

**UBND TỈNH BÌNH DƯƠNG**  
**TRƯỜNG ĐẠI HỌC THỦ DẦU MỘT**

**TRẦN NGUYỄN THANH TUYỀN**

**HỆ THỐNG ĐIỂM DANH HỌC SINH TẠI TRƯỜNG  
PHỔ THÔNG SỬ DỤNG CÔNG NGHỆ NHẬN DẠNG  
KHUÔN MẶT**

**CHUYÊN NGÀNH: HỆ THỐNG THÔNG TIN**  
**MÃ SỐ: 8480104**

**LUẬN VĂN THẠC SĨ**

**BÌNH DƯƠNG - NĂM 2019**

**UBND TỈNH BÌNH DƯƠNG**  
**TRƯỜNG ĐẠI HỌC THỦ DẦU MỘT**

**TRẦN NGUYỄN THANH TUYỀN**

**HỆ THỐNG ĐIỂM DANH HỌC SINH TẠI TRƯỜNG  
PHỔ THÔNG SỬ DỤNG CÔNG NGHỆ NHẬN DẠNG  
KHUÔN MẶT**

**CHUYÊN NGÀNH: HỆ THỐNG THÔNG TIN  
MÃ SỐ: 8480104**

**LUẬN VĂN THẠC SỸ**

**NGƯỜI HƯỚNG DẪN KHOA HỌC:  
PGS. TS. QUẢN THÀNH THƠ**

**BÌNH DƯƠNG - NĂM 2019**

## LỜI CAM ĐOAN

Tôi xin cam đoan rằng, đề tài “Hệ thống điểm danh học sinh tại trường phổ thông sử dụng công nghệ nhận dạng khuôn mặt” là công trình nghiên cứu của tôi dưới sự hướng dẫn của thầy PGS. TS. Quản Thành Thor, xuất phát từ nhu cầu thực tế tại đơn vị tôi công tác và nguyện vọng tìm hiểu của bản thân.

Ngoại trừ kết quả tham khảo từ các công trình khác đã ghi rõ trong luận văn, các nội dung trình bày trong luận văn này là kết quả nghiên cứu do tôi thực hiện và kết quả của luận văn chưa từng công bố trước đây dưới bất kỳ hình thức nào.

*Bình Dương, ngày 25 tháng 02 năm 2019*

Tác giả

Trần Nguyễn Thanh Tuyền

## LỜI CẢM ƠN

Qua thời gian học tập và rèn luyện tại trường Đại học Thủ Dầu Một, được sự chỉ bảo và giảng dạy nhiệt tình của quý thầy cô, đặc biệt là quý thầy cô khoa Khoa Kỹ thuật - Công nghệ đã truyền đạt cho tôi những kiến thức về lý thuyết và thực hành trong suốt thời gian học ở trường.

Tôi xin chân thành cảm ơn thầy PGS.TS. Quán Thành Thơ đã tận tình hướng dẫn tôi hoàn thành tốt đề tài luận văn thạc sỹ này. Một lần nữa em chân thành cảm ơn thầy và chúc thầy nhiều sức khỏe.

Tuy nhiên Do kiến thức còn hạn hẹp nên không tránh khỏi những thiếu sót trong cách diễn đạt và trình bày. Tôi rất mong nhận được sự đóng góp ý kiến của quý thầy cô để báo cáo luận văn đạt được kết quả tốt nhất.

Tôi xin kính chúc quý thầy cô thật nhiều sức khỏe, niềm vui và luôn thành công trong công việc và cuộc sống.

Tôi xin chân thành cảm ơn!

## TÓM TẮT LUẬN VĂN

Hiện nay, cùng với sự phát triển của xã hội, vấn đề an ninh bảo mật được yêu cầu khắt khe tại mọi quốc gia trên thế giới. Các hệ thống nhận dạng con người được ra đời với độ tin cậy ngày càng cao. Một trong các bài toán nhận dạng rất được quan tâm hiện nay là bài toán nhận dạng khuôn mặt là cách mà con người sử dụng để phân biệt nhau. Bên cạnh đó, ngày nay việc thu thập, xử lý thông tin qua ảnh để nhận biết đối tượng đang được quan tâm và ứng dụng rộng rãi. Với phương pháp này, chúng ta có thể thu thập được nhiều thông tin từ đối tượng mà không cần tác động nhiều đến đối tượng nghiên cứu. Sự phát triển của khoa học máy tính tạo môi trường thuận lợi cho bài toán nhận dạng mặt người qua camera, qua ảnh, qua video. Các ứng dụng nhận dạng đã ra đời và có độ tin cậy cao.

Trong luận văn này tôi ứng dụng mô hình mạng Convolutional Neural Network để nhận dạng khuôn mặt học sinh để đưa vào hệ thống điểm danh học sinh bằng công nghệ nhận dạng. Tôi sẽ trình bày về kiến thức nền tảng của Deep Learning, các áp dụng Deep Learning vào bài toán nhận dạng khuôn mặt. Từ các kiến thức trên và tham khảo từ các công trình liên quan trên thế giới, tôi đã ứng dụng một thuật toán có tên là FaceNet sẽ học cách ánh xạ từ ảnh khuôn mặt vào không gian Euclide với khoảng cách đo được tương ứng với độ tương đồng của khuôn mặt. Thuật toán này có thể tạo ra vector đặc trưng và nhúng vào bài toán nhận dạng khuôn mặt, kiểm tra khuôn mặt và phân cụm khuôn mặt. Sử dụng Mạng Tích Chập (Convolution Network - CNN) được huấn luyện để tự tối ưu hóa bài toán. Sau đó tôi đã thực hiện một hệ thống nhằm thu thập hình ảnh của học sinh khối 8 trường THCS Định Hòa để thực hiện kiểm tra và đánh giá độ chính xác của ứng dụng.

## MỤC LỤC

CHƯƠNG 1 - GIỚI THIỆU .....	1
1.1. Đặt vấn đề .....	1
1.2. Mục tiêu và phạm vi nghiên cứu .....	1
1.3. Tổng quan về mô hình .....	2
CHƯƠNG 2 - KIẾN THỨC NỀN TẢNG .....	3
2.1. Mạng noron nhân tạo .....	3
2.1.1. Giới thiệu .....	3
2.1.2. Kiến trúc mạng .....	3
2.1.3. Huấn luyện mạng .....	7
2.1.4. Giải thuật Back – Propagation .....	9
2.1.5. Giảm lỗi cho mạng .....	12
2.2. Mô Hình Convolutional Neural Network – CNN .....	14
2.2.1. Giới thiệu mạng CNN .....	14
2.2.2. Mô hình kiến trúc mạng CNN .....	15
2.2.3. Các vấn đề cơ bản về mạng CNN .....	18
2.2.5. Huấn luyện mô hình .....	21
2.2.6. Bộ lọc và sắp đặt các tính năng (Filters And Feature Map) .....	23
2.2.7. Ứng dụng của mạng CNN .....	24
CHƯƠNG 3 - CÁC CÔNG TRÌNH LIÊN QUAN .....	25
3.1. Nhận dạng khuôn mặt sử dụng Bag-of-Words .....	25
3.2. Khoanh vùng một phần khuôn mặt sử dụng các mẫu đồng dạng .....	26
CHƯƠNG 4 - MÔ HÌNH ĐỀ XUẤT NHẬN DẠNG KHUÔN MẶT BẰNG DEEP LEARNING SỬ DỤNG MẠNG FACENET VÀ THỰC NGHIỆM .....	29
4.1 . Mô hình mạng CNN cho bài toán nhận dạng khuôn mặt .....	29
4.2 . Tổng quát hóa kích thước .....	30
4.2.1. Tổng quát hóa kích thước ảnh .....	30
4.2.2. Tổng quát hóa kích thước bộ lọc của Convolution .....	31
4.2.3. Tổng quát hóa kích thước bộ lọc của max pooling .....	31
4.2.4. Kích thước ảnh sau mỗi tầng .....	32

4.3. Phương pháp huấn luyện .....	32
4.3.1. Huấn luyện CNN cho việc extract feature .....	32
4.3.2. Huấn luyện mô hình phân loại .....	34
4.4. Thực hiện Thuật Toán.....	35
4.5. Thực Nghiệm .....	36
4.6. Ưu và Nhược Điểm của Thuật Toán .....	39
4.6.1. Ưu Điểm .....	39
4.6.2. Nhược Điểm .....	39
4.7. Nhận Xét Thuật Toán .....	39
4.8. Hiển thị kết quả và xuất file.....	40
4.8.1. Giao diện chính của ứng dụng.....	40
4.8.2. Xuất kết quả ra file Excel .....	41
KẾT LUẬN .....	43
CÔNG NGHỆ SỬ DỤNG .....	44
TÀI LIỆU THAM KHẢO.....	47

## **DANH MỤC CHỮ VIẾT TẮT**

1. ANN - Artificial Neural Network.
2. CNN - Convolutional Neural Network.
3. FC - Fully-connected.
4. ReLU - Rectified Linear Unit
5. SIFT –Scale Invariant Feature Transform
6. SVM - Support Vector Machine



## DANH MỤC HÌNH, ĐỒ THỊ

Hình 1. Mô hình tổng quan .....	2
Hình 2. Mô hình mạng nơron nhân tạo .....	3
Hình 3. Ảnh một tế bào Nơron nhân tạo .....	4
Hình 4. Mạng nơron truyền thẳng .....	7
Hình 5. Quá trình học của Supervised ANN.....	8
Hình 6. Quá trình học của mạng nơron.....	9
Hình 7. Mô hình tính toán một nơron .....	10
Hình 8. Giảm lỗi cho mạng .....	12
Hình 9. Underfitting .....	12
Hình 10. Overfitting .....	13
Hình 11. Convolution với filter.....	16
Hình 12. Tính toán với phương pháp MaxPooling .....	17
Hình 13. Kết nối cục bộ .....	18
Hình 14. Trọng số .....	19
Hình 15. Ví dụ về 1 ảnh trong CIFAR-10.....	20
Hình 16. Kết quả sau khi covolution ảnh 32x32 với filter 3x3.....	20
Hình 17. Tác động của trọng số đến loss function.....	22
Hình 18. Bước học (learning rate) .....	23
Hình 19. Bộ lọc và sắp đặt các tính năng.....	23
Hình 20. Sơ đồ thuật toán Khối Bag of Word.....	25
Hình 21. Ảnh kết quả sau khi khoanh vùng của nhóm tác giả.....	27
Hình 22. kết quả xác định điểm chính trong bộ dữ liệu LFPW với một số điểm lỗi.....	28
Hình 23. Sai số trung bình của kết quả .....	28
Hình 24. thuật toán nhận dạng khuôn mặt .....	28
Hình 25. Hình minh họa output khoảng cách khi sử dụng FaceNet .....	29
Hình 26. Kích thước ảnh đầu vào .....	30
Hình 27. Filter concatenation.....	31
Hình 28. Tiền xử lý ảnh .....	33

Hình 29. Mô phỏng quá trình tính toán tripletloss .....	34
Hình 30. Sơ đồ tổng quát thuật toán .....	35
Hình 31. Thực nghiệm trên hệ điều hành Ubuntu.....	36
Hình 32. Thực nghiệm nhận dạng khuôn mặt học sinh .....	37
Hình 33. Thực nghiệm nhận dạng khuôn mặt học sinh .....	37
Hình 34. Thực nghiệm nhận dạng khuôn mặt học sinh .....	38
Hình 35. Thực nghiệm nhận dạng khuôn mặt học sinh .....	38
Hình 36. Thực nghiệm nhận dạng khuôn mặt học sinh .....	39
Hình 37. Giao diện chính của ứng dụng .....	40
Hình 38. Thực nghiệm nhận dạng khuôn mặt và in kết quả trên giao diện .....	40
Hình 39. Thực nghiệm nhận dạng khuôn mặt và in kết quả trên giao diện .....	41
Hình 40. Xuất kết quả ra file Excel.....	41
Hình 41. Xuất kết quả ra file Excel.....	42

## DANH MỤC BẢNG

Bảng 1. Một số hàm truyền thông dụng .....	6
Bảng 2. Kết quả thực nghiệm trên bộ dữ liệu AR.....	26
Bảng 3. Kích thước ảnh sau mỗi tầng .....	32

## CHƯƠNG 1 - GIỚI THIỆU

### 1.1. Đặt vấn đề

Hiện nay, cuộc cách mạng công nghiệp 4.0 có sự tác động mạnh mẽ trên nhiều lĩnh vực, với sự xuất hiện của robot có trí tuệ nhân tạo, người máy làm việc thông minh, có khả năng ghi nhớ, học hỏi vô biên, trong khi khả năng đó ở con người thường chỉ có trong thời gian giới hạn. Chính vì vậy, việc các công nghệ cao và máy móc thông minh sẽ tạo cơ hội cho con người làm việc và hoạt động kinh doanh hiệu quả hơn bằng cách tận dụng những lợi thế mà cuộc cách mạng công nghệ 4.0 mang lại.

Tại Việt Nam, với việc đi sau và thừa hưởng những thành tựu từ cuộc cách mạng công nghiệp 4.0 do thế giới để lại cũng giúp chúng ta tiết kiệm được một cơ số thời gian nghiên cứu. Thay vào đó chúng ta có thể tập trung phát triển những thành tựu đó sao cho phù hợp và mang lại hiệu quả tốt nhất cho nền kinh tế đất nước.

Bên cạnh đó, vấn đề an ninh bảo mật đang được yêu cầu khắc khe tại mọi quốc gia trên thế giới. Các hệ thống xác định, nhận dạng con người được ra đời với độ tin cậy cao. Một trong những bài toán nhận dạng con người được quan tâm nhất hiện nay đó là nhận dạng qua khuôn mặt.

Riêng đối với lĩnh vực giáo dục: khi học sinh quá đông mà tài nguyên thì có hạn, nên việc đưa thành tựu của cuộc cách mạng công nghiệp 4.0 trong quản lý giáo dục là rất cần thiết, cụ thể ở đây tôi ứng dụng công nghệ AI (Artificial Intelligence), Machine Learning và Deep Learning vào việc điểm danh học sinh bằng công nghệ nhận dạng khuôn mặt. Với mục đích đưa những tiến bộ công nghệ vào phục vụ cho cuộc sống, tôi xin chọn đề tài nghiên cứu: **“Hệ thống điểm danh học sinh tại trường phổ thông sử dụng công nghệ nhận dạng khuôn mặt”**.

### 1.2. Mục tiêu và phạm vi nghiên cứu

**Mục tiêu nghiên cứu:**

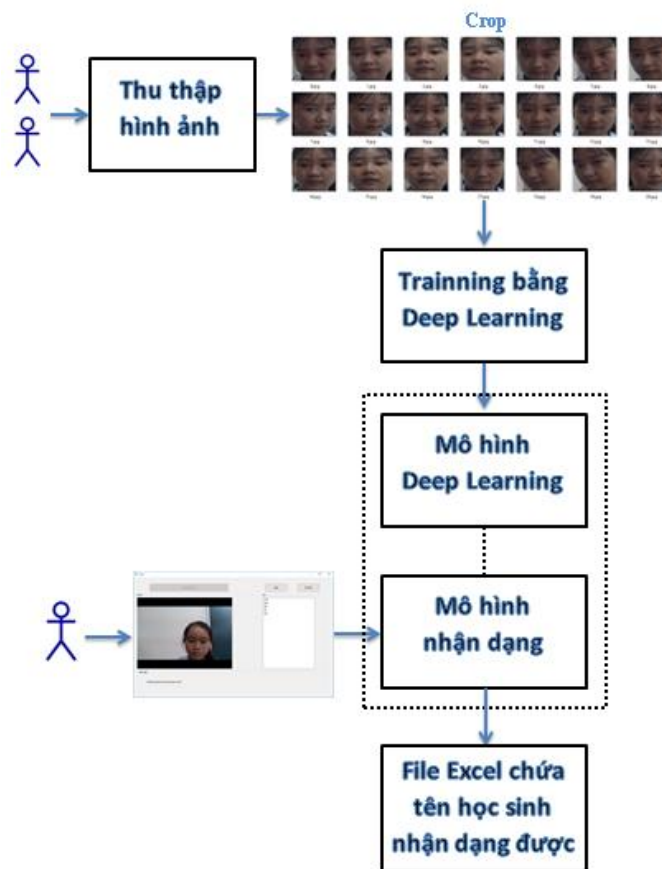
- Nghiên cứu phương pháp nhận dạng bằng mạng Neural.
- Tìm hiểu mô hình Convolutional Neural Network – cnn

- Áp dụng thành công mô hình Convolutional Neural Network – cnn phương pháp trên vào việc nhận dạng khuôn mặt.

