

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA CÔNG NGHỆ THÔNG TIN**

----------------------------------------

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC

**Đề tài: PHÁT TRIỂN HỆ THỐNG ĐIỂM DANH TRONG LỚP HỌC SỬ DỤNG CÔNG NGHỆ NHẬN DIỆN KHUÔN MẶT**

| **Giảng viên hướng dẫn:** | ThS. Nguyễn Văn B |
| --- | --- |
| **Sinh viên thực hiện:** | Nguyễn Văn A |
| **Mã sinh viên:** | BXXDCCNYYY |
| **Lớp:** | DXHTTTY |
| **Niên khóa:** | 20xx-20xx |
| **Hệ đào tạo:** | Đại học chính quy |

**Hà Nội 202x**



**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA CÔNG NGHỆ THÔNG TIN**

--------------------------------------

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC

**Đề tài: PHÁT TRIỂN HỆ THỐNG ĐIỂM DANH TRONG LỚP HỌC SỬ DỤNG CÔNG NGHỆ NHẬN DIỆN KHUÔN MẶT**

| **Giảng viên hướng dẫn:** | ThS. Nguyễn Văn B |
| --- | --- |
| **Sinh viên:** | Nguyễn Văn A |
| **Mã sinh viên:** | BXXDCCNYYY |
| **Lớp:** | DXHTTTY |
| **Niên khóa:** | 20xx-20xx |
| **Hệ đào tạo:** | Đại học chính quy |

**Hà Nội 202x**

**NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN**

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

…………………………………………………………………………………………

**Điểm: (Bằng chữ: )**

Hà Nội, ngày tháng năm 20…

**Giảng viên hướng dẫn**

**NHẬN XÉT CỦA GIẢNG VIÊN PHẢN BIỆN**

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

…………………………………………………………………………………………

**Điểm: (Bằng chữ: )**

Hà Nội, ngày tháng năm 20…

**Giảng viên phản biện**

**LỜI CẢM ƠN**

Đây là mục tùy chọn, nên viết phần cảm ơn ngắn gọn, tránh dùng các từ sáo rỗng, giới hạn trong khoảng 100-150 từ.

**Hà Nội, tháng 1 năm 202x**

**Sinh viên**

**Nguyễn Văn B**

**MỤC LỤC**

[**LỜI MỞ ĐẦU 7**](#_heading=h.4d34og8)

[**CHƯƠNG 1: TÌM HIỂU BÀI TOÁN ĐIỂM DANH SỬ DỤNG NHẬN DIỆN KHUÔN MẶT 1**](#_heading=h.17dp8vu)

[**1.1.**](#_heading=h.3rdcrjn) **Tổng quan về bài toán 1**

[***1.1.1.***](#_heading=h.26in1rg) ***Bài toán điểm danh bằng khuôn mặt 1***

[***1.1.2.***](#_heading=h.lnxbz9) ***Những khó khăn của việc điểm danh bằng khuôn mặt 1***

[**1.2.**](#_heading=h.35nkun2) **Cơ sở lý thuyết và tổng quan về mạng nơ-ron tích chập (Convolution Neural Network) trong Deep Learning 2**

[***1.2.1.***](#_heading=h.1ksv4uv) ***Mạng nơ-ron nhân tạo (Neural Network) 2***

[***1.2.2.***](#_heading=h.1y810tw) ***Mạng nơ-ron tích chập (Convolution Neural Network) 6***

[**a.**](#_heading=h.2xcytpi) **Kiến trúc cơ bản của mạng nơ-ron tích chập 7**

[**b.**](#_heading=h.2p2csry) **Một số kiến trúc mạng nơ-ron tích chập 12**

[**CHƯƠNG 2: KHẢO SÁT CÁC CÔNG NGHỆ XỬ LÝ ẢNH, HỌC MÁY TRONG HỆ THỐNG NHẬN DIỆN GƯƠNG MẶT 17**](#_heading=h.41mghml)

[***Tổng quan hệ thống nhận diện khuôn mặt 17***](#_heading=h.2grqrue)

[**2.1.**](#_heading=h.3fwokq0) **Bộ phát hiện khuôn mặt (Face Detector) 18**

[***❖***](#_heading=h.4f1mdlm) ***Detect dựa trên các hand-craft feature (đặc trưng cứng) 18***

[***❖***](#_heading=h.19c6y18) ***Detect dựa trên các feature được trích chọn từ mạng CNN 19***

[***Một số mô hình phát hiện khuôn mặt dựa trên mạng CNN. 19***](#_heading=h.28h4qwu)

[**2.2.**](#_heading=h.2lwamvv) **Bộ căn chỉnh khuôn mặt (Face Alignment) 24**

[**2.3.**](#_heading=h.4k668n3) **Bộ chống giả mạo khuôn mặt (Face Anti-spoofing) 25**

[***Một số phương pháp chính cho bài toán chống tấn công giả mạo khuôn mặt: 26***](#_heading=h.1egqt2p)

[***Một số nghiên cứu mạng CNN cho bài toán chống giả mạo khuôn mặt: 27***](#_heading=h.3ygebqi)

[**2.4.**](#_heading=h.3cqmetx) **Bộ trích xuất đặc trưng khuôn mặt (Face Feature Extraction) 29**

[***Một số Loss Function: 30***](#_heading=h.2r0uhxc)

[***❖***](#_heading=h.3q5sasy) ***ArcFace (Additive Angular Margin Loss) 31***

[**2.5.**](#_heading=h.kgcv8k) **Bộ nhận diện khuôn mặt 32**

[***Một số phương pháp tính khoảng cách: 32***](#_heading=h.34g0dwd)

[***❖***](#_heading=h.43ky6rz) ***Cosine Similarity 33***

[***❖***](#_heading=h.xvir7l) ***Hamming Distance 33***

[**CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG 35**](#_heading=h.1x0gk37)

[**3.1.**](#_heading=h.4h042r0) **Phân tích hệ thống 35**

[***3.1.1.***](#_heading=h.2w5ecyt) ***Mô tả các chức năng chính của hệ thống: 35***

[***3.1.2.***](#_heading=h.1baon6m) ***Use Case tổng quan của hệ thống: 36***

[***3.1.3.***](#_heading=h.2afmg28) ***Đặc tả các Use Case của hệ thống: 36***

[**3.2.**](#_heading=h.j8sehv) **Thiết kế hệ thống 50**

[***3.2.1.***](#_heading=h.338fx5o) ***Biểu đồ lớp của hệ thống 50***

[***3.2.2.***](#_heading=h.42ddq1a) ***Biểu đồ CSDL 50***

[***3.2.3.***](#_heading=h.wnyagw) ***Thiết kế vật lý của hệ thống 52***

[**3.3.**](#_heading=h.2uxtw84) **Tổng quan chức năng điểm danh nằng khuôn mặt 53**

[**CHƯƠNG 4: THỬ NGHIỆM CÀI ĐẶT VÀ XÂY DỰNG ỨNG DỤNG 57**](#_heading=h.3ls5o66)

[**4.1.**](#_heading=h.20xfydz) **Cài đặt Server 57**

[**4.2.**](#_heading=h.4kx3h1s) **Cài đặt Client 57**

[**4.3.**](#_heading=h.302dr9l) **Thiết kế giao diện 58**

[***4.3.1.***](#_heading=h.1f7o1he) ***Giao diện đăng nhập 58***

[***4.3.2.***](#_heading=h.2eclud0) ***Giao diện danh sách lớp học 59***

[***4.3.3.***](#_heading=h.3dhjn8m) ***Giao diện điểm danh 60***

[***4.3.4.***](#_heading=h.4cmhg48) ***Giao diện kết quả điểm danh 61***

[***4.2.5.***](#_heading=h.16x20ju) ***Giao diện đăng nhập của Admin 62***

[***4.2.6.***](#_heading=h.261ztfg) ***Giao diện thêm mới tài khoản 62***

[***4.2.7.***](#_heading=h.356xmb2) ***Giao diện thêm ảnh vào tài khoản sinh viên 63***

[***4.2.8.***](#_heading=h.44bvf6o) ***Giao diện sửa tài khoản 63***

[***4.2.9.***](#_heading=h.ymfzma) ***Giao diện xóa tài khoản 64***

[**KẾT LUẬN 65**](#_heading=h.1xrdshw)

[**1.**](#_heading=h.4hr1b5p) **Kết quả đạt được 65**

[**2.**](#_heading=h.2wwbldi) **Hướng phát triển 65**

[**TÀI LIỆU THAM KHẢO 66**](#_heading=h.3w19e94)

[**Tiếng Anh 66**](#_heading=h.2b6jogx)

[**Danh mục các website tham khảo 66**](#_heading=h.qbtyoq)

**DANH MỤC CÁC KÝ HIỆU VÀ CHỮ VIẾT TẮT**

| **STT** | **Từ viết tắt** | **Diễn giải** |
| --- | --- | --- |
| 1 | AI | Artificial Intelligence |
| 2 | CNN | Convolution Neural Network |
| 3 | GAP | Global Average Pooling |
| 4 | SVM | Support Vector Machine |
| 5 | LBP | Local Binary Pattern |
| 6 | SIFT | Scale-Invariant Feature Transform |
| 7 | HOG | Histogram of Oriented Gradients |

**DANH MỤC CÁC HÌNH VẼ**

**DANH MỤC CÁC BẢNG**

# LỜI MỞ ĐẦU

Hiện nay, công tác quản lý sinh viên bao gồm điểm danh sinh viên ra vào lớp học, xác thực danh tính sinh viên dự thi, kiểm soát sinh viên ra vào cổng trường, ... đều đượận diện, trong đó có nhận diện bằng khuôn mặt. Vì vậy, ứng dụng điểm danh trong lớp học sử dụng công nghệ điểm danh bằng khuôn mặt sẽ giúp cho việc điểm danh dễ dàng, nhanh chóng và chính xác hơn rất nhiều so với các phương pháp truyền thống.

Mục tiêu của đồ án là tìm hiểu, nghiên cứu các công nghệ nhận diện khuôn mặt và triển khai xây dựng ứng dụng cho bài toán điểm danh sử dụng công nghệ nhận diện khuôn mặt. Ngoài ra, đồ án cc thực hiện một cách thủ công như quẹt thẻ hoặc có người giám sát. Một sinh viên có thể điểm danh cho nhiều bạn vì hệ thống chỉ nhận diện được thông tin từ thẻ mà không nhận diện được người cầm thẻ, hoặc những người giám sát không thể so sánh, đối chiếu chính xác thông tin của tất cả sinh viên. Việc làm này với số lượng lớn sinh viên sẽ mất thời gian và thường khó đảm bảo sự chính xác cao.

Trong thời đại công nghệ thông tin ngày càng phát triển, mọi vấn đề đều được xử lý đơn giản và tối ưu hóa. Công nghệ nhận diện bằng sinh trắc học chính là chìa khóa cho những bài toán nhòn tìm hiểu và phát triển khả năng chống giả mạo cho ứng dụng nhận diện khuôn mặt.

Trong đồ án em sẽ tập trung trình bày một số nội dung chính như sau:

* **Chương 1: Tổng quan về các tác vụ cho bài toán nhận dạng giọng nói:**

Nội dung chương 1 sẽ khái quát các vấn đề và phương pháp nhận dạng giọng nói, khảo sát về các phương pháp học máy đang được sử dụng cho ba nhiệm vụ con, và trình bày về phạm vi của đồ án.

* **Chương 2: Nhận dạng giọng nói bằng mô hình chưng cất và học đa tác vụ:**

Nội dung của chương 2 sẽ giới thiệu kiến thức cơ bản về trí tuệ nhân tạo và các mạng thần kinh học sâu cũng như các bước xây dựng mô hình chưng cất gọn nhẹ đa tác vụ dựa trên cơ chế chú ý và các kỹ thuật như cắt tỉa mô hình và chưng cất tri thức.

* **Chương 3: Thực nghiệm và kết quả:**

Nội dung của chương 3 trình bày quá trình thu thập dữ liệu, mô tả phương pháp thực nghiệm và đánh giá mô hình chưng cất ứng dụng vào nhận dạng giọng nói và trình bày các kết quả của quá trình thực nghiệm.

* **Chương 4: Tổng kết:**

Tổng kết bài toán, tóm tắt những kết quả đã đạt được và còn chưa đạt được. Từ đó đề xuất mục tiêu hướng tới cũng như hướng nghiên cứu, phát triển tiếp theo.

# CHƯƠNG 1: TÌM HIỂU BÀI TOÁN ĐIỂM DANH SỬ DỤNG NHẬN DIỆN KHUÔN MẶT

## Tổng quan về bài toán

### Bài toán điểm danh bằng khuôn mặt

Khái quát bài toán điểm danh bằng khuôn mặt là bài toán xử lý kết quả sinh viên thực hiện điểm danh bằng hình ảnh khuôn mặt. Đầu vào là một ảnh khuôn mặt của sinh viên do sinh viên chụp tại thời điểm giảng viên yêu cầu điểm danh, đầu ra là sinh viên đó điểm danh thành công hay không thành công. Cụ thể là khuôn mặt nhận diện sẽ được phân vào một trong các khuôn mặt nằm trong tập CSDL khi vượt qua một giá trị ngưỡng cho phép. Trường hợp giá trị đo độ giống nhau không vượt qua ngưỡng thì khuôn mặt đó không có trong tập dữ liệu và không được định danh. Kết quả điểm danh thành công sẽ được cập nhật vào cơ sở dữ liệu nhằm mục đích tổng hợp sau này.

### Những khó khăn của việc điểm danh bằng khuôn mặt

* Vấn đề chống giả mạo khuôn mặt

Với sự phát triển của các thiết bị công nghệ cũng như hiệu năng tính toán, xuất hiện ngày càng nhiều các dịch vụ sử dụng công nghệ nhận diện khuôn mặt. Kéo theo đó là những phương pháp cố gắng đánh lừa hệ thống này. Đây là một công việc khó khăn. Ngoài việc nhận diện xem người đó là ai hoặc người đó có tồn tại trong cơ sở dữ liệu của chúng ta không, còn phải xác định thêm người đó là thật hay là ảnh 2D hoặc 3D giả mạo. Với bài toán điểm danh sử dụng khuôn mặt trong lớp học, điều này càng quan trọng hơn.

* Góc chụp

Với các tư thế khác nhau, các phần trên khuôn mặt có thể không rõ hoặc bị khuất hết. Với nhiều góc chụp, ví dụ chụp thẳng nhưng mặt lại quay về hướng khác hoặc là chụp một hướng khác của khuôn mặt đều là những khó khăn rất lớn trong việc nhận diện khuôn mặt.

* Khuôn mặt thay đổi theo thời gian

Các chi tiết như râu ria mép hoặc kiểu tóc có thể thay đổi trong thời gian ngắn có thể làm thay đổi kết quả của bài toán. Ngoài ra còn gặp khó khăn khi khuôn mặt có nhiều trạng thái khác nhau, ví dụ như cùng một khuôn mặt nhưng có trạng thái rất khác nhau khi vui, buồn, ....

* Điều kiện ảnh

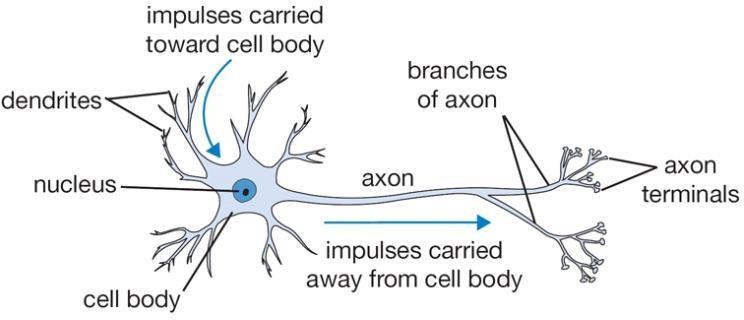
Ảnh được chụp trong các điều kiện khác nhau: ánh sáng, các thiết bị khác nhau, nền ảnh phức tạp.

Trên đây là một số những khó khăn gặp phải khi thực hiện bài toán này.

## Cơ sở lý thuyết và tổng quan về mạng nơ-ron tích chập (Convolution Neural Network) trong Deep Learning

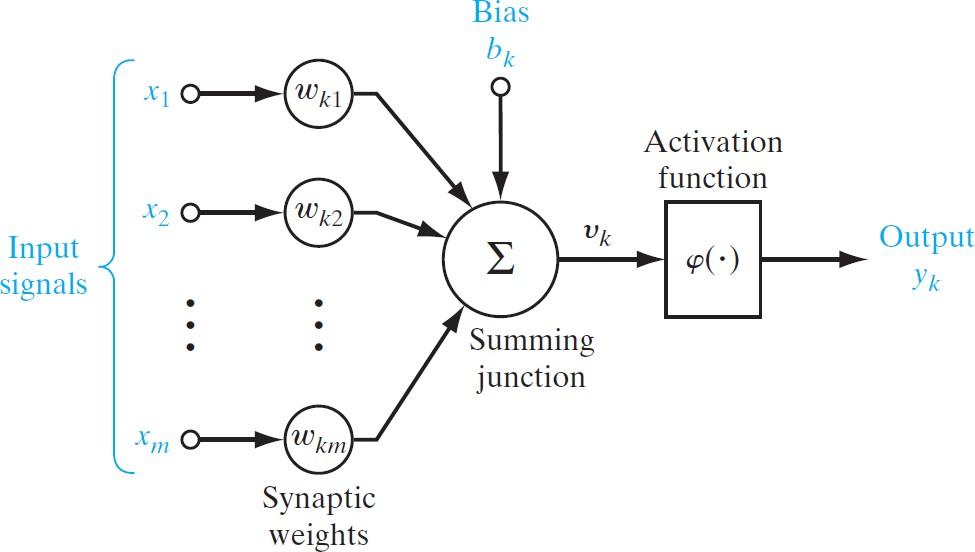
### Mạng nơ-ron nhân tạo (Neural Network)

Mạng nơ-ron nhân tạo *(Neural Network - NN)* là một mô hình lập trình mô phỏng cách thức hoạt động của mạng nơ-ron thần kinh [1]. Nó được cấu thành bởi các nơ-ron đơn lẻ được gọi là các perceptron.

