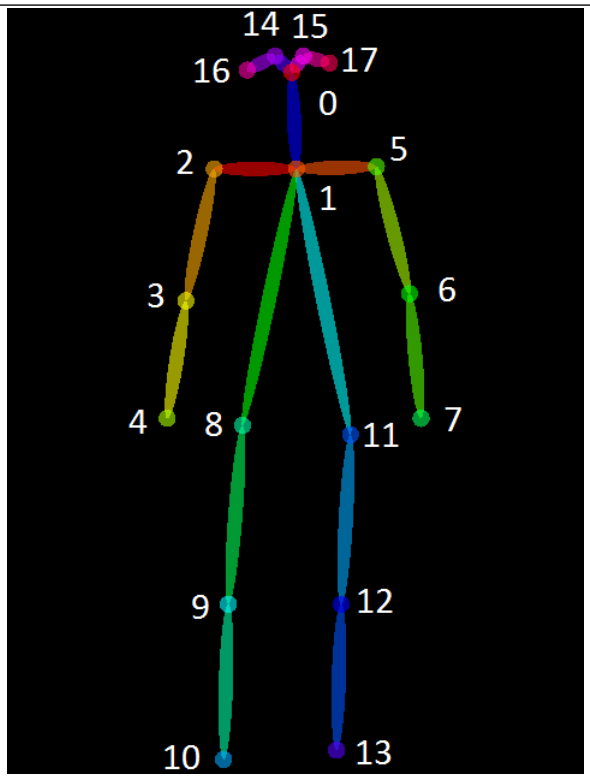
• Nếu một phần cơ thể bị che khuất bởi các vật thể thì việc xác định hành động của con người sẽ gặp khó khăn hơn rất nhiều so với trường hợp không bị che khuất, phạm vi luận văn không xem xét đến vấn đề này. Tuy nhiên đây là một điểm cần xem xét đến để có thể hoàn thiện hệ thống nhận diện cử chỉ trong 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 4.1.

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	Cùi chỏ phải
4	Cổ tay phải
5	Vai trái
6	Cùi chỏ trái
7	Cổ tay trái
8	Hông phải
9	Đầu gối phải
10	Cổ chân phải
11	Hông trái
12	Đầu gối trái
13	Cổ chân trái
14	Mắt phải
15	Mắt trái
16	Tai phải
17	Tai trái



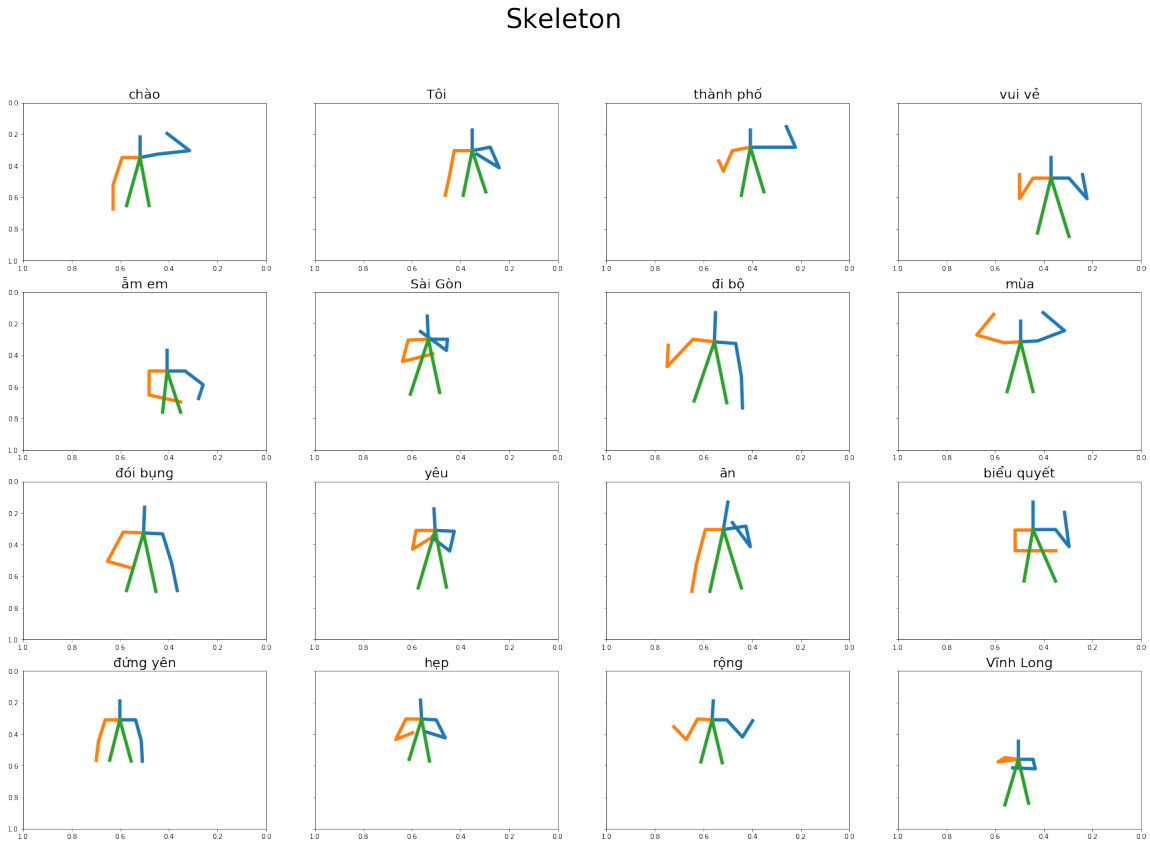
Hình 4.1: Sơ đồ khớp xương xuất ra từ mạng mobilenet