***Phạm vi nghiên cứu:***

- Dữ liệu học sinh được xử lý là file ảnh tĩnh và động với ảnh được chụp với nhiều góc khác nhau hoặc lấy trực tiếp từ camera
- Ảnh được chụp trong điều kiện ánh sáng bình thường (không chụp ngược sáng, chụp bằng máy kỹ thuật số hoặc bằng camera của máy tính).

### 1.3. Tổng quan về mô hình



**Hình 1.** Mô hình tổng quan

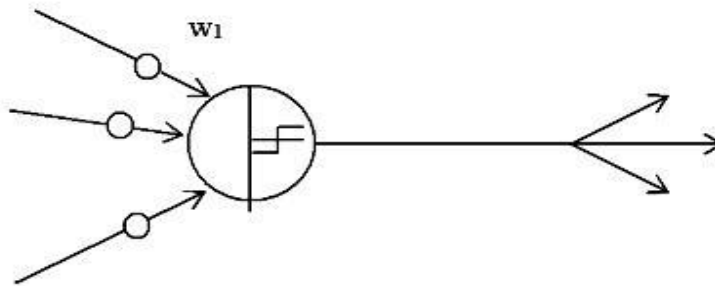
## CHƯƠNG 2 -KIẾN THỨC NỀN TẢNG

### 2.1. Mạng nơron nhân tạo

#### 2.1.1. Giới thiệu

Mạng nơron nhân tạo (ANN) là một mô phỏng xử lý thông tin, được nghiên cứu ra từ hệ thống thần kinh của sinh vật, giống như bộ não để xử lý thông tin. Nó bao gồm số lượng lớn các mối gắn kết cấp cao để xử lý các yếu tố làm việc trong mối liên hệ giải quyết vấn đề rõ ràng. ANN giống như con người, được học bởi kinh nghiệm, lưu những kinh nghiệm hiểu biết và sử dụng trong những tình huống phù hợp.

Đầu tiên ANN được giới thiệu năm 1943 bởi nhà thần kinh học Warren McCulloch và nhà logic học Walter Pitts. Nhưng với những kỹ thuật trong thời gian này chưa cho phép họ nghiên cứu được nhiều. Những năm gần đây mô phỏng ANN xuất hiện và phát triển. Các nghiên cứu ứng dụng đã được thực hiện trong các ngành: điện, điện tử, kỹ thuật chế tạo, y học, quân sự, kinh tế...và mới nhất là các nghiên cứu ứng dụng trong lĩnh vực quản lý dự án xây dựng. Tại Việt Nam việc nghiên cứu ứng dụng ANN cũng được chú trọng trong những năm gần đây và đang trên đà phát triển.

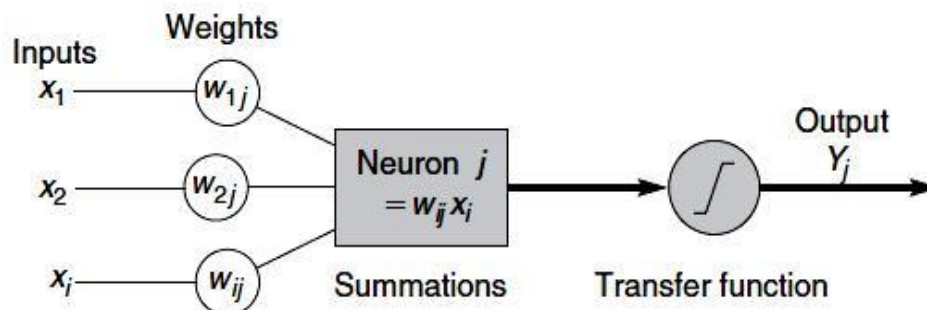


Hình 2. Mô hình mạng nơron nhân tạo [1]

#### 2.1.2. Kiến trúc mạng

Mạng nơron nhân tạo với cấu trúc tương tự trong đó một nơron là một khối tính toán gồm nhiều đầu vào, một đầu ra và một giá trị ngưỡng để cho phép tín hiệu có truyền qua nơron này hay không. Các giá trị đầu vào sẽ được nhân với một bộ trọng số  $w$  và tính tổng các kết quả cùng với một hằng số phụ  $b$  (bias). Nếu tổng vượt giá trị ngưỡng thì đầu ra sẽ có giá trị thể hiện nơron này đã được kích hoạt (thường là +1),

ngược lại sẽ là giá trị chưa được kích hoạt (thường là -1). Các giá trị trọng số đầu vào của mỗi nơron sẽ được điều chỉnh thông qua quá trình học để xây dựng được mạng nơron phù hợp cho bài toán đang cần giải quyết.



**Hình 3. Ảnh một tế bào Nơron nhân tạo** [1]

**Inputs:** Mỗi đầu vào (Input) tương ứng với 1 thuộc tính (attribute) của dữ liệu mẫu (patterns) Các tín hiệu này thường được đưa vào dưới dạng một vector N chiều

**Output:** Đầu ra là kết quả của một ANN là một giải pháp cho một vấn đề.

**Connection Weights (Trọng số liên kết):** Đây là thành phần rất quan trọng của một ANN, nó thể hiện mức độ quan trọng (độ mạnh) của dữ liệu đầu vào đối với quá trình xử lý thông tin (quá trình chuyển đổi dữ liệu từ lớp này sang lớp khác) Quá trình học (Learning Processing) của ANN thực ra là quá trình điều chỉnh các trọng số (Weight) của các dữ liệu đầu vào để có được kết quả mong muốn. Thông thường, các trọng số này được khởi tạo một cách ngẫu nhiên ở thời điểm khởi tạo mạng và được cập nhật liên tục trong quá trình học của mạng.

**Summation Function (Hàm tổng):** Tính tổng trọng số của tất cả các đầu vào được đưa vào mỗi nơron. Hàm tổng của một nơron đối với n đầu vào được tính theo công thức sau:

$$Y_j = \sum_{i=1}^n X_i W_i$$

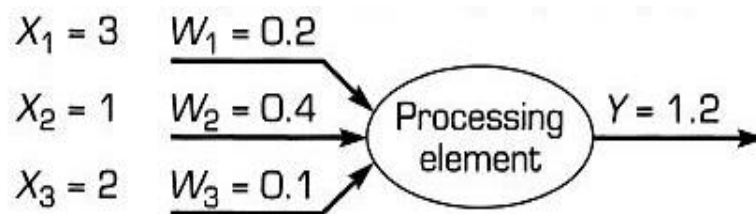
Hàm tổng đối với nhiều nơron trong cùng một lớp:

$$Y_j = \sum_{i=1}^n X_i W_{ij}$$

**Transformation (Transfer) Function (Hàm chuyển đổi):** Hàm tổng (Summation Function) của một nơron cho biết khả năng kích hoạt (Activation) của nơron đó còn gọi là kích hoạt bên trong (internal activation). Các Nơron này có thể sinh ra một kết quả (output) hoặc không trong ANN (nói cách khác rằng có thể output của 1 nơron có thể được chuyển đến lớp tiếp theo trong mạng nơron hoặc không). Mối quan hệ giữa Internal Activation và kết quả (output) được thể hiện bằng hàm chuyển đổi (Transfer Function)

Hàm tổng:  $Y = 3(0.2) + 1(0.4) + 2(0.1) = 1.2$

Hàm chuyển đổi:  $Y_T = 1/(1 + e^{-1.2}) = 0.77$



Việc lựa chọn chuyển đổi có tác động lớn đến kết quả của ANN. Hàm chuyển đổi phi tuyến được sử dụng phổ biến trong ANN là sigmoid (logical activation) function.

$$Y_T = 1/(1 + e^{-Y})$$

Trong đó :

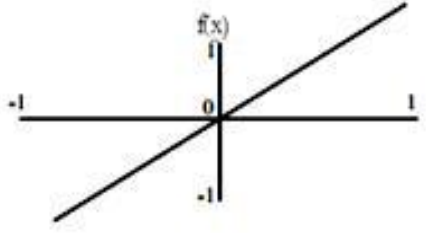
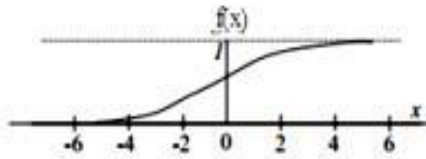
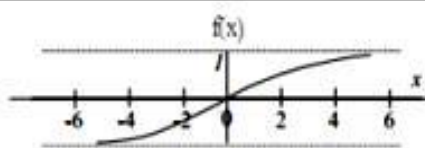
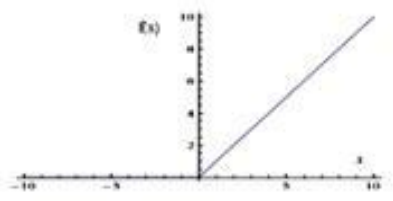
$Y_T$ : Hàm chuyển đổi

$Y$ : Hàm tổng

Kết quả xử lý tại các nơron (Output) đôi khi rất lớn, vì vậy Transfer function được sử dụng để xử lý output này trước khi chuyển đến lớp tiếp theo. Đôi khi thay vì sử dụng Transfer Function người ta sử dụng giá trị ngưỡng (Threshold value) để kiểm soát các output của các nơron tại một lớp nào đó trước khi chuyển các output này đến



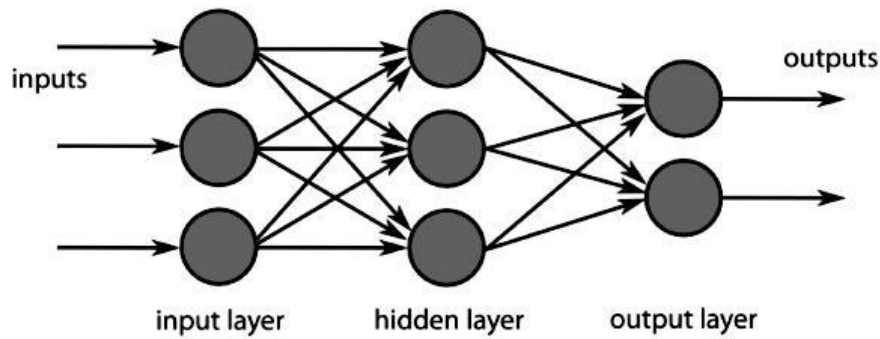
các lớp tiếp theo. Nếu output của một nơron nào đó nhỏ hơn Threshold thì nó sẽ không được chuyển đến lớp tiếp theo

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

**Bảng 1. Một số hàm truyền thông dụng** [1]

Kiến trúc chung của một ANN gồm 3 thành phần đó là Input layer, Hidden layer và output layer. Trong đó, lớp ẩn (Hidden Layer) gồm các nơron nhận dữ liệu input từ các nơron ở lớp (Layer) trước đó và chuyển đổi các input này cho các lớp xử lý tiếp theo. Trong một ANN có thể có nhiều Hidden Layer.

Mô hình đơn giản nhất của mạng nơron chính là mạng nơron truyền thẳng. Các nơron trong mạng truyền thẳng liên kết với nhau mà không hình thành chu trình nên tín hiệu sẽ truyền thẳng từ đầu vào qua các lớp n và đến đầu ra.



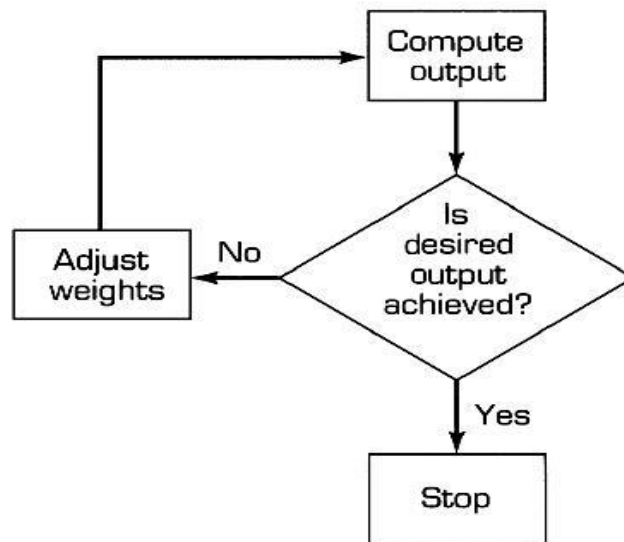
**Hình 4. Mạng nơron truyền thẳng** [1]

### 2.1.3. Huấn luyện mạng

Tiến trình học là tiến trình quan trọng của con người, nhờ học mà bộ não ngày càng tích lũy những kinh nghiệm để thích nghi với môi trường và xử lý tình huống tốt hơn. Mạng nơron xây dựng lại cấu trúc bộ não thì cần phải có khả năng nhận biết dữ liệu thông qua tiến trình học, với các thông số tự do của mạng có thể thay đổi liên tục bởi những thay đổi của môi trường và mạng nơron ghi nhớ giá trị đó.

ANN được huấn luyện (Training) hay được học (Learning) theo 2 kỹ thuật cơ bản đó là học có giám sát (Supervised Learning) và học không giám sát (Unsupervised Learning). Trong phạm vi của luận văn này tôi chủ yếu đề cập về quá trình huấn luyện ANN theo kỹ thuật Supervised learning.

**Supervised learning:** Quá trình Training được lặp lại cho đến kết quả(output) của ANN đạt được giá trị mong muốn (Desired value) đã biết. Để hình thành cho kỹ thuật này là mạng Neuron lan truyền ngược (Backpropagation).

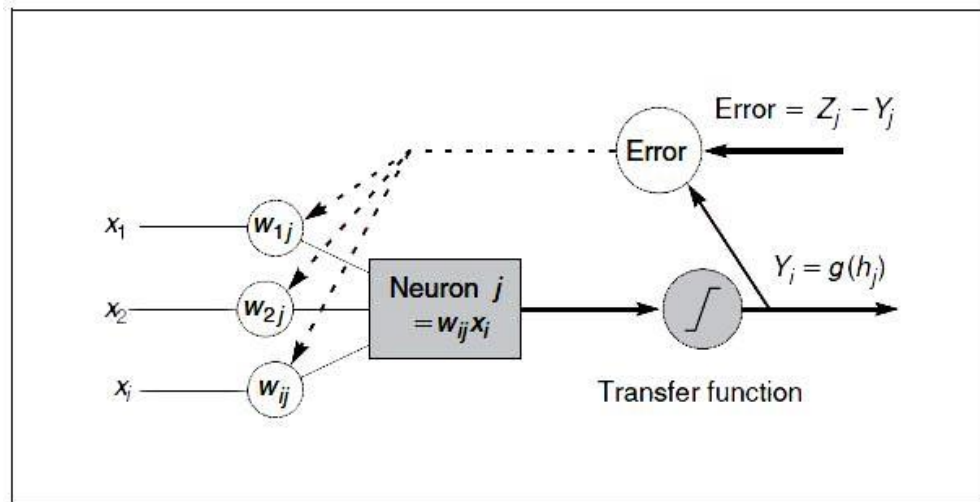


**Hình 5. Quá trình học của Supervised ANN [1]**

Mạng Neuron có 3 cách huấn luyện chính:

- + On-line training: Các trọng số của mạng (weight) được cập nhật ngay lập tức sau khi một input pattern được đưa vào mạng
- + Stochastic training: cũng giống như on-line training nhưng việc chọn các input patterns để đưa vào mạng từ training set được thực hiện ngẫu nhiên (random)
- + Batch training thì tất cả các input patterns được đưa vào mạng cùng lúc và sau đó cập nhật các trọng số mạng đồng thời

Giả sử sau khi tính toán từ các input value đã cho, ta có output là Y Giá trị mong muốn (Desired) là Z đã biết trước. Sự chênh lệch giữa Y và kết quả mong muốn Z được biểu diễn bởi tham số delta (gọi là lỗi) =  $Z - Y$  Mục đích của việc huấn luyện là làm sao cho delta càng nhỏ càng tốt (Nếu  $\text{delta} = 0$  là hoàn hảo nhất) bằng cách điều chỉnh trọng số (weight) của các dữ liệu vào



**Hình 6. Quá trình học của mạng nơron [1]**

#### 2.1.4. Giải thuật Back – Propagation

Thuật toán Back – Propagation được sử dụng để điều chỉnh các trọng số kết nối sao cho tổng sai số e nhỏ nhất.

