

**BỘ GIÁO DỤC VÀ ĐÀO TẠO  
TRƯỜNG ĐẠI HỌC ĐÀ LẠT**

**BÁO CÁO TỔNG KẾT  
ĐỀ TÀI KHOA HỌC SINH VIÊN NĂM 2022**

**XÂY DỰNG HỆ THỐNG ĐIỂM DANH SINH VIÊN DỰA TRÊN  
NHẬN DIỆN KHUÔN MẶT**

Thuộc nhóm ngành khoa học: Công nghệ Thông Tin

**Lâm Đồng, tháng 5/2022**

**BỘ GIÁO DỤC VÀ ĐÀO TẠO  
TRƯỜNG ĐẠI HỌC ĐÀ LẠT**

**BÁO CÁO TỔNG KẾT  
ĐỀ TÀI KHOA HỌC SINH VIÊN NĂM 2022**

**XÂY DỰNG HỆ THỐNG ĐIỂM DANH SINH VIÊN DỰA TRÊN  
NHẬN DIỆN KHUÔN MẶT**

Thuộc nhóm ngành khoa học: Công nghệ Thông Tin

Sinh viên thực hiện: Nguyễn Bảo Long

Nam, Nữ: Nam

Dân tộc: Kinh

Lớp, khoa: CTK42, Khoa CNTT

Năm thứ: 4 /Số năm đào tạo: 4.5

Ngành học: Kỹ thuật phần mềm

Giảng viên hướng dẫn: TS. Nguyễn Thị Lương

**Lâm Đồng, tháng 5/2022**

# MỤC LỤC

MỤC LỤC .....	1
DANH MỤC HÌNH ẢNH.....	3
DANH MỤC BẢNG .....	5
DANH MỤC VIẾT TẮT.....	6
THÔNG TIN KẾT QUẢ NGHIÊN CỨU CỦA ĐỀ TÀI.....	7
THÔNG TIN VỀ SINH VIÊN .....	11
TÓM TẮT .....	13
MỞ ĐẦU .....	14
1. Tổng quan tình hình nghiên cứu của đề tài .....	14
2. Lý do chọn đề tài .....	15
3. Mục tiêu đề tài .....	15
3.1. Mục tiêu tổng quát.....	15
3.2. Mục tiêu cụ thể.....	15
4. Phương pháp nghiên cứu .....	16
5. Đối tượng nghiên cứu .....	16
6. Phạm vi nghiên cứu .....	17
CHƯƠNG 1. GIỚI THIỆU TỔNG QUAN.....	18
1.1. Vấn đề bài toán .....	18
1.2. Hướng giải quyết .....	18
1.3. Máy học là gì?.....	19
1.4. Phân nhánh máy học .....	21
1.4.1. Học có giám sát .....	22
1.4.1.1. Hồi quy .....	22
1.4.1.2. Phân loại học có giám sát.....	29
1.4.2. Học không giám sát .....	34
1.4.2.1. Phân cụm.....	34
1.4.2.2. Giảm chiều dữ liệu .....	35
1.4.3. Học bán giám sát .....	38
1.4.3.1. Phân loại học bán giám sát.....	39
1.4.3.2. Phân cụm ràng buộc .....	39
1.4.3.3. Ứng dụng của học bán giám sát .....	39
1.4.4. Học tăng cường.....	40
1.5. Các phương pháp tiếp cận vấn đề trong học máy .....	42
1.5.1. Tiếp cận theo phương pháp Học máy truyền thống (ML).....	42
1.6. Giới thiệu mạng nơron nhân tạo .....	45

1.6.1.	Mô hình toán học mạng nơron.....	45
1.6.2.	Mạng nơron nhân tạo.....	46
1.6.3.	Mạng nơron một lớp.....	46
1.6.4.	Mạng nơron nhiều lớp ẩn .....	47
1.7.	Một số thư viện học máy hiện nay.....	49
1.7.1.	TensorFlow .....	49
1.7.2.	Scikit-learn .....	50
1.7.3.	Keras.....	50
1.7.4.	ONNX.....	51
1.7.5.	PyTorch .....	51
CHƯƠNG 2. PHÁT HIỆN GƯƠNG MẶT .....		52
2.1.	Mô hình Ultra-light fast face detection.....	52
2.2.	Viết chương trình phát hiện gương mặt.....	54
2.3.	So sánh với các mô hình phát hiện gương mặt khác.....	56
CHƯƠNG 3. NHẬN DIỆN GƯƠNG MẶT .....		63
3.1.	Căn chỉnh thẳng hàng gương mặt .....	63
3.2.	Mô hình FaceNet .....	64
3.2.1.	Kiến trúc mô hình.....	64
3.2.2.	Mô hình pretrain .....	65
3.2.3.	Sử dụng mô hình để nhận diện.....	67
3.2.3.1.	Trích xuất đặc trưng để nhận diện gương mặt.....	67
3.2.3.2.	Nhận diện gương mặt .....	68
3.2.3.3.	So sánh với các mô hình nhận diện gương mặt hiện nay.....	71
CHƯƠNG 4. TRIỂN KHAI CHƯƠNG TRÌNH.....		74
4.1	Công nghệ .....	74
4.2	Quy trình của chương trình:.....	74
4.2.1	Thu thập dữ liệu các khuôn mặt để nhận diện.....	74
4.2.2	Xử lý phát hiện và nhận diện khuôn mặt trong thời gian thực.....	78
4.2.3	Xem database điểm danh theo ngày.....	81
CHƯƠNG 5. KẾT LUẬN .....		82
TÀI LIỆU THAM KHẢO .....		84

## DANH MỤC HÌNH ẢNH

Hình 1: Những cột mốc quan trọng của Deep Learning .....	14
Hình 2: Mô tả cách thực hiện dự án .....	18
Hình 3: Mô hình thực hiện chương trình.....	19
Hình 4: Mối quan hệ của các lĩnh vực AI, ML và DL .....	20
Hình 5. Cấu trúc phân nhánh của Machine Learning .....	21
Hình 6. Biểu đồ phân tán hồi quy tuyến tính.....	24
Hình 7. Hyperplane ở đồ thị hai chiều (hình I) và ba chiều (hình II).....	25
Hình 8. Đồ thị hàm số phi tuyến tính (Non Linear Function) .....	27
Hình 9. Biểu đồ hồi quy đa thức (thể hiện tương quan của cá qua từng độ tuổi) .....	28
Hình 10. Biểu đồ thể hiện chỉ số đường huyết bệnh tiểu đường .....	30
Hình 11. Biểu đồ phân loại các ca bệnh tiểu đường.....	31
Hình 12. Ma trận lỗi .....	32
Hình 13. Ma trận lỗi bài toán phân loại nhị phân.....	33
Hình 14.Ví dụ về phân cụm (học không giám sát).....	35
Hình 15. Chu trình học của học tăng cường.....	41
Hình 16. Sơ đồ hoạt động của phương pháp học máy truyền thống .....	42
Hình 17. Biểu đồ sánh hiệu suất giữa Học máy và Học sâu .....	44
Hình 18. Mô hình toán học mạng Nơron.....	45
Hình 19. Nơron nhân tạo mô phỏng nơron sinh học .....	46
Hình 20. Mạng nơron đơn giản (1 lớp ẩn).....	47
Hình 21. Mạng nơron sâu (nhiều lớp ẩn) .....	48
Hình 22. Mạng nơron với lớp ẩn kết nối ngẫu nhiên .....	49
Hình 23. Sơ đồ hoạt động của ONNX.....	51
Hình 24. Mô hình nhận diện vật thể trong một bước .....	52
Hình 25. Sơ đồ hoạt động của mô hình Pytorch SSD .....	53
Hình 26.Sơ đồ chi tiết của mô hình Pytorch SSD .....	53
Hình 27. Sự can thiệp của module RFB vào Pytorch SSD .....	53
Hình 28. Mô phỏng giải thích trường cảm thụ RFB .....	54
Hình 29. Phần code hỗ trợ xử lý hình ảnh.....	55
Hình 30. Bắt đầu quá trình sử dụng ONNX .....	56
Hình 31. Câu lệnh dùng để xác định khuôn mặt .....	56

Hình 32. Lọc các cặp dữ liệu giống nhau .....	57
Hình 33. Kết quả các cặp dữ liệu giống nhau.....	58
Hình 34. Lọc các cặp dữ liệu khác nhau .....	59
Hình 35. Kết quả các cặp dữ liệu khác nhau .....	60
Hình 36. Gộp kết quả các cặp dữ liệu giống và khác nhau .....	60
Hình 37. Sử dụng thư viện DeepFace để gọi các mô hình cho huấn luyện .....	61
Hình 38. Vẽ đồ thị từ kết quả .....	62
Hình 39. Đồ thị của kết quả.....	62
Hình 40. Căn chỉnh gương mặt theo mốc.....	63
Hình 41. Tạo Vector từ gương mặt .....	64
Hình 42. Mô hình Inception resnet v1 .....	65
Hình 43. Sử dụng hàm loss để tăng độ nhận diện .....	66
Hình 44. Thuật toán Triplet loss mô phỏng quá trình nhận diện.....	67
Hình 45. Đưa mô hình đã huấn luyện vào file pickle.....	68
Hình 46. Khoảng cách tuyệt đối giữa hai Vector .....	69
Hình 47. Hiển thị kết quả gương mặt đã phát hiện.....	70
Hình 48. Đồ thị so sánh hiệu quả giữa các mô hình.....	72
Hình 49. Mô hình huấn luyện của chương trình.....	75
Hình 50. Giao diện huấn luyện của chương trình.....	76
Hình 51. Dữ liệu kết quả thu được sau khi xử lý .....	77
Hình 52. Giao diện nhận diện khuôn mặt của chương trình .....	78
Hình 53. Mô hình nhận diện gương mặt của chương trình .....	79
Hình 54. kết quả nhận diện theo thời gian thực.....	79
Hình 55. Giao diện tùy chỉnh biến số nhận diện .....	80
Hình 56. Giao diện của truy xuất dữ liệu điểm danh sinh viên.....	81

## DANH MỤC BẢNG

Bảng 1. Mục tiêu cụ thể (nguyên tắc SMART).....	16
Bảng 2. Dữ liệu về thông số và giá nhà.....	24
Bảng 3. Dữ liệu về chỉ số đường huyết của các bệnh nhân.....	29
Bảng 4. Bảng đối chiếu $y$ và $y^{\wedge}$ của mô hình luận lý.....	32
Bảng 5. Dữ liệu về thông tin chiều cao và cân nặng .....	36
Bảng 6. Dữ liệu thông tin sức khỏe và phân loại mức độ béo phì .....	37
Bảng 7. So sánh giữa 2 phiên bản với bộ dữ liệu WiderFace .....	54
Bảng 8. Đánh giá độ chính xác giữa hai mô hình như dưới.....	65
Bảng 9. So sánh hiệu quả giữa các mô hình.....	73

## DANH MỤC VIẾT TẮT

STT	Chữ viết tắt	Viết đầy đủ
1	AI	Artificial Intelligence
2	A, P, R	Accuracy, Precision, Recall
3	ANN	Artificial Neural Networks
4	API	Application Programming Interface
5	APK	Android Package Kit
6	CNN	Convolution Neural Network
7	CNTT	Công Nghệ Thông Tin
8	CPU	Central Processing Unit
9	DL	Deep Learning
10	DNN	Deep Neural Network
11	ĐLC	Độ Lệch Chuẩn
12	GPU	Graphics Processing Unit
13	ML	Machine Learning
14	NN	Neural Network
15	RAM	Random Access Memory
16	ReLU	Rectified Linear Unit
17	Sklearn	Scikit-learn
18	TF	TensorFlow
19	TB	Trung Bình



**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐÀ LẠT**

**THÔNG TIN KẾT QUẢ NGHIÊN CỨU CỦA ĐỀ TÀI**

**1. Thông tin chung:**

- Tên đề tài: “Xây dựng hệ thống điểm danh sinh viên dựa trên nhận diện khuôn mặt”
- Sinh viên thực hiện:

STT	Họ và tên	MSSV	Lớp	Khoa	Năm thứ
1	Nguyễn Danh	1812736	CTK42-PM	CNTT	4
2	Nguyễn Bảo Long	1812795	CTK42-PM	CNTT	4
3	Nguyễn Trọng Hiếu	1812756	CTK42-PM	CNTT	4
4	Lê Hoàng Nhật	1812814	CTK42-PM	CNTT	4
5	Trần Hữu Khải Quân	1813857	CTK42-PM	CNTT	4

- Giảng viên hướng dẫn: TS. Nguyễn Thị Lương

**2. Mục tiêu đề tài**

**2.1. Về lý thuyết**

Tìm hiểu được lịch sử hình thành và phát triển cùng với các kiến thức cơ bản của Học máy và Học sâu, tìm hiểu cơ sở toán học bên dưới của các mô hình. Tìm hiểu các thư viện học máy nổi tiếng hiện nay, cách sử dụng thư viện để huấn luyện mô hình và triển khai xây dựng ứng dụng nhận diện trên nền tảng Web. Nắm rõ kiến thức về kiến trúc CNN, tìm hiểu các mô hình học máy nổi tiếng những năm gần đây.

**2.2. Về thực nghiệm**

Đề tài hướng đến việc xây dựng mô hình huấn luyện và triển khai mô hình trên nền tảng Web , cho phép nhận diện khuôn mặt từ các ảnh thư viện, từ ảnh chụp và nhận diện *thời gian thực (realtime)*, dự đoán tên của sinh viên với tốc độ nhận diện và độ chính xác cao và đưa ra thông tin của sinh viên đó.

### **3. Giới thiệu đề tài**

Hiện nay thị giác máy tính (computer vision) được áp dụng rộng rãi vào trong đời sống của con người. Một trong những ứng dụng phổ biến nhất của thị giác máy tính đó chính là nhận diện gương mặt. Mặc dù hiện nay đã có rất nhiều ứng dụng và thiết bị nhận diện gương mặt xuất hiện trên thị trường, chúng vẫn còn bị hạn chế trong phạm vi sử dụng cũng như khả năng xử lý và độ hiệu quả chưa đáp ứng được với số tiền đầu tư. Trong bài đồ án này sẽ ta cố gắng nghiên cứu và thiết kế xây dựng ra một hệ thống nhận diện gương mặt có thể linh hoạt đáp ứng được với mọi nhu cầu của người sử dụng và áp dụng vào được trong nhiều ngành nghề lĩnh vực, nền tảng sử dụng khác nhau.

### **4. Kết quả nghiên cứu**

Xây dựng thành công mô hình – Đề tài đã tiến hành xây dựng, huấn luyện thành công mô hình Máy học, so sánh các mô hình đã huấn luyện và chọn ra mô hình tối ưu nhất.

Triển khai ứng dụng Web – Sau khi huấn luyện và kết xuất mô hình, sử dụng mô hình đã kết xuất để xây dựng thành ứng dụng chạy trên nền tảng Web. Xây dựng giao diện và các chức năng cho ứng dụng. Nhóm chúng tôi đã thử nghiệm ứng dụng điểm danh sinh viên và nhận về được kết quả tốt, với thời gian nhận diện thấp và độ chính xác cao.

### **5. Đóng góp về mặt kinh tế - xã hội, giáo dục và đào tạo, an ninh, quốc phòng và khả năng áp dụng của đề tài:**

#### **5.1. Hiệu quả xã hội:**

Hệ thống nhận diện gương mặt cung cấp cho giảng viên một cách thức đơn giản và nhanh chóng trong việc điểm danh sinh viên có mặt trong lớp học. Điều này góp phần cải thiện môi trường học tập, giảng dạy của sinh viên và giảng viên.

#### **5.2. Hiệu quả an ninh:**

Có thể sử dụng kết quả nghiên cứu của đề tài, thay đổi tập dữ liệu đầu vào là khuôn mặt và thông tin của người dân. Khi áp dụng nhận diện thời gian thực của đề

tài, ta có thể tiến hành nhận diện đối tượng người, thực hiện các quản lý về an ninh của doanh nghiệp và tổ chức.

### **5.3. Khả năng áp dụng của đề tài:**

Kết quả nghiên cứu của đề tài có thể áp dụng rộng rãi trong mọi mặt của đời sống hay ứng dụng trong các ngành khác. Ví dụ như trong công nghệ sinh học, có thể sử dụng công nghệ nhận dạng loài hoa thời gian thực để tự động theo dõi tình trạng sức khỏe của cây hoa, dự đoán thời gian thu hoạch, dự đoán các loại sâu bệnh,...

## **6. Công bố khoa học của sinh viên từ kết quả nghiên cứu của đề tài**

Ngày            tháng            năm

**Sinh viên chịu trách nhiệm chính  
thực hiện đề tài**

*(ký, họ và tên)*

**Nhận xét của người hướng dẫn về những đóng góp khoa học của sinh viên thực hiện đề tài** *(phần này do người hướng dẫn ghi)*:

Đề tài đã xây dựng được một mô hình học máy cho bài toán nhận diện khuôn mặt và mô hình này được áp dụng trong việc xây dựng ứng dụng nhận diện trên nền tảng Web giúp Giảng viên dễ dàng sử dụng để điểm danh sinh viên.

**Xác nhận của trường đại học**

*(ký tên và đóng dấu)*

Ngày      tháng      năm

**Người hướng dẫn**

*(ký, họ và tên)*

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC ĐÀ LẠT

THÔNG TIN VỀ SINH VIÊN

CHỊU TRÁCH NHIỆM CHÍNH THỰC HIỆN ĐỀ TÀI

## I. SƠ LƯỢC VỀ SINH VIÊN

Họ và tên: Nguyễn Bảo Long

Sinh ngày: 14 tháng 12 năm 2000

Nơi sinh: Đà Lạt

Lớp: CTK42-PM

Khóa: 42

Khoa: Công nghệ Thông tin

Địa chỉ liên hệ:      Lớp CTK42-PM

Khoa Công nghệ Thông tin, Trường Đại học Đà Lạt

Điện thoại: 0914551380

Email: [1812795@dlu.edu.vn](mailto:1812795@dlu.edu.vn)

## II. QUÁ TRÌNH HỌC TẬP

### ● *Năm thứ 1:*

Ngành học:

Khoa: Công nghệ Thông tin

Kết quả xếp loại học tập: ĐTBTL: 2.7/4.0

### ● *Năm thứ 2:*

Ngành học:

Khoa: Công nghệ Thông tin

Kết quả xếp loại học tập: ĐTBTL: 2.7/4.0

### ● *Năm thứ 3:*

Ngành học:

Khoa: Công nghệ Thông tin

Kết quả xếp loại học tập: ĐTBTL: 2.95/4.0

Sơ lược thành tích: Học bổng KKHT học kỳ I

● ***Năm thứ 4 – Học kỳ I:***

Ngành học: Kỹ thuật phần mềm      Khoa: Công nghệ Thông tin

Kết quả xếp loại học tập: ĐTBTL: 3.03/4.0

Ngày      tháng      năm

**Xác nhận của trường đại học**

*(ký tên và đóng dấu)*

**Sinh viên chịu trách nhiệm chính**

**thực hiện đề tài**

*(ký, họ và tên)*

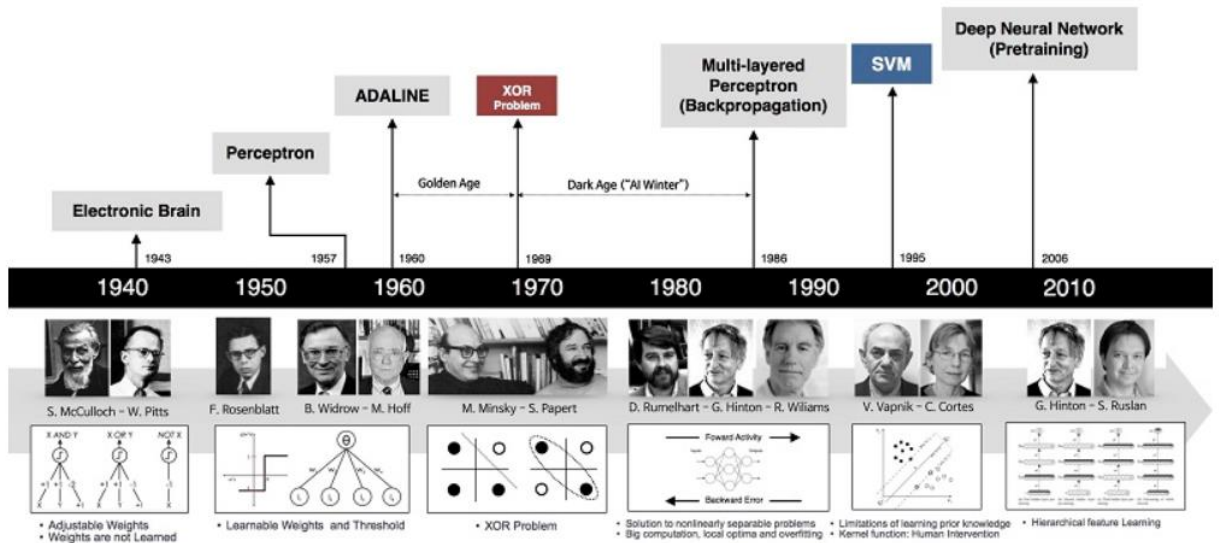
## TÓM TẮT

Trong đề tài nghiên cứu này. Nhóm chúng tôi thực hiện nghiên cứu việc xác định thông tin của sinh viên dựa trên ảnh chụp một cách tự động hóa bằng công nghệ phần mềm và khoa học máy tính. Nghiên cứu áp dụng công nghệ *Học máy (Machine learning)* và *Học sâu (Deep learning)* vào việc huấn luyện mô hình để nhận diện gương mặt người. Thực nghiệm mô hình, tiến hành đánh giá mô hình thông qua các thông số đặc trưng và triển khai thành ứng dụng trên nền tảng Web. Việc huấn luyện và thực nghiệm mô hình máy học sẽ sử dụng thư viện PyTorch, được phát triển bởi *Viện Nghiên cứu AI của Facebook (Facebook's AI Research Lab)*.

Từ khóa: face recognition, identify student by faces, neural network, computer vision, nhận diện gương mặt, nhận diện sinh viên, tự động hóa, thị giác máy tính.

## MỞ ĐẦU

### 1. Tổng quan tình hình nghiên cứu của đề tài



Hình 1: Những cột mốc quan trọng của Deep Learning

Hình 1 thể hiện những dấu mốc quan trọng của Deep Learning. Vào đầu những năm 1940, với sự xuất hiện và phát triển mạnh mẽ của thiết bị bán dẫn, linh kiện điện tử và máy tính đã đặt những nền móng đầu tiên cho sự xuất hiện của *trí tuệ nhân tạo (AI)*. Tuy nhiên trí tuệ nhân tạo thời gian này vẫn chưa thật sự có ứng dụng thực tiễn hoặc thành tựu nổi bật nào. Khoảng thời gian từ năm 1960 đến 2000, giới chuyên gia và các nhà nghiên cứu đã phải trải qua hai *mùa đông AI (AI Winter)*, do sự bế tắc và các gián đoạn nghiên cứu xảy ra trong khoảng thời gian này.

Sự đột phá bắt đầu từ năm 2006, khi mà Hinton [1] giới thiệu ý tưởng về *tiền huấn luyện không giám sát (unsupervised pre-training)* thông qua *deep belief nets (DBN)*. Điểm nổi bật trong bài báo này là đã tạo ra được một nơron nhân tạo với nhiều lớp ẩn (*hidden layer*) thay vì chỉ một lớp như trước đây. Từ thời gian này, neural networks với nhiều lớp ẩn được gọi với cái tên là **Deep Learning**.

Sự phát triển không ngừng của lĩnh vực trí tuệ nhân tạo, máy học và học sâu cũng chỉ để nhằm vào mục đích duy nhất, đó là phục vụ cho nhu cầu lợi ích và cuộc



sống của con người. Như giúp thiết bị điện tử nhận diện trắc sinh học (vân tay, quét võng mạc, nhận diện khuôn mặt, ...), giúp dự đoán thời tiết, chẩn đoán các loại bệnh hay dịch các ngôn ngữ khác nhau. Để phục vụ những công việc phức tạp, muôn hình vạn trạng của con người, trí tuệ nhân tạo chia ra làm các lĩnh vực để chuyên biệt hóa như Hệ chuyên gia, Cây ngữ nghĩa, Xử lý ngôn ngữ tự nhiên, Robotics, Quy hoạch, Thị giác máy tính,...

Tuy nhiên, việc nhận diện gương mặt là một vấn đề không hề đơn giản. Gương mặt mỗi người đều có những đặc trưng riêng biệt, điều này đặt ra các thách thức không nhỏ cho việc huấn luyện mô hình nhận diện. Do đó, công việc này phải thực hiện bởi phương pháp *học sâu (deep learning)*, với dữ liệu đầu vào là ảnh gương mặt, được đưa vào mạng CNN đã được huấn luyện lại bằng các mô hình ResNet, DenseNet và các biến thể, cuối cùng là so sánh các mô hình với nhau và chọn ra mô hình huấn luyện có độ chính xác cao nhất.