Ngôn ngữ ký hiệu với đặc trưng là phần trên cơ thể xuất hiện

4.3 Xử lý dữ liệu đầu vào

4.3.1 Loại bỏ các phần SJM dư thừa

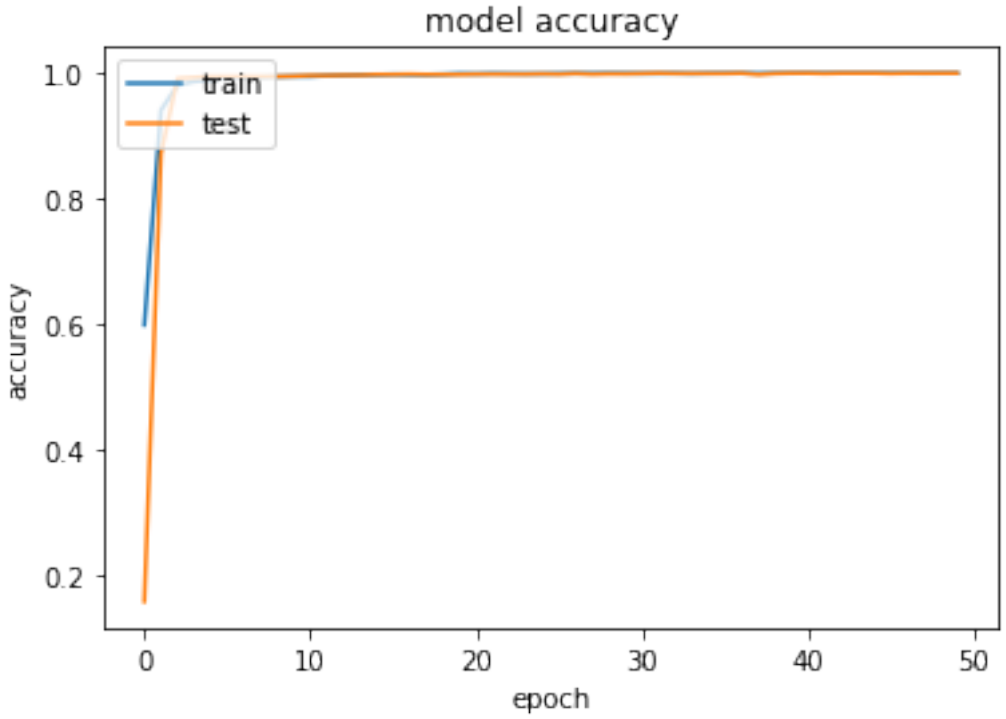
4.3.2 Chuẩn hóa SJM để phân loại đặc trưng



