

ĐẠI HỌC THĂNG LONG
KHOA HỌC MÁY TÍNH



KHÓA LUẬN TỐT NGHIỆP

**TÀI LIỆU 2
PHƯƠNG PHÁP NHẬN DIỆN VÀ XÁC THỰC KHUÔN MẶT**

GIẢNG VIÊN HƯỚNG DẪN
TS. MAI THÚY NGA
CN. NGUYỄN ĐỨC THẮNG

SINH VIÊN THỰC HIỆN
BÙI TUẤN ANH - A26820
NGUYỄN ĐĂNG KHÁNH - A26181
NGUYỄN TIẾN HOÀNG - A27193

HÀ NỘI - 2019

GIỚI THIỆU CHUNG

Trong nhiều năm qua, trường Đại học Thăng Long đã và đang được mở rộng về quy mô, đa dạng hóa cơ sở vật chất nhằm phục vụ một cách tốt nhất cho quá trình học tập của toàn bộ sinh viên trong trường. Đi kèm với sự phát triển về công tác đào tạo trực tuyến, Trường Đại học Thăng Long cũng dần mở rộng tiếp cận với xu hướng ứng dụng công nghệ cao trong quản lý giáo dục. Đặc biệt là bài toán quản lý điểm danh sinh viên.

Hiện nay, việc điểm danh đầu giờ tốn rất nhiều thời gian với những lớp họp như hội trường có thể với lượng 200 sinh viên và có thể điểm danh hộ. Hoặc trong các kỳ thi cuối kỳ với số lượng sinh viên liên tục vào phòng thi để đảm bảo đúng thời gian quy định các sinh viên thường được nhanh chóng điểm danh. Tuy nhiên việc này còn hạn chế bởi khả năng xác thực danh tính từ con người còn hạn chế và có khả năng sai sót. Nhận thức được điều này, nhà trường đã chủ động nâng cấp hạ tầng và ứng dụng công nghệ hiện đại vào bài toán quản lý điểm danh sinh viên.

Là một nhóm gồm các giảng viên bộ môn Tin và sinh viên Khoa Toán Tin đã quyết định cùng nhau xây dựng dự án xác thực sinh viên bằng công nghệ nhận diện khuôn mặt. Cùng thời điểm, nhóm sinh viên gồm ba thành viên chúng em cũng thực hiện khoá luận tốt nghiệp phát triển hệ thống – Xây dựng hệ thống quản lý điểm danh sinh viên bằng công nghệ nhận diện khuôn mặt tại Đại học Thăng Long. Mục tiêu của khoá luận là nhằm giải quyết bài toán tồn đọng trong việc quản lý tình trạng điểm danh sinh viên một cách linh động nhất, ngoài ra khoá luận này còn giúp chúng em đạt mục tiêu sử dụng những kiến thức đã học tập và tích lũy được trong quá trình học để nắm bắt thành thục các công nghệ lập trình và các công nghệ trí tuệ nhân tạo.

Với mục tiêu phát triển phần mềm theo đúng quy trình, chúng em đã viết đầy đủ tài liệu cho từng giai đoạn, bộ tài liệu gồm 4 tài liệu như sau:

- Tài liệu 1: Tài liệu tổng quan;
- Tài liệu 2: Phương pháp nhận diện và xác thực khuôn mặt;
- Tài liệu 3: Phân tích thiết kế hệ thống;
- Tài liệu 4: Công nghệ sử dụng và triển khai;

Nhận diện khuôn mặt là một trong những lĩnh vực của trí tuệ nhân tạo, được phát triển mạnh mẽ và có nhu cầu. Dựa vào lịch sử phát triển và đóng góp rất nhiều thuật toán của các nhà khoa học trên thế giới về xử lý nhận diện khuôn mặt.

Đây tài liệu phân tích cơ sở lý thuyết và các thuật toán được sử dụng trong khoá luận. Nội dung tài liệu gồm 6 chương:

Chương 1: Tổng quan về bài toán nhận diện khuôn mặt. Chương này của khoá luận trình bày khái quát về bài toán nhận diện khuôn mặt. Từ đó thiết lập động lực và mục tiêu của khoá luận.

Chương 2: Cở sở lý thuyết. Chương này trình bày chi tiết các khái niệm cơ sở được sử dụng trong tài liệu.

Chương 3: Mạng nơ-ron. Chương này trình bày cấu trúc, thành phần và cách hoạt động của mạng nơ-ron và mạng nơ-ron tích chập. Một số vấn đề gặp phải khi xây dựng mô hình học máy và phương pháp giải quyết.

Chương 4: Phương pháp nhận diện và xác thực khuôn mặt. Chương này trình bày các phương pháp được sử dụng để phát triển hệ thống nhận diện và xác thực khuôn mặt.

Chương 5: Thực nghiệm. Chương này trình bày cài đặt phương pháp huấn luyện và kết quả thực nghiệm.

Chương 6: Kết luận và hướng phát triển. Chương này tóm tắt kết quả đạt được và kế hoạch phát triển trong tương lai.

Mục lục

Danh sách hình vẽ	iv
Danh sách hình vẽ	1
1 Tổng quan về bài toán nhận diện khuôn mặt	1
1.1 Tổng quan về công nghệ học sâu	1
1.2 Bài toán xác thực khuôn mặt	1
1.3 Một số nghiên cứu	2
1.3.1 Biểu đồ mô hình nhị phân cục bộ	2
1.3.2 Haar Cascades	2
1.3.3 FaceNet	3
1.3.4 ArcFace	4
2 Cơ sở lý thuyết	6
2.1 Đại số tuyến tính	6
2.1.1 Ma trận	6
2.1.2 Norm	7
2.2 Giải tích	8
2.2.1 Đạo hàm	8
2.3 Xác suất	9
2.3.1 Các phép toán xác suất	9
2.3.2 Xác suất hậu nghiệm - Bayes	10
2.3.3 Ước lượng tham số bằng cực đại khả dĩ	10
2.4 Học máy	11
2.4.1 Khái niệm	11
2.4.2 Các phương pháp học	11
2.4.3 Xây dựng sản phẩm ứng dụng học máy	12
2.4.4 Các vấn đề trong học máy	13
2.4.5 Phương pháp giải quyết	14
2.4.6 Phương pháp đánh giá mô hình	18
2.4.7 Một số hàm tính lỗi phổ biến	22
2.4.8 Một số tham số huấn luyện	24
2.4.9 Một vài phương pháp tối ưu	25
3 Mạng Nơ-ron	29
3.1 Tổng quan	29
3.1.1 Định nghĩa	29
3.2 Cấu tạo của nơ-ron	29
3.2.1 Nơ-ron sinh học	29
3.2.2 Nơ-ron nhân tạo	30
3.3 Mạng nơ-ron nhiều tầng	32

3.3.1	Kiến trúc chung	32
3.3.2	Mạng lan truyền thẳng	33
3.3.3	Hàm mất mát	34
3.3.4	Thuật toán lan truyền ngược	36
3.4	Mạng nơ-ron tích chập	38
3.4.1	Tổng quan	38
3.4.2	Tầng tích chập - Convolutional layer	38
3.4.3	Tầng tổng hợp	43
3.4.4	Tầng Fully-Connected (Fully-Connected layer)	44
3.4.5	Một số mạng nơ-ron tích chập điển hình	45
4	Phương pháp nhận diện và xác thực khuôn mặt	48
4.1	Giới thiệu tổng quan	48
4.2	Kiến trúc phương pháp	48
4.2.1	Pha huấn luyện	49
4.2.2	Pha xác thực	51
4.3	Module 1: Nhận diện khuôn mặt	51
4.3.1	Phương pháp	51
4.3.2	Kiến trúc thuật toán	52
4.4	Module 2: Xác thực khuôn mặt	53
4.4.1	Phương pháp	53
4.4.2	Kiến trúc thuật toán	54
4.5	Module 2: Theo dõi khuôn mặt	57
4.5.1	Phương pháp	57
4.5.2	Kiến trúc thuật toán	58
5	Thực nghiệm	61
5.1	Tập dữ liệu	61
5.2	Môi trường huấn luyện	61
5.3	Phương pháp	61
5.3.1	Phân chia tỉ lệ dữ liệu	62
5.3.2	Nhận diện và cắt khuôn mặt	62
5.3.3	Căn chỉnh khuôn mặt	62
5.3.4	Trích rút và phân loại đặc trưng	63
5.4	Kết quả thực nghiệm	65
6	Kết luận và hướng phát triển	66
6.1	Kết luận	66
6.2	Hướng phát triển	66

Danh sách hình vẽ

1.1	Nhận diện bằng biểu đồ mô hình nhị phân cục bộ	2
1.2	Sliding window	3
2.1	Sơ đồ quy trình học máy	11
2.2	Các phương pháp trong học máy	12
2.3	Machine Learning workflow	13
2.4	Một số vấn đề thường gặp khi xây dựng mô hình	14
2.5	Early stopping	15
2.6	Dánh giá chéo k-fold	16
2.7	Ví dụ tăng cường dữ liệu	16
2.8	Dropout với $p = 0.5$	17
2.9	Ma trận nhầm lẫn	18
2.10	Ma trận nhầm lẫn với lớp cat	19
2.11	Một số phương pháp khác	20
2.12	Tỷ lệ TP so với FP ở các ngưỡng phân loại khác nhau	21
2.13	AUC	21
2.14	AUC	21
2.16	Hàm sigmoid	23
2.17	Hàm ReLU	24
2.18	Tốc độ tối ưu của một số phương pháp	25
2.19	Dồ thị	27
2.20	SGD kết hợp với momentum	28
3.1	Mạng nơ-ron	29
3.2	Minh họa cấu tạo nơ-ron sinh học	30
3.5	Cấu trúc chung của mạng nơ-ron	33
3.6	Mạng lan truyền thẳng	34
3.7	Lan truyền ngược	36
3.8	Ví dụ lan truyền ngược tại một nơ-ron	37
3.9	Cách máy tính nhìn một hình ảnh	38
3.11	Tầng tích chập ¹	39
3.12	Tích chập ²	39
3.13	Bộ lọc phát hiện biên	40
3.14	Kết quả sử dụng bộ lọc sobel	40
3.15	Bước trượt bằng 2 ³	40
3.16	Lề bằng 1, bước trượt bằng 2 ⁴	41
3.17	Tích chập khối ⁵	42
3.18	Tính chi tiết trong tích chập khối ⁶	43
3.19	Tầng giảm số chiều với bước trượt bằng 2	44
3.20	Tầng Fully Connected	45
3.21	Mạng LeNet-5	45

3.22	Mạng AlexNet	46
3.23	Mạng LeNet-5	47
4.1	Kiến trúc tổng quan thuật toán	49
4.7	Image Pyramid	52
4.8	Embedding	53
4.12	Khoảng cách Euclide giữa hai điểm	56
4.13	Khoảng cách Euclide giữa hai vector đặc trưng	57
4.14	Theo dõi khuôn mặt sinh viên khi vào lớp	58
5.1	Bộ dữ liệu khuôn mặt của sinh viên	61
5.2	Nhận diện và cắt khuôn mặt	62
5.4	Khuôn mặt sau khi căn chỉnh	63
5.6	Biểu đồ mô tả giá trị mắt, độ chính xác trong quá trình huấn luyện	65

Bảng 1: Ký Hiệu

Ký hiệu	Ý nghĩa
a	Một vector
A	Một ma trận
a_i	Phần tử thứ i của vector a
a_{ij}	Phần tử ở dòng i cột j của ma trận trật A
A_i	Dòng thứ i của ma trận A
\mathbf{A}^T	Ma trận chuyển vị của ma trận A
$f(x)$	Hàm số f với biến là x
$f'(x)$	Đạo hàm của hàm $f(x)$
f'_x hoặc $\frac{df}{dx}$	Đạo hàm riêng của hàm f với x

Bảng 2: Ký hiệu riêng tại phần mạng nơ-ron

Ký hiệu	Ý nghĩa
X	Tập dữ liệu (quan sát) đầu vào (inputs)
$\mathbf{x}_{(i)}$	Dữ liệu thứ i của đầu vào
y	Đầu ra thực tế (labels)
w	Trọng số (weights)
l	Tổng số layer
L	Giá trị mất mát
$\hat{\mathbf{y}}$	Đầu ra dự đoán
SGD	Static gradient descent
ReLu	Rectified linear unit
CNN	Convolutional neural networks
FC	Full-connected layer

Chương 1

Tổng quan về bài toán nhận diện khuôn mặt

1.1 Tổng quan về công nghệ học sâu

Học sâu, hay deep learning, là thuật toán dựa trên ý tưởng máy móc học hỏi như não bộ của con người. Là phương pháp trong học máy, sử dụng nhiều tầng mạng nơ ron để trích xuất đặc trưng và phân loại đặc trưng. Deep learning được ứng dụng trong nhận diện hình ảnh, nhận diện giọng nói, xử lý ngôn ngữ tự nhiên,... Với nguồn dữ liệu khổng lồ và sự phát triển của hạ tầng tính toán thì deep learning được ứng dụng rất mạnh mẽ và đạt kết quả chính xác cao so với các phương pháp lập trình truyền thống.

Những năm gần đây, chúng ta đã chứng kiến được rất nhiều thành tựu vượt bậc trong ngành Thị giác máy tính (Computer Vision). Thị giác máy tính là một lĩnh vực bao gồm các phương pháp xử lý ảnh kỹ thuật số, phân tích và nhận diện. Việc phát triển lĩnh vực này từ ý tưởng tương tự khả năng thị giác con người bởi sự nhận diện và hiểu biết một hình ảnh nhưng mang tính số học.

Mạng nơ ron tích chập (Convolution Neural Network) là một trong những mô hình deep learning giúp chúng ta xây dựng được thuật toán thông minh trong thị giác máy tính. Đây cũng là phương pháp trọng tâm trong khoá luận này.

1.2 Bài toán xác thực khuôn mặt

Hệ thống xác thực khuôn mặt là một hệ thống tự động xác thực danh tính khuôn mặt một người từ một hình ảnh kỹ thuật số hoặc một khung hình video dựa vào những đặc trưng. Một trong những cách để thực hiện điều này là so sánh các đặc trưng khuôn mặt trong khung hình với các khuôn mặt trong bộ phân loại đặc trưng máy đã học.

Nhận diện khuôn mặt là một lĩnh vực nghiên cứu của ngành thị giác máy tính. Các hãng lớn trong ngành công nghệ của Mỹ như Facebook, Google, Apple, Amazon đã và đang đưa công nghệ nhận diện khuôn mặt vào sản phẩm của họ. Như Facebook nổi tiếng với ứng dụng tự động tag tài khoản bạn bè, Apple ứng dụng vào mở điện thoại thông minh bằng khuôn mặt, Amazon cũng đã thử nghiệm nhận diện khuôn mặt trong các trung tâm mua sắm để tự động thanh toán. Trung Quốc đưa công nghệ này vào giám sát và đánh giá độ tín nhiệm công dân các thành phố lớn như dự án Skynet với hơn 600 triệu camera giám sát. Tại Việt Nam nổi bật như hệ thống nhận diện khuôn mặt của EyeqTech giúp các doanh nghiệp đo lường quảng cáo, cung cấp thông tin về người xem quảng cáo như: giới tính, độ tuổi, xem bao nhiêu lần, cảm xúc của người xem,...

1.3 Một số nghiên cứu

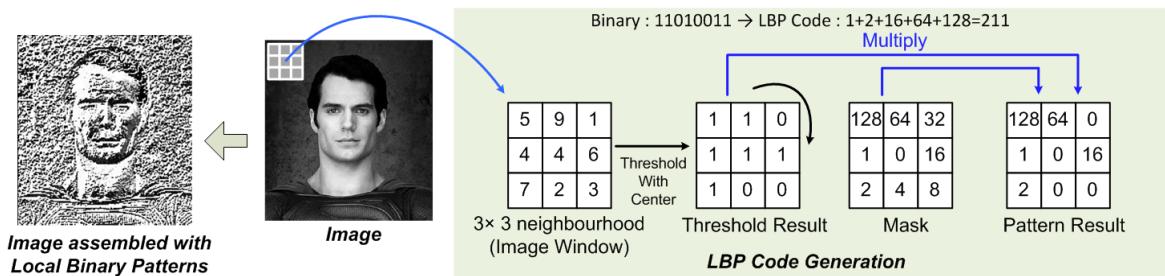
Bài toán nhận dạng khuôn mặt được sự quan tâm từ rất sớm và có rất nhiều phương pháp giải quyết có kết quả tốt. Tuy nhiên mỗi phương pháp đều có những ưu và nhược điểm. Với sự phát triển của công nghệ học máy và sự phát triển của hạ tầng tính toán, các thuật toán xử lý nhận dạng khuôn mặt được cải tiến theo thời gian.

1.3.1 Biểu đồ mô hình nhị phân cục bộ

Biểu đồ mô hình nhị phân cục bộ (Local Binary patterns histograms, gọi tắt là LBP) được dùng để mô tả kết cấu, ngữ cảnh của một bức ảnh. LBP miêu tả theo từng vùng điểm ảnh, bằng cách so sánh giá trị của một điểm ảnh với các điểm ảnh khác bao quanh nó.

Để xây dựng một LBP, chuyển đổi ảnh đầu vào về grayscale. Với mỗi điểm ảnh trong ảnh grayscale, chọn vùng lân cận bao quanh điểm ảnh trung tâm với ma trận 3×3 . Một giá trị LBP sẽ được tính toán cho điểm ảnh trung tâm và lưu vào một ma trận hai chiều có kích thước giống ảnh gốc đầu vào.

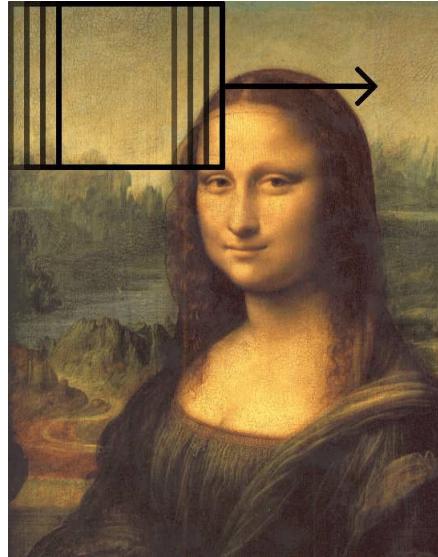
Khi so sánh pixel trung tâm và các pixel khác trong vùng lân cận, nếu các pixel này có giá trị lớn hơn hoặc bằng pixel trung tâm thì sẽ được đánh dấu là 1, ngược lại đánh dấu là 0.



Hình 1.1: Nhận diện bằng biểu đồ mô hình nhị phân cục bộ

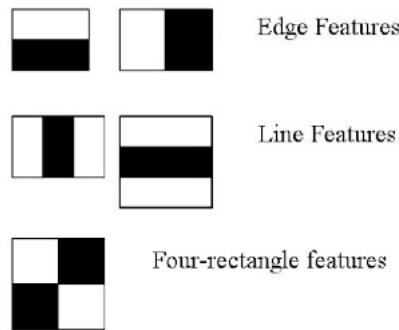
1.3.2 Haar Cascades

Trong bài toán này, mỗi bức ảnh được đưa vào detector được lọc bởi thuật toán sliding window có kích thước cố định ở nhiều tỉ lệ của bức ảnh.



Hình 1.2: Sliding window

Tại mỗi ma trận $n \times n$ gọi là một window, các Haar-like features (Hình 1.3) có trong vùng ảnh đó sẽ được tính toán bằng cách áp dụng thuật toán Summed Area Tables và xác định vùng chưa khuôn mặt. Để làm được điều này cần một bộ phân loại đặc trưng được huấn luyện bởi rất nhiều ảnh có khuôn mặt và ảnh không có khuôn mặt. Thư viện OpenCV đã cung cấp mô hình Haar-cascade có sẵn.



Hình 1.3: Haar-like features ¹

Từ mỗi window có thể tính toán ra rất nhiều Haar-like features (khoảng 180000 đặc trưng cho window có kích thước 24×24) nhưng đa số chúng sẽ không mang đặc trưng của khuôn mặt.

1.3.3 FaceNet

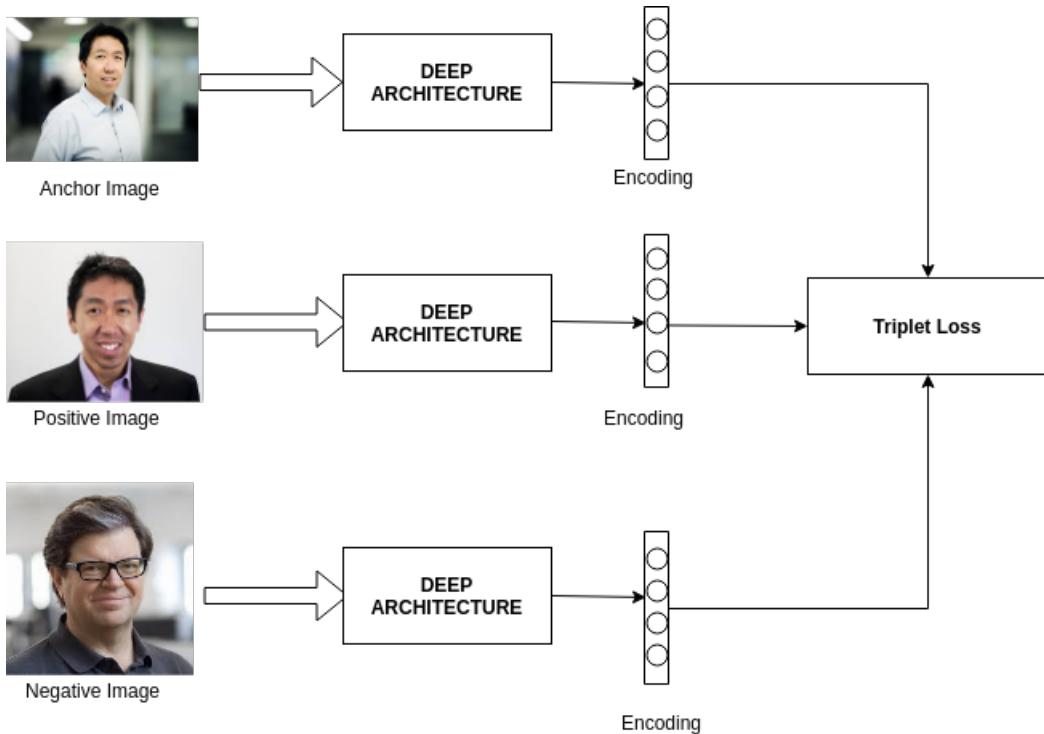
Sự tiến bộ của công nghệ deep learning đã đưa thuật toán nhận diện khuôn mặt đạt độ chính xác cao hơn rất nhiều. Mạng nơ ron tích chập trong FaceNet gồm các tầng tích chập thực hiện trích xuất đặc trưng của một ảnh và sau đó sử dụng mô hình học máy như K-nearest neighbors hoặc Support Vector Machine để phân loại.

FaceNet [1] được xây dựng dựa trên mạng tích chập truyền thống, có nhiệm vụ trích xuất bộ đặc trưng có đầu ra là những vector 128 đo lường của một khuôn mặt (hay đặc trưng).

¹nguồn: https://docs.opencv.org/face_detection.html

FaceNet sử dụng hàm mất mát triplet (??) để tối thiểu hóa khoảng cách giữa các khuôn mặt tương đồng và tối đa hóa khoảng cách khuôn mặt không tương đồng. Vì vậy, FaceNet có thể phân biệt rất chính xác các khuôn mặt của người với người.

Nó thực hiện điều này bằng cách huấn luyện trên ba hình ảnh khác nhau trong đó một hình ảnh khuôn mặt đầu vào được gọi là anchor, sau đó một hình ảnh khác của cùng một người positive, trong khi hình ảnh thứ 3 là hình ảnh của một người khác negative.



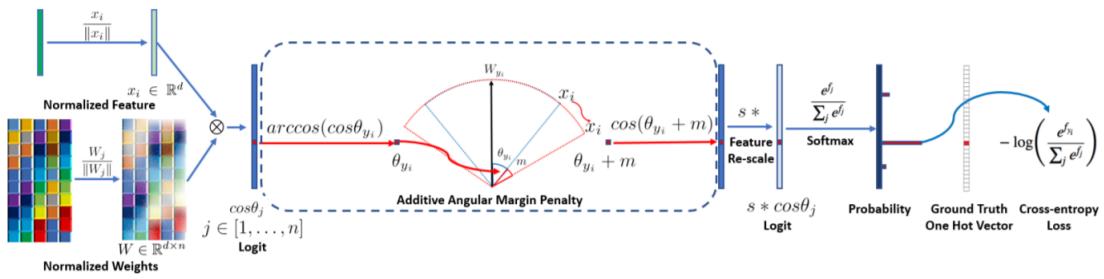
Hình 1.4: Sơ đồ hoạt động của hàm triplet ²

1.3.4 ArcFace

Một trong những thách thức chính trong việc học những đặc trưng trong mạng tích chập nhiều tầng, để nhận diện nhiều khuôn mặt trong ảnh độ phân giải lớn cần thiết kế các hàm mất mát thích hợp giúp tăng cường khả năng phân biệt các khuôn mặt. Gần đây, một trong những nghiên cứu phổ biến là kết hợp các tỷ suất có lợi trong các hàm mất mát để tối đa hóa khả năng phân phân loại. Đó là ArcFace (Additive Angular Margin Loss) [2], có khả năng phân tích được các đặc trưng phân biệt các khuôn mặt đạt chính xác cao. ArcFace sử dụng hàm mất mát trung tâm (centre loss) sử phạt (penalty) khoảng cách giữa các đặc trưng và các nhãn tương ứng trong không gian Euclidean để đạt được kết quả tối ưu.

ArcFace được đánh giá và thực nghiệm rộng rãi trên tất cả các phương pháp nhận diện khuôn mặt hiện đại gần đây trên cơ sở dữ liệu lớn. Theo nghiên cứu cho thấy ArcFace luôn vượt trội so với công nghệ trước đó và có thể được thực hiện dễ dàng với chi phí tính toán không đáng kể.

²nguồn: www.inanalyticsai.com/blog



Hình 1.5: Mô hình huấn luyện ArcFace ³

³nguồn: ArcFace: Additive Angular Margin Loss for Deep Face Recognition

Chương 2

Cơ sở lý thuyết

2.1 Đại số tuyến tính

2.1.1 Ma trận

2.1.1.1 Định nghĩa

Một ma trận \mathbf{A} cấp $m \times n$ trên trường \mathbb{K} (\mathbb{K} – là trường thực \mathbb{R} hoặc phức \mathbb{C}) là một bảng chữ nhật gồm $m \times n$ phần tử trong \mathbb{K} được viết thành m dòng và n cột như sau:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \dots & a_{mn} \end{bmatrix}$$

Trong đó:

a_{ij} là phần tử của ma trận \mathbf{A} nằm ở giao điểm của dòng i và cột j

m : số dòng của ma trận \mathbf{A}

n : số cột của ma trận \mathbf{A}

$[a_{i1} \ a_{i2} \ a_{i3} \ \dots \ a_{in}]$: dòng thứ i của ma trận \mathbf{A}

$\begin{bmatrix} a_{j1} \\ a_{j2} \\ a_{j3} \\ \vdots \\ a_{jm} \end{bmatrix}$: cột thứ j của ma trận \mathbf{A}

2.1.1.2 Phép cộng ma trận

Cho hai ma trận \mathbf{A}, \mathbf{B} cùng cỡ $m \times n$, ta có tổng $\mathbf{A} + \mathbf{B}$ là ma trận có cùng kích thước $(m \times n)$ với phần tử trong vị trí tương ứng bằng tổng của hai phần tử tương ứng của mỗi ma trận:

$$(\mathbf{A} + \mathbf{B}) = a_{ij} + b_{ij} \text{ với } 1 \leq i \leq m \text{ và } 1 \leq j \leq n$$

2.1.1.3 Phép nhân ma trận với ma trận

Xét ma trận $\mathbf{A}_{m \times p}$ và ma trận $\mathbf{B}_{p \times n}$, trong đó số cột của ma trận \mathbf{A} bằng số hàng của ma trận \mathbf{B} . Tích \mathbf{AB} là ma trận \mathbf{C} có m hàng và n cột, phần tử c_{ij} được xác định theo tích vô hướng của hàng tương ứng trong \mathbf{A} với cột tương ứng trong \mathbf{B} :

$$c_{ij} = a_{i1}b_{j1} + a_{i2}b_{j2} + \dots + a_{ip}b_{jp} = \sum_{k=1}^p (a_{ik}b_{jk})$$

Ngoài ra có một phép nhân khác được gọi là *element-wise* (hay *hadamard*) được sử dụng khá nhiều trong học máy. Tích element-wise của hai ma trận cùng kích thước A, B được ký hiệu là $\mathbf{C} = \mathbf{A} \odot \mathbf{B}$, trong đó:

$$c_{ij} = a_{ij}b_{ij}$$

2.1.1.4 Ma trận chuyển vị

Ma trận chuyển vị là một ma trận ở đó các hàng được thay thế bằng các cột, và ngược lại hay nói cách khác nếu ma trận \mathbf{B} là ma trận chuyển vị của ma trận \mathbf{A} thì: $b_{ij} = a_{ji}$

Ma trận chuyển vị của ma trận \mathbf{A} được ký hiệu là \mathbf{A}^T .