## **2. Lý do chọn đề tài**

Hiện nay việc nhận diện gương mặt đã trở nên cần thiết đối với nhiều ngành nghề, lĩnh vực khác nhau. Việc xây dựng một hệ thống gương mặt có thể đáp ứng được mọi nhu cầu sử dụng vẫn đang là một vấn đề nan giải mặt dù đã tồn tại rất nhiều mô hình nhận diện gương mặt. Để có thể giải bài toán này, ta cần phải thiết kế hệ thống có khả năng linh hoạt cao cũng như phạm vi ứng dụng lớn để người dùng có thể truy cập và sử dụng ở mọi lúc, mọi nơi.

## **3. Mục tiêu đề tài**

### **3.1. Mục tiêu tổng quát**

Xây dựng một ứng dụng trên nền tảng Web bằng tiếng Việt, cho phép nhận diện sinh viên từ các ảnh thư viện, từ ảnh chụp và nhận diện *thời gian thực (realtime)*, sử dụng thư viện PyTorch để huấn luyện, dự đoán tên của sinh viên và kết xuất ra tập tin tổng hợp để giảng viên theo dõi.

### **3.2. Mục tiêu cụ thể**

Mục tiêu cụ thể của đề tài được trình bày trong Bảng 1, theo theo nguyên tắc SMART [6]:

*Bảng 1. Mục tiêu cụ thể (nguyên tắc SMART)*

<b>Tính cụ thể (Specific)</b>	Tìm hiểu một số phương pháp Học máy điển hình, áp dụng thuật toán trên tập dữ liệu đã thu thập, đánh giá kết quả của hai thuật toán để từ đó chọn lựa được một mô hình phù hợp với độ chính xác cao để triển khai lên website để người dùng tương tác.
<b>Tính đo lường (Measurable)</b>	Thu thập hình ảnh và thông tin của sinh viên, với mỗi sinh viên thì thu thập khoảng từ 50 – 100 bức ảnh để đưa vào quá trình huấn luyện, sau đó thực hiện tiền xử lý hình ảnh để kết quả dự đoán cao hơn 85%.
<b>Tính khả thi (Achievable)</b>	Ứng dụng được xây dựng và có thể triển khai tại khoa Công nghệ thông tin trường Đại học Đà Lạt để thử nghiệm điểm danh tự động cho các sinh viên trong khoa.
<b>Tính thực tế (Realistic)</b>	Phạm vi nghiên cứu của đề tài này là phù hợp với trình độ của người thực hiện cũng như là kết quả mà đề tài mang lại là phù hợp với tình hình thực tế hiện nay.
<b>Tính thời hạn (Timely)</b>	Đề tài sẽ hoàn thành theo đúng tiến độ của nghiên cứu khoa học mà trường Đại học Đà Lạt đã công bố.

#### 4. Phương pháp nghiên cứu

Đề tài này sử dụng một vài phương pháp nghiên cứu nhưng chủ yếu là phương pháp phân tích và tổng kết kinh nghiệm. Cụ thể là từ những công trình nghiên cứu liên quan đến đề tài và sự hỗ trợ từ thư viện Máy học PyTorch để đề xuất một cách tiếp cận mới trong giải quyết vấn đề đặt ra.

#### 5. Đối tượng nghiên cứu

Đối tượng nghiên cứu của đề tài là phương pháp và ứng dụng nhận diện sinh viên từ gương mặt. Trước mắt, chúng tôi sẽ thực hiện nhận diện những sinh viên thuộc khoa Công nghệ thông tin trường Đại học Đà Lạt.

## **6. Phạm vi nghiên cứu**

Đối tượng khảo sát trong đề tài này chỉ giới hạn trong các sinh viên thuộc khoa Công nghệ thông tin trường Đại học Đà Lạt. Với khoảng 100 sinh viên, với mỗi sinh viên sẽ thu thập ít nhất 50 hình, tỷ lệ ảnh chụp là 1:1. Thời gian thực hiện đề tài nằm trong tiến độ nghiên cứu khoa học cấp trường. Nội dung nghiên cứu về lý thuyết sẽ tập trung giới thiệu Học máy và các phương pháp Học máy ở mức tổng quan, sau đó chọn ra các mô hình phù hợp trong thư viện PyTorch để triển khai hệ thống nhận diện gương mặt.

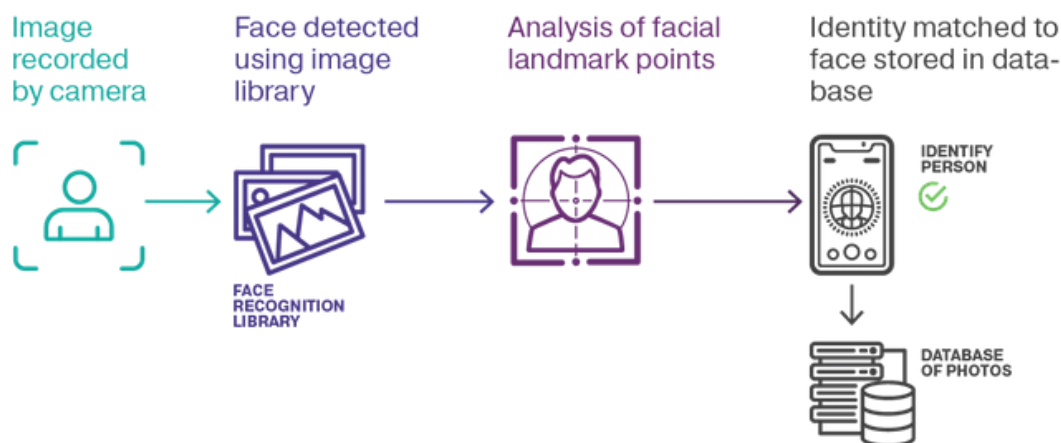
## CHƯƠNG 1. GIỚI THIỆU TỔNG QUAN

### 1.1. Vấn đề bài toán

Hiện nay việc nhận diện gương mặt đã trở nên cần thiết đối với nhiều ngành nghề, lĩnh vực khác nhau. Việc xây dựng một hệ thống gương mặt có thể đáp ứng được mọi nhu cầu sử dụng vẫn đang là một vấn đề nan giải mặt dù đã tồn tại rất nhiều mô hình nhận diện gương mặt. Để có thể giải bài toán này, ta cần phải thiết kế hệ thống có khả năng linh hoạt cao cũng như phạm vi ứng dụng lớn để người dùng có thể truy cập và sử dụng ở mọi lúc, mọi nơi.

### 1.2. Hướng giải quyết

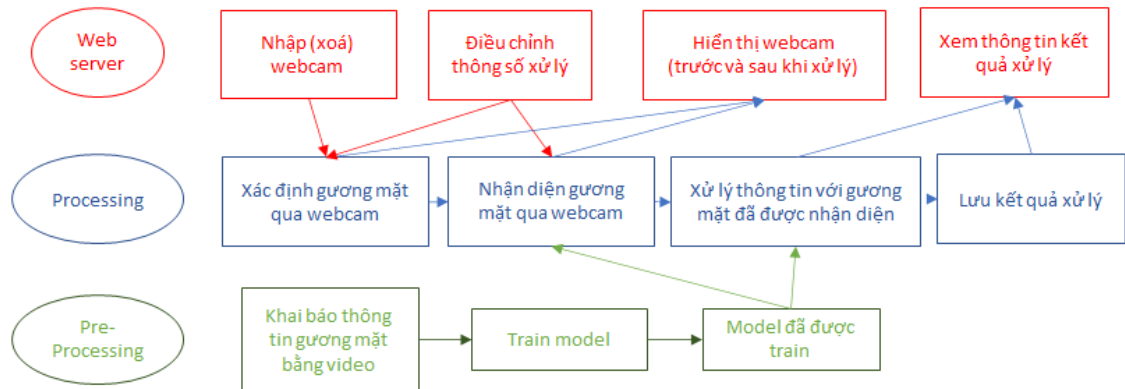
Hệ thống nhận diện gương mặt được chia thành hai giai đoạn chính là phát hiện gương mặt (Face Detection) và nhận diện gương mặt (Face Verifiaction). Mỗi giai đoạn hiện nay đều có nhiều thuật toán và các mô hình khác nhau. Để lựa chọn được mô hình phù hợp ta sẽ so sánh dựa trên 3 yếu tố sau: tốc độ xử lý, độ chính xác và độ lớn của mô hình. Sau khi lựa chọn được mô hình phù hợp và hiệu quả nhất, ta sẽ xây dựng một hệ thống nhận diện gương mặt theo các mô hình được chọn. Quá trình xử lý của hệ thống có thể được mô tả theo như hình ở phía dưới:



Hình 2: Mô tả cách thực hiện dự án

Thông qua hệ thống này, ta có thể tạo ra chương trình bất kỳ có sử dụng chức năng nhận diện gương mặt. Trong bài báo cáo sẽ tạo một chương trình chạy trên nền tảng web vì có độ linh hoạt và khả năng tiếp cận cao hơn so với các nền tảng

khác. Chương trình sau đó sẽ được triển khai trên nền tảng Google Cloud để có thể truy cập vào ở bất kỳ chỗ nào có kết nối với internet. Sơ đồ phía dưới mô tả hệ thống kiến trúc của ứng dụng sử dụng hệ thống nhận diện gương mặt:

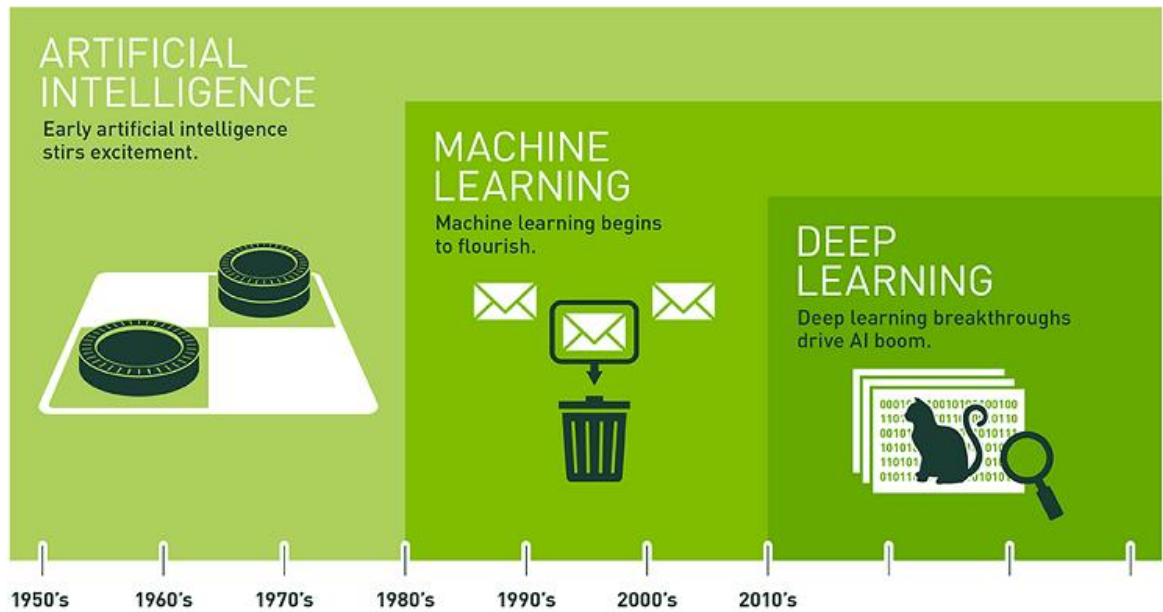


Hình 3: Mô hình thực hiện chương trình

### 1.3. Máy học là gì?

Ngày nay, *Học máy (Machine Learning)* đang là một làn sóng mới, được ứng dụng rộng rãi vào đời sống của con người và mọi mặt của xã hội [7], như trong nhận diện khuôn mặt, xe hơi tự lái, đề xuất mua hàng trên Lazada, Tiki, Amazon, và hàng loạt các ứng dụng Trí tuệ nhân tạo tiên tiến khác. Machine learning tập trung vào việc tạo ra các hệ thống, cỗ máy có khả năng tự mình học hỏi mà không cần phải được lập trình một cách cụ thể. Đây là việc tạo ra các cỗ máy có khả năng tự học hỏi dựa trên các kinh nghiệm mà chúng thu thập được trong suốt quá trình hoạt động, điều này mô phỏng gần giống với hoạt động trí tuệ của con người.

Về định nghĩa khái niệm, máy học là một lĩnh vực thuộc Khoa học máy tính, đồng thời là một lĩnh vực con của *Trí tuệ nhân tạo (AI)*. Mối quan hệ giữa *Trí tuệ nhân tạo*, *Học máy* và *Học sâu* được thể hiện trong Hình 4:



Hình 4: Mối quan hệ của các lĩnh vực AI, ML và DL

Arthur Samuel đã định nghĩa vào năm 1959, “ML là một lĩnh vực nghiên cứu cung cấp cho máy vi tính khả năng học mà không cần phải lập trình rõ ràng” [8]. Một định nghĩa thể hiện rõ ràng hơn vào năm 1997: “Một chương trình máy tính được cho là học hỏi từ kinh nghiệm E có liên quan với một vài nhiệm vụ T và hiệu suất đo lường P, nếu hiệu suất của nó trên T được đo bằng P cải thiện sau khi trải qua kinh nghiệm E” [9].

Ví dụ: AlphaGo là một chương trình máy tính do Google DeepMind phát triển đã học được cách chơi cờ vây và đánh bại cờ thủ nổi tiếng Lee Se-dol [10]. Theo định nghĩa của Tom Mitchell, ta có:

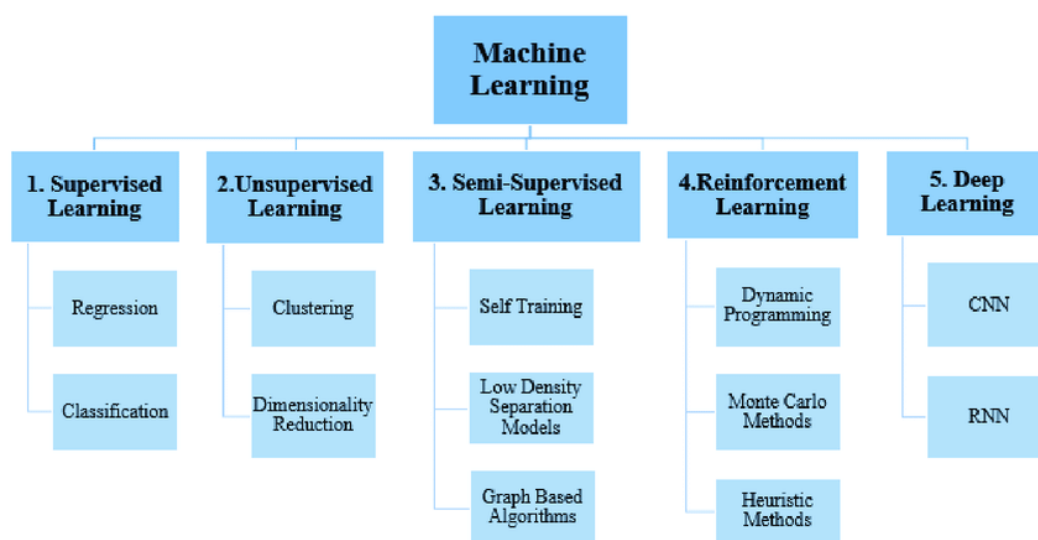
- E: Kinh nghiệm có được khi chơi với các đối thủ và tự tăng cường.
- T: Nhiệm vụ là chơi cờ vây.
- P: Khả năng mà AlphaGo sẽ thắng trong trận đấu tiếp theo.

Nhìn chung, để xác định được một vấn đề học tập rõ ràng cần phải nhận biết được ba đặc trưng sau: các nhiệm vụ, hiệu suất cần được cải thiện và nguồn gốc kinh nghiệm.

## 1.4. Phân nhánh máy học

*Máy học (Machine learning)* là một thuật ngữ với khái niệm rộng, trong đó, nó chia ra các nhánh nhỏ, mỗi nhánh như vậy là một lĩnh vực khác nhau. Việc huấn luyện mô hình để đáp ứng yêu cầu đưa ra là rất phức tạp. Mỗi công việc khác nhau sẽ có từng cách giải quyết tương ứng. Để thỏa mãn những thách thức này, ML sinh ra nhiều phương pháp học máy khác nhau. Hình 5 là sơ đồ cấu trúc của ML và các phân nhánh của nó:

- **Học có giám sát (Supervised learning):** Một tập dữ liệu chính xác bao gồm các cặp (data, label), đưa vào một thuật toán để huấn luyện, sau quá trình huấn luyện, nếu đưa một đầu vào mới chưa có trong tập dữ liệu, thuật toán sẽ dự đoán đầu ra chính xác nhất có thể. Quá trình này gọi là học từ ví dụ hay học có giám sát.
- **Học không giám sát (Unsupervised learning):** Khác với phương pháp trên, dữ liệu trong tập dữ liệu không có đầu ra, thay vì thuật toán gán nhãn thì thuật toán sẽ phải *phân cụm (cluster)* chúng. Cách tiếp cận theo thống kê của phương pháp này là *ước tính mật độ (density estimation)*.



Hình 5. Cấu trúc phân nhánh của Machine Learning

- **Học bán giám sát (Semi-supervised learning):** Học bán giám sát là sự kết hợp giữa hai phương pháp kể trên, chính vì vậy mà trong tập dữ liệu chia thành hai phần: một phần nhỏ dữ liệu đã được gán nhãn và phần còn lại dữ

liệu chưa được gán nhãn. Đa số các bài toán thuộc về loại này vì chi phí để gán nhãn dữ liệu là khá lớn so với dữ liệu có được từ Internet.

- **Học tăng cường (Reinforcement learning):** Phương pháp này cũng là sự kết hợp giữa học có giám sát và học không giám sát. Tuy nhiên, điểm khác biệt so với bán giám sát là thuật toán chỉ cần được biết nhiệm vụ này có làm sai hay không mà không cần biết cách để làm đúng nhiệm vụ. Thuật toán sẽ thám hiểm và thử các khả năng có thể (giống như thuật toán vét cạn) cho đến khi nó được biết là nó đang làm đúng.
- **Học sâu (Deep Learning):** Là một lĩnh vực con của ML. Về bản chất, DL cố gắng bắt chước cách thức hoạt động của bộ não con người. Mọi người, nhất là giới khác chuyên môn, thường hay nhầm lẫn về hai thuật ngữ này. Trong khi machine learning là việc dạy cho máy tính biết cách giải quyết một nhiệm vụ nào đó mà không cần lập trình chi tiết, ML là một phương pháp học nông, chỉ giải quyết được các vấn đề hay bài toán đơn giản. Thì DL lại giúp chúng ta giải quyết được các vấn đề phức tạp trong thế giới thật, những bài toán mà đôi khi con người cũng có thể bị bối rối. Học sâu sử dụng mạng *neuron nhân tạo (Artificial Neural Networks)*, để giải quyết những bài toán với dữ liệu trừu tượng, dữ liệu không rõ ràng, dữ liệu mờ,...

#### **1.4.1. Học có giám sát**

##### **1.4.1.1. Hồi quy**

###### **a. Hồi quy tuyến tính**

Hồi quy là một hình thức học trong Học máy với mục đích là tạo ra một mô hình dự đoán đầu ra là các giá trị liên tục, chẳng hạn giá cả, số lượng, khối lượng hoặc các giá trị vô hướng khác.

Hồi quy cho thấy được mối quan hệ giữa các biến trong dữ liệu (đặc trưng) mà chúng ta cần quan sát và biến mà chúng ta cần dự đoán (nhãn). Trong mô hình này, một tập dữ liệu cần phải được gán nhãn. Tập dữ liệu sau đó được chia thành hai phần:

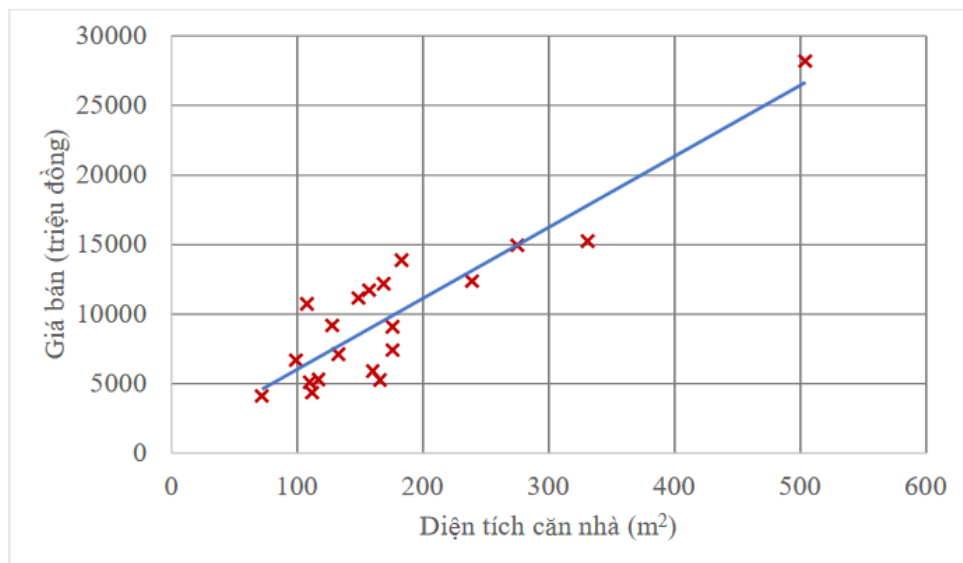


- **Tập dữ liệu huấn luyện (training dataset):** Dữ liệu sử dụng trong mô hình nhằm xác định một hàm số biểu thị mối quan hệ giữa các biến đặc trưng và các nhãn.
- **Tập dữ liệu thẩm định (validation dataset)/ kiểm tra (test dataset):** Dữ liệu dùng để đánh giá hàm số đã tìm được trong mô hình sau khi sử dụng tập dữ liệu huấn luyện ở bước trước đó bằng cách so sánh giá trị dự đoán và giá trị thực tế.

Ví dụ: Người ta thu thập các số liệu của hơn 20.000 căn nhà ở thành phố Portland, tiểu bang Oregon, Mỹ được các thông tin như sau: diện tích ngôi nhà ( $x_1, m^2$ ), diện tích đất ( $x_2, m^2$ ), số phòng ngủ ( $x_3$ ), số tầng ( $x_4$ ), số nhà vệ sinh/nhà tắm ( $x_5$ ), tuổi căn nhà ( $x_6$ ) và cuối cùng là giá bán ( $y$ , triệu đồng). Lưu ý là các số liệu đã được chuyển đổi feet<sup>2</sup> sang m<sup>2</sup>, Đô-la Mỹ sang VNĐ. Liệu rằng chúng ta có thể đoán biết giá của một căn nhà nếu dựa vào các số liệu còn lại được hay không?

Bảng 2. Dữ liệu về thông số và giá nhà

Diện tích căn nhà (m2)	Diện tích đất(m2)	Số phòng ngủ	Số tầng	Số nhà vệ sinh/nhà tắm	Tuổi căn nhà	Giá bán (triệu đồng)
110	525	3	1	1	5	5103
239	673	3	2	3	5	12374
72	929	2	1	1	5	4140
183	465	4	1	3	5	13892
157	751	3	1	2	5	11730
...	...	...	...	...	...	...



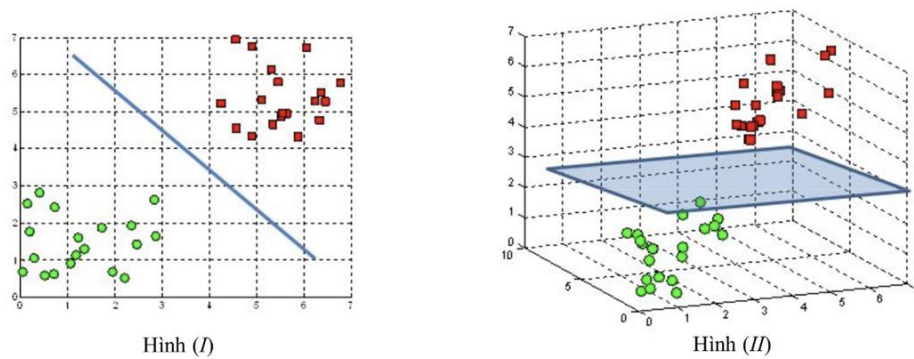
Hình 6. Biểu đồ phân tán hồi quy tuyến tính

Tập dữ liệu thu thập được ký hiệu là  $\{(x^{(i)}, y^{(i)}), \forall i = 1, 2, \dots, m\}$  Trong đó,  $x(i)$  là véc tơ tương ứng với dữ liệu đầu vào của căn nhà thứ  $i, x \in R^n$  ( $x$  là biến phụ thuộc),  $y(i)$  là giá bán của căn nhà thứ  $i$  ( $y$  là biến độc lập),  $m$  là số lượng căn nhà thu thập được. Với mỗi đặc trưng  $j$  của  $x(i)$  ta ký hiệu là  $x_j^{(i)}, \forall j = 1, 2, \dots, n$ . Giá trị

$y$  là một số thực  $R$ ,  $x^{(i)} = \{x_1^{(i)}, x_2^{(i)}, \dots, x_n^{(m)}\}$ . Nói cách khác, mục đích của mô hình này là tìm ra một hàm số  $f$  nào đó sao cho  $y = f(x)$ .