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

Trong đó:

$t(x_i, w)$ : giá trị của tập mẫu

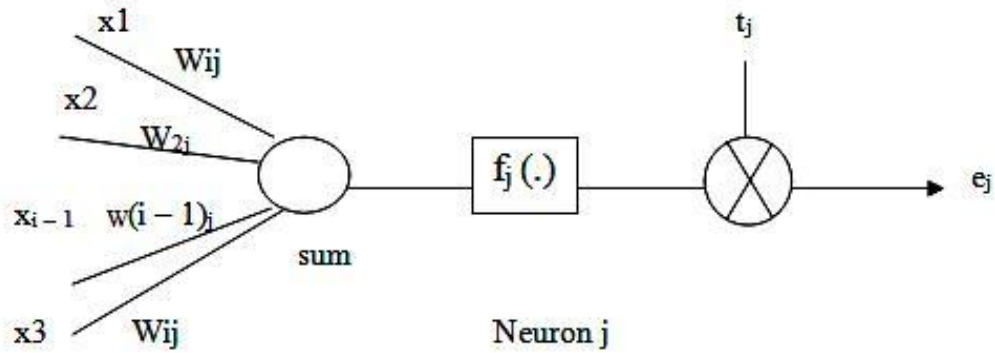
$y(x_i)$ : giá trị kết xuất của mạng

Trước tiên, ta xét trên 1 nơron, mỗi nơron đều có giá trị vào và ra, mỗi giá trị đều có một trọng số để đánh giá mức độ ảnh hưởng của giá trị vào đó. Thuật toán Back - Propagation sẽ điều chỉnh các trọng số đó để giá trị  $e_j = t_j - y_j$  là nhỏ nhất.

Trước hết ta phải xác định vị trí của mỗi nơron. Nơron nào là của lớp ẩn và nơron nào là của lớp xuất. Ta cần biết các ký hiệu:

$W_{ij}$ : vector trọng số của nơron j số đầu vào i

$u_j$ : vector giá trị kết xuất của nơron trong lớp j



**Hình 7. Mô hình tính toán một nơron [1]**

- Giá trị sai số của nơron j tại vòng lặp thứ n
- Tổng bình phương sai số của mạng nơron:

$$e_j(n) = t_j(n) - y_j(n)$$

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

- Tại nơron j ta có tổng trọng số input:

$$u_j(n) = \sum_{i=0}^p w_{ij} x_j(n)$$

- Giá trị kết xuất của nơron j:

$$y_j(n) = f_j(u_j(n))$$

- Tính toán giá trị đạo hàm sai số cho mỗi nơron  $w_{ij}$ :

$$\frac{\partial E(n)}{\partial w_{ij}(n)} = \frac{\partial E(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial u_j(n)} \frac{\partial u_j(n)}{\partial w_{ij}(n)};$$

Trong đó:

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

$$\frac{\partial e_j(n)}{\partial y_j(n)} = \frac{\partial(t_i(n) - y_i(n))}{\partial y_j(n)} = -1;$$

$$\frac{\partial y_j(n)}{\partial u_j(n)} = f'_j(u_j(n));$$

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

$$\Rightarrow \frac{\partial E(n)}{\partial w_{ij}(n)} = -e_j(n) \cdot f'_j(u_j(n)) x_i(n)$$

- Giá trị điều chỉnh trọng số:

$$\Delta w_{ij} = -\eta \frac{\partial E(n)}{\partial w_{ij}(n)} = -\eta e_j(n) \cdot f'_j(u_j(n)) \cdot x_i(n);$$

Đặt

$$\delta_j = \frac{\partial E(n)}{\partial w_{ij}(n)} = \frac{\partial E(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial u_j(n)} = e_j(n) \cdot f'_j(u_j(n))$$

Ta có:

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

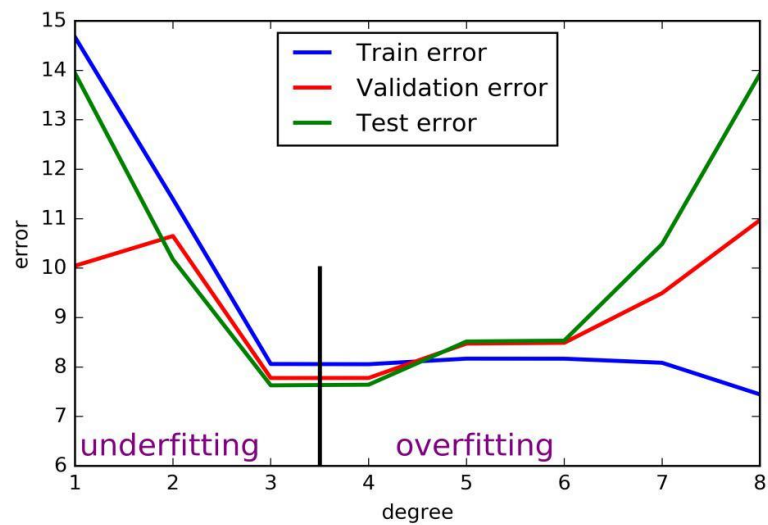
Từ đó ta có công thức điều chỉnh trọng số

$$w_{ij(n+1)} = w_{ij}(n) + \Delta w_{ij}(n)$$

Như vậy quá trình điều chỉnh trọng số có thể được xác định theo các công thức trên, tuy nhiên ta cần phải xác định vị trí của nơron thuộc lớp nào (lớp ẩn hay lớp xuất). Điều này rất quan trọng trong việc tính toán cho từng hệ số điều chỉnh trọng số.

### 2.1.5. Giảm lỗi cho mạng

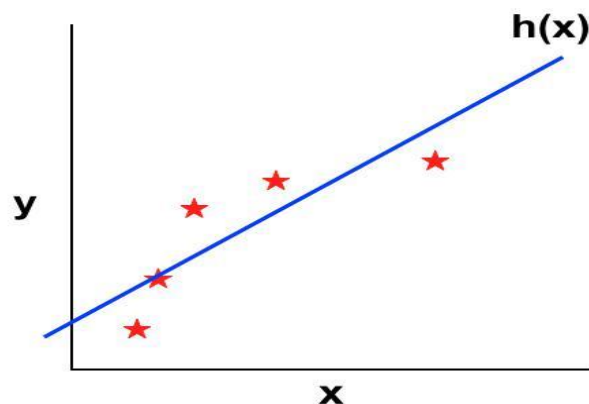
Ở đây tôi đề cập đến Overfitting, Underfitting.



Hình 8. Giảm lỗi cho mạng [8]

#### 1.1.5.1. Underfitting

Underfitting là khi mô hình của ta quá đơn giản, không thể giảm thiểu được đáng kể loss function nên cũng không thể mô tả được xu hướng của dữ liệu (còn được gọi là High Bias).



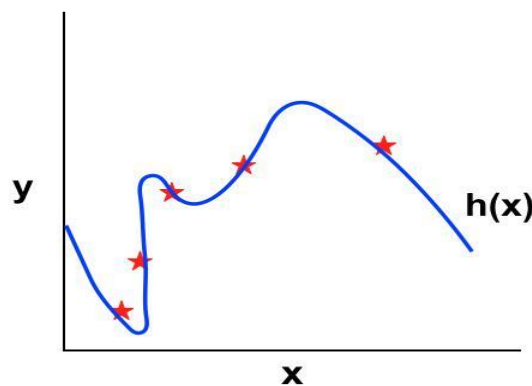
Hình 9. Underfitting [8]

Hàm  $h(x)$  thu được ở trên có vẻ không được tốt, bản thân đối với dữ liệu training ta đã thấy được sai số rõ rệt chứ chưa nói tới việc thử trên dữ liệu mới trong trường hợp này vấn đề chúng ta gặp phải được gọi là Underfitting.

Với Underfitting thì chỉ đơn thuần là mô hình của ta quá đơn giản, nên ta cần thêm những thành phần đa thức để nó phức tạp hơn. Nên khi giá trị của loss function lớn ta sẽ đẩy bậc của hàm số lên. Dĩ nhiên đẩy lên cao quá ta sẽ gặp vấn đề về Overfitting.

#### 2.1.5.2. Overfitting

Overfitting lại là khi mô hình của ta quá phức tạp, tuy giảm thiểu được đáng kể, thậm chí toàn bộ sai số nhưng cũng không thể mô tả được xu hướng của dữ liệu (còn được gọi là High Variance).



Hình 10. Overfitting [8]

Hàm  $h(x)$  dường như đi qua mọi dữ liệu training, có nghĩa là loss function của ta gần xấp xỉ 0. Đúng ra khi loss function càng thấp, ta phải thu được  $h(x)$  tốt hơn, nhưng trong trường hợp này, rõ ràng  $h(x)$  không đi theo xu thế của dữ liệu mà chỉ đơn thuần fit được nhiều dữ liệu training hay nói cách khác khi ta đưa  $h(x)$  vào sử dụng, nó cũng không thể dự đoán tốt được. Trường hợp đơn giản nhất ở đây là khi bậc của  $h(x)$  cao hơn hoặc bằng số lượng dữ liệu training.

Với Overfitting, do mô hình quá phức tạp nên ta cần giảm bậc của hàm số hay giảm số lượng feature. Việc giảm feature ngoài bỏ bớt những thành phần đa thức, ta còn có thể bỏ bớt những feature không cần thiết.



## 2.2. Mô Hình Convolutional Neural Network – CNN

### 2.2.1. Giới thiệu mạng CNN

CNN là một trong những mô hình Deep learning tiên tiến giúp cho chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao. Trong phạm vi của luận văn, tôi sẽ trình bày về xoắn (Convolution - còn được gọi là tích chập trong giải tích hàm) cũng như tương của mô hình CNN trong phân lớp hình ảnh áp dụng cho bài toán nhận dạng ảnh.

ANN đã được áp dụng nhiều trong các bài toán nhận dạng. Tuy nhiên, ANN không thể hiện tốt lắm đối với các dữ liệu hình ảnh. Chính sự liên kết quá đầy đủ tạo nên những hạn chế cho mô hình. Dữ liệu hình ảnh có kích thước khá lớn, một tấm ảnh xám có kích thước  $32 \times 32$  (pixels) sẽ cho ra vector đặc trưng có 1024 chiều, còn đối với ảnh màu cùng kích thước sẽ là 3072 chiều. Điều này cũng có nghĩa là cần tới 3072 trọng số kết nối giữa lớp vào và một nút (node) ở lớp ẩn kế tiếp. Số lượng trọng số sẽ càng nhân rộng hơn nữa nếu số lượng node trong lớp ẩn tăng lên, số lượng lớp  $n$  tăng lên. Như vậy chỉ với một bức ảnh nhỏ  $32 \times 32$  thì cũng cần đến một mô hình khá đồ sộ. Điều này khiến cho việc thao tác với các ảnh có kích thước lớn hơn trở nên khó khăn.

Dựa trên tư tưởng này CNN ra đời với một kiến trúc khác so với mạng truyền thẳng. Thay vì toàn bộ ảnh nối với một node thì chỉ có một phần cục bộ trong ảnh nối đến một node trong lớp tiếp theo (Local connectivity). Dữ liệu hình ảnh thông qua các lớp của mô hình này sẽ được “học” ra các đặc trưng để tiến hành phân lớp một cách hiệu quả.

CNN có kiến trúc hợp lý hơn so với mạng ANN, đặc biệt không giống một mạng thần kinh truyền thống, các lớp của một CNN sắp xếp theo 3 chiều: chiều rộng, chiều cao và chiều sâu.

Trong mô hình CNN các lớp liên kết được với nhau thông qua cơ chế convolution. Lớp tiếp theo là kết quả convolution từ lớp trước đó, nhờ vậy mà ta có được các kết nối cục bộ. Nghĩa là mỗi neuron ở lớp tiếp theo sinh ra từ các bộ lọc (filter) áp đặt lên một vùng ảnh cục bộ của neuron lớp trước đó.

Mỗi lớp như vậy được áp đặt các bộ lọc khác nhau, thông thường có vài trăm đến vài nghìn bộ lọc như vậy. Một số lớp khác như pooling/subsampling layer dùng để chắt lọc lại các thông tin hữu ích hơn (loại bỏ các thông tin nhiễu). Tuy nhiên, ta sẽ không đi sâu vào khái niệm của các lớp này. Trong suốt quá trình huấn luyện, CNN sẽ tự động học được các thông số cho các bộ lọc.

### 2.2.2. Mô hình kiến trúc mạng CNN

Như đã đề cập ở trên, một CNN là một dãy các lớp và các lớp của một CNN chuyển một thể tích kích hoạt sang một lớp khác thông qua một hàm khác biệt [7]

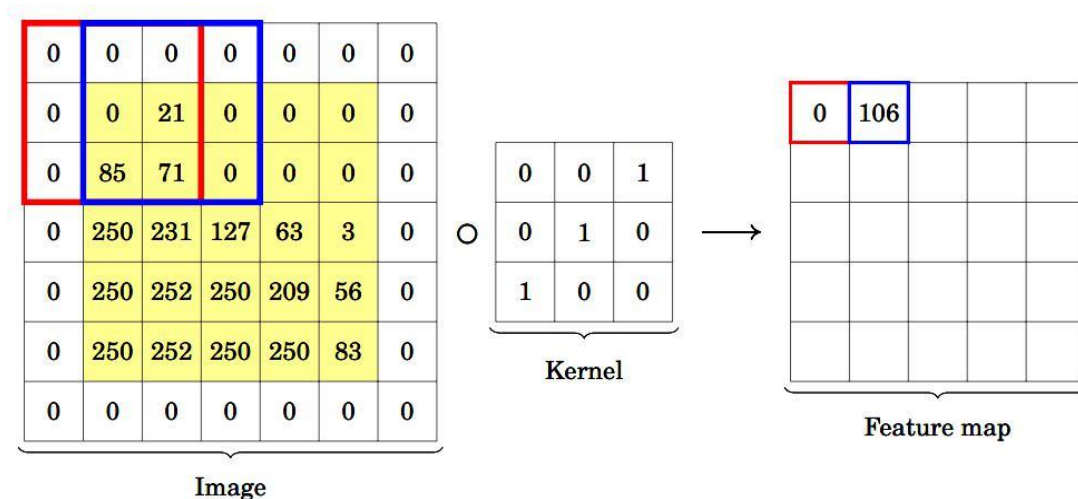
Về cơ bản mô hình mạng nơron xoắn bao gồm các lớp sau:

- + Lớp Convolutional
- + Lớp RELU
- + Lớp Pooling
- + Lớp Fully connected.

Các lớp này được xếp chồng lên nhau để tạo thành kiến trúc CNN đầy đủ. Sự sắp xếp về số lượng và thứ tự giữa các lớp này sẽ tạo ra những mô hình khác nhau phù hợp cho các bài toán khác nhau.

#### 2.2.2.1. Lớp Convolutional

Lớp này chính là nơi thể hiện tư tưởng ban đầu của mạng nơron xoắn. Thay vì kết nối toàn bộ điểm ảnh, lớp này sẽ sử dụng một bộ các bộ lọc (filter) có kích thước nhỏ so với ảnh (thường là  $3 \times 3$  hoặc  $5 \times 5$ ) áp vào một vùng trong ảnh và tiến hành tính xoắn giữa bộ filter và giá trị điểm ảnh trong vùng cục bộ đó. Bộ filter sẽ lần lượt được dịch chuyển theo một giá trị bước trượt (stride) chạy dọc theo ảnh và quét toàn bộ ảnh.



**Hình 11. Convolution với filter [9]**

Như vậy với một bức ảnh  $32 \times 32$  và một filter  $3 \times 3$ , ta sẽ có kết quả là một tấm ảnh mới có kích thước  $32 \times 32$  (với điều kiện đã thêm padding vào ảnh gốc để tính xoắn cho các trường hợp filter quét ra các biên cạnh) là kết quả xoắn của filter và ảnh. Với bao nhiêu filter trong lớp này thì ta sẽ có bấy nhiêu ảnh tương ứng mà lớp này trả ra và được truyền vào lớp tiếp theo. Các trọng số của filter ban đầu sẽ được khởi tạo ngẫu nhiên và sẽ được học dần trong quá trình huấn luyện mô hình.