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^T = \begin{bmatrix} a & c \\ b & d \end{bmatrix}$$

2.1.2 Norm

2.1.2.1 Định nghĩa

Một hàm số f ánh xạ một điểm x từ không gian n chiều sang tập số thực \mathbb{R} một chiều nếu nó thỏa mãn ba điều kiện sau đây:

- $f(x) \geq 0$. Dấu bằng xảy ra $\Leftrightarrow x = 0$
- $f(\alpha x) = |\alpha|f(x), \forall \alpha \in \mathbb{R}$
- $f(x_1) + f(x_2) \geq f(x_1 + x_2), \forall x_1, x_2 \in \mathbb{R}^n$

Điều kiện thứ nhất. Vì khoảng cách không âm. Khoảng cách giữa hai điểm \mathbf{y} và \mathbf{z} bằng 0 nếu và chỉ nếu hai điểm nó trùng nhau, tức $x = y - z = 0$

Điều kiện thứ hai. Nếu ba điểm \mathbf{y}, \mathbf{v} và \mathbf{z} thẳng hàng và $v - y = \alpha(v - z)$ thì khoảng cách giữa \mathbf{v} và \mathbf{y} gấp $|\alpha|$ lần khoảng cách giữa \mathbf{v} và \mathbf{z} .

Điều kiện thứ ba. Là bất đẳng thức tam giác nếu ta coi $x_1 = w - y, x_2 = z - w$ với \mathbf{w} là một điểm bất kỳ trong cùng không gian.

2.1.2.2 Một số chuẩn thường dùng

Giả sử các vectors $x = [x_1; x_2; \dots; x_n]$, $y = [y_1; y_2; \dots; y_n]$

Nhận thấy rằng khoảng cách Euclid chính là một norm, norm này thường được gọi là **norm 2**:

$$\|x\|_2 = \sqrt{x_1^2 + x_2^2 + \dots + x_n^2} \quad (1)$$

Với p là một số không nhỏ hơn 1 bất kỳ, hàm số sau đây:

$$\|x\|_p = (|x_1|^p + |x_2|^p + \dots + |x_n|^p)^{\frac{1}{p}} \quad (2)$$

được chứng minh thỏa mãn ba điều kiện bên trên, và được gọi là **norm p**.
Có một vài giá trị của p thường được dùng:

1. Khi $p = 2$ chúng ta có norm 2 như ở trên.
2. Khi $p = 1$ chúng ta có:

$$\|x\|_1 = |x_1| + |x_2| + \dots + |x_n| \quad (3)$$

Khi $p \rightarrow \infty$, ta có norm p chính là trị tuyệt đối của phần tử lớn nhất của vector đó:

$$\|x\|_\infty = \max_{i=1,2,\dots,n} |x_i| \quad (4)$$

2.1.2.3 Chuẩn của ma trận

Với một ma trận $A \in \mathbb{R}^{m \times n}$, chuẩn thường được dùng nhất là chuẩn Frobenius, ký hiệu là $\|A\|_F$ là căn bậc hai của tổng bình phương tất cả các phần tử của ma trận đó.

$$\|x\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n a_{ij}^2} \quad (4)$$

2.2 Giải tích

2.2.1 Đạo hàm

2.2.1.1 Định nghĩa

Cho hàm số $y = f(x)$ xác định trên khoảng $(a; b)$ (khoảng $(a; b) = \{x \in \mathbb{R} | a < x < b\}$). Xét giá trị x_0 và giá trị $x \in (a; b), x \neq x_0$.

Đặt $\Delta x = x - x_0$ thì $x = x_0 + \Delta x$ và Δx được gọi là số gia đổi số.

Đặt $\Delta y = f(x) - f(x_0)$ và Δy được gọi là số gia hàm số.

Xét tỷ số $\frac{\Delta y}{\Delta x}$. Nếu khi $\Delta x \rightarrow 0$, tỷ số đó dần tới một giới hạn thì giới hạn đó được gọi là đạo hàm của hàm số $y = f(x)$ tại điểm x_0 ký hiệu là $f'(x)$

$$f'(x) = \lim_{\Delta x \rightarrow 0} \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x}$$

2.2.1.2 Đạo hàm riêng

Đạo hàm riêng của một hàm số đa biến là ta coi tất cả các biến khác như là hằng số và tiến hành đạo hàm theo biến đã chọn.

Đạo hàm riêng của f đối với biến x được ký hiệu khác nhau bởi: f'_x , $\frac{df}{dx}$.

Ví dụ: Hàm số $f(x, y) = ax^2 + bxy + cy^5$ thì ta có:

- Đạo hàm theo x : $f'_x = 2ax + by$
- Đạo hàm theo y : $f'_y = bx + 5cy^4$

Vector gradient: Cho một hàm số $f(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}$. Trong trường hợp này f có các đạo hàm riêng $\frac{df}{dx_j}$ đối với mỗi biến x_j ($1 \leq j \leq n$) thì vector chứa các đạo hàm riêng này là vector gradient.

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f(\mathbf{x})}{\partial x_1} \\ \frac{\partial f(\mathbf{x})}{\partial x_2} \\ \vdots \\ \frac{\partial f(\mathbf{x})}{\partial x_n} \end{bmatrix}$$

2.2.1.3 Chain rule

Đạo hàm hàm hợp là công thức để tính đạo hàm của hàm số gồm nhiều hàm số kết hợp với nhau. Đó là, nếu f, g là hai hàm số và hàm $h(x) = f(g(x))$ thì ta có

$$h'(x) = f(g(x))' = f'(g(x)).g'(x)$$

hay chúng ta có công thức quen thuộc hơn với cách đặt $z = f(y), y = g(x)$:

$$\frac{dz}{dx} = \frac{dz}{dy} \frac{dy}{dz} = f'(y)g'(x) = f'(g(x)).g'(x)$$

2.3 Xác suất

2.3.1 Các phép toán xác suất

2.3.1.1 Tổng xác suất

Tổng xác suất là xác suất của sự kiện hợp. Cho tập sự kiện $A_i, i = \overline{1, n}$ khi đó ta có:

$$P\left(\bigcup_{i=1}^n A_i\right) = \sum_{i=1}^n (-1)^{i+1} \sum_{1 \leq k_1 \leq \dots \leq k_i \leq n} P\left(\bigcap_{j=1}^i A_{k_j}\right)$$

Trong đó, tổng $\sum_{1 \leq k_1 \leq \dots \leq k_i \leq n} P\left(\bigcap_{j=1}^i A_{k_j}\right)$ là tổng của tất cả các xác suất giao của tập con gồm i phần tử từ tập $1, 2, \dots, n$

2.3.1.2 Xác suất có điều kiện

Là xác suất của một sự kiện xảy ra khi biết xác suất của sự kiện đã xảy ra. Xác suất của sự kiện A khi biết B đã xảy ra được ký hiệu là $P(A|B)$. Công thức tính xác suất của A được xác định như sau:

$$P(A|B) = \frac{P(AB)}{P(B)} \forall P(B) \geq 0$$

Nếu A và B là độc lập, tức A không phụ thuộc vào B thì: $P(A|B) = P(A)$ và $P(B|A) = P(B)$
Xác xuất có điều kiện cũng có các tính chất như xác suất thông thường:

- $P\left(\bigcup_{i=1}^n A_i|B\right) = \sum_{i=1}^n (-1)^{i+1} \sum_{k_1 \leq \dots \leq k_i} P\left(\bigcap_{j=1}^i A_{k_j}|B\right)$
- $P(\bar{A}|B) = 1 - P(A|B)$

2.3.1.3 Tích xác suất

Tích xác suất là xác suất của sự kiện giao. Từ công thức xác suất có điều kiện ta có thể tính được xác suất giao như sau:

$$P(AB) = P(B)P(A|B) = P(A)P(B|A)$$

Trường hợp tổng quát, cho $A_i, i = \overline{1, n}$ thì tích xác suất của chúng được tính như sau:

$$P\left(\bigcap_{i=1}^n A_i\right) = P(A_1)P(A_2|A_1)P(A_3|A_2A_1)\dots P(A_n|A_1A_2 \dots A_{n-1})$$

Hay viết gọn thành:

$$P\left(\bigcap_{i=1}^n A_i\right) = \prod_{i=1}^n P(A_i | \bigcap_{j=1}^{i-1} A_j)$$

Tích xác suất còn được gọi là quy tắc chuỗi xác suất bởi cách biểu diễn liên hoàn thành chuỗi như trên.

Nếu A_j là độc lập từng đôi một thì ta có:

$$P\left(\bigcap_{i=1}^n A_i\right) = \prod_{i=1}^n P(A_i)$$

2.3.2 Xác suất hậu nghiệm - Bayes

Từ công thức tính xác suất ta có suy luận sau:

$$P(A)P(B|A) = P(B)P(A|B)$$

Từ đó, ta có thể tính xác suất của A khi biết B như sau:

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)}$$

Trong đó:

- $P(A|B)$: Xác suất có điều kiện
- $P(A)$: Xác suất tiền nghiệm
- $P(B)$: Xác suất hậu nghiệm
- $P(B|A)$: Khả dĩ (likelihood)

2.3.3 Ước lượng tham số bằng cực đại khả dĩ

Ước lượng hợp lý cực đại (có người gọi là khả năng cực đại, tiếng Anh thường được viết là MLE, gọi tắt từ Maximum-Likelihood Estimation) [3] là một kỹ thuật trong thống kê dùng để ước lượng giá trị tham số của một mô hình xác suất dựa trên những dữ liệu có được. Phương pháp này được định nghĩa như sau:

Giả sử $X = x_1, x_2, \dots, x_n$ là tập n quan sát và $Y = y_1, y_2, \dots, y_n$ là số nhãn của quan sát; x, y là hai biến độc lập ngẫu nhiên. Ta cần phải tìm tham số θ để biểu thức sau đây đạt được giá trị lớn nhất

$$p = P(Y|X; \theta) \quad (2.1)$$

hay biểu thức 2.1 được viết lại như sau:

$$\hat{\theta} = \arg \max_{\theta} P(Y|X; \theta) \quad (2.2)$$

Do các quan sát là biến độc lập ngẫu nhiên nên ta có thể viết lại thành:

$$P(Y|X; \theta) = \prod_{i=1}^N P(y_i|x_i, \theta) \quad (2.3)$$

Nhưng trực tiếp hàm số trên không hề đơn giản, hơn nữa khi N lớn thì tích của N số nhỏ hơn một có thể dẫn đến sai số trong tính toán. Một phương pháp thường được sử dụng đó là lấy logarit tự nhiên (cơ số e) của hàm khả dĩ ta được:

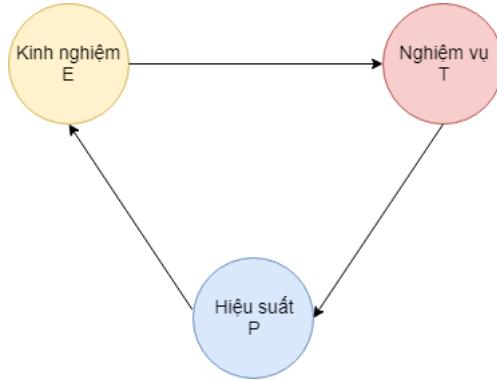
$$\log(P(Y|X; \theta)) = \log \prod_{i=1}^N P(y_i|x_i; \theta) = \sum_{i=1}^N \log P(y_i|x_i; \theta) \quad (2.4)$$

2.4 Học máy

2.4.1 Khái niệm

Học máy được đề ra vào năm 1959 bởi Arthur Samuel. Sau đó, Tom M. Mitchell đã đưa ra một định nghĩa chính thức hơn về các thuật toán được nghiên cứu trong lĩnh vực học máy:

"Một chương trình máy tính được cho là học hỏi từ kinh nghiệm E đối với một số loại nhiệm vụ T và hiệu suất đo P. Nếu hiệu suất của nó trong các nhiệm vụ trong T, được đo bằng P, sẽ cải thiện kinh nghiệm E [4]."



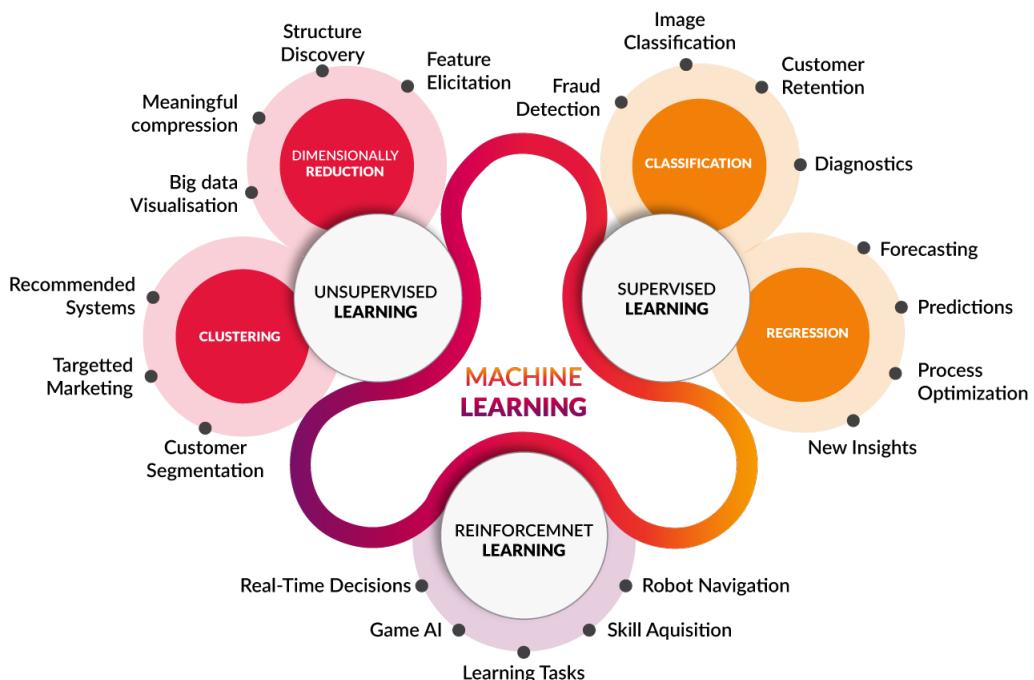
Hình 2.1: Sơ đồ quy trình học máy

2.4.2 Các phương pháp học

Có ba phương pháp học phổ biến là học có giám sát (supervised learning), học không giám sát (unsupervised learning), học tăng cường (reinforcement learning):

- Học có giám sát** Là phương pháp sử dụng những dữ liệu đã được gán nhãn từ trước để suy luận ra quan hệ giữa đầu vào và đầu ra. Các dữ liệu này được gọi là dữ liệu huấn luyện và chúng là cặp các dữ liệu đầu vào/đầu ra. Học có giám sát sẽ xem xét các tập huấn luyện này để từ đó có thể đưa ra dự đoán đầu ra cho một đầu vào mới chưa gặp bao giờ. Ví dụ dự đoán giá nhà, phân loại email,...

- **Học không giám sát:** Khác với học có giám sát, học không giám sát sử dụng những dữ liệu chưa được gán nhãn từ trước để suy luận. Phương pháp này thường được sử dụng để tìm cấu trúc của tập dữ liệu. Tuy nhiên lại không có phương pháp đánh giá được cấu trúc tìm ra được là đúng hay sai. Ví dụ như phân cụm dữ liệu, triết xuất thành phần chính của một chất nào đó.
- **Học tăng cường:** Phương pháp học tăng cường tập trung vào việc làm sao để cho một tác tử trong môi trường có thể hành động sao cho lấy được phần thưởng nhiều nhất có thể. Khác với học có giám sát nó không có cặp dữ liệu gán nhãn trước làm đầu vào và cũng không có đánh giá các hành động là đúng hay sai.



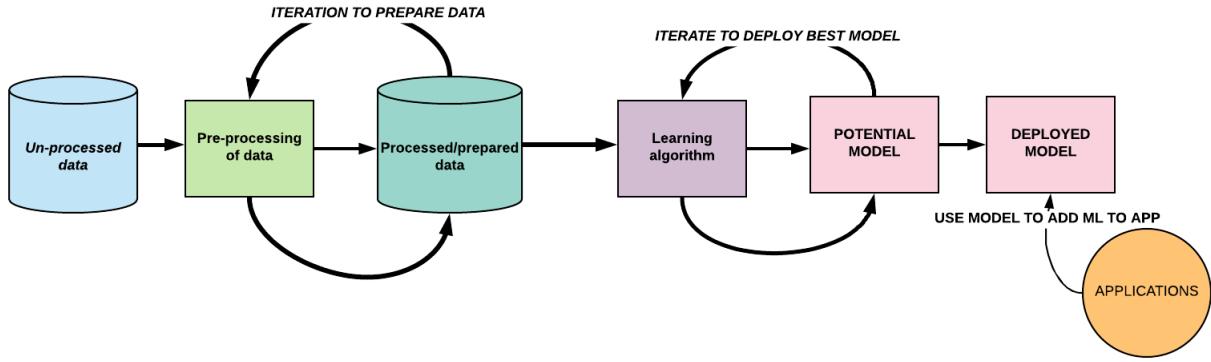
Hình 2.2: Các phương pháp trong học máy

2.4.3 Xây dựng sản phẩm ứng dụng học máy

2.4.3.1 Các giai đoạn trong quy trình xây dựng sản phẩm

Một sản phẩm ứng dụng học máy cần trải qua 5 bước chính:

- Thu thập dữ liệu thô
- Tiền xử lý dữ liệu
- Lựa chọn thuật toán học máy
- Huấn luyện và đánh giá mô hình
- Triển khai mô hình



Hình 2.3: Machine Learning workflow

2.4.3.2 Dữ liệu

Bất cứ một bài toán học máy nào cũng đều cần có dữ liệu để huấn luyện, ta có thể coi nó là điều kiện tiên quyết. Dữ liệu sau khi có được cần phải:

- **Chuẩn hóa:** Tất cả các dữ liệu đầu vào đều cần được chuẩn hóa để máy tính có thể xử lý được. Quá trình chuẩn hóa bao gồm số hóa dữ liệu, co giãn thông số cho phù hợp với bài toán. Việc chuẩn hóa này ảnh hưởng trực tiếp tới tốc độ huấn luyện cũng như cả hiệu quả huấn luyện.
- **Phân chia:** Việc mô hình được chọn rất khớp với tập dữ liệu đang có không có nghĩa là giả thuyết của ta là đúng mà có thể xảy ra tình huống dữ liệu thật lại không khớp. Vấn đề này trong học máy được gọi là quá khớp (Overfitting). Vì vậy khi huấn luyện người ta phải phân chia dữ liệu ra thành 3 loại để có thể kiểm chứng được phần nào mức độ tổng quát của mô hình.
 - **Tập huấn luyện:** Dùng để học khi huấn luyện.
 - **Tập kiểm chứng:** Dùng để kiểm chứng mô hình khi huấn luyện.
 - **Tập kiểm tra:** Dùng để kiểm tra xem mô hình đã phù hợp chưa sau khi huấn luyện.

Tập kiểm tra ta phải lọc riêng ra và không được sử dụng trong quá trình huấn luyện. Còn tập huấn luyện và tập kiểm chứng thì nên xáo trộn đổi cho nhau để mô hình của ta được huấn luyện với các mẫu ngẫu nhiên nhất có thể.

2.4.4 Các vấn đề trong học máy

2.4.4.1 Khái niệm

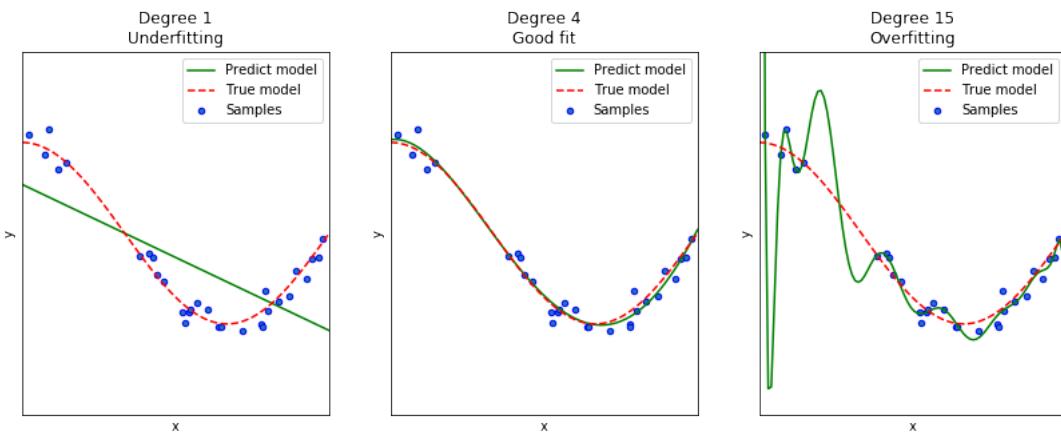
Khi ta xây dựng một mô hình thì sẽ thường xảy ra các hiện tượng không mong muốn như mô hình dự đoán không chính xác, mô hình chỉ chính xác trên tập huấn luyện hay giá trị mất mát quá lớn,... Để tránh các trường hợp trên xảy ra, người xây dựng mô hình cần nắm được một số khái niệm cũng như kỹ thuật để giải quyết khi các vấn đề trên xảy ra.

Underfitting là hiện tượng mô hình chưa được phù hợp với tập dữ liệu huấn luyện và cả các mẫu mới khi dự đoán. Nguyên nhân có thể là do mô hình chưa đủ độ phức tạp cần thiết để bao quát được tập dữ liệu.

Overfitting là hiện tượng mô hình quá khớp với *tập dữ liệu huấn luyện (training set)*, việc này sẽ gây ra hậu quả vô cùng nghiêm trọng nếu tập dữ liệu huấn luyện xuất hiện

nhiều. Mô hình sẽ chỉ chú trọng vào việc xấp xỉ với tập dữ liệu huấn luyện mà quên đi mục đích ban đầu là tổng quát hóa, làm cho mô hình sẽ không thật sự tốt đối với dữ liệu nằm ngoài dữ liệu huấn luyện (dữ liệu test và dữ liệu thực tế). Overfitting xảy ra khi *độ phức tạp của mô hình quá lớn hoặc quá ít dữ liệu*.

Good fitting là mô hình nằm giữa 2 mô hình chưa khớp (*underfitting*) và quá khớp (*overfitting*) cho ra kết quả hợp lý với cả tập dữ liệu huấn luyện và các giá trị mới, tức là nó mang được tính tổng quát như hình 1 ở giữa phía trên. Lý tưởng nhất là khớp được với nhiều dữ liệu mẫu và cả các dữ liệu mới. Tuy nhiên trên thực tế được mô hình như vậy rất hiếm.



Hình 2.4: Một số vấn đề thường gặp khi xây dựng mô hình

Hình 2.4 minh họa rõ các hiện tượng trên qua mô hình được xây dựng bằng hồi quy tuyến tính (*linear regression*) với các feature là bậc mũ. Trong đó đường nét liền thể hiện *mô hình dự đoán (predicted model)*, đường nét đứt thể hiện *mô hình thực (true model)*, các chấm hình tròn là các điểm dữ liệu.

Ở hình thứ nhất chúng ta có thể thấy mô hình dự đoán là một hàm tuyến tính (bậc bằng 1) rất khác với mô hình thực, xa với các điểm dữ liệu. Hiện tượng này ta nói mô hình bị *underfitting*. Với mô hình dự đoán là đa thức bậc 4 chúng ta có thể thấy mô hình dự đoán xấp xỉ như mô hình thực (hình thứ 2). Trường hợp này ta nói mô hình phù hợp (*good fit*). Ở hình thứ 3, khi ta tăng bậc đa thức lên thì mô hình dự đoán quá khớp với các điểm dữ liệu, gần như mọi điểm dữ liệu đều nằm trên mô hình. Tuy nhiên việc khớp hoàn toàn dữ liệu lại không hề tốt vì dữ liệu thường bị nhiễu và có thể khiến mô hình dự đoán bị nhiễu hơn. Trường hợp này ta nói mô hình bị *overfitting*. Để tránh vấn đề này xảy ra chúng ta có phương pháp khá là hữu dụng đó là *regularization*.

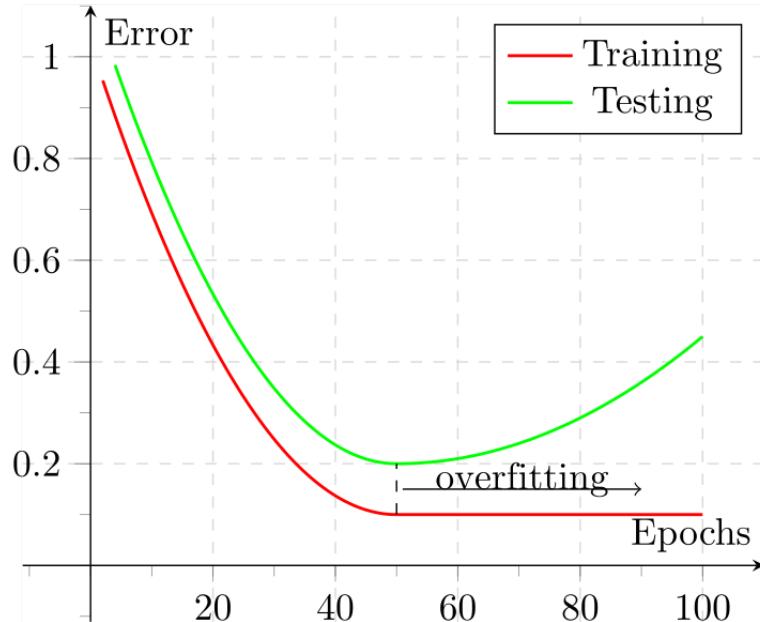
2.4.5 Phương pháp giải quyết

Các phương pháp để giải quyết các vấn đề trên nói chung là làm thay đổi nhỏ các tham số hay kiến trúc mô hình, chấp nhận hy sinh độ chính xác trong quá trình huấn luyện để giảm độ phức tạp của mô hình giúp tránh được hiện tượng không mong muốn mà vẫn giữ được tính tổng quát của mô hình. Một số kỹ thuật như: early stopping, thêm số hạng vào hàm mất mát, drop-out,...

1. Early stopping

Khi ta dùng một phương pháp tối ưu hàm số để giảm thiểu giá trị mất mát thì J_{train}, J_{test} sẽ cùng giảm theo thời gian nhưng nếu sau một thời gian J_{test} tăng lên còn J_{train} tiếp

tục giảm thì đó là lúc bắt đầu dẫn đến overfitting. Cách đơn giản nhất để giảm thiểu overfitting đó là dừng huấn luyện tại ngay thời điểm bắt đầu overfitting và phương pháp này được gọi là *early stopping*. Nếu ta có biểu đồ về sự thay đổi giá trị mất mát của training và testing như Hình 2.5 thì ta có thể thấy thời điểm sử dụng early stopping là vào khoảng epochs 50.



Hình 2.5: Early stopping

2. Dánh giá chéo k-fold

Dánh giá chéo k-fold, có tên tiếng anh là k-fold cross-validation, thường được áp dụng khi chúng ta có ít dữ liệu để huấn luyện và đánh giá làm cho mô hình khó có thể trích xuất được đặc trưng.