Trong Hình 6, với mỗi dấu  $x$  đỏ trên biểu đồ tương ứng với một cặp giá trị  $(x_1^{(i)}, y^{(i)})$  của 20 căn nhà đầu tiên và một đường thẳng đi ngang qua với các điểm giá trị bao quanh nó. Đường thẳng chính là mối quan hệ tuyến tính của  $x_1^{(i)}$  và  $y^{(i)}$ , chính là đồ thị của hàm số  $y^{(i)} = f(x_1^{(i)})$ , mô hình này gọi là hồi quy tuyến tính. Tuy nhiên, không phải lúc nào mà đường tuyến tính cũng có thể vừa khớp với tất cả dữ liệu được, những điểm nằm ngoài đường thẳng sẽ tạo ra một sai số hoặc phương sai. Một thuật toán trong Hồi quy sẽ tìm hàm số sao cho đường tuyến tính có phương sai nhỏ nhất có thể. Lúc này, gọi giá trị dự đoán là  $\hat{y}$ , giá trị thực tế  $y \approx f(x) = \hat{y}$ .

Hồi quy đơn tuyến tính được biểu diễn dưới dạng:  $f(x_1) = \theta_0 + \theta_1 x_1$ . Với nhiều hơn một đặc trưng, thì mối quan hệ giữa các biến đặc trưng và giá nhà là  $f(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4 + \theta_5 x_5 + \theta_6 x_6$ . Đây được gọi là hồi quy đa tuyến tính. Trong đó,  $\theta_j, \forall j = 1, 2, \dots, 6$  là các hằng số,  $\theta_0$  là thiên kiến (bias). Mục tiêu của hồi quy tuyến tính sẽ tìm ra các hệ số  $\theta_j$  tối ưu.



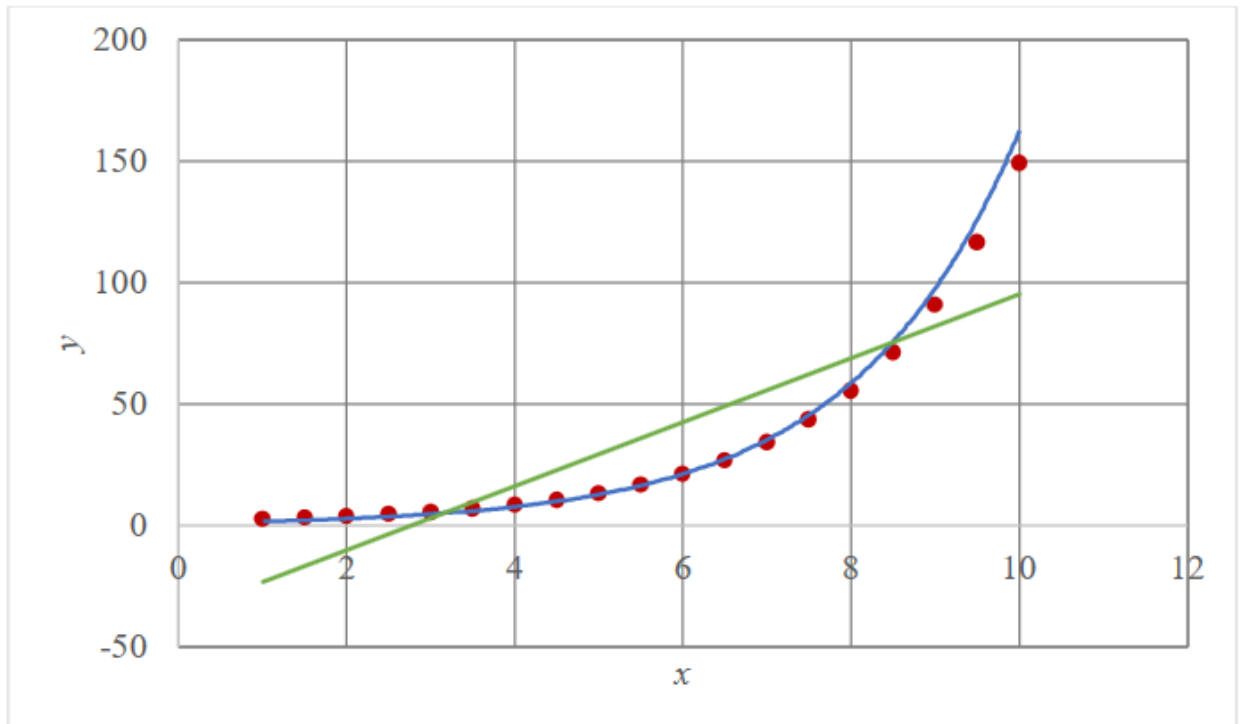
Hình 7. Hyperplane ở đồ thị hai chiều (hình I) và ba chiều (hình II)

Đồ thị hàm số ở Hình 7 trong không gian hai chiều (hai trục tọa độ) là một đường thẳng (line). Nếu trong không gian ba chiều, hàm số được gọi là tuyến tính nếu đồ thị của hàm số là một mặt phẳng (plane). Còn trong không gian nhiều hơn 3 chiều, đồ thị hàm số là một siêu phẳng (hyperplane).

### **b. Hồi quy phi tuyến tính**

Trong hồi quy phi tuyến tính, hàm số  $f(x)$  không thể được xem như là tuyến tính nếu như các tham số không phải là tuyến tính [11]. Đồ thị của hàm số phi tuyến tính là một đường cong. Hàm số phi tuyến tính được ký hiệu như sau:  $\hat{y} = f(x, \theta)$ .

Ví dụ: Một hàm số phi tuyến tính là  $\hat{y} = \theta_0 + \theta_1 \exp(\theta_3 x)$ .



Hình 8. Đồ thị hàm số phi tuyến tính (Non Linear Function)

Trong Hình 8, các điểm giá trị có xu hướng tạo thành một đường cong, nếu sử dụng hồi quy tuyến tính (đường thẳng màu xanh) thì hàm số tìm được sẽ có phương sai lớn. Để tìm được hàm số phi tuyến tính này, sử dụng thuật toán bình phương tối thiểu Levenberg–Marquardt.

### c. Hồi quy đa thức

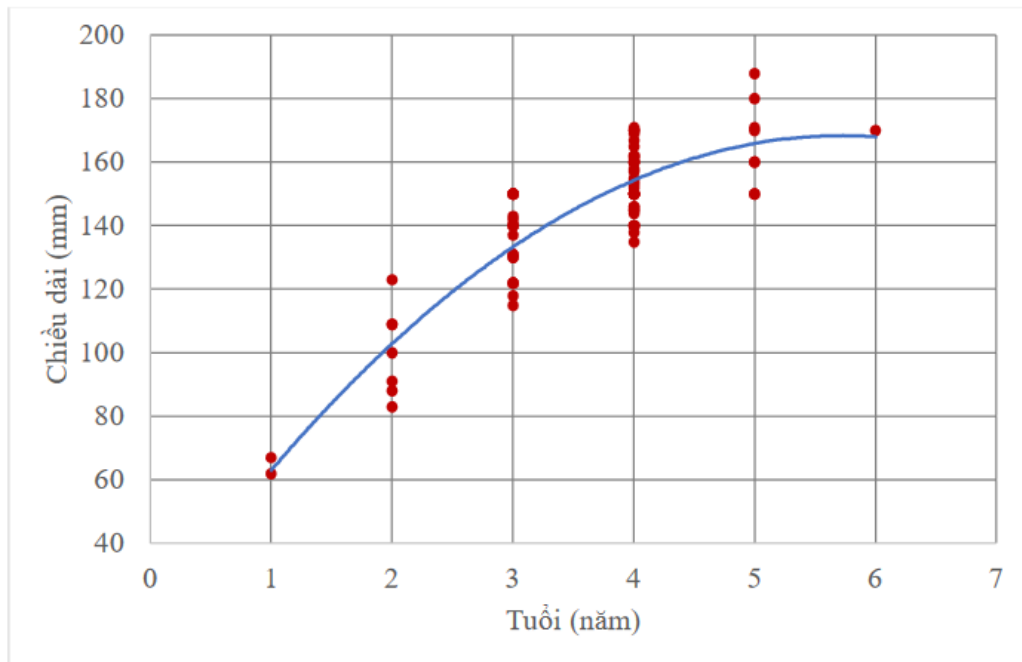
Như đã đề cập ở phần trên, đa hồi quy là một dạng của hồi quy, trong đó có nhiều hơn một biến độc lập. Đa hồi quy bao gồm một kỹ thuật gọi là hồi quy đa thức. Trong hồi quy đa thức, biến phụ thuộc hồi quy vào lũy thừa của các biến độc lập [12].

Ví dụ: Vào năm 1981,  $n = 78$  con cá *Thái Dương xanh* (*Bluegrill*) được lấy mẫu ngẫu nhiên ở Lake Mary, tiểu bang Minnesota, Mỹ. Nhà nghiên cứu đã đo lường và ghi lại các dữ liệu sau [13]:

- Chiều dài ( $y$ ) của con cá, đơn vị mili-mét.

- Độ tuổi ( $x$ ) của con cá đó, đơn vị năm.

Kết quả thu thập được được thể hiện qua biểu đồ sau (Hình 9):



Hình 9. Biểu đồ hồi quy đa thức (thể hiện tương quan của cá qua từng độ tuổi)

Mặc dù chiều dài của con cá tăng lên qua từng độ tuổi, nhưng lại không hoàn toàn theo tuyến tính (Hình 9). Để mô hình hoá dữ liệu này, người ta xây dựng một mô hình đa thức bậc 2, hay còn gọi là hàm số bậc 2 như sau:

$$\hat{y}^{(i)} = \theta_0 + \theta_1 x^{(i)} + \theta_2 (x^{(i)})^2$$

Trong đó:

- $\hat{y}^{(i)}$  là chiều dài của con cá Thái dương xanh thứ  $i$  (mm)
- $x^{(i)}$  là tuổi của con cá Thái dương xanh thứ  $i$  (năm)

Bên cạnh đa thức bậc 2, hồi quy đa thức còn có các dạng khác từ bậc 3 đến  $n$ . Công thức tổng quát:

$$\hat{y} = \theta_0 + \theta_1 x + \theta_2 x^2 + \dots + \theta_i x^i, \forall i = 1, 2, \dots, n$$

Để tìm các tham số ta sử dụng phương pháp bình phương tối thiểu.

#### 1.4.1.2. Phân loại học có giám sát

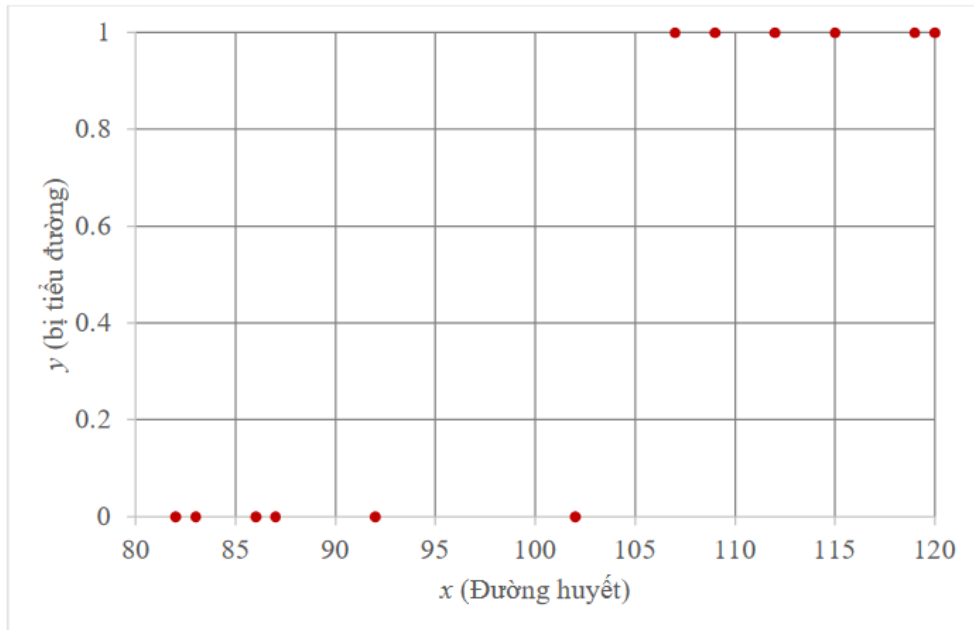
##### 1. Phân loại Nhị phân và Đa lớp

Phân loại là một hình thức học trong Học máy với mục đích là tạo ra một mô hình dự đoán đầu ra là các giá trị rời rạc: thể loại (category) hoặc lớp (class), chẳng hạn dựa vào thông tin xét nghiệm máu, chiều cao, cân nặng, huyết áp,... mà dự đoán có mắc bệnh tiểu đường hay không.

Ví dụ: Một trạm y tế sử dụng thông tin đường huyết trong máu của các bệnh nhân, thu được bảng số liệu sau:

*Bảng 3. Dữ liệu về chỉ số đường huyết của các bệnh nhân*

Chỉ số đường huyết	Bị tiểu đường
82	0
92	0
112	1
102	0
107	1
109	1
...	...



Hình 10. Biểu đồ thể hiện chỉ số đường huyết bệnh tiểu đường

Trong ví dụ này, kết quả tiểu đường được chia thành hai trường hợp (hoặc hai lớp) là không bị tiểu đường (non-diabetic) và bị tiểu đường (diabetic). Đây được gọi là phân loại nhị phân. Kết quả phân loại dựa vào xác suất để có giá trị 0 (không thể) và 1 (chắc chắn). Tổng xác suất cho mỗi lớp là 1 (hoặc bị tiểu đường hoặc không bị tiểu đường). Điều đó có nghĩa là nếu một bệnh nhân có xác suất dự đoán bị tiểu đường là 0,4 thì xác suất tương ứng cho không bị tiểu đường là 0,6. Có một giá trị ngưỡng, thường là 0,5 nhằm xác định kết quả lớp dự đoán. Nếu xác suất dự đoán lớn hơn hoặc bằng ngưỡng, thì lớp dự đoán được gọi là positive class (trong trường hợp này là bị tiểu đường) và ngược lại, negative class (không bị tiểu đường). Người ta gọi ngưỡng đó là ranh giới quyết định.

- Nếu  $f(x) \geq 0,5$ , dự đoán  $y = 1$  (bị tiểu đường)
- Nếu  $f(x) < 0,5$ , dự đoán  $y = 0$  (không bị tiểu đường)

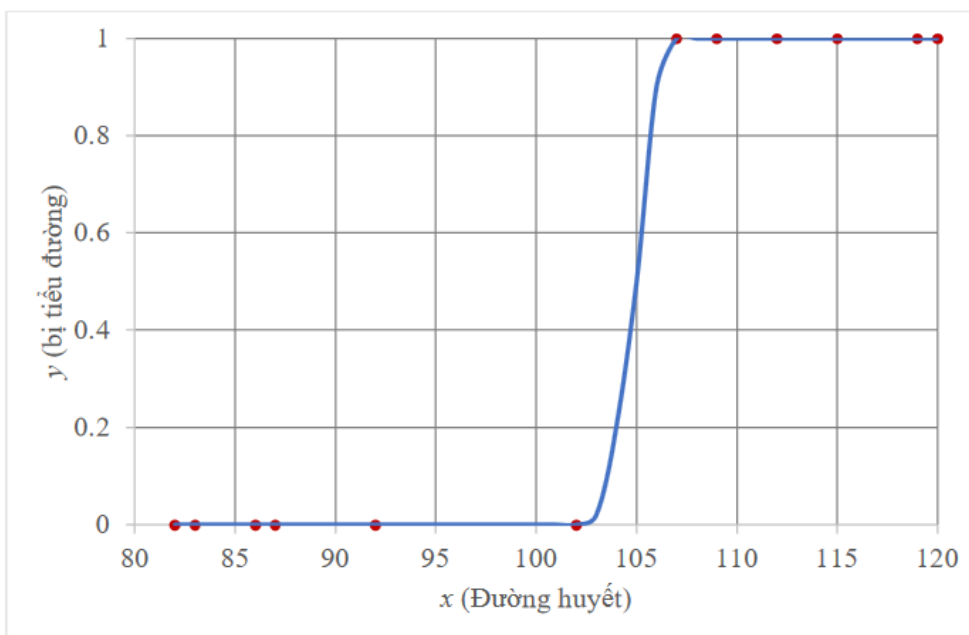
Hồi quy luận lý (Logistic regression):  $0 \leq f(x) \leq 1$

Mô hình hồi quy luận lý:  $f(x) = g(\theta^T x)$

Trong đó:  $g(z) = \frac{1}{1+e^{-z}}$  được gọi là hàm sigmoid hay hàm luận lý.



Đồ thị của hàm sigmoid có dạng một đường cong bị chặn trên và chặn dưới, được minh họa trong Hình 11:



Hình 11. Biểu đồ phân loại các ca bệnh tiểu đường

$f(x) = P(y = 1|x; \theta)$  là xác suất để  $y = 1$  với  $x$  đã có và tham số  $\theta$ .

Giả sử nếu:  $f(x) = P(x; \theta) = 0,4$  thì  $P(x; \theta) = 1 - 0,4 = 0,6$

Ngoài ra còn có mô hình phân loại đa lớp với số lớp phân loại nhiều hơn 2. Chẳng hạn, để làm rõ hơn mức độ bị tiểu đường, người ta chia thành các trường hợp: không bị tiểu đường, tiểu đường loại 1 và tiểu đường loại 2. Tổng xác suất của mỗi lớp vẫn là 1, có nghĩa là tình trạng của bệnh nhân chỉ rơi vào một trong ba trường hợp đã kể trên.

Có một số thuật toán để tối ưu trong mô hình này: Gradient descent, BFGS, L-BFGS, và Conjugategradient.

Quay lại ví dụ trên, khi so sánh các nhãn dự đoán dựa trên hàm luận lý của mô hình ( $\hat{y}$ ) và các nhãn thực tế ( $y$ ) trên một vài dữ liệu  $x$ , ta được:

Bảng 4. Bảng đối chiếu  $y$  và  $\hat{y}$  của mô hình luận lý

$x$	$y$	$\hat{y}$
83	0	0
119	1	1
104	1	0
105	0	1
86	0	0
109	1	1

Ta chuyển đổi bảng trên thành ma trận lỗi tương ứng, được:

		$\hat{y}$	
		0	1
$y$	0	2	1
	1	1	2

Hình 12. Ma trận lỗi

Ma trận tổng quát cho hai nhãn dự đoán (0 và 1):

		Giá trị dự đoán	
		0	1
Giá trị thực tế	0	True Negative (TN)	False Positive (FP)
	1	False Negative (FN)	True Positive (TP)

Hình 13. Ma trận lỗi bài toán phân loại nhị phân

Từ ma trận này có thể tính được các giá trị sau:

- $Accuracy (A) = \frac{TP+TN}{TP+TN+FP+FN}$  là tỷ lệ dự đoán đúng cho cả 2 trường hợp  $\hat{y} = 1, y = 0$ .
- $Precision (P) = \frac{TP}{TP+FP}$  là tỷ lệ giữa số lần dự đoán  $\hat{y} = 1$  **đúng (true positives)** so với tổng số dự đoán  $\hat{y} = 1$  (**true positives + false positives**).
- $Recall (R) = \frac{TP}{TP+FN}$  là tỷ lệ giữa số lần dự đoán  $\hat{y} = 1$  **đúng (true positives)** so với tổng số  $y = 1$  thực tế (**true positives + false negatives**).

Có nhiều cách đánh giá nên gây khó khăn trong việc so sánh mô hình, để thống nhất, trong đề tài này sẽ sử dụng một đại lượng trung bình kết hợp P và R. Trung bình Pythagore bao gồm bộ ba số: trung bình cộng, trung bình nhân và trung bình điều hòa. Trung bình cộng cần các giá trị có cùng đơn vị, trung bình nhân phù hợp với các giá trị có nhiều đơn vị, trong khi đó thì trung bình điều hòa dùng cho các giá trị là tỷ lệ [14]. Vì vậy, **F<sub>1</sub> score**, hay còn gọi là trung bình điều hòa của P và R được áp dụng để đánh giá mô hình và được tính bằng công thức sau:

$$F_1 score = 2 \frac{P \cdot R}{P + R}$$

## **2. Ứng dụng của phân loại**

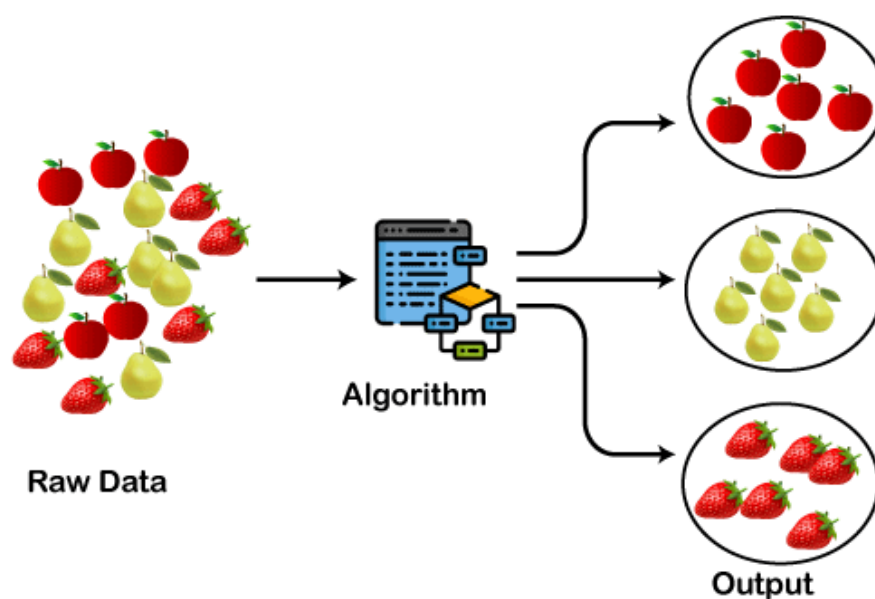
- Robot Willow Garage PR2 có thể hoạt động như một người phục vụ, nhận biết một số loại nước uống và đưa chúng đến với người ra lệnh [15].
- Tự động phân loại hoa trên một số lượng lớn các lớp dựa trên các đặc trưng hình dáng/kết cấu cục bộ, hình dáng viền, sự phân bố không gian tổng thể của cánh hoa và màu sắc [16].

### **1.4.2. Học không giám sát**

#### **1.4.2.1. Phân cụm**

Phân cụm (tiếng anh là Clustering) là một hình thức học trong Học máy không giám sát, trong đó các quan sát được nhóm thành các cụm dựa trên các điểm tương đồng trong dữ liệu hoặc các đặc trưng. Sở dĩ gọi là không giám sát bởi vì nó không sử dụng các nhãn đã biết trước đó để huấn luyện mô hình, thay vào đó là các nhãn là các cụm được gán cho các cụm, hoàn toàn dựa trên các đặc trưng của dữ liệu.

Ví dụ: Trong Hình 14, giả sử dữ liệu đầu vào là một hình ảnh có chứa các loại trái cây và thuật toán cần phải phân loại chúng mà không cần dựa trên các nhãn.



Hình 14. Ví dụ về phân cụm (học không giám sát)

Các thuật toán trong học có giám sát tốt nếu tối thiểu hoá được hàm chi phí giữa giá trị  $y$  và  $\hat{y}$ . Đây là công việc khả thi vì dữ liệu đã được gắn nhãn từ trước, tuy nhiên điều đó không đúng trong học không giám sát bởi vì không thể sử dụng bất kỳ điều kiện bên ngoài để nhận biết lỗi.

Có nhiều thuật toán có thể phân cụm dữ liệu, thông dụng nhất là sử dụng thuật toán phân cụm K-Means.

Một trong những lĩnh vực ứng dụng của phân cụm là trong nghiên cứu tiếp thị. Girish Punj và cộng sự đưa ra một số phương pháp phân tích phân cụm và đánh giá dựa trên thực nghiệm [17]. Gần đây hơn là phân tích cụm chất lượng của ngành dịch vụ vận chuyển nhằm cá nhân hoá chiến lược tiếp thị trong giao thông công cộng với dữ liệu khảo sát được thực hiện ở Tây Ban Nha [18].