#### 2.2.2.2. Lớp *RELU* – Rectified linear unit

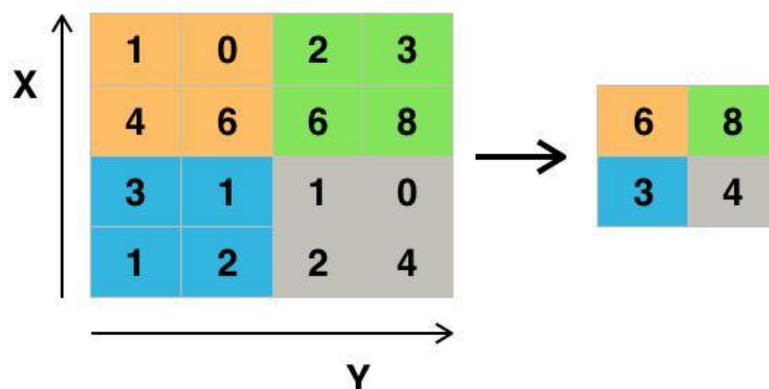
Lớp này thường được cài đặt ngay sau lớp Convolutional. Lớp này sử dụng hàm kích hoạt  $f(x) = \max(0, x)$ . Nói một cách đơn giản, lớp này có nhiệm vụ chuyển toàn bộ giá trị âm trong kết quả lấy từ lớp Convolutional thành giá trị 0. Ý nghĩa của cách cài đặt này chính là tạo nên tính phi tuyến cho mô hình. Tương tự như trong mạng truyền thẳng, việc xây dựng dựa trên các phép biến đổi tuyến tính sẽ khiến việc xây dựng đa tầng đa lớp trở nên vô nghĩa có rất nhiều cách để khiến mô hình trở nên phi tuyến như sử dụng các hàm kích hoạt sigmoid, tanh, ... nhưng hàm  $f(x) = \max(0, x)$  dễ cài đặt, tính toán nhanh mà vẫn hiệu quả.

#### 2.2.2.3. Lớp *Pooling*

Lớp này sử dụng một cửa sổ trượt quét qua toàn bộ ảnh dữ liệu, mỗi lần trượt theo một bước trượt (stride) cho trước. Khác với lớp Convolutional, lớp Pooling không

tính xoắn mà tiến hành lấy mẫu (subsampling). Khi cửa sổ trượt trên ảnh, chỉ có một giá trị được xem là giá trị đại diện cho thông tin ảnh tại vùng đó (giá trị mẫu) được giữ lại. Các phương thức lấy phổ biến trong lớp Pooling là MaxPooling (lấy giá trị lớn nhất), MinPooling (lấy giá trị nhỏ nhất) và AveragePooling (lấy giá trị trung bình).

Xét một ảnh có kích thước  $32 \times 32$  và lớp Pooling sử dụng filter có kích thước  $2 \times 2$  với bước trượt  $\text{stride} = 2$ , phương pháp sử dụng là MaxPooling. Filter sẽ lần lượt duyệt qua ảnh, với mỗi lần duyệt chỉ có giá trị lớn nhất trong 4 giá trị nằm trong vùng cửa sổ  $2 \times 2$  của filter được giữ lại và đưa ra đầu ra. Như vậy sau khi qua lớp Pooling, ảnh sẽ giảm kích thước xuống còn  $16 \times 16$  (kích thước mỗi chiều giảm 2 lần).



**Hình 12. Tính toán với phương pháp MaxPooling [9]**

Lớp Pooling có vai trò giảm kích thước dữ liệu. Với một bức ảnh kích thước lớn qua nhiều lớp Pooling sẽ được thu nhỏ lại tuy nhiên vẫn giữ được những đặc trưng cần cho việc nhận dạng (thông qua cách lấy mẫu). Việc giảm kích thước dữ liệu sẽ làm giảm lượng tham số, tăng hiệu quả tính toán và góp phần kiểm soát hiện tượng quá khớp (overfitting).

#### **2.2.2.4. Lớp Lớp FC – fully connected**

Lớp này tương tự với lớp trong mạng nơron truyền thẳng, các giá trị ảnh được liên kết đầy đủ vào nút trong lớp tiếp theo. Sau khi ảnh được xử lý và rút trích đặc trưng từ các lớp trước đó, dữ liệu ảnh sẽ không còn quá lớn so với mô hình truyền thẳng nên ta có thể sử dụng mô hình truyền thẳng để tiến hành nhận dạng.

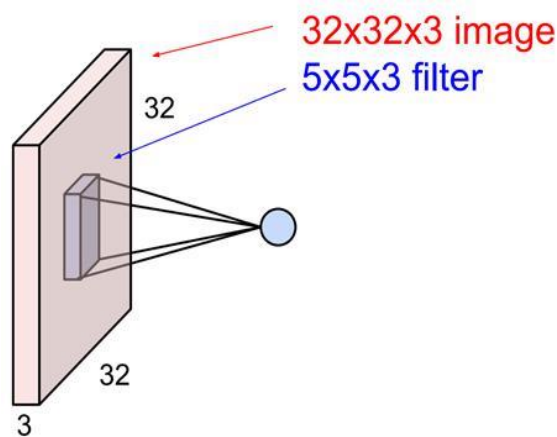
Tóm lại, lớp fully-connected đóng vai trò như một mô hình phân lớp và tiến hành dựa trên dữ liệu đã được xử lý ở các lớp trước đó

### 2.2.3. Các vấn đề cơ bản về mạng CNN

#### 2.2.3.1. Kết nối cục bộ

Trong mô hình mạng truyền ngược (feedforward neuron network) thì mỗi neuron đầu vào (input node) cho mỗi neuron đầu ra trong các lớp tiếp theo. Mô hình này gọi là mạng kết nối đầy đủ (fully connected layer) hay mạng toàn vẹn (affine layer). Còn trong mô hình CNN thì ngược lại. Các layer liên kết được với nhau thông qua cơ chế convolution. Layer tiếp theo là kết quả convolution từ layer trước đó, nhờ vậy mà ta có được các kết nối cục bộ. Như vậy mỗi neuron ở lớp kế tiếp sinh ra từ kết quả của filter áp đặt lên một vùng ảnh cục bộ của neuron trước đó

Giả sử, chúng ta có một ảnh đầu vào  $32 \times 32$  và  $5 \times 5$  trường tiếp nhận cục bộ, sau đó sẽ có  $28 \times 28$  neuron trong lớp ẩn.



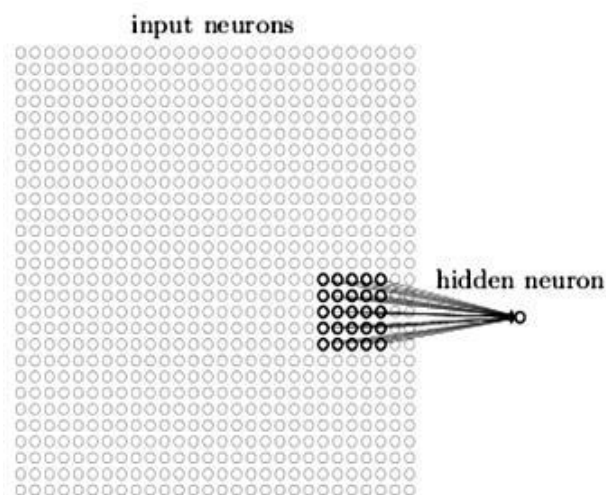
Hình 13. Kết nối cục bộ [9]

#### 2.2.3.2. Trọng số

Tương tự như mạng neuron truyền thẳng, mạng neuron xoắn cũng là một mô hình học cho nên khởi tạo ban đầu cho các trọng số trong mạng là ngẫu nhiên và sẽ được điều chỉnh thông qua quá trình học. Thuật toán học cho mạng neuron xoắn cũng tương tự như mạng neuron truyền thẳng là thuật toán lan truyền ngược sử dụng gradient

descent; chỉ khác nhau ở chỗ mạng xoắn không liên kết đầy đủ nên độ lỗi ở mỗi lớp chỉ tính dựa vào một phần các node trong lớp tiếp theo chứ không phải toàn bộ.

Đầu vào của mạng CNN là một ảnh, ví dụ như ảnh có kích thước 32x32 thì tương ứng đầu vào là một ma trận có 32x32 và giá trị mỗi điểm ảnh là một ô trong ma trận. Trong mô hình mạng ANN truyền thống thì chúng ta sẽ kết nối các neuron đầu vào vào tầng ảnh. Tuy nhiên trong CNN chúng ta không làm như vậy mà chúng ta chỉ kết nối trong một vùng nhỏ của các neuron đầu vào như một filter có kích thước 5x5. Mỗi một kết nối sẽ học một trọng số và mỗi neuron ẩn sẽ học một bias.



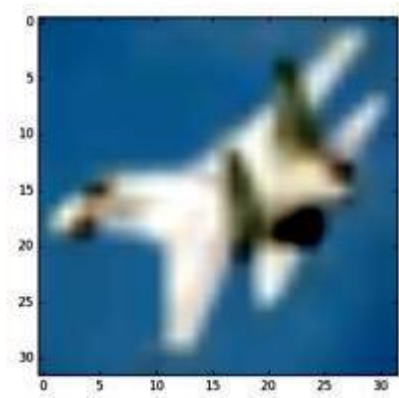
**Hình 14. Trọng số [9]**

#### **2.2.4. Mô hình nhiều lớp**

Một mạng nơron xoắn được hình thành bằng cách ghép các lớp nêu trên lại với nhau. Mô hình bắt đầu với lớp Convolutional. Lớp RELU thường luôn được cài đặt ngay sau lớp Convolutional hoặc thậm chí kết hợp cả hai lớp này thành một lớp. Các lớp tiếp theo có thể là Convolutional hay Pooling tùy theo kiến trúc mà ta muốn xây dựng. Cuối cùng sẽ là lớp fully-connected để tiến hành phân lớp. Xét một kiến trúc sau đây:

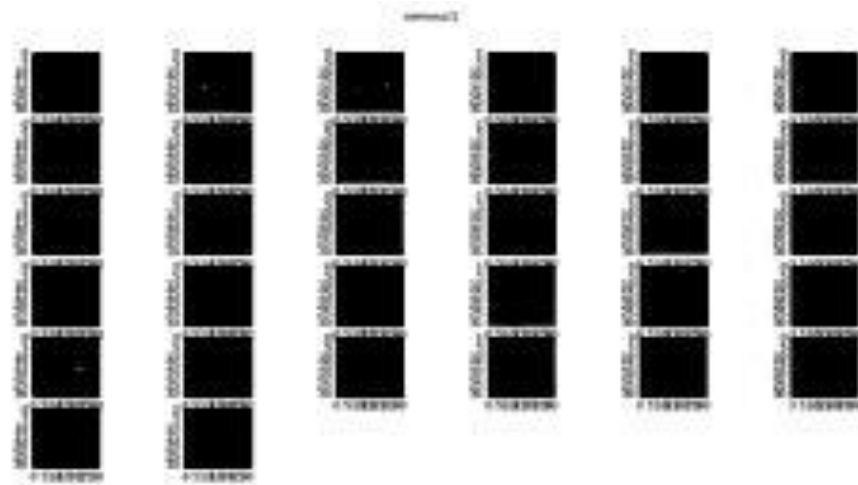
Conv1 (with RELU) – Pooling – Conv2 (with RELU) – Pooling – FC – FC

Lấy một hình ảnh cần nhận dạng có kích thước 32×32 như sau (lấy từ bộ dữ liệu cifar-10):



**Hình 15. Ví dụ về 1 ảnh trong CIFAR-10 [9]**

Hình ảnh sẽ được đưa vào lớp Conv1 (Convolutional kết hợp RELU) gồm 32 filter có kích thước  $3 \times 3$ , mỗi filter sẽ được dùng để tính xoắn với ảnh và cho ra một ảnh kết quả tương ứng. Với 32 filter ta sẽ có 32 ảnh kết quả như sau:



**Hình 16. Kết quả sau khi covolution ảnh  $32 \times 32$  với filter  $3 \times 3$  [9]**

Mỗi ảnh trên đều có kích thước tương ứng là  $32 \times 32$ . Sau đó, cả 32 ảnh này đều được cho qua lớp Pooling và kết quả trả ra sẽ là 32 ảnh có kích thước  $16 \times 16$ .

Tiếp tục dữ liệu sẽ đi vào lớp Conv2 Tương tự như Conv1, ảnh sẽ được tính xoắn với filter và trả ra kết quả. Lớp Pooling tiếp theo sẽ tiếp tục giảm kích thước của ảnh xuống còn  $8 \times 8$ . Với kích thước đủ nhỏ như vậy, lớp Fully-connected tiếp theo sẽ xử lý và đưa ra kết quả phân lớp hay kết quả nhận dạng.

### 2.2.5. Huấn luyện mô hình

Để huấn luyện được mô hình CNN, ban đầu, trọng số và giá trị bộ lọc được chọn ngẫu nhiên. Với mỗi hình ảnh đã được gán nhãn đi qua mô hình CNN gọi là quá trình đào tạo cho mô hình vì vậy chúng ta cần một tập các hình ảnh đủ lớn đã được gán nhãn cho biết hình ảnh đó là gì, sau đó lần lượt đưa từng ảnh qua mô hình CNN Cũng giống như ANN, CNN sử dụng giải thuật lan truyền ngược để huấn luyện mô hình.

Có thể mô tả mạng được huấn luyện như sau:

- + Trong quá trình lan truyền thuận (forward pass): Xét một hình ảnh đào tạo với kích thước 32x32x3 và truyền nó qua toàn bộ mạng. Vì tất cả các trọng số và bộ lọc đã được khởi tạo ngẫu nhiên, đầu ra trong trường hợp này tương tự như: [ 1.1 .1 .1 .1 .1 .1 .1 .1 ] có thể nói với đầu ra kể trên không ưu tiên cho một đối tượng cụ thể nào do đó sẽ không có kết luận hợp lý ở đây, điều này dẫn đến hàm mất mát được lan truyền ngược.

- + Hàm mất mát (loss function): Loss function có thể được định nghĩa theo nhiều cách khác nhau nhưng phổ biến là MSE

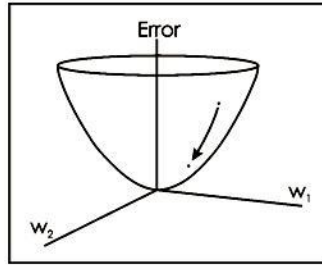
$$E_{total} = \sum \frac{1}{2} (target - output)^2$$

Trong đó:      target – thực tế

                 output – dự đoán

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





**Hình 17. Tác động của trọng số đến loss function [10]**

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

+ Cập nhật trọng số (weight update): Ở bước này chúng ta lấy tất cả trọng số của các bộ lọc và cập nhật lại chúng để chúng thay đổi theo hướng có lợi cho mô hình

$$w = w_i - \eta \frac{dL}{dW}$$

w=Weight

w<sub>i</sub>=Initial Weight

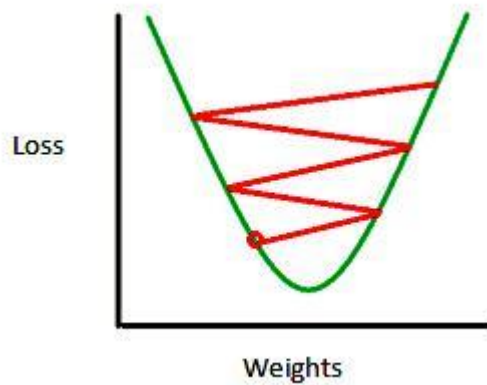
η=Learning Rate

Trong đó : W trọng số sau khi cập nhật

W<sub>i</sub>: trọng số ban đầu

η: bước học.

Bước học là một tham số được khởi tạo bởi người lập trình. Bước học lớn có nghĩa là việc thực hiện trong việc cập nhật trọng lượng vì vậy có thể mất ít thời gian hơn để mô hình tập trung vào một bộ trọng số tối ưu. Tuy nhiên, bước học cao quá có thể dẫn đến tới bước nhảy quá lớn và khó để xác định được các điểm tối ưu cần tìm.



**Hình 18. Bước học (learning rate) [10]**

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

#### 2.2.6. Bộ lọc và sắp đặt các tính năng (Filters And Feature Map)

Bộ lọc (filter) là một ma trận có kích thước nhỏ thường là  $3 \times 3$  hoặc  $5 \times 5$ , được khởi tạo ban đầu. Khi “trượt” filter trên ảnh đầu vào ứng với mỗi vùng trượt sẽ có một neuron n trong lớp n đầu tiên, quá trình này được lặp lại liên tục đến khi trượt xong toàn bộ bức ảnh cuối cùng ta sẽ được 1 feature map. Tuy nhiên trong thực tế chúng ta cần rất nhiều feature map.



**Hình 19. Bộ lọc và sắp đặt các tính năng [10]**

Mỗi một lớp được sử dụng các filter khác nhau thông thường có hàng trăm hàng nghìn filter và kết hợp kết quả của chúng lại. Ngoài ra có một số layer khác như pooling/subsampling layer dùng để chắt lọc lại các thông tin hữu ích hơn (loại bỏ các thông tin nhiễu). Trong quá trình huấn luyện mạng (training) CNN tự động học các giá trị qua các lớp filter dựa vào cách thức mà chúng ta thực hiện.