1. *Perceptron*

*Hình* ***Error! No text of specified style in document.****.1Cấu trúc một nơ-ron thần kinh [9]*

Một nơ-ron sinh học có phần thân *(cell body)* chứa nhân *(nucleus)*, các tín hiệu đầu vào qua sợ nhánh *(dendrites)* và các tín hiệu đầu ra qua sợi trục *(axon)* kết nối với các nơ- ron khác. Hiểu đơn giản mỗi nơ-ron nhận dữ liệu đầu vào qua sợi nhánh và truyền dữ liệu đầu ra qua các sợi trục, đến các sợi nhánh của các nơ-ron khác. Mỗi nơ-ron nhận các xung điện từ các nơ-ron khác qua sợi nhánh. Nếu các xung điện này đủ lớn để kích hoạt nơ-ron thì tín hiệu này đi qua sợi trục đến các sợi nhánh của các nơ-ron khác.

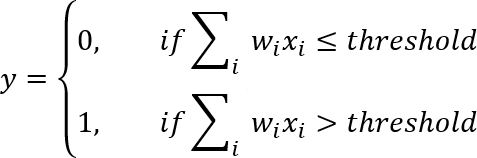
Mô hình của perceptron cũng như vậy:

*Hình* ***Error! No text of specified style in document.****.2 Cấu trúc một nơ-ron nhân tạo*

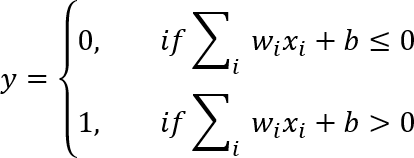
Một perceptron sẽ nhận một hoặc nhiều đầu vào ***x*** (thể hiện bằng các node) và cho ra một kết quả dạng nhị phân. Các đầu vào được điều phối tầm ảnh hưởng bởi các tham số

trọng lượng tương ứng ***w*** của nó, còn kết quả đầu ra được quyết định dựa vào một ngưỡng

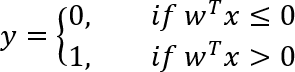
***b*** quyết định nào đó [9]:



Đặt b = -threshold, ta có thể viết lại thành:



Gắn 𝒙𝟎 ***= 1*** và 𝒘𝟎 = 𝒃 , ta có thể viết gọn thành:



Với đầu ra là dạng nhị phân, ta rất khó có thể điều chỉnh một lượng nhỏ đầu vào để đầu ra thay đổi chút ít, nên để linh động ta mở rộng đầu ra thành khoảng [*0*, *1*]. Lúc này đầu ra được quyết định bởi một hàm 𝜎(𝑤𝑇𝑥) có giá trị trong khoảng [*0*, *1*] gọi là hàm kích hoạt *(activation function)* [9]. Ví dụ hàm kích hoạt sigmoid:

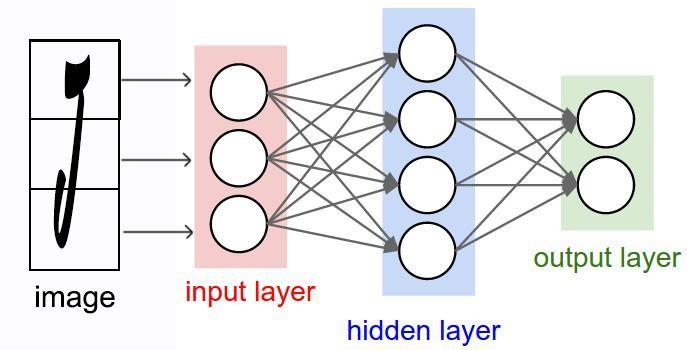


*Hình 1.3. Hàm sigmoid [9]*

Ngoài hàm sigmoid ra còn có các hàm kích hoạt khác như tanh, ReLU, softmax, ...

1. *Multi-layer perceptron (mạng nơ-ron nhân tạo)*

Khi kết hợp các tầng perceptron ta có một mạng nơ-ron tổng quát gồm 3 layer:

* + Input layer: nhiều node ở input tạo thành một lớp đầu vào của mạng.
  + Hidden layer: các layer trung gian thể hiện cho việc suy luận logic của mạng.
  + Output layer: nhiều node ở đầu ra tạo thành một lớp đầu ra của mạng.

*Hình 1.4. Mô hình neural network [9]*

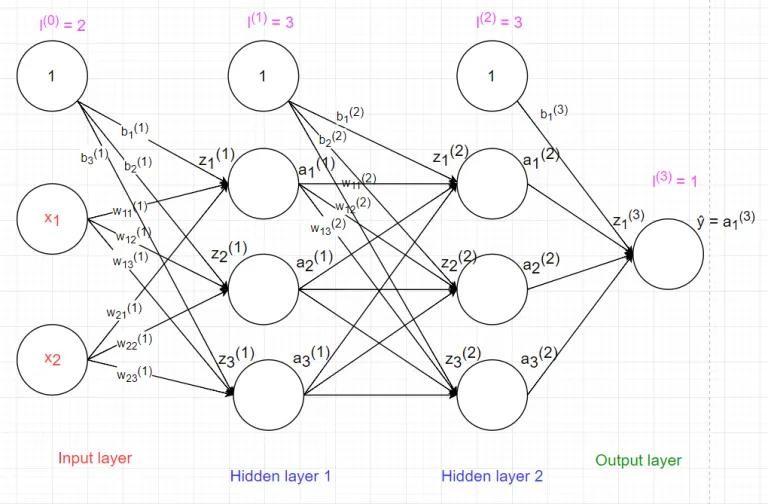
Mỗi mô hình luôn có 1 input layer, 1 output layer, có thể có hoặc không các hidden layer. Tổng số layer trong mô hình được quy ước là số layer – 1 (Không tính input layer) [9].

Mỗi node trong hidden layer và output layer:

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

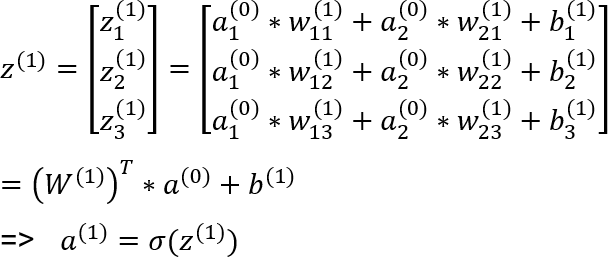
Quá trình suy luận từ lớp đầu vào tới lớp đầu ra của mạng nơ-ron là quá trình lan truyền tiến *(feedforward)*, tức là đầu vào của các nơ-ron tại một tầng là lấy kết quả từ nơ- ron tầng trước đó mà không có quá trình suy luận ngược lại [1].

*Feedforward*

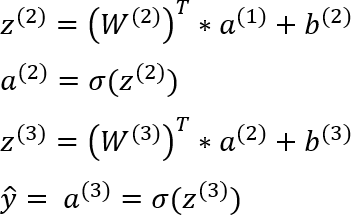


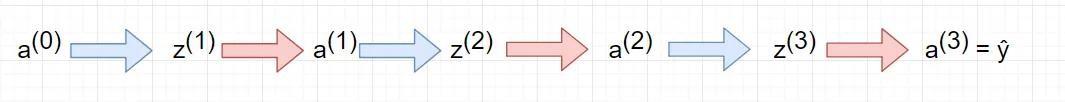
*Hình 1.5. Mô hình neural network với 2-3-3-1 (input layer có 2 node, hidden layer 1 có 3 node, hidden layer 2 có 3 node, output layer có 1 node) [1]*

Gọi input layer là 𝑎(*0*) = 𝑥, (*ở ℎì*𝑛*ℎ* 𝑣*ẽ* 𝑑*ướ*𝑖 *đâ*𝑦 𝑡*ℎì* 𝑎(*0*) = {𝑥*1*, 𝑥*2*}), thực hiện tính 𝑧(*1*)là đầu ra của input layer, đầu vào của hidden layer 1:



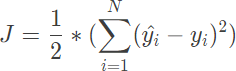
Tương tự ta có kết quả tính tại mỗi lớp:



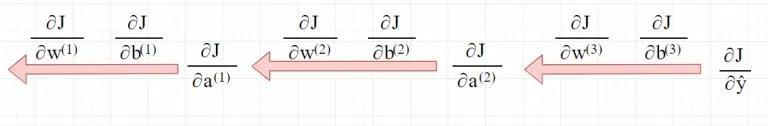


*Quá trình feedforward*

Từ input *X* ta có thể tính được giá trị dự đoán 𝑌̂, sau đó sẽ tính *loss function* (hàm thể hiện sự chênh lệch giữa giá trị dự đoán 𝑌̂ và giá trị thực *Y)* để cập nhật hệ số và tối ưu loss function [1].



*Ví dụ hàm loss function*

Vì vậy ta cần đi tìm hệ số *W* và *b*. Để làm được điều đó ta sử dụng thuật toán *gradient descent* (thuật toán tìm giá trị nhỏ nhất của hàm số f(x) dựa trên đạo hàm). Ở đây ta cần tính đạo hàm của các hệ số *W* và *b* đối với hàm *loss function.* Sau khi thực hiện ta được:

Quá trình này được gọi là *backpropagation (lan truyền ngược).*

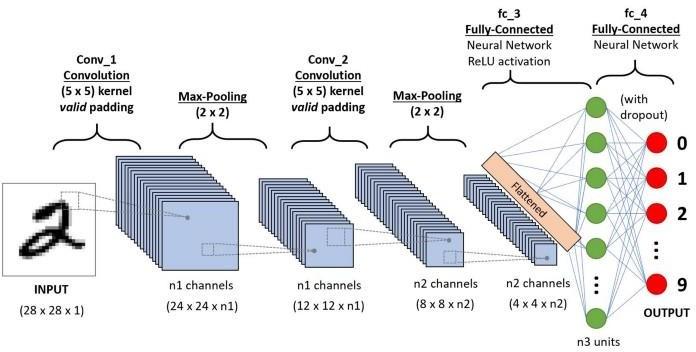
Như vậy, mạng nơ-ron nhân tạo là một mô hình được tạo nên từ một lượng lớn các phần tử (nơ-ron) kết nối với nhau thông qua các liên kết (trọng số liên kết) làm việc như một thể thống nhất để giải quyết một vấn đề cụ thể (phân loại dữ liệu, nhận dạng mẫu, …) thông qua một quá trình học từ tập các mẫu huấn luyện. Về bản chất học ở đây chính là quá trình hiệu chỉnh trọng số liên kết giữa các nơ-ron.

### Mạng nơ-ron tích chập (Convolution Neural Network)

Phân loại ảnh là một bài toán quan trọng bậc nhất trong lĩnh vực “Thị giác máy tính” *(Computer Vision)*. Tuy nhiên các kiến trúc mạng nơ-ron truyền thống có xu hướng kém hiệu quả khi kích thước ảnh đầu vào của mạng trở nên lớn hơn. Với ảnh có kích thước là 32\*32\*3 (cao 32 pixel, rộng 32 pixel, có 3 kênh màu) thì với mạng nơ-ron truyền thống lớp nơ-ron ẩn đầu tiên sẽ có 32\*32\*3 = 3072 trọng số. Số lượng trọng số này thì vẫn có thể “kiểm soát” được nhưng khi ảnh đầu vào có kích thước 200\*200\*3 thì số lượng trọng số lên tới 120.000 trọng số tại lớp đầu tiên, cộng thêm các trọng số qua các lớp hidden layers thì số lượng trọng số của mạng sẽ tăng cực kì nhanh. Điều này làm cho mạng nơ-ron truyền thống (dạng *fully connected*) dễ bị *overfitting* (quá khít với dữ liệu dùng để huấn luyện nhưng hoạt động kém với tập dữ liệu đánh giá, kiểm tra) và tiêu tốn nhiều tài nguyên tính toán.

*Convolution Neural Network* (mạng nơ-ron tích chập) hiện nay đang là giải pháp tốt nhất để giải quyết bài toán đó. Bởi trong ảnh các pixel ở cạnh nhau thường có *liên kết với nhau hơn* là những pixel ở xa, vì vậy áp dụng phép tính convolution vào layer trong neural network ta có thể giải quyết được vấn đề lượng lớn parameter mà vẫn lấy ra được các đặc trưng của ảnh.

Mạng nơ-ron tích chập bao gồm tập hợp các lớp chính cơ bản là: lớp tích chập (*Convolution layer*), lớp lấy mẫu (*Pooling layer*) và lớp kết nối đầy đủ (*Fully connected layer*).



*Hình 1.6. Kiến trúc cơ bản của một mạng nơ-ron tích chập [10]*

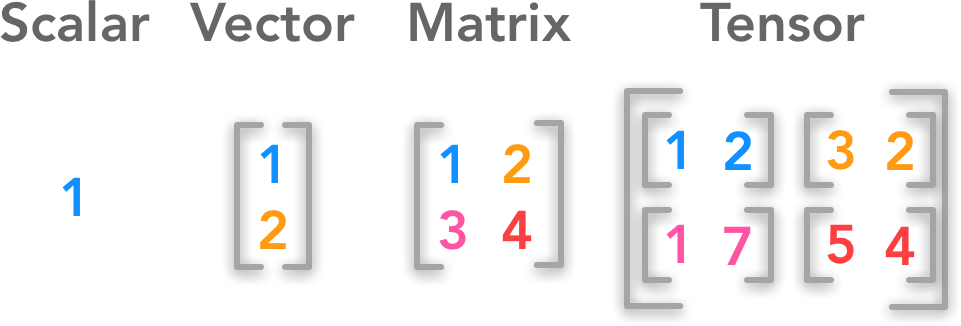
## Kiến trúc cơ bản của mạng nơ-ron tích chập

* + *Vector, ma trận và tensor*

Ma trận và vector là những đại lượng phổ biến trong toán học. Một vector

𝑥 𝜖 𝑅𝐷 là một vector chứa D phần tử được xếp theo chiều dọc. Ma trận 𝑋 thuộc 𝑅𝑊∗𝐻 là một ma trận với 𝐻 hàng và 𝑊 cột. Vector 𝑥 còn được gọi là một ma trận với 1 cột và 𝐷 hàng.

Để biểu diễn dữ liệu phức tạp hơn, chúng ta cần một đại lượng với nhiều chiều được gọi là tensor. Ví dụ 𝑥 thuộc 𝑅𝑊𝑥𝐻𝑥𝐷 là một tensor 3 chiều. Tensor x này chứa 𝑊 ∗ 𝐻 ∗ 𝐷 phần tử, mỗi phần tử được định vị bởi tọa độ (𝑖, 𝑗, 𝑑). Một tensor với 3 chiều và chiều thứ 3 có độ dài 𝐷 tương đương với 𝐷 ma trận chồng lên nhau. Mỗi chiều ứng với một ma trận có kích thước 𝐻 ∗ 𝑊. Chiều thứ nhất chứa tất cả các số nằm trên tensor mà có tọa độ (𝑖, 𝑗, *0*).

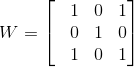


*Hình 1.7. Ví dụ về Vector, Matrix, Tensor*

Một ảnh RGB với 𝐻 hàng và 𝑊 cột là một tensor với chiều 𝐻 ∗ 𝑊 ∗ *3*. Mỗi chiều là một ma trận 𝐻 ∗ 𝑊. Trong CNN, ảnh đầu vào được biểu diễn dưới dạng tensor. Các tầng trung gian giữa các lớp tích chập cũng vậy. 2 lớp tích chập nhận đầu vào là 1 tensor và sau khi đưa qua mạng ta cũng sẽ nhận lại được một tensor.

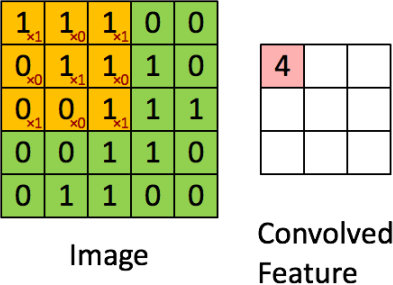
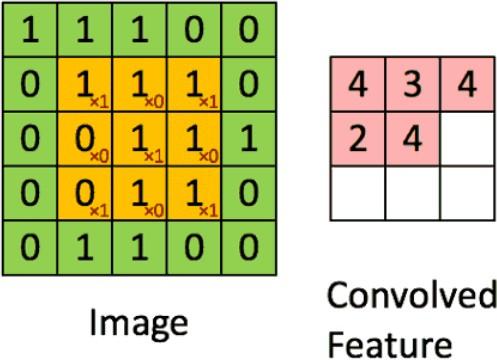
* + *Convolution layer*

Đây là lớp đầu tiên và cũng là lớp quan trọng nhất của mạng. Lớp này có chức năng chính là phát hiện các đặc trưng có tính không gian một cách hiệu quả. Các đặc trưng này có thể là các đặc trưng đơn giản như đặc trưng về góc, cạnh, màu sắc hoặc là các đặc trưng trừu tượng, phức tạp hơn. Các đặc trưng này sẽ giúp cho mô hình phân loại được dữ liệu đầu vào tốt hơn so với các phương pháp trích chọn đặc trưng khác (LBP, HoG, …).

Để thực hiện tính convolution, ta dùng một *bộ lọc* (*kernel/filter -* ma trận vuông kích thước k\*k, trong đó k là số lẻ, giá trị của k thường là 1, 3, 5, 7, 9, ...).

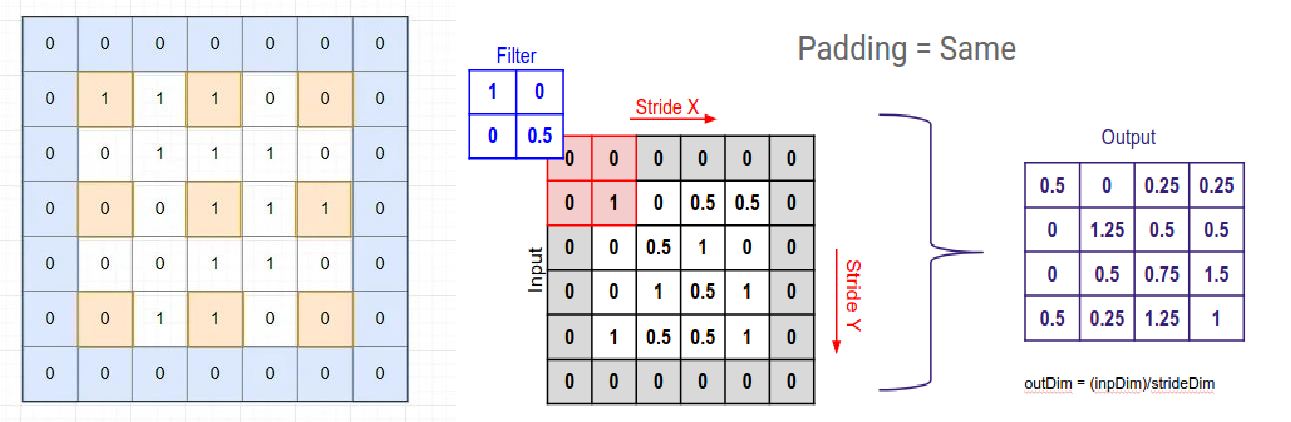
*Hình 1.8. Ví dụ bộ lọc kích thước 3x3*

Với mỗi phần tử trong ma trận điểm ảnh lấy ra một ma trận có kích thước bằng kích thước của bộ lọc W có trung tâm bộ lọc trùng với phần tử đó (đây là vì sao kích thước của kernel thường lẻ) gọi là ma trận A. Sau đó tính tổng các phần tử của phép tính nhân chập của ma trận A và ma trận W, rồi viết vào ma trận kết quả Y.