#### 1.4.2.2. Giảm chiều dữ liệu

Trong quá trình huấn luyện, nếu dữ liệu càng có nhiều chiều thì quá trình tính toán diễn ra sẽ lâu hơn, chi phí tính toán sẽ cao hơn [19]. Đó là lý do tại sao giảm chiều dữ liệu lại cần thiết. Có nhiều cách để giảm chiều dữ liệu chẳng hạn như lựa chọn các đặc trưng thật sự cần thiết, liên quan đến dữ liệu đầu ra. Phương pháp thứ hai là sử dụng dẫn xuất đặc trưng, nghĩa là chuyển đổi các đặc trưng cũ thành

đặc trưng mới thông qua một phương pháp biến đổi tập dữ liệu. Phương pháp thứ ba là phân cụm các điểm dữ liệu tương đồng thành từng nhóm.

Gọi dữ liệu đầu vào là  $x \in {}^{\circ}n$  với  $n$  là số chiều của dữ liệu, thuật toán giảm chiều dữ liệu là  $f(x)$ , thông qua  $f(x)$  tạo ra một tập dữ liệu mới là  $x' \in {}^{\circ}n'$  với  $n' < n$ .

Ví dụ: Cho một tập dữ liệu chứa các thông tin như giới tính, chiều cao (cm), cân nặng (kg) của 500 người (dữ liệu sinh ngẫu nhiên). Người ta cần từ các thông tin này để có thể suy ra tình trạng béo phì hay suy dinh dưỡng một cách khoa học. Dưới đây là một vài dữ liệu mẫu từ tập dữ liệu này.

*Bảng 5. Dữ liệu về thông tin chiều cao và cân nặng*

Giới tính	Chiều cao (cm)	Cân nặng (kg)
Nam	189	87
Nữ	185	110
Nữ	195	104
Nữ	172	67
Nam	178	52
...	...	...

Từ thông tin về cân nặng không đủ cơ sở để nhận định người đó có bị béo phì hay không. Chính vì vậy mà chỉ số BMI của Adolphe Quetelet [20] đã ra đời. Công thức được tính như sau:

$$BMI = \frac{w}{h^2} \text{ với } w \text{ là trọng lượng (kg) và } h \text{ là chiều cao (m)}$$

Từ công thức trên khi tính ra chỉ số BMI, ta có thể phân loại mức độ béo phì dựa vào các mốc giá trị (theo Tổ chức Y tế thế giới):

- 1: Gầy, dưới 18,5
- 2: Bình thường, từ 18,5 đến 24,9
- 3: Tiền béo phì, từ 25 đến 29,9
- 4: Béo phì độ I, từ 30 đến 34,9
- 5: Béo phì độ II, từ 35 đến 39,9
- 6: Béo phì độ III, trên 40

Bảng 6 phân loại mức độ béo phì của dữ liệu được cho trước trong Bảng 5 dựa vào công thức tính BMI và các mốc để phân loại:

*Bảng 6. Dữ liệu thông tin sức khỏe và phân loại mức độ béo phì*

Giới tính	Chiều cao (cm)	Cân nặng (kg)	Chỉ số BMI	Phân Loại
Nam	189	87	24.36	2
Nữ	185	110	32.14	4
Nữ	195	104	27.35	3
Nữ	172	67	22.65	2
Nam	178	52	16.41	1
...	...	...	...	...

Bây giờ, ta có thể giảm đi ba đặc trưng chiều cao, cân nặng và chỉ số BMI và để lại đặc trưng giới tính và phân loại. Như vậy, ta đã giảm đi được số chiều của dữ liệu, từ  $x \in^{\circ 3}$  thành  $x' \in^{\circ 2}$ .

Trong Giảm chiều dữ liệu có nhiều thuật toán, trong đó có các thuật toán nằm trong nhóm có giám sát, chẳng hạn thuật toán *Phân tích phân biệt tuyến tính*

(LDA) và không giám sát, chẳng hạn *Phân tích thành phần chính (PCA)*. Vì vậy, trong mục này sẽ đề cập đến thuật toán PCA.

*Một số ứng dụng của Giảm chiều dữ liệu như:*

- Ella Bingham cùng cộng sự sử dụng phép chiếu ngẫu nhiên để giảm kích thước dữ liệu trong xử lý hình ảnh nhiễu, không nhiễu và truy xuất thông tin trong tài liệu văn bản [21]. Ella Bingham còn chỉ ra được kết quả của phương pháp chiếu và PCA tốn kém ít chi phí tính toán hơn.
- S.A. Bleha cùng cộng sự sử dụng giảm chiều dữ liệu và trích xuất đặc trưng để ứng dụng trong nhận diện người dùng máy vi tính. Người dùng gõ một mật khẩu và hệ thống nhận diện hợp lệ, không những xác định từ mà còn xác định thời gian giữa mỗi lần nhấn phím [22].
- Irina Perfilieva sử dụng *phép biến đổi mờ (fuzzy transforms)* để giảm chiều dữ liệu và áp dụng vào việc ước tính sự biến động thị trường [23].

### **1.4.3. Học bán giám sát**

Học bán giám sát có một nửa dữ liệu là có giám sát và nửa kia không giám sát, đôi khi chỉ có một phần nhỏ dữ liệu được gắn nhãn. Tùy theo thiết lập mà học bán giám sát dựa trên sự mở rộng của học có giám sát và học không giám sát. Vì vậy mà học bán giám sát được chia thành các phương pháp khác nhau: phân loại bán giám sát và phân cụm ràng buộc. Ngoài ra, bán giám sát còn có thể ứng dụng vào hồi quy và giảm chiều dữ liệu,...

Trong toán học, tri thức về  $P(x)$  học được từ dữ liệu không có nhãn phải chứa các thông tin hữu ích trong suy luận của  $P(y|x)$ . Điều đó có nghĩa là nếu không có thông tin từ suy luận  $P(x) = \frac{P(y \cap x)}{P(x)} = \frac{P(y) \cdot P(y)}{P(x)}$ , học bán giám sát sẽ không thể cải tiến hơn so với học có giám sát, mặt khác, nếu thông tin từ suy luận là sai, thì độ chính xác qua tập dữ liệu không có nhãn sẽ giảm xuống.



Một số phương pháp trong học bán giám sát phổ biến: Generative models, Low-density separation, Graph-based methods,...

#### 1.4.3.1. Phân loại học bán giám sát

Phân loại bán giám sát là sự mở rộng của phân loại có giám sát. Tập dữ liệu đầu vào của mô hình là  $x^{(i)} = \{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$  được chia làm hai phần:  $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(l)}, y^{(l)})\}$  tương ứng với các cặp dữ liệu đã được gán nhãn và  $\{x^{(l+1)}, x^{(l+2)}, \dots, x^{(l+u)}\}$  tương ứng với các dữ liệu chưa được gán nhãn,  $m = l + u$  và thường là  $u \gg l$ . Mục đích của phân loại bán giám sát là huấn luyện một mô hình  $f$  từ dữ liệu đã được gán nhãn và chưa được gán nhãn sao cho hiệu suất tốt hơn phân loại có giám sát trên tập dữ liệu chỉ gán nhãn.

#### 1.4.3.2. Phân cụm ràng buộc

Phân cụm ràng buộc là sự mở rộng của phân cụm không giám sát. Dữ liệu huấn luyện bao gồm các dữ liệu chưa gán nhãn  $\{x^{(1)}, x^{(2)}, \dots, x^{(n)}\}$  và một vài “thông tin giám sát” liên quan đến các cụm. Ví dụ, thông tin các ràng buộc phải liên kết của  $x^{(m)}$  và  $x^{(n)}$  nên nằm chung một cụm; thông tin các ràng buộc không thể liên kết của  $x^{(m)}$  và  $x^{(n)}$  không nên nằm chung một cụm; thông tin ràng buộc kích thước của các cụm. Tuy nhiên, trên thực tế, các ràng buộc trên có thể vi phạm nhưng không bị cấm hoàn toàn. Cũng như phân loại bán giám sát, mục đích của phân cụm ràng buộc là huấn luyện một mô hình có thể phân cụm tốt hơn nếu chỉ mỗi phân cụm trên dữ liệu chưa được gán nhãn.

Một số thuật toán phân cụm ràng buộc có thể áp dụng: COP K-means, PCKmeans(Pairwise Constrained K-means), CMWK-Means (Constrained Minkowski Weighted K-Means), HMRFs (Hidden Markov Random Fields).

#### 1.4.3.3. Ứng dụng của học bán giám sát

- Nhận diện và phân loại lưu lượng mạng thông qua phương pháp phân loại bán giám sát ngoại tuyến và trực tuyến của Jeffrey Ertan và các cộng sự

là công trình đầu tiên ứng dụng các kỹ thuật học bán giám sát vào giải quyết các vấn đề phân loại lưu lượng mạng [24].

- Một sự kết hợp giữa phân loại bán giám sát và CNN để nhận diện điện não đồ. Việc lấy dữ liệu gán nhãn từ điện não đồ rất khó khăn nên Minjie Liu cùng cộng sự sử dụng thuật toán lượng tử hoá bán giám sát dựa trên K-Means, sử dụng mô hình huấn luyện trước là CNN [25].

#### 1.4.4. Học tăng cường

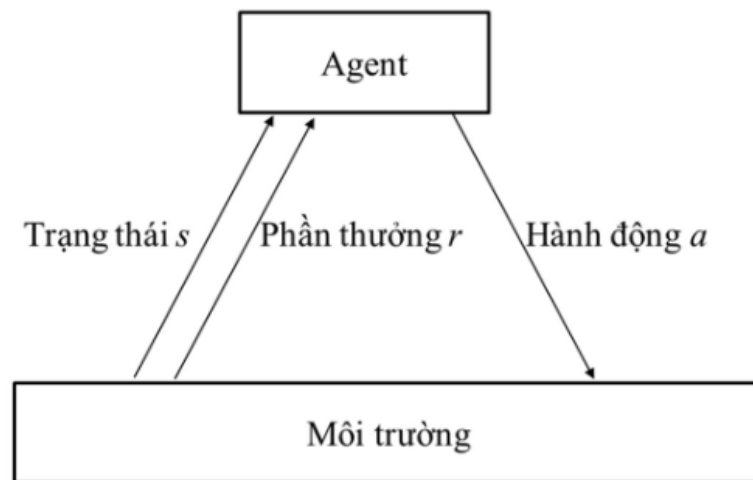
Trong các phương pháp học thì phương pháp học giám sát sử dụng thuật toán chạy trên tập dữ liệu chính xác, trong khi đó thì học không giám sát, thuật toán chỉ có thể tìm sự tương đồng trong dữ liệu để phân loại. Có một vấn đề, nếu dữ liệu bị sai thì thuật toán không biết cải thiện như thế nào. Thuật toán học tăng cường sẽ thử một vài chiến lược khác nhau và chọn ra chiến lược nào tốt nhất.

Ví dụ: Quá trình huấn luyện robot như sau: Robot, gọi là *agent*, có một vài cảm biến ghi nhận các *trạng thái* của môi trường và một vài *hành động* có thể thực hiện để thay đổi trạng thái này.

Chẳng hạn, robot có cảm biến camera và cảm biến siêu âm và hành động là “tiến về phía trước” và “xoay”. Nhiệm vụ của robot là học một chiến lược điều khiển, gọi là *policy*, chọn các hành động phù hợp để đạt được mục tiêu. Chẳng hạn, robot có một mục tiêu là tự sạc điện khi pin gần cạn kiệt. Vậy *agent* làm sao để học được *policy* thông qua các thử nghiệm trong môi trường?

Giả sử mục tiêu của *agent* được định nghĩa bởi một hàm phần thưởng được gán bằng một con số, đó là phần thưởng cho *agent* với mỗi hành động mà *agent* thực hiện trong từng state. Ví dụ phần thưởng ngay lập tức cho robot sạc điện thành công là 100 và -10 nếu làm các việc đáng lẽ phải tránh. Phần thưởng này được tạo bởi môi trường xung quanh. Phần thưởng cho robot biết mục tiêu là gì, không phải cách để đạt được mục tiêu đó, giống với học có giám sát. Robot sẽ thực hiện một chuỗi các hành động, quan sát kết quả (có được thưởng hay không) và học được *policy*. Trong *policy*  $\pi = S \rightarrow A$ , từ bất kỳ trạng thái khởi đầu nào, *agent* chọn một

hành động để đạt được phần thưởng tích lũy lớn nhất, trong đó  $S$  là tập trạng thái,  $A$  là tập hành động.



Hình 15. Chu trình học của học tăng cường

Mục tiêu là chọn các hành động để đạt giá trị lớn nhất:

$$R = r_0 + \gamma r_1 + \gamma^2 r_2 + \dots + \gamma^k r_k + \dots = \sum_{k=0}^{\infty} (\gamma^k r_k) \text{ với } (0 \leq \gamma \leq 1)$$

Các nhiệm vụ đưa ra liên tục và không có điểm dừng, nên dự đoán phần thưởng ở tương lai vô hạn là điều không thể. Giải pháp cho vấn đề này được gọi là chiết khấu. Không có sự chắc chắn về những gì xảy ra trong tương lai, vì vậy mà phần thưởng dự đoán trong tương lai cần phải được chiết khấu tùy theo khả năng dự đoán bị sai. Vì vậy mà phần thưởng mong đợi đến càng sớm thì chắc chắn sẽ dự đoán chính xác hơn các phần thưởng xuất hiện rất lâu, bởi vì có rất nhiều thứ có thể bị thay đổi. Tham số  $\gamma$  được thêm vào để phần thưởng thứ  $i$  chiết khấu theo hàm mũ  $\gamma^i$ . Nếu  $\gamma \rightarrow 0, \gamma^k = 0$  thì chúng ta nên bỏ qua hầu hết các dự đoán tương lai, chỉ xem xét các phần thưởng hiện tại. Ngược lại, nếu  $\gamma \rightarrow 1, \gamma^k = 1$ , không có sự khấu trừ, do đó nên tập trung vào dự đoán tương lai hơn là phần thưởng hiện tại.

Một số thuật toán học tăng cường là Q-learning, Deep Q-Network (DQN), SARSA, Monte Carlo,...

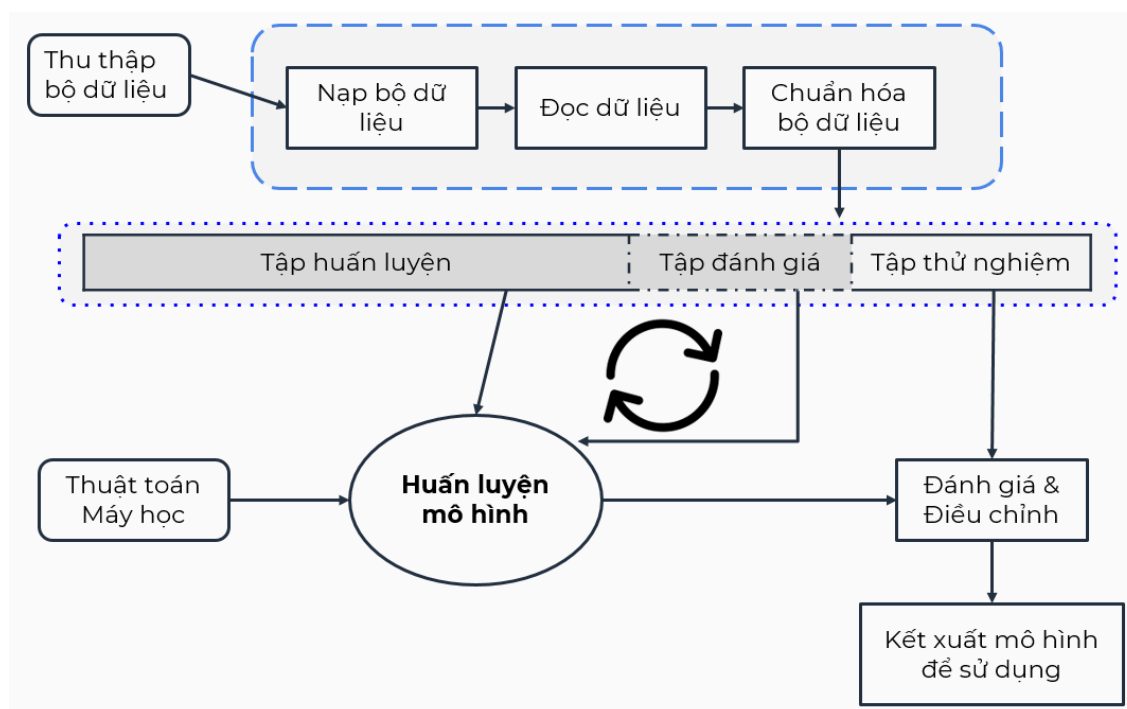
Một số ứng dụng của học tăng cường:

- Chương trình AlphaGo của Google đánh bại bậc thầy cờ vây Lee Se-dol với tỷ số 3-0. Đây là trận đấu được xem như khoảnh khắc quan trọng của trí tuệ nhân tạo [10].
- Ngoài ra, học tăng cường còn ứng dụng rất nhiều trong lĩnh vực xử lý ngôn ngữ tự nhiên, hệ thống chẩn đoán, máy dịch, sinh văn bản, thị giác máy tính [26].

## 1.5. Các phương pháp tiếp cận vấn đề trong học máy

### 1.5.1. Tiếp cận theo phương pháp Học máy truyền thống (ML)

Phương pháp tiếp cận và giải quyết vấn đề bằng phương pháp Học máy truyền thống (Machine Learning) được minh họa bằng sơ đồ dưới đây:



Hình 16. Sơ đồ hoạt động của phương pháp học máy truyền thống

Ở cách tiếp cận truyền thống theo phương pháp Học máy, để giải quyết các vấn đề sẽ áp dụng một trong các thuật toán thuộc nhóm có giám sát, không giám sát, bán giám sát hoặc học tăng cường đã nêu ở mục trước tùy theo loại dữ liệu. Bước quan trọng trong cách tiếp cận này ngoài việc thu thập số lượng dữ liệu cực lớn thì còn phải *trích chọn các đặc trưng (features extraction)* phù hợp của hình

ảnh, sau đó là xây dựng một thuật toán tối ưu để tạo được mô hình tốt. Ngoài ra còn có bước tiền xử lý nằm sau giai đoạn thu thập dữ liệu ảnh nhằm loại bỏ các chi tiết nhiễu, hay *tăng số lượng ảnh (data augmentation)*. Tập các kỹ thuật liên quan đến đặc trưng được gọi là *Feature engineering*.

Chẳng hạn trong ứng dụng vào phân loại hình ảnh loài hoa, để phân loại được, thuật toán cần phải biết được các đặc trưng tương ứng với hình ảnh. Để phân loại tốt trên nhiều loài, các đặc trưng cần phải được chuyển đổi theo chuẩn thống nhất từ ban đầu, bao gồm đơn vị đo, sai số đo và cân bằng tỷ lệ (*scale out*). Cân bằng tỷ lệ tập dữ liệu tức bao gồm các thao tác tiêu chuẩn hóa (normalization, standardization) và chính quy hóa (regularization) [27].

Hàm Normalization đóng vai trò cân bằng tỷ lệ của đặc trưng  $j$  của dữ liệu đầu vào thứ  $i$ :

$$x_j^{(i)'} = \frac{x_j^{(i)} - (x_j)}{(x_j) - (x_j)} x_j^{(i)'} \in [0,1]$$

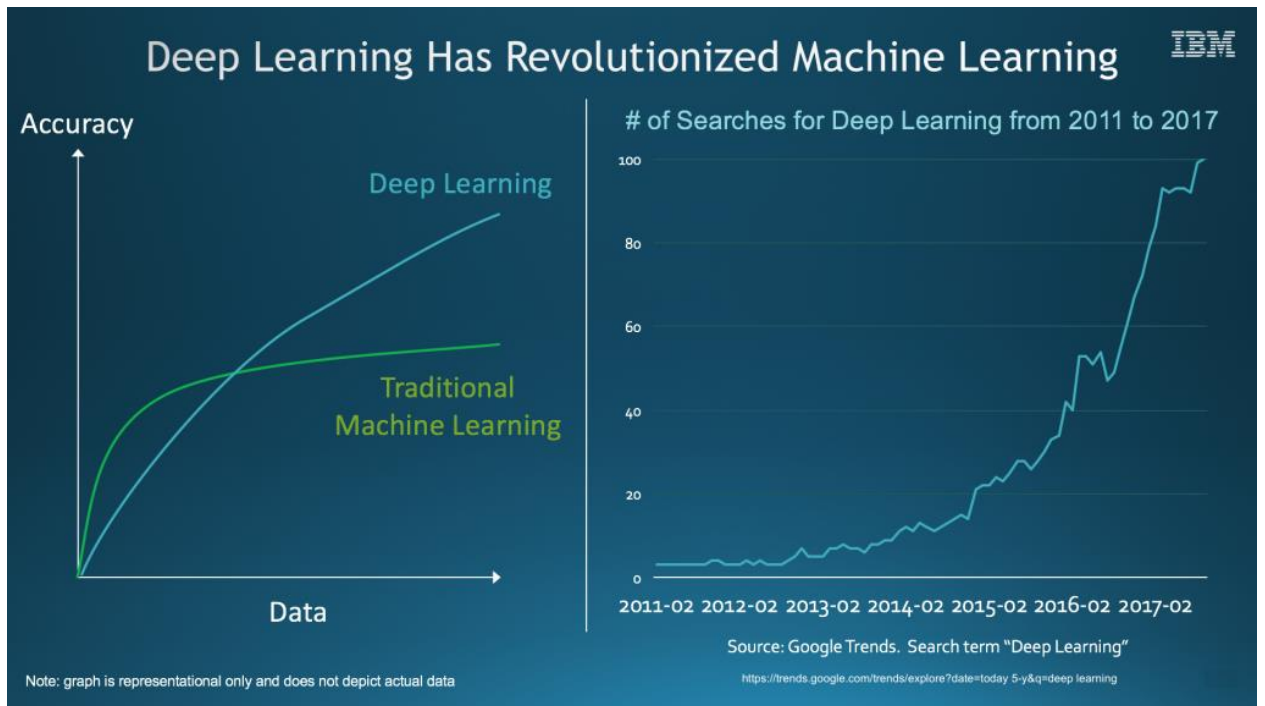
$$\text{hoặc } x_j^{(i)'} = \frac{x_j^{(i)} - (x_j)}{(x_j) - (x_j)} x_j^{(i)'} \in [-1,1]$$

Đối với công thức biến đổi thứ hai, phạm vi dữ liệu trong đoạn  $[-1, 1]$  có kỳ vọng không (*zero-mean*) theo phân phối chuẩn tắc. Còn công thức bên dưới cân bằng tỷ lệ của đặc trưng theo kỳ vọng không và *phương sai đơn vị (unit-variance)*:

$$\text{Standardization: } x_j^{(i)'} = \frac{x_j^{(i)} - \mu}{\sigma}$$

Chính quy hóa giúp giải quyết kết quả dự đoán bị *overfitting*. Bằng cách thêm vào hàm thất thoát (loss function), các tham số  $\theta$  trong thuật toán có thể hội tụ về giá trị nhỏ hơn, làm giảm đáng kể bị *overfitting*.

$$\text{Regularization: } J(\theta) = \frac{1}{2m} \left[ \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n \theta_j^2 \right]$$



Hình 17. Biểu đồ sánh hiệu suất giữa Học máy và Học sâu

Tận dụng được sức mạnh của phần cứng máy vi tính và tiến bộ của kỹ thuật mạng Học sâu mà các công việc trong tiền xử lý dữ liệu được tự động hóa và đạt tính ổn định và chính xác cao. Tuy nhiên, trên thực tế, một kiến trúc mạng Học sâu đơn giản không thể áp dụng cho nhiều nhiệm vụ khác nhau và việc tạo ra được một kiến trúc mạng đầy đủ là rất khó. Nguyên nhân khó khăn chính là vì mạng Học sâu cần rất nhiều tham số để tối ưu hóa và đạt hiệu suất cao cùng với tập dữ liệu phải rất lớn. Đây cũng chính là những bế tắc của Học sâu trong những năm cuối thế kỷ 20. Chính vì vậy, các nhà khoa học của khắp các nơi trên thế giới đã và đang nghiên cứu, cải tiến được kiến trúc của các mạng Học sâu, gọi là mô hình huấn luyện trước để giải quyết được nhiều vấn đề khó hơn, có độ chính xác cao hơn, đã đưa thế giới thoát khỏi hai mùa đông AI kéo dài gần 40 năm để đến với kỷ nguyên của Học sâu.