### 2.2.7. Ứng dụng của mạng CNN

Mạng nơron xoắn được áp dụng khá nhiều trong các bài toán nhận dạng như nhận dạng vật thể trong ảnh, nhận dạng chữ viết tay (chuyển đổi chữ viết trong hình ảnh thành văn bản thô trong máy tính), nhận dạng vật thể 3D, xử lý tiếng nói, xử lý ngôn ngữ tự nhiên, ... với độ chính xác trên 90%. Đã có rất nhiều bài báo khoa học đề xuất ra nhiều kiến trúc mạng nơron xoắn khác nhau cho các bài toán khác nhau như: LeNet, AlexNet, VGGNet, GoogleNet, ...

Tuy nhiên, sự phát triển của mạng nơron xoắn đến một ngưỡng nào đó bị chậm dần do những hạn chế của mô hình. Kích thước và độ sâu của mạng nơron khiến mạng nơron trở nên không linh hoạt. Giả sử ta đã huấn luyện thành công mô hình nhận diện 10 đối tượng khác nhau, một nhu cầu cần nhận diện đối tượng thứ 11 nảy sinh và để có thể nhận diện đối tượng thứ 1 này ta phải xây dựng một kiến trúc khác và huấn luyện lại từ đầu.

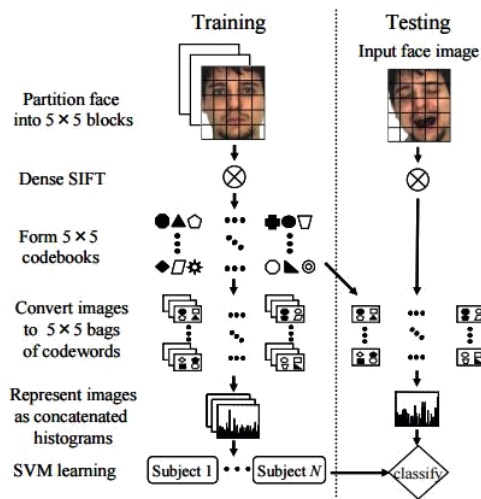
Dù vậy, trong thực tế, ta có thể chọn giải quyết các bài toán cụ thể mà nhu cầu mở rộng không quá lớn và không quá đòi hỏi độ linh hoạt cao (vd nhận diện các nhân viên cấp cao có quyền truy cập các hồ sơ quan trọng trong công ty, số lượng nhân viên cấp cao không quá nhiều và cũng không phải là thường xuyên thay đổi). Với những bài toán như vậy thì mạng nơron xoắn vẫn là một mô hình hiệu quả.

## CHƯƠNG 3 - CÁC CÔNG TRÌNH LIÊN QUAN

### 3.1. Nhận dạng khuôn mặt sử dụng Bag-of-Words

Nhóm tác giả [11] đề xuất một thuật toán khối Bag of Word để nhận dạng khuôn mặt bằng cách chia khuôn mặt thành nhiều khối đặc trưng SIFT, từ đó tính toán và lượng tử hóa vector thành các codeword khác nhau. Cuối cùng, ở mỗi khối ta tính tần số phân phối của mỗi codeword, sau đó nối dài các tần số từ các khối để biểu diễn khuôn mặt.

Tác giả đánh giá rằng các ảnh khuôn mặt đều cùng một loại vật thể, cho nên nếu ta trích xuất đặc trưng khuôn mặt bằng cách thành tập các phần nhỏ thì điều này không đảm bảo rõ thông tin của khuôn mặt. Do đó, nhóm đề xuất thuật toán rút trích đặc trưng khuôn mặt như Hình 20.



**Hình 20. Sơ đồ thuật toán Khối Bag of Word.**

Ta chia ảnh thành các khối  $5 \times 5$  và xem mỗi khối nhỏ là vùng quan tâm (ROI – Region of Interest). Với mỗi ROI, ta tính đặc trưng SIFT đặc trên mỗi đoạn lấy mẫu dài 2 điểm ảnh, thu được vector SIFT 128 chiều, từ đó, mỗi khối ta thu được một tập các vector SIFT. Ở bước huấn luyện, sử dụng thuật toán -means chuyển đổi vector SIFT ở mỗi ROI thành các codeword. Ở trong một ROI, ta phân vùng các đặc trưng SIFT ở mỗi đoạn thành cụm, khi đó ta định nghĩa codeword là tâm của cụm. Một codebook bao gồm codeword của cùng một ROI và từ dữ liệu huấn luyện, ta được  $5 \times 5$

codebook. Cuối cùng, ta đối chiếu mỗi vector SIFT của mỗi đoạn ở mỗi ROI với codebook tương ứng, sử dụng biểu đồ tần số của các codeword khác nhau và dùng biểu đồ này làm đặc trưng của ROI, sau đó ta nối dài  $5 \times 5$  biểu đồ để thu được một vector biểu diễn ảnh khuôn mặt. Sử dụng SVM tuyến tính để huấn luyện biểu đồ của từng người.

Bước kiểm tra, ta cũng chia ảnh thành  $5 \times 5$  khối, thu được  $5 \times 5$  biểu đồ codeword sử dụng codebook đã huấn luyện. Nối dài biểu đồ này thu được vector biểu diễn ảnh, ta phân loại ảnh bằng phân loại SVM với mô hình huấn luyện.

Nhóm tác giả sử dụng bộ dữ liệu AR và XM2VTS để thực nghiệm. Ảnh vào được nét xuống kích thước  $270 \times 230$ , nhóm thực nghiệm trên bộ dữ liệu AR với 119 đối tượng, huấn luyện trong bộ AR01, sử dụng bộ AR02 - AR08, AR11, AR15 - AR21 và AR24 để kiểm tra. Kết quả thực nghiệm thu được ở hình sau: với 3 trạng thái: Cảm xúc khuôn mặt (Facial expressions), Ánh sáng (Illumination), Che khuất (Occlusions).

Recognition results (%)															
Facial expressions						Illumination						Occlusions			
AR02	AR03	AR04	AR15	AR16	AR17	AR05	AR06	AR07	AR18	AR19	AR20	AR08	AR11	AR21	AR24
100	100	95.80	97.48	97.48	77.31	100	100	98.32	99.16	93.28	80.67	94.96	99.16	77.31	89.92

**Bảng 2. Kết quả thực nghiệm trên bộ dữ liệu AR**

### 3.2. Khoanh vùng một phần khuôn mặt sử dụng các mẫu đồng dạng

Trong nhận dạng khuôn mặt, đầu tiên cần phải xác định khuôn mặt trong bức ảnh. Các thuật toán xác định khuôn mặt thường trả về hộp chữ nhật bao quanh khuôn mặt, từ đó xác định và khoanh vùng các phần khuôn mặt như góc lông mày, góc mắt, đỉnh mũi, góc miệng, cằm, ... là những điểm đặc trưng chính của khuôn mặt. Tuy nhiên, có những điểm chính không nằm ở vị trí có độ dốc cao trong ảnh (ví dụ như đỉnh mũi), và việc xác định những điểm này đòi hỏi nhiều ảnh dữ liệu. Nhóm tác giả đề xuất một thuật toán khoanh vùng các điểm chính đã được định sẵn trước dưới nhiều điều kiện ảnh khác nhau. Trong bài báo, nhóm tác giả này sử dụng thuật toán cho ảnh khuôn mặt chân dung, chiếu sáng, cảm xúc, kiểu tóc, tuổi của đối tượng, mặt bị che một phần.



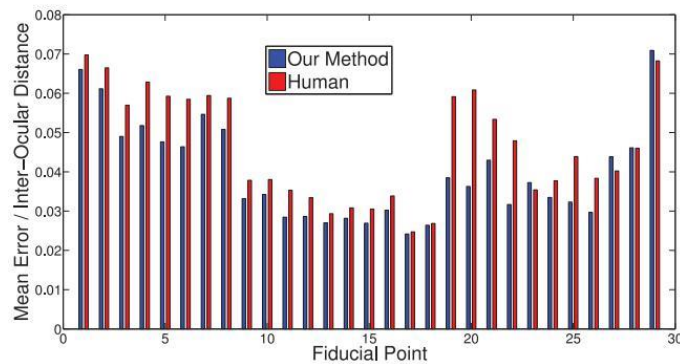
**Hình 21. Ảnh kết quả sau khi khoanh vùng của nhóm tác giả**

Nhóm tác giả [2] tạo bộ dữ liệu LFPW đã có sẵn các điểm chính chuẩn do công nhân từ công ty MTurk thực hiện thủ công (mỗi điểm có 3 công nhân đánh giá), sau đó thực nghiệm trên chính bộ dữ liệu này (không dùng điểm chính chuẩn), dùng 1100 ảnh huấn luyện và 300 ảnh kiểm tra. Ảnh huấn luyện dùng để huấn luyện bộ xác định dựa trên SVM và dùng để tính mô hình toàn cục. Sau đó, nhóm đánh giá kết quả sai số từng vị trí so với điểm chính chuẩn bằng cách tính tỉ lệ khoảng cách trung bình điểm chính thu được so với kết quả của 3 công nhân MTurk và khoảng cách đồng tử (đối với điểm chính thu được), và tỉ lệ giữa trung bình 3 điểm chính của 3 công nhân và khoảng cách đồng tử (đối với điểm chính chuẩn). Kết quả so sánh thuật toán xác định điểm chính so với khoảng cách trung bình của 3 công nhân MTurk, ta thấy rằng khoảng cách này thường lớn hơn khoảng cách của thuật toán đến trung bình điểm được làm thủ công. Cần lưu ý rằng các điểm mắt (9-18) chính xác nhất, còn điểm mũi và miệng (19-29) khá tệ, và cằm và long mày (1-8, 29) kém nhất. Có một số điểm lỗi, ví dụ như điểm chính ở cằm không chính xác, lỗi này xảy ra khi miệng mở.



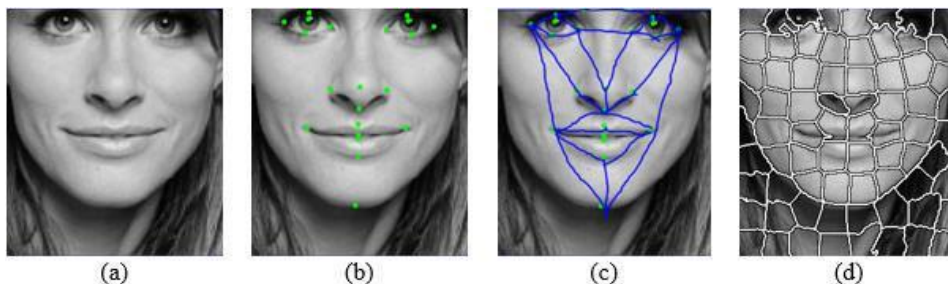
**Hình 22. kết quả xác định điểm chính trong bộ dữ liệu LFPW với một số điểm lỗi.**

Sai số trung bình của kết quả xác định điểm chính so với trung bình phương sai của điểm chính chuẩn được làm thủ công trên bộ dữ liệu LFPW. Kết quả cho thấy thuật toán đưa ra chính xác hơn.



**Hình 23. Sai số trung bình của kết quả**

Thuật toán này có khả năng tìm ra các điểm chính trên khuôn mặt dưới nhiều điều kiện ảnh khác nhau, do đó ta có thể sử dụng thuật toán này hỗ trợ cho việc nhận dạng khuôn mặt. Xem hình 24, từ ảnh đầu vào (a), đầu tiên ta xác định điểm chính trên khuôn mặt (b), từ đó tách khuôn mặt ra thành các phần mặt (c). Ngoài ra, từ khuôn mặt ban đầu, sử dụng superpixel (d), sau đó với mỗi phần mặt đã tách, chọn các superpixel nằm trong phần mặt này. Sau đó có thể áp dụng thuật toán nhận dạng khuôn mặt.



**Hình 24. thuật toán nhận dạng khuôn mặt**



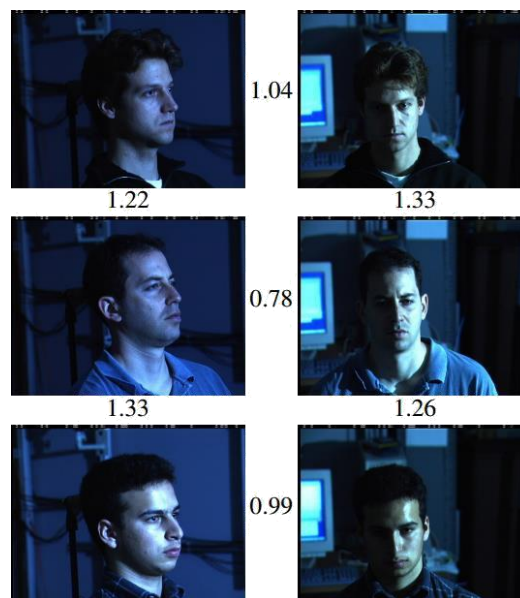
## CHƯƠNG 4 -MÔ HÌNH ĐỀ XUẤT NHẬN DẠNG KHUÔN MẶT BẰNG DEEP LEARNING SỬ DỤNG MẠNG FACENET VÀ THỰC NGHIỆM

### 4.1 . Mô hình mạng CNN cho bài toán nhận dạng khuôn mặt

#### Tóm Tắt

Nhóm tác giả từ Google [3] đề xuất một thuật toán có tên là FaceNet sẽ học cách ánh xạ từ ảnh khuôn mặt vào không gian Euclide compact với khoảng cách đo được tương ứng với độ tương đồng của khuôn mặt. Thuật toán này có thể tạo ra vector đặc trưng và nhúng vào bài toán nhận dạng khuôn mặt, kiểm tra khuôn mặt và phân cụm khuôn mặt. Nhóm tác giả sử dụng Mạng Tích Chập Sâu (Deep Convolution Neural Network - DCNN) được huấn luyện để tự tối ưu hóa bài toán. Mạng được huấn luyện sao cho khoảng cách L2 bình phương trong không gian nhúng tương ứng với mức độ tương đồng của khuôn mặt: Mặt cùng người sẽ có khoảng cách nhỏ, mặt khác người sẽ có khoảng cách lớn

Hình minh họa output khoảng cách khi sử dụng FaceNet giữa các cặp khuôn mặt. Nếu lấy ngưỡng là 1.1,ta thấy rằng 2 mặt có khoảng cách nhỏ hơn ngưỡng đều thuộc về một người (ví dụ như 2 ảnh ở dòng đầu tiên) và ngược lại.



**Hình 25.** Hình minh họa output khoảng cách khi sử dụng FaceNet



Sau khi thực hiện phép nhúng, thu được vector đặc trưng thì ta có thể thực hiện được 3 bài toán: Kiểm tra khuôn mặt, ta chỉ cần phân ngưỡng khoảng cách giữa 2 vector đặc trưng của 2 khuôn mặt. Nhận dạng khuôn mặt là bài toán phân loại k-NN. Phân cụm khuôn mặt sử dụng k-mean

Nhiều thuật toán nhận dạng khuôn mặt sử dụng DNN trước đây sử dụng lớp phân loại đã qua huấn luyện trên toàn bộ ảnh đã biết nhãn, sau đó lấy lớp thất cổ chai trung bình để biểu diễn tổng quát cho tập huấn luyện. Tuy nhiên, mặt tiêu cực là lớp thất cổ chai đôi khi không rõ và không tiện lợi do lớp thất cổ chai này thường rất lớn (khoảng 1000 chiều). Để khắc phục điều này, FaceNet huấn luyện output thành nhúng compact 128 chiều sử dụng hàm bộ ba sai số dựa trên Triplet Loss, mẫu bộ ba này gồm 2 ảnh cùng loại và 1 ảnh khác loại và hàm sai số có nhiệm vụ tách ảnh đúng ra khỏi ảnh sai dựa vào biên khoảng cách. Nhóm tác giả sử dụng 2 kiến trúc Mạng Tích Chập Sâu, một mạng dựa theo mô hình của Zeiler và Fergus, mạng còn lại sử dụng mô hình Inception từ GoogLeNet.