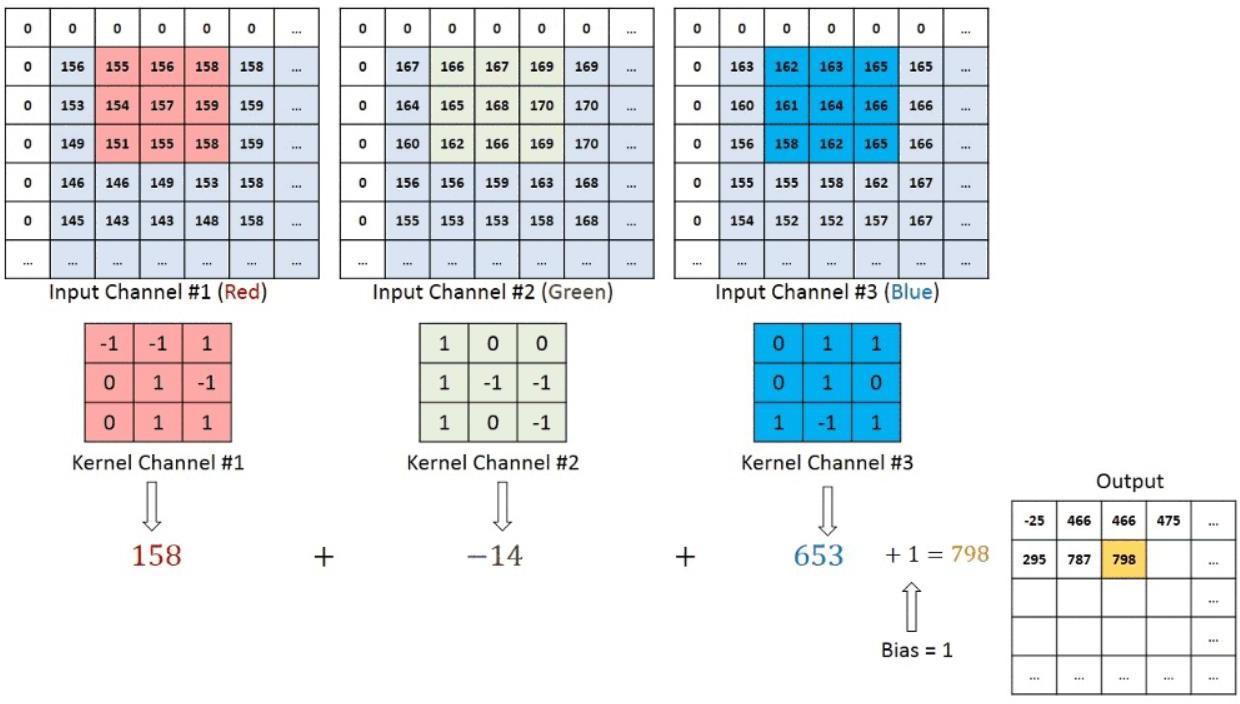
*Hình 1.9. Tích chập một ảnh 5x5x1 với một bộ lọc 3x3x1 để có một hình ảnh chập 3x3x1 [10]*

Bộ lọc di chuyển từ phải sang trái, từ trên xuống dưới lần lượt cho đến khi toàn bộ hình ảnh được duyệt qua [10]. Khoảng cách mà bộ lọc di chuyển từ điểm ảnh này tới điểm ảnh sau đó gọi là *Stride* (bước nhảy). Khi *Stride* = 1 thì chúng ta sẽ di chuyển các bộ lọc 1-pixel mỗi lần. Khi *Stride* = 2 thì chúng ta di chuyển các bộ lọc 2-pixel cùng một lúc và cứ thế di chuyển bộ lọc với *Stride* tương ứng.

Lựa chọn *Stride* càng lớn thì chiều của kết quả tích chập càng giảm. *Padding* là việc thêm các giá trị ở xung quanh biên ảnh ban đầu. *Padding = 1* tức là thêm 1 ô bọc xung quanh các cạnh của input, điều này đảm bảo phép tích chập được thực hiện đủ trên toàn ảnh.

*Hình 1.10. Ví dụ Stride = 2, Padding = 1 (bên trái) và Stride = 1, Padding = 1 (Padding = Same) (bên phải)*

Trong trường hợp đầu vào là ảnh màu có 3 kênh màu RGB ta biểu diễn ảnh dưới dạng *tensor* 3 chiều *(Số thực là dữ liệu 0D, vector 1D, ma trận 2D còn dữ liệu từ 3D trở đi được gọi là tensor)* nên bộ lọc cũng là một *tensor* 3 chiều kích thước k\*k\*3.



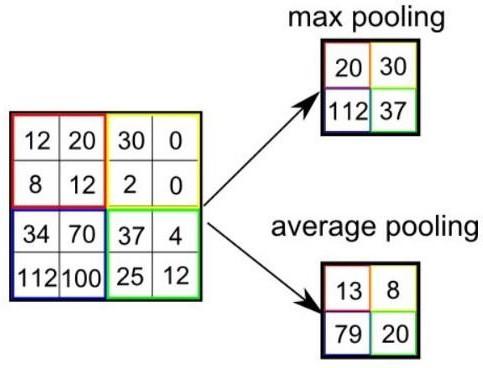
*Hình 1.11. Thực hiện phép tính convolution trên ảnh màu [10]*

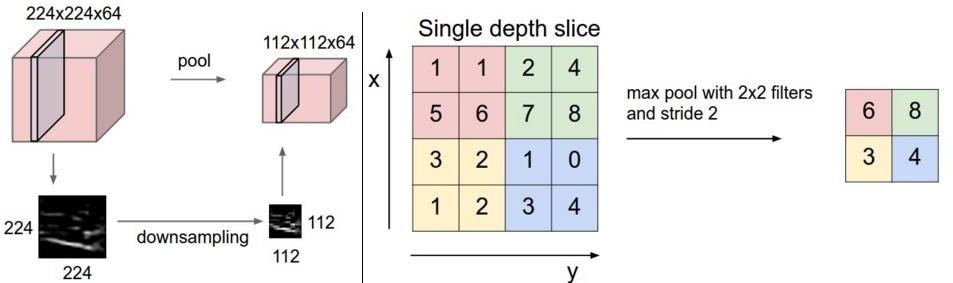
* + *Pooling layer*

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

Có 2 loại *Pooling layer* phổ biến là: *Max pooling* và *Average pooling*. Phép gộp cực đại *(Max pooling)* trả về giá trị lớn nhất từ phần hình ảnh được bao phủ bởi bộ lọc. Trong khi đó, phép gộp trung bình *(Average pooling)* trả về giá trị trung bình của tất cả các giá trị từ phần hình ảnh được bao phủ bởi bộ lọc.

Phép gộp cực đại cũng hoạt động như một công cụ khử nhiễu. Nó loại bỏ các nguồn nhiễu và thực hiện khử nhiễu song song với giảm kích thước. Mặt khác, phép gộp trung bình chỉ đơn giản thực hiện giảm kích thước như một cơ chế khử nhiễu. Do đó, chúng ta có thể nói rằng phép gộp cực đại hoạt động tốt hơn rất nhiều so với phép gộp trung bình.



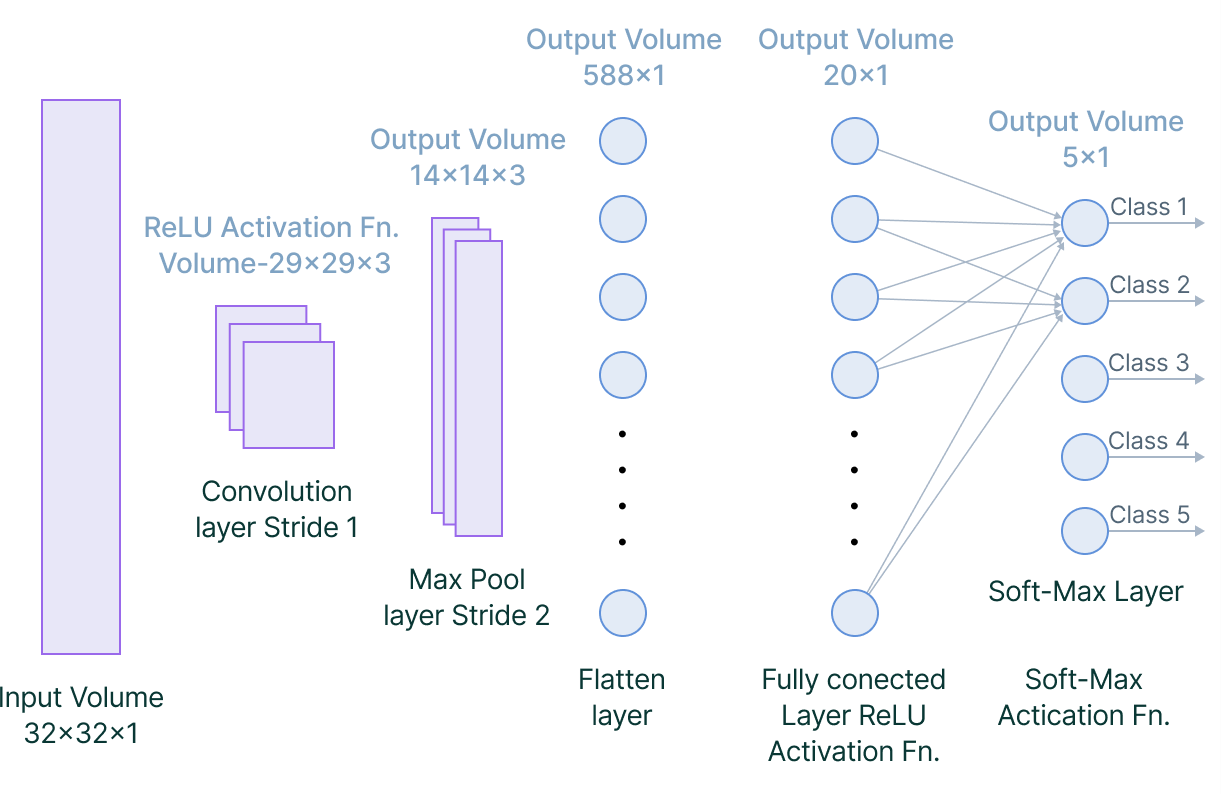


*Hình 1.12. Ví dụ về Pooling layer*

* + *Fully connected layer*

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

Sau đó sử dụng mạng nơ-ron kết nối đầy đủ *(Fully connected layer)* để học các tổ hợp phi tuyến của các đặc trưng trích xuất từ kết quả ma trận tích chập đầu ra, từ đó ra được output của model.



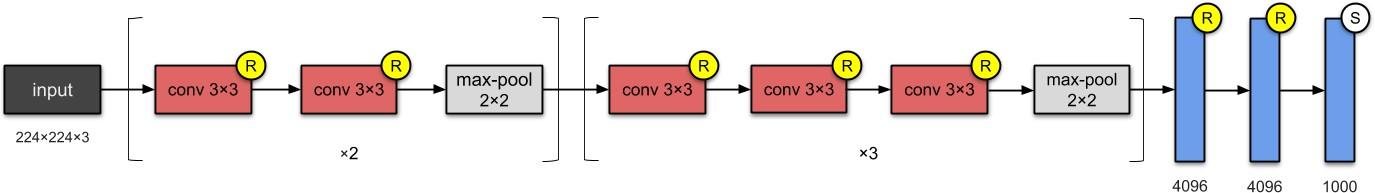
*Hình 1.13. Ví dụ về Fully Connected Layer*

Như vậy, chúng ta đã chuyển đổi hình ảnh đầu vào thành một dạng thích hợp cho mạng nơ-ron đa lớp. Chúng ta sẽ làm phẳng hình ảnh đầu vào này thành một vec-tơ cột. Vec-tơ đầu ra đã được làm phẳng sẽ được đưa vào một mạng nơ-ron suy luận tiến *(feedforward)* và phương pháp truyền ngược *(backpropagation)* được áp dụng cho quá trình huấn luyện. Qua một loạt lần lặp, mô hình có thể phân biệt giữa các đặc trưng cốt lõi và các đặc trưng không quan trọng trong hình ảnh và phân loại chúng bằng hàm kích hoạt *(activation function)*.

## Một số kiến trúc mạng nơ-ron tích chập

* + *Kiến trúc mạng VGG 16*

Mạng *VGG 16* được phát triển bởi K. Simonyan and A. Zisserman, University of Oxford với quan điểm về một mạng nơ-ron sâu hơn sẽ giúp ích cho cải thiện độ chính xác của mô hình tốt hơn.



*Hình 1.14. Kiến trúc mạng VGG 16 [11]*

Kiến trúc *VGG 16* bao gồm 16 *Convolution Layers,* 13 layers tích chập và 3 layers fully connected. Mạng này chỉ sử dụng convolutions 3x3 nhưng với rất nhiều *filters,* và được huấn luyện với 4 GPUs trong 2-3 tuần. Việc sử dụng các bộ lọc có

kích thước nhỏ 3x3 cũng giúp giảm số lượng tham số cho mô hình và mang lại hiệu quả tính toán hơn. Ngoài ra, lần đầu tiên trong *VGG 16* xuất hiện khái niệm về khối tích chập *(block).* Đây là những kiến trúc gồm một tập hợp các layers CNN được lặp lại giống nhau, và nó đã khởi nguồn cho các kiến trúc mạng CNN khác.

Model sau khi train bởi mạng VGG16 đạt độ chính xác 92.7% top-5 test trong dữ liệu ImageNet gồm 14 triệu hình ảnh thuộc 1000 lớp khác nhau.

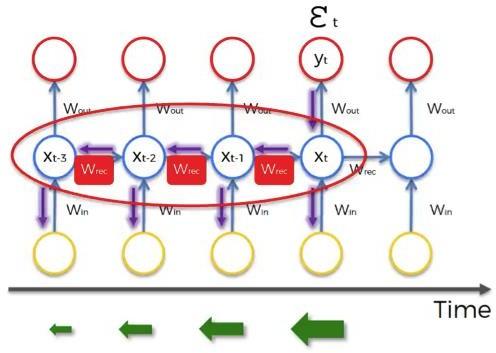
Tuy nhiên *VGG 16* lại chứa tới những 138 triệu tham số, điều này ảnh hưởng phần nào tới việc kiểm soát và sử dụng nó hiệu quả [11].

* + *Kiến trúc mạng ResNet*

Kiến trúc mạng ResNet được phát triển bởi Kainming He (một nhà nghiên cứu ở Microsoft). Mạng này được cấu thành từ rất nhiều các lớp tích chập xếp chồng lên nhau khiến cho mạng sâu hơn. Một vấn đề xảy ra khi xây dựng mạng CNN với nhiều lớp chập sẽ xảy ra hiện tượng mất mát đạo hàm *(Vanishing Gradient)* dẫn tới quá trình học tập không tốt [11].

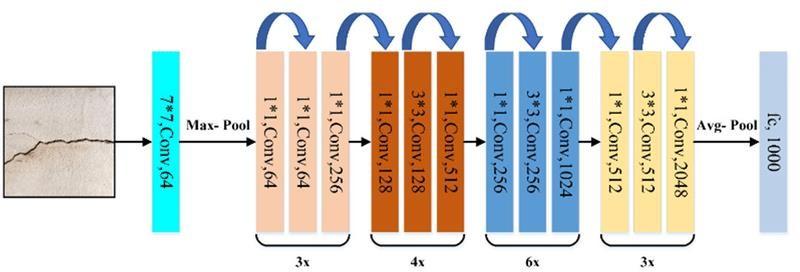
*Vanishing Gradient*

Trước hết *backpropagation* (lan truyền ngược) là một kỹ thuật sử dụng trong quá trình học, tính toán đạo hàm của *loss function* từ output layer đến input layer sau đó dùng *gradient descent* để hiệu chỉnh các trọng số của mạng. Toàn bộ quá trình sẽ được lặp lại cho tới khi các trọng số của network được hội tụ. Nếu số lượng vòng lặp quá nhỏ thì mạng có thể sẽ không cho ra kết quả tốt và ngược lại thời gian học sẽ lâu nếu số lượng vòng lặp quá lớn.



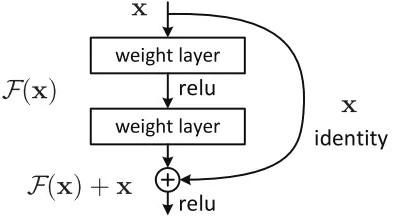
*Hình 1.15. Hiện tượng Vanishing Gradient*

Tuy nhiên, trong thực tế đạo hàm thường sẽ có giá trị nhỏ dần khi đi xuống các layer thấp hơn. Dẫn đến kết quả là các cập nhật thực hiện bởi *gradient descent* không làm thay đổi nhiều trọng số của các layer và làm chúng không thể hội tụ để thu được kết quả tốt. Hiện tượng như vậy được gọi là *Vanishing Gradient.*

Kiến trúc ResNet sinh ra để giải quyết vấn đề này.

*Hình 1.16. Kiến trúc mạng ResNet*

Ý tưởng chính của ResNet là kiến trúc gồm các khối dư (residual) trong đó xuất hiện các nhánh cộng (còn gọi là nhánh nối tắt) giữa giá trị đầu vào với giá trị đầu ra của các khối dư đó.