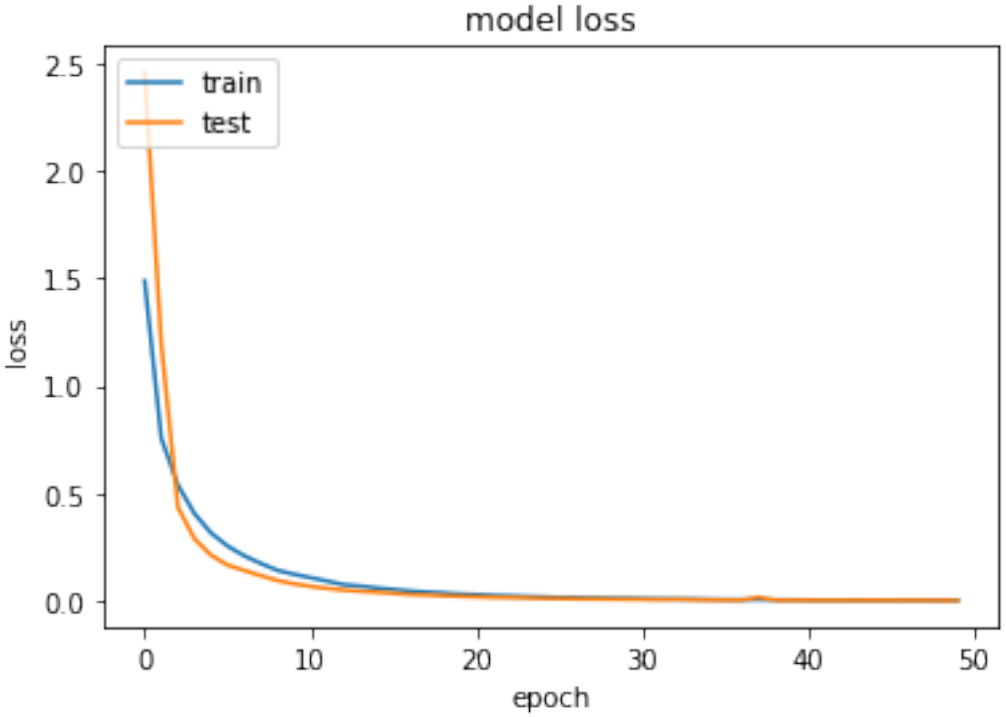
Hình 4.2: Dữ liệu khớp xương sau khi đã loại bỏ hết các phần không cần thiết

4.4 Cấu trúc mạng neural network đề xuất

4.5 Huấn luyện mạng

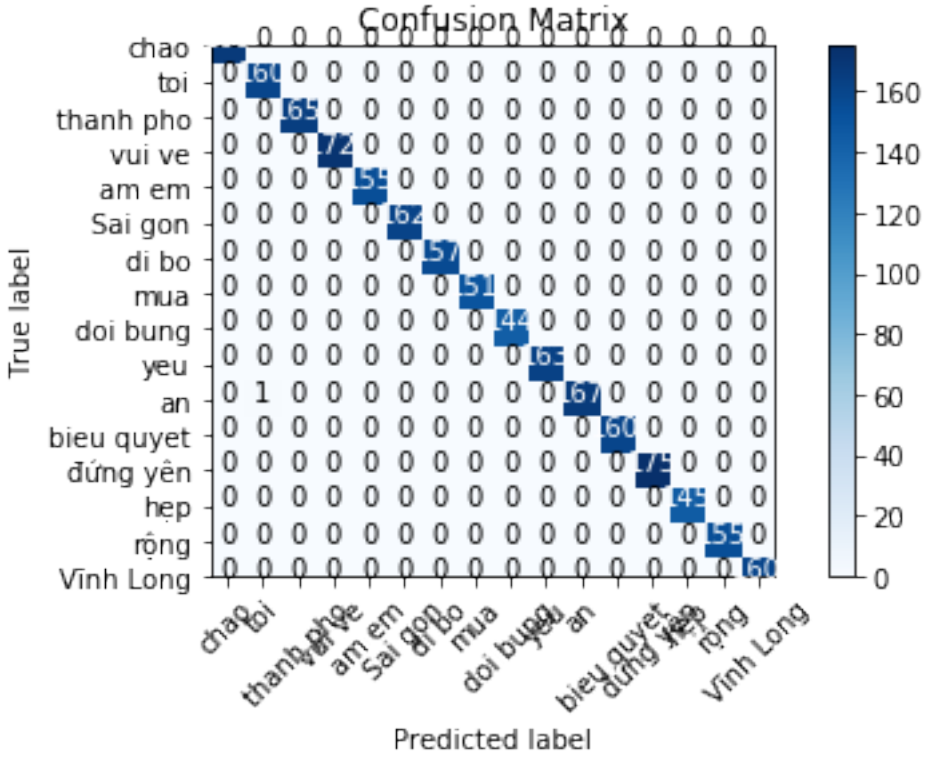


Hình 4.3: Accurracy của tập train và validate



Hình 4.4: Loss của tập train và validate

4.6 Kết quả



Hình 4.5: Confusion matrix của 16 lớp phân loại

Chương 5

CÁC THỬ NGHIỆM VÀ KẾT QUẢ

Chương 6

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1 Kết luận

Tài liệu tham khảo

[1] X. Chai, "Sign language recognition and translation with kinect," *IEEE Conf. on AFGR*, 2013.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

Phụ lục

Các từ nhận dạng được