Để giải quyết vấn đề này, chúng ta sẽ chia dữ liệu thành k phần bằng nhau, được gọi là *fold*. Một phần sẽ được làm tập đánh giá, $k - 1$ phần còn lại sẽ làm tập huấn luyện. Sau mỗi vòng lặp tập đánh giá sẽ được chuyển sang một tập khác.



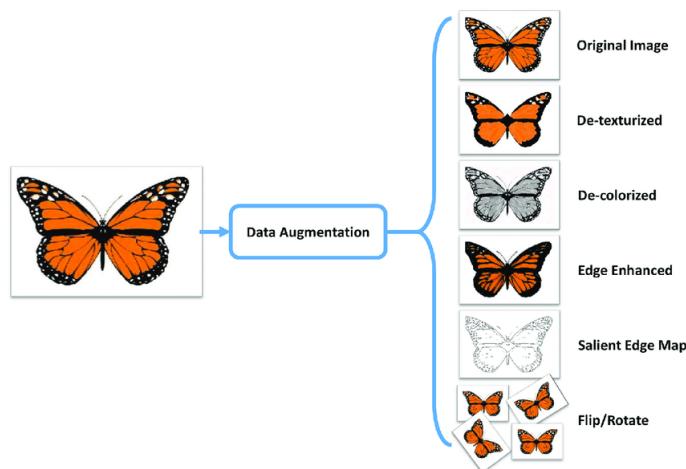
Hình 2.6: Đánh giá chéo k-fold

3. Tăng cường dữ liệu

Tăng cường dữ liệu (Data augmentation) là tạo thêm dữ liệu từ dữ liệu có sẵn.

- Giả sử ta có tập dữ liệu nhưng số lượng quá ít sẽ khiến cho mô hình của ta không học được các đặc trưng do đầu vào không đủ. Như vậy ta cần phải tìm cách làm tăng dữ liệu lên.
- Nếu ta đã có lượng lớn dữ liệu, nhưng chúng lại mất cân bằng. Có nghĩa là sự chênh lệch số lượng đầu vào lớn giữa các lớp với nhau. Khi đó mô hình sẽ ưu tiên những lớp mà có số lượng phần tử nhiều hơn. Ta cần phải tìm cách để dữ liệu chúng ta được cân bằng.

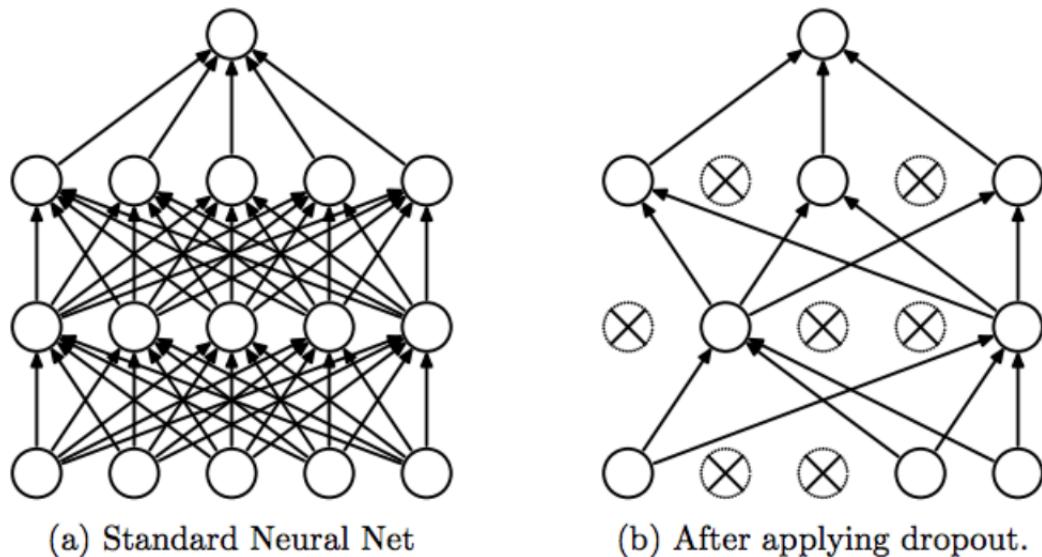
Với hai trường hợp điển hình trên, thì ta sẽ làm tăng cường dữ liệu thông qua một số thủ thuật như: co dãn kích thước ảnh, dịch chuyển ảnh, tăng giảm độ sáng, thêm độ nhiễu cho ảnh,....



Hình 2.7: Ví dụ tăng cường dữ liệu

4. Drop-out

Drop-out [5] là một kĩ thuật Regularization để chống lại vấn đề overfitting. Cách drop out thực hiện là xoá bỏ một số unit trong các step training ứng với một giá trị xác suất p cho trước. Các mạng mới sau khi áp dụng drop out được gọi là subsample. Thông thường xác suất ở layer input bằng 0.8 hay ta loại bỏ khoảng 20% số unit, ở hidden layer thì xác suất là 0.5 có nghĩa là ta loại bỏ 50% số unit ở layer sử dụng drop out.



Hình 2.8: Dropout với $p = 0.5$

Cách hoạt động của dropout

- Dropout được áp dụng trên một layer của mạng neural networks với một xác suất p cho trước (có thể sử dụng nhiều Drop-Out khác nhau cho những layer khác nhau, nhưng trên 1 layer sẽ chỉ có 1 dropout)
- Tại mỗi step trong quá trình training, khi thực hiện feedforward đến layer sử dụng dropout, thay vì tính toán tất cả unit có trên layer, tại mỗi unit ta "gieo xúc xắc" với xác suất p xem unit đó được tính (active) hay không được tính (deactive). Những unit active ta tính toán bình thường còn với những unit deactivate thì ta set giá trị tại unit đó bằng 0
- Trong quá trình test thì tất cả các unit đều được active và chúng ta mong muốn đầu ra của các units giống với đầu ra mong đợi trong quá trình trainning. Ví dụ đầu ra của một unit (trước khi drop out) là \mathbf{a} , khi áp dụng dropout thì đầu ra mong đợi của unit đó sẽ là $\mathbf{pa} + (1 - p)\mathbf{0}$, vì unit bị deactivate thì giá trị của unit đó là 0. Do đó trong quá trình test chúng ta điều chỉnh đầu ra $\mathbf{a} \rightarrow \mathbf{pa}$ để giống với đầu ra mong đợi.

Thời gian test khá là quan trọng nên nếu chúng ta điều chỉnh đầu ra ở các layer áp dụng dropout thì hiệu suất test sẽ bị giảm đi. Vì thế thay vì chỉnh trong quá trình test thì chúng ta sẽ thực hiện việc này trong quá trình trainning. Ta sẽ lấy *dropout mask* (vector xác suất được khởi tạo ngẫu nhiên, tại vị trí có giá trị nhỏ hơn p sẽ được giữ nguyên còn lớn hơn p sẽ set lại giá trị vị trí đó là 0) chia cho p trong quá trình trainning. Trường hợp này được gọi là *inverted dropout*.

2.4.6 Phương pháp đánh giá mô hình

Khi xây dựng một mô hình học máy, 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. Hiệu năng của mô hình thường được đánh giá dựa trên tập dữ liệu kiểm thử (test data) do không chứa dữ liệu đã được huấn luyện. Cụ thể, giả sử đầu ra của mô hình khi đầu vào là tập kiểm thử là một vector dự đoán đầu ra với mỗi phần tử là nhãn được dự đoán của một điểm dữ liệu trong tập kiểm thử. Ta sẽ so sánh giữa vector dự đoán vừa tính được với vector chử nhãn thật của dữ liệu.

2.4.6.1 Dựa vào tỉ lệ trùng nhẫn

Cách này đơn giản, tự nhiên nhất và hay được sử dụng nhất. Cách đánh giá này đơn giản tính tỉ lệ giữa số lượng các dữ liệu được dự đoán đúng và tổng số dữ liệu trong tập dữ liệu kiểm thử.

$$\text{Accuracy} = \frac{\text{Tổng số dữ liệu đầu vào dự đoán đúng}}{\text{Tổng số dữ liệu đầu vào}}$$

2.4.6.2 Ma trận nhầm lẫn

Cách tính sử dụng accuracy như ở trên chỉ cho chúng ta biết được bao nhiêu phần trăm lượng dữ liệu được phân loại đúng mà không chỉ ra được cụ thể mỗi loại được phân loại như thế nào, lớp nào được phân loại đúng nhiều nhất, và dữ liệu thuộc lớp nào thường bị phân loại nhầm vào lớp khác. Để có thể đánh giá được các giá trị này, chúng ta sử dụng một ma trận được gọi là ma trận nhầm lẫn hay confusion matrix. Về cơ bản, confusion matrix thể hiện có bao nhiêu điểm dữ liệu thực sự thuộc vào một class, và được dự đoán là rơi vào một class.

		Actual class		
		Cat	Dog	Rabbit
Predicted class	Cat	5	2	0
	Dog	3	3	2
	Rabbit	0	1	11

Hình 2.9: Ma trận nhầm lẫn

Nhìn vào hình 2.9, ta có thể biết rằng tổng số dữ liệu kiểm tra cho lớp 'cat' là 8, số lượng đoán chính xác là 5, 3 là số lượng bị đoán sai và đoán vào lớp 'dog'. Đối với lớp 'dog' thì có tổng số dữ liệu kiểm tra là 6, trong đó đoán chính xác là 3, đoán nhầm vào lớp 'cat' là 2, lớp 'rabbit' là 1. Và cuối cùng là lớp 'rabbit' thì có tổng số dữ liệu kiểm tra là 13, đoán chính xác là 11, đoán nhầm sang lớp 'dog' là 2.

2.4.6.3 Recall, Precision và F1-score

Từ ma trận nhầm lẫn chúng ta có thể có nhiều cách đánh giá độ chính xác của mô hình hơn như: recall, precision, f1-score,... Trước tiên chúng ta sẽ xem lại ma trận nhầm lẫn được viết lại với lớp muôn so sánh là lớp 'cat'.

		Actual class	
		Cat	Non-cat
Predicted class	Cat	5 True Positives	2 False Positives
	Non-cat	3 False Negatives	17 True Negatives

Hình 2.10: Ma trận nhầm lẫn với lớp cat

Trong đó:

- True positive là mẫu mang nhãn cat được phân lớp đúng vào lớp cat.
- False negative là mẫu mang nhãn cat bị phân lớp sai vào lớp non-cat.
- False positive là mẫu mang nhãn non-cat bị phân lớp sai vào lớp cat.
- True negative là mẫu mang nhãn non-cat được phân lớp đúng vào lớp non-cat.

Hình 2.10 có thể cho biết tỉ lệ số lượng mẫu lớp cat *bị bỏ sót* trên tổng số lượng mẫu trong lớp cat, cách tính này còn có tên gọi là *recall*. Recall sẽ bằng số lượng mẫu cat được dự đoán đúng so với tổng số lượng mẫu thực sự thuộc lớp cat.

$$\text{recall} = \frac{TP}{TP + FN} \quad (2.5)$$

Nếu ta muốn tính *độ chính xác* của lớp cat xem cao hay thấp thì ta lấy số lượng mẫu cat dự đoán đúng so với số lượng mẫu được dự đoán vào lớp cat. Cách tính này được gọi là *precision*.

$$\text{precision} = \frac{TP}{TP + FP} \quad (2.6)$$

Chúng ta thấy rằng cả *precision* và *recall* đều quan trọng. Có lúc thì cái này quan trọng hơn cái kia. Vậy trong thực tế, ta sẽ lấy cái nào làm cái chính? Ta phải điều chỉnh sao cho cả hai cái này thật sự hợp lý? Với **F1-score**, chúng ta chỉ cần quan tâm đến một chỉ số duy nhất (thay vì hai – precision và recall). F1-score được tính thông qua precision và recall bởi công thức sau đây:

$$F1 = \frac{\text{precision} \bullet \text{recall}}{\text{precision} + \text{recall}} \quad (2.7)$$

Dưới đây là một số cách đánh giá được dựa trên ma trận nhầm lẫn 2×2 này.

		True condition		Prevalence = $\frac{\sum \text{Condition positive}}{\sum \text{Total population}}$	Accuracy (ACC) = $\frac{\sum \text{True positive} + \sum \text{True negative}}{\sum \text{Total population}}$
Total population	Condition positive	Condition negative			
Predicted condition	Predicted condition positive	True positive, Power	False positive, Type I error	Positive predictive value (PPV), Precision = $\frac{\sum \text{True positive}}{\sum \text{Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\sum \text{False positive}}{\sum \text{Predicted condition positive}}$
	Predicted condition negative	False negative, Type II error	True negative	False omission rate (FOR) = $\frac{\sum \text{False negative}}{\sum \text{Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\sum \text{True negative}}{\sum \text{Predicted condition negative}}$
	True positive rate (TPR), Recall, Sensitivity, probability of detection = $\frac{\sum \text{True positive}}{\sum \text{Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm = $\frac{\sum \text{False positive}}{\sum \text{Condition negative}}$	Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) = $\frac{\text{LR}^+}{\text{LR}^-}$	F ₁ score = $\frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}$
	False negative rate (FNR), Miss rate = $\frac{\sum \text{False negative}}{\sum \text{Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) = $\frac{\sum \text{True negative}}{\sum \text{Condition negative}}$	Negative likelihood ratio (LR-) = $\frac{\text{FNR}}{\text{TNR}}$		

Hình 2.11: Một số phương pháp khác

2.4.6.4 ROC Curve và AUC

Dường cong ROC (Receiver Operating Characteristic Curve) là một biểu đồ cho thấy hiệu suất của mô hình phân loại ở tất cả các ngưỡng phân loại. Đường cong này vẽ hai tham số:

- Tỉ lệ True Positive (TPR)
- Tỉ lệ False Positive (FPR)

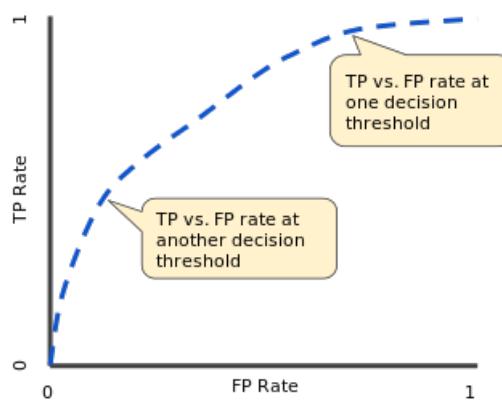
TPR tương tự như Recall và nó được định nghĩa như sau:

$$TPR = \frac{TP}{TP + FN} \quad (2.8)$$

FPR được định nghĩa:

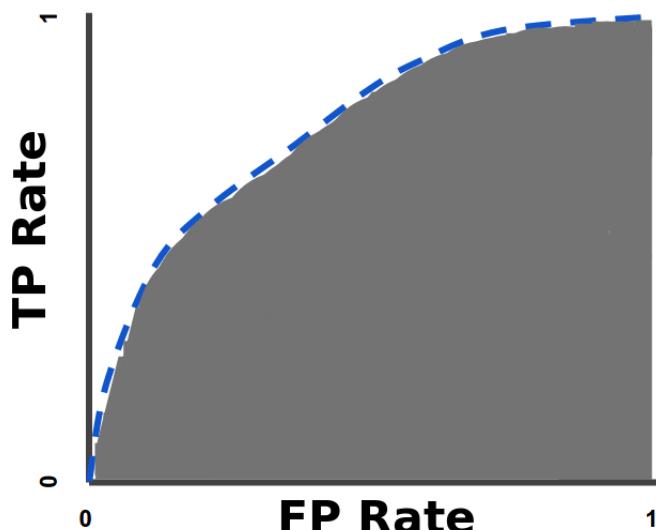
$$FPR = \frac{FP}{FP + FN} \quad (2.9)$$

Đường cong ROC vẽ đồ thị TPR so với FPR ở các ngưỡng phân loại khác nhau. Việc hạ thấp ngưỡng phân loại sẽ phân loại nhiều nhãn hơn theo hướng tích cực, do đó làm tăng cả FN và TP. Hình dưới đây cho thấy đường cong ROC điển hình:



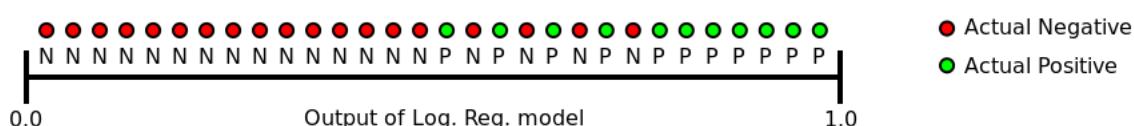
Hình 2.12: Tỷ lệ TP so với FP ở các ngưỡng phân loại khác nhau

Để tính các điểm trong một đường cong ROC, chúng ta có thể đánh giá mô hình hồi quy logistic nhiều lần với các ngưỡng phân loại khác nhau, nhưng điều này sẽ không hiệu quả. Tuy nhiên, có một thuật toán dựa trên thuật toán sắp xếp có thể cung cấp thông tin này, được gọi là AUC (Area Under the ROC Curve). AUC đo toàn bộ khu vực hai chiều bên dưới toàn bộ đường cong ROC (tính toán tích phân) từ $(0,0)$ đến $(1,1)$.



Hình 2.13: AUC

AUC cung cấp một thước đo tổng hợp về hiệu suất trên tất cả các ngưỡng phân loại có thể. Một cách giải thích AUC là xác suất mà mô hình xếp hạng một mẫu positive ngẫu nhiên cao hơn một mẫu negative ngẫu nhiên. Ví dụ, đưa ra các mẫu sau, được sắp xếp từ trái sang phải theo thứ tự tăng dần theo dự đoán hồi quy logistic:



Hình 2.14: AUC

AUC đại diện cho xác suất rằng một mẫu positive (màu xanh lá cây) ngẫu nhiên được đặt ở bên phải của mẫu negative (màu đỏ) ngẫu nhiên. AUC có giá trị từ 0 đến 1. Một mô hình có dự đoán sai 100% thì AUC là 0,0; ngược lại, có dự đoán đúng 100% thì AUC là 1,0.

2.4.7 Một số hàm tính lỗi phổ biến

2.4.7.1 Hàm sai số bình phương trung bình

Hàm sai số toàn平方 trung bình hay mean squared error (MSE), là hàm tiếp cận dễ nhất để hiểu hàm mất mát. Nó được định nghĩa là trung bình của bình phương các sai số giữa giá trị ước lượng được và thực tế.

Giả sử $\hat{\mathbf{y}}$ là một vector đầu ra dự đoán, và \mathbf{y} là vector đầu ra thực tế quan sát được, tương ứng với các dữ liệu đầu vào, thì MSE của phép dự báo có thể ước lượng theo công thức:

$$MSE = \frac{1}{n} \sum_{i=1}^n (\mathbf{y}_i - \hat{\mathbf{y}})^2 \quad (2.10)$$

2.4.7.2 Hàm cross-entropy

Cross entropy [3] được dùng để đo sự giống nhau của hai phân phối xác suất, giá trị của hàm số càng nhỏ thì hai xác suất càng gần nhau và thường dùng cho các bài toán phân loại nhiều nhãn. Giả sử ta có một phân bố xác suất $\mathbf{p} = [p_1, p_2, \dots, p_n]$ với $p_i \in [0, 1]$ và $\sum_{i=1}^n p_i = 1$.

Nếu ta có một phân bố xác suất bất kỳ $\mathbf{q} = [q_1, q_2, \dots, q_n]$ và $q_i \neq 0, \forall i$ thì hàm số cross entropy giữa hai phân bố \mathbf{p} và \mathbf{q} được định nghĩa là:

$$H(\mathbf{p}, \mathbf{q}) = - \sum_{i=1}^C p_i \log q_i \quad (2.11)$$

Chú ý: Hàm cross entropy không có tính đối xứng $H(\mathbf{p}, \mathbf{q}) \neq H(\mathbf{q}, \mathbf{p})$. Theo công thức 2.11 chúng ta có thể thấy giá trị của \mathbf{q} không thể nhận giá trị là 0. Vì thế khi sử dụng cross entropy trong các bài toán học có giám sát, chúng ta phải để \mathbf{p} là đầu ra thực tế vì chỉ có vị trí nhãn là được đánh dấu 1, các vị trí còn lại được đánh dấu 0 (sử dụng one-hot encoding), \mathbf{q} là đầu ra dự đoán vì không có xác suất nào bằng 0 tuyệt đối.

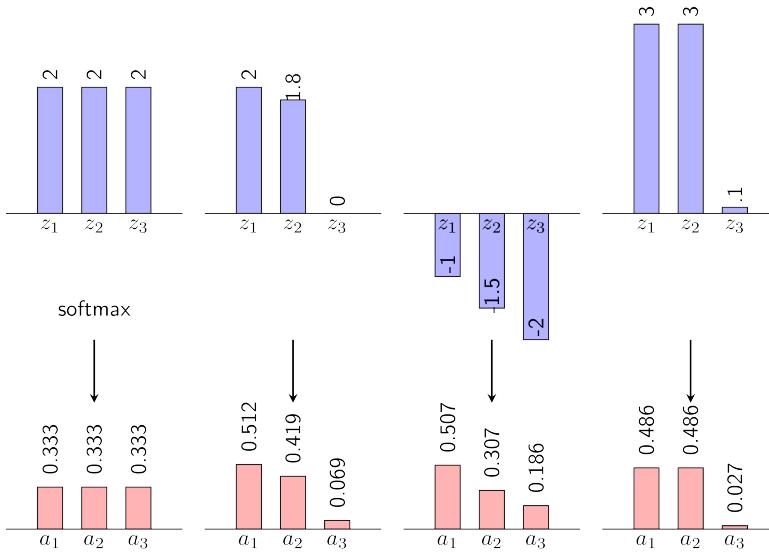
2.4.7.3 Hàm softmax

Hàm softmax hay hàm trung bình mũ [3] là sự khái quát hóa của hàm sigmoid biến không gian K-chiều véc tơ với giá trị thực bất kỳ đến không gian K-chiều véc tơ mang giá trị trong phạm vi $(0, 1)$. Hàm softmax có phương trình toán học như sau:

$$y_i = \frac{\exp(x_i)}{\sum_{j=1}^n \exp(x_j)}; \forall i = 1, \dots, n, \text{ với } n \text{ là số lượng phần tử trong vector } \mathbf{x}$$

Hàm softmax có công dụng trong việc phân loại tập dữ liệu. Vì nếu tại giá trị phần tử x_i lớn vượt trội so với toàn bộ dữ liệu ở vector \mathbf{x} thì giá trị đầu ra y_i cũng sẽ lớn vượt trội so với đầu ra ở các phần tử khác. Do tổng giá trị đầu ra của hàm softmax luôn bằng một, luôn dương và mỗi đầu ra đều phụ thuộc vào tất cả các đầu vào nên ta có thể coi giá trị đầu ra thể hiện xác suất của dữ liệu rơi vào từng tập dữ liệu tương ứng.

Ví dụ dữ liệu đầu vào là vector \mathbf{z} , kết quả giá trị đầu ra tương ứng là vector \mathbf{a} (Hình 2.15). Chúng ta có thể thấy các giá trị a_1, a_2, a_3 thể hiện xác suất của dữ liệu \mathbf{z} rơi vào.



Hình 2.15: Ví dụ hàm softmax¹

2.4.7.4 Hàm triplet

Hàm triplet hay *hàm bộ ba* [?] là một hàm mất mát cho mạng nơ ron để đánh giá mức độ chính xác của tầng phân loại. Nó thực hiện điều này bằng cách huấn luyện trên ba dữ liệu khác nhau, trong đó đầu vào - anchor, sau đó một dữ liệu khác của cùng một loại - positive, trong khi dữ liệu thứ ba là của một loại khác - negative. Mục tiêu cuối cùng của quá trình huấn luyện với hàm mất mát bộ ba số với đặc trưng được mã hoá (embedding) được đo trên khoảng cách euclidean nhằm xác định dữ liệu có tính chất giống nhau gần nhau hơn và dữ liệu có tính chất khác nhau thì xa nhau hơn.

Hàm mất có thể được mô tả bằng hàm khoảng cách Euclidean:

$$\mathcal{L}(A, P, N) = \max(||f(anchor) - f(positive)||^2 - ||f(anchor) - f(negative)||^2 + \alpha, 0),$$

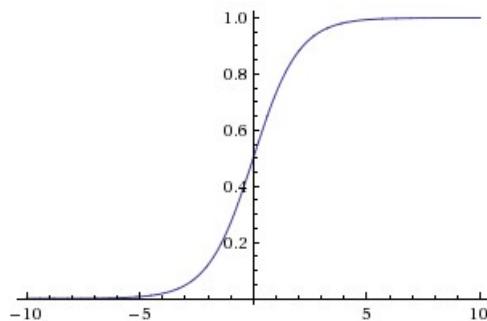
với α là một tham số cân bằng giữa positive và negative

2.4.7.5 Hàm sigmoid

Hàm sigmoid là hàm phi tuyến tính có đồ thị một dạng đường cong hình dạng chữ "S" (Hình 2.16) và có công thức toán học là

$$f(x) = \frac{1}{1 + e^{-x}} \quad (2.12)$$

Sự tăng trưởng của đồ thị gồm giai đoạn tăng trưởng ban đầu được xấp xỉ hàm mũ và khi quá trình bão hòa bắt đầu, sự phát triển sẽ chậm lại, và tới giai đoạn trưởng thành thì dừng hẳn.



Hình 2.16: Hàm sigmoid

Giá trị của hàm số luôn nằm trong khoảng $(0,1]$, cụ thể hơn là với đầu vào lớn hàm số sẽ cho đầu ra gần với 1 còn với đầu vào nhỏ hàm số sẽ cho đầu ra gần với 0. Và đạo hàm của hàm sigmoid rất đẹp.

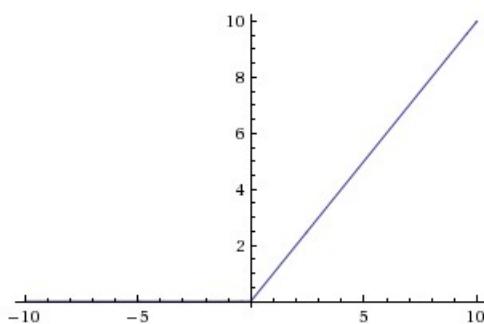
$$f'(x) = f(x)(1 - f(x)) \quad (2.13)$$

2.4.7.6 Hàm ReLU

Hàm ReLU (rectified linear unit function) là hàm phi tuyến tính đơn giản nhất, có công thức toán học là

$$f(s) = \max(0, s) \quad (2.14)$$

Đồ thị hàm ReLU được thể hiện ở Hình 2.17



Hình 2.17: Hàm ReLU

Ta thấy hàm ReLU có đầu ra là 0 nếu đầu vào nhỏ hơn hoặc bằng 0, và đầu ra bằng đầu vào trong trường hợp ngược lại. Hàm ReLU có đạo hàm như sau:

$$f'(x) = \begin{cases} 0, & \text{if } x < 0, \\ 1, & \text{otherwise.} \end{cases}$$

Hàm ReLU thường được sử dụng làm hàm truyền trong các hidden layer, còn ở layer cuối cùng thì ta sẽ sử dụng hàm khác để có thể tính toán được xác suất dự đoán vào vùng phân loại.

2.4.8 Một số tham số huấn luyện

2.4.8.1 Model hyperparameter

Trước hết, ta hiểu về model parameter là các giá trị của mô hình được sinh ra từ dữ liệu huấn luyện giúp thể hiện mối liên hệ giữa các đại lượng trong dữ liệu.

Model hyperparameter khác với model parameter. Nó là các tham số được sử dụng trong quá trình huấn luyện, giúp mô hình tìm ra được các model parameter hợp lý nhất. Nó thường được lựa chọn thủ công bởi những người tham gia trong việc huấn luyện mô hình.

Một vài model hyperparameter thường dùng trong học máy như: learning rate, batch size, epoch, iteration,...