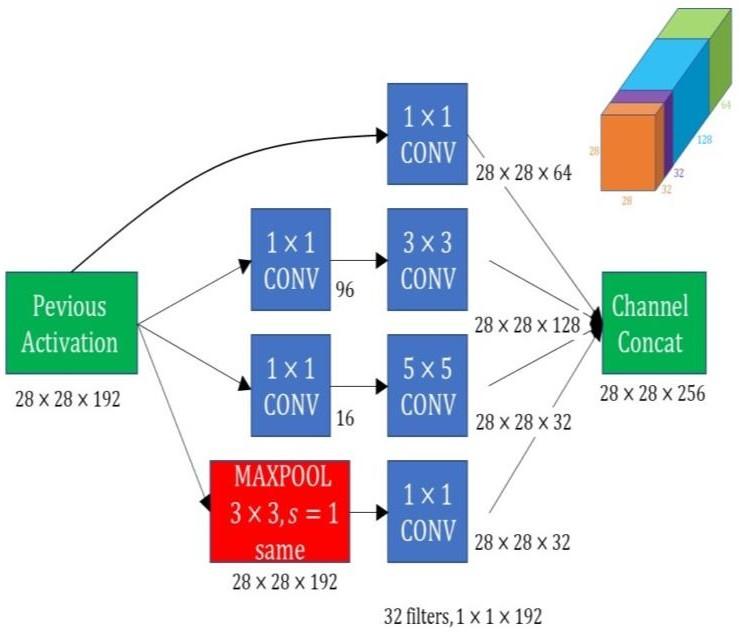
*Hình 1.17. Khối Residual trong ResNet [11]*

Hình trên mô tả khối dư trong ResNet gồm giá trị đầu vào (x) qua một số lớp tích chập thu được F(x), sau đó cộng thêm x vào F(x) để được đầu ra của khối dư là F(x) + x. Nhờ nhánh cộng này mà đạo hàm có thể lan truyền ngược lại lên các lớp trên cùng của mạng mà tránh được hiện tượng mất mát đạo hàm.

* + *Kiến trúc mạng GoogleNet - Inception*

Kiến trúc mạng *GoogleNet* được phát triển bởi Google. Kiến trúc này đã giải quyết một câu hỏi lớn trong mạng CNN đó là sử dụng kích thước filter bao nhiêu thì hợp lý. Các kiến trúc nơ-ron trước đó đều sử dụng các bộ lọc với đa dạng kích thước 11x11, 5x5, 3x3, nhỏ nhất là 1x1. Việc cùng kết hợp đồng thời các bộ lọc này vào cùng một block tạo thành kiến trúc khối *Inception* trong mạng *GoogleNet* đã mang lại hiệu quả [11].

Cấu trúc khối *Inception*

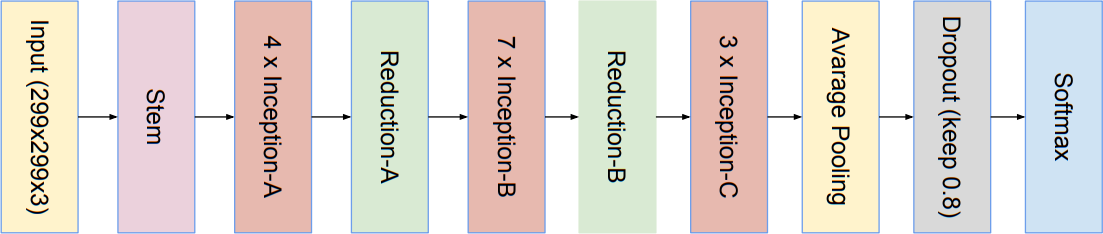


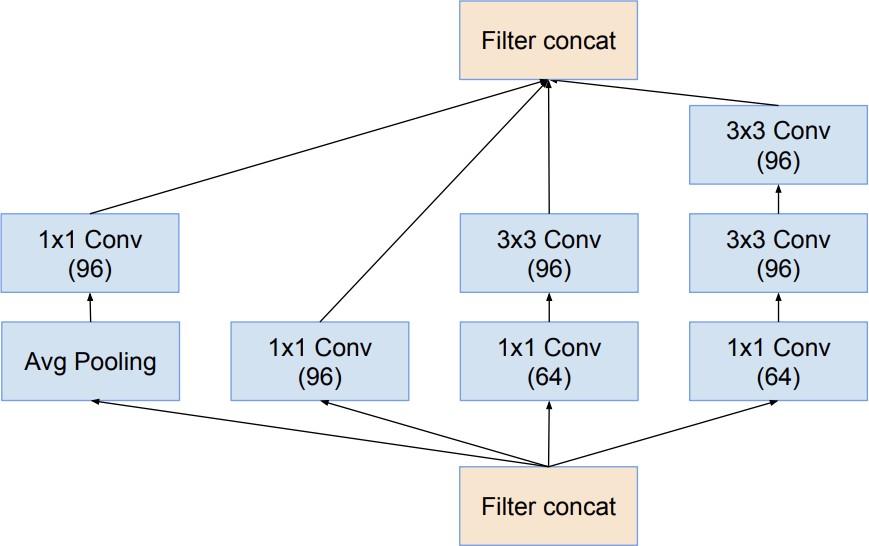
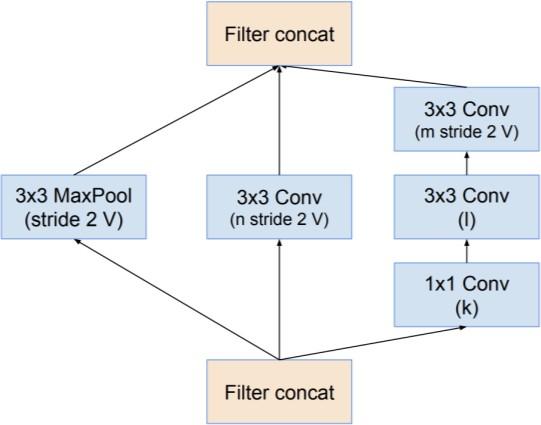
*Hình 1.18. Cấu trúc khối Inception [11]*

Như hình trên, khối *Inception* bao gồm 4 nhánh song song với nhau. Các bộ lọc kích thước lần lượt là 1x1, 3x3, 5x5 được áp dụng giúp trích lọc được đa dạng đặc trưng trên những vùng nhận thức có kích thước khác nhau. Sử dụng phép tích chập 1x1 trên từng điểm ảnh như một kết nối *fully connected* nhằm giảm độ sâu kênh và số lượng tham số của mô hình *(ở hình vẽ trên sau khi nhân chập với 64 bộ lọc kích thước 1x1 thì kích thước ma trận điểm ảnh đã giảm từ 28x28x192 xuống 28x28x64)*. Ngoài ra, khối *Inception* còn sử dụng *Max pooling* sau đó áp dụng bộ lọc kích thước 1x1 cũng làm giảm chiều dữ liệu [11].

Khối *Inception* được lặp lại nhiều lần trong kiến trúc *GoogleNet.* Toàn bộ mạng *Inception V1* có 22 layers, lớn hơn gấp đôi so với *VGG 16*, nhưng nhờ áp dụng tích chập 1x1 giúp tiết kiệm số lượng tham số xuống chỉ còn 5 triệu, ít hơn 27 lần so với *VGG 16*.

Với mục đích xây dựng mạng sâu để học được nhiều hơn các đặc trưng từ ảnh đầu vào nhưng vẫn ổn định, ít tốn kém hơn và hiệu quả, các phiên bản *Inception* sau này đã được trang bị thêm kỹ thuật mới: *Factorization* (là kỹ thuật để giảm thêm kích thước, ví dụ thay vì sử dụng bộ lọc 3x3 có số lượng các trọng số là 3x3 = 9 thì sử dụng bộ lọc 3x1 và bộ lọc 1x3 sẽ có số lượng các trọng số là 3x1 + 1x3 = 6) và *Reduction* block (các khối chuyên biệt để giảm kích thước một cách rõ ràng).



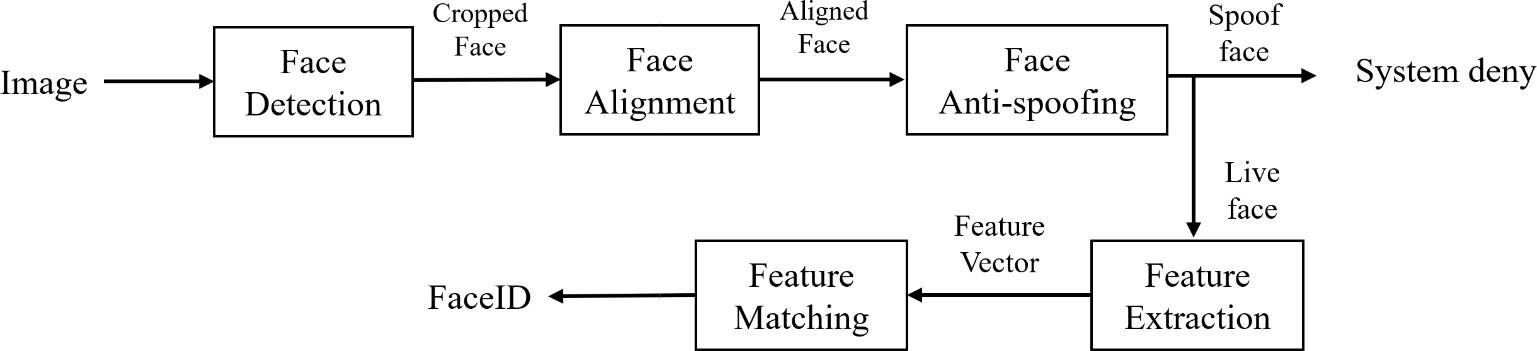
 

*Hình 1.19. Kiến trúc Inception V4 và Inception-A (trái dưới), Reduction-A (phải dưới) [12]*

Chương đầu đã trình bày khái niệm bài toán điểm danh bằng khuôn mặt, lý thuyết mạng nơ-ron nhân tạo và các kiến trúc CNN cơ bản được ứng dụng cho bài toán liên quan đến thị giác máy tính.

# CHƯƠNG 2: KHẢO SÁT CÁC CÔNG NGHỆ XỬ LÝ ẢNH, HỌC MÁY TRONG HỆ THỐNG NHẬN DIỆN GƯƠNG MẶT

**Tổng quan hệ thống nhận diện khuôn mặt**



*Hình 2.1. Tổng quan hệ thống nhận diện khuôn mặt*

Một hệ thống nhận diện khuôn mặt gồm nhiều phần để từ đó xây dựng nên một hệ thống nhận diện khuôn mặt hoàn chỉnh. Trong bài viết này sẽ khảo sát các thành phần của hệ thống nhận diện khuôn mặt gồm 5 phần chính:

* Bộ phát hiện khuôn mặt *(Face Detection)*: Xác định vị trí khuôn mặt trong toàn bộ bức ảnh.
* Bộ căn chỉnh khuôn mặt *(Face Alignment)*: Xác định các điểm mốc trên khuôn mặt *(facial landmark detector)* sau đó chuẩn hóa, căn chỉnh lại khuôn mặt được phát hiện trong bộ phát hiện khuôn mặt trước đó.
* Bộ phát hiện chống giả mạo *(Face Anti-spoofing)*: Kiểm tra khuôn mặt của người dùng hệ thống là khuôn mặt thật hay khuôn mặt giả.
* Bộ trích xuất đặc trưng khuôn mặt *(Feature Extraction)*: Mã hóa, trích xuất các thông tin có trên gương mặt của ảnh đầu vào thành một vector đặc trưng đại diện cho khuôn mặt đó.
* Bộ nhận diện khuôn mặt *(Face Matching)*: Tìm kiếm vector đặc trưng của khuôn mặt đại diện cho ai trong hệ thống.

Trải qua nhiều năm phát triển, mỗi mô-đun trong hệ thống nhận diện khuôn mặt đều được cải tiến và phát triển không ngừng, giúp cho hệ thống nhận diện khuôn mặt tổng quan ngày một chính xác và từ đó là cơ sở để hệ thống nhận diện được ứng dụng vào thực tế ngày càng phổ biến. Dưới đây sẽ là những thuật toán được sử dụng để xây dựng từng mô- đun của hệ thống.

## Bộ phát hiện khuôn mặt (Face Detector)

Phát hiện khuôn mặt là một bài toán nhỏ trong bài toán tổng thể phát hiện đối tượng *(Object Detection)*. Phát hiện khuôn mặt là quá trình xác định vùng hình ảnh chứa khuôn mặt xuất hiện trong ảnh. Đầu vào của hầu hết mọi thuật toán phát hiện khuôn mặt là một ảnh. Đầu ra của thuật toán là vùng ảnh chứa khuôn mặt có dạng hình chữ nhật có thể được biểu diễn bởi 4 điểm (hoặc 2 điểm và chiều dài, rộng) kèm theo đó là xác suất khuôn mặt có trong vùng ảnh đó. Hiệu suất của phát hiện khuôn mặt ảnh hưởng trực tiếp và rất lớn đến các các nhiệm vụ đằng sau của hệ thống nhận dạng khuôn mặt.

Trong nhiều thập kỷ qua, các nhà nghiên cứu đã phát triển nhiều phương pháp để xác định khuôn mặt từ đơn giản đến phức tạp. Và quan trọng nhất là việc ảnh hưởng qua lại giữa độ chính xác và tốc độ detect (phát hiện và xác định vị trí).

Nhìn chung, phát hiện khuôn mặt vẫn còn là một thách thức lớn bởi sự thay đổi của tư thế, sự biến dạng hình ảnh, sự che khuất khuôn mặt và nhiều các yếu tố khác.



*Hình 2.2. Các khuôn mặt khó trong thực tế [2]*

Có thể chia các phương pháp detect thành 2 loại chính là detect dựa trên các *hand- craft feature* (đặc trưng cứng) và detect dựa trên các feature được trích chọn từ mạng CNN.

* **Detect dựa trên các hand-craft feature (đặc trưng cứng)**

Các phương pháp detect truyền thống là detect dựa trên cửa sổ trượt và các *hand- craft feature* (đặc trưng cứng) như: HOG, SIFT, LBP, … đi qua một bộ phân loại như SVM để xác định xem vùng đó có chứa khuôn mặt không, điển hình là thuật toán *HOG-SVM* detector được triển khai trên thư viện *Dlib*.



*Hình 2.3. Đặc trưng HOG của ảnh chứa khuôn mặt*

Tuy nhiên, các phương pháp này đều trở nên kém hiệu quả khi áp dụng trong môi trường thực tế.

* **Detect dựa trên các feature được trích chọn từ mạng CNN**

Với sự phát triển vượt bậc của trí tuệ nhân tạo nói chung và học máy, học sâu nói riêng đã giúp việc detect khuôn mặt trở nên chính xác hơn rất nhiều. Thay vì sử dụng các đặc trưng *handcraft* và lựa chọn các đặc trưng tốt nhất như trước đây, các mô hình hiện nay có thể trích xuất đặc trưng tự động thông qua mạng CNN.

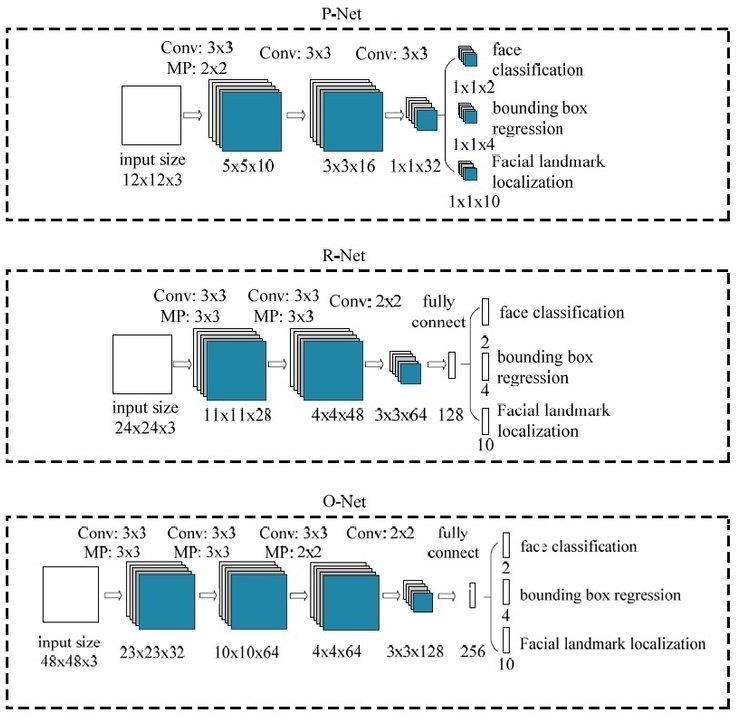


*Hình 2.4. Mô hình phát hiện khuôn mặt giai đoạn 2015-2020 [2]*

**Một số mô hình phát hiện khuôn mặt dựa trên mạng CNN.**

* *Mô hình Multi-task Cascaded Convolution Networks (MTCNN):*

MTCNN có kiến trúc gồm 3 phần: P-Net, R-Net và O-Net.



*Hình 2.5. Kiến trúc P-Net, R-Net, O-Net trong MTCNN [3]*

Phần P-Net: nhận dạng toàn bộ số khuôn mặt trong bức ảnh bằng cách tạo nhiều bản copy từ ảnh gốc với kích thước khác nhau tạo thành một Image Pyramid để mạng dễ dàng nhận diện các khuôn mặt. Sau đó thiết lập ngưỡng và loại bỏ các hộp giới hạn có tỉ lệ trùng nhau.

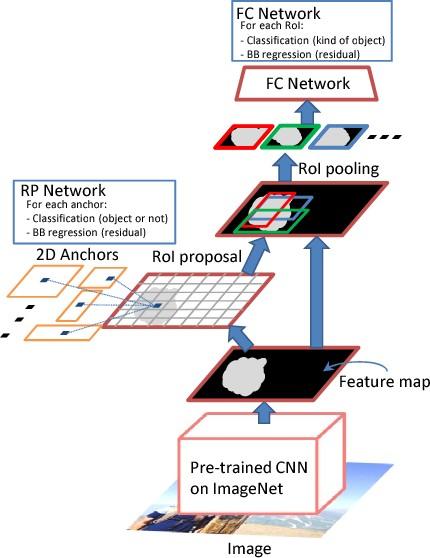
Phần R-Net: cũng thực hiện như P-Net và tinh chỉnh những hộp giới hạn. Phần O-Net: xác định vị trí khuôn mặt và những điểm quan trọng trên khuôn

mặt.

* *Mô hình phát hiện khuôn mặt dựa trên Faster R-CNN (Faster Region Proposal Network):*

Mô hình Faster R-CNN là một trong những mô hình điển hình của thuật toán Two stage detection (sử dụng 2 giai đoạn xử lý riêng biệt là phát hiện các region proposal và phân loại hình ảnh) trong họ R-CNN (regions with CNN features) gồm R-CNN, Fast R-CNN, Faster R-CNN.

Trong giai đoạn đầu của Faster R-CNN, bức ảnh sẽ được cho qua pre-trained model để lấy feature map, sau đó feature map được đưa qua một mạng CNN gọi là RPN *(Region Proposal Network)* với nhiệm vụ xác định các vùng trên ảnh có khả năng chứa đối tượng. Kết hợp với các anchor *(anchor là một bounding box cơ sở để xác định bounding box bao quanh vật thể dựa trên các phép dịch tâm và scale kích thước chiều dài, rộng)* sẽ cho kết quả bounding box chắc chắn chứa đối tượng.