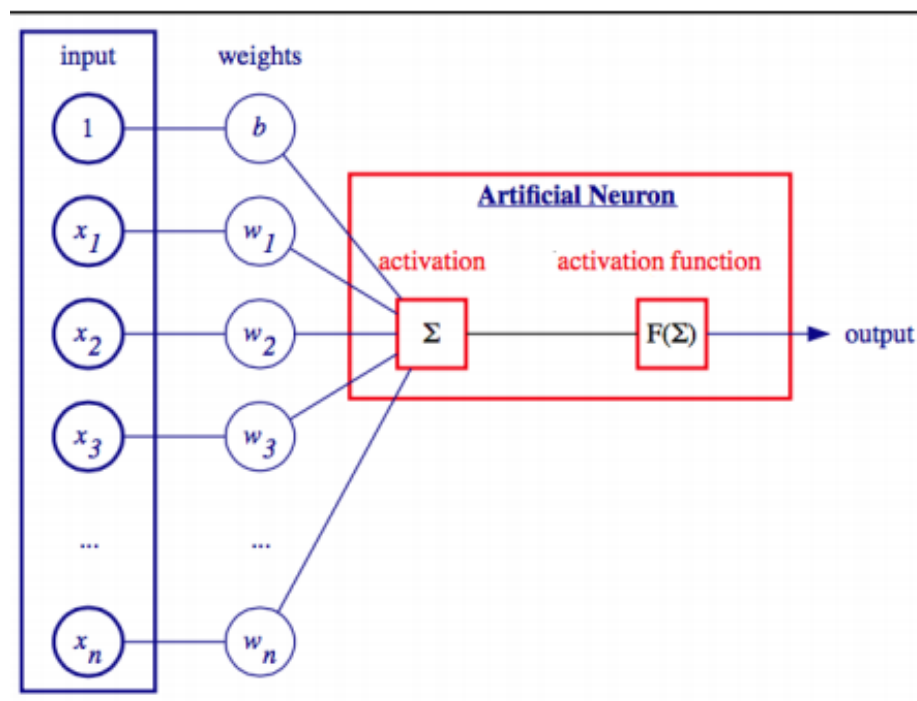
Trong Học sâu, trích chọn đặc trưng là một phần không thể thiếu trong quá trình huấn luyện. Mô hình không chỉ học cách dự đoán mà còn học cách trích chọn đặc trưng từ dữ liệu thô. Ví dụ trong nhận diện hình ảnh, lớp đầu tiên thường là phát hiện cạnh, một trong những kỹ thuật của *Feature engineering* trong nhận diện hình ảnh [29].

Đề tài này sẽ đi theo hướng tiếp cận thứ hai trong việc giải quyết vấn đề đặt ra ban đầu, đó là sử dụng phương pháp *Học sâu* đã được tích hợp sẵn trong hầu hết các thư viện Học máy phổ biến hiện nay.

## 1.6. Giới thiệu mạng nơron nhân tạo

### 1.6.1. Mô hình toán học mạng nơron

Trong học máy, nơron được định nghĩa là một hàm toán học nhận vào một hay nhiều giá trị đầu vào được nhân với các trọng số. Giá trị này sau đó được chuyển đến một hàm số, được gọi là hàm kích hoạt, để trở thành đầu ra của nơron.



Hình 18. Mô hình toán học mạng Nơron

Nơron được định nghĩa như công thức sau:

$$y = f\left(\sum x_i w_i + b\right)$$

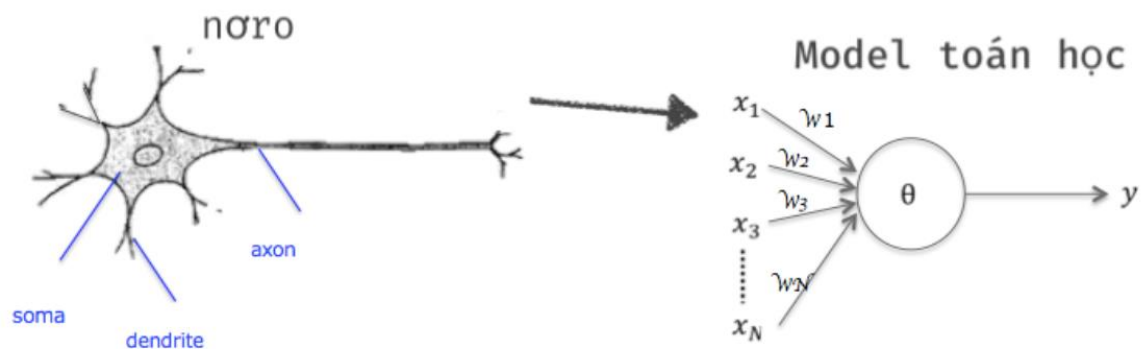
Trong đó:

- $x_i$  là giá trị đại diện cho dữ liệu đầu vào của nơron
- $w_i$  là trọng số, đại diện cho tầm quan trọng của dữ liệu đầu vào

- $b$  là một tham số bổ sung cho neuron, dùng để điều chỉnh giá trị đầu ra của neuron
- $\sum x_i w_i$  được gọi là giá trị kích hoạt

Hàm  $f$  là hàm kích hoạt, hay còn gọi là hàm chuyển đổi. Có nhiều loại hàm kích hoạt khác nhau.

### 1.6.2. Mạng neuron nhân tạo



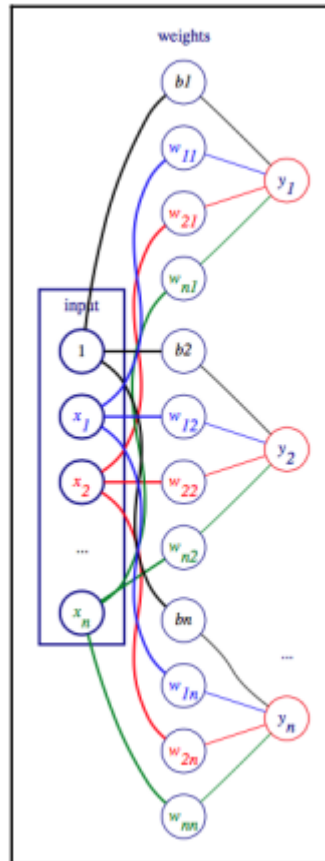
Hình 19. Neuron nhân tạo mô phỏng neuron sinh học

Mạng neuron nhân tạo (*Artificial Neural Networks*) mô phỏng theo neuron sinh học của con người, là một mạng lưới gồm nhiều neuron được tổ chức thành các lớp kết nối với nhau, với nhiều nhánh đầu vào và một nhánh đầu ra. Mạng đem lại nhiều ứng dụng trong các lĩnh vực khác nhau trong thực tế, chẳng hạn trong xử lý và nhận diện hình ảnh.

### 1.6.3. Mạng neuron một lớp

Trong phần trước, ta biết được Neuron chỉ cung cấp 1 giá trị đầu ra duy nhất nhưng trong các bài toán thực tế, chúng ta cần nhiều hơn một giá trị đầu ra. Ví dụ, trong bài toán nhận dạng chữ số viết tay, chúng ta cần 10 kết quả đầu ra – tương ứng với tỉ lệ phần trăm dự đoán của các chữ số từ 0 đến 9. Trong trường hợp này, ta cần dùng đến mạng neuron (Hình 20).





Hình 20. Mạng nơron đơn giản (1 lớp ẩn)

Chúng ta sẽ dự đoán được chữ số bằng cách lấy kết quả đầu ra cao nhất giữa các nơron. Nếu  $y_7$  có giá trị cao nhất nghĩa là mô hình đoán chữ số đầu vào là số 7. Sử dụng mạng nơron khiến kết quả đầu ra là nhiều giá trị khác nhau và thay vì một giá trị duy nhất như khi sử dụng một nơron đơn thuần. Bằng cách này, chúng ta có thể có nhiều hơn một kết quả đầu ra, nghĩa là mạng nơron có thể giải quyết được nhiều bài toán thực tế hơn.

#### 1.6.4. Mạng nơron nhiều lớp ẩn

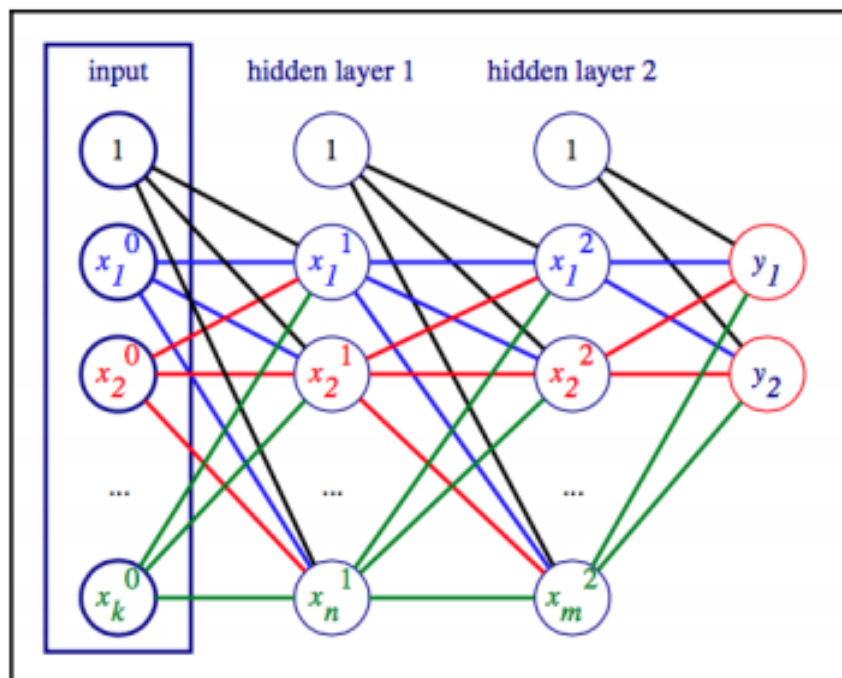
Mạng nơron ban đầu chỉ có một lớp ẩn duy nhất, đây là mạng nơron đơn giản (simple neural network). Nhưng khi nơron có nhiều lớp ẩn trở lên, sẽ được gọi là mạng nơron sâu (deep neural network).

Tại sao chúng ta phải chia các nơron thành nhiều lớp? Như chúng ta đã biết, một nơron chỉ có thể cung cấp một kết quả đầu ra duy nhất, nghĩa là một nơron chỉ có thể xử lý một lượng thông tin hạn chế. Nhưng khi ta chia nơron thành các lớp kết

nối với nhau, kết quả đầu ra của chúng sẽ được thể hiện dưới dạng vector, gồm nhiều giá trị thay vì chỉ một giá trị đơn thuần. Bằng cách này, ta có thể xử lý nhiều dữ liệu hơn.

Một mạng nơron gồm những thành phần sau:

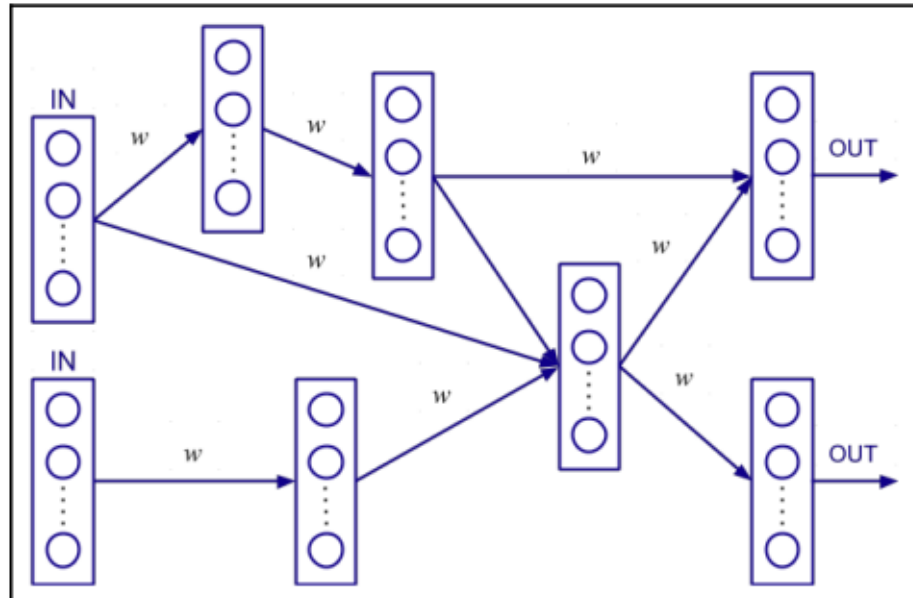
- Input layer: dữ liệu đầu vào,  $x$
- Hidden layer: các lớp ẩn
- Output layer: dữ liệu đầu ra,  $y$
- Các nơron ( $w$ ,  $b$ , và hàm kích hoạt)



Hình 21. Mạng nơron sâu (nhiều lớp ẩn)

Hình 21 mô tả một mạng nơron 3 lớp có đầy đủ kết nối (nghĩa là các nơron ở lớp liền trước kết nối với toàn bộ nơron ở lớp liền sau). Lớp đầu vào có  $k$  nơron đầu vào, lớp ẩn đầu tiên có  $n$  nơron ẩn, lớp ẩn thứ hai có  $m$  nơron ẩn. Lớp đầu ra, trong ví dụ này, có 2 giá trị là  $y_1$  và  $y_2$ . Mỗi kết nối có giá trị trọng số  $w$  riêng biệt. Mạng nơron này là một mô hình tuần tự có hướng, nghĩa là dữ liệu chỉ có thể chạy từ lớp đầu vào đến lớp đầu ra, dữ liệu chỉ có thể được truyền đến mỗi nơron một lần duy nhất.

Ngoài ra, còn có nhiều dạng mạng nơron khác, như hình dưới đây mô tả một mạng nơron có 2 lớp đầu vào, 2 lớp đầu ra, và các lớp ẩn được kết nối ngẫu nhiên với nhau.



Hình 22. Mạng nơron với lớp ẩn kết nối ngẫu nhiên

Hay đặc biệt hơn, mạng nơron còn có thể có các vòng lặp, cho phép dữ liệu chạy qua một nơron nhiều hơn một lần (mạng nơron hồi quy).

## 1.7. Một số thư viện học máy hiện nay

### 1.7.1. TensorFlow

TF là một nền tảng Học máy mã nguồn mở, được thiết kế bởi đội ngũ Google Brain và tổ chức nghiên cứu trí tuệ máy của Google nhằm triển khai các ứng dụng của Học máy và Học sâu theo cách đơn giản. Nó là kết hợp giữa Đại số tính toán của các kỹ thuật tối ưu hoá để dễ dàng tính toán các biểu thức toán học. TF có một hệ sinh thái toàn diện, linh hoạt bao gồm các công cụ, thư viện và tài nguyên cộng đồng cho phép các nhà nghiên cứu xây dựng và triển khai các ứng dụng Học máy. Đây cũng là một trong những thư viện máy học lâu đời nhất.

Trang chủ của TF tại địa chỉ sau: <https://www.tensorflow.org/>. Trang mã nguồn Github của TF nằm tại: <https://github.com/tensorflow/tensorflow>.

TF có tài liệu đầy đủ và bao gồm nhiều thư viện Học máy nên rất phổ biến hiện nay. Vì là một sản phẩm của Google, nên hiện tại, TF đang được ứng dụng rất nhiều trong các sản phẩm của Google như phân loại chữ viết tay, nhận diện hình ảnh, xử lý ngôn ngữ tự nhiên,...

### **1.7.2. Scikit-learn**

Sklearn là một thư viện Học máy mã nguồn mở hữu ích và mạnh mẽ trong Python. Dự án của David Cournapeau bắt đầu vào năm 2007 với tư cách là một dự án của Google Summer of Code. Hiện tại, Sklearn đang được duy trì bởi một đội ngũ các tình nguyện viên. Sklearn cung cấp một sự lựa chọn các công cụ hiệu quả cho Học máy và mô hình thống kê, bao gồm phân loại, hồi quy, phân cụm và giảm chiều dữ liệu với giao diện nhất quán trong Python. Thư viện này phần lớn được viết bằng Python, được xây dựng dựa trên NumPy, SciPy và Matplotlib.

Trang chủ của Sklearn tại địa chỉ sau: <https://scikit-learn.org/>. Trang Github chứa mã nguồn Sklearn nằm tại: <https://github.com/scikit-learn/scikit-learn>.

Một số sản phẩm thương mại sử dụng Sklearn như Spotify, Evernote, Booking.com, J.P.Morgan, Hugging Face, Télécom ParisTech, Aweber,...

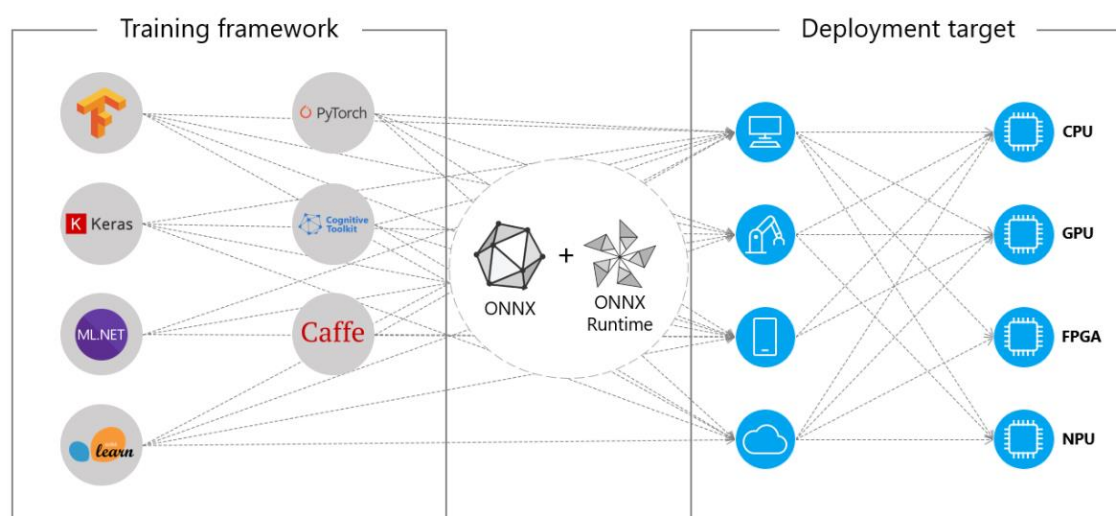
### **1.7.3. Keras**

Keras là một thư viện Học sâu mã nguồn mở dành cho Python. Nó được phát triển bởi một nhà nghiên cứu trí tuệ nhân tạo của Google là Francois Chollet. Keras có thể chạy trên các thư viện mã nguồn mở như TensorFlow, Theano, R hay CognitiveToolkit (CNTK). Mục tiêu thiết kế của Keras là cho phép thử nghiệm các mạng Học sâu nhanh chóng. Các tổ chức hàng đầu như Google, Square, Netflix, Huawei và Uber hiện đang sử dụng Keras.

Trang chủ của Keras tại địa chỉ sau: <https://keras.io/>. Trang Github chứa mã nguồn của Keras nằm tại: <https://github.com/keras-team/keras>.

#### 1.7.4. ONNX

ONNX (Open Neural Network Exchange) là một hệ sinh thái trí tuệ nhân tạo mã nguồn mở của các công ty công nghệ và tổ chức nghiên cứu. ONNX giúp thiết lập các tiêu chuẩn mở để đại diện cho các thuật toán ML và các phần mềm công cụ nhằm thúc đẩy đổi mới và hợp tác trong lĩnh vực AI. ONNX cung cấp framework bao gồm các mô hình đồ thị tính toán có thể mở rộng, các toán tử được tích hợp sẵn và các kiểu dữ liệu tiêu chuẩn, tập trung vào phân suy diễn (đánh giá).



Hình 23. Sơ đồ hoạt động của ONNX

#### 1.7.5. PyTorch

PyTorch được định nghĩa là một thư viện Học máy mã nguồn mở cho Python. Nó được sử dụng cho các ứng dụng như xử lý ngôn ngữ tự nhiên. Ban đầu nó được phát triển bởi nhóm nghiên cứu trí tuệ nhân tạo của Facebook và phần mềm Uber's Pyro để lập trình xác suất.

PyTorch được Hugh Perkins phát triển như một trình bao bọc Python cho LusJIT dựa trên khuôn khổ Torch.

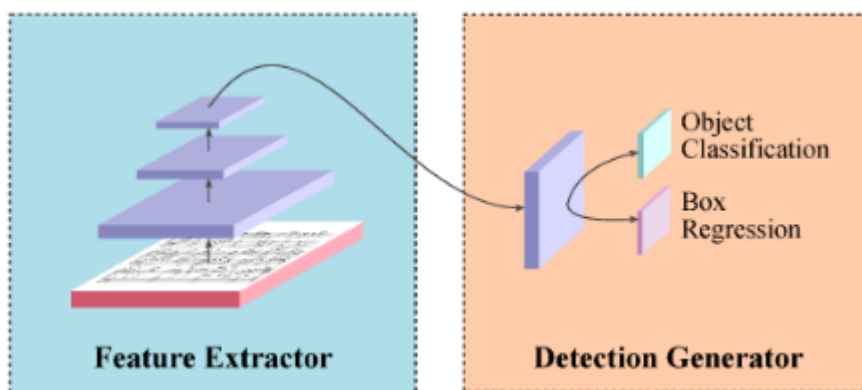
Trang web thông tin chính thức của thư viện PyTorch nằm tại địa chỉ sau: <https://pytorch.org/>. Trang Github chứa mã nguồn của thư viện PyTorch nằm địa chỉ: <https://github.com/pytorch/pytorch>.

## CHƯƠNG 2. PHÁT HIỆN GƯƠNG MẶT

Phát hiện gương mặt luôn là bước đầu tiên thiết yếu trong hệ thống nhận diện gương mặt. Hiện nay có rất nhiều phương pháp và mô hình để phát hiện gương mặt. Trong bài báo cáo sẽ sử dụng mô hình Ultra-lightweight face detection. Mô hình này được xây dựng trên thư viện PyTorch cùng với ONNX và có tốc độ xử lý vượt trội hơn nhiều so với các mô hình phổ biến hiện nay.

### 2.1. Mô hình Ultra-light fast face detection

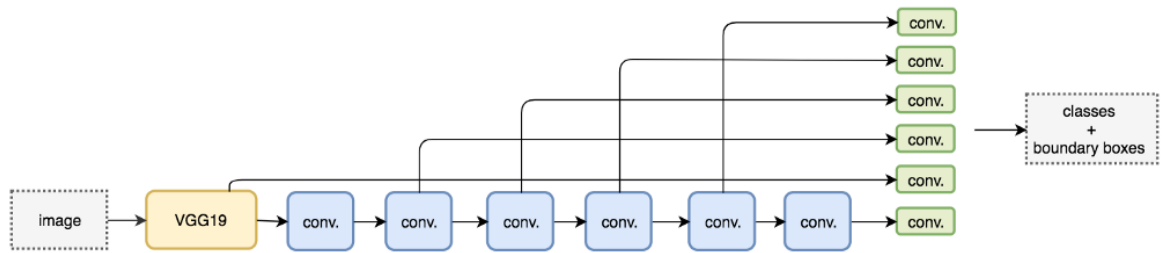
Đây là một mô hình được xây dựng bằng Pytorch-SSD (Single Shot Detector) theo kiểu kiến trúc nhận diện vật thể trong một bước (One-stage Detector).



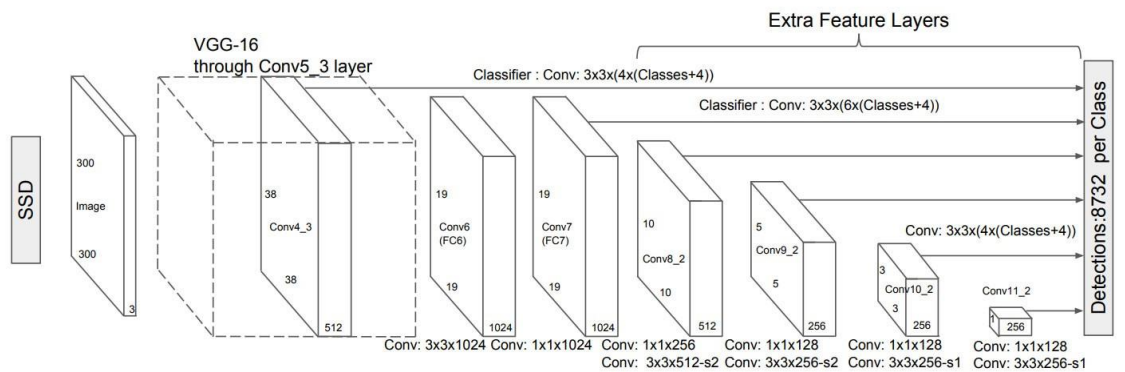
Hình 24. Mô hình nhận diện vật thể trong một bước

Việc gộp hai quy trình xác định và nhận diện vật thể vào trong cùng một bước giúp giảm thiểu thời gian xử lý cho mô hình. So sánh với mô hình phải thực hiện 2 bước như R-CNN thì SSD có tốc độ nhanh hơn cũng như độ chính xác cao hơn.