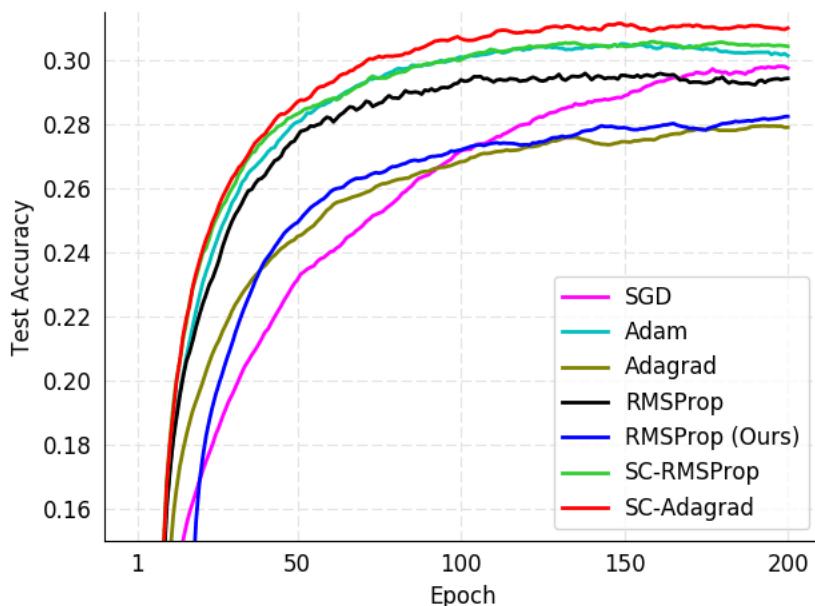
- **learning rate:** Tốc độ học hay độ dốc của quá trình đạo hàm hàm tối ưu Gradient Descent.
- **batch size:** Số lượng mẫu mà thuật toán tối ưu Mini-batch Gradient Descent sử dụng cho một lần cập nhật trọng số.

- **epoch:** Mỗi epoch là mỗi lần duyệt qua hết số lượng mẫu trong tập huấn luyện.
- **iteration:** Là số lượng batch mà thuật toán phải duyệt trong một epoch.

2.4.9 Một vài phương pháp tối ưu

Mục tiêu của bài toán tối ưu là tìm ra nghiệm sao cho tại đó hàm số đạt giá trị nhỏ nhất *global minimum* hoặc hàm số đạt giá trị lớn nhất *global maximum*. Tuy nhiên, việc tìm global minimum hay global maximum của các hàm số là rất phức tạp, thậm chí là bất khả thi. Thay vào đó, người ta thường cố gắng tìm các điểm lân cận của chúng ở một mức độ nào đó và coi đó là nghiệm cần tìm của bài toán.

Giả sử ta có N quan sát (\mathbf{x}, \mathbf{y}) có ánh xạ $\mathbf{x} \rightarrow \mathbf{y}$ với $f(\theta, \mathbf{x}) = \mathbf{y}$. Ta cần tìm global minimum cho hàm $f(\theta; \mathbf{x}, \mathbf{y})$ trong đó θ là một vector, $\theta = [\theta_1, \theta_2, \dots, \theta_m]$. Ta có một số phương pháp giúp ta giải quyết bài toán đó là: gradient descent, stochastic gradient descent, adam, rmsprop,... và chúng có tốc độ tối ưu khác nhau (hình 2.18). Gradient của hàm số f tại một điểm θ bất kỳ được ký hiệu là $\nabla_{\theta} f(\theta; \mathbf{x}, \mathbf{y})$



Hình 2.18: Tốc độ tối ưu của một số phương pháp

2.4.9.1 Gradient descent

Thuật toán gradient descent [3] giúp ta tìm θ sao cho $f(\theta, \mathbf{x})$ gần \mathbf{y} nhất.

Algorithm 1 Gradient descent

Require: Tập N quan sát (\mathbf{x}, \mathbf{y})
Require: $\theta = [\theta_1, \theta_2, \dots, \theta_m]$

- 1: **repeat**
- 2: **for all** $i = 1; i \leq N; i++$ **do**
- 3: $\nabla\theta_i := -\nabla_\theta f(\theta; \mathbf{x}^{(i)}, y_i)$
- 4: **end for**
- 5: $\nabla\theta = \frac{1}{N} \sum_{i=1}^N \nabla\theta_i$
- 6: Chọn bước nhảy η
- 7: Cập nhật $\theta := \theta - \eta \nabla\theta$
- 8: **until** thuật toán hội tụ

Trong đó η (đọc là eta) là một số dương được gọi là learning rate (tốc độ học) và giá trị của learning rate thường là 0.001. Việc lựa chọn learning rate rất quan trọng trong các bài toán thực tế. Việc lựa chọn giá trị này phụ thuộc nhiều vào từng bài toán và phải làm một vài thí nghiệm để chọn ra giá trị tốt nhất. Và dấu trừ tại $\nabla\theta := -\nabla_\theta f(\theta)$ thể hiện việc chúng ta phải đi *ngược với đạo hàm* (Đây cũng chính là lý do phương pháp này được gọi là Gradient Descent - descent nghĩa là đi ngược).

Nếu dữ liệu có kích thước N lớn thì mỗi lần cập nhật θ đòi hỏi chúng ta sử dụng tất cả các quan sát \mathbf{x}_i do đó khối lượng tính toán lớn do phải tính đạo hàm trên toàn bộ dữ liệu, thuật toán chạy chậm. Do vậy để tiết kiệm khối lượng tính toán, chúng ta sẽ cập nhật tính toán sau mỗi dữ liệu quan sát, phương pháp này gọi là *stochastic gradient descent (SGD)*

Algorithm 2 Stochastic gradient descent

Require: Tập N quan sát (\mathbf{x}, \mathbf{y})
Require: $\theta = [\theta_1, \theta_2, \dots, \theta_m]$

- 1: **repeat**
- 2: Xáo trộn dữ liệu
- 3: **for all** $i = 1; i \leq N; i++$ **do**
- 4: $\nabla\theta_i := -\nabla_\theta f(\theta; \mathbf{x}^{(i)}, y_i)$
- 5: Chọn bước nhảy η
- 6: Cập nhật $\theta := \theta + \eta \nabla\theta$
- 7: **if** hội tụ **then**
- 8: break
- 9: **end if**
- 10: **end for**
- 11: **until** thuật toán hội tụ

Khác với SGD, thay vì mỗi *iteration* ta tính toán trên một quan sát thì ta sẽ tính toán với k quan sát ($1 < k \ll N$). Phương pháp này được gọi là *mini-batch gradient descent*.

Algorithm 3 Mini-batch Gradient descent

Require: Tập n quan sát (\mathbf{x}, \mathbf{y})

Require: $\theta = [\theta_1, \theta_2, \dots, \theta_m]$

1: **repeat**

2: Xáo trộn dữ liệu

3: **for all** $i = 1; i \leq N; i = i + k$ **do**

4: $\nabla\theta_i := - \sum_{j=i}^{i+k} \nabla_\theta f(\theta; \mathbf{x}^{(j)}, y_j)$

5: Chọn bước nhảy η

6: Cập nhật $\theta := \theta - \eta \nabla\theta_i$

7: **if** hội tụ **then**

8: break

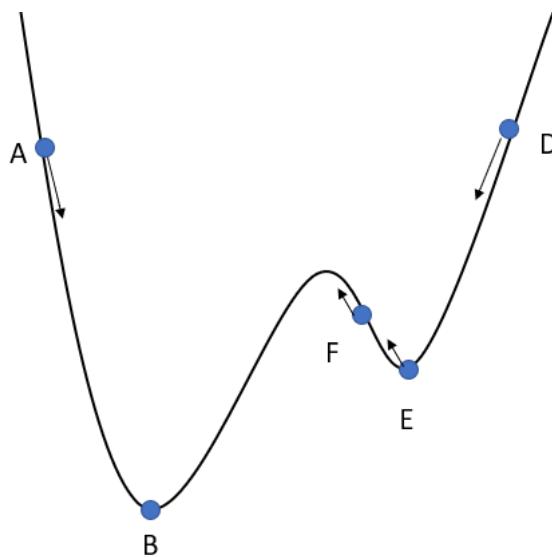
9: **end if**

10: **end for**

11: **until** thuật toán hội tụ

2.4.9.2 Gradient descents với Momentum

Giả sử ta vẽ được một thung lũng như Hình 2.19, ta thả một viên bi vào trong đó và mong muốn nó lăn đến điểm B, điểm sâu nhất của thung lũng. Nếu ta may mắn thả viên bi ở điểm A hoặc điểm G thì viên bi dễ dàng đến điểm B. Nhưng nếu ta thả viên bi tại điểm D thì viên bi có thể sẽ giao động xung quanh điểm E và dừng tại đó do chưa đủ lực để đẩy viên bi qua điểm F rồi đến điểm G. Khi đó E chính là một điểm local minimum mà chúng ta không muốn. Do đó nếu ta tác động thêm một lực vào viên bi giúp nó có thể từ điểm E vượt qua F và tiến đến G.



Hình 2.19: Đồ thị

Thuật toán gradient descent được ví như trọng lực tác dụng vào viên bi giúp nó di chuyển, B được coi điểm global minimum và E là một điểm local minimum. Để tránh hiện tượng nghịch của gradient descent rơi vào một điểm local minimum không mong muốn thì ta tác động thêm một lực giúp gradient descent có thể bật ra khỏi vị trí của local minimum, lực này gọi là đà (*momentum*). Vì thế cách cập nhật θ sẽ thay đổi một chút như sau:

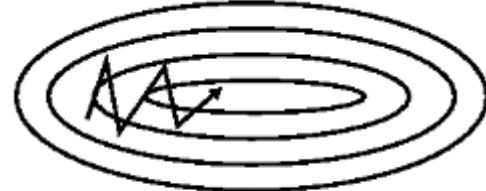
$$v_t := \beta v_{t-1} + \eta \nabla \theta_t$$

$$\theta_t := \theta_{t-1} - v_t$$

Với t là bước lặp thứ t , biến v được khởi tạo bằng 0 và trong tối ưu β được gọi là đà (*momentum*) với giá trị thường là 0.9.



(a) SGD without momentum



(b) SGD with momentum

Hình 2.20: SGD kết hợp với momentum

2.4.9.3 RMSprop

RMSprop [6] là một cách giúp cho learning rate có thể thích nghi, điều chỉnh dựa trên độ lớn của gradient, được đưa ra bởi Geoff Hinton. Thuật toán này là một cách khắc phục việc dừng huấn luyện quá sớm khi áp dụng thuật toán Adagrad bằng cách chia learning rate cho . Khi áp dụng RMSprop, learning rate sẽ được thay đổi như sau:

$$m_t := \beta m_{t-1} + (1 - \beta) \nabla \theta_t^2$$

$$\theta_t := \theta_{t-1} - \frac{\eta}{\sqrt{m_{t-1}} + \epsilon} \nabla \theta_t$$

Ta có biến m được khởi tạo bằng 0 và β là tốc độ giảm của learning rate, thường có giá trị là 0.9, 0.95 hoặc 0.99. Và ϵ giúp chúng ta tránh trường hợp chia cho 0, vì thế giá trị của ϵ thường là 10^{-8}

2.4.9.4 Adam

Adam (Adaptive Moment Estimation)[7] là một bản cập nhật được đưa ra gần đây, nó khá giống với RMSprop và momentum. Cách cập nhật của thuật toán Adam như sau:

$$m_t := \beta_1 m_{t-1} + (1 - \beta_1) \nabla \theta_t$$

$$v_t := \beta_2 v_{t-1} + (1 - \beta_2) \nabla \theta_t^2$$

$$\widehat{m}_t := \frac{m_t}{1 - \beta_1^t}$$

$$\widehat{v}_t := \frac{v_t}{1 - \beta_2^t}$$

$$\theta_t := \theta_1 + \frac{\eta}{\sqrt{\widehat{v}_t} + \epsilon} \widehat{m}_t$$

m_t , v_t là giá trị ước lượng giữa thời điểm trước và thời điểm sau của các gradient tương ứng. Các giá trị của m_t , v_t đều được khởi tạo bằng 0. Các tác giả của Adam thấy rằng chúng bị lệch về 0, đặc biệt là các bước lặp đầu. Do vậy họ tạo ra \widehat{m}_t và \widehat{v}_t để chống lại việc giá trị của m_t và v_t lệch về 0. Các giá trị của β_1 , β_2 được tác giả đề xuất là 0.9 cho β_1 , 0.999 cho β_2 và 10^{-8} cho ϵ như ở RMSprop.

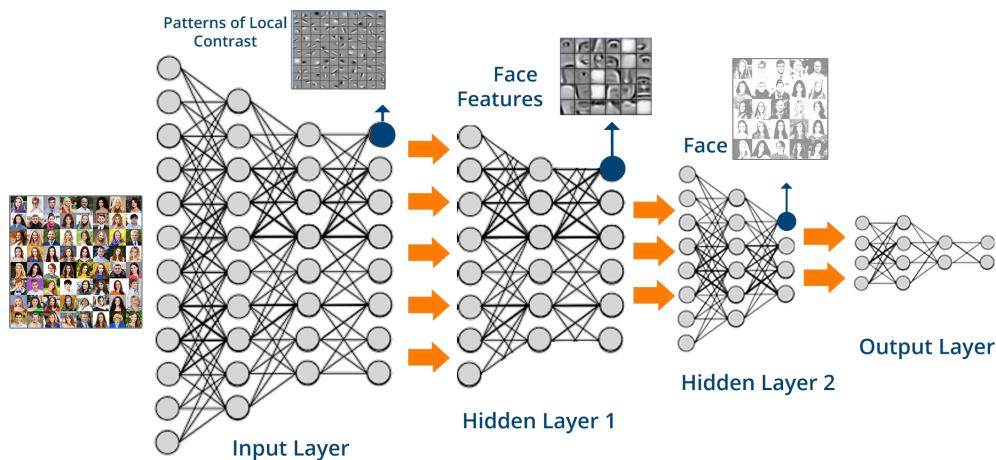
Chương 3

Mạng Nơ-ron

3.1 Tổng quan

3.1.1 Định nghĩa

Mạng nơ-ron nhân tạo, Artificial Neural Network (ANN) là một mô hình xử lý thông tin phỏng theo cách thức xử lý thông tin của các hệ nơ-ron sinh học. Nó được tạo nên từ một số lượng lớn các phần tử (nơ-ron) kết nối với nhau thông qua các liên kết (trọng số liên kết) làm việc như một thẻ thống nhất để giải quyết một vấn đề cụ thể nào đó. Một mạng nơ-ron nhân tạo được cấu hình cho một ứng dụng cụ thể (nhận dạng mẫu, phân loại dữ liệu,...) thông qua một quá trình học từ tập các mẫu huấn luyện. Về bản chất học chính là quá trình hiệu chỉnh trọng số liên kết giữa các nơ-ron, tìm ra các đặc trưng tổng quát.



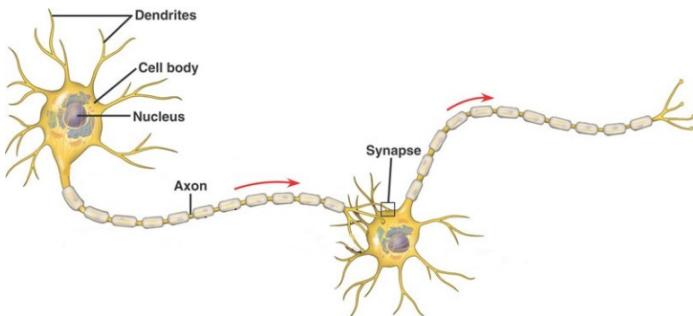
Hình 3.1: Mạng nơ-ron

3.2 Cấu tạo của nơ-ron

3.2.1 Nơ-ron sinh học

Cách thức hoạt động của bộ não nói riêng và của hệ thần kinh nói chung đã được con người quan tâm nghiên cứu từ rất lâu nhưng cho đến nay các nhà khoa học vẫn chưa thực sự hiểu rõ chi tiết về hoạt động của bộ não và hệ thần kinh. Đặc biệt là trong các hoạt động liên quan đến trí óc như suy nghĩ, học tập, tư duy, trí nhớ, sáng tạo.... Tuy nhiên, các nhà khoa học cũng

có một số thông tin căn bản về bộ não con người. Theo đó, một bộ não con người trung bình cân nặng khoảng 1,5kg và có thể tích là 235 cm^3 , cấu tạo bộ não được chia ra làm nhiều vùng khác nhau, mỗi vùng kiểm soát một hay nhiều hoạt động của con người. Hoạt động của cả hệ thống thần kinh bao gồm não bộ và các giác quan như sau: đầu tiên con người nhận được kích thích bởi các giác quan từ bên ngoài hoặc trong cơ thể. Các kích thích này được biến thành các xung điện bởi chính các giác quan tiếp nhận kích thích. Những tín hiệu này được chuyển về trung ương thần kinh là bộ não để xử lý. Tại bộ não các thông tin sẽ được xử lý, đánh giá và so sánh với các thông tin đã được lưu trữ để đưa ra các quyết định dưới dạng các xung điện. Từ những quyết định từ bộ não sẽ sinh ra các mệnh lệnh cần thiết và gửi đến những bộ phận thi hành thích hợp như các cơ tay, chân, giác quan....



Hình 3.2: Minh họa cấu tạo nơ-ron sinh học

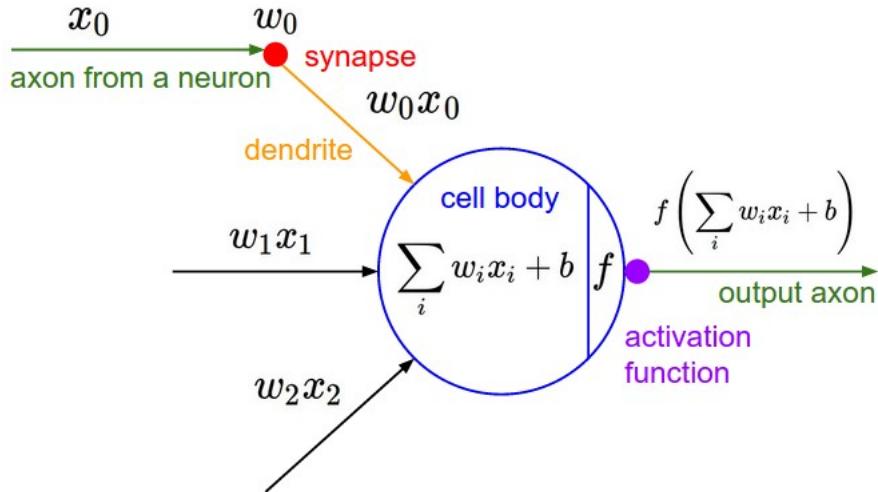
Khi xem xét ở mức độ tế bào thì bộ não được hình thành từ 10^{11} phần tử gọi là nơ-ron (hay neural sinh học). Mỗi neural được liên kết chặt chẽ với 10^4 neural khác. Các neural này có nhiều đặc điểm chung với các tế bào khác trong cơ thể, ngoài ra chúng có những khả năng mà các tế bào khác không có được đó là khả năng nhận, xử lý và truyền các tín hiệu điện hóa làm cơ sở hình cách thức xử lý thông tin của bộ não. Hình 3.2 mô tả cấu tạo và cách thức hoạt động của neural sinh học, trong đó mỗi nơ-ron sinh học có 4 thành phần cơ bản:

- Các nhánh tín hiệu vào (denrites) đây chính là các mạng dây của các dây thần kinh truyền tín hiệu vào đến thân nơ-ron.
- Thân nơ-ron (cell body) chứa nhân (nucleus) hay một số tài liệu gọi là soma có nhiệm vụ chính là tổng hợp và xử lý các tín hiệu điện nhận vào từ các đầu vào. Bản chất của quá trình này chính là việc lấy tổng tất cả các tín hiệu neural nhận được.
- Sợi trực ra (axon) có chức năng truyền tín hiệu từ thân tế bào này sang nơ-ron khác. Phần cuối của axon được chia thành nhiều nhánh nhỏ (cả của denrites và axon) kết thúc tại khớp nối (Synapse).
- Khớp nối (Synapse) là điểm liên kết giữa sợi trực ra của nơ-ron này với các nhánh denrites của neural khác. Liên kết giữa các nơ-ron và độ nhạy của mỗi synapse được xác định bởi quá trình học phức tạp. Khi điện thế của synapses tăng lên do xung điện phát ra từ axon thì synapses sẽ tiết ra một loại hóa chất để kết nối mở ra cho các ion đi qua nó. Các ion này làm thay đổi tín hiệu điện thế trên các điểm tiếp xúc tạo ra các xung điện lan truyền tới các neural khác.

3.2.2 Nơ-ron nhân tạo

Cũng giống như nơ-ron sinh học, nơ-ron nhân tạo cũng nhận đầu vào và thông qua quá trình xử lý đầu vào đó để thu được kết quả là đầu ra. Các nhánh tín hiệu vào sẽ được nhận

từ các sợi trực ra của các nơ-ron khác tại khớp nối và chuyển thông tin vào trong nhân, nhân nơ-ron sẽ là với hàm tổng, hàm kích hoạt và sợi trực ra của nơ-ron tương đương với đầu ra. Các thành phần trên được trình bày qua Hình 3.3.



Hình 3.3: Cấu trúc của một nơ-ron ¹

- *Dầu vào (input)*: là các tín hiệu vào của nơ-ron, các tín hiệu này thường đưa vào dưới dạng một vector, kí hiệu là \mathbf{x} .
- *Trọng số (weight)*: mỗi liên kết được thể hiện bởi một trọng số liên kết và thường được kí hiệu w . Thông thường, các trọng số này được khởi tạo một cách ngẫu nhiên theo phân phối chuẩn ở thời điểm khởi tạo mạng và được cập nhật liên tục trong quá trình học mạng.
- *Nguõng (bias)*: là tham số nhằm tăng khả năng thích ứng của mạng nơron trong quá trình học và thường được kí hiệu là b . Bias gần giống như trọng số, trừ một điều là nó luôn có tín hiệu vào không đổi bằng 1.
- *Hàm kết hợp (combination function)*: Mỗi một đơn vị trong một mạng kết hợp các giá trị đưa vào nó thông qua các liên kết với các đơn vị khác, sinh ra một giá trị gọi là *netinput*. Thông thường hàm này sẽ là hàm tổng của các tích giữa trọng số với đầu vào sau đó cộng thêm bias, được biểu diễn thông qua biểu thức $z = \sum_{i=1}^n (x_i w_i) + b$.
- *Hàm kích hoạt (activation function hoặc transfer function)*: Hàm này được dùng để giới hạn phạm vi đầu ra của mỗi nơ-ron. Nó nhận đầu vào là kết quả của hàm kết hợp và nguõng đã cho. Thông thường hàm kích hoạt sẽ là các hàm phi tuyến tính.
- *Dầu ra (output)*: là tín hiệu đầu ra của một nơ-ron, với mỗi nơ-ron sẽ có tối đa là một đầu ra. Nếu như nơ-ron đó ở các hidden layer (3.3.1) thì đầu ra của nó được gọi là activation và được kí hiệu là a .

Khi quy về toán học thì một nơ-ron sẽ được thể hiện thông qua hai hàm sau:

¹nguồn: <https://cs231n.github.io/>

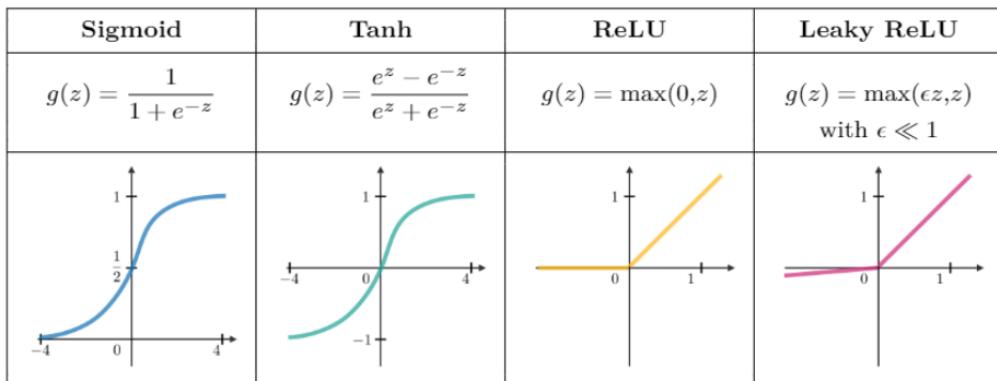
$$z = b + \sum_{i=1}^n (w_i x_i) + b = b + \mathbf{w} \mathbf{x}$$

$$a = f(z)$$

trong đó:

- x_i, w_i là giá trị thứ i của đầu vào và trọng số tương ứng
- Hàm f là hàm truyền và có đầu vào là giá trị của bộ tổng z
- a là giá trị được tính bởi hàm truyền và là đầu ra của nơ-ron

Có rất nhiều hàm kích hoạt được sử dụng cho tới nay như: sigmoid, tanh, ReLU, leaky ReLU, softmax,...



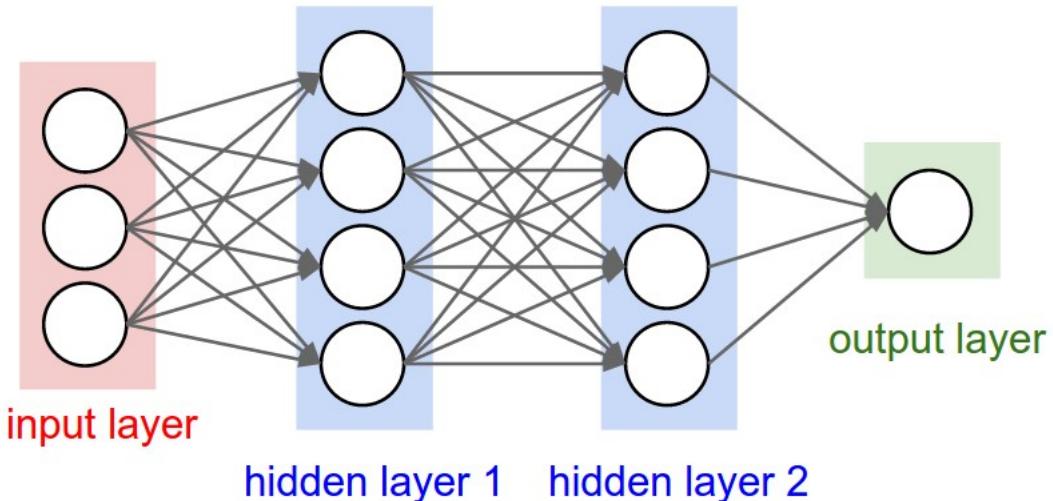
Hình 3.4: Một số hàm kích hoạt ²

3.3 Mạng nơ-ron nhiều tầng

3.3.1 Kiến trúc chung

Mạng nơ-ron là sự kết hợp của các tầng perceptron hay còn gọi là perceptron nhiều tầng (multilayer perceptron) sẽ có nhiều tầng liên kết với nhau và được chia làm 3 tầng chính: tầng đầu vào (*input layer*), tầng ẩn (*hidden layer*), tầng đầu ra (*output layer*). Mỗi tầng sẽ có một số lượng perceptron khác nhau, không có quy định chung và tùy theo người thiết kế mạng. Kiến trúc chung của mạng nơ-ron được minh họa trong Hình 3.5.

²nguồn <https://github.com/afshinea/stanford-cs-229-machine-learning/tree/master/en>



Hình 3.5: Cấu trúc chung của mạng nơ-ron

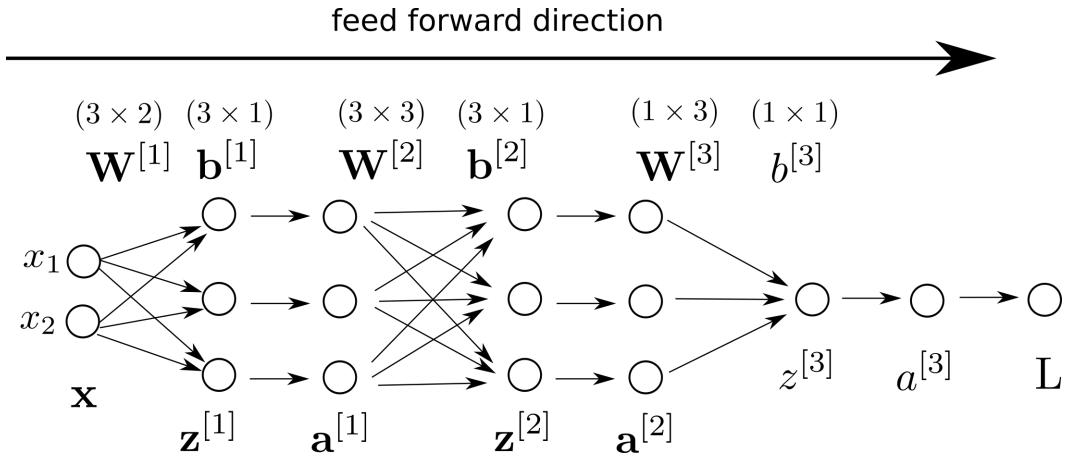
- Tầng đầu vào hay còn gọi là *input layer*: Biểu diễn tổng quát của mỗi quan sát
- Tầng đầu ra hay còn gọi là *output layer*: Thể hiện đầu ra dự đoán của model
- Tầng ẩn hay còn gọi là *hidden layer*: Là tầng thể hiện cấu trúc, suy luận logic của mạng nơ-ron. Hidden layer là các tầng nằm giữa tầng đầu ra và tầng đầu vào.