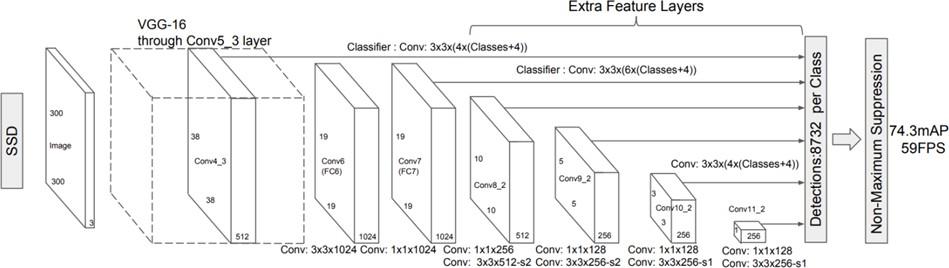
Sau khi đã thu được các vùng đặc trưng từ RPN, model Faster-RCNN sẽ thực hiện tiếp việc phân loại đối tượng và xác định vị trí nhờ vào việc chia làm 2 nhánh con tại phần cuối của mô hình (phân lớp cho đối tượng và tìm bounding box tốt nhất dựa vào thuật toán regression).

*Hình 2.6. Kiến trúc Faster R-CNN*

* *Mô hình phát hiện khuôn mặt dựa trên SSD (Single Shot Detection):*

Mặc dù các kiến trúc họ R-CNN cho độ chính xác cao nhưng tốc độ xử lý thấp, không đáp ứng được trong việc phát hiện đối tượng trong thời gian thực. SSD được thiết kế với tốc độ xử lý nhanh mà vẫn đảm bảo độ chính xác.

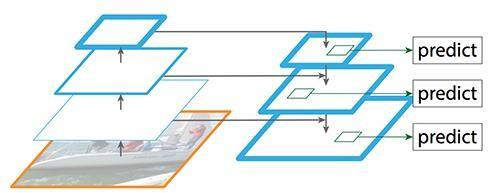
SSD bỏ qua bước tạo mặt nạ region proposal network để đề xuất vùng vật thể, thay vào đó cả quá trình phát hiện vật thể và phân loại vật thể được thực hiện trong cùng một mạng. SSD sử dụng VGG16 làm mạng cơ sở để trích xuất feature map, sau đó là 6 lớp Convolution phụ trợ, 5 trong số chúng để detect object và đưa ra kết quả dự đoán (gồm bounding box và xác suất xuất hiện của các đối tượng trong bounding box). Nhờ sử dụng feature map có kích thước giảm dần theo độ sâu và bounding box với các tỉ lệ khác nhau trên khung hình đã cải thiện được độ chính xác nhận diện vùng vật thể ở nhiều kích thước khác nhau, đồng thời phân loại vật thể làm cho tốc độ xử lý tăng lên nhiều lần [13].



*Hình 2.7. Kiến trúc mô hình SSD [13]*

* *Mô hình phát hiện khuôn mặt dựa trên FPN (Feature Pyramid Network)*

Với phương pháp SSD ta đạt được tốc độ phát hiện đối tượng nhanh hơn, tuy nhiên phương pháp này dễ bỏ sót những đối tượng có kích thước nhỏ. FPN là mô hình mạng được thiết kế dựa trên khái niệm Pyramid để giải quyết vấn đề đó.

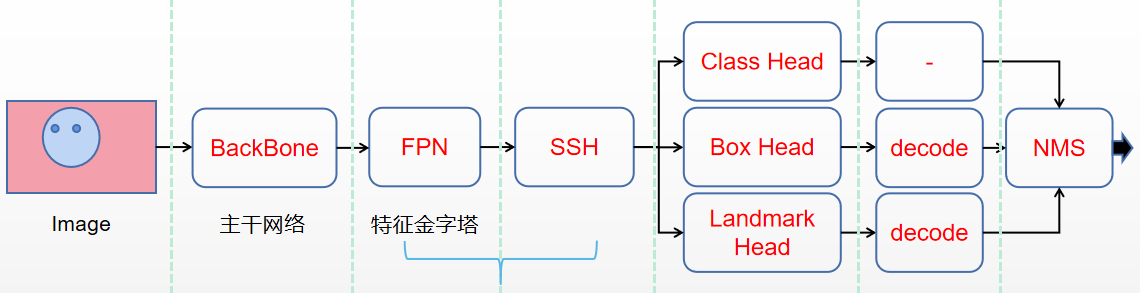
SSD đưa ra quyết định dựa vào nhiều feature map, những layer ở dưới có độ phân giải cao nhưng giá trị ngữ nghĩa thấp không được sử dụng để nhận dạng đối tượng nên SSD không nhận dạng được những đối tượng có kích thước nhỏ. Trong khi đó, FPN xây dựng thêm mô hình *Top-Down* gồm 2 nhánh là *Bottom-Up* và *Top- Down*, nhằm xây dựng các layer có độ phân giải cao từ các layer có ngữ nghĩa cao.

*Hình 2.8. Kiến trúc mô hình FPN*

Trong quá trình xây dựng các layer từ top xuống bottom, chúng ta sẽ gặp vấn đề nghiêm trọng là bị mất mát thông tin của các đối tượng. Để giải quyết vấn đề này, FPN tạo các kết nối *(skip connection)* giữa các reconstruction layer và các feature map.

* *Mô hình phát hiện khuôn mặt dựa trên Retina Face*

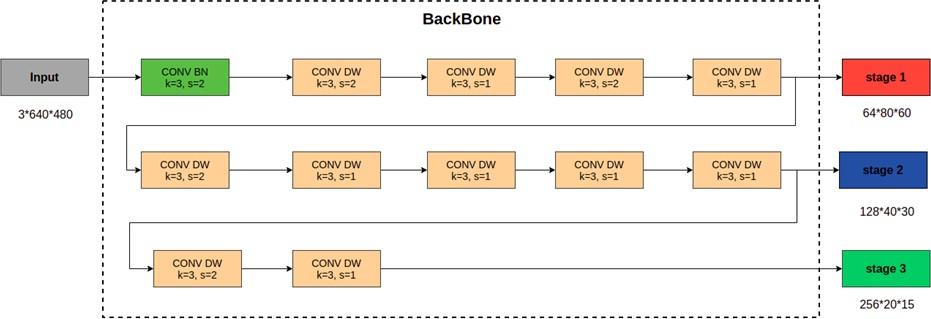
Retina Face là một mô hình phát hiện khuôn mặt đạt được độ chính xác cao trong những bài toán phát hiện khuôn mặt khó (khuôn mặt nhỏ, khuôn mặt nghiêng, bị mờ, bị che khuất). Bằng việc gán nhãn thủ công 5 điểm landmark trên tập dữ liệu

WIDER FACE (bộ dữ liệu được chia thành 3 cấp dễ, trung bình, khó tùy vào độ khó của việc phát hiện và các khuôn mặt có sự thay đổi cao về tỷ lệ, tư thế) và sử dụng hàm *Focal Loss Function* góp phần tăng độ chính xác trong việc phát hiện các khuôn mặt.

*Hình 2.9. Mô hình Retina Face*

Các hình ảnh được phát hiện bởi Retina Face được đưa qua 5 bước xử lý như trên hình:

Sử dụng MobileNet hoặc ResNet50 để khai thác mạng tính năng đường trục (BackBone). BackBone ResNet50 giúp phát hiện khuôn mặt với độ chính xác cao, BackBone MobileNet giúp phát hiện khuôn mặt với tốc độ nhanh hơn và hiệu quả trên các thiết bị di động.



*Hình 2.10. Cấu trúc mạng đường trục MobileNet*

Sau đó sử dụng mô hình FPN và SSH (mô hình Single Stage Headless phát hiện khuôn mặt với các kích thước khác nhau từ các lớp khác nhau trong quá trình lan truyền tiến) để phát hiện các đặc trưng tốt hơn của khuôn mặt.

Cuối cùng sử dụng Class Head, Box Head, Landmark Head để xác định vị trí và 5 điểm chính trên khuôn mặt.

Để tăng cường độ chính xác khi dự báo vị trí chứa khuôn mặt, Retina Face đã sử dụng hàm Focal Loss:



Bài toán object detection thường có sự mất cân bằng nghiêm trọng giữa lớp bounding box có chứa object và lớp bounding box không chứa object (thường background của mỗi bức ảnh sẽ nhiều hơn). Xét hai trường hợp dễ dự báo và khó dự báo:

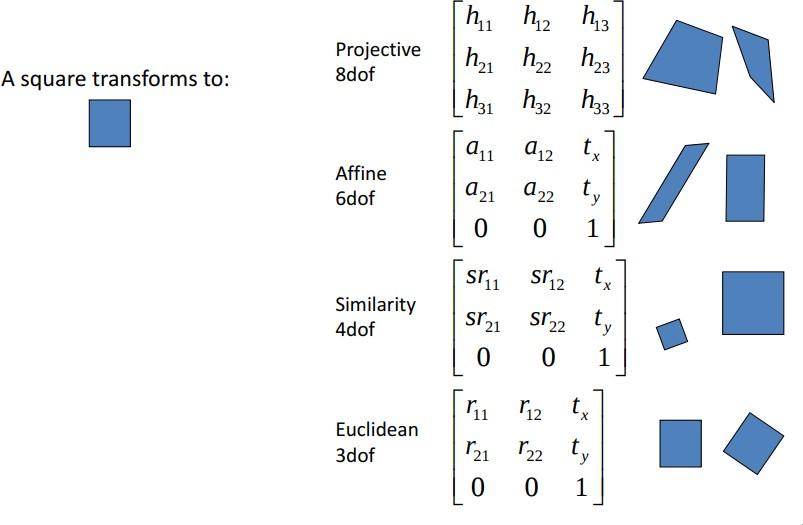
* + Dễ dự báo: Chúng ta thấy rằng mô hình huấn luyện trên mẫu mất cân bằng dễ dàng dự báo chính xác các mẫu đa số. Những trường hợp này được gọi là dễ dự báo. Xác suất 𝑞𝑖 của của các trường hợp dễ dự báo có xu hướng cao hơn. Do đó (*1* − 𝑞𝑖)𝛾 có xu hướng rất nhỏ và dường như không tác động lên loss function đáng kể.
  + Khó dự báo: Trường hợp khó dự báo thì 𝑞𝑖 là một giá trị nhỏ hơn. Do đó độ lớn tác động của nó lên loss function là (*1* − 𝑞𝑖)𝛾 sẽ gần bằng 1. Mức độ tác động này lớn hơn nhiều lần so với trường hợp dễ dự báo.

Do đó hàm Focal Loss có tác dụng rất lớn trong việc điều chỉnh *ảnh hưởng của nhãn* lên đồng thời *loss function* và *gradient descent* đem lại hiệu quả chính xác cao.

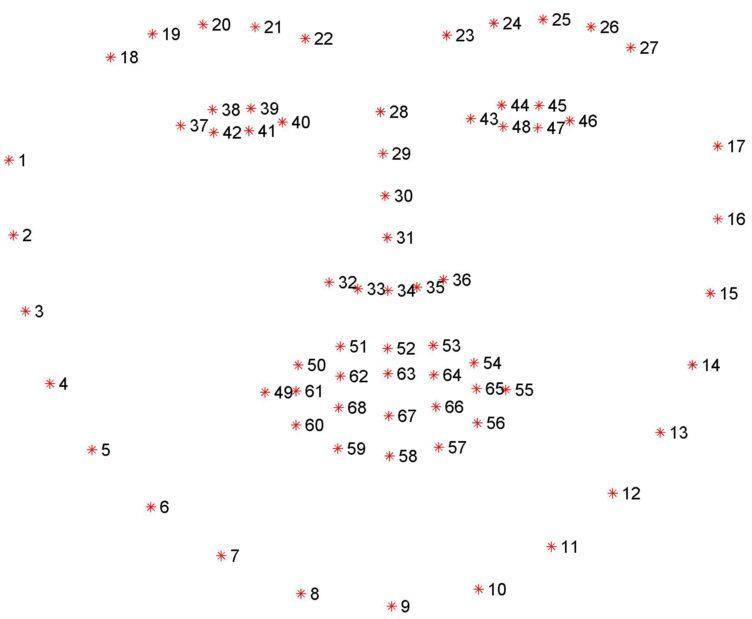
## Bộ căn chỉnh khuôn mặt (Face Alignment)

Sau khi phát hiện và xác định được vị trí khuôn mặt, vùng khuôn mặt được tách ra từ ảnh có thể ở các trạng thái khác nhau, các góc độ khác nhau. Có những khuôn mặt bị chéo và cũng có thể bị lệch do bước detect chưa chính xác trong việc lấy ra khung hình chuẩn của mặt. Vì thế việc căn chỉnh lại khuôn mặt là cần thiết để nâng cao độ chính xác khi nhận diện khuôn mặt.

Để làm được điều này, chúng ta cần xác định được những điểm đặc biệt trên khuôn mặt như mắt, mũi, ... từ đó áp dụng các phương pháp biến đổi hình học như: *Euclidean Transformation, Similarity Transformation, Affine Transformation, Projective Transformation* để khuôn mặt trở về theo quy chuẩn.

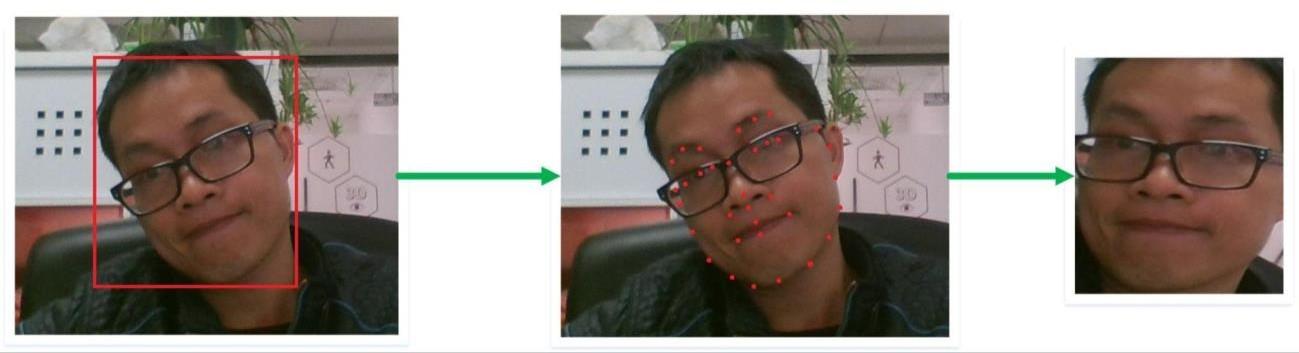


*Hình 2.11. Các phép biến đổi hình học*

Để làm được điều này cần xác định được cấu trúc khuôn mặt. Có rất nhiều kiểu cấu trúc khuôn mặt khác nhau nhưng về cơ bản, chúng ta sẽ xác định các phần miệng, lông mày phải, lông mày trái, mắt phải, mắt trái, mũi, quai hàm. Có rất nhiều thuật toán cho vấn đề này, trong đó có bộ xác định *facial landmark* của thư viện *Dlib* được huấn luyện trên bộ dữ liệu iBUG 300-W. Bộ xác định facial landmark này sẽ xác định 68 điểm chính theo tọa độ (x, y) cấu tạo nên khuôn mặt người.

*Hình 2.12. Ví dụ các điểm facial landmark*

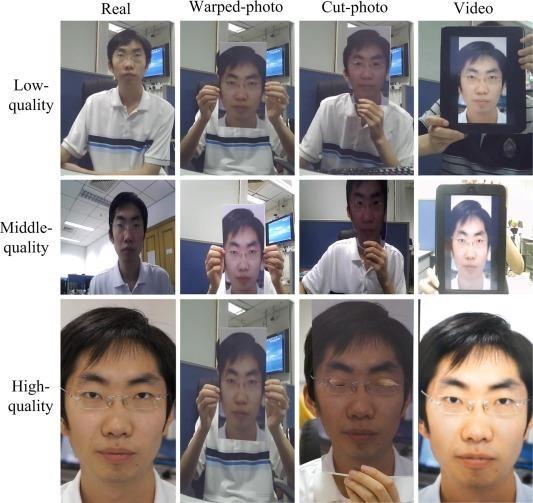
Việc xác định được càng nhiều điểm landmark thì khuôn mặt người càng cụ thể, rõ ràng, chính xác hơn. Từ đó căn chỉnh lại khuôn mặt chuẩn hơn.



*Hình 2.13. Ví dụ alignment face*

## Bộ chống giả mạo khuôn mặt (Face Anti-spoofing)

Phát hiện giả mạo khuôn mặt là một bước quan trọng trong các hệ thống nhận dạng khuôn mặt. Việc hình ảnh hoặc video chứa khuôn mặt của ai đó trên các trang web hay mạng xã hội khiến cho các hệ thống sinh trắc học nhận diện dựa trên khuôn mặt rất dễ bị tấn công khi kẻ tấn công thay vì đưa mặt mình để hệ thống nhận diện thì sử dụng hình ảnh khuôn mặt đã được in trên giấy hoặc xuất hiện trong video quay sẵn của một người có trong hệ thống.



*Hình 2.14. Ví dụ các cách tấn công giả mạo trong các hệ thống bảo mật*

***Một số phương pháp chính cho bài toán chống tấn công giả mạo khuôn mặt:***

* + - *Phương pháp dựa trên các tác động của con người*

Phương pháp này chủ yếu dựa trên những hành động, cử chỉ tự nhiên của người dùng hoặc do hệ thống yêu cầu. Trong đó có 2 phương pháp chính là phát hiện nháy mắt và thử thách - phản hồi.

* + - *Phương pháp phát hiện nháy mắt*

Nháy mắt là hành động tự nhiên của con người, do đó hệ thống sẽ thực hiện quay kiểm tra một chuỗi hình ảnh chứa khuôn mặt có nháy mắt hay không, nếu có nháy mắt thì khuôn mặt là thật và ngược lại, nếu không có nháy mắt thì khuôn mặt là giả mạo.

Phương pháp này dễ triển khai trên các thiết bị, chi phí thực hiện thấp, thuận tiện với người dùng do nháy mắt là hoạt động hoàn toàn tự nhiên của con người, chống tấn công giả mạo bằng hình ảnh in sẵn. Tuy nhiên, phương pháp này có một số hạn chế lớn là không thể phát hiện chống tấn công giả mạo bằng video hoặc sử dụng các sản phẩm 3D.