## 4.2 . Tổng quát hóa kích thước

### 4.2.1. Tổng quát hóa kích thước ảnh

Kích thước ảnh đầu vào 224x224 với 3 kênh màu



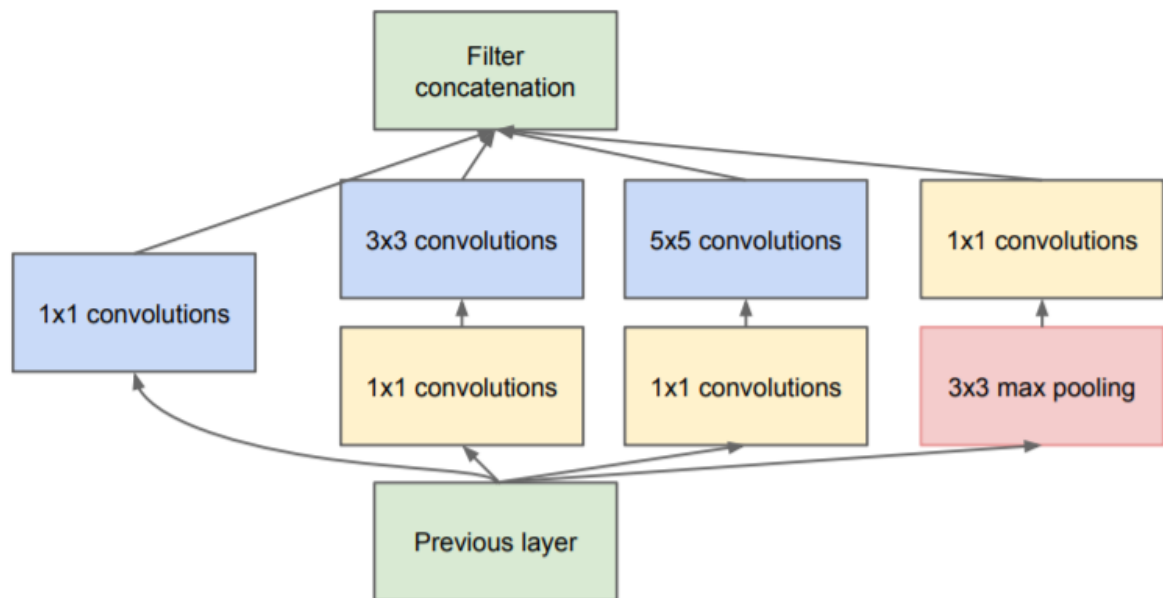
224px

224px

**Hình 26. Kích thước ảnh đầu vào**

#### 4.2.2. Tổng quát hóa kích thước bộ lọc của Convolution

Mỗi Inception module gồm 6 bộ lọc convolution gồm các bộ lọc 1x1, 3x3, 5x5. Với kích thước như ảnh ở dưới, mạng google lenet gồm 9 module như vậy số bộ lọc convolution là  $6 \times 9 = 54$  bộ lọc.



Hình 27. Filter concatenation

#### 4.2.3. Tổng quát hóa kích thước bộ lọc của max pooling

Trong mạng google lenet sử dụng 13 bộ lọc max pooling với kích thước mỗi bộ lọc là 3x3. Kích thước pooling luôn luôn 3x3 và tính song song với module tích chập trong mỗi module Inception

#### 4.2.4. Kích thước ảnh sau mỗi tầng

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

**Bảng 3. Kích thước ảnh sau mỗi tầng**

### 4.3. Phương pháp huấn luyện

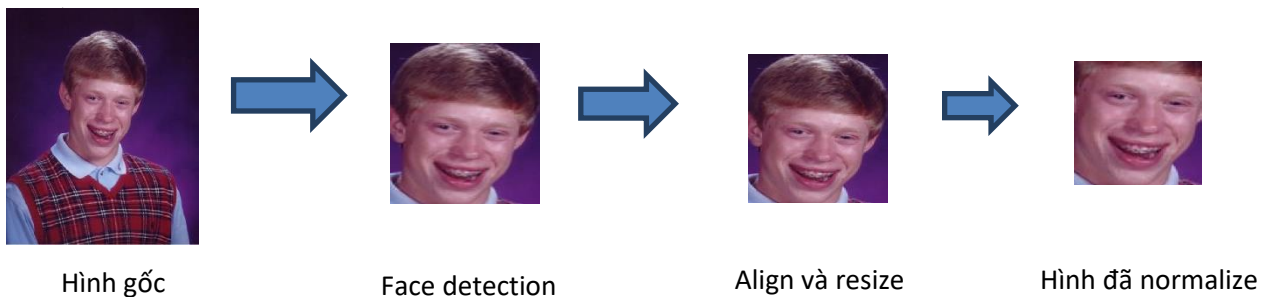
#### 4.3.1. Huấn luyện CNN cho việc extract feature

Facenet có bản chất one-shot learning, CNN chỉ đóng vai trò extract feature, việc huấn luyện CNN ở đây là khiến nó tách đặc trưng tốt hơn. Ta cũng có thể thực hiện huấn luyện một CNN với số đầu ra là số người, nhưng điều đó rất khó khăn, đòi hỏi thay đổi cấu trúc mạng và việc thu thập dataset cũng mất thời gian hơn, còn có thể mất cân bằng bộ dataset. Facenet, thực chất là một CNN có nhiệm vụ tách các đặc trưng của một ảnh mặt. Điểm đặc biệt tạo nên sự khác biệt của Facenet là nó sử dụng hàm lỗi Triplet để tối thiểu hóa khoảng cách giữa các gương mặt tương đồng và tối đa hóa khoảng cách đến những gương mặt không tương đồng, vì vậy facenet có thể phân biệt rất chính xác người với người. Quá trình train CNN được thực hiện trên các bộ dataset lớn (ví dụ như vggface hoặc MS 1 million), kết quả ta muốn là tập embedding vector ra phải tách nhau ra thành từng cụm để thực hiện phân loại bằng kNN hoặc SVM. Quá trình training dựa trên triplet loss như sau: Tiền xử lý data: Chú ý là các bộ dataset kể trên thường có dạng là các folder với tên là tên của một người, trong folder

là hình của người đó, mỗi người tầm 100-1000 hình, một bộ data thì khoảng vài nghìn người.

1. Sử dụng một bộ face detector [4] để tách phần ảnh có gương mặt người ra. Sau đó ta resize hình lại một kích cỡ xác định (tùy chọn), facenet thì là 160x160, gần đây thì arcface dùng cỡ 112x112, việc chọn cỡ này là tùy theo yêu cầu về phần cứng hoặc độ chính xác.

Ta sử dụng thuật toán face detection MTCNN (Multi Cascade Convolution Neural Network)[6] để tách phần ảnh có chứa gương mặt ra, sau đó sử dụng các hàm của OpenCV để điều chỉnh kích thước ảnh về 160x160 sau đó chiếu không gian ảnh vào miền  $[-1,1]$  bằng cách normalize hình.



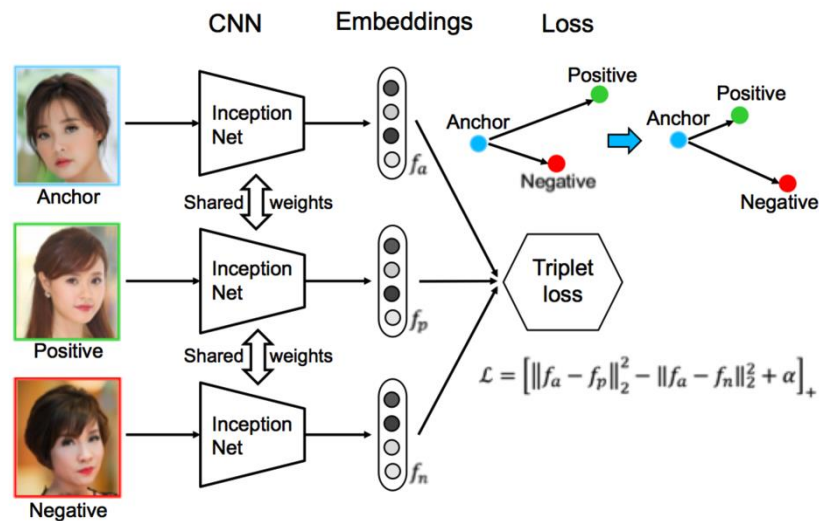
**Hình 28. Tiền xử lý ảnh**

2. Thực hiện các bước “lọc” bộ dataset. Thực tế là các bộ dataset lớn như vggface hay ms 1m đều có nhiều “nhiều”, tức các hình không chứa gương mặt hoặc gương mặt khác với người cần xét. Vì vậy có thể dùng một mô hình nhận diện gương mặt khác để tìm và loại bỏ những hình “nhiều” ấy. Hoặc đơn giản là ta có thể lấy clean list (danh sách các hình chuẩn) đã được lọc sẵn.

3. Thực hiện bước normalize cho tập dataset đã chuẩn bị. Cách normalize cũng giống như cách normalize cho CNN bình thường. Các bước train: Đầu vào là tập dataset đã chuẩn bị, CNN có thể là các CNN như inception, mobilenet,... với đầu ra là embedding vector như đã nói, embedding vector này có thể có 128 chiều hoặc 512 chiều,... tùy ý. Tóm lại ta chỉ cần lấy cấu trúc inception hoặc mobilenet, bỏ lớp cuối và thêm vào lớp embedding vector (không có activation)

4. Forward pass trên tập data, lưu các embedding vector đã tính trên minibatch lại.

5. Chia minibatch thành 3 phần là positive, negative và anchor, trong đó anchor là phần “neo” giống như “tâm” của cụm embedding vector, sau đó chọn phần positive là những embedding vector là cùng người với anchor, negative là khác người



**Hình 29. Mô phỏng quá trình tính toán tripletloss**

6. Tính Triplet loss là khoảng cách giữa các embedding vector anchor và positive trừ cho khoảng cách giữa các embedding vector anchor và negative (tức ta tìm cách làm giảm khoảng cách từ các vector positive tới anchor và ngược lại đẩy các vector negative đi xa khỏi anchor)

7. Sử dụng thuật toán backpropagation để huấn luyện CNN theo hàm lỗi triplet

#### 4.3.2. Huấn luyện mô hình phân loại

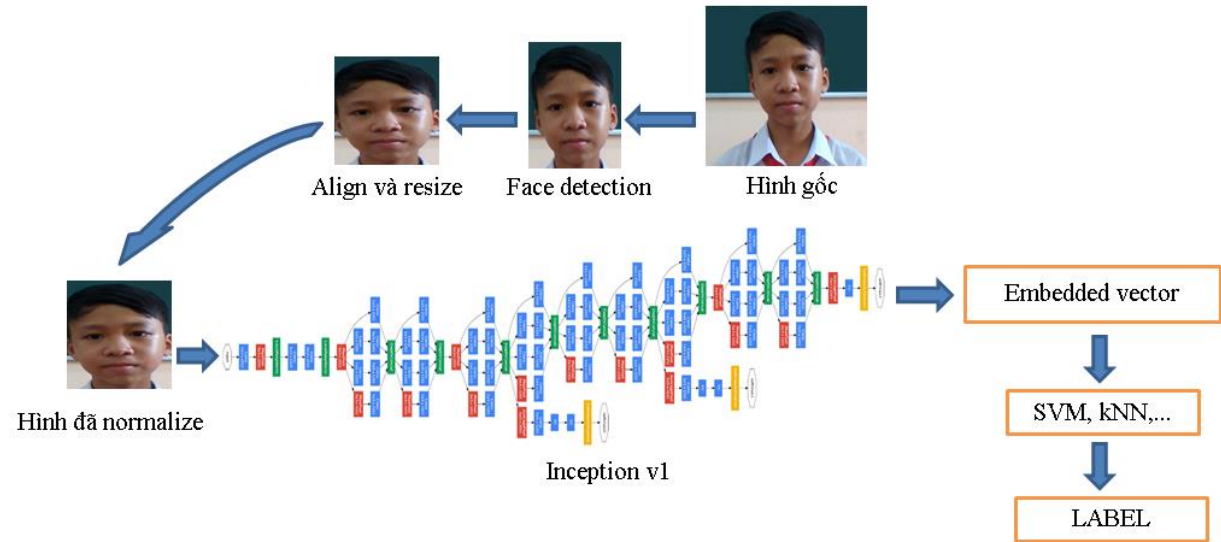
1. Sử dụng một model face detector [5] để tìm bounding box của gương mặt chú ý là face detector phải giống với face detector được dùng cho quá trình train mới cho kết quả chính xác

2. Cắt phần gương mặt đã tìm được ra, resize lại và thực hiện prewhiten (bước normalize data)

3. Đưa vào CNN đã train để tính ra được embedding vector.

4. Dùng kNN, SVM, hoặc so sánh khoảng cách để tìm “cụm” mà embedding vector đó thuộc về, từ đó suy ra danh tính người được chọn

#### 4.4. Thực hiện Thuật Toán



Hình 30. Sơ đồ tổng quát thuật toán

##### Bước 1. Tiền xử lý

Thực hiện như bước huấn luyện trên cho 8000 ảnh trên 200 học sinh, mỗi học sinh sau khi xử lý sẽ được đưa vào mỗi thư mục được gán nhãn với tên và lớp của học sinh

##### Bước 2. Tính embedding vector

Hình ảnh đã được tiền xử lý như ở bước 4.3.1 được đưa vào CNN đã huấn luyện, kết quả ra là embedding vector của gương mặt ấy.

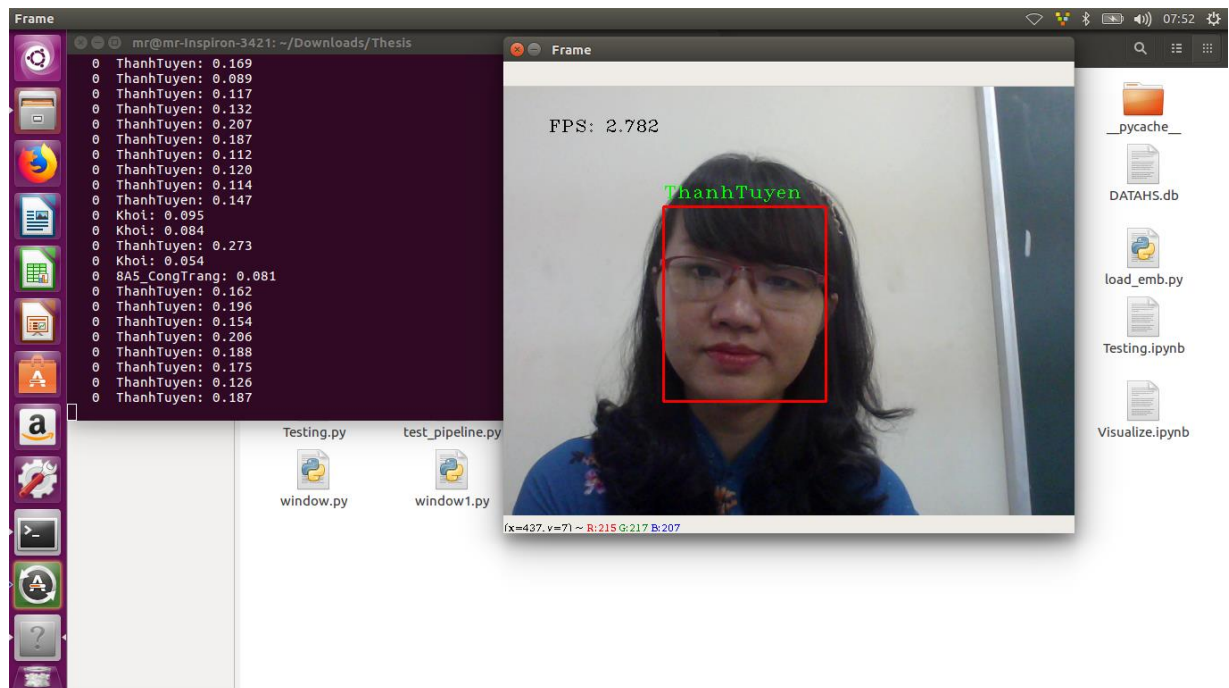
##### Bước 3. Thực hiện xác định gương mặt

Sau khi có embedding vector, việc tiếp theo ta dùng embedding vector này đưa vào mô hình SVM hoặc kNN đã được huấn luyện dựa trên tập embedding vector để phân biệt các gương mặt với nhau.