Khi ảnh đầu vào qua các custom convolution layer (màu xanh dương) sẽ bị giảm chiều và hạ mất đi độ phân giải, lúc đó ta chỉ có thể lấy được thông tin của những vật thể có kích thước lớn. Mô hình SSD giải quyết vấn đề này bằng cách sử dụng thêm các lớp convolutional filter layer (màu xanh lá cây) vào từng convolution layer ở phía trước. Từ đó ta có thể trích xuất được đặc điểm và nhận biết được các vật thể có kích thước nhỏ hơn.

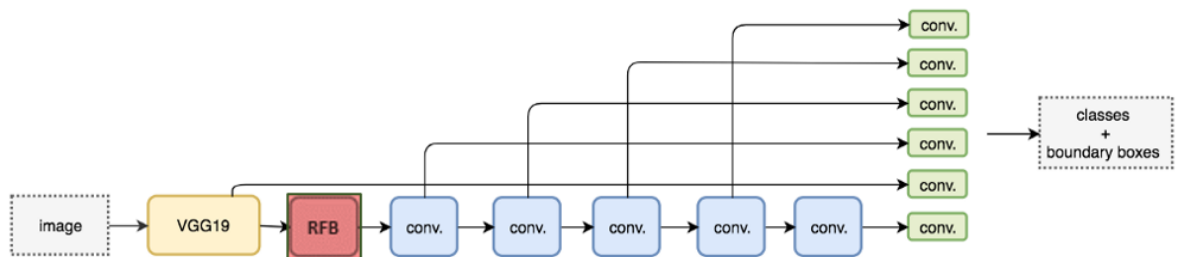


Hình 25. Sơ đồ hoạt động của mô hình Pytorch SSD



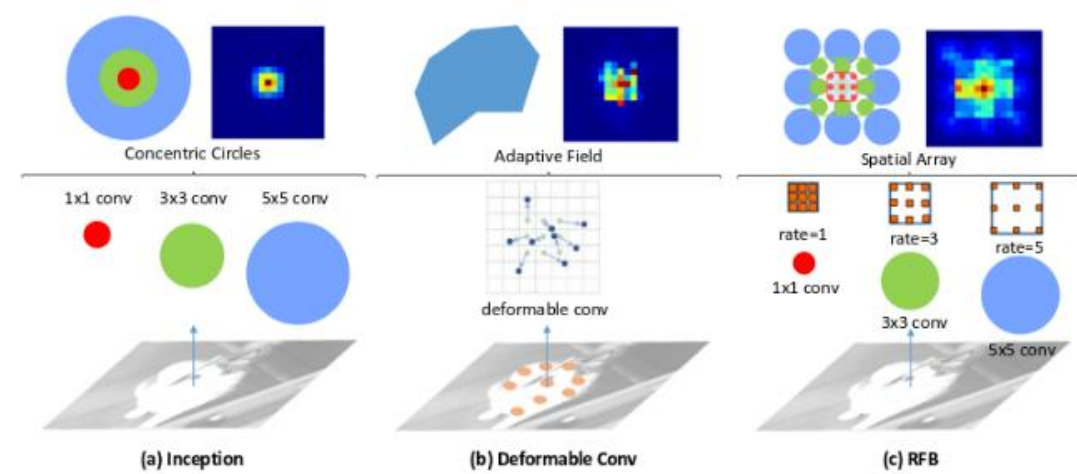
Hình 26. Sơ đồ chi tiết của mô hình Pytorch SSD

Để tăng khả năng nhận diện của mô hình, ta có thể thêm vào một module RFB (Receptive Fields Block) trước khi đi qua các lớp tích chập trích xuất đặc trưng.



Hình 27. Sự can thiệp của module RFB vào Pytorch SSD

Module RFB được thiết kế dựa trên trường cảm thụ (Receptive Fields – RFs) có trong hệ thống thị giác của con người. RFB sử dụng tích chập dạng hờ có tác dụng làm tăng tầm nhìn bao quát của lớp tích chập. Việc này giúp cho việc trích xuất các đặc trưng từ hình ảnh trở nên hiệu quả hơn so với mô hình ban đầu mà không ảnh hưởng quá nhiều đến tốc độ xử lý.



Hình 28. Mô phỏng giải thích trường cảm thụ RFB

Mô hình được huấn luyện bằng dữ liệu WIDER FACE training set bao gồm 16078 ảnh cùng với các nhãn gương mặt được xác định sẵn cung cấp bởi RetinaFace. Độ chính xác được đánh giá trên tập WIDER FACE val dataset cho 2 mô hình pretrain với input đầu vào là 320x240 và 640x480

Bảng 7. So sánh giữa 2 phiên bản với bộ dữ liệu WiderFace

Model	Easy Set	Medium Set	Hard Set
Version-RFB-320	0.787	0.698	0.438
Version-RFB-640	0.855	0.822	0.579

Với input đầu vào 640x480 có độ phân giải cao hơn nên sẽ cho ra kết quả phát hiện gương mặt tốt hơn so với input 320x240. Ta sẽ sử dụng mô hình pretrain với input 640x480 cho chương trình. Lưu ý rằng mô hình này đã được chuyển đổi sang framework ONNX để triển khai mô hình dễ dàng hơn.

## 2.2. Viết chương trình phát hiện gương mặt

Để sử dụng được mô hình, ta cần phải thay đổi kích thước hình ảnh sang 640x480



```
import cv2
import numpy as np
video_capture = cv2.VideoCapture()
# ...
ret, frame = video_capture.read()
h, w, _ = frame.shape
# preprocess img acquired
img = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
img = cv2.resize(img, (640, 480))
img_mean = np.array([127, 127, 127])
img = (img - img_mean) / 128
img = np.transpose(img, [2, 0, 1])
img = np.expand_dims(img, axis=0)
img = img.astype(np.float32)
# ...
```

*Hình 29. Phần code hỗ trợ xử lý hình ảnh*

Sau khi quá trình tiền xử lý hình ảnh hoàn tất, ta cần phải chuẩn bị mô hình ONNX và tạo một ONNX session cho quá trình dự đoán này.

```

import onnx
import onnxruntime as ort
from onnx_tf.backend import prepare
from caffe2.python.onnx import backend

# load the model, create runtime session & get input variable name
onnx_path = 'models/onnx/version-RFB-640.onnx'
predictor = onnx.load(onnx_path)
predictor = backend.prepare(predictor, device="CPU") # default CPU
ort_session = ort.InferenceSession(onnx_path)
input_name = ort_session.get_inputs()[0].name

```

Hình 30. Bắt đầu quá trình sử dụng ONNX

```

confidences, boxes = ort_session.run(None, {input_name: image})

```

Hình 31. Câu lệnh dùng để xác định khuôn mặt

Biến *confidences* sẽ chứa một list cặp độ tin cậy cho mỗi khung dự đoán trong biến *boxes*. Giá trị thứ nhất và thứ 2 trong cặp độ tin cậy lần lượt thể hiện xác suất chứa background và mặt người. Vì giá trị *boxes* chứa tất cả khung dự đoán, chúng ta chỉ cần lấy khung nào có xác suất cao nhất mà có chứa mặt người và loại bỏ đi các khung trùng nhau dựa trên thuật toán Jaccard Index (hay còn được gọi là IOU - Intersection of Union). Sau khi đã lọc các khung được dự đoán, ta có thể vẽ chúng lại vào ảnh

### 2.3. So sánh với các mô hình phát hiện gương mặt khác

Tiến hành đánh giá và so sánh tập dữ liệu với 4 model nhận diện khuôn mặt khác nhau (VGG-face, FaceNet, DeepFace, OpenFace). Tập dữ liệu bao gồm 6 sinh viên, mỗi sinh viên chứa 10 ảnh.

```

[ ] #Positives

positives = []

for key, values in identities.items():

    #print(key)
    for i in range(0, len(values)-1):
        for j in range(i+1, len(values)):
            #print(values[i], " and ", values[j])
            positive = []
            positive.append(values[i])
            positive.append(values[j])
            positives.append(positive)

positives = pd.DataFrame(positives, columns = ["file_x", "file_y"])
positives["decision"] = "Yes"

print(positives.shape)

```

*Hình 32. Lọc các cặp dữ liệu giống nhau*

Hàm nhận diện các dữ liệu giống nhau, chạy hàm trên ta được các cặp dữ liệu giống nhau ví dụ như:

	file_x	file_y	decision
0	1812736_0_0.jpg	1812736_25_0.jpg	Yes
1	1812736_0_0.jpg	1812736_60_0.jpg	Yes
2	1812736_0_0.jpg	1812736_85_0.jpg	Yes
3	1812736_0_0.jpg	1812736_220_0.jpg	Yes
4	1812736_25_0.jpg	1812736_60_0.jpg	Yes
5	1812736_25_0.jpg	1812736_85_0.jpg	Yes
6	1812736_25_0.jpg	1812736_220_0.jpg	Yes
7	1812736_60_0.jpg	1812736_85_0.jpg	Yes
8	1812736_60_0.jpg	1812736_220_0.jpg	Yes
9	1812736_85_0.jpg	1812736_220_0.jpg	Yes
10	1812795_0_0.jpg	1812795_30_0.jpg	Yes
11	1812795_0_0.jpg	1812795_55_0.jpg	Yes
12	1812795_0_0.jpg	1812795_160_0.jpg	Yes
13	1812795_0_0.jpg	1812795_80_0.jpg	Yes

*Hình 33. Kết quả các cặp dữ liệu giống nhau*

Tương tự ta có:

```

#Negatives

samples_list = list(identities.values())

negatives = []

for i in range(0, len(identities) - 1):
    for j in range(i+1, len(identities)):
        #print(samples_list[i], " vs ", samples_list[j])
        cross_product = itertools.product(samples_list[i], samples_list[j])
        cross_product = list(cross_product)
        #print(cross_product)

        for cross_sample in cross_product:
            #print(cross_sample[0], " vs ", cross_sample[1])
            negative = []
            negative.append(cross_sample[0])
            negative.append(cross_sample[1])
            negatives.append(negative)

negatives = pd.DataFrame(negatives, columns = ["file_x", "file_y"])
negatives["decision"] = "No"

negatives = negatives.sample(positives.shape[0])

print(negatives.shape)
print(negatives)

```

*Hình 34. Lọc các cặp dữ liệu khác nhau*

Hàm nhận diện các dữ liệu khác nhau, chạy hàm trên ta được các cặp dữ liệu khác nhau ví dụ như:

	file_x	file_y	decision
42	1812736_85_0.jpg	1812814_80_0.jpg	No
173	1812795_80_0.jpg	1813857_130_0.jpg	No
244	1812814_120_0.jpg	1813857_285_0.jpg	No
159	1812795_30_0.jpg	1813857_285_0.jpg	No
226	1812814_0_0.jpg	1813857_30_0.jpg	No
171	1812795_80_0.jpg	1813857_30_0.jpg	No
169	1812795_160_0.jpg	1813857_285_0.jpg	No
334	1813857_30_0.jpg	NguyenTrongHieu_0_17.jpg	No
287	1812814_80_0.jpg	NguyenTrongHieu_0_15.jpg	No
195	1812795_80_0.jpg	NguyenThiHa_0_4.jpg	No
346	1813857_285_0.jpg	NguyenTrongHieu_0_0.jpg	No
265	1812814_120_0.jpg	NguyenThiHa_0_4.jpg	No
123	1812736_220_0.jpg	NguyenTrongHieu_0_11.jpg	No
209	1812795_30_0.jpg	NguyenTrongHieu_0_17.jpg	No
166	1812795_160_0.jpg	1813857_30_0.jpg	No
335	1813857_75_0.jpg	NguyenTrongHieu_0_19.jpg	No
37	1812736_60_0.jpg	1812814_80_0.jpg	No
8	1812736_25_0.jpg	1812795_160_0.jpg	No
4	1812736_0_0.jpg	1812795_80_0.jpg	No
48	1812736_220_0.jpg	1812814_120_0.jpg	No
344	1813857_130_0.jpg	NguyenTrongHieu_0_17.jpg	No
13	1812736_60_0.jpg	1812795_160_0.jpg	No
20	1812736_220_0.jpg	1812795_0_0.jpg	No
110	1812736_60_0.jpg	NguyenTrongHieu_0_19.jpg	No
36	1812736_60_0.jpg	1812814_40_0.jpg	No
272	1812814_225_0.jpg	NguyenThiHa_0_8.jpg	No
260	1812814_80_0.jpg	NguyenThiHa_0_4.jpg	No
374	NguyenThiHa_0_16.jpg	NguyenTrongHieu_0_17.jpg	No
91	1812736_85_0.jpg	NguyenThiHa_0_20.jpg	No
127	1812795_0_0.jpg	1812814_80_0.jpg	No

Hình 35. Kết quả các cặp dữ liệu khác nhau

Nhập các kết quả giống nhau và khác nhau lại để đánh giá

```
[ ] #Merge positive and negative ones

df = pd.concat([positives, negatives]).reset_index(drop = True)

print(df.decision.value_counts())

df.file_x = "dataset/"+df.file_x
df.file_y = "dataset/"+df.file_y
```

Hình 36. Gộp kết quả các cặp dữ liệu giống và khác nhau

Gọi thư viện Deepface các mô hình VGG-face, FaceNet, DeepFace, OpenFace vào để tiến hành đánh giá và so sánh

```

#DeepFace

from deepface import DeepFace
from deepface.basemodels import VGGFace, OpenFace, Facenet, FbDeepFace

instances = df[["file_x", "file_y"]].values.tolist()

models = ['VGG-Face', 'Facenet', 'OpenFace', 'DeepFace']
metrics = ['cosine', 'euclidean_l2']

if True:

    pretrained_models = {}

    pretrained_models["VGG-Face"] = VGGFace.loadModel()
    print("VGG-Face loaded")
    pretrained_models["Facenet"] = Facenet.loadModel()
    print("Facenet loaded")
    pretrained_models["OpenFace"] = OpenFace.loadModel()
    print("OpenFace loaded")
    pretrained_models["DeepFace"] = FbDeepFace.loadModel()
    print("FbDeepFace loaded")

    for model in models:
        for metric in metrics:

            resp_obj = DeepFace.verify(instances
                                      , model_name = model
                                      , model = pretrained_models[model]
                                      , distance_metric = metric
                                      , enforce_detection = False)

            distances = []

            for i in range(0, len(instances)):
                distance = round(resp_obj["pair_%s" % (i+1)]["distance"], 4)
                distances.append(distance)

            df['%s_%s' % (model, metric)] = distances

    df.to_csv("face-recognition-pivot.csv", index = False)
else:
    df = pd.read_csv("face-recognition-pivot.csv")

df_raw = df.copy()

```

Hình 37. Sử dụng thư viện DeepFace để gọi các mô hình cho huấn luyện

Bắt đầu vẽ biểu đồ xuất kết quả phân bố giữa các mô hình sử dụng cosine và euclidean\_l2

```

fig = plt.figure(figsize=(15, 15))

figure_idx = 1
for model in models:
    for metric in metrics:

        feature = '%s_%s' % (model, metric)

        ax1 = fig.add_subplot(len(models) * len(metrics), len(metrics), figure_idx)

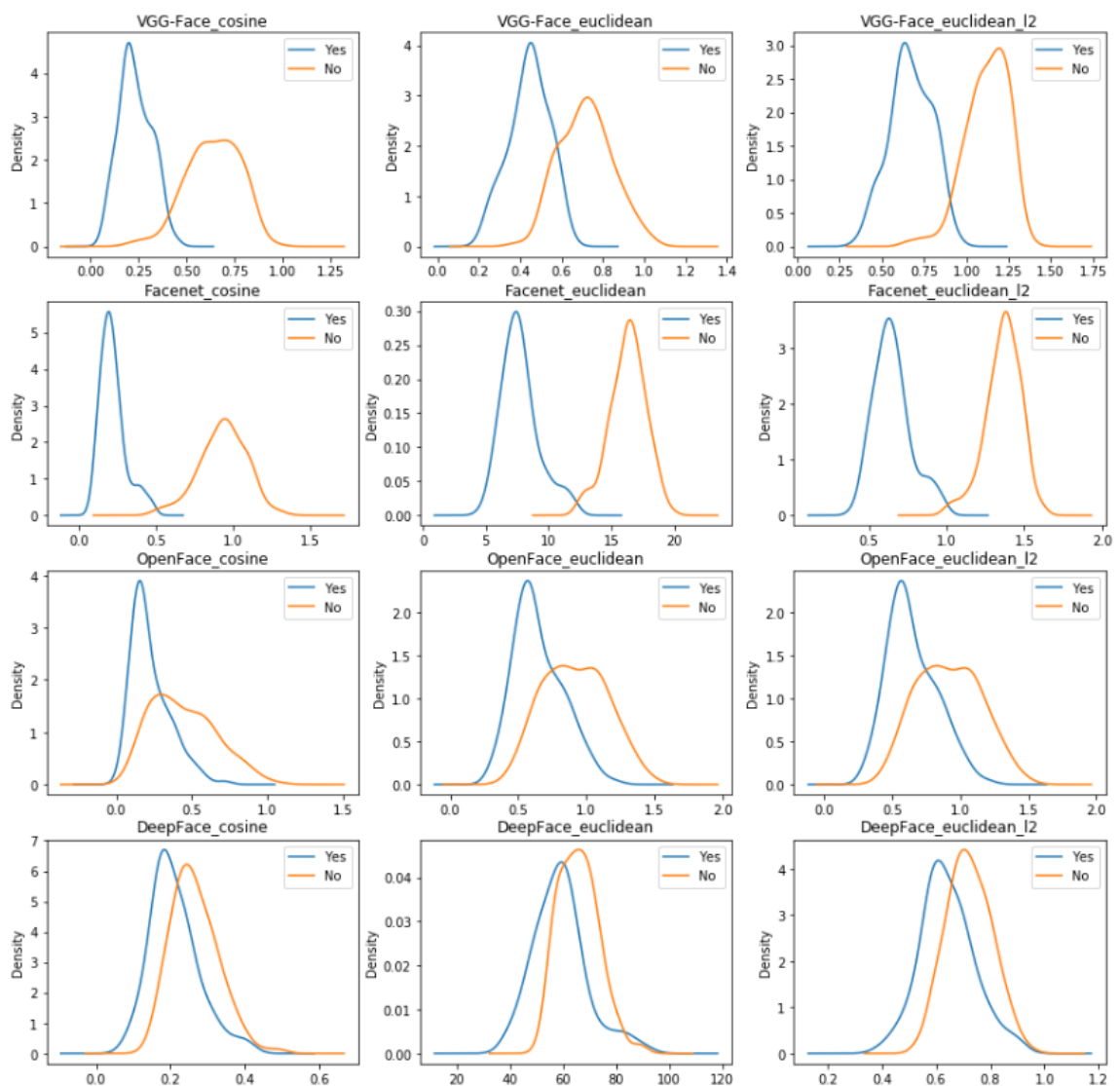
        df[df.decision == "Yes"][feature].plot(kind='kde', title = feature, label = 'Yes', legend = True)
        df[df.decision == "No"][feature].plot(kind='kde', title = feature, label = 'No', legend = True)

        figure_idx = figure_idx + 1

plt.show()

```

Hình 38. Vẽ đồ thị từ kết quả



Hình 39. Đồ thị của kết quả



Dựa vào biểu đồ kết quả trên, ta có thể thấy được mật độ phân bố của mô hình FaceNet vượt trội hẳn so với các mô hình khác. Do đó, nhóm quyết định sử dụng model FaceNet cho việc nhận diện khuôn mặt của chương trình.

### CHƯƠNG 3. NHẬN DIỆN GƯƠNG MẶT

Tương tự như phát hiện gương mặt, hiện nay có rất nhiều thuật toán và mô hình cho việc nhận diện gương mặt. Để có thể nhận diện tốt gương mặt, ta cần mô hình trích xuất hiệu quả được các điểm đặc trưng trên gương mặt cho nhiều người khác nhau. Đồng thời cũng nên kèm theo các quá trình tiền xử lý để cải thiện khả năng nhận diện gương mặt chính xác của mô hình.

#### 3.1. Căn chỉnh thẳng hàng gương mặt

Google công bố nghiên cứu rằng việc căn chỉnh thẳng hàng lại gương mặt giúp tăng độ chính xác mô hình FaceNet của họ từ 98.87% lên đến 99.63%. Có thể thấy việc căn chỉnh gương mặt giúp nâng cao khả năng nhận diện gương mặt cho mô hình.

Ta có thể căn chỉnh lại gương mặt thẳng hàng dựa trên các điểm facial landmark. Việc gương mặt được căn chỉnh giúp cho dữ liệu đầu vào trở nên nhất quán và giúp cho mô hình dễ dàng nhận diện và phân biệt gương mặt hiệu quả hơn.



*Hình 40. Căn chỉnh gương mặt theo mốc*

Để căn chỉnh thẳng hàng, ta dùng hai thư viện dlib cùng với imutils. Thư viện dlib có khả năng nhận biết đến 62 facial landmark trên một gương mặt. Tuy vậy ta chỉ cần nhận diện 5 điểm trên gương mặt bao gồm 2 điểm trên mắt trái và mắt phải, 1 điểm trước mũi là có thể căn chỉnh thẳng hàng gương mặt.

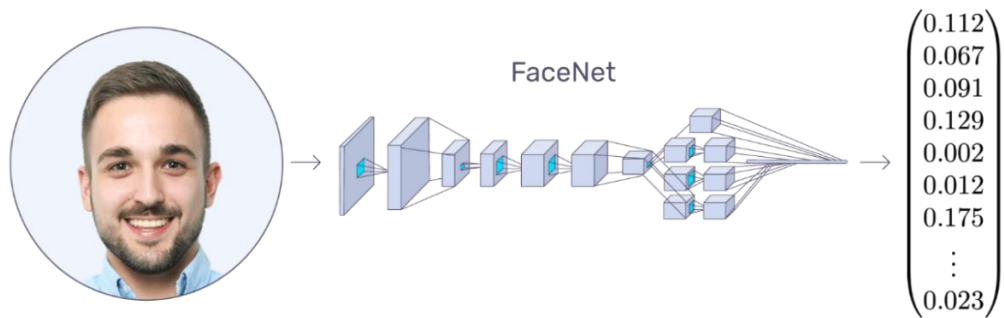
Thông số của *desiredLeftEye* thường sẽ nằm trong khoảng từ 0.2 đến 0.4. Thông số này nếu nhỏ dần thì gương mặt căn chỉnh sẽ cho ra to hơn.

Tiếp theo ta sẽ căn chỉnh từng vị trí gương mặt đã được phát hiện nằm trong biến *boxes*. Để phù hợp với input của model FaceNet, ta cần đổi kích thước các gương mặt thành 160x160. Sau đó ta sẽ lưu lại các gương mặt được căn chỉnh thẳng hàng này vào trong 1 list mới.

## 3.2. Mô hình FaceNet

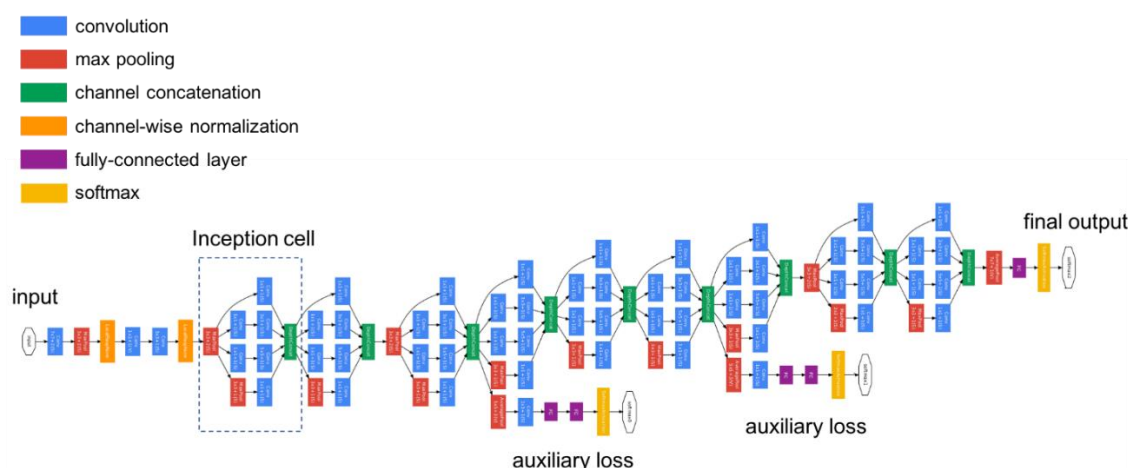
### 3.2.1. Kiến trúc mô hình

FaceNet là một mạng lưới thần kinh học sâu được dùng để trích xuất các điểm đặc trưng của gương mặt người có trong ảnh. Mô hình được công bố vào năm 2015 bởi các nhà nghiên cứu trong Google.