* + - *Phương pháp thử thách - phản hồi*

Hệ thống sẽ yêu cầu người dùng thực hiện một hoặc một số thao tác, hành động nào đó trên khuôn mặt (thử thách) như: quay mặt sang trái, phải, trên, xuống, nhắm mắt, cười, ..., các hành động này được yêu cầu ngẫu nhiên, không theo thứ tự có sẵn và số lượng để tăng tính bảo mật, sau đó người dùng phải thực hiện (phản hồi) để hệ thống kiểm tra.

Phương pháp thử thách - phản hồi đã khắc phục được nhược điểm lớn nhất của phương pháp phát hiện nháy mắt là người dùng sử dụng video đã quay sẵn. Tuy nhiên, nhược điểm lớn là người dùng có những trải nghiệm không tốt khi phải thực hiện nhiều thao tác mới được nhận diện. Nếu thực hiện sai thao tác nào thì sẽ phải làm lại từ đầu và rất khó cho những người khiếm thị.

* + - *Phương pháp dựa trên các đặc trưng của khuôn mặt*

Phương pháp này chủ yếu dựa trên những đặc trưng của khuôn mặt mà hệ thống trích xuất được sau đó sử dụng các thuật toán phân loại để phân biệt ảnh đầu vào là khuôn mặt thật hay giả mạo. Có 2 phương pháp chính là dùng Local Binary Pattern (LBP) và Support Vector Machine (SVM) và phương pháp dùng mạng nơ- ron tích chập.

* + - * *Phương pháp sử dụng LBP và SVM*

LBP (Local Binary Pattern) là một phương pháp trích rút đặc trưng của ảnh khuôn mặt thành các vector đặc trưng. Các vector đặc trưng sau đó có thể dùng để đưa vào các mô hình học máy để huấn luyện và phân loại. SVM (Support Vector Machine) là một thuật toán học máy nhận đầu vào là các dữ liệu đó để phân chia chúng thành các nhóm riêng biệt.

Phương pháp sử dụng LBP và SVM là phương pháp đơn giản, dễ triển khai trên các thiết bị. Tuy nhiên các đặc trưng mà LBP trích rút được thường bị ảnh hưởng bởi các điều kiện khác nhau về ánh sáng, môi trường, hướng khuôn mặt, nhiễu, ... làm giảm khả năng phân loại.

* + - * *Phương pháp sử dụng mạng nơ-ron tích chập*

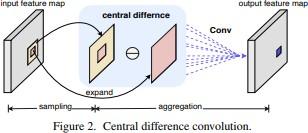
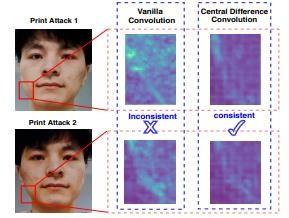
Mạng nơ-ron tích chập như đã trình bày phía trên là một giải pháp tốt nhất cho các bài toán về “Thị giác máy tính” hiện nay. Mạng nơ-ron tích chập chủ yếu sử dụng các lớp tích chập chồng lên nhau, lên bức ảnh khuôn mặt đầu vào nhắm trích xuất các đặc trưng, thông tin trừu tượng tốt hơn về ảnh để phân loại tốt.

Phương pháp sử dụng mạng nơ-ron tích chập phân loại trên ảnh có ưu điểm lớn nhất là dễ triển khai, trích xuất được nhiều đặc trưng của ảnh hơn và không bị ảnh hưởng nhiều bởi các yếu tố như ánh sáng, nhiễu, Tuy nhiên phương pháp này

đòi hỏi nhiều dữ liệu để huấn luyện mạng và đôi khi xảy ra hiện tượng Overfitting (kết quả phân biệt tốt trên bộ dữ liệu huấn luyện nhưng lại không tốt trên các bộ dữ liệu khác).

***Một số nghiên cứu mạng CNN cho bài toán chống giả mạo khuôn mặt:***

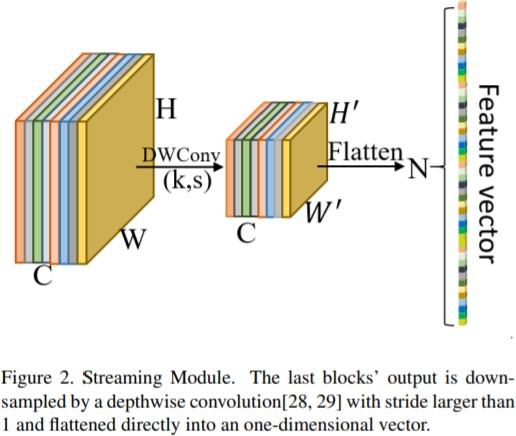
*Nghiên cứu chống giả mạo khuôn mặt sử dụng CDC:*

*Hình 2.15. Cấu trúc Central Different Convolution [5]*

Ý tưởng chính của phương pháp này là sử dụng Central Different Convolution (CDC) thay vì chỉ sử dụng các Convolution bình thường để phát hiện ra những đặc điểm nhất quán trong các bức ảnh giả mạo.

*Nghiên cứu chống giả mạo khuôn mặt FeatherNet:*



*Hình 2.16. Cấu trúc Streaming Module [6]*

Phương pháp này sử dụng các khối Streaming Module, các khối này sử dụng Depthwise Convolution sau đó làm phẳng trực tuyến (tọa độ của phần tử thứ n trong feature vector là tọa độ của phần tử (y, x) trong kênh thứ m.



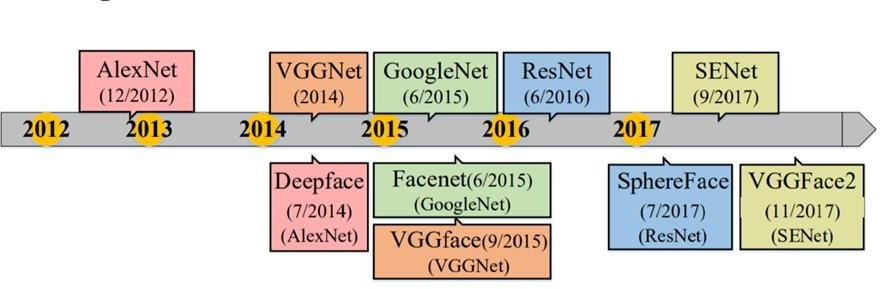
Depthwise Convolution là một loại tích chập trong đó nó chia features map đầu vào thành các group có số channel cố định bằng 1 để thực hiện nhân chập rồi xếp chồng lên nhau để được kết quả đầu ra. Với phương pháp đó, Depthwise Convolution giúp giảm kích thước mô hình và giảm độ phức tạp tính toán.

Làm phẳng trực tuyến giúp khắc phục điểm yếu của Global Average Pooling (GPA - làm giảm kích thước feature map, coi vị trí các điểm ảnh có tầm quan trọng như nhau và điều đó không tốt cho các bài toán về khuôn mặt).

* + - *Phương pháp 3D face shape*

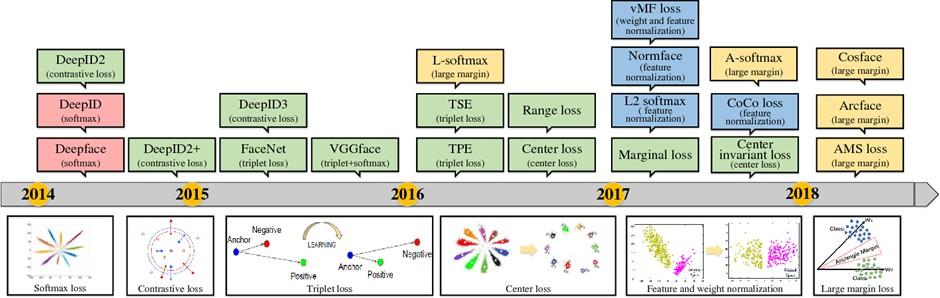
Phương pháp 3D face shape là phương pháp sử dụng các thiết bị cảm biến chiều sâu và cảm biến nhiệt để hỗ trợ phân biệt đầu vào là ảnh thật hay ảnh giả vì cơ thể người sẽ có mức nhiệt cụ thể là khoảng 37℃. Phương pháp này có ưu điểm là tốc độ nhận diện nhanh, chống được các trường hợp giả mạo như dùng ảnh, hoặc video có sẵn, .... Tuy nhiên phương pháp này tốn kém rất nhiều chi phí và không triển khai được trên mobile.

## Bộ trích xuất đặc trưng khuôn mặt (Face Feature Extraction)

Sau khi căn chỉnh khuôn mặt, bước tiếp theo là trích chọn đặc trưng của khuôn mặt phục vụ cho nhận diện khuôn mặt. Trích xuất đặc trưng của khuôn mặt là một bước rất quan trọng trong các công nghệ như face tracking, nhận dạng cảm xúc khuôn mặt hay nhận diện khuôn mặt. Ảnh khuôn mặt sẽ được đi qua một bộ trích chọn đặc trưng mà đầu ra ở đây là một vector biểu thị cho khuôn mặt. Có rất nhiều phương pháp để trích chọn đặc trưng của một bức ảnh. Các phương pháp truyền thống thường sử dụng các bộ trích chọn đặc trưng hand-craft feature (đặc trưng cứng) như: HOG, SIFT, LBP, … Tuy nhiên, các bộ trích chọn đặc trưng sử dụng mạng CNN lại có hiệu quả vượt bậc so với các bộ trích chọn đặc trưng truyền thống.

*Hình 2.17. Các kiến trúc CNN được dùng cho nhận diện khuôn mặt qua các năm [7]*

Cùng với sự phát triển của các kiến trúc mạng là sự phát triển của các hàm Loss Function để tối ưu kết quả cho bài toán. Nhiều loss function có thể kể đến như *Euclidean Distance Based Loss, Angular/Cosine-Margin-Based Loss* và *Softmax Loss* và một số biến thể của nó.



*Hình 2.18. Các Loss Functions phục vụ cho nhận diện khuôn mặt qua các năm [7]*

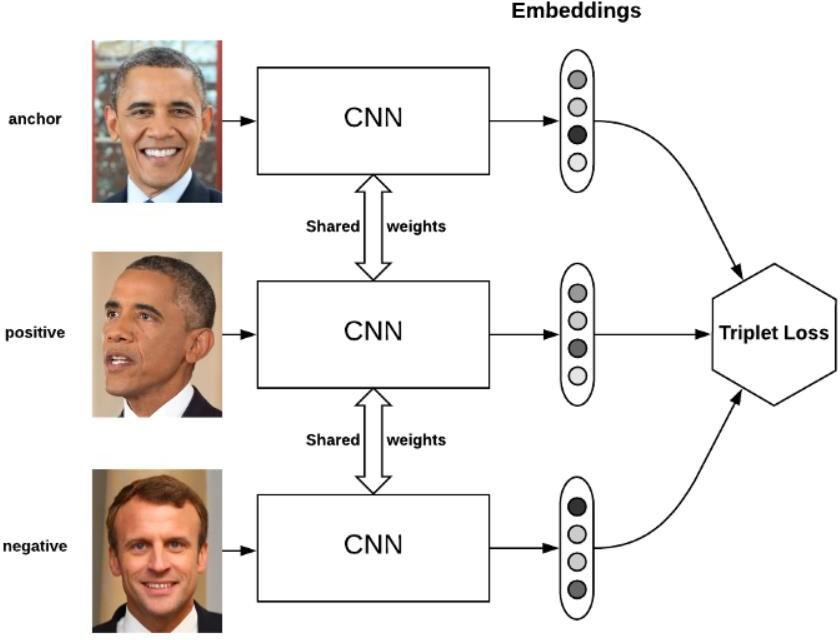
***Một số Loss Function:***

* + - ***Triplet Loss***

Thay vì lấy hai giá trị đầu vào, Triplet Loss đưa ra một công thức mới gồm 3 giá trị đầu vào: Anchor ( 𝑥𝑎 ảnh đầu vào của mạng), Positive ( 𝑥𝑝 ảnh cùng là một

𝑖 𝑖

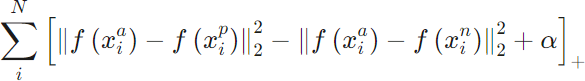
người với Anchor), Negative ( 𝑥𝑛 ảnh khuông cùng là một người với Anchor). Sau đó các ảnh đầu vào sẽ được trích xuất đặc trưng thành các embedding (embedding là một vector với số chiều cố định nhỏ hơn feature vector bình thường, đại diện cho các feature và được dùng trong nhiệm vụ phân loại các đối tượng) rồi thực hiện phân loại [15].



*Hình 2.19. Hàm Triplet Loss [14]*

Mô tả đầu vào của Triplet Loss

Hàm Triplet Loss được tính theo công thức:



𝑓(𝑥𝑎) chính là embedding của 𝑥𝑎. Công thức trên cho ta thấy khoảng cách

𝑖 𝑖

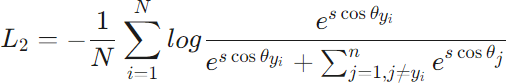
giữa 2 embedding của Anchor và Positive với Anchor và Negative phải nhỏ hơn ít nhất một số 𝛼. Để mạng học được nhiều hơn, điểm Positive được chọn phải nằm xa nhất có thể so với Anchor và điểm Negative phải nằm gần nhất có thể so với Anchor là những trường hợp xấu nhất mà mạng học cần vượt qua [15].

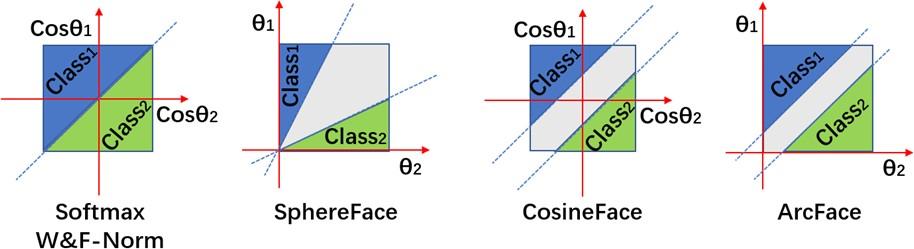
Như vậy khi áp dụng Triple loss vào các mô hình convolutional neural network chúng ta có thể tạo ra các biểu diễn vector tốt nhất cho mỗi một bức ảnh. Những biểu diễn vector này sẽ phân biệt tốt các ảnh Negative rất giống ảnh Positive. Và đồng thời các bức ảnh thuộc cùng một label sẽ trở nên gần nhau hơn trong không gian chiếu euclidean.

Tuy nhiên, khi gặp dataset lớn sẽ có sự bùng nổ về số lượng bộ ba triplet samples dẫn đến sự lặp lại đáng kể các bước.

***ArcFace (Additive Angular Margin Loss)***

ArcFace được cải tiến dựa trên hàm Softmax Loss. Công thức tính ArcFace:





*Hình 2.20. ArcFace và một số hàm Loss khác [4]*

Ý nghĩa của việc dùng hàm mất mát ArcFace là làm cho các mẫu cùng loại co cụm, tăng sự khác biệt giữa các lớp. Việc sử dụng Softmax sẽ giúp chúng ta phân tách các vector đặc trưng tuy nhiên lại tạo nên một sự mơ hồ trên đường ranh giới quyết định, trong khi đó hàm ArcFace lại có sự phân chia rõ rệt giữa các lớp gần nhau. Như vậy hàm ArcFace có thể kết hợp cùng với rất nhiều mô hình CNN để tạo ra các mô hình nhận diện khuôn mặt tốt hơn.

## Bộ nhận diện khuôn mặt

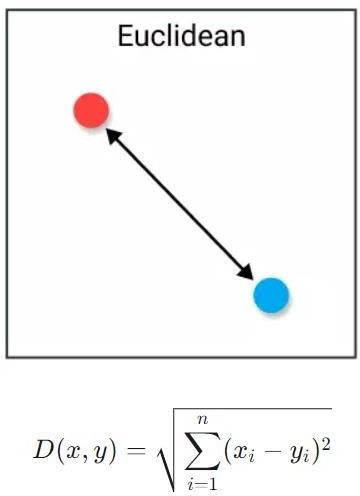
Sau khi trích chọn đặc trưng, mỗi khuôn mặt sẽ có các feature vector. Tùy vào ứng dụng thực tế mà việc nhận diện khuôn mặt được phân loại thành các bài toán khác nhau: *Face Verification* hay *Face Identification*.

* *Face Verification:* bài toán xác minh các ảnh có phải là của chung một người không, ứng dụng nhiều nhất của bài toán này là bảo mật thông tin. Một khuôn mặt sẽ được đăng ký với hệ thống và sau đó so sánh với khuôn mặt đầu vào để mở khóa hệ thống.
* *Face Identification:* bài toán xác định danh tính của khuôn mặt với một tập các khuôn mặt. Trong đó có 2 bài toán con là *Open-set Face Identification* (khuôn mặt nhận diện sẽ được phân vào một trong các khuôn mặt nằm trong tập thư viện) và *Close-set Face Identification* (bài toán xác định khuôn mặt có nằm trong tập thư viện không rồi mới xác định danh tính của khuôn mặt đó).

Ở đây, bài toán sẽ là xác định khuôn mặt *Close-set Face Identification*. Với mỗi khuôn mặt đầu vào cần xác minh, sử dụng một độ đo thích hợp: *Euclidean Distance, Manhattan Distance, Cosine Similarity*, ... để tính độ tương đồng giữa vector đặc trưng của ảnh đầu vào với vector đặc trưng của các ảnh trong cơ sở dữ liệu. Cụ thể là khuôn mặt nhận diện sẽ được phân vào một trong các khuôn mặt nằm trong thư viện khi vượt quá một giá trị ngưỡng cho phép. Trường hợp giá trị đo độ giống nhau không vượt qua ngưỡng thì khuôn mặt đó không có trong tập dữ liệu và không được định danh.

### Một số phương pháp tính khoảng cách:

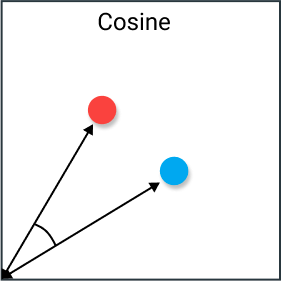
* + - ***Euclidean Distance***

Euclidean Distance còn được biết đến với cái tên 𝐿*2* distance và được tính bằng căn bậc hai tổng bình phương hiệu của các điểm.

*Hình 2.21. Euclidean Distance [16]*

Phương pháp tính này phổ biến và có hiệu quả tốt, đặc biệt với những dữ liệu ít chiều. Tuy nhiên Euclide distance có thể bị ảnh hưởng bởi đơn vị của feature, vì vậy cần phải normalize trước khi tính toán.