## 4.5. Thực Nghiệm

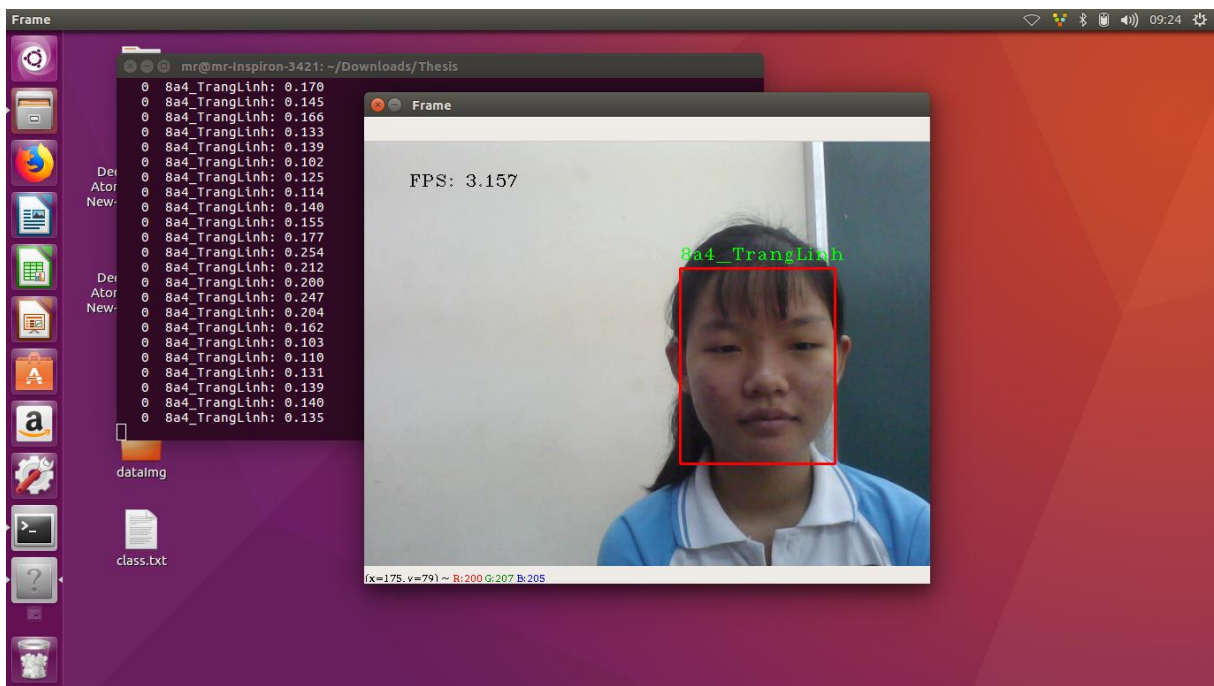
Tôi thực nghiệm trên 8000 ảnh khuôn mặt từ 200 đối tượng học sinh, sau khi cắt phần khuôn mặt trong ảnh để tạo thành ảnh khuôn mặt, tôi thay đổi kích thước ảnh khuôn mặt từ  $96 \times 96$  điểm ảnh đến  $160 \times 160$  điểm ảnh.

Với bài toán nhận dạng khuôn mặt, sau khi huấn luyện thu được vector đặc trưng 128 chiều thì ta sử dụng phân loại k-NN. Đánh giá trên bộ dữ liệu LFW với 8000 ảnh khuôn mặt từ 200 học sinh và thu được độ chính xác 92.5%

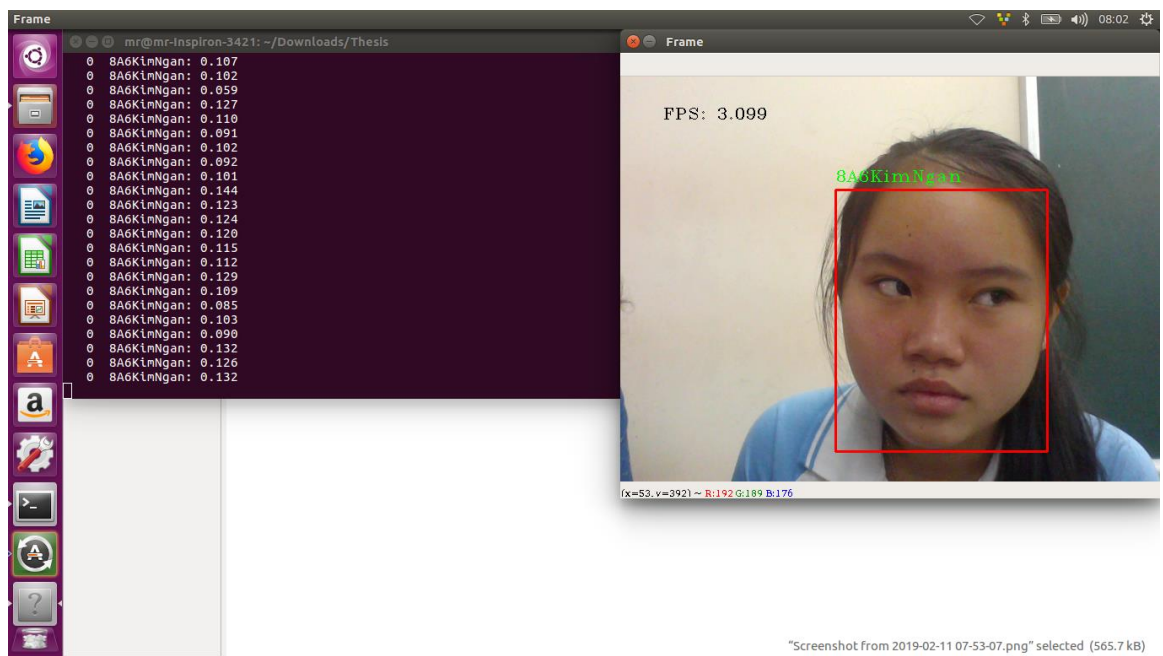


Hình 31. Thực nghiệm trên hệ điều hành Ubuntu



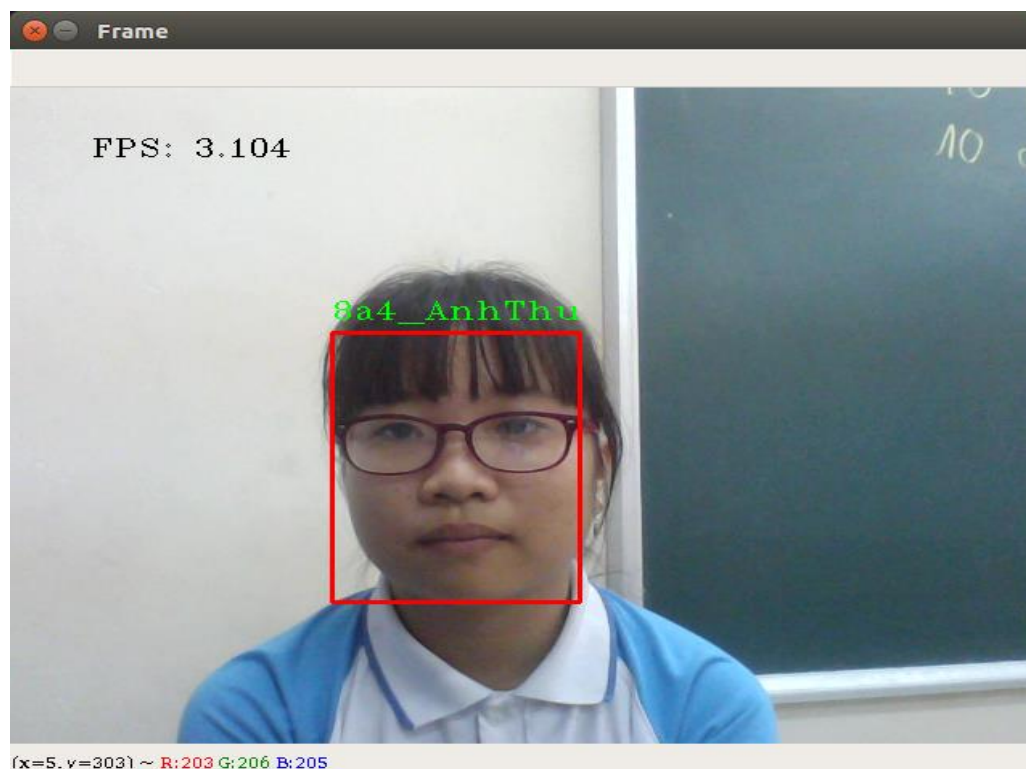


Hình 32. Thực nghiệm nhận dạng khuôn mặt học sinh

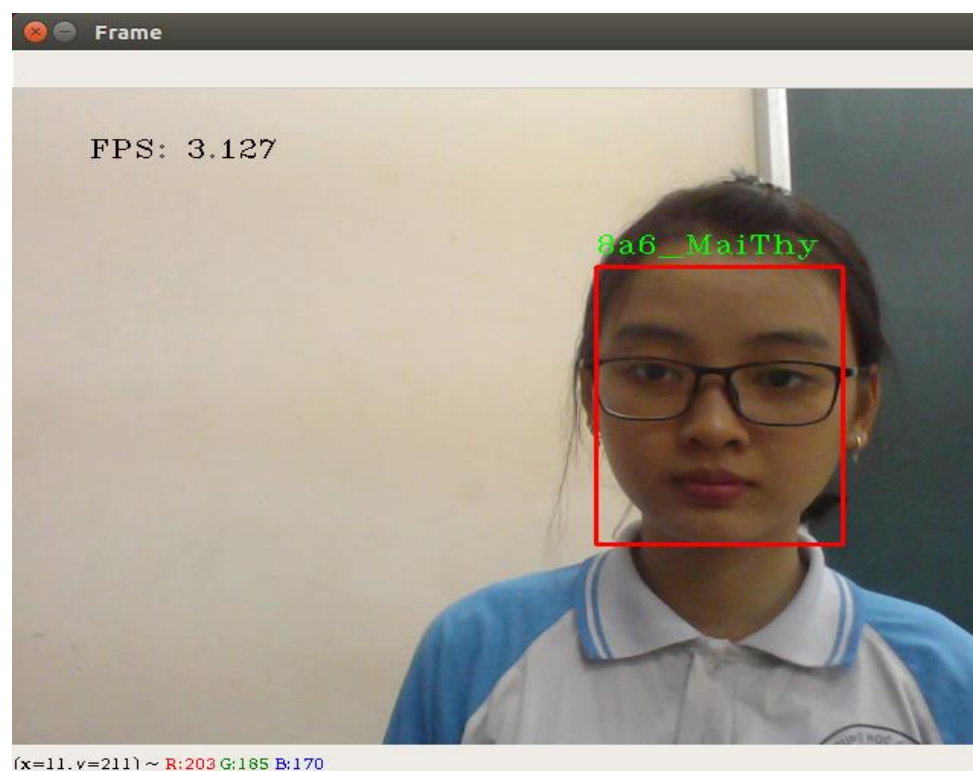


Hình 33. Thực nghiệm nhận dạng khuôn mặt học sinh

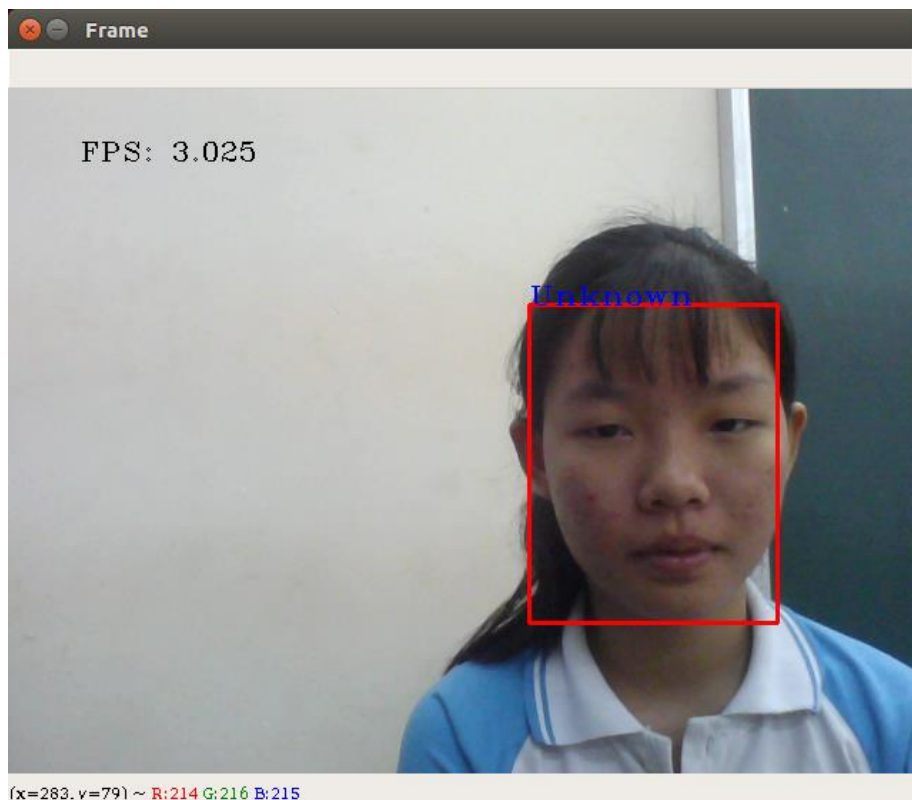




Hình 34. Thực nghiệm nhận dạng khuôn mặt học sinh



Hình 35. Thực nghiệm nhận dạng khuôn mặt học sinh



**Hình 36. Thực nghiệm nhận dạng khuôn mặt học sinh**

## **4.6. Ưu và Nhược Điểm của Thuật Toán**

### **4.6.1. Ưu Điểm**

Tính đến thời điểm FaceNet ra đời, thuật toán này đã lập nên kỷ lục mới trong nhận dạng khuôn mặt dưới nhiều điều kiện ảnh khác nhau

### **4.6.2. Nhược Điểm**

FaceNet huấn luyện với một số lượng lớn hình ảnh (hơn 200 triệu ảnh của 8 nghìn đối tượng), lớn gấp 3 lần so với các bộ dữ liệu hiện có. Để xây dựng bộ dữ liệu lớn như vậy rất khó thực hiện trong các phòng thiết bị, học thuật do đòi hỏi kiến trúc máy lớn.

## **4.7. Nhận Xét Thuật Toán**

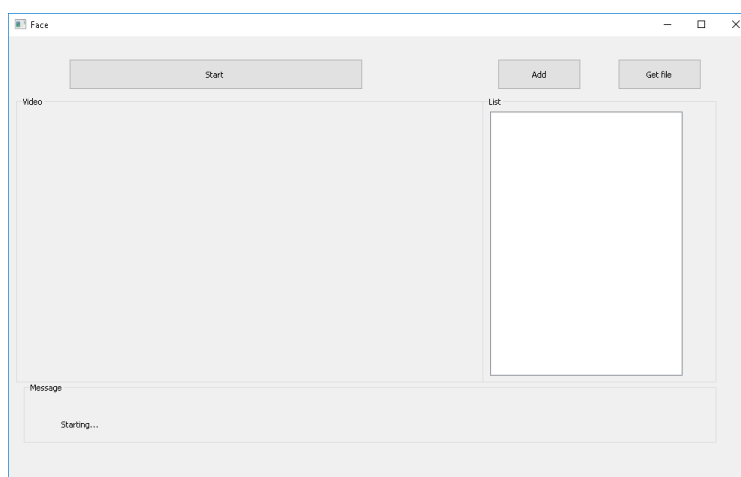
Thuật toán FaceNet sử dụng bộ ba sai số và CNN để huấn luyện, có thể sử dụng ý tưởng này vào đề tài. Tuy nhiên, vấn đề gặp phải là ta không có đủ thiết bị để huấn

luyện triệu ảnh như FaceNet. Do đó, thay vì huấn luyện trên toàn bộ khuôn mặt, ta có thể huấn luyện từng phần trên khuôn mặt dựa trên ý tưởng trình bày ở mục 2.3.5.