Hình 41. Tạo Vector từ gương mặt

FaceNet hoạt động bằng cách lấy hình ảnh từ gương mặt người làm input và sẽ trả ra một vector output có 128 giá trị số đại diện cho những nét đặc trưng có trên gương mặt. Trong máy học, vector này được gọi là vector nhúng (embedding vector). Lý do các vector được gọi như vậy là vì tất cả các thông tin quan trọng trên gương mặt được “nhúng” vào vector này. Theo lý thuyết, vector nhúng của các gương mặt thuộc cùng một người sẽ tương đương với nhau.



Hình 42. Mô hình Inception resnet v1

FaceNet được xây dựng trên mô hình Inception Resnet V1. Inception có điểm đặc trưng là có thêm 2 output phụ. Hai output phụ này không ảnh hưởng quá nhiều tới chất lượng của mạng lưới trong khi huấn luyện những epoch đầu. Chúng chỉ giúp việc huấn luyện mô hình diễn ra nhanh hơn bằng cách tối ưu những layer đầu dựa trên các output phụ (trong những epoch đầu). Có thể hiểu một cách khác là trong những epoch đầu, các layer càng về cuối sẽ chưa được tối ưu ngay. Sau một thời gian tối ưu các layer đầu rồi mới tối ưu các layer tiếp theo dựa trên final output. Việc này cải thiện khả năng tính toán và tốc độ huấn luyện lên nhiều lần.

### 3.2.2. Mô hình pretrain

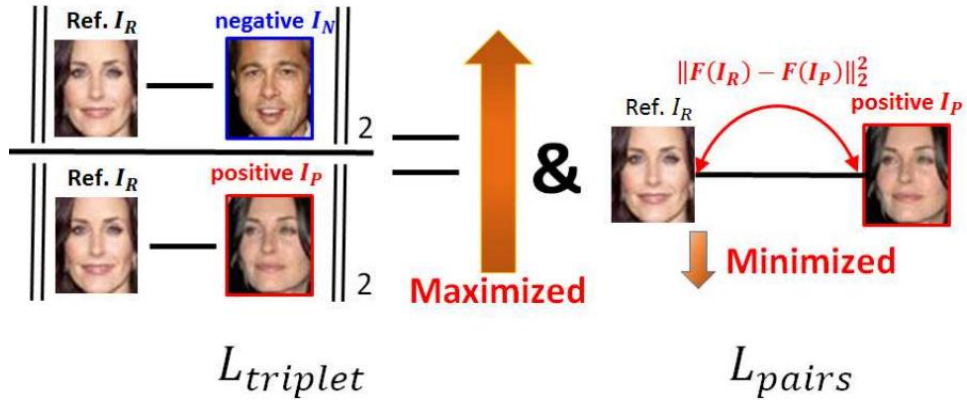
Hiện nay có 2 mô hình pretrain của FaceNet được train từ CASIA-WebFace dataset và VGGFace2 dataset. Dataset CASIA-WebFace (2014) chứa 494414 hình ảnh với 10575 nhân dạng (trung bình khoảng 46.8 ảnh cho một cá thể). Trong khi đó dataset của VGGFace2 (2018) chứa 3,31 triệu hình ảnh với 9131 nhân dạng (trung bình khoảng 362,6 ảnh cho một cá thể) từ nhiều châu lục khác nhau. Độ chính xác từ 2 mô hình pretrain được xác định trên LFW val set:

Bảng 8. Đánh giá độ chính xác giữa hai mô hình như dưới

LFW accuracy	Training dataset
0.9905	CASIA-WebFace
0.9965	VGGFace2

Có thể tăng kết quả dự đoán của mô hình nhận diện gương mặt bằng cách sử dụng thêm hàm loss function như Triplet Loss và Pairwise Loss trong khi huấn

luyện mô hình. Từ đó giúp mô hình đưa các vector nhúng cho gương mặt hiệu quả hơn từ đó tăng khả năng nhận diện gương mặt chính xác cho mô hình.



Hình 43. Sử dụng hàm loss để tăng độ nhận diện

Hàm Triplet Loss là một hàm loss function sử dụng một bộ 3 ảnh để huấn luyện mô hình. Hàm được định nghĩa như sau:

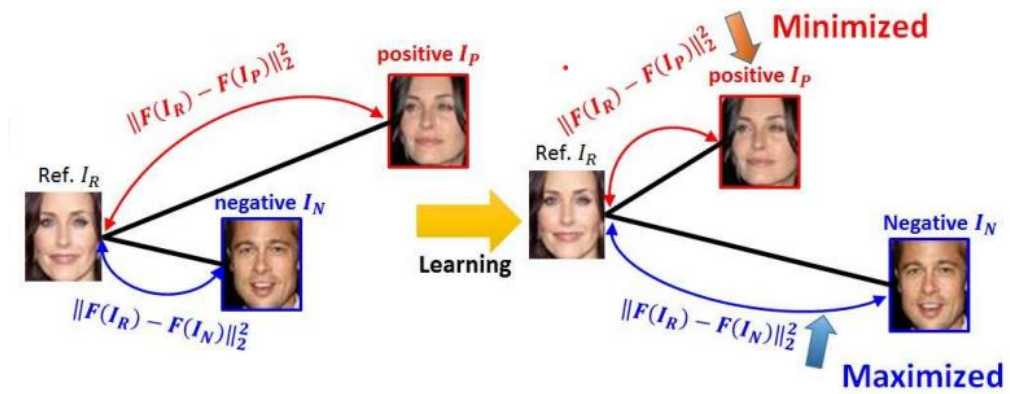
$$L_{triplet} = \sum_{\forall T} \max \left( 0, 1 - \frac{\|F(I_R) - F(I_N)\|_2}{\|F(I_R) - F(I_P)\|_2 + m} \right)$$

Với  $F(I_R)$ ,  $F(I_P)$ , và  $F(I_N)$  lần lượt là các vector nhúng của các ảnh tham chiếu ( $I_R$ ), positive ( $I_P$ ) và negative ( $I_N$ ). Từ các vector nhúng có thể tính được khoảng cách Euclidean giữa cặp negative  $\|F(I_R) - F(I_N)\|_2$  và cặp positive  $\|F(I_R) - F(I_P)\|_2$ . Giá trị  $L_{triplet}$  trong quá trình huấn luyện được tối ưu sao cho tỉ lệ giữa hai khoảng cách này lớn nhất. Đồng nghĩa với việc khoảng cách giữa cặp positive sẽ giảm đi, trong khi khoảng cách giữa cặp negative sẽ được tăng lên.

Tuy nhiên sau khi huấn luyện bằng hàm Triplet Loss, mặc dù khoảng cách giữa các cặp dữ liệu được nằm trong một khoảng tỉ lệ giá trị nhất định, giá trị khoảng cách tuyệt đối giữa chúng lại không hề đồng nhất. Vì vậy ta cần dùng thêm hàm Pairwise Loss để tối thiểu toàn bộ khoảng cách giữa các cặp positive đã được chọn trong hàm Triplet Loss  $T$ .

$$L_{pairs} = \sum_{(I_R, I_P) \in T} \|F(I_R) - F(I_P)\|_2^2$$

Quá trình huấn luyện sẽ tối thiểu khoảng cách tuyệt đối giữa các cặp vector positive này. Từ đó giới hạn lại được khoảng cách tuyệt đối giữa chúng. Hình phía dưới mô tả sự thay đổi khoảng cách giữa các cặp dữ liệu sau khi sử dụng hai hàm loss function trong quá trình huấn luyện mô hình.



Hình 44. Thuật toán Triplet loss mô phỏng quá trình nhận diện

### 3.2.3. Sử dụng mô hình để nhận diện

#### 3.2.3.1. Trích xuất đặc trưng để nhận diện gương mặt

Để chương trình có thể nhận diện được gương mặt hiệu quả nhất, ta cần có một cụm vector nhúng của gương mặt ở nhiều vị trí góc độ khác nhau bằng cách quay lại video. Để tăng khả năng nhận diện gương mặt chính xác của mô hình, ta có thể căn chỉnh thẳng hàng lại các ảnh như chương 3.1.

Sau khi tiền xử lý xong gương mặt, tiếp theo ta sẽ mở mô hình bằng pretrain Facenet bằng TensorFlow

Sau đó ta khai báo input cho mạng lưới, chuyển đổi ảnh thành các vector nhúng và lưu vào file pickle:

```

import pickle
# ...

# ...

phase_train_placeholder = tf.get_default_graph().
    get_tensor_by_name("phase_train:0")

# calc face embeddings

feed_dict = { images_placeholder: images,
               phase_train_placeholder: False }

embeds = sess.run(embeddings, feed_dict=feed_dict)

print("saving embeddings")

with open("embeddings/embeddings.pkl", "wb") as f:
    pickle.dump((embeds, names), f)

```

*Hình 45. Đưa mô hình đã huấn luyện vào file pickle*

### 3.2.3.2. Nhận diện gương mặt

Để nhận diện được gương mặt, ta cần trích xuất các đặc trưng của gương mặt thành một vector nhúng. Từ vector nhúng đó ta có thể nhận diện danh tính gương mặt chủ yếu thông qua hai cách dưới đây:

- Dùng các thuật toán phân loại như k-NN, SVM hoặc Random Forest.
- So sánh khoảng cách giữa hai vector nhúng.

Sử dụng phương pháp so sánh khoảng cách sẽ tốn nhiều thời gian hơn vì cần phải thực hiện tính toán với tất cả các vector nhúng đã được xác định sẵn. Phương pháp sử dụng thuật toán phân loại sẽ tốn ít tài nguyên hơn do chỉ cần xác định vị trí của vector nhúng mới nằm ở trong cụm nào. Tuy vậy phương pháp phân loại cần phải xác định biên giới cho từng cụm vector nhúng sao cho các vector này ít nằm ở sai cụm càng tốt. Trên thực tế cần rất nhiều thời gian cũng như chất lượng đầu vào tốt để xác định được ranh giới này tối ưu và hiệu quả nhất. Vì vậy đối với trường hợp số lượng danh tính cần nhận diện với số lượng vừa và nhỏ, ta sẽ sử dụng phương pháp so sánh khoảng cách. Với trường hợp danh tính được lưu với số lượng lớn, ta sẽ sử dụng phương pháp phân loại để tăng tốc thời gian xử lý.

Trong chương trình ta sẽ sử dụng phương pháp so sánh khoảng cách. Hiện nay có hai phương trình Cosine hoặc Euclidean được dùng để so sánh khoảng cách giữa hai vector.

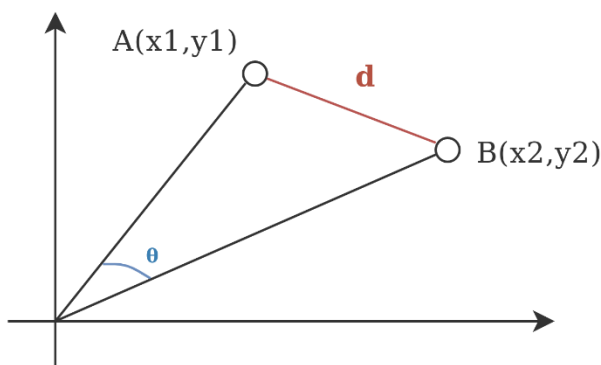
Phương trình Euclidean:

$$\sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

Phương trình Cosine:

$$\frac{x \cdot y}{\sqrt{x \cdot x} \sqrt{y \cdot y}}$$

Với  $x$  và  $y$  là các vector. Biểu đồ miêu tả giá trị trả về sau khi sử dụng phương trình Euclidean ( $d$ ) và phương trình Cosine ( $\theta$ ) để tính khoảng cách giữa hai vector. Trong khi phương trình Cosine sẽ cho kết quả là độ lớn tương đối (một góc giữa hai vector), phương trình Euclidean sẽ giống như một cái thước đo lại khoảng cách tuyệt đối giữa hai vector.



Hình 46. Khoảng cách tuyệt đối giữa hai Vector

Phương trình Cosine thường được sử dụng để đo khoảng cách khi mà độ lớn cũng như trọng số của các vector không quan trọng như trường hợp xử lý với dữ liệu text. Trong khi đó các đặc trưng trên gương mặt tác động ít nhiều lên trọng số của các vector nhúng. Vì vậy ta sẽ sử dụng phương trình Euclidean để xác định khoảng cách giữa hai vector.



Sau khi đã lưu các vector nhúng đã được gán nhãn vào file pickle, ta sẽ mở lại file

Sau đó ta sẽ trích xuất các đặc trưng thành vector nhúng cho từng gương mặt mới được phát hiện.

Tiếp theo ta sẽ tính khoảng cách giữa các vector. Sau đó lấy giá trị khoảng cách nhỏ nhất đem đi so sánh với một ngưỡng cho trước. Nếu khoảng cách này nhỏ hơn giá trị ngưỡng, gương mặt đó sẽ được xác định theo danh tính tương ứng với vector nhúng có khoảng cách nhỏ nhất. Ngược lại, nếu khoảng cách này lớn hơn giá trị ngưỡng, gương mặt lúc đó sẽ không được nhận diện danh tính. Để xác định được giá trị ngưỡng tối ưu (thường được gọi là fine tuning), ta cần phải thử hết từng giá trị trong một khoảng xác định cho đến khi thu được kết quả nhận diện chính xác nhất. Giá trị này có thể thay đổi theo từng hoàn cảnh và điều kiện môi trường khác nhau.

Cuối cùng hiển thị kết quả nhận diện lên từng gương mặt được phát hiện trong ảnh:

```
while True:
    ...
    for i in range(boxes.shape[0]):
        box = boxes[i, :]
        text = f"{predictions[i]}"
        x1, y1, x2, y2 = box
        cv2.rectangle(frame, (x1, y1), (x2, y2), (80,18,236), 2)
        # Draw a label with a name below the face
        cv2.rectangle(frame, (x1, y2 - 20), (x2, y2),
                       (80,18,236), cv2.FILLED)
        font = cv2.FONT_HERSHEY_DUPLEX
        cv2.putText(frame, text, (x1 + 6, y2 - 6), font,
                    0.3, (255, 255, 255), 1)
```

*Hình 47. Hiển thị kết quả gương mặt đã phát hiện*

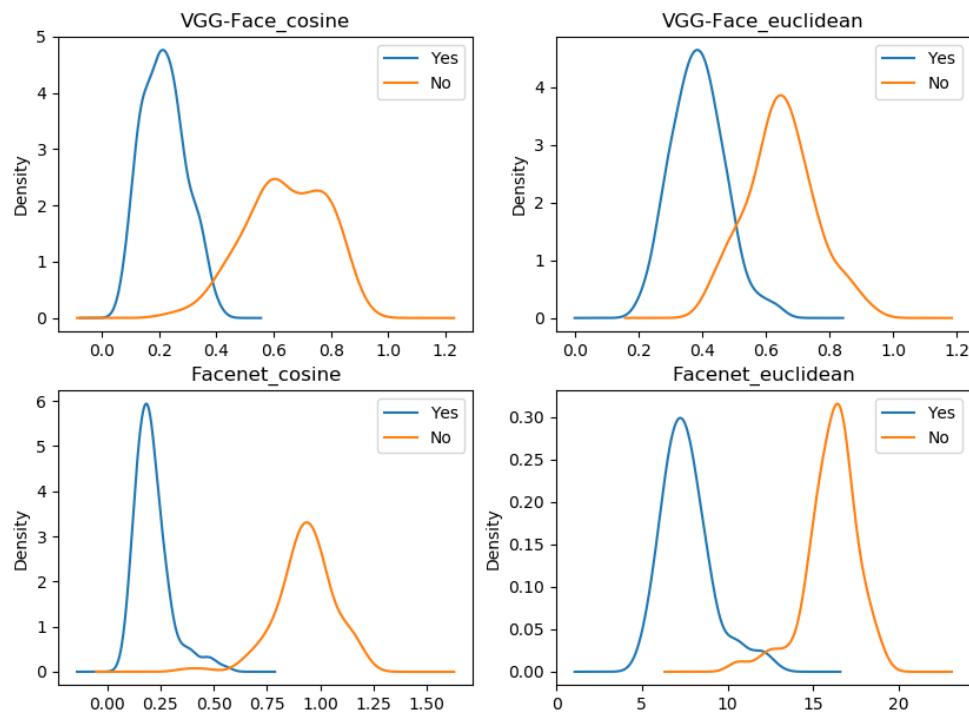


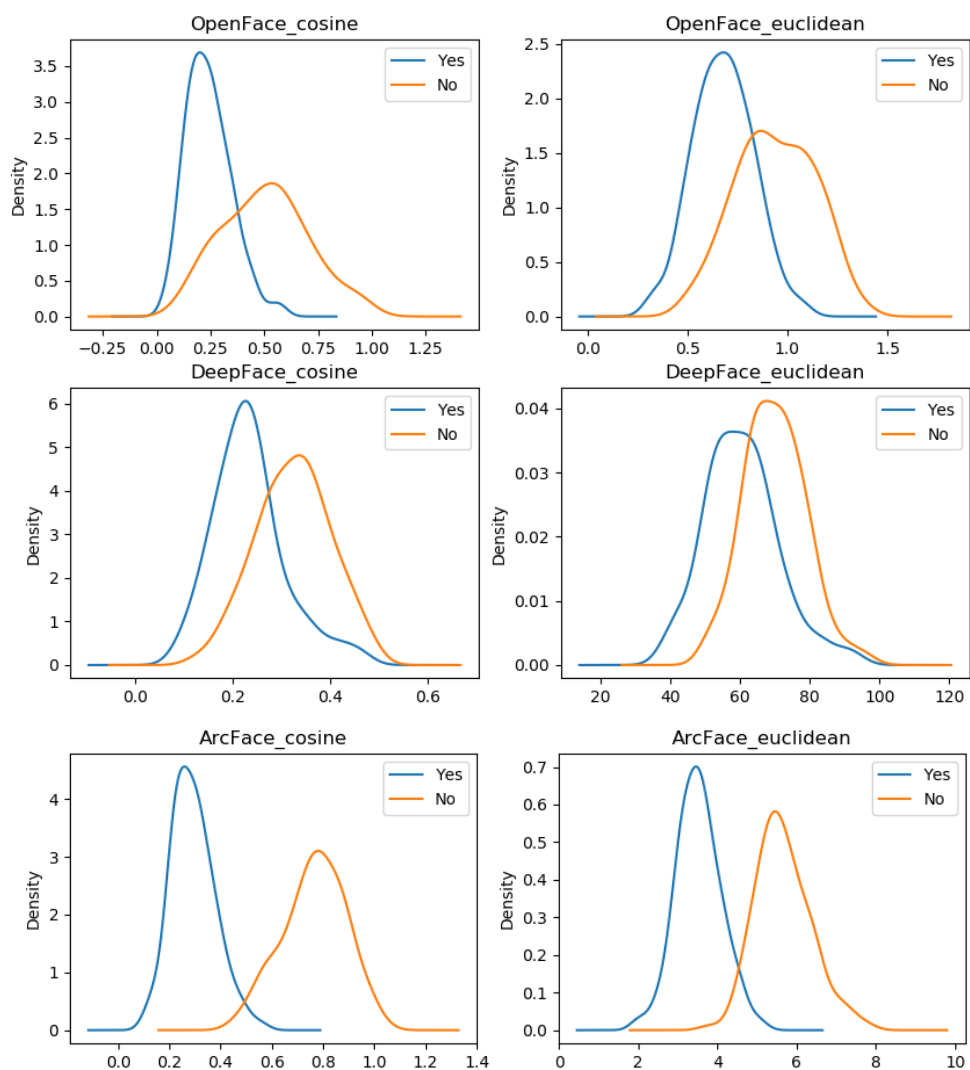
### 3.2.3.3. So sánh với các mô hình nhận diện gương mặt hiện nay

Để so sánh giữa độ hiệu quả giữa các mô hình, ta sẽ sử dụng một dataset ảnh chứa mặt người rồi ghép hai ảnh khác nhau thành một cặp theo hai trường hợp sau (mỗi trường hợp bao gồm 140 cặp):

- Hai hình ảnh thuộc cùng một người.
- Hai hình ảnh thuộc hai người khác nhau.

Sau đó ta tính toán khoảng cách cho tất cả các cặp trường hợp theo từng mô hình nhận diện gương mặt. Sau khi có kết quả ta sẽ vẽ biểu đồ KDE biểu diễn mật độ phân phối giá trị khoảng cách đối với 2 trường hợp ứng theo từng loại mô hình.





Hình 48. Đồ thị so sánh hiệu quả giữa các mô hình

Dựa vào đồ thị có thể thấy mô hình FaceNet có đường phân bố giá trị khoảng cách cho 2 trường hợp ít bị dính liền với nhau nhất. Điều này có nghĩa là sẽ ít xảy ra trường hợp nhận diện gương mặt hai người khác nhau thành giống nhau (False Negative) so với các mô hình khác.

Ngoài ra ta có thể sử dụng thuật toán C4.5 Decision Tree để xác định ngưỡng tối ưu từ các kết quả phía trên. Từ đó ta có thể đánh giá được độ chính xác cho từng mô hình dựa trên ngưỡng tối ưu này. Bảng dưới đây cho kết quả đánh giá của từng mô hình:

Bảng 9. So sánh hiệu quả giữa các mô hình

	<b>VGG-Face</b>	<b>FaceNet</b>	<b>OpenFace</b>
<b>Cosine</b>	Threshold: 0.43 Accuracy: 96.43 Precision: 93.33 Recall: 100.0 F1: 96.55	Threshold: 0.57 Accuracy: 99.29 Precision: 98.59 Recall: 100.0 F1: 99.29	Threshold: 0.37 Accuracy: 80.71 Precision: 76.22 Recall: 89.29 F1: 82.24
<b>Euclidean</b>	Threshold: 0.52 Accuracy: 91.07 Precision: 88.59 Recall: 94.29 F1: 91.35	Threshold: 11.84 Accuracy: 97.5 Precision: 97.84 Recall: 97.14 F1: 97.49	Threshold: 1.03 Accuracy: 67.5 Precision: 60.79 Recall: 98.57 F1: 75.2

	<b>Deep Face</b>	<b>ArcFace</b>
<b>Cosine</b>	Threshold: 0.28 Accuracy: 77.14 Precision: 75.0 Recall: 81.43 F1: 78.08	Threshold: 0.53 Accuracy: 97.86 Precision: 96.53 Recall: 99.29 F1: 97.89
<b>Euclidean</b>	Threshold: 54.52 Accuracy: 63.57 Precision: 89.58 Recall: 30.71 F1: 45.74	Threshold: 4.59 Accuracy: 96.07 Precision: 95.1 Recall: 97.14 F1: 96.11

Có thể thấy FaceNet cho độ chính xác cũng như F1 cao nhất so với các mô hình còn lại. ArcFace cũng cho độ chính xác xấp xỉ tương đương với mô hình FaceNet.

## CHƯƠNG 4. TRIỂN KHAI CHƯƠNG TRÌNH

Hệ thống nhận diện gương mặt có thể áp dụng trong nhiều trường hợp thực tiễn như camera giám sát, chấm công, điểm danh, eKYC, ... Trong bài báo cáo sẽ thiết kế một ứng dụng điểm danh đơn giản và triển khai chương trình lên trên website.

### 4.1 Công nghệ

Flask là một Web Framework rất nhẹ giúp người giúp các lập trình viên tạo ra một hệ thống web server dễ dàng và đơn giản trên python. Ngoài ra Flask còn có tính mở rộng và là một microframework không bao gồm ORM (Object Relational Manager) hoặc các tính năng tương tự.

Để hiển thị các dữ liệu bị thay đổi (dữ liệu động) trên website mà không cần phải tải lại trang (dynamic website), ta sử dụng Ajax để truyền thông tin truy vấn (query) từ client cho sever và Server-Sent Events (SSE) để sever truyền ngược lại dữ liệu liên tục cho client.