Số lượng layer trong một mạng nơ-ron được ký hiệu là l và được tính bằng số hidden layer cộng thêm một, ví như mạng nơ-ron trong Hình 3.5 là 3. Người ta gọi mỗi một nơ-ron trong mạng là một *unit*.

3.3.2 Mạng lan truyền thẳng

3.3.2.1 Khái niệm

Mạng lan truyền thẳng (*feedforward*) [8] là mạng nơ-ron mà các nơ-ron được liên kết với nhau theo một hướng với đầu ra của nơ-ron này là đầu vào của nơ-ron tầng tiếp theo. Các nơ-ron đầu vào thực chất không phải các nơron theo đúng nghĩa, bởi lẽ chúng không thực hiện bất kỳ một tính toán nào trên dữ liệu vào, đơn giản nó chỉ tiếp nhận các dữ liệu vào và chuyển cho các tầng kế tiếp. Các nơron ở tầng ẩn và tầng ra mới thực sự thực hiện các tính toán, kết quả được định dạng bởi hàm đầu ra. Cụm từ “truyền thẳng” (feed forward) (không phải là trái nghĩa của lan truyền ngược) liên quan đến một thực tế là tất cả các nơ-ron chỉ có thể được kết nối với nhau theo một hướng: tối đa hay nhiều các nơron khác trong tầng kế tiếp (loại trừ các nơron ở tầng ra). Cụ thể hơn là không có các liên kết từ các nơ-ron ở tầng đầu ra đến các nơ-ron ở các tầng đầu và hay các nơ-ron trong cùng một tầng cũng không có liên kết với nhau.



Hình 3.6: Mạng lan truyền thẳng

Giả sử ta có trọng số (\mathbf{W}), ngưỡng (\mathbf{b}) và cấu trúc mạng nơ-ron như hình 3.6. Khi đó mạng lan truyền thẳng đơn giản là tính đầu ra $a^{[3]}$ thông qua $\mathbf{x}, \mathbf{y}, \mathbf{W}^{[1]}, \mathbf{W}^{[2]}, \mathbf{W}^{[3]}$ và $\mathbf{b}^{[1]}, \mathbf{b}^{[2]}, \mathbf{b}^{[3]}$. Dưới đây là từng bước để tính đầu ra cho mạng lan truyền thẳng:

$$\begin{aligned} \mathbf{z}^{[1]} &= \mathbf{W}^{[1]}\mathbf{x} + \mathbf{b}^{[1]} \\ \mathbf{a}^{[1]} &= f(\mathbf{z}^{[1]}) \\ \mathbf{z}^{[2]} &= \mathbf{W}^{[2]}\mathbf{a}^{[1]} + \mathbf{b}^{[2]} \\ \mathbf{a}^{[2]} &= f(\mathbf{z}^{[2]}) \\ \mathbf{z}^{[3]} &= \mathbf{W}^{[3]}\mathbf{a}^{[2]} + \mathbf{b}^{[3]} \\ \mathbf{a}^{[3]} &= f(\mathbf{z}^{[3]}) \end{aligned}$$

trong đó hàm f là các hàm kích hoạt và có thể khác nhau ở mỗi layer.

3.3.2.2 Thuật toán

Algorithm 4 Forward propagation

Require: Network depth, l
Require: $\mathbf{W}^{(i)}, i \in \{1, \dots, l\}$, ma trận trọng số của model
Require: $\mathbf{b}^{(i)}, i \in \{1, \dots, l\}$ tham số bias của model
Require: $\mathbf{X}, \mathbf{x}_i, i \in \{1, \dots, N\}$, tập dữ liệu đầu vào với N là số lượng dữ liệu.
for $j = 1, \dots, N$ **do**
 $\mathbf{a}^{(0)} = \mathbf{x}_j$
 for $i = 1, \dots, l$ **do**
 $\mathbf{z}^{(i)} = \mathbf{b}^{(i)} + \mathbf{W}^{(i)}\mathbf{a}^{(i-1)}$
 $\mathbf{a}^{(i)} = f(\mathbf{z}^{(i)})$
 end for
 $\hat{\mathbf{y}} = \mathbf{a}^{(L)}$
end for

3.3.3 Hàm mất mát

Hàm mất mát hay còn gọi là *loss function* hoặc *cost function*, là sự chêch lệch, khác biệt giữa đầu ra dự đoán và đầu ra thực tế bằng một số thực không âm, có chức năng là đo độ chính xác của đầu ra dự đoán, thường được ký hiệu là L . Giả sử ta cần ánh xạ: $\mathbf{x} \rightarrow \mathbf{y}$, trong

đó \mathbf{x} là tập các dữ liệu và \mathbf{y} là tập các nhãn tương ứng cho từng dữ liệu thì ta cần *ước lượng các tham số* \mathbf{w} sao cho:

$$\hat{\mathbf{y}} = f(\mathbf{x}, \mathbf{w}) \approx \mathbf{y}$$

Khi đó hàm mất mát là độ chênh lệch, sự khác biệt giữa $\hat{\mathbf{y}}$ và \mathbf{y} . Nếu như giá trị mất mát càng lớn thì điều đó có nghĩa rằng đầu ra dự đoán càng sai, các tham số truyền vào chưa chính xác. Khi đó chúng ta cần điều chỉnh lại các tham số sao cho tập giá trị của $\hat{\mathbf{y}}$ càng gần \mathbf{y} càng tốt. Có nghĩa rằng ta cần tìm \mathbf{w} sao cho độ lệch giữa \mathbf{y} và $\hat{\mathbf{y}}$ nhỏ nhất.

$$L = \text{distance}(\mathbf{y}, \hat{\mathbf{y}})$$

$$\mathbf{w} = \arg \min_{\mathbf{w}} L$$

Tùy theo bài toán thì cách tính *distance* giữa nhãn và đầu ra dự đoán khác nhau, có một số cách tính như: sai số bình phương trung bình (phần 2.4.7.1), hinge, cross-entropy (phần 2.4.7.2),...

Xây dựng hàm mất mát với hàm mất mát là hàm cross entropy

Giả sử ta có mạng lan truyền thẳng như Hình 3.6 với hàm kích hoạt ở tầng cuối cùng là hàm softmax, nhãn là dạng one-hot và hàm cross-entropy 2.4.7.2 là hàm mất mát. Ta xây dựng được hàm mất mát cho dữ liệu thứ i như sau:

$$\begin{aligned} L(\mathbf{x}_i, \mathbf{y}_i) &= - \sum_{j=1}^C y_{ij} \log \hat{y}_{ij} \\ &= - \sum_{j=1}^C y_{ij} \log \frac{e^{z_{ij}^{(l)}}}{\sum_{k=1}^C e^{z_{ik}^{(l)}}} \\ &= - \sum_{j=1}^C \left(y_{ij} z_{ij}^{(l)} - \log \left(\sum_{k=1}^C e^{z_{ik}^{(l)}} \right) \right) \\ &= - \sum_{j=1}^C y_{ij} z_{ij}^{(l)} + \sum_{j=1}^C y_{ji} \log \left(\sum_{k=1}^C e^{z_{ik}^{(l)}} \right) \end{aligned}$$

vì nhãn ở dạng onehot nên ta có $\sum_{j=1}^C y_{ij} = 1$ mà $\log \left(\sum_{k=1}^C e^{z_{ik}^{(l)}} \right)$ là hằng số nên ta có

$\sum_{j=1}^C y_{ji} \log \left(\sum_{k=1}^C e^{z_{ik}^{(l)}} \right) = \log \left(\sum_{k=1}^C e^{z_{ik}^{(l)}} \right)$. Hàm mất mát được rút gọn lại như sau:

$$L(\mathbf{x}_i, \mathbf{y}_i) = - \sum_{j=1}^C y_{ij} z_{ij}^{(l)} + \log \left(\sum_{k=1}^C e^{z_{ik}^{(l)}} \right) \quad (3.1)$$

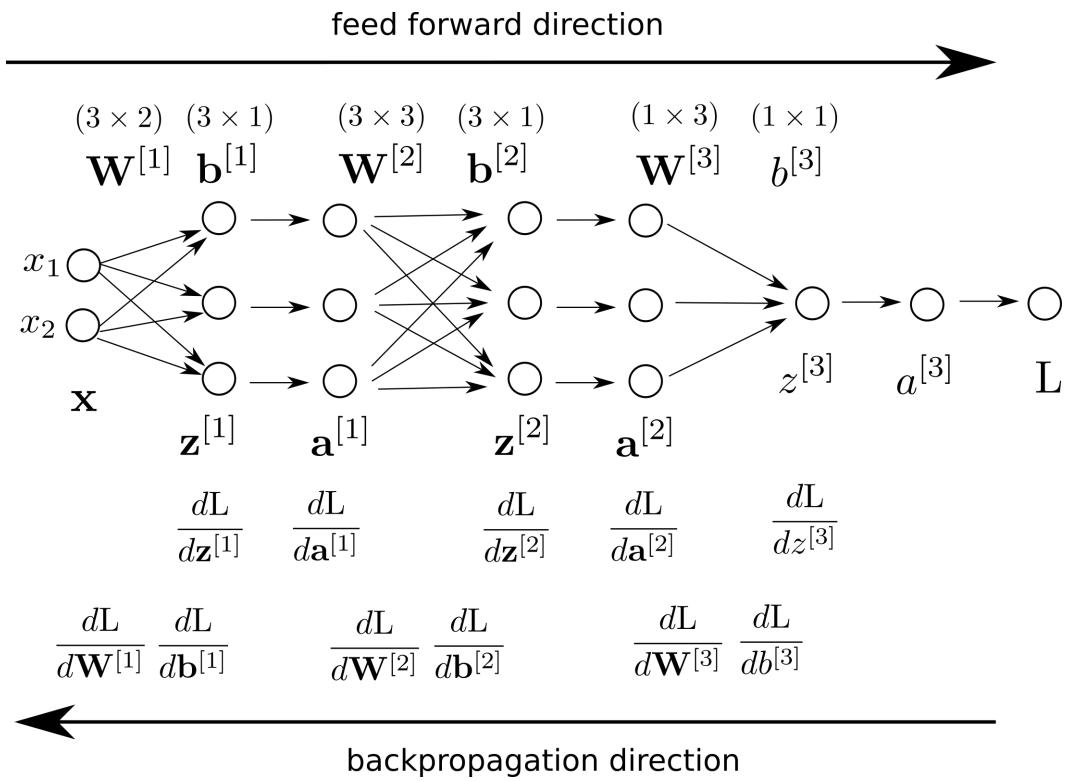
Như vậy đối với tập dữ liệu (\mathbf{x}_i, y_i) , $i = 1, 2, \dots, N$ thì hàm mất mát được tính theo biểu thức sau đây:

$$L(\mathbf{X}, \mathbf{Y}) = - \frac{1}{N} \sum_{i=1}^N \sum_{j=1}^C y_{ij} \log \hat{y}_{ij} = \frac{1}{N} \sum_{i=1}^N \left(- \sum_{j=1}^C y_{ij} z_{ij}^{(l)} + \log \sum_{k=1}^C (e^{z_{ik}^{(l)}}) \right) \quad (3.2)$$

3.3.4 Thuật toán lan truyền ngược

3.3.4.1 Khái niệm

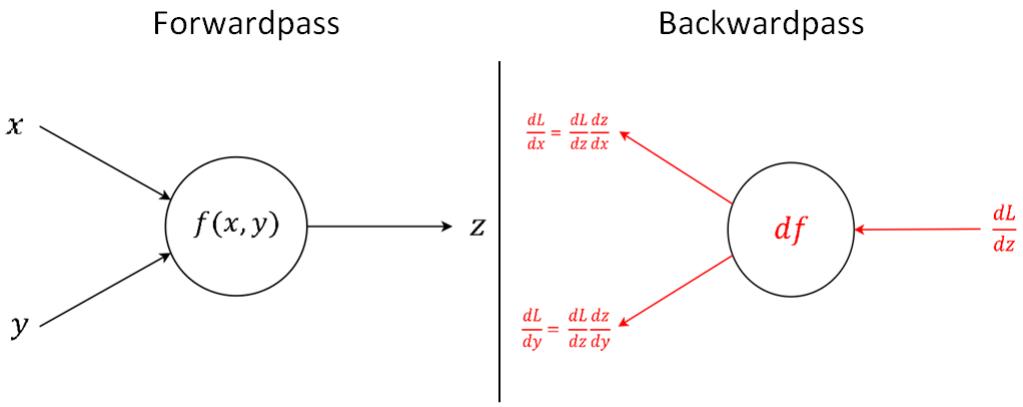
Trong tiếng Anh, lan truyền ngược là backpropagation [8], là một từ viết tắt cho "backward propagation of errors" tức là "truyền ngược của sai số", là một phương pháp phổ biến để huấn luyện các mạng thần kinh nhân tạo được sử dụng kết hợp với một phương pháp tối ưu hóa như gradient descent. Phương pháp này tính toán gradient của hàm mất mát với tất cả các trọng số có liên quan trong mạng nơ ron đó. Gradient này được đưa vào phương pháp tối ưu hóa, sử dụng nó để cập nhật các trọng số thông qua một số thuật toán như gradient descent, gradient descent với momentum, adam, rmsprop để cực tiểu hóa hàm mất mát.



Hình 3.7: Lan truyền ngược

Vì lan truyền ngược đơn giản là tính đạo hàm của hàm mất mát với mỗi tham số trong mạng để đưa vào chúng vào gradient descent và cập nhật lại các tham số trong mạng. Vì thế với mạng nơ-ron tại hình 3.8, các tham số chúng ta cần tính là:

$$\frac{dL}{d\mathbf{W}^{(i)}}, \quad \frac{dL}{d\mathbf{b}^{(i)}} \quad \forall i \in \{1, 2, 3\}$$



Hình 3.8: Ví dụ lan truyền ngược tại một nơ-ron

Sau khi đã có được các tham số đó, chúng ta tiến hành cập nhật lại các tham số theo phương pháp đã chọn.

3.3.4.2 Thuật toán

Algorithm 5 Backpropagation

Require: Network depth, l

Require: $\mathbf{W}^{(i)}, i \in \{1, \dots, l\}$, ma trận trọng số của model

Require: $\mathbf{b}^{(i)}, i \in \{1, \dots, l\}$ tham số bias của model

Require: $\mathbf{x}_i, i \in \{1, \dots, N\}$, tập dữ liệu đầu vào

Require: $\mathbf{y}_i, i \in \{1, \dots, N\}$, đầu ra thực tế

Require: $\hat{\mathbf{y}}$, đầu ra dự đoán

Require: \mathbf{z}, \mathbf{a} , đầu ra của hàm kết hợp và hàm kích hoạt

Require: L , giá trị của hàm mất mát

Sau khi thực hiện thuật toán lan truyền thẳng, tính gradient ở tầng đầu ra:

$$\mathbf{g} \leftarrow \frac{dL}{d\hat{\mathbf{y}}}$$

for $k = 1, \dots, l$ **do**

Từ tầng đầu ra, tính gradient quay lui với hàm kích hoạt ở tầng thứ k theo chain rule

$$\mathbf{g} \leftarrow \frac{dL}{d\mathbf{z}^{(k)}} = \mathbf{g} \odot f'(\mathbf{z}^{(k)})$$

Tính gradient so với trọng số, ngưỡng

$$\frac{dL}{d\mathbf{b}^{(k)}} = \mathbf{g}$$

$$\frac{dL}{d\mathbf{W}^{(k)}} = \mathbf{g}\mathbf{a}^{(k-1)T}$$

Cập nhật trọng số $\mathbf{W}^{(k)}, \mathbf{b}^{(k)}$ theo một số thuật toán như gradient descent, adam, rmsprop,...

Gradient được thực hiện ở tầng đầu ra thấp hơn

$$\mathbf{g} \leftarrow \frac{dL}{d\mathbf{a}^{(k-1)}} = \mathbf{W}^{(k)T} \mathbf{g}$$

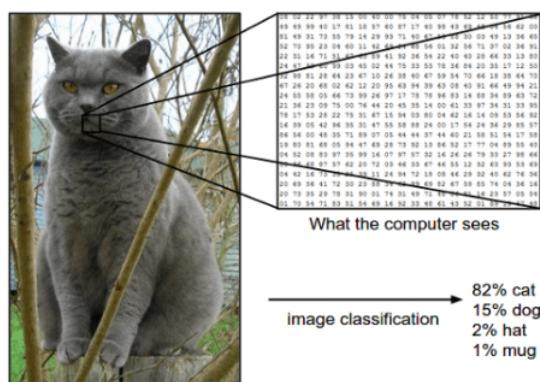
end for

3.4 Mạng nơ-ron tích chập

3.4.1 Tổng quan

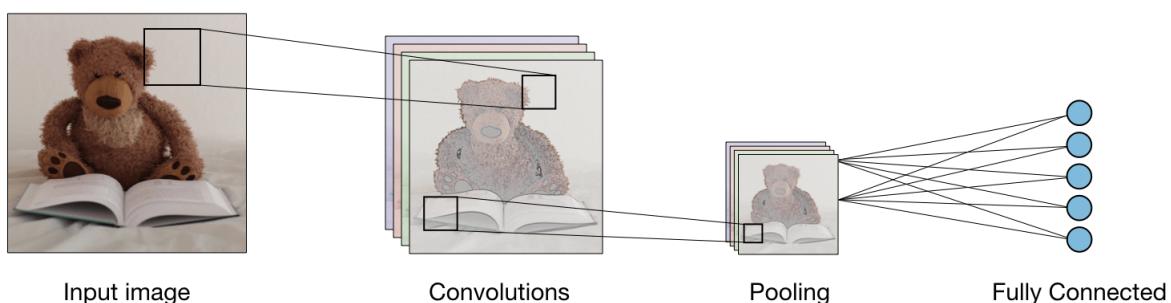
Mạng nơ-ron tích chập là một trong những mô hình deep learning được thiết kế để trích xuất đặc trưng dữ liệu ảnh n chiều.

Tương tự như việc con người học cách nhận diện đối tượng, chúng ta cần cho thuật toán học rất nhiều hình ảnh trước khi nó có thể đưa ra dự đoán cho hình ảnh đầu vào mà nó chưa từng thấy. Máy tính *nhin* theo cách khác con người. Trong thế giới máy tính chỉ có những con số. Mỗi hình ảnh có thể được biểu diễn dưới dạng mảng n chiều, những phần tử trong mảng được gọi là các điểm ảnh. Một bức ảnh dùng làm đầu vào được biểu diễn bởi ma trận các điểm ảnh (từ 0 đến 255) với kích thước $[w \times h \times c]$; trong đó w là số lượng điểm ảnh trên chiều ngang, h là số lượng điểm ảnh trên chiều dọc, c gọi là kênh màu.



Hình 3.9: Cách máy tính nhìn một hình ảnh

Một mạng nơ-ron tích chập gồm hai phần chính là trích xuất đặc trưng và phân loại. Trong đó trích xuất đặc trưng của ảnh là tầng tích chập (*convolutional layer*), tầng giảm số chiều (*pooling layer*) còn phân loại là tầng *fully-connected*.



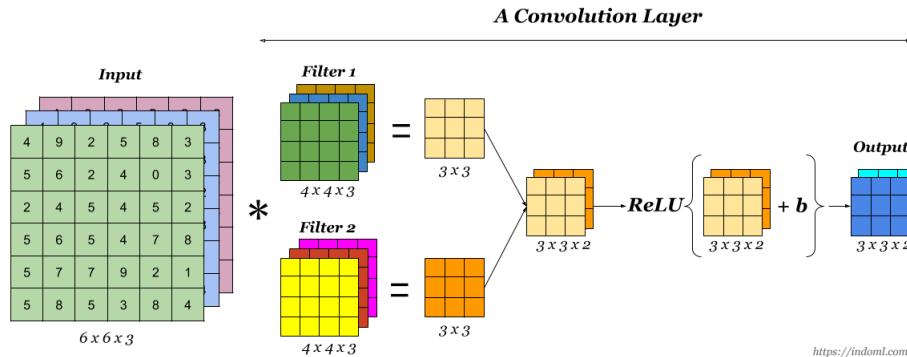
Hình 3.10: Kiến trúc mạng nơ-ron tích chập³

3.4.2 Tầng tích chập - Convolutional layer

Tầng tích chập, tên tiếng anh là *Convolution layer* [8], là khối cốt lõi, cơ bản của ConvNets, làm mạng nơ-ron tích chập trở nên khác biệt so với mạng nơ-ron truyền thống và hoạt động cực kỳ hiệu quả trong bài toán phân tích ảnh.

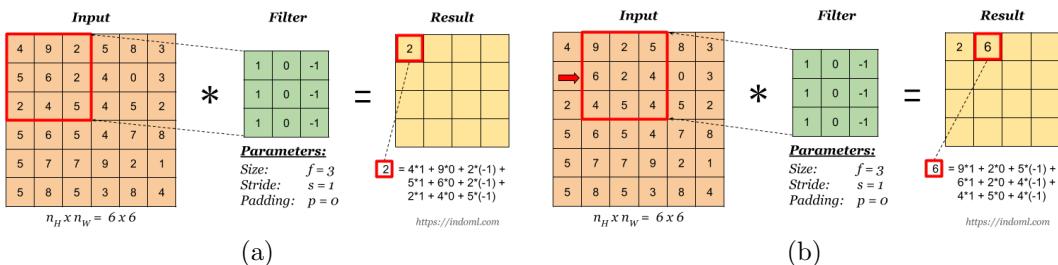
³nguồn: stanford.edu/shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks

Tầng tích chập lấy dữ liệu đầu vào, thực hiện các phép chuyển đổi để tạo ra dữ liệu đầu vào cho tầng kế tiếp (đầu ra của tầng này là đầu vào của tầng sau). Phép biến đổi được sử dụng là phép tính tích chập. Mỗi tầng tích chập chứa một hoặc nhiều bộ lọc - bộ phát hiện đặc trưng (filter - feature detector) cho phép phát hiện và trích xuất những đặc trưng khác nhau của ảnh.



Hình 3.11: Tầng tích chập ⁴

Thay vì kết nối toàn bộ điểm ảnh, tầng này sẽ sử dụng *bộ lọc (filter)* áp vào một vùng trong ảnh có kích thước bằng với bộ lọc và tiến hành tính element-wise giữa bộ lọc và vùng ảnh đó. Vùng ảnh này được gọi *vùng tiếp nhận cục bộ (local receptive filter)*. Ta tiếp tục trượt bộ lọc để có thể quét hết các vùng trong ảnh. Mỗi bước trượt sẽ cho ra một giá trị duy nhất và tập đầu ra này được gọi là feature map. Khi đó ta thu được kết quả sẽ trích xuất ra các đặc trưng của ảnh, có thể là bộ lọc góc, cạnh, đường chéo, hình tròn, hình vuông,... Tiếp tục làm như vậy cho các tầng tiếp theo thì các tầng tiếp theo sẽ lại trích xuất tiếp các đặc trưng của đặc trưng của các đối tượng đó, việc có nhiều layer như vậy cho phép chúng ta chia nhỏ đặc trưng của ảnh tới mức nhỏ nhất có thể.



Hình 3.12: Tích chập ⁵

3.4.2.1 Bộ lọc

Bộ lọc hay còn gọi là filter hoặc kernel, giúp phát hiện và trích xuất các đặc trưng của ảnh, có thể là bộ lọc góc, cạnh, đường chéo, hình tròn, hình vuông. Kích thước của bộ lọc thường là các ma trận có kích thước nhỏ ($3 \times 3, 5 \times 5, 7, \dots$) giúp ta thay đổi các giá trị đầu vào dựa vào các giá trị lân cận theo một nguyên tắc, công thức nào đó. Độ phức tạp của đặc trưng được phát hiện bởi bộ lọc tỉ lệ thuận với độ sâu của tầng tích chập mà nó thuộc về hay bộ lọc ở tầng tích chập càng sâu thì phát hiện các đặc trưng càng phức tạp. Dưới đây là ví dụ về bộ lọc phát hiện biên vật thể trong ảnh hay còn gọi là sobel filter.

⁴nguồn <https://indoml.com>

⁵nguồn <https://indoml.com>

X – Direction Kernel

-1	0	1
-2	0	2
-1	0	1

Y – Direction Kernel

-1	-2	-1
0	0	0
1	2	1

Hình 3.13: Bộ lọc phát hiện biên



(a) Trước khi sử dụng bộ lọc sobel



(b) Sử dụng bộ lọc cạnh ngang

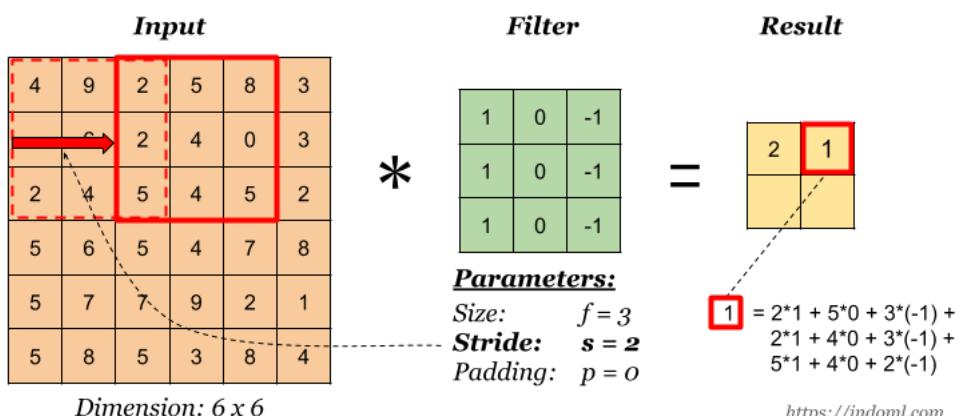


(c) Sử dụng bộ lọc cạnh dọc

Hình 3.14: Kết quả sử dụng bộ lọc sobel

3.4.2.2 Bước trượt

Bước trượt hay còn có tên tiếng anh là slide là khoảng cách dịch chuyển của bộ lọc sau mỗi lần tính và được trượt theo chiều dọc và chiều ngang. Các bước trượt là khoảng cách mà ta dịch chuyển filter trên đầu vào và đó gọi là *stride*, ký hiệu là s . Ví dụ tại Hình 3.12a, 3.12b ta có bước trượt là 1 nhưng ở Hình 3.15 thì bước trượt bằng 2.



Hình 3.15: Bước trượt bằng 2 ⁶

Ở ví dụ trên với ma trận đầu vào kích thước 6×6 nhân chập với bộ lọc kích thước 3×3 . Kết quả thu được với **stride = 1** là ma trận đầu ra kích thước 4×4 vì chỉ có 4×4 vị trí trên ma trận đầu vào để đặt bộ lọc, còn với **stride = 2** thì kết quả là ma trận 2×2 do chỉ đủ 2×2 vị trí trên ma trận đầu vào thỏa mãn để đặt bộ lọc. Như vậy với bước trượt là 2 thì ma trận đầu vào sẽ không được tính ở dòng, cột cuối cùng.