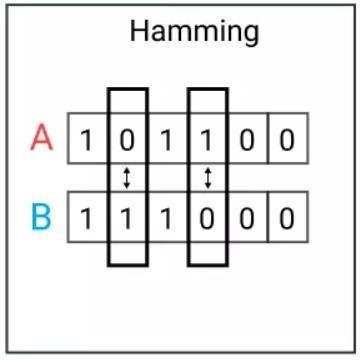
### Cosine Similarity

Cosine Similarity thường được sử dụng để giải quyết vấn đề của Euclidean distance ở không gian nhiều chiều. Ý tưởng đơn giản là tính góc tạo thành giữa 2 vector.

*Hình 2.22. Cosine Similarity [16]*

Phương pháp này thường được sử dụng trong các dữ liệu đa chiều và không quá phụ thuộc vào độ lớn của vector.

### Hamming Distance



*Hình 2.23. Hamming Distance [16]*

Khoảng cách Hamming là số giá trị khác nhau giữa 2 vector. Phương pháp này thường được dùng để so sánh 2 chuỗi binary có độ dài bằng nhau. Phương pháp này cũng có thể so sánh độ tương đồng của vector bằng cách tính số lượng ký tự khác nhau.

Chương 2 đã trình bày kiến trúc chính nhận diện khuôn mặt của hệ thống điểm danh bằng khuôn mặt gồm 5 phần: bộ phát hiện khuôn mặt, bộ căn chỉnh khuôn mặt, bộ chống giả mạo, bộ trích xuất đặc trưng, bộ nhận diện khuôn mặt và khảo sát các phương pháp được sử dụng trong các khâu của hệ thống. Ở phần sau sẽ trình bày chi tiết thiết kế của hệ thống cũng như các phương pháp được lựa chọn thực thi cho từng phần.

# CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

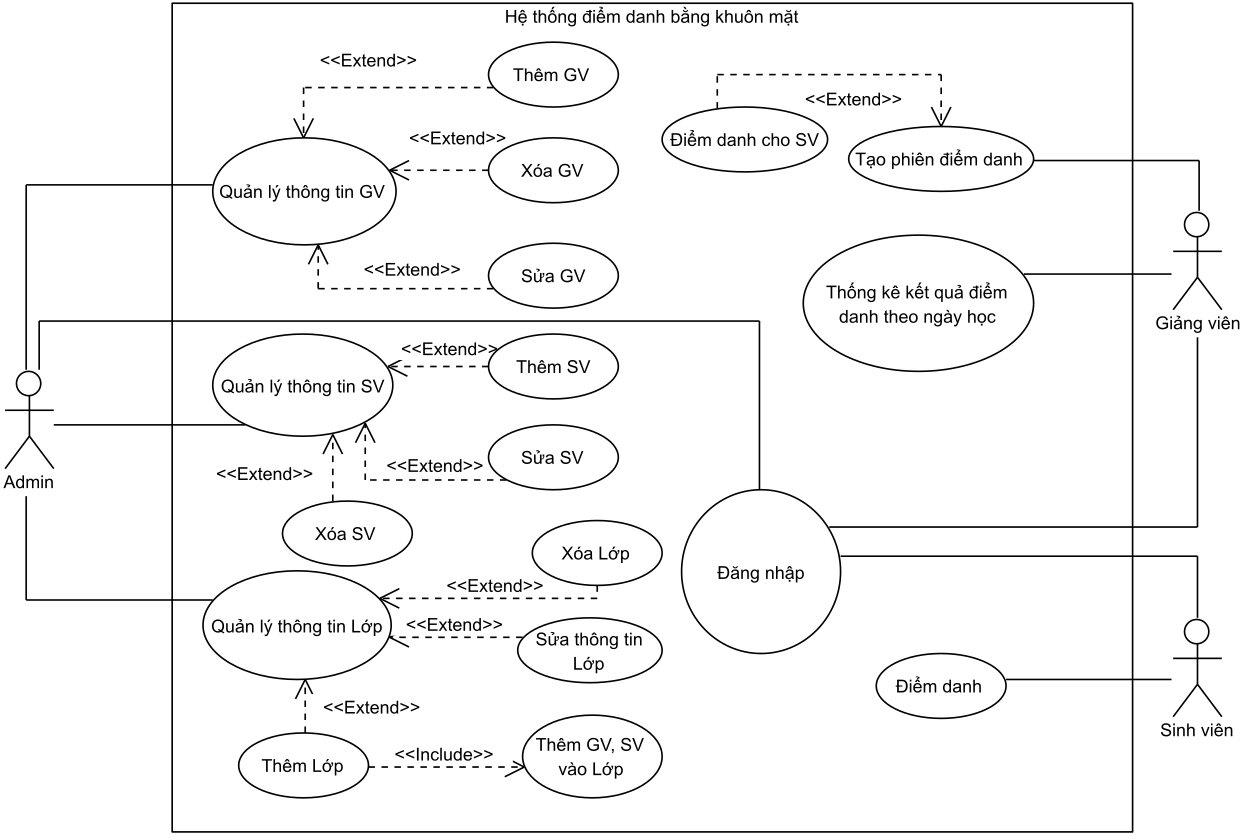
## Phân tích hệ thống

### Mô tả các chức năng chính của hệ thống:

Hệ thống điểm danh bằng khuôn mặt sẽ gồm 3 actor chính: Admin, Giảng viên, Sinh viên. Các chức năng chính của các actor sau khi đăng nhập vào hệ thống:

* Admin:
  + Thêm, sửa, xóa tài khoản giảng viên.
  + Thêm, sửa xóa tài khoản sinh viên.
  + Thêm, sửa, xóa lớp học: thêm sinh viên vào lớp học, gán giảng viên cho lớp học.
* Giảng viên:
  + Tạo phiên điểm danh cho lớp học. Mỗi ngày học của một lớp học có thể không có hoặc có nhiều phiên điểm danh. Mỗi khi điểm danh Giảng viên tạo một phiên mới và gửi thông báo cần phải điểm danh đến mỗi sinh viên trong lớp học.
  + Giảng viên có quyền chỉnh sửa kết quả điểm danh của session khi đang open.
  + Giảng viên có chức năng thống kê kết quả điểm danh của lớp học.
* Sinh viên:
  + Sinh viên có chức năng điểm danh trong session của lớp học.

### Use Case tổng quan của hệ thống:



*Hình 3.1. Biểu đồ Use Case của toàn hệ thống*

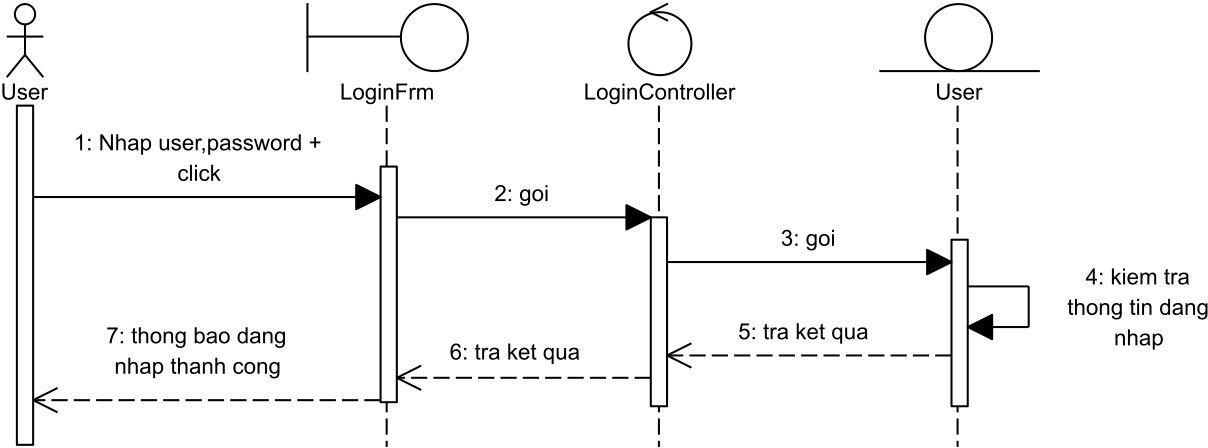
### Đặc tả các Use Case của hệ thống:

Đăng nhập

| **Use Case** | Đăng nhập |
| --- | --- |
| **Actor** | Người dùng |
| **Tiền điều kiện** | Người dùng truy cập vào ứng dụng |
| **Hậu điều kiện** | Người dùng đăng nhập thành công |
| **Chuỗi sự kiện chính** | 1. Người dùng nhập thông tin đăng nhập gồm username (mã người dùng) và password rồi click nút Đăng nhập. 2. Hệ thống thông báo đăng nhập thành công và hiện lên giao diện chính ứng với người dùng. |

|  |  |
| --- | --- |
| **Ngoại lệ** | Người dùng nhập sai username hoặc password. |

*Bảng 3.1. Kịch bản ca sử dụng đăng nhập*

*Sequence diagram*

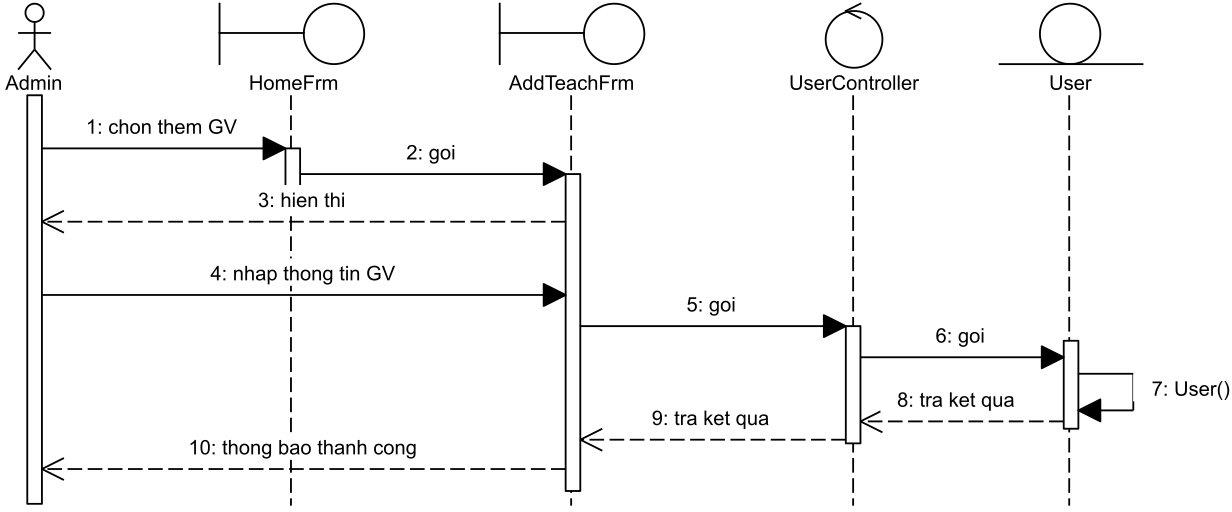
*Hình 3.2. Biểu đồ tuần tự chức năng đăng nhập*

Các Use Case của Admin

* Thêm thông tin GV

| **Use Case** | Thêm thông tin GV |
| --- | --- |
| **Actor** | Admin |
| **Tiền điều kiện** | Đăng nhập thành công vào hệ thống |
| **Hậu điều kiện** | Thêm thông tin GV thành công |
| **Chuỗi sự kiện chính** | 1. Amin chọn chức năng Quản lý thông tin GV trên giao diện chính của hệ thống. 2. Hệ thống hiển thị các chức năng: Thêm GV, Sửa GV, Xóa GV. 3. Admin chọn chức năng Thêm GV. 4. Hệ thống hiển thị giao diện Thêm GV. 5. Admin nhập các thông tin của GV và click Thêm GV. 6. Hệ thống lưu GV vào CSDL và thông báo thành công. |
| **Ngoại lệ** |  |

*Bảng 3.2. Kịch bản ca sử dụng thêm thông tin GV*

*Sequence diagram*

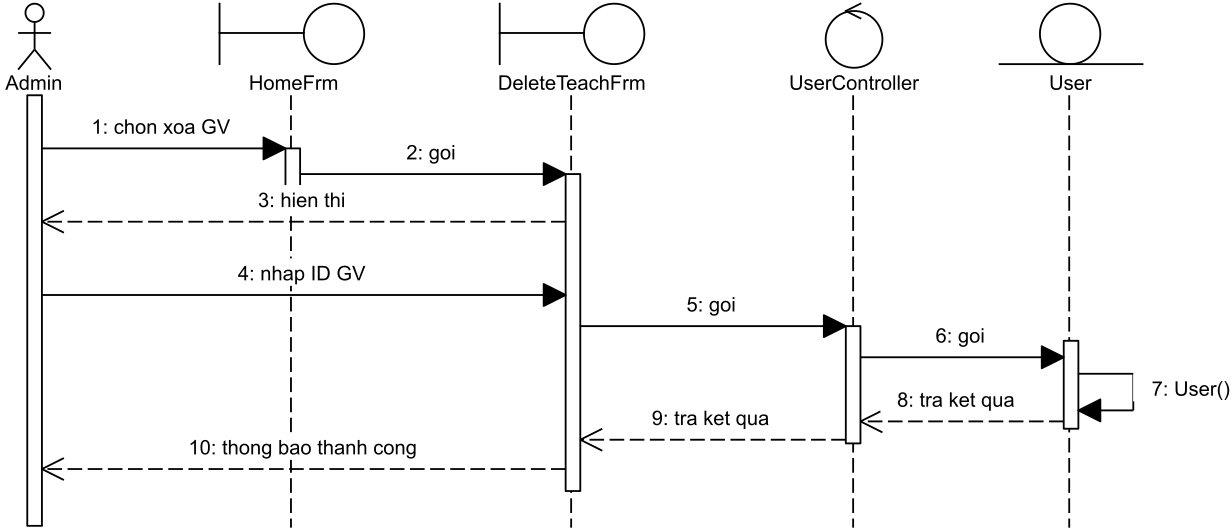
*Hình 3.3. Biểu đồ tuần tự chức năng thêm thông tin GV*

* Xóa Thông tin GV

| **Use Case** | Xóa thông tin GV |
| --- | --- |
| **Actor** | Admin |
| **Tiền điều kiện** | Đăng nhập thành công vào hệ thống |
| **Hậu điều kiện** | Xóa thông tin GV thành công |
| **Chuỗi sự kiện chính** | 1. Amin chọn chức năng Quản lý thông tin GV trên giao diện chính của hệ thống. 2. Hệ thống hiển thị các chức năng: Thêm GV, Sửa GV, Xóa GV. 3. Admin chọn chức năng Xóa GV. 4. Hệ thống hiển thị giao diện Xóa GV. 5. Admin nhập vào mã của GV và click Xóa GV. 6. Hệ thống Xóa GV và thông báo thành công. |
| **Ngoại lệ** | Mã GV nhập vào không tồn tại |

*Bảng 3.3. Kịch bản ca sử dụng xóa GV*

*Sequence diagram*

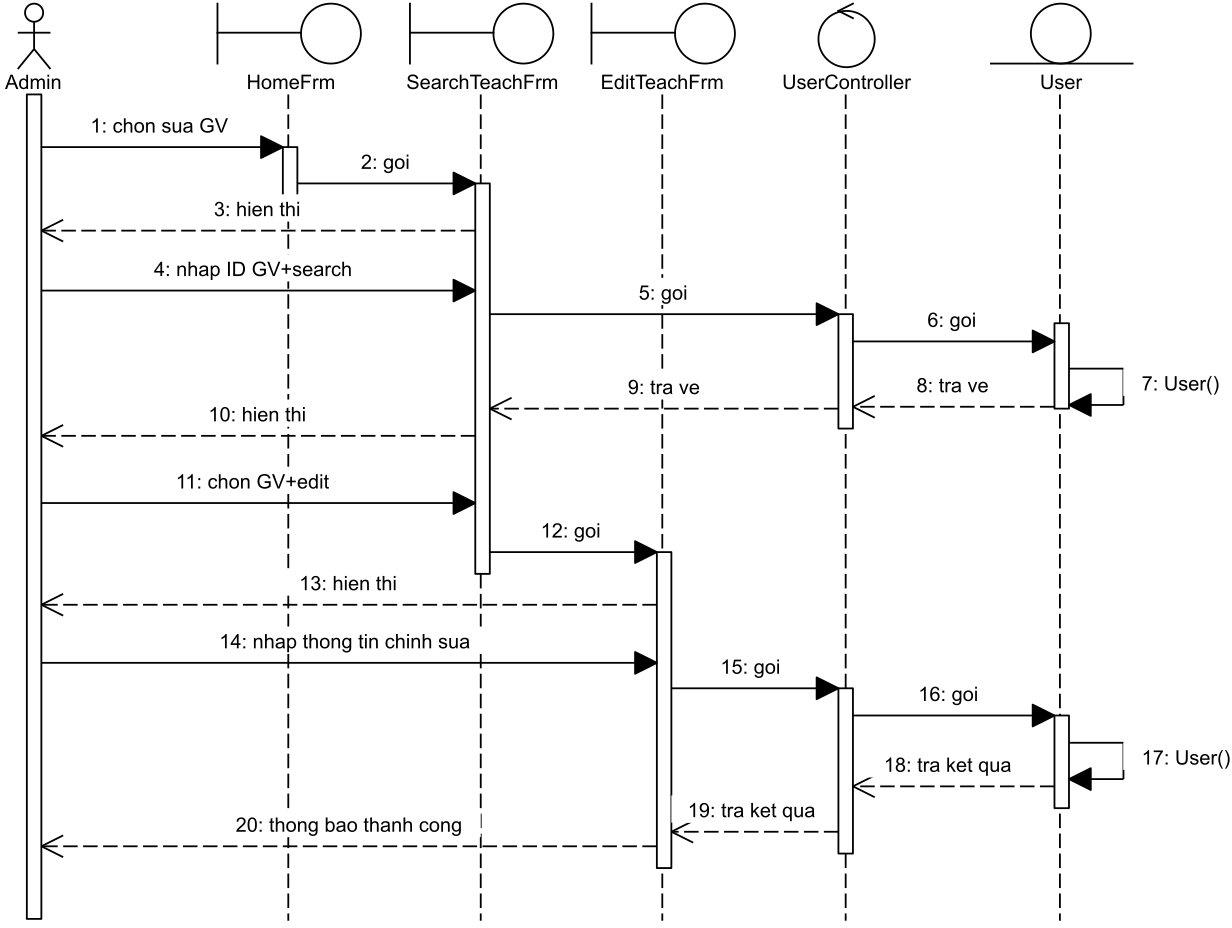


*Hình 3.4. Biểu đồ tuần tự chức năng xóa GV*

* Chỉnh sửa thông tin GV

| **Use Case** | Sửa thông tin GV |
| --- | --- |
| **Actor** | Admin |
| **Tiền điều kiện** | Đăng nhập thành công vào hệ thống |
| **Hậu điều kiện** | Sửa thông tin GV thành công |
| **Chuỗi sự kiện chính** | 1. Amin chọn chức năng Quản lý thông tin GV trên giao diện chính của hệ thống. 2. Hệ thống hiển thị các chức năng: Thêm GV, Sửa GV, Xóa GV. 3. Admin chọn chức năng Sửa GV. 4. Hệ thống hiển thị giao diện Sửa GV. 5. Admin nhập vào mã của GV và click chọn GV. 6. Hệ thống hiển thị giao diện thông tin của GV. 7. Admin thực hiện sửa thông tin của GV rồi click Sửa. 8. Hệ thống lưu lại thông tin GV vào CSDL và thông báo thành công. |
| **Ngoại lệ** | Mã GV nhập vào không tồn tại |