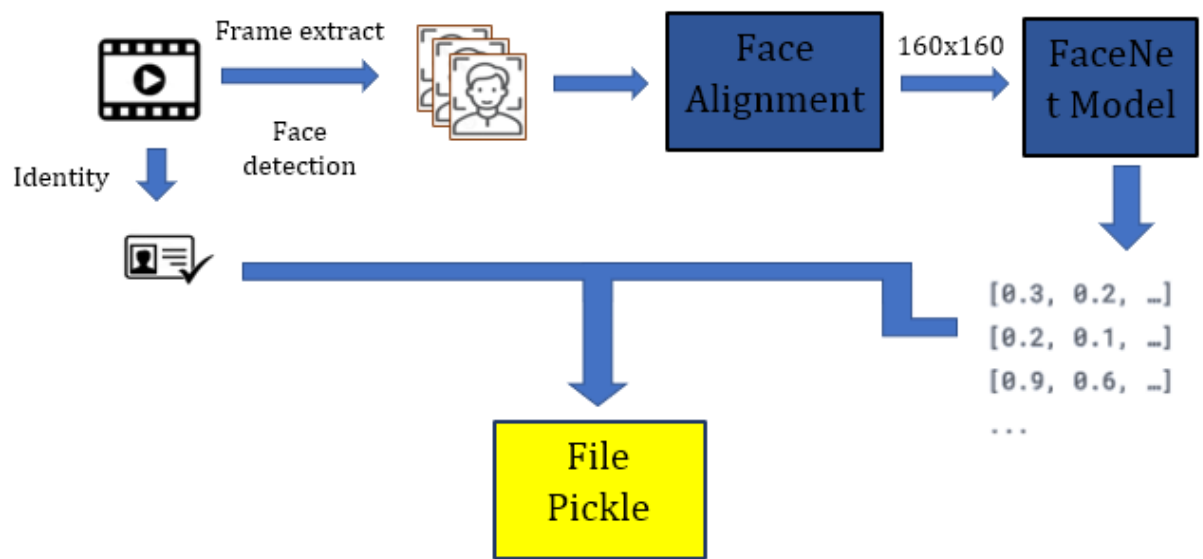
Ta sẽ xây dựng web app với 3 chức năng chính như sau:

- Quan sát trực tiếp kết quả xử lý của hệ thống nhận diện gương mặt
- Tính toán và lưu lại các vector nhúng cho từng gương mặt qua các video được lưu. Ngoài ra có thể dùng webcam để quay và lưu lại video cho gương mặt cần nhận diện.
- Xem database điểm danh theo ngày

### 4.2 Quy trình của chương trình:

#### 4.2.1 Thu thập dữ liệu các khuôn mặt để nhận diện

Việc thu thập dữ liệu để huấn luyện cho mô hình sẽ dựa vào trang Web xây dựng trước với đầu vào của dữ liệu sẽ có dạng là hình ảnh, chương trình sẽ tích hợp với camera để lấy dữ liệu hình ảnh trên và bắt đầu huấn luyện. Hình 49 là mô hình huấn luyện của chương trình:



Hình 49. Mô hình huấn luyện của chương trình

Hình 50 là giao diện chính của chương trình cho việc huấn luyện dữ liệu



Hình 50. Giao diện huấn luyện của chương trình

Như đã mô tả ở phần trên, dữ liệu đầu vào sẽ có dạng là hình ảnh, và hình ảnh này có thể cắt từ video, nếu dữ liệu đầu vào là một video thì chương trình sẽ thông qua một bước riêng, đó là tách các ảnh có hình khuôn mặt từ video đó và format lại theo dữ liệu ban đầu là hình ảnh

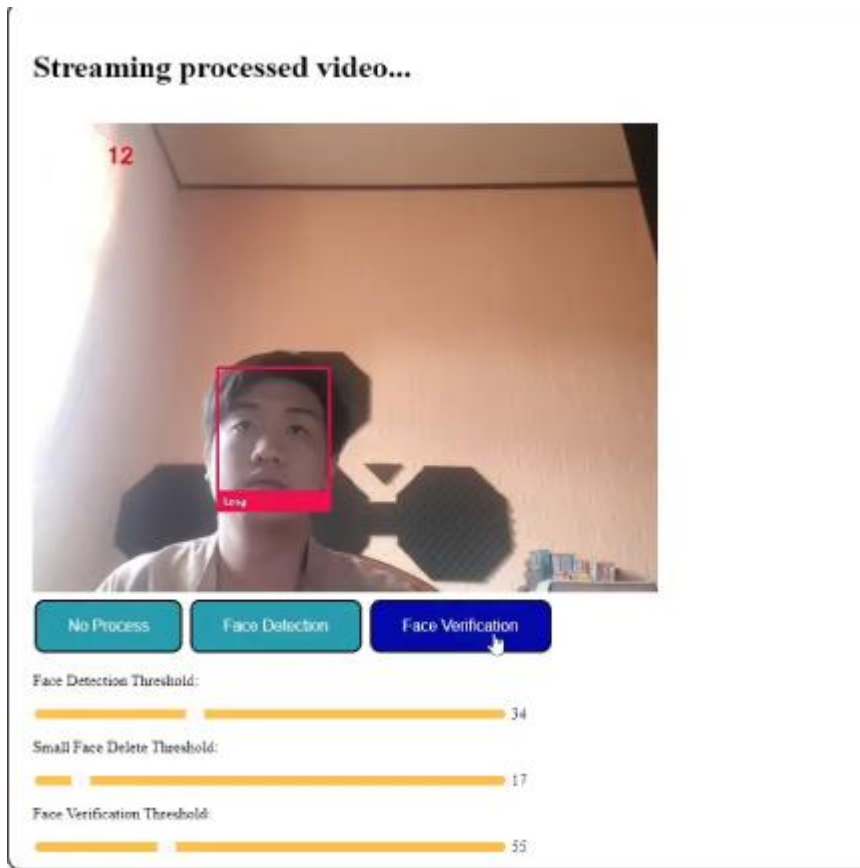
Sau quá trình tách cảnh từ video, ta tiến hành cắt khuôn mặt ra, căn chỉnh lại bằng cách sử dụng aligned face và cuối cùng là resize ảnh lại (160x160)

Tiếp đến ta lưu dữ liệu ảnh sau khi xử lý lại với tên lấy ở trường định danh của trang Web để bắt đầu huấn luyện:



*Hình 51. Dữ liệu kết quả thu được sau khi xử lý*

Cuối cùng ta tiến hành đọc file dữ liệu ảnh chuyển đổi theo quá trình mô tả ở trên sang embedding vertex đổ vào một mô hình pre train có sẵn của facenet ( 20180402-114759.pb ) sau đó lưu lại dữ liệu đó vào file mới (facemodel.pkl) thông qua pickle để phục vụ cho nhận diện và kết quả thu được như sau:

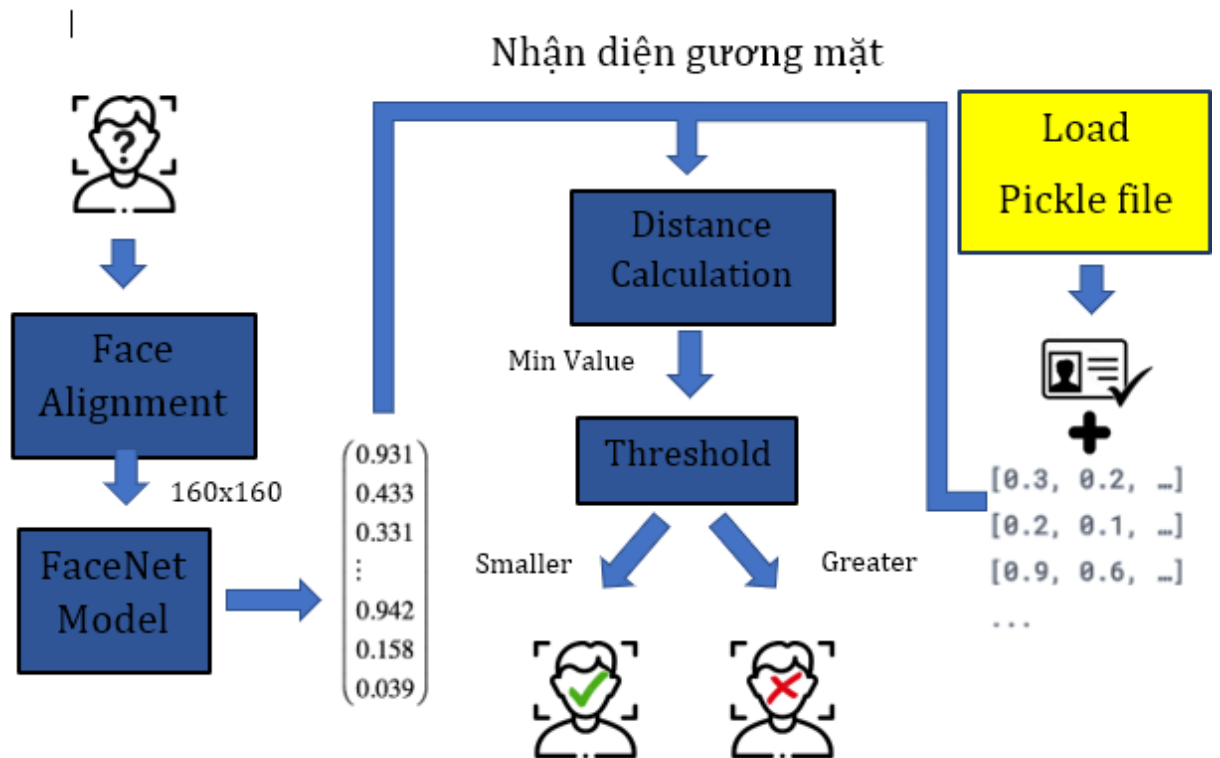


Hình 52. Giao diện nhận diện khuôn mặt của chương trình

#### 4.2.2 Xử lý phát hiện và nhận diện khuôn mặt trong thời gian thực

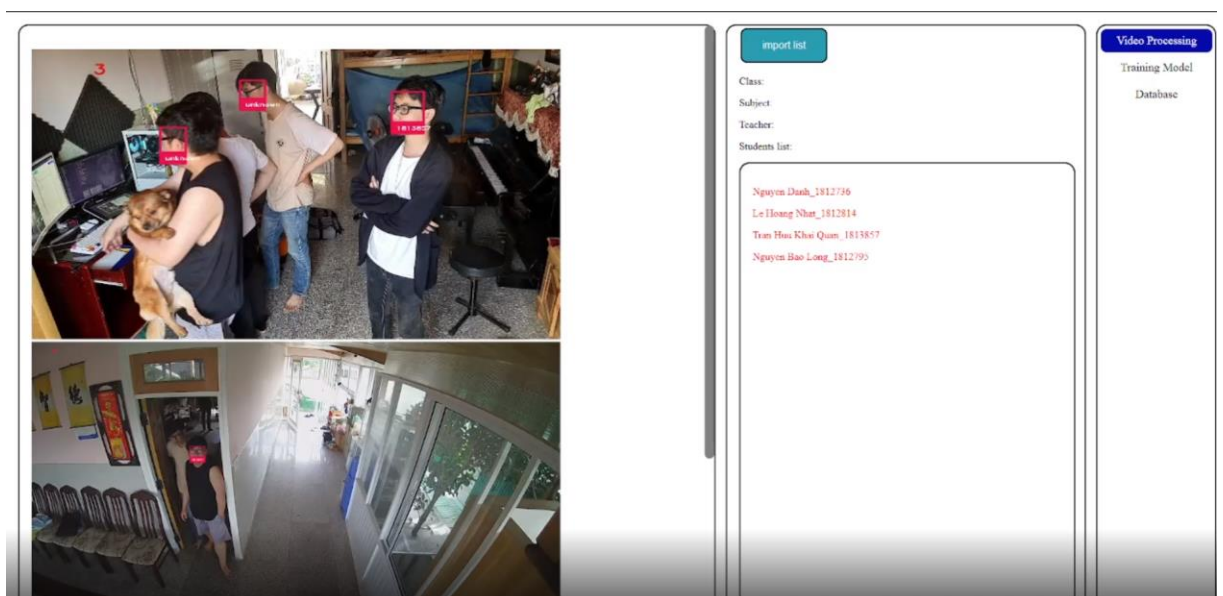
Tương tự như phần thu thập dữ liệu trên, ta cũng cần một đầu vào là input của camera nhưng lần này ta thêm vào những chức năng khác để phục vụ cho công tác điểm danh sinh viên như phần điều chỉnh thông số chung (các slider threshold) và phần đọc danh sách điểm danh để kiểm tra với mô hình sau:





Hình 53. Mô hình nhận diện gương mặt của chương trình

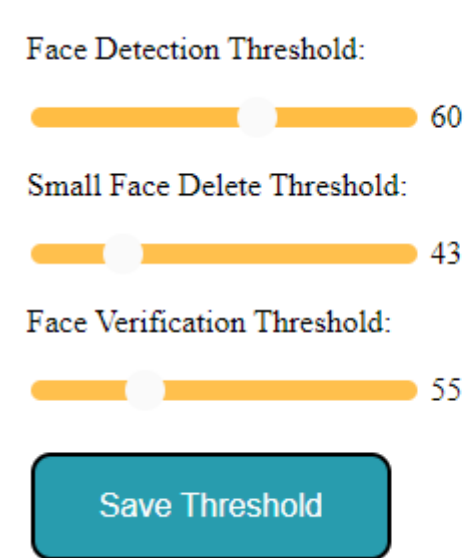
Giao diện của chương trình sau khi nhận diện được:



Hình 54. kết quả nhận diện theo thời gian thực

Ta vẫn xử lý và nhận diện thông tin khuôn mặt như đã mô tả ở phần trên thông qua Aligned, Facenet sau đó chuyển chúng sang dữ liệu Vector và tiến hành so sánh với các Embedded Vertex đã được lưu từ trước lấy từ file Pickel

Để phục vụ cho việc so sánh dữ liệu ta có các Slider Theshhold sau, tùy vào thông số của các silder này ta có thể thay đổi kết quả theo ý muốn cũng như độ lớn của các khuôn mặt cần nhận, độ chính xác của quá trình nhận diện.



Face Detection Threshold: 60

Small Face Delete Threshold: 43

Face Verification Threshold: 55

Save Threshold

*Hình 55. Giao diện tùy chỉnh biến số nhận diện*

Ta sử dụng thuật toán Triplet Loss so sánh độ tương đồng để đẩy xa hoặc thu gọn khoảng cách giữa các vector nhằm phân nhóm kết quả dựa vào các Threshold trên.

Tiếp đến ta đọc thông tin điểm danh từ file CSV đồng thời so sánh danh sách đó với các sinh viên có mặt trong danh sách nhận diện được.

Cuối cùng ta lưu thông tin danh sách đó vào một file CSV bao gồm thông tin và thời điểm nhận được của sinh viên đó theo từng ngày một dưới định dạng CSV

### 4.2.3 Xem database điểm danh theo ngày

Kèm theo các hàm truy vấn đơn giản như lọc file theo ngày, file của 5 ngày gần nhất ta cho phép người sử dụng chọn file cần truy vấn sau đó xuất kết quả đọc được lên màn hình

Name	Time
Long	20:38:21

Choose 5 recent files: 14/05/2022 ▼

or select a date: 05 / 14 / 2022 📅

Video Processing

Training Model

Database

Hình 56. Giao diện của truy xuất dữ liệu điểm danh sinh viên

## CHƯƠNG 5. KẾT LUẬN

Như vậy ta đã hoàn thành xây dựng xong hệ thống nhận diện gương mặt cùng với ứng dụng điểm danh cơ bản chạy trên nền tảng web. Mặc dù cấu trúc hệ thống nhận diện gương mặt thoát nhìn rất đơn giản, tuy nhiên để xây dựng được một hệ thống hoàn chỉnh cần phải bỏ ra rất nhiều công sức nghiên cứu lựa chọn mô hình phù hợp cũng như tối ưu tốc độ xử lý và độ chính xác cho chương trình. Tuy vậy hệ thống vẫn còn tồn tại một số hạn chế như sau:

- Không kiểm soát được số lượng vector nhúng cho từng nhân dạng để so sánh. Đồng nghĩa với việc kết quả nhận dạng sẽ thiên về người có nhiều vector nhúng hơn ở trong file pickle. Ta có thể giải quyết vấn đề này bằng cách tạo ra mô hình 3D cho từng người rồi tạo ra một bộ vector nhúng dựa trên mô hình 3D này. Ngoài ra việc này còn làm tối thiểu số lượng vector nhúng cho từng danh tính giúp tăng tốc độ xử lý nhận diện gương mặt cho hệ thống.
- Khi thêm gương mặt của một người mới hay cập nhật lại một gương mặt của người đã có sẵn, chương trình phải tính toán và lưu lại từ đầu danh sách các vector nhúng của toàn bộ gương mặt. Để tối ưu quá trình này ta sẽ lưu thông tin vào trong một dictionary với các key là tên nhân và value là list các vector nhúng ứng với tên nhân thay vì lưu thành 2 list song song. Khi đó chỉ cần thêm vào một cặp key value vào trong dictionary cho một người mới cần được nhận diện. Việc thay đổi lại hoặc xóa dữ liệu của gương mặt đã được lưu cũng trở nên dễ dàng hơn thông qua truy vấn các key.
- Trong trường hợp nhận diện với số lượng danh tính đã được lưu lớn, hệ thống sẽ tốn rất nhiều thời gian để so sánh và đưa ra kết quả nhận diện. Lúc này ta sử dụng đến thuật toán phân cụm như thuật toán tìm kiếm k-NN (k-nearest neighbors) hoặc thuật toán tìm kiếm lân cận xấp xỉ (a-NN).
- Mô hình pretrain của facenet còn khá nặng (~100 Mb) nên sẽ bị hạn chế triển khai trên một số nền tảng thiết bị.
- Việc nhận diện có thể bị giả mạo bằng hình ảnh hoặc video quay sẵn. Vấn đề này có thể giải quyết bằng cách sử dụng phương pháp challenge-response hoặc sử dụng camera chiều sâu (depth camera). Ngoài ra có thể kết hợp thêm với các công cụ nhận dạng sinh trắc khác để tăng tính bảo mật.

- Ngoài ra hệ thống nhận diện không được hiệu quả tốt trong các trường hợp ngoại lệ như đeo khẩu trang hay đeo kính. Vì vậy ta cần phải huấn luyện thêm cho mô hình nhận diện được trong các trường hợp đặc biệt này. Khi đã huấn luyện xong mô hình, để có thể áp dụng thực tế ta cần phải quay lại gương mặt ở trường hợp này và xuất ra các vector nhúng tương ứng để so sánh.

Hệ thống nhận diện gương mặt hiện tại vẫn còn rất nhiều tiềm năng phát triển cũng như mở rộng thêm, và nếu bỏ qua được các hạn chế nêu trên, chương trình có thể áp dụng rộng rãi vào trong đời sống của con người

## TÀI LIỆU THAM KHẢO

- [1] G. Hinton, “Deep Belief Nets,” *IEEE Trans. Neural Networks*, vol. 17, no. 6, pp. 1623–1629, 2006, doi: 10.1109/TNN.2006.880582.
- [2] L. Shi, Z. Li, and D. Song, “A Flower Auto-Recognition System Based on Deep Learning,” *IOP Conf. Ser. Earth Environ. Sci.*, vol. 234, no. 1, 2019, doi: 10.1088/1755-1315/234/1/012088.
- [3] J. Kim, R.-G. Huang, S. Jin, and K. Hong, “Mobile-Based Flower Recognition System,” *2009 Third Int. Symp. Intell. Inf. Technol. Appl.*, vol. 3, pp. 580–583, 2009.
- [4] T. Tiay, P. Benyaphaichit, and P. Riyamongkol, “Flower recognition system based on image processing,” in *2014 Third ICT International Student Project Conference (ICT-ISPC)*, 2014, pp. 99–102, doi: 10.1109/ICT-ISPC.2014.6923227.
- [5] A. Angelova and S. Zhu, “Efficient Object Detection and Segmentation for Fine-Grained Recognition,” in *Proceedings / CVPR, IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2013, pp. 811–818, doi: 10.1109/CVPR.2013.110.
- [6] G. Doran, “S.M.A.R.T-Way-Management-Review.pdf,” *Management Review*, vol. 70, no. 11, pp. 35–36, 1981, [Online]. Available: <https://community.mis.temple.edu/mis0855002fall2015/files/2015/10/S.M.A.R.T-Way-Management-Review.pdf>.
- [7] K. Chauhan, S. Jani, R. Dave, J. Bhatia, S. Tanwar, and M. Obaidat, “Automated Machine Learning: The New Wave of Machine Learning,” 2020, doi: 10.1109/ICIMIA48430.2020.9074859.
- [8] M. Awad and R. Khanna, *Efficient learning machines: Theories, concepts, and applications for engineers and system designers*, no. April, 2015.
- [9] R. K. Mitchell, B. R. Agle, and D. J. Wood, “Toward a Theory of Stakeholder Identification and Salience: Defining the Principle of Who and What Really

- Counts,” *Acad. Manag. Rev.*, vol. 22, no. 4, pp. 853–886, May 1997, doi: 10.2307/259247.
- [10] BBCNews, “Artificial intelligence: Google’s AlphaGo beats Go master Lee Sedol,” 2016. <https://www.bbc.com/news/technology-35785875>.
- [11] W. Vogt, “Nonlinear Regression,” *ncss.com*, no. 1, pp. 1–9, 2015, doi: 10.4135/9781412983907.n1291.
- [12] E. Ostertagová, “Modelling using polynomial regression,” *Procedia Eng.*, vol. 48, no. May, pp. 500–506, 2012, doi: 10.1016/j.proeng.2012.09.545.
- [13] S. Depart of Statistics, “Polynomial Regression Examples,” 2020, [Online]. Available: <https://online.stat.psu.edu/stat501/lesson/9/9.8>.
- [14] J. Brownlee, “Arithmetic, Geometric, and Harmonic Means for Machine Learning,” 2019. <https://machinelearningmastery.com/arithmetic-geometric-and-harmonic-means-for-machine-learning/>.
- [15] I. J. Goodfellow, N. Koenig, M. Muja, C. Pantofaru, A. Sorokin, and L. Takayama, “Help me help you: Interfaces for personal robots,” in *2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2010, pp. 187–188, doi: 10.1109/HRI.2010.5453203.
- [16] M. Nilsback and A. Zisserman, “Automated Flower Classification over a Large Number of Classes,” in *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, 2008, pp. 722–729, doi: 10.1109/ICVGIP.2008.47.
- [17] G. Punj and D. W. Stewart, “Cluster Analysis in Marketing Research: Review and Suggestions for Application,” *J. Mark. Res.*, vol. 20, no. 2, p. 134, 1983, doi: 10.2307/3151680.
- [18] J. de Oña, R. de Oña, and G. López, “Transit service quality analysis using cluster analysis and decision trees: a step forward to personalized marketing in public transportation,” vol. 43, pp. 725–747, 2016.
- [19] S. Marsland, *Machine Learning: An Algorithmic Perspective, Second Edition*.

Stephen Marsland, 2014.

- [20] G. Eknayan, “Adolphe Quetelet (1796–1874)—the average man and indices of obesity,” *Nephrol. Dial. Transplant.*, vol. 23, no. 1, pp. 47–51, Jan. 2008, doi: 10.1093/ndt/gfm517.
- [21] E. Bingham and H. Mannila, “Random Projection in Dimensionality Reduction: Applications to Image and Text Data,” in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2001, pp. 245–250, doi: 10.1145/502512.502546.
- [22] S. A. Bleha and M. S. Obaidat, “Dimensionality reduction and feature extraction applications in identifying computer users,” *IEEE Trans. Syst. Man. Cybern.*, vol. 21, no. 2, pp. 452–456, 1991, doi: 10.1109/21.87093.
- [23] I. Perfiljeva, “Dimensionality Reduction by Fuzzy Transforms with Applications to Mathematical Finance,” in *Studies in Computational Intelligence*, 2018, pp. 243–254.
- [24] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson, “Offline/realtime traffic classification using semi-supervised learning,” *Perform. Eval.*, vol. 64, no. 9, pp. 1194–1213, 2007, doi: <https://doi.org/10.1016/j.peva.2007.06.014>.
- [25] M. Liu, M. Zhou, T. Zhang, and N. Xiong, “Semi-supervised learning quantization algorithm with deep features for motor imagery EEG Recognition in smart healthcare application,” *Appl. Soft Comput.*, vol. 89, p. 106071, 2020, doi: <https://doi.org/10.1016/j.asoc.2020.106071>.
- [26] Y. Li, “Deep Reinforcement Learning: An Overview,” 2017.
- [27] K. Yan, “Differences between Normalization, Standardization and Regularization,” 2018. <https://maristie.com/2018/02/Normalization-Standardization-and-Regularization/>.
- [28] S. Gupta, “Deep learning performance breakthrough,” 2018. <https://www.ibm.com/blogs/systems/deep-learning-performance-breakthrough/>.
- [29] A. Wasicek, “Artificial Intelligence vs. Machine Learning vs. Deep Learning:



What's the Difference?," 2018. <https://www.sumologic.com/blog/machine-learning-deep-learning/>.

- [30] P. Đ. Thắng, "Các kiến trúc CNN hiện đại," 2020.  
<https://phamdinhhkhanh.github.io/2020/05/31/CNNHistory.html>.
- [31] M. Sandler, M. Zhu, A. Zhmoginov, and C. V. W. Ho, "MobileNetV2: Inverted Residuals and Linear Bottlenecks."