Tổng quát hoá, nếu ta nhân chập ma trận đầu vào kích thước $w \times h$ với bộ lọc kích thước $f \times f$, ta thu được kết quả là một ma trận kích thước

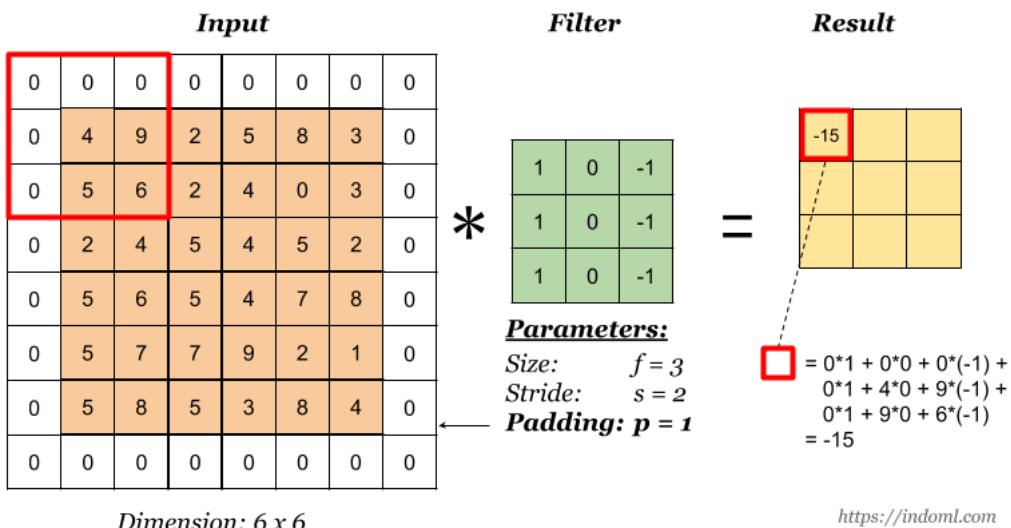
$$(w - f + 1) \times (h - f + 1) \quad (3.3)$$

Và ta có một số nhận xét:

- Nếu chọn không chính xác kích thước bộ lọc thì sẽ dẫn đến một số giá trị đầu vào không được tính toán
- Điểm ảnh ở khoảng trung tâm của ma trận đầu vào được bao phủ bởi rất nhiều vùng bộ lọc nghĩa là được sử dụng để tính nhiều giá trị đầu ra, trong khi những điểm ảnh ở góc hoặc cạnh chỉ được sử dụng 1 hoặc 2 lần vì bộ lọc chỉ trượt qua 1 hoặc 2 lần. Vì thế chúng ta đánh mất rất nhiều thông tin (có thể quan trọng) tại các vùng gần cạnh của ảnh.

3.4.2.3 Thêm lề

Thêm lề, trong tiếng anh còn gọi là padding và kí hiệu là p , nó có tác dụng khắc phục được nhược điểm trên của bước trượt, giúp giữ lại được thông tin ở lề của ảnh. Thêm lề được thực hiện bằng cách ta thêm một đường viền phụ vào xung quanh ma trận đầu, việc này làm tăng kích thước của ma trận đầu vào, dẫn tới tăng kích thước ma trận đầu ra. Từ đó độ chênh lệch giữa ma trận đầu ra với ma trận đầu vào gốc giảm. Những ô nằm trên cạnh/góc của ma trận đầu vào gốc cũng lùi sâu vào bên trong hơn, dẫn tới được sử dụng nhiều hơn trong việc tính toán ma trận đầu ra, tránh được việc mất mát thông tin. Hình 3.17 mô tả sự khác biệt với của đầu ra với hình 3.15 khi ta thêm lề vào đầu vào.



Hình 3.16: Lề bằng 1, bước trượt bằng 2⁷

⁶nguồn <https://indoml.com>

⁷nguồn <https://indoml.com>

Trong hầu hết các trường hợp, đường viền phụ đổi xứng trái-phải, trên-dưới so với ma trận gốc, vì thế kích thước của ma trận đầu vào được tăng lên $2p$ mỗi chiều. Vì thế đầu ra có kích thước

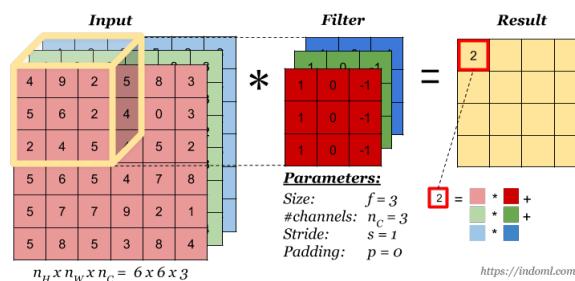
$$(w + 2p - f + 1) \times (h + 2p - f + 1)$$

Theo quy ước, kích thước bộ lọc f là số lẻ vì hai lý do chính sau:

- Nếu f là số chẵn, chúng ta phải thêm vào bên trái của ma trận đầu vào nhiều hơn bên phải (hoặc ngược lại), việc này dẫn tới hệ đầu vào không đối xứng (asymmetric).
- Nếu f là số lẻ, ma trận đầu vào có một điểm ảnh ở trung tâm. Trong lĩnh vực thị giác máy tính, việc có một nhân tố khác biệt (distinguisher) - một điểm đại diện cho vị trí của bộ lọc thường mang lại hiệu năng cao cho bài toán.

3.4.2.4 Tích chập khối

Tích chập khối cũng tương tự như tích chập hai chiều như phần trên đã trình bày nhưng có một điều quan trọng cần phải nhớ đó là độ sâu của bộ lọc phải bằng với độ sâu của đầu vào.



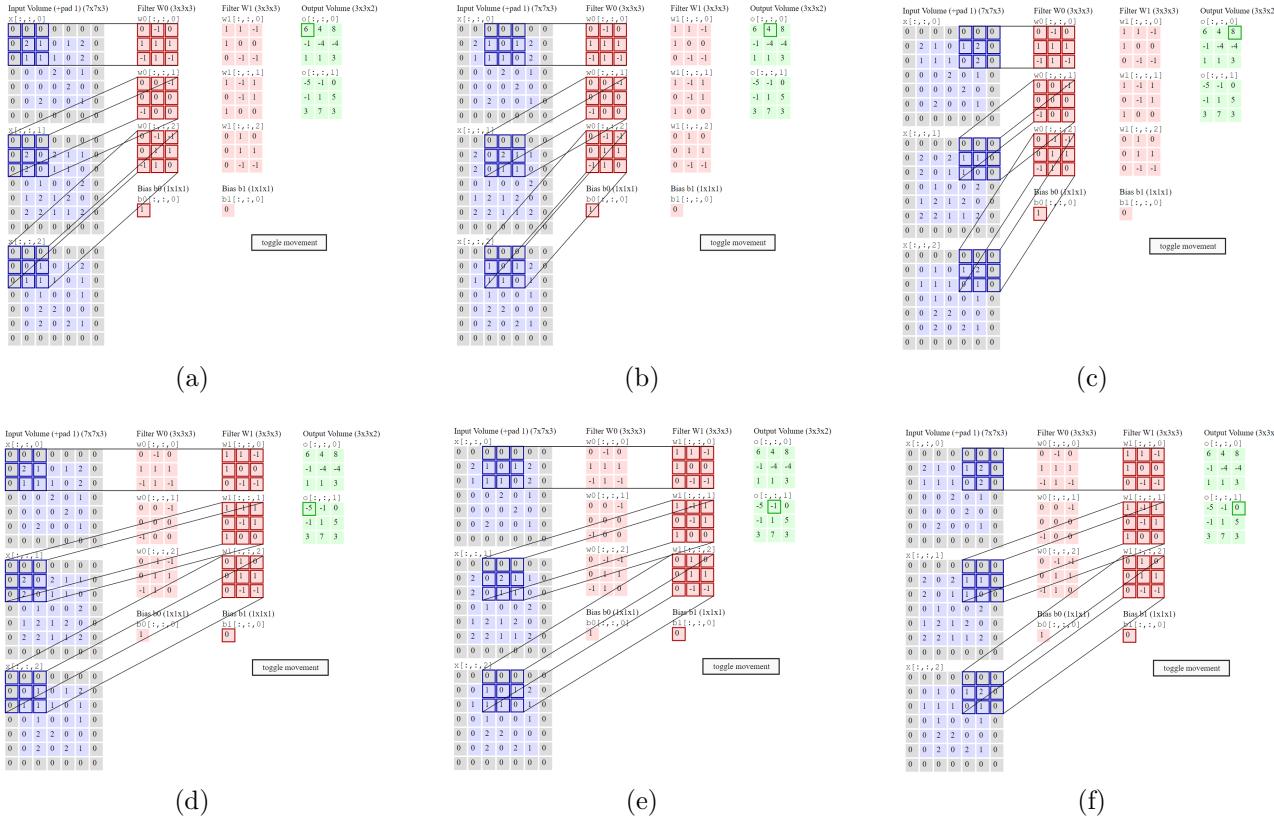
Hình 3.17: Tích chập khối ⁸

Và chúng ta có công thức tính tổng quát cho tích chập khối:

- Kích thước đầu vào $w \times h \times c$,
- Kích thước bộ lọc f ,
- Tốc độ trượt s
- Thêm lè p
- Kích thước đầu ra $(\frac{w + 2p - f}{s} + 1) \times (\frac{h + 2p - f}{s} + 1) \times c$

Dưới đây là một ví dụ về tính tích chập khối với đầu vào là $7 \times 7 \times 3$, filter là $3 \times 3 \times 2$, bước trượt bằng 2

⁸nguồn <https://indoml.com>



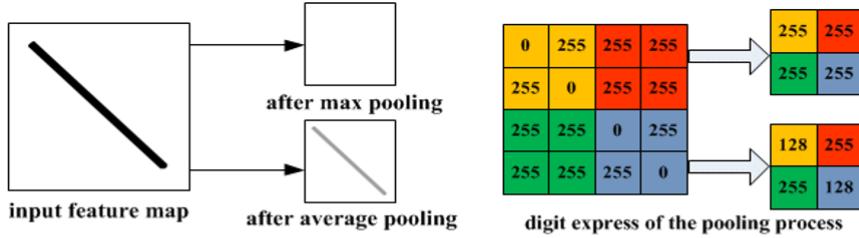
Hình 3.18: Tính chi tiết trong tích chập khối⁹

3.4.3 Tầng tổng hợp

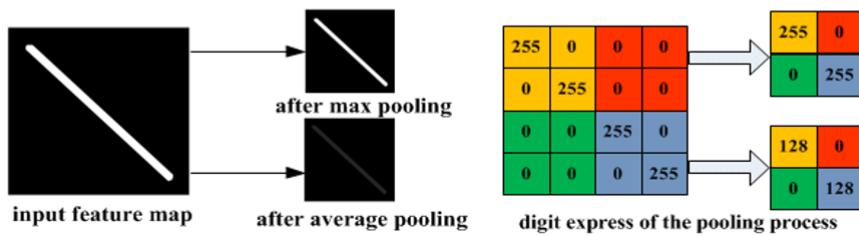
Tầng giảm số chiều (pooling layer) [8] trong mạng CNNs thực hiện công việc loại bỏ dữ liệu không cần thiết sau khi thực hiện tích chập và được chèn giữa các tầng convolutional với nhau hoặc sau một tập tầng convolutional. Nó có vai trò giảm kích thước dữ liệu. Với một bức ảnh kích thước lớn qua nhiều tầng pooling sẽ được thu nhỏ lại tuy nhiên vẫn giữ được những đặc trưng cần cho việc nhận dạng (thông qua cách lấy mẫu). Việc giảm kích thước dữ liệu sẽ làm giảm lượng tham số, tăng hiệu quả tính toán và góp phần kiểm soát hiện tượng quá khớp (overfitting). Tuy nhiên nếu lạm dụng loại layer này cũng có thể khiến data bị mất dữ liệu.

Cách thức hoạt động: Pooling layer sử dụng bộ lọc để trượt trên đầu vào nhưng không thực hiện element-wise, kết quả của mỗi bước trượt là một giá trị được xem là giá trị đại diện cho thông tin ảnh tại vùng đó (giá trị mẫu) được giữ lại và đó gọi là tiến hành lấy mẫu (*subsampling*). Các phương thức lấy phổ biến trong tầng giảm số chiều là *max pooling* (lấy giá trị lớn nhất), *min pooling* (lấy giá trị nhỏ nhất) và *average pooling* (lấy giá trị trung bình). Hình 3.19 mô tả tầng giảm số chiều với bước trượt bằng 2.

⁹nguồn <http://cs231n.github.io/>



(a) Illustration of max pooling drawback



(b) Illustration of average pooling drawback

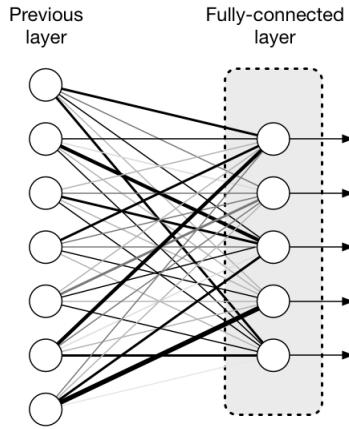
Hình 3.19: Tầng giảm số chiều với bước trượt bằng 2

Ta có ma trận đầu vào 8×8 như hình 3.19, với bước trượt là 2, bộ lọc có kích thước là 2×2 thì kết quả ta thu được là các ma trận 4×4 tương ứng với mỗi phương pháp. Với max pooling ta có kết quả sau mỗi bước trượt là giá trị pixel lớn nhất trong vùng tiếp nhận còn với average pooling thì kết quả là giá trị trung bình pixel của vùng tiếp nhận. Tổng quát hóa, ta có công thức tính kích thước đầu ra cho tầng này như sau:

- Kích thước đầu vào $w \times h \times d$,
- Kích thước bộ lọc f ,
- Tốc độ trượt s
- Kích thước đầu ra $(\frac{w-f}{s} + 1) \times (\frac{h-f}{s} + 1) \times d$

3.4.4 Tầng Fully-Connected (Fully-Connected layer)

Dầu ra từ các tầng chập và tầng tổng hợp thể hiện các đặc trưng cấp cao của ảnh đầu vào. Mục đích của tầng Fully-Connected là sử dụng các đặc trưng này để phân loại hình ảnh đầu vào thành các lớp khác nhau dựa trên tập dữ liệu huấn luyện. Bằng cách làm phẳng (flatten) đầu ra cuối cùng của giai đoạn trước đó và áp dụng mạng nơ-ron truyền thẳng với các thuật toán phân loại để phân loại đối tượng dựa vào vector ta vừa làm phẳng. Hay nói cách khác Fully-Connected chính là một mạng nơ-ron được gắn vào phần cuối của giai đoạn trích xuất đặc trưng với đầu vào là đầu ra của các tầng trước đó. Nó đóng vai trò như một mô hình phân tầng và tiến hành dựa trên dữ liệu đã được xử lý ở các tầng trước đó.

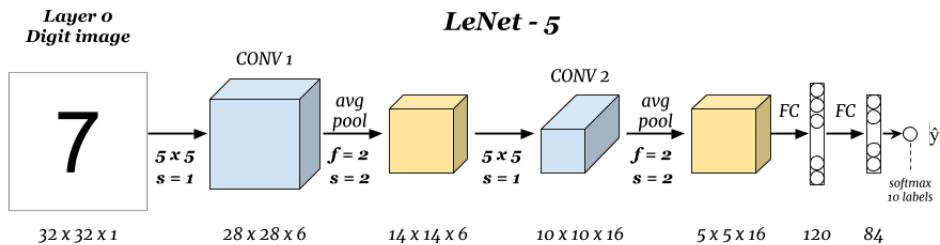


Hình 3.20: Tầng Fully Connected

3.4.5 Một số mạng nơ-ron tích chập điển hình

3.4.5.1 LeNet(1988)

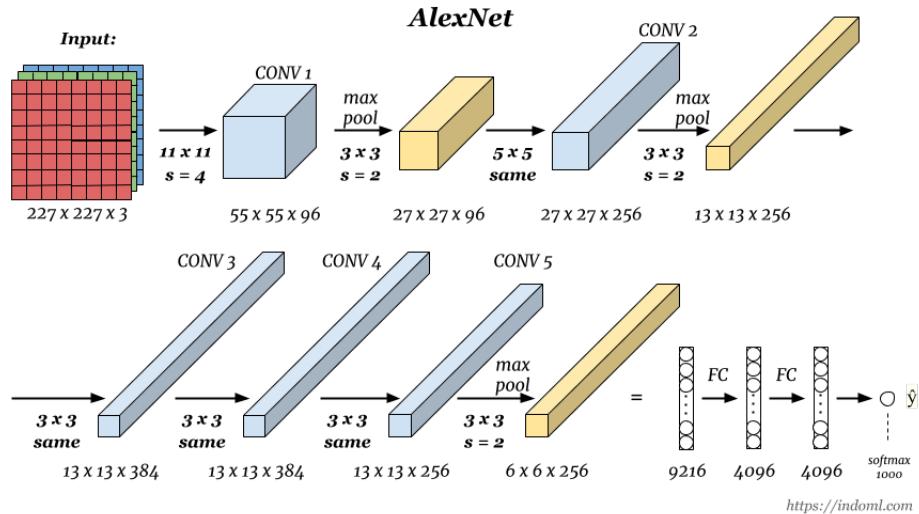
LeNet là một trong những mạng CNN lâu đời nổi tiếng nhất được Yann LeCun phát triển vào những năm 1990s. Cấu trúc của LeNet gồm 2 layer (Convolution + maxpooling) và 2 layer fully connected layer và output là softmax layer.



Hình 3.21: Mạng LeNet-5

3.4.5.2 AlexNet(2012)

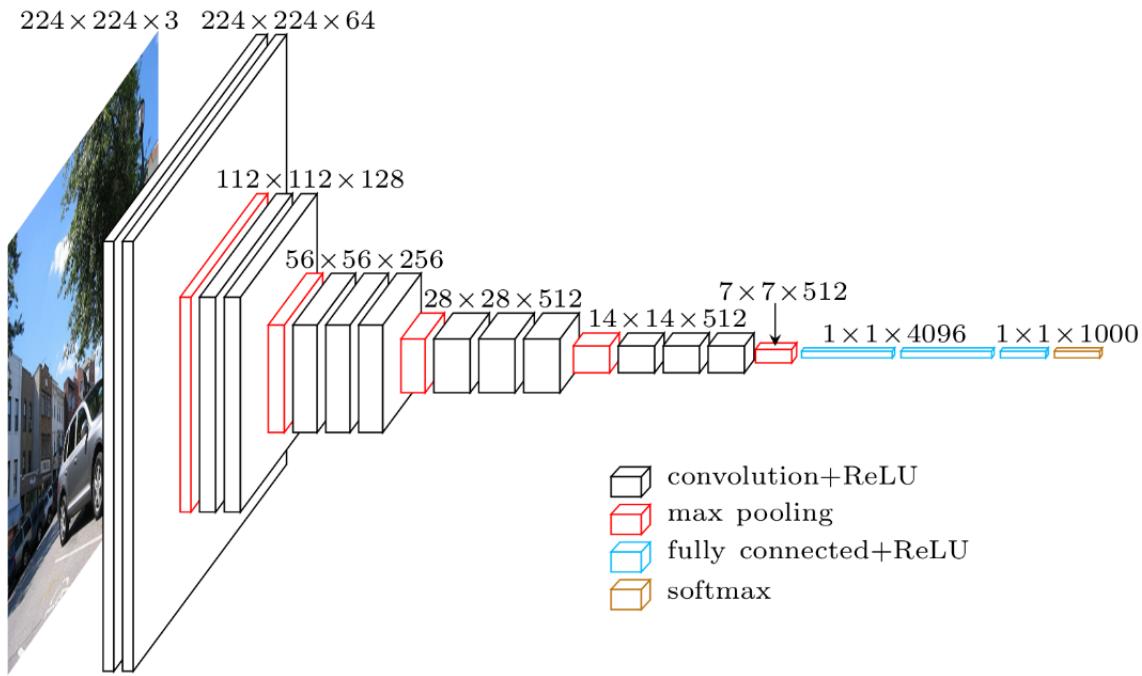
AlexNet là một mạng CNN đã dành chiến thắng trong cuộc thi ImageNet LSVRC-2012 năm 2012 với large margin (15.3% VS 26.2% error rates). AlexNet là một mạng CNN training với một số lượng parameter rất lớn (60 million) so với LeNet. Kiến trúc của Alexnet gồm 5 convolutional layer và 3 fully connection layer. Activation Relu được sử dụng sau mỗi convolution và fully connection layer.



Hình 3.22: Mạng AlexNet

3.4.5.3 VGGNet(2015)

Sau AlexNet thì VGG ra đời với một số cải thiện hơn , trước tiên là model VGG sẽ deeper hơn, tiếp theo là thay đổi trong thứ tự conv. Từ LeNet đến AlexNet đều sử dụng Conv-maxpooling còn VGG thì sử dụng 1 chuỗi Conv liên tiếp Conv-Conv-Conv ở middle và end của architect VGG. Việc này sẽ làm cho việc tính toán trở nên lâu hơn nhưng những feature sẽ vẫn được giữ lại nhiều hơn so với việc sử dụng maxpooling sau mỗi Conv. Hơn nữa hiện nay với sự ra đời của GPU giúp tốc độ tính toán trở nên nhanh hơn rất nhiều lần thì vấn đề này không còn đáng lo ngại. VGG cho small error hơn AlexNet trong ImageNet Large Scale Visual Recognition Challenge (ILSVRC) năm 2014. VGG có 2 phiên bản là VGG16 và VGG19. Kiến trúc của VGG16 bao gồm 16 layer: 13 layer Conv (2 layer conv-conv, 3 layer conv-conv-conv) đều có kernel 3×3 , sau mỗi layer conv là maxpooling downsize xuống 0.5, và 3 layer fully connection. VGG19 tương tự như VGG16 nhưng có thêm 3 layer convolution ở 3 layer conv cuối (thành 4 conv stack với nhau).



Hình 3.23: Mạng LeNet-5

Chương này được tham khảo chính từ tài liệu khoá luận *Nghiên cứu mạng học sâu và ứng dụng cho bài toán nhận dạng biển báo giao thông* [9].

Chương 4

Phương pháp nhận diện và xác thực khuôn mặt

4.1 Giới thiệu tổng quan

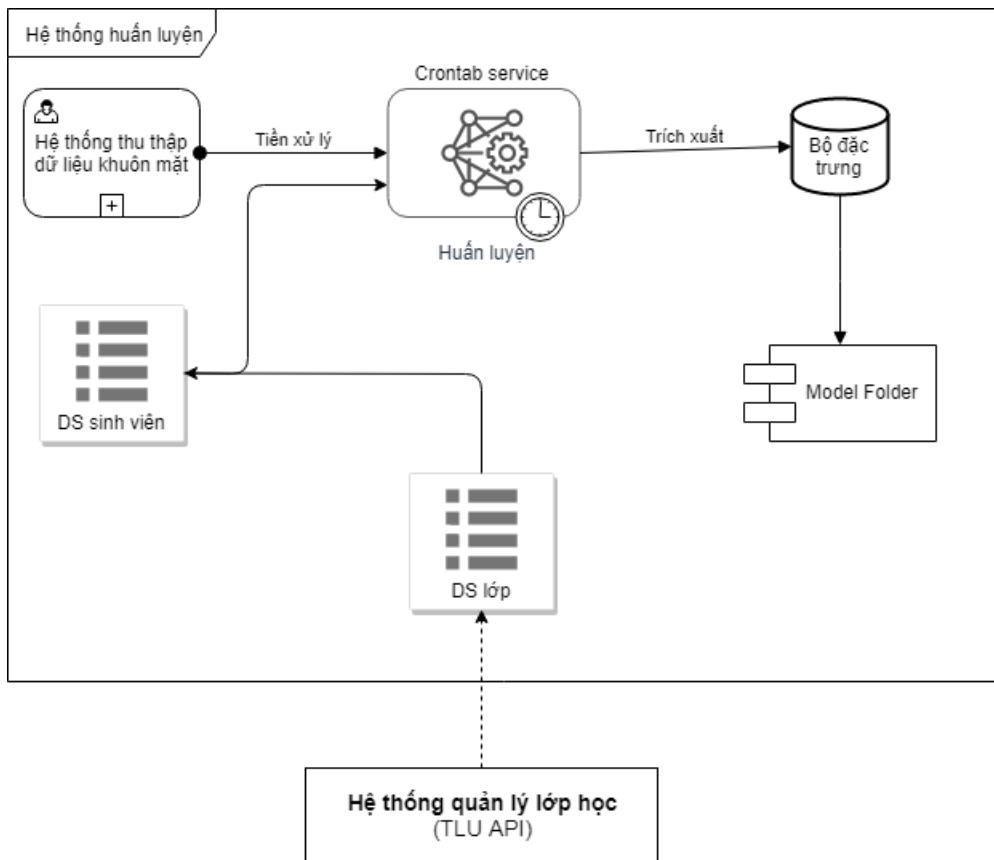
Hiện nay có rất nhiều phương pháp để nhận diện và xác thực khuôn mặt, tuy nhiên điểm chung của các phương pháp này đều thực hiện theo 3 bước:

1. Nhận diện các khuôn mặt có trong khung hình.
2. Từ các khuôn mặt lấy được, thực hiện phân tích, căn chỉnh, trích xuất đặc trưng của các khuôn mặt đó.
3. Sử dụng mô hình và bộ phân loại đặc trưng đã học, xác thực danh tính của khuôn mặt trong khung hình.

Bộ não con người có thể nhận diện khuôn mặt một cách tự động và ngay lập tức. Trên thực tế, con người nhận ra khuôn mặt ai đó khi họ gặp nhau thường xuyên. Máy tính không có khả năng khái quát hóa như vậy. Do đó, chúng ta phải dạy chúng từng bước theo một quy trình.

4.2 Kiến trúc phương pháp

Bài toán xác thực khuôn mặt thuộc bài toán *học có giám sát* (2.4.2). Ta có tập dữ liệu khuôn mặt của sinh viên, là ảnh đầu vào và nhãn (mã sinh viên) được sử dụng cho hai pha chính: pha huấn luyện và pha xác thực.

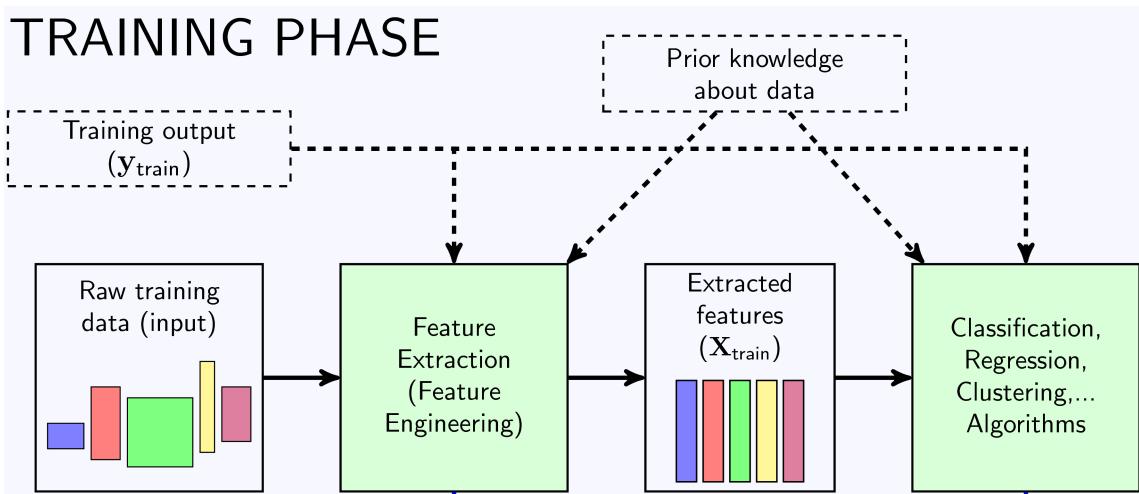


Hình 4.1: Kiến trúc tổng quan thuật toán

Pha huấn luyện sử dụng bộ dữ liệu đã thu thập được để phân loại đặc trưng. Pha xác thực dựa vào bộ phân loại đó để xác thực sinh viên.

4.2.1 Pha huấn luyện

Là pha trích xuất các đặc trưng. Các tính chất riêng biệt của mỗi loại dữ liệu gọi là đặc trưng. Thao tác lấy đặc trưng của dữ liệu gọi là trích xuất. Từ bộ đặc trưng đã phân loại có thể sử dụng để xác thực một đối tượng mới.