*Bảng 3.4. Kịch bản ca sử dụng sửa thông tin GV*

*Sequence diagram*

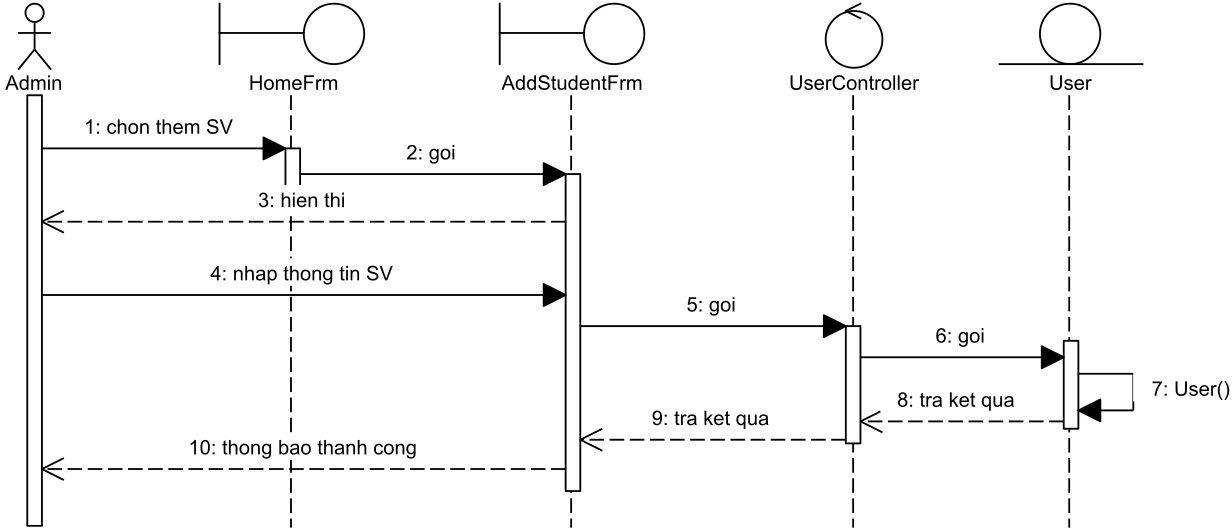
*Hình 3.5. Biểu đồ tuần tự chức năng sửa thông tin GV*

* Thêm thông tin SV

| **Use Case** | Thêm thông tin SV |
| --- | --- |
| **Actor** | Admin |
| **Tiền điều kiện** | Đăng nhập thành công vào hệ thống |
| **Hậu điều kiện** | Thêm thông tin SV thành công |
| **Chuỗi sự kiện chính** | 1. Amin chọn chức năng Quản lý thông tin SV trên giao diện chính của hệ thống. 2. Hệ thống hiển thị các chức năng: Thêm SV, Sửa SV, Xóa SV. 3. Admin chọn chức năng Thêm SV. 4. Hệ thống hiển thị giao diện Thêm SV. |

|  | 1. Admin nhập các thông tin của SV và click Thêm SV. 2. Hệ thống lưu SV vào CSDL và thông báo thành công. |
| --- | --- |
| **Ngoại lệ** |  |

*Bảng 3.5. Kịch bản ca sử dụng thêm thông tin SV*

*Sequence diagram*

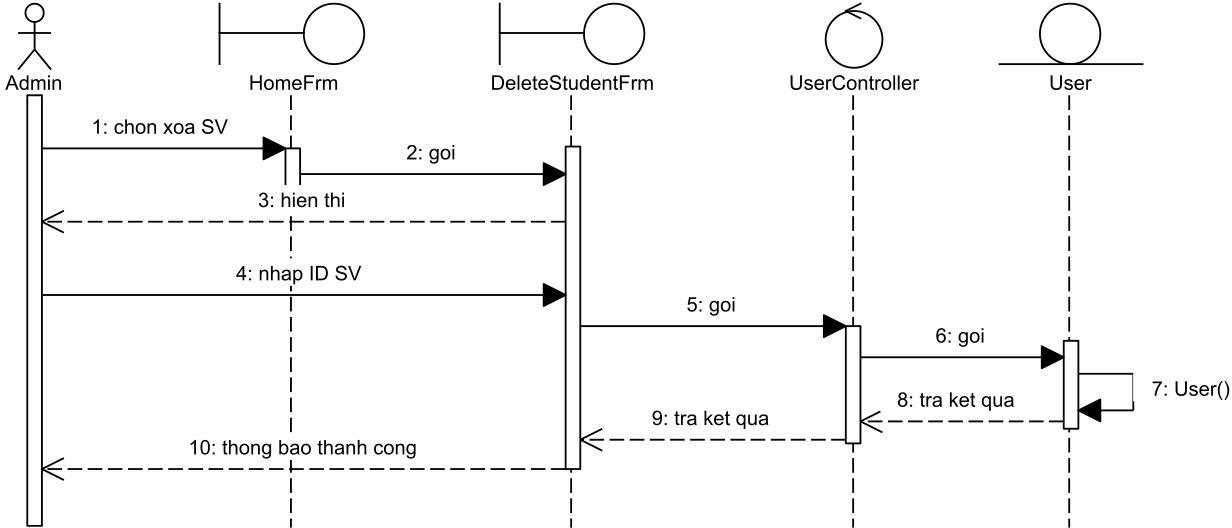
*Hình 3.6. Biểu đồ tuần tự chức năng thêm thông tin SV*

* Xóa Thông tin SV

| **Use Case** | Xóa thông tin SV |
| --- | --- |
| **Actor** | Admin |
| **Tiền điều kiện** | Đăng nhập thành công vào hệ thống |
| **Hậu điều kiện** | Xóa thông tin SV thành công |
| **Chuỗi sự kiện chính** | 1. Amin chọn chức năng Quản lý thông tin SV trên giao diện chính của hệ thống. 2. Hệ thống hiển thị các chức năng: Thêm SV, Sửa SV, Xóa SV. 3. Admin chọn chức năng Xóa SV. 4. Hệ thống hiển thị giao diện Xóa SV. 5. Admin nhập vào mã của SV và click Xóa SV. |

|  | 6. Hệ thống Xóa SV và thông báo thành công. |
| --- | --- |
| **Ngoại lệ** | Mã SV nhập vào không tồn tại |

*Bảng 3.6. Kịch bản ca sử dụng xóa SV*

*Sequence diagram*

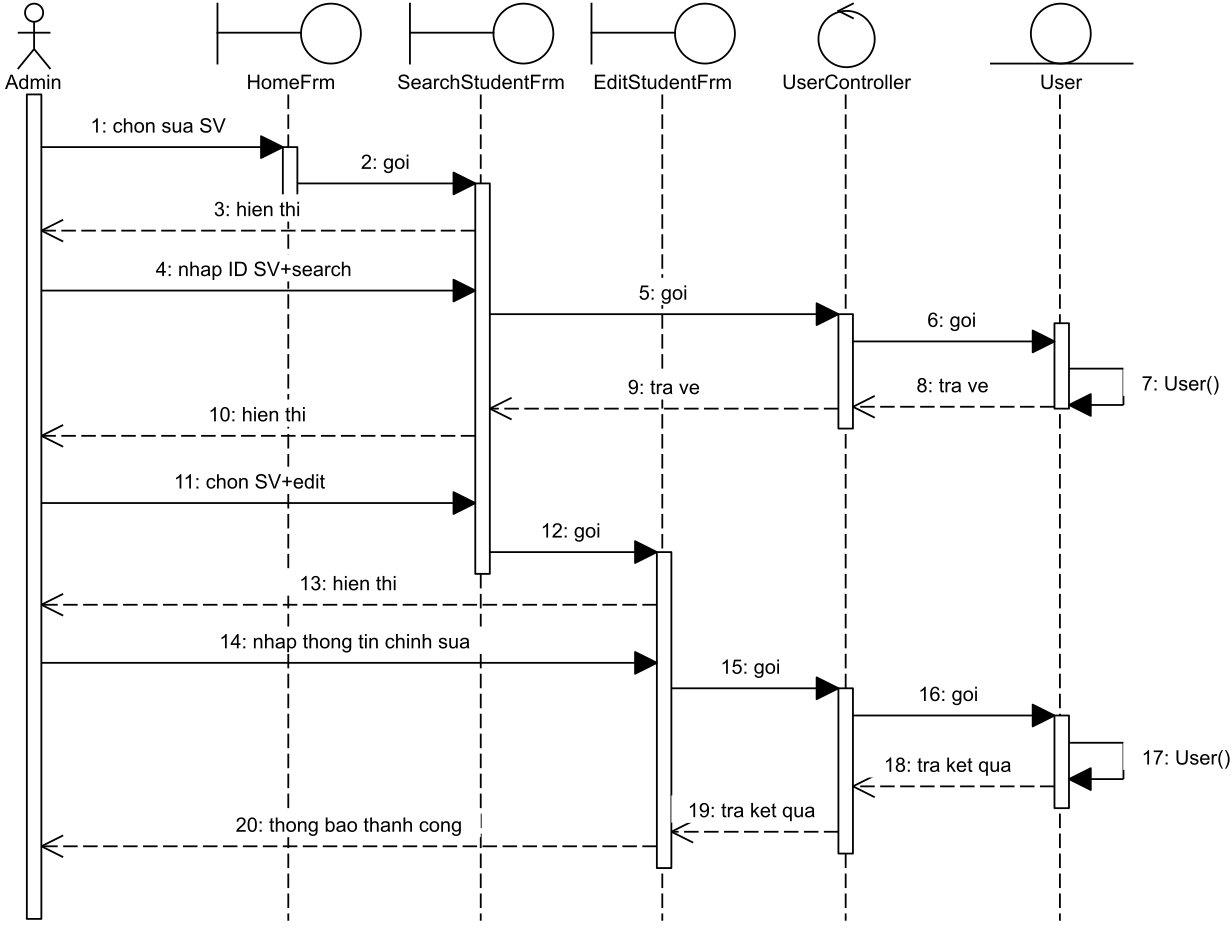
*Hình 3.7. Biểu đồ tuần tự chức năng xóa SV*

* Chỉnh sửa thông tin SV

| **Use Case** | Sửa thông tin SV |
| --- | --- |
| **Actor** | Admin |
| **Tiền điều kiện** | Đăng nhập thành công vào hệ thống |
| **Hậu điều kiện** | Sửa thông tin SV thành công |
| **Chuỗi sự kiện chính** | 1. Amin chọn chức năng Quản lý thông tin SV trên giao diện chính của hệ thống. 2. Hệ thống hiển thị các chức năng: Thêm SV, Sửa SV, Xóa SV. 3. Admin chọn chức năng Sửa SV. 4. Hệ thống hiển thị giao diện Sửa SV. 5. Admin nhập vào mã của SV và click chọn SV. 6. Hệ thống hiển thị giao diện thông tin của SV. |

|  | 1. Admin thực hiện sửa thông tin của SV rồi click Sửa. 2. Hệ thống lưu lại thông tin SV vào CSDL và thông báo thành công. |
| --- | --- |
| **Ngoại lệ** | Mã SV nhập vào không tồn tại |

*Bảng 3.7. Kịch bản ca sử dụng sửa thông tin SV*

*Sequence diagram*

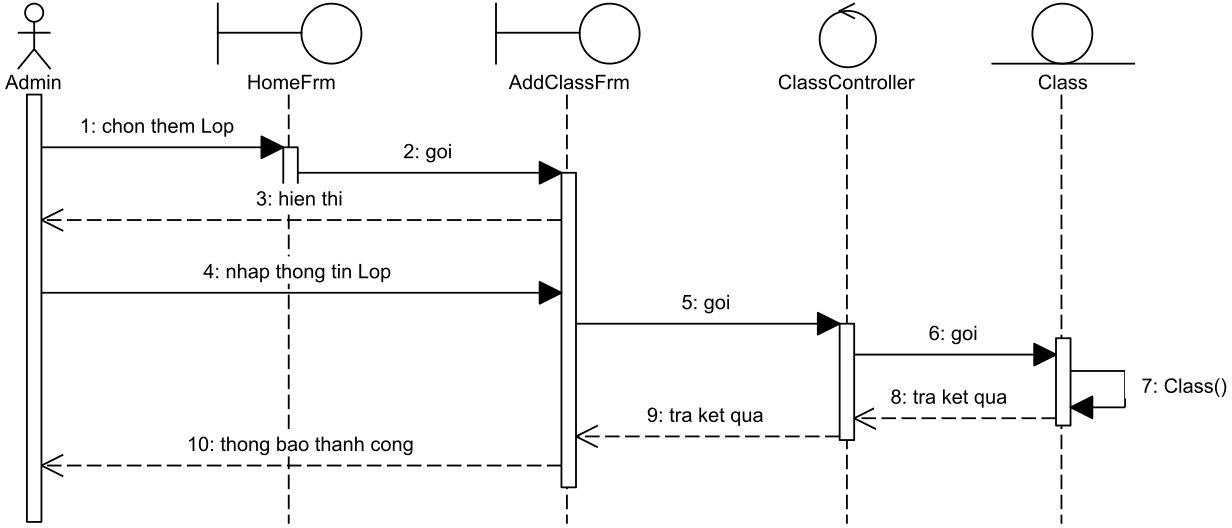
*Hình 3.8. Biểu đồ tuần tự chức năng sửa thông tin SV*

* Thêm thông tin Lớp học

| **Use Case** | Thêm thông tin Lớp |
| --- | --- |
| **Actor** | Admin |
| **Tiền điều kiện** | Đăng nhập thành công vào hệ thống |
| **Hậu điều kiện** | Thêm thông tin Lớp thành công |

| **Chuỗi sự kiện chính** | 1. Amin chọn chức năng Quản lý thông tin Lớp trên giao diện chính của hệ thống. 2. Hệ thống hiển thị các chức năng: Thêm Lớp, Sửa Lớp, Xóa Lớp. 3. Admin chọn chức năng Thêm Lớp. 4. Hệ thống hiển thị giao diện Thêm Lớp. 5. Admin nhập các thông tin của Lớp gồm thông tin GV, danh sách SV và click Thêm Lớp. 6. Hệ thống lưu Lớp vào CSDL và thông báo thành công. |
| --- | --- |
| **Ngoại lệ** |  |

*Bảng 3.8. Kịch bản ca sử dụng thêm thông tin lớp học Sequence diagram*



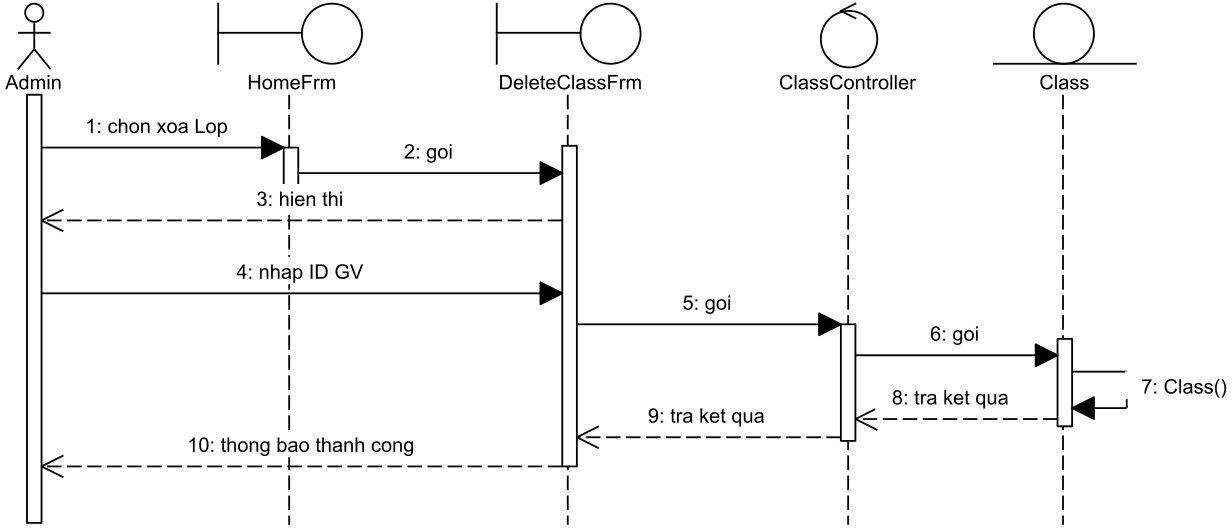
*Hình 3.9. Biểu đồ ca sử dụng thêm thông tin lớp học*

* Xóa Thông tin Lớp

| **Use Case** | Xóa thông tin Lớp |
| --- | --- |
| **Actor** | Admin |
| **Tiền điều kiện** | Đăng nhập thành công vào hệ thống |
| **Hậu điều kiện** | Xóa thông tin Lớp thành công |

| **Chuỗi sự kiện chính** | 1. Amin chọn chức năng Quản lý thông tin Lớp trên giao diện chính của hệ thống. 2. Hệ thống hiển thị các chức năng: Thêm Lớp, Sửa Lớp, Xóa Lớp. 3. Admin chọn chức năng Xóa Lớp. 4. Hệ thống hiển thị giao diện Xóa Lớp. 5. Admin nhập vào mã của Lớp và click Xóa Lớp. 6. Hệ thống Xóa Lớp và thông báo thành công. |
| --- | --- |
| **Ngoại lệ** | Mã Lớp nhập vào không tồn tại |

*Bảng 3.9. Kịch bản ca sử dụng xóa lớp học*

*Sequence diagram*