## 4.8. Hiển thị kết quả và xuất file

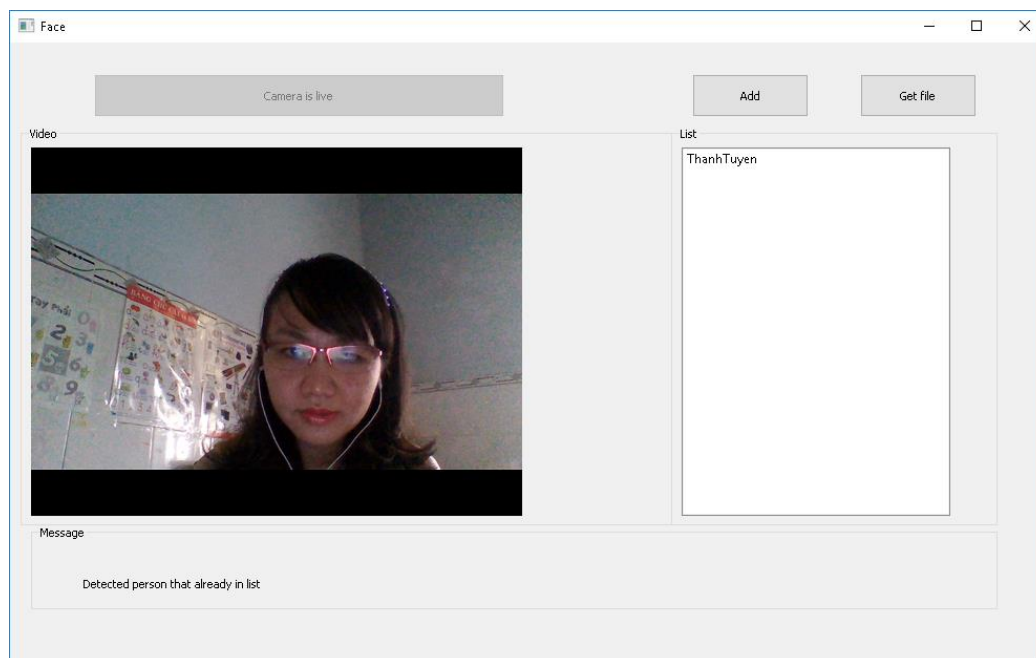
### 4.8.1. Giao diện chính của ứng dụng

Lập trình giao diện với PyQt5

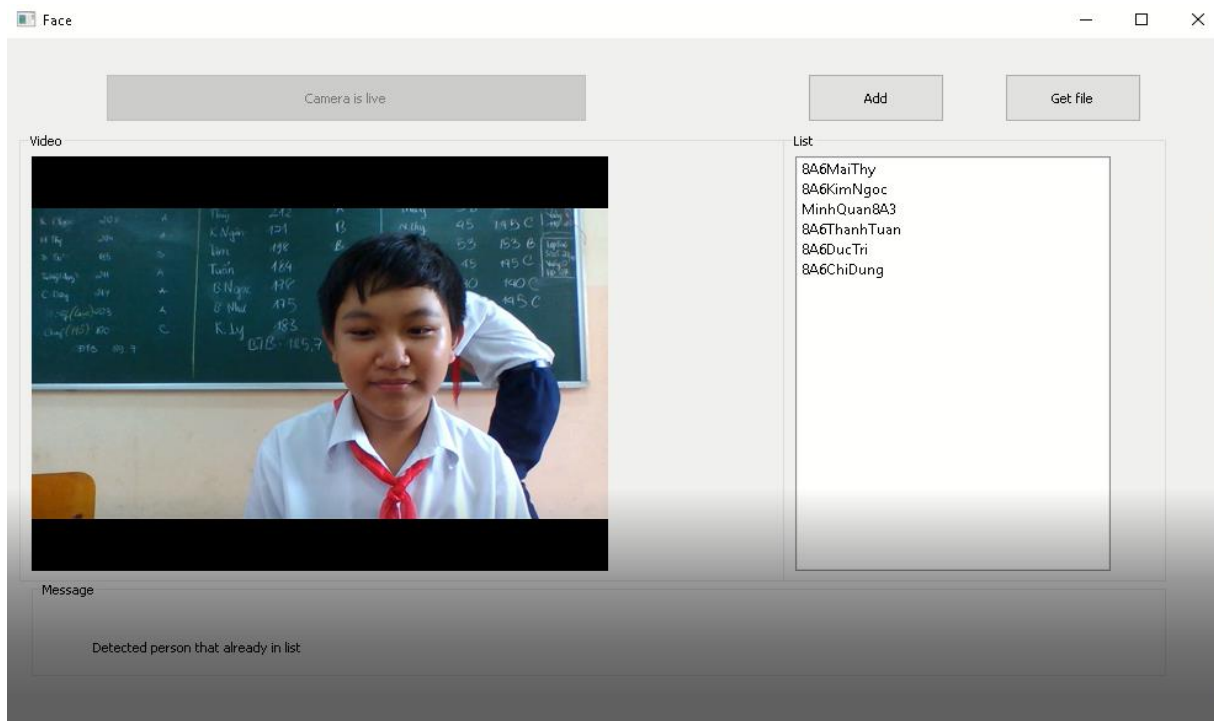


**Hình 37. Giao diện chính của ứng dụng**

Nháy nút Start để kiểm tra kết quả



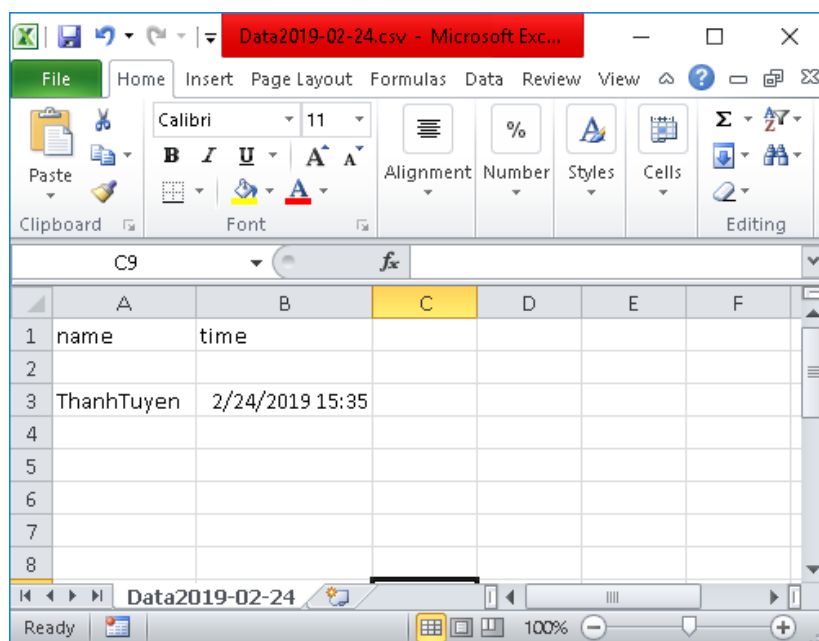
**Hình 38. Thực nghiệm nhận dạng khuôn mặt và in kết quả trên giao diện**



**Hình 39. Thực nghiệm nhận dạng khuôn mặt và in kết quả trên giao diện**

#### 4.8.2. Xuất kết quả ra file Excel

Sau khi nhận dạng xong ứng dụng hiện tên người bên khung bên phải, để xuất ra file Excel ta nhấn nút Get File.



**Hình 40. Xuất kết quả ra file Excel**

	A	B	C	D	E	F	G	H	I	J	K	L
1	name	time										
2	8A6ThanhLap	4/13/2019 16:10										
3	8A6PhuongThuy	4/13/2019 16:11										
4	8A6KimNgoc	4/13/2019 16:11										
5	8A6ManVy	4/13/2019 16:12										
6	8A6HuynhNhu	4/13/2019 16:12										
7	8A6NgocNgan	4/13/2019 16:12										
8	8A6TuongVy	4/13/2019 16:12										
9	8A6MaiThy	4/13/2019 16:13										
10	GiaBao8A3	4/13/2019 16:13										
11	8A6NgocHien	4/13/2019 16:14										
12	8A6TuanVy	4/13/2019 16:15										
13	8A6NhuQuynh	4/13/2019 16:15										
14	8A6KimNgan	4/13/2019 16:15										
15	8A6BaoNhu	4/13/2019 16:16										
16	8A6ThaoNhi	4/13/2019 16:16										
17	8A6BaoAnh	4/13/2019 16:16										
18	Khoa	4/13/2019 16:16										

Hình 41. Xuất kết quả ra file Excel

## KẾT LUẬN

### \* Kết quả đạt được

Sau khi thực nghiệm trên ứng dụng điểm danh học sinh bằng công nghệ nhận dạng ảnh, ứng dụng có thể giúp trường điểm danh trực tiếp học sinh bằng công nghệ nhận dạng khuôn mặt tương đối chính xác, bắt đầu đưa cuộc cách mạng công nghiệp 4.0 vào cuộc sống.

### \* Tồn tại và hạn chế

Thuật toán FaceNet sử dụng bộ ba sai số và CNN để huấn luyện, có thể sử dụng ý tưởng này vào đề tài. Tuy nhiên, vấn đề gặp phải là ta không có đủ thiết bị để huấn luyện triệu ảnh như FaceNet.

#### - Ưu điểm:

Tính đến thời điểm FaceNet ra đời, thuật toán này đã lập nên kỷ lục mới trong nhận dạng khuôn mặt dưới nhiều điều kiện ảnh khác nhau

#### - Nhược điểm:

FaceNet huấn luyện với một số lượng lớn hình ảnh (hơn 200 triệu ảnh của 8 triệu đối tượng), lớn gấp 3 lần so với các bộ dữ liệu hiện có. Để xây dựng bộ dữ liệu lớn như vậy rất khó thực hiện trong các phòng thiết bị, học thuật do đòi hỏi kiến trúc máy lớn.

### \* Hướng phát triển

Thuật toán FaceNet sử dụng bộ ba sai số và CNN để huấn luyện, có thể sử dụng ý tưởng này vào đề tài. Tuy nhiên, vấn đề gặp phải là ta không có đủ thiết bị để huấn luyện triệu ảnh như FaceNet.

Đề tài có thể được phát triển thành một phần mềm nhận dạng mặt người tốt hơn, bằng cách kết hợp với một số thuật toán nhận dạng phụ thuộc rất nhiều vào tập huấn luyện, vị trí các khuôn mặt trong hình. Cho ra kết quả chính xác hơn.

## CÔNG NGHỆ SỬ DỤNG

### \* Ngôn ngữ lập trình Python

Python là ngôn ngữ lập trình thông dịch, được thiết kế chú trọng vào tính dễ đọc của đoạn mã và cho phép lập trình viên có thể diễn tả cái khái niệm với chỉ vài dòng lệnh. Python sử dụng hệ thống kiểu động (dynamic type system), cơ chế cấp phát bộ nhớ tự động và hỗ trợ nhiều mô hình lập trình như lập trình hướng đối tượng, lập trình hàm và lập trình thủ tục.

Tôi sử dụng Python phiên bản 3.6 làm ngôn ngữ lập trình chính vì ngoài những ưu điểm ở trên, Python còn là ngôn ngữ được rất nhiều framework về học sâu hỗ trợ (Tensorflow, Keras,...).

### \* Các Framework và library

#### - *TensorFlow*

TensorFlow là một thư viện mã nguồn mở dùng cho việc tính toán số học sử dụng mô hình đồ thị luồng dữ liệu (data flow graphs). Các node trong đồ thị biểu thị cho một phép tính toán, các cạnh của đồ thị biểu diễn các mảng đa chiều chứa dữ liệu (còn gọi là tensor) được truyền qua các cạnh đó. Kiến trúc đặc biệt này giúp cho việc triển khai các tác vụ tính toán một cách linh hoạt trên một hay nhiều CPU hoặc trên các GPU trong một máy tính cá nhân, hệ thống máy chủ hay thiết bị di động mà không cần viết lại mã nguồn.

TensorFlow còn hỗ trợ rất mạnh trong việc xây dựng và sử dụng các mô hình học máy và học sâu. Tôi sử dụng TensorFlow phiên bản 1.12 (thông qua API cấp cao là Keras) để xây dựng các mô hình và thực hiện huấn luyện trên dữ liệu.

#### - *Keras*

Keras là một thư viện mã nguồn mở dùng cho Deep Learning, có thể chạy trên nền của Theano hoặc Tensorflow.

Keras được thiết kế để hỗ trợ hiện thực các mô hình Deep Learning nhanh nhất có thể phục vụ cho nghiên cứu và phát triển ứng dụng. Keras được phát triển với 4 nguyên tắc chính:

- Mô đun: Một mô hình có thể được hiểu là một chuỗi hoặc một đồ thị riêng lẻ. Các thành phần của một mô hình học sâu trong Keras được tách thành các phần riêng và có thể kết hợp tùy ý lại với nhau một cách dễ dàng để hình thành nên một mô hình mới.

- Thân thiện với người dùng: Keras là một API được thiết kế cho con người. Keras đặt trải nghiệm người dùng lên hàng đầu, nó đưa ra các API đồng nhất và đơn giản, tối thiểu hóa số thao tác cần thiết cho các tác vụ phổ biến.

- Dễ dàng mở rộng: Các mô đun mới có thể dễ dàng được thêm vào và sử dụng bên trong thư viện, điều này đặc biệt hữu dụng đối với các nhà nghiên cứu muốn thử nghiệm và khám phá các ý tưởng mới.

- Làm việc với Python: Thư viện được viết hoàn toàn bằng Python. Các mô hình được miêu tả bằng Python, với các đặc tính nhỏ gọn, dễ dàng kiểm tra lỗi và cho phép dễ dàng mở rộng.

Keras chú trọng vào ý tưởng của một mô hình. Kiểu của một mô hình được gọi là Sequence - một ngăn xếp (stack) các tầng (layer).

#### **- *Scikit-learn***

Scikit-learn (viết tắt là sklearn) là một thư viện mã nguồn mở trong ngành machine learning, rất mạnh mẽ và thông dụng với cộng đồng Python, được thiết kế trên nền NumPy và SciPy. Scikit-learn chứa hầu hết các thuật toán machine learning hiện đại nhất, đi kèm với comprehensive documentations. Điểm mạnh của thư viện này là nó được sử dụng phổ biến trong academia và industry, do đó nó luôn được updated và có một very active user community.

#### **- *PyQt***



Qt là một Application framework đa nền tảng viết trên ngôn ngữ C++ , được dùng để phát triển các ứng dụng trên desktop, hệ thống nhúng và mobile. Hỗ trợ cho các platform bao gồm : Linux, OS X, Windows, VxWorks, QNX, Android, iOS, BlackBerry, Sailfish OS và một số platform khác. PyQt là Python interface của Qt, kết hợp của ngôn ngữ lập trình Python và thư viện Qt, là một thư viện bao gồm các thành phần giao diện điều khiển (**widgets , graphical control elements**).

PyQt API bao gồm các module bao gồm số lượng lớn với các **classes** và **functions** hỗ trợ cho việc thiết kế ra các giao diện giao tiếp với người dùng của các phần mềm chức năng. Hỗ trợ với Python 2.x và 3.x.

PyQt được phát triển bởi Riverbank Computing Limited

### **Công cụ và thiết bị**

#### ***Anaconda***

Anaconda là nền tảng (platform) mã nguồn mở về Khoa học dữ liệu (Data Science) trên Python thông dụng nhất hiện nay. Với hơn 6 triệu người dùng, Anaconda là cách nhanh nhất và dễ nhất để học Khoa học dữ liệu với Python hoặc R trên Windows, Linux và Mac OS X.

#### ***Visual Studio Code***

Visual Studio Code là sản phẩm của Microsoft, ra mắt vào tháng 4 năm 2015 ở hội nghị Build. Đặc điểm nổi bật là đơn giản, gọn nhẹ, dễ dàng cài đặt. Visual Studio Code có thể cài đặt được trên cả Windows, Linux và Mac OS và hỗ trợ nhiều ngôn ngữ

## TÀI LIỆU THAM KHẢO

- [1] Phan Văn Hiền, 2013, Giáo trình Mạng neural - Trường Đại học Bách Khoa Đà Nẵng
- [2] P. N. Belhumeur, D. W. Jacobs, D. J. Kriegman and N. Kumar, Localizing parts of faces using a consensus of exemplars, IEEE transactions on pattern analysis and machine intelligence, vol. 35(12), pp. 2930-2940, 2013.
- [3] F. Schroff, D. Kalenichenko and J. Philbin, Facenet: A unified embedding for face recognition and clustering, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 815-823, 2015.
- [4] <https://viblo.asia/p/tim-hieu-mtcnn-va-ap-dung-de-xac-dinh-vi-tri-cac-khuon-mat-3Q75wkO75Wb>
- [5] <https://towardsdatascience.com/face-detection-neural-network-structure-257b8f6f85d1>
- [6] <https://towardsdatascience.com/mtcnn-face-detection-cdcb20448ce0>
- [7] CS231n. Convolutional neural networks for visual recognition.  
<http://cs231n.github.io/convolutional-networks>
- [8] <http://machinelearningcoban.com/2017/03/04/overfitting>
- [9] <http://nhiethuyettre.me/mang-no-ron-tich-chap-convolutional-neural-network>
- [10] Adit Deshpande (July 20, 2016). A Beginner's Guide To Understanding Convolutional Neural Networks, <https://adeshpande3.github.io/A-Beginner'sGuide-To-Understanding-Convolutional-Neural-Networks/>
- [11] C.-F. Tsai, "Bag-of-words representation in image annotation: A review.," ISRN Artificial Intelligence 2012, 2012.