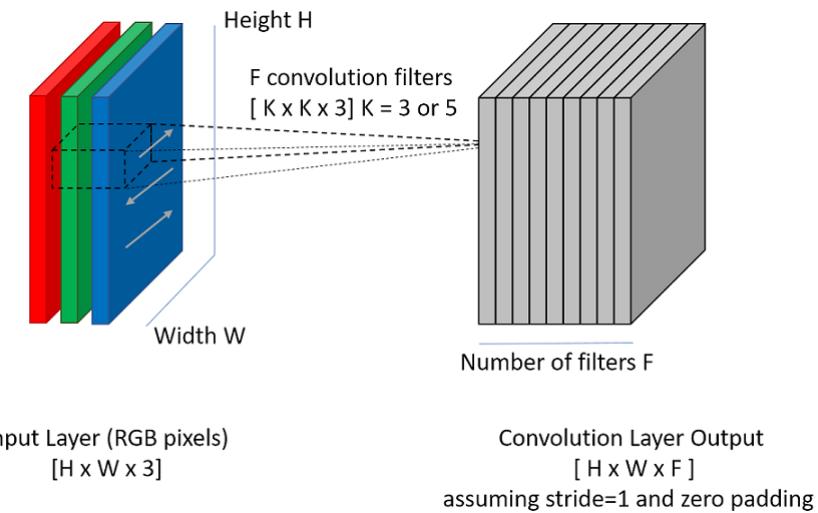


Hình 4.2: Mô tả pha huấn luyện ¹

4.2.1.1 Đầu vào

Bộ dữ liệu ảnh thô đã thu thập được. Dữ liệu thô này thường không ở dạng vector, không có số chiều như nhau. Nó thường là những ảnh màu có 3 khung gian màu Red, Green, Blue (gọi tắt là RGB) hoặc ảnh grayscale.

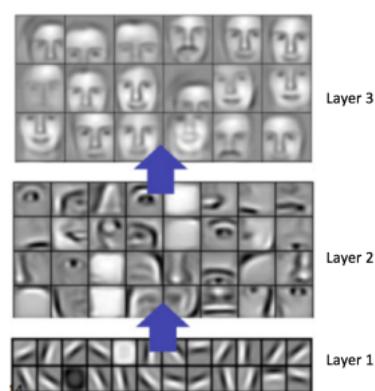
Trong thị giác máy tính, những ảnh đầu vào này được biểu diễn dưới dạng ma trận. Máy sẽ học đặc trưng của ảnh dựa theo các điểm ảnh.



Hình 4.3: Mô tả ảnh đầu vào²

4.2.1.2 Đầu ra

Mục đích của trích xuất đặc trưng là tạo ra một bộ đặc trưng biến dữ liệu thô ban đầu thành dữ liệu được chuẩn hóa, phù hợp với mục đích của mỗi bài toán. Sau khi học qua thuật toán học máy thu được bộ đặc trưng của dữ liệu. Những đặc trưng này được dùng làm đầu vào cho các thuật toán phân loại (Classification), phân cụm (Clustering), hồi quy (Regression),...



Hình 4.4: Mô tả đặc trưng dữ liệu khuôn mặt³

Trong bài toán xác thực khuôn mặt, mục tiêu cuối cùng là thu được bộ phân loại đặc trưng của các khuôn mặt bằng các thuật toán phân loại.

¹nguồn: machinelearningcoban.com/general/2017/02/06/featureengineering/

4.2.2 Pha xác thực

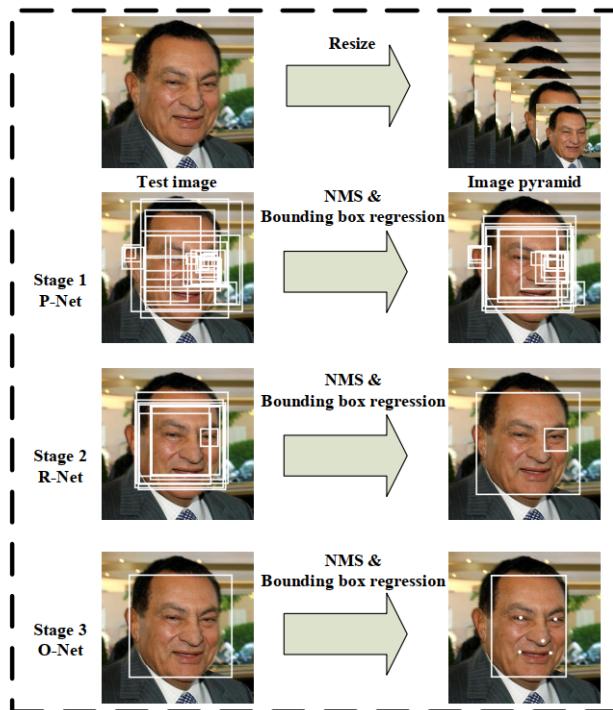
Khi có được bộ phân loại, chúng ta sử dụng những đặc trưng này để xác thực được các khuôn mặt. Để xác thực khuôn mặt, trước đó hệ thống sử dụng thuật toán nhận diện ra vùng chứa khuôn mặt. Sau đó, cắt khuôn mặt và đưa vào bộ phân loại đặc trưng, xác thực khuôn mặt trong khung hình.

4.3 Module 1: Nhận diện khuôn mặt

4.3.1 Phương pháp

Trong các bài toán về thị giác máy tính, các hình ảnh là các ma trận có kích thước khác nhau. Để nhận diện khuôn mặt trong ảnh, ta cần sử dụng các thuật toán phát hiện đối tượng (Object Detection), tức là tìm vùng chứa khuôn mặt chúng ta cần dự đoán. Trong bài toán nhận diện khuôn mặt, chúng ta cần tìm được vị trí các khuôn mặt trong ảnh và cắt các khuôn mặt đó. Có rất nhiều phương pháp phổ biến để phát hiện khuôn mặt trong một bức hình. Khoá luận này đã ứng dụng phương pháp dựa trên bài báo *Joint face detection and alignment using multitask cascaded convolutional networks* [10].

Thuật toán *Multi-task Cascaded Convolutional Networks* [10] (gọi tắt là MTCNN) nhận diện khuôn mặt và căn chỉnh trong điều kiện môi trường không bị giới hạn như nhiều tư thế, ánh sáng và độ tương phản. MTCNN sử dụng nhiều tầng mạng xếp chồng nhau, nhận diện và căn chỉnh khuôn mặt để tăng hiệu xuất nhận diện khuôn mặt. Kiến trúc MTCNN sử dụng kiến trúc xếp chồng nhiều tầng theo ba giai đoạn của mạng tích chập, được thiết kế để dự đoán vị trí khuôn mặt.



Hình 4.5: Kiến trúc tổng quan thuật toán MTCNN ⁴

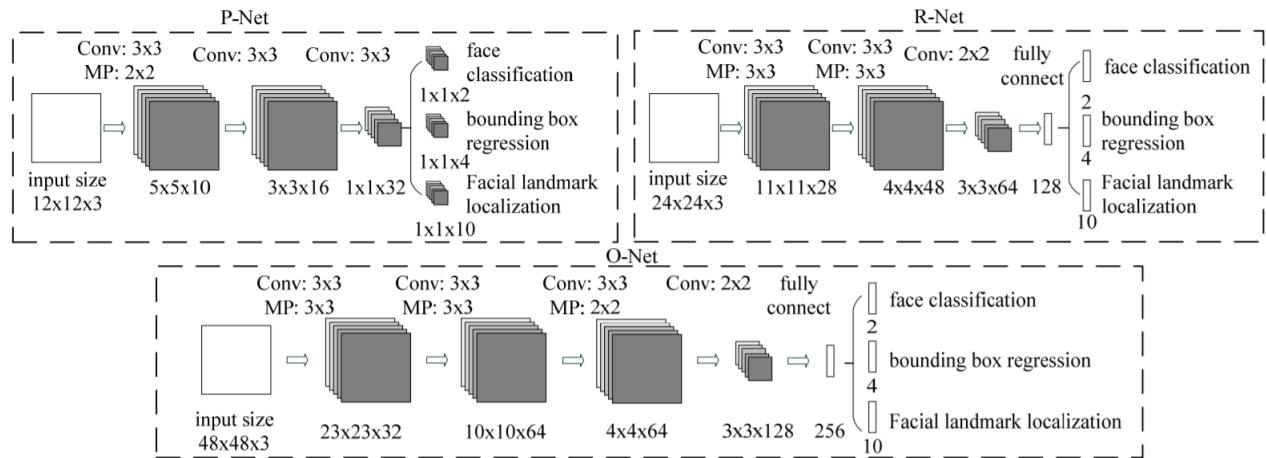
MTCNN bao gồm các tầng mạng tích chập ba tầng, hay ba giai đoạn. Đầu tiên, các bounding box được sinh ra thông qua Mạng đề xuất nhanh (Fast Proposal Network P-Net). Sau đó,

⁴nguồn: Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks

thuật toán tinh chỉnh các bounding box này trong giai đoạn tiếp theo thông qua Mạng sàng lọc (Refinement Network R-Net). Trong giai đoạn thứ ba, Mạng đầu ra (Output Network O-Net) tạo ra bounding box cuối cùng là vị trí khuôn mặt cần dự đoán.

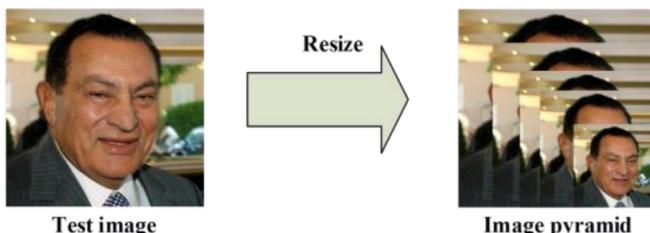
4.3.2 Kiến trúc thuật toán

MTCNN hoạt động theo 3 giai đoạn, mỗi giai đoạn có một mạng nơ ron lần lượt là: P-Net, R-Net và O-Net.



Hình 4.6: Kiến trúc tổng quan thuật toán MTCNN

Với mỗi ảnh đầu vào, nó sẽ tạo ra nhiều bản sao với các tỉ lệ khác nhau, biểu diễn như một kim tự tháp các ảnh (pyramid) để đảm bảo thuật toán có thể phát hiện các khuôn mặt ở tất cả các tỉ lệ khác nhau.



Hình 4.7: Image Pyramid

Giai đoạn 1: MTCNN sử dụng Mạng đề xuất (P-Net). Tại P-Net, thuật toán sử dụng một kernel 12×12 trượt qua mỗi ảnh để tìm kiếm khuôn mặt. Trích xuất được các bounding box dự đoán của khuôn mặt.

Sau đó, các bounding box này được hiệu chuẩn dựa trên các vectơ bounding box regression dự đoán được. Sau đó, sử dụng thuật toán non-maximum suppression (NMS) để lược bỏ các bounding box không cần thiết. Cuối cùng, lấy được ba kết quả đầu ra là các đặc trưng khuôn mặt, tọa độ của các bounding box, vị trí landmark làm đầu vào giai đoạn 2.

Giai đoạn 2: Tất cả các bounding box dự đoán từ giai đoạn 1 được đưa đến Mạng sàng lọc (R-Net). R-Net có cấu trúc tương tự với P-Net. Tuy nhiên sử dụng nhiều tầng mạng hơn. Tại đây, mạng sẽ sử dụng các bounding box được trích xuất từ P-Net và tinh chỉnh lại tọa độ. Trong mạng R-Net thêm một tầng FCs nhằm kết nối đầy đủ thông tin đặc trưng làm kết quả

đầu vào cho giai đoạn sau. Đồng thời cũng tiếp tục loại bỏ thêm một số lượng lớn các bounding box nhiễu, bằng cách thực hiện hiệu chuẩn với bouding box regression và thuật toán NMS.

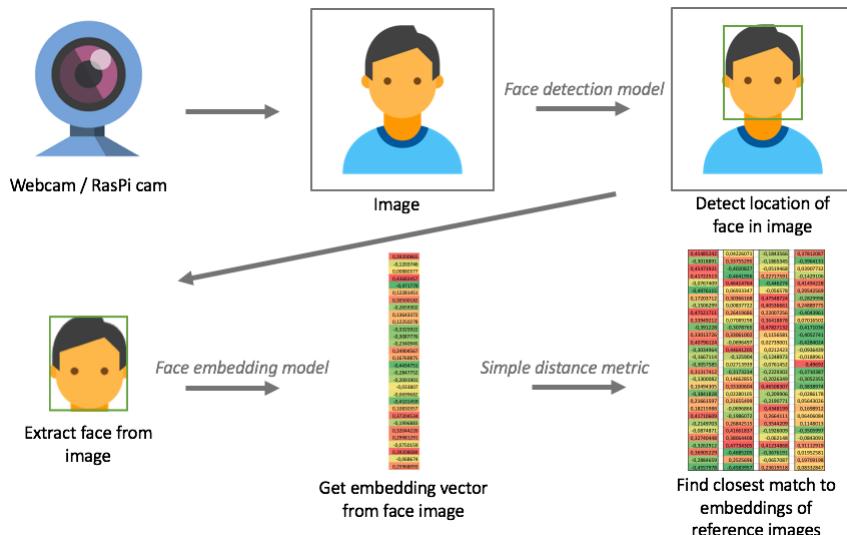
Giai đoạn 3: Giai đoạn này tương tự như giai đoạn 2. Nhưng trong giai đoạn này, thuật toán hướng đến việc xác định các vùng khuôn mặt với sự giám sát nhiều hơn. O-Net lấy các bounding box từ R-Net làm đầu vào và đánh dấu các tọa độ của các landmarks trên khuôn mặt. Đặc biệt, mạng sẽ xuất ra năm vị trí trên khuôn mặt, đó là các vị trí mắt, mũi và miệng.

Dầu ra cuối cùng của toàn bộ ba giai đoạn bao gồm: tọa độ của bounding box, tọa độ của 5 landmarks (2 mắt, 1 mũi, 2 bên miệng) và độ tin cậy của bounding box.

4.4 Module 2: Xác thực khuôn mặt

4.4.1 Phương pháp

Hệ thống xác thực khuôn mặt là một hệ thống máy tính tự động xác thực một khuôn mặt từ một ảnh kỹ thuật số hoặc một khung hình video. Một trong những cách để thực hiện điều này là so sánh các đặc trưng khuôn mặt trong khung hình nhận được với bộ phân loại dữ liệu đã phân loại.



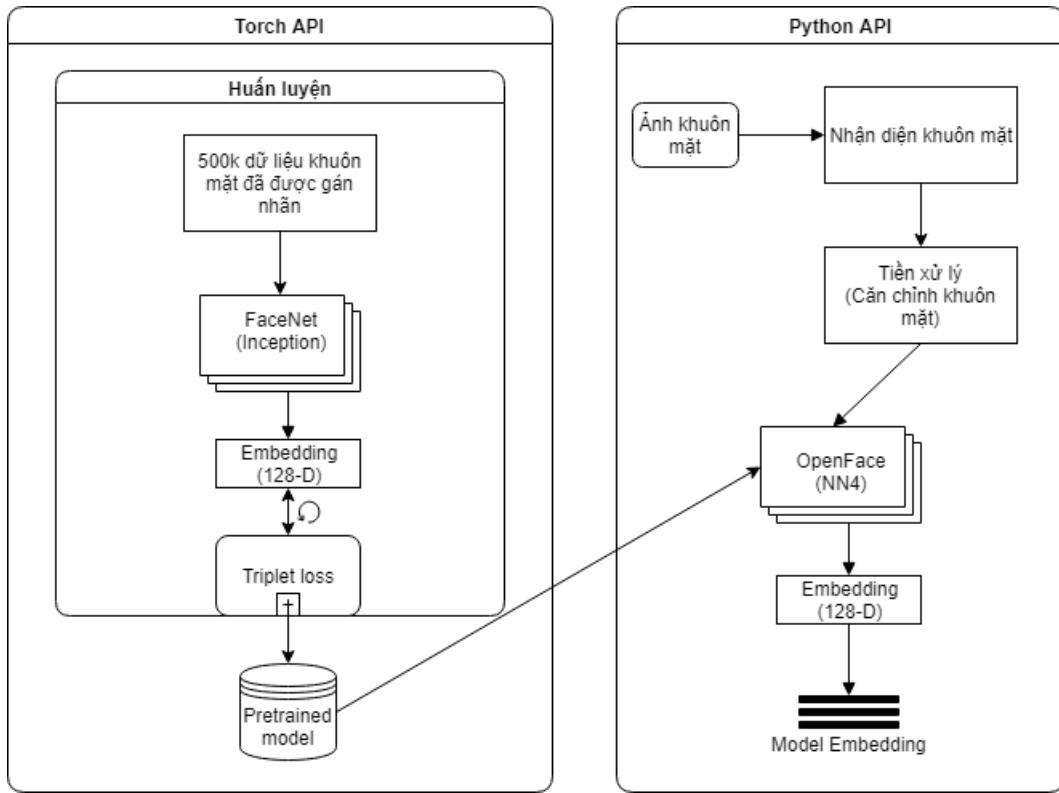
Hình 4.8: Embedding

Công nghệ học sâu thời điểm hiện tại có rất nhiều thuật toán, thư viện nổi bật về khả năng xử lý xác thực khuôn mặt như FaceNet, OpenFace, OpenCV, Amazon Rekognition, FaceX, Google Cloud Vision,... Khoá luận này sử dụng thuật toán OpenFace [11] bởi đây là một thư viện mã nguồn mở cùng với độ chính xác thuật toán khá cao.

Sau khi nhận diện khuôn mặt là việc phân tích khuôn mặt đó. Con người xác thực khuôn mặt con người qua các đặc điểm nhận diện nhưng máy tính không có khả năng khái quát như vậy. Để xác thực khuôn mặt này là của ai, chúng ta cần phân biệt giống/khác nhau giữa các đặc trưng của khuôn mặt. Để đơn giản, thuật toán sẽ quy về bài toán tính khoảng cách giữa các vector trong không gian hai chiều. OpenFace sẽ biểu diễn các khuôn mặt về các vector có 128 *đo lường* (gọi là embedding). Sau đó, tìm khuôn mặt có khoảng cách đo lường gần nhất trong tập dữ liệu gần nhất với khuôn mặt cần xác thực và dự đoán được nhãn của đối tượng đó.

4.4.2 Kiến trúc thuật toán

Thuật toán OpenFace [11] được chia làm bốn thành phần chính: tiền xử lý khuôn mặt, mô hình, nhận diện khuôn mặt, phân loại khuôn mặt.



Hình 4.9: Kiến trúc tổng quan thuật toán OpenFace ⁵

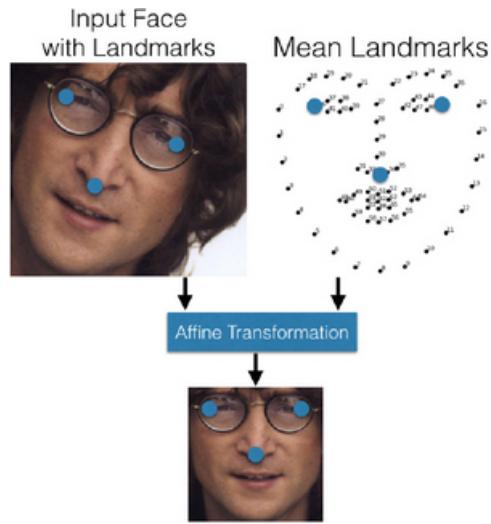
4.4.2.1 Tiền xử lý khuôn mặt

Trước khi đưa dữ liệu vào mô hình huấn luyện, một bước quan trọng là tiền xử lý dữ liệu. Bước này làm cho dữ liệu thô có thể thích hợp với các mô hình học máy nhằm trích được đặc trưng tốt nhất.

Thuật toán OpenFace sử dụng phương pháp căn chỉnh hướng khuôn mặt để giải quyết vấn đề dữ liệu khuôn mặt được chụp có góc độ khác nhau về cùng hướng. Mục tiêu của bước này là đưa mọi khuôn mặt trong toàn bộ quy trình thuật toán về một hướng cố định. Thuật toán biến đổi bức ảnh sao cho mắt, mũi và môi ở cùng một vị trí hướng về giữa ảnh. Điều này cũng khiến việc so sánh khuôn mặt ở bước kế tiếp dễ dàng hơn.

Sau đó, sử dụng thuật toán face landmark estimation [12]. Ý tưởng đơn giản của thuật toán này là xác định 68 điểm (gọi là landmarks - như trong châm cứu) tồn tại trên mỗi khuôn mặt. OpenFace sử dụng phép biến đổi affine không gian hai chiều, giúp xoay khuôn mặt và làm cho vị trí của mắt, mũi và miệng phù hợp với từng khuôn mặt. Có 68 landmarks được sử dụng trong phép affine để phát hiện đặc điểm và khoảng cách giữa các điểm đó được đo và so sánh với các điểm được tìm thấy trong khuôn mặt.

⁵nguồn: OpenFace: A general-purpose face recognition library with mobile applications



Hình 4.10: Chuyển đổi theo hàm affine⁶

4.4.2.2 Mô hình

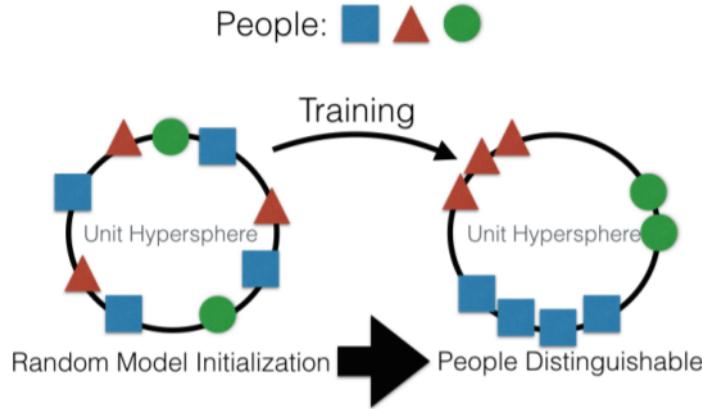
Trong phần huấn luyện của OpenFace, 500.000 ảnh được huấn luyện qua mạng nơ ron. Những hình ảnh này được lấy từ hai bộ dữ liệu mở: CASIA-WebFace, bao gồm 10.575 người với tổng số 494.414 ảnh và FaceScrub, được tạo thành từ 530 người với tổng số 106.863 ảnh [13].

Mục tiêu khi huấn luyện 500.000 ảnh này để trích xuất ra các véc tơ gồm 128 độ lường là đặc trưng cơ bản của mỗi khuôn mặt. Sau đó, thay vì so khớp một hình ảnh trong không gian đa chiều, OpenFace chỉ sử dụng dữ liệu chiều thấp, giúp thuật toán có tốc độ tính toán nhanh.

Như đã đề cập trước đó, OpenFace sử dụng kiến trúc mô hình tương tự FaceNet [1] để trích xuất đặc trưng và sử dụng hàm mất triplet (2.4.7.4) để đánh giá mức độ chính xác của tầng phân loại. Nó thực hiện điều này bằng cách huấn luyện trên ba hình ảnh khác nhau, trong đó một hình ảnh khuôn mặt đầu vào, một hình ảnh khác của cùng một người, trong khi ảnh thứ ba là ảnh một người hoàn toàn khác. Mục tiêu của cuối cùng của quá trình huấn luyện với hàm mất mát bộ ba số với mỗi ảnh đã được mã hóa được đo trên khoảng cách euclidean nhằm xác định ảnh có tính chất giống nhau gần nhau hơn và ảnh có tính chất khác nhau xa nhau hơn.

Sau khi lặp lại quá trình huấn luyện lần với hàng nghìn bức ảnh của nhiều người khác nhau, mạng nơ ron học được cách tạo ra vector có 128 độ lường thể hiện đặc trưng khuôn mặt cho mỗi người.

⁶nguồn: OpenFace: A general-purpose face recognition library with mobile applications



Hình 4.11: Minh họa về quy trình huấn luyện theo hàm tính lối triplet của FaceNet ⁷

Khi mô hình đã được huấn luyện, nó có thể tạo ra các phép đo cho bất kỳ khuôn mặt nào, ngay cả những khuôn mặt chưa từng thấy. Vì vậy, bước này chỉ cần được thực hiện một lần. May mắn, OpenFace đã làm điều này và thư viện đã cung cấp một số mô hình đã được huấn luyện (5.1) mà có thể trực tiếp sử dụng ngay.

4.4.2.3 Nhận diện và xác thực khuôn mặt

OpenFace sử dụng phương pháp Histogram of Oriented Gradient (HOG) kết hợp Support Vector Machine (SVM) để nhận diện khuôn mặt. Tuy nhiên, khoá luận sử dụng phương pháp tốt hơn, đó là MTCNN để nhận diện, như đã trình bày ở phần (4.3.1).

Sau khi nhận diện và tiền xử lý khuôn mặt (4.4.2.1) sau đó biến diễn khuôn mặt dưới một vector 128 đo lường bằng cách sử dụng mô hình mạng được trình bày trong phần (4.4.2.2).

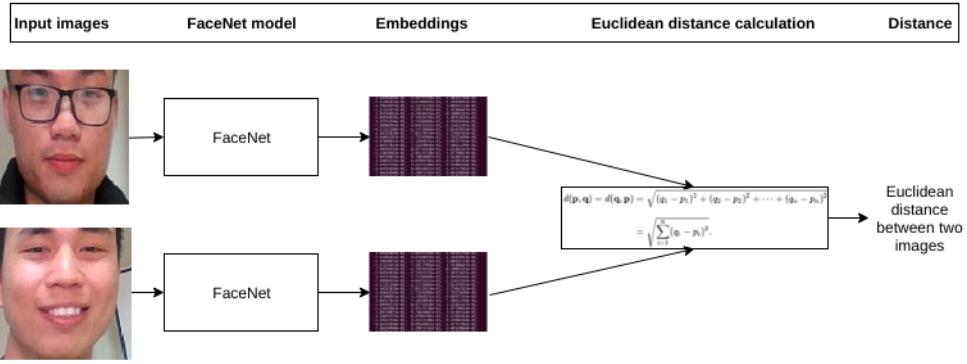
Việc biểu diễn khuôn mặt dưới một vector 128 chiều bằng mô hình tương tự như FaceNet, kết quả trả về có thể được sử dụng để xác thực khuôn mặt một cách hiệu quả. Giá trị khoảng cách giữa hai khuôn mặt trong không gian Euclidean (4.13) sẽ gần hơn đối với các khuôn mặt tương tự và xa hơn đối với các mặt không giống nhau.

$$\begin{aligned}
 d(\mathbf{p}, \mathbf{q}) = d(\mathbf{q}, \mathbf{p}) &= \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} \\
 &= \sqrt{\sum_{i=1}^n (q_i - p_i)^2}.
 \end{aligned}$$

Hình 4.12: Khoảng cách Euclidean giữa hai điểm

Để so sánh hai khuôn mặt, trích xuất đặc trưng cho cả hai khuôn mặt bằng cách thông qua mô hình riêng biệt. Sau đó, sử dụng khoảng cách Euclidean để tìm khoảng cách. Khoảng cách có giá trị thấp hơn là các khuôn mặt giống nhau và giá trị cao hơn cho các khuôn mặt khác nhau.

⁷nguồn: OpenFace: A general-purpose face recognition library with mobile applications



Hình 4.13: Khoảng cách Euclide giữa hai vector đặc trưng

Dối với mỗi khuôn mặt nhận diện được, thuật toán tính khoảng cách euclidean giữa khuôn mặt đó với từng khuôn mặt có trong tập dữ liệu. Khuôn mặt có khoảng cách euclidean *nhỏ nhất* có thể được coi là nhãn của cùng một người.

4.5 Module 2: Theo dõi khuôn mặt

4.5.1 Phương pháp

Ngoài những giải pháp về nhận diện khuôn mặt chúng ta cần theo dõi khuôn mặt nhằm giảm tải cho việc hệ thống liên tục phải nhận diện liên tục. Chúng ta có thể phát hiện các khuôn mặt trong khung hình một lần duy nhất. Khi có nhiều khuôn mặt được phát hiện trong khung hình, việc theo dõi giúp xác thực danh tính của các sinh viên trên các khung hình chính xác hơn. Trong một số trường hợp, nhận diện khuôn mặt có thể sai nhưng vẫn có thể theo dõi khuôn mặt đó vì theo dõi vị trí và sự xuất hiện của khuôn mặt trong khung hình trước đó. Vì vậy, có thể có được FPS cao cho hệ thống.

Dối với những khuôn mặt được theo dõi, bắt đầu với tất cả các phát hiện có thể có trong một khung hình và cung cấp cho khuôn mặt đó một ID là mã sinh viên của sinh viên đó. Trong các khung hình tiếp theo, hệ thống liên tục theo dấu ID đó. Nếu trường hợp sinh viên rời khỏi khung hình thì ID đó sẽ bị hủy. Nếu một sinh viên mới xuất hiện thì sinh viên đó được hệ thống xác nhận bắt đầu với một ID mới.

Phổ biến nhất và là một trong những thuật toán đơn giản nhất để theo dõi là SORT (Simple Online and Realtime Tracking) [14]. Nó 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 hình khác nhau dựa trên bounding box từ kết quả nhận diện, như sau:



Hình 4.14: Theo dõi khuôn mặt sinh viên khi vào lớp

Ý tưởng của thuật toán này là sử dụng thuật toán lọc kalman. SORT không cần phải biết loại đối tượng đang theo dõi. Để thực hiện các liên kết, SORT sử dụng các thuật toán như IOU giao nhau giữa các bounding boxes trong các khung hình liên tiếp. Mỗi bounding box gán được gán ID và nếu không có box liên quan trong khung tiếp theo, thuật toán giả định rằng đối tượng đã rời khỏi khung hình.

Cách tiếp cận như vậy phụ thuộc rất nhiều vào độ chính xác của bước nhận diện khuôn mặt. Quan điểm của bài báo SORT ban đầu là chỉ ra rằng các thuật toán nhận diện đã tiến bộ đến mức không cần phải làm bất cứ điều gì về việc theo dõi và có thể đạt được kết quả tốt với các phương pháp dự đoán đơn giản. Kể từ đó, các cải tiến đã xuất hiện, đặc biệt là thế hệ tiếp theo của thuật toán SORT, Deep SORT (SORT ra mắt vào năm 2016 và Deep SORT đã có trong năm 2017). Nó được thiết kế đặc biệt để giảm số lượng chuyển đổi giữa các danh tính, đảm bảo rằng việc theo dõi ổn định hơn. Tuy nhiên, khoa luận này chỉ dùng lại sử dụng thuật toán SORT. Thuật toán Deep SORT có thể được ứng dụng trong phiên bản sau.

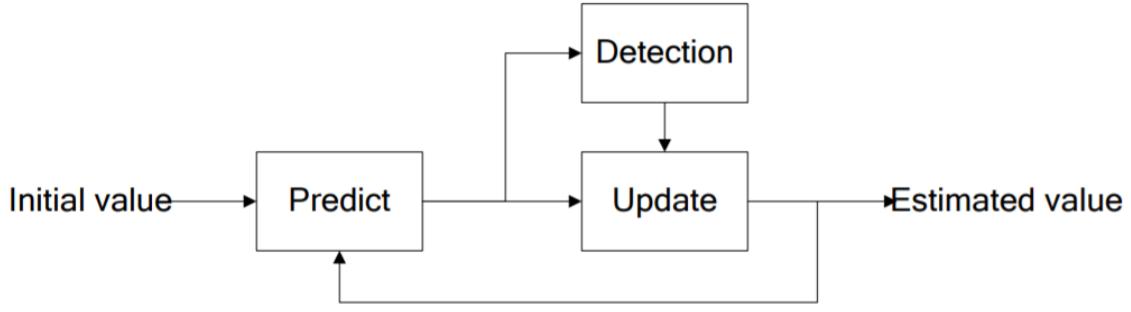
4.5.2 Kiến trúc thuật toán

4.5.2.1 Thuật toán lọc Kalman

Thuật toán SORT dựa vào ý tưởng cốt lõi từ thuật toán lọc Kalman [14].

Trong lý thuyết thống kê, lọc Kalman, còn được gọi là ước lượng bậc hai tuyến tính (LQE), là một thuật toán sử dụng một loạt các phép đo được quan sát theo *thời gian*, chứa nhiều thông kê và các điểm không chính xác, và đưa ra các ước tính về các biến chưa biết có xu hướng nhiều hơn chính xác hơn so với chỉ dựa trên một phép đo duy nhất, bằng cách ước tính phân phối xác suất chung trên các biến cho từng khung thời gian.

Thuật toán hoạt động theo quy trình gồm hai bước. Trong bước *dự đoán*, bộ lọc Kalman tạo ra các ước tính về trạng thái hiện tại, cùng với trạng thái lưỡng lự của đối tượng. Khi kết quả của phép đo tiếp theo được quan sát, các ước tính này được *cập nhật* bằng cách sử dụng trung bình có trọng số, với trọng số được đưa ra để ước tính với độ chắc chắn cao hơn.



Hình 4.15: Sơ đồ của phương pháp theo dõi chuyển động đối tượng⁸

Thiết kế của các mô hình dự đoán và cập nhật là khía cạnh quan trọng của lý thuyết lọc Kalman. Trong thuật toán này, chuyển động của một đối tượng trong hình ảnh hai chiều có thể được xem là sự kết hợp của các chuyển động trong trục x và trục y và hai thành phần chuyển động này có thể được xử lý độc lập.

Giả sử có các đối tượng chuyển động M được phát hiện, các thành phần chuyển động 2M (tức là, các thành phần M trong trục x và các thành phần M trong trục y) là cần thiết để ước tính để theo dõi. Ở đây vectơ đo Y có thể được biểu thị là:

$$Y = [x_1, y_1, x_2, y_2, \dots, x_M, y_M]^T \quad (4.1)$$

Trong đó, $(x_i, y_i), i = 1, \dots, M$ biểu diễn thông tin vị trí cho đối tượng được phát hiện thứ i trong trục x và hướng trục y.

Đối với mỗi chuyển động, chúng ta cần sử dụng một phương trình thích hợp để phản ánh động lực học, vì vậy cả hai vận tốc và gia tốc của vật đều được xem xét. Do đó, biến trạng thái X được định nghĩa là:

$$X = [x_1, v_{x1}, a_{x1}, y_1, v_{y1}, a_{y1}, \dots, x_M, v_{xM}, a_{xM}, y_M, v_{yM}, a_{yM}]^T \quad (4.2)$$

Trong đó, $(v_{xi}, a_{xi}), i = 1, \dots, M$ và $(a_{xi}, a_{yi}), i = 1, \dots, M$ lần lượt biểu thị vận tốc và gia tốc của đối tượng thứ i. Do đó, mô hình di chuyển cho đối tượng thứ i được xác định bởi một hệ phi tuyến như sau:

$$\begin{aligned} x_i(n) &= x_i(n-1) + v_{xi}(n-1)dt + \frac{1}{2}a_{xi}(n-1)dt^2 \\ y_i(n) &= y_i(n-1) + v_{yi}(n-1)dt + \frac{1}{2}a_{yi}(n-1)dt^2 \end{aligned} \quad (4.3)$$

Theo đó, ma trận chuyển tiếp có thể được biểu diễn dưới dạng:

$$F = \begin{bmatrix} F_1 & 0 & \dots & 0 \\ 0 & \ddots & & \\ \vdots & & F_1 & \vdots \\ 0 & \dots & 0 & F_M \end{bmatrix} \quad (4.4)$$

⁸nguồn: Tracking Multiple Moving Objects Using Unscented Kalman Filtering Techniques

Xét dạng biến đo lường Y, ma trận đo H được định nghĩa là:

$$H = \begin{bmatrix} H_1 & 0 & \dots & 0 \\ 0 & \ddots & & \\ \vdots & & H_1 & \vdots \\ & & \ddots & 0 \\ 0 & \dots & 0 & H_M \end{bmatrix} \quad (4.5)$$

Dựa trên ý tưởng ở trên, thuật toán theo dõi sử dụng lọc Kalman có thể được tóm tắt như sau:

1. **Bước 1:** Theo kết quả từ phát hiện đối tượng, chúng ta có thể biết có bao nhiêu đối tượng được phát hiện và sau đó cần phải được theo dõi. Ma trận chuyển tiếp F và ma trận đo H có thể được xác định. Giá trị ban đầu của vectơ trạng thái X thu được tương ứng.
2. **Bước 2:** Áp dụng mô hình dự đoán của Kalman để tìm các vị trí và vận tốc tiếp theo của các vật thể.
3. **Bước 3:** Dựa tốc độ của từng đối tượng trả lại thuật toán phát hiện để giải quyết sự mờ hồ trong phát hiện đối tượng khi xảy ra sự bế tắc.
4. **Bước 4:** Cập nhật vectơ trạng thái với giá trị đo của từng đối tượng. Sau đó, quay lại Bước 2 để bắt đầu lần lặp tiếp theo.

4.5.2.2 Thuật toán Sort

SORT là một thuật toán theo dõi nhiều đối tượng trực quan dựa trên các liên kết dữ liệu thô sơ và kỹ thuật ước tính trạng thái. Nó được thiết kế cho các ứng dụng theo dõi trực tuyến xử lý cho các khung hình lịch sử và hiện tại và phương thức tạo ra danh tính đối tượng một cách nhanh chóng.

Bộ lọc Kalman là một thành phần quan trọng trong SORT. Thuật toán chứa 8 biến $(u, v, a, h, u', v', a', h')$. Trong đó (u, v) là tâm của các khung giới hạn, a là tỷ lệ khung hình và h là chiều cao của hình ảnh. Các biến khác là vận tốc tương ứng của các biến. Các biến chỉ có các yếu tố vị trí và vận tốc tuyệt đối, vì chúng ta đang giả sử một mô hình vận tốc tuyến tính đơn giản. Bộ lọc Kalman giúp chúng ta xác định nhiễu trong quá trình phát hiện đối tượng và sử dụng trạng thái trước để dự đoán mức độ phù hợp cho các hộp giới hạn.

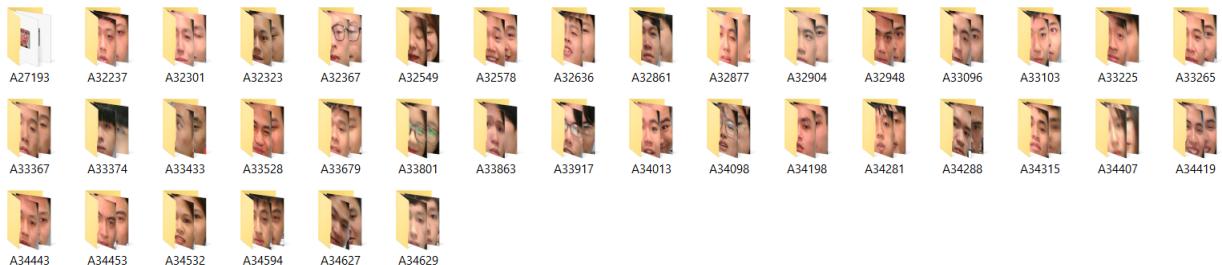
Đối với mỗi lần phát hiện, thuật toán tạo ra một đối tượng người theo dõi (tracker), có tất cả các thông tin trạng thái cần thiết. Nó cũng có một tham số để theo dõi và xóa các đối tượng có lần phát hiện thành công cuối cùng của chúng từ trước đó.

Chương 5

Thực nghiệm

5.1 Tập dữ liệu

Dữ liệu thực nghiệm sử dụng trong khoá luận này được thu thập từ một lớp học của sinh viên Khoa Toán Tin khoá K31. Bao gồm khoảng 1000 ảnh với 38 sinh viên đã được gán nhãn (mã sinh viên).



Hình 5.1: Bộ dữ liệu khuôn mặt của sinh viên

Trung bình mỗi sinh viên có khoảng 30 ảnh được chụp từ nhiều góc độ và điều kiện môi trường khác nhau. Mỗi ảnh có khích thước khoảng 140x190 pixels, định dạng '.jpg'.

5.2 Môi trường huấn luyện

Sử dụng thuật toán OpenFace đã được cài đặt và thực nghiệm huấn luyện mô hình trên môi trường máy tính có cấu hình như sau:

Thông số
OS: Ubuntu 16.04 LTS
HDD: 1Tb
SSD: 256Gb
RAM: 16GB RAM
CPU: Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz
GPU: Geforce RTX 2080 TI

5.3 Phương pháp

Mô hình huấn luyện trong khoá luận này chủ yếu sử dụng phương pháp dựa trên kiến trúc mạng của OpenFace và pretrained model. Để tiếp tục huấn luyện dữ liệu sinh viên khoá luận

này sử dụng phương pháp Transfer learning [15].

Quá trình huấn luyện được triển khai theo các bước chính sau:

- Phân chia tỉ lệ dữ liệu
- Nhận diện và cắt khuôn mặt
- Căn chỉnh khuôn mặt
- Trích rút và phân loại đặc trưng

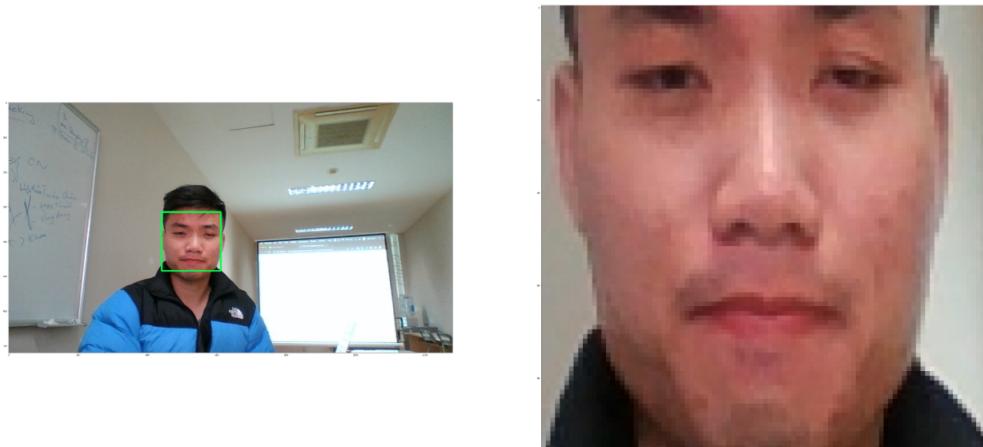
5.3.1 Phân chia tỉ lệ dữ liệu

Chia dữ liệu tất của sinh viên thành ba thư mục lần lượt là train/test/val tương ứng với bộ dữ liệu huấn luyện/kiểm thử/đánh giá theo tỉ lệ 80%/10%/10%.

Mỗi sinh viên có khoảng 30 ảnh. Sinh viên sẽ được chia lấy 20 ảnh cho dữ liệu huấn luyện, 5 ảnh cho dữ liệu kiểm thử và 5 ảnh còn lại cho dữ liệu đánh giá trong quá trình huấn luyện.

5.3.2 Nhận diện và cắt khuôn mặt

Bước này sử dụng thuật toán MTCNN nhằm bóc tách các khuôn mặt sau khi nhận diện được từ bộ dữ liệu gốc. Bộ dữ liệu khuôn mặt được cắt về cùng một kích thước 96×96 .

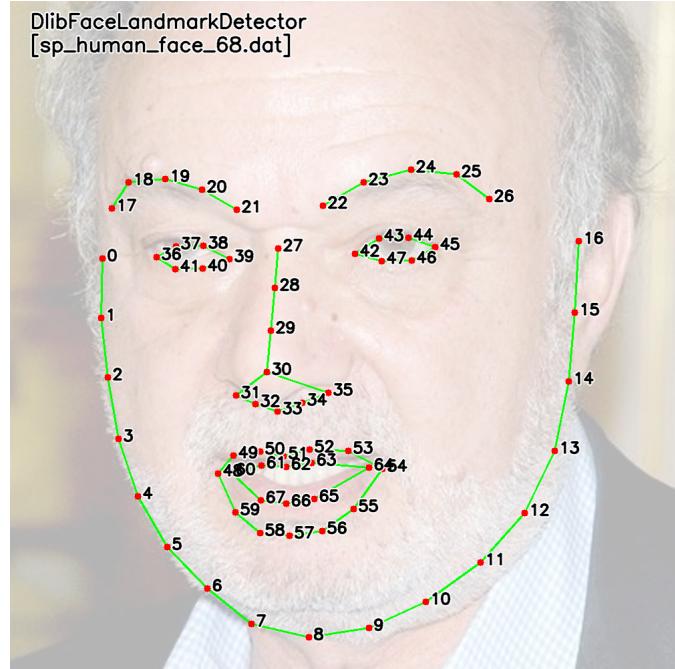


Hình 5.2: Nhận diện và cắt khuôn mặt

5.3.3 Căn chỉnh khuôn mặt

Để lấy được toàn bộ khuôn mặt với tính chất tốt nhất tại bước này cần căn chỉnh những khuôn mặt có hướng nghiêng hoặc chéo về chính diện.

OpenFace cung cấp một hàm AlignDlib để giải quyết vấn đề này. Dựa vào ý tưởng landmark estimation [12]. Tiềm xử lý căn chỉnh khuôn mặt trước khi đưa vào mạng nơ ron. Các khuôn mặt được thay đổi kích thước về cùng kích thước như đã đề cập là 96×96 và được biến đổi để làm cho các mốc (như mắt và mũi) xuất hiện tại cùng một vị trí trên mỗi hình ảnh (tức hướng nhìn thẳng). Dựa theo các landmarks như hình dưới:



Hình 5.3: Landmarks ¹

Khoa luận này sử dụng phương pháp *INNER_EYES_AND_BOTTOM_LIP* theo các toạ độ 39, 42, 57 như hình trên. Đó là căn chỉnh khuôn mặt dựa theo các hốc mắt trong và môi dưới. Theo thực nghiệm thì dựa vào các toạ độ này kết quả phân loại đạt kết quả cao nhất.



Hình 5.4: Khuôn mặt sau khi căn chỉnh

5.3.4 Trích rút và phân loại đặc trưng

Khoa luận sử dụng mô hình nn4.small2.v1 và bộ pre-trained nn4.small2.v1 của OpenFace được mô tả trong hình 5.5.

¹nguồn: <https://openface-api.readthedocs.io/en/latest/openface.html#openface-alignlib-class>

type	output size	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj
conv1 ($7 \times 7 \times 3, 2$)	$48 \times 48 \times 64$						
max pool + norm	$24 \times 24 \times 64$						$m 3 \times 3, 2$
inception (2)	$24 \times 24 \times 192$		64	192			
norm + max pool	$12 \times 12 \times 192$						$m 3 \times 3, 2$
inception (3a)	$12 \times 12 \times 256$	64	96	128	16	32	$m, 32p$
inception (3b)	$12 \times 12 \times 320$	64	96	128	32	64	$\ell_2, 64p$
inception (3c)	$6 \times 6 \times 640$		128	256,2	32	64,2	$m 3 \times 3, 2$
inception (4a)	$6 \times 6 \times 640$	256	96	192	32	64	$\ell_2, 128p$
inception (4e)	$3 \times 3 \times 1024$		160	256,2	64	128,2	$m 3 \times 3, 2$
inception (5a)	$3 \times 3 \times 736$	256	96	384			$\ell_2, 96p$
inception (5b)	$3 \times 3 \times 736$	256	96	384			$m, 96p$
avg pool	736						
linear	128						
ℓ_2 normalization	128						

Hình 5.5: Kiến trúc mô hình nn4.small2.v1 ²

Mô hình này tuy huấn luyện chậm nhưng đạt độ chính xác cao dựa vào điểm chuẩn được đánh giá từ bộ dữ liệu LFW với độ chính xác 92.92 % và chỉ số đo AUC (2.4.6.4) là 0.973 theo bảng 5.6a

Mô hình	Độ chính xác	AUC
nn4.small2.v1	0.9292	0.973
nn4.small1.v1	0.9210	0.973
nn4.v2	0.9157	0.966
nn4.v1	0.7612	0.853

Bảng 5.1: Các mô hình OpenFace

Phương pháp này sử dụng chiến lược fine-turing [15] tầng cuối cùng của mô hình nn4.small2.v1 thành tầng phân loại sử dụng thuật toán *softmax* (2.4.7.3). Kích thước đầu ra của mô hình sau khi fine-turing bằng số lượng nhãn trong tập dữ liệu. Với các tham số huấn luyện hyper-parameter trong bảng 5.2.

Hyparparameter	Chỉ số
batch size	16
epoch	100
learning rate	0.01
optimizer	Adam
loss	categorical crossentropy

Bảng 5.2: Tham số huấn luyện

Mục tiêu của khoá luận hướng tới bài toán xử lý thời gian thực những vẫn đảm bảo độ chính xác cao. Vì vậy, so với các kiến trúc mạng như nn4.v1, nn4.v2, nn4.small1.v1 thì nn4.small2.v1 đạt hiệu năng tốt nhất trên môi trường GPU. Đánh giá theo bài báo và được biểu diễn trong bảng 5.3.

²nguồn: OpenFace: A general-purpose face recognition library with mobile applications

Mô hình	CPU	GPU
nn4.small2.v1	58.9 ms	13.72 ms
nn4.small1.v1	69.58 ms	15.90 ms
nn4.v2	82.74 ms	20.82 ms
nn4.v1	75.67 ms	21.96 ms

Bảng 5.3: Hiệu năng của mô hình

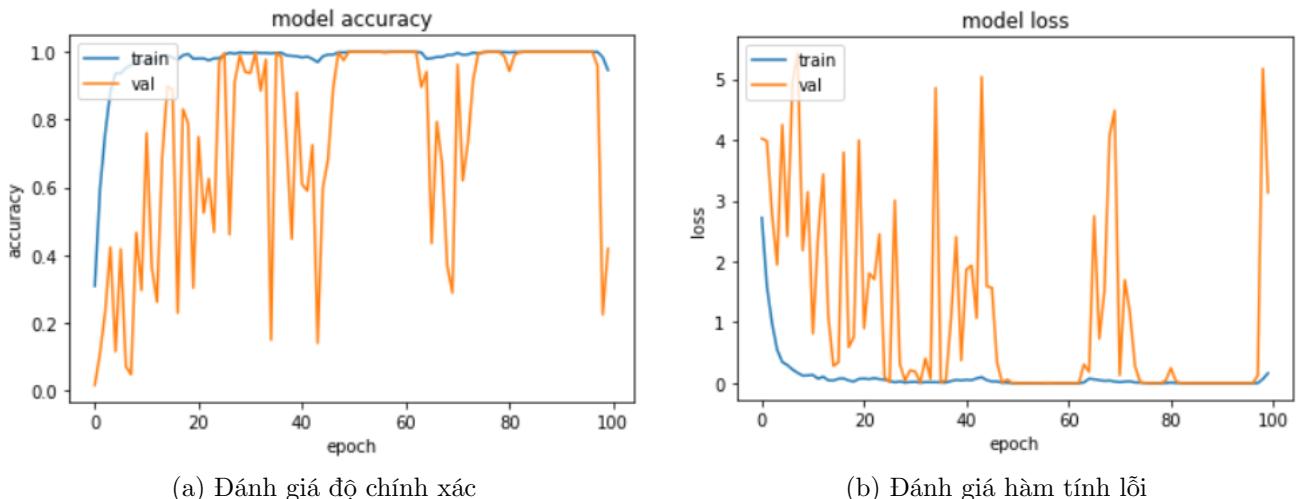
Đồng thời, với mô hình kiến trúc của nn4.small2 đơn giản hơn so với các mô hình khác. Do đó, mô hình này tối ưu được hiệu tăng tính toán của hệ thống được mô tả trong bảng 5.4.

Mô hình	Số lượng
nn4.small2.v1	3733968
nn4.small1.v1	5579520
nn4.v2	6959088
nn4.v1	7472144

Bảng 5.4: Số lượng tham số

5.4 Kết quả thực nghiệm

Sử dụng pre-trained đạt kết quả tốt giúp tối ưu được cơ sở hạ tầng, giảm thời gian tính toán do không cần thiết phải huấn luyện lại những gì là đặc trưng cơ bản nhất của dữ liệu khuôn mặt đã có.



Hình 5.6: Biểu đồ mô tả giá trị mất mát, độ chính xác trong quá trình huấn luyện

Kết quả cho thấy với mô hình này giá trị mất mát (5.6b) được giảm theo thời gian và tiến gần về giá trị 0, độ chính xác (5.6a) tiến xát đến 1. Kết quả cuối cùng sau khi thử nghiệm với tập dữ liệu nhỏ tương ứng với một lớp học khoảng 38 sinh viên thì độ chính xác đạt kết quả là 94,63%. Phương pháp này có thể cải thiện độ chính xác cao hơn nếu tiếp tục thử nghiệm các tham số huấn luyện khác so với bảng 5.2.

Chương 6

Kết luận và hướng phát triển

6.1 Kết luận

Tài liệu này tập trung làm rõ bài lý thuyết thuật toán xác thực điểm danh sinh viên bằng nhận diện khuôn mặt. Đồng thời trình bày cơ sở lý thuyết về toán học và học máy cơ bản.

Nội dung chính của tài liệu là làm rõ phương pháp ứng dụng công nghệ học máy vào xây dựng hệ thống điểm danh, từ đó đưa ra kết quả thực nghiệm.

Đặc biệt phần chính của tài liệu này hướng đến trình bày phương pháp nhận diện và xác thực khuôn mặt bằng thuật toán OpenFace [11]. OpenFace là thuật toán mã nguồn mở, dễ dàng nghiên cứu và phát triển mở rộng. Tuy nhiên, việc ứng dụng thực tế với số lượng hàng nghìn sinh viên thì thuật toán này tuy đưa ra kết quả khá tốt nhưng chưa phải là tốt nhất. Trong quá trình thử nghiệm, nhóm đã thực nghiệm huấn luyện và kiểm tra dựa trên dữ liệu khuôn mặt của một lớp học. Cuối cùng, đạt kết quả độ chính xác 94,63%

Tóm lại, tài liệu về cơ bản đã hoàn thành mục tiêu đề ra trong việc nghiên cứu lý thuyết đến ứng dụng thực tế giải quyết bài toán tự động điểm danh sinh viên bằng công nghệ nhận diện khuôn mặt. Tuy nhiên, để áp dụng và triển khai trong thực tế cần phải được đầu tư nghiêm túc về hạ tầng, thời gian nghiên cứu để xây dựng thành hệ thống thông minh có thể phục vụ đúng yêu cầu.

6.2 Hướng phát triển

Qua quá trình làm tài liệu này, chúng tôi đã nắm vững được mô hình hoạt động cũng như nguyên lý hoạt động của công nghệ học sâu trong bài toán nhận diện. Trong thời gian gần, nhóm tiếp tục học hỏi và thử nghiệm thêm các thuật toán mới. Đặc biệt các phương pháp về phân tích hành vi, nhận diện cảm xúc của sinh viên cũng sẽ được đưa vào nghiên cứu nhằm áp dụng cho bài toán đánh giá chất lượng đào tạo sinh viên, hành vi gian lận trong thi cử. Ngoài ra, nhận diện khuôn mặt có thể áp dụng được trong nhiều lĩnh vực khác. Tuy nhiên, để có thể ứng dụng được trong thực tế chúng tôi sẽ tiếp tục học và đọc thêm các phương pháp tối ưu và công nghệ mạnh mẽ hơn để cải thiện độ chính xác khi xác thực. Đồng thời làm chủ các công nghệ xây dựng thành sản phẩm có thể đóng gói và phát triển kinh doanh.

Tài liệu tham khảo

- [1] F. Schroff, D. Kalenichenko, and J. Philbin, “Facenet: A unified embedding for face recognition and clustering,” *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2015.
- [2] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, “Arcface: Additive angular margin loss for deep face recognition,” 2018.
- [3] V. H. Tiep, “Machine learning cơ bản,” 2016.
- [4] T. M. Mitchell, “Machine learning,” 1997.
- [5] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.
- [6] T. Tieleman and G. Hinton, “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude,” *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.
- [7] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [8] F.-F. Li, A. Karpathy, and J. Johnson, “Cs231n: Convolutional neural networks for visual recognition 2016,” *Stanford University*, 2016.
- [9] B. C. Hoàng, “Nghiên cứu mạng học sâu và ứng dụng cho bài toán nhận dạng biển báo giao thông,” *Dai hoc Thăng Long*, pp. 12–39, 2019.
- [10] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, “Joint face detection and alignment using multitask cascaded convolutional networks,” *IEEE Signal Processing Letters*, vol. 23, pp. 1499–1503, 2016.
- [11] B. Amos, B. Ludwiczuk, and M. Satyanarayanan, “Openface: A general-purpose face recognition library with mobile applications,” 2016.
- [12] D. E. King, “Real-time face pose estimation.”
- [13] S. Kim, “Understanding facial recognition through openface.”
- [14] A. Bewley, Z. Ge, L. Ott, F. T. Ramos, and B. Upcroft, “Simple online and realtime tracking,” *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 3464–3468, 2016.
- [15] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, pp. 1345–1359, 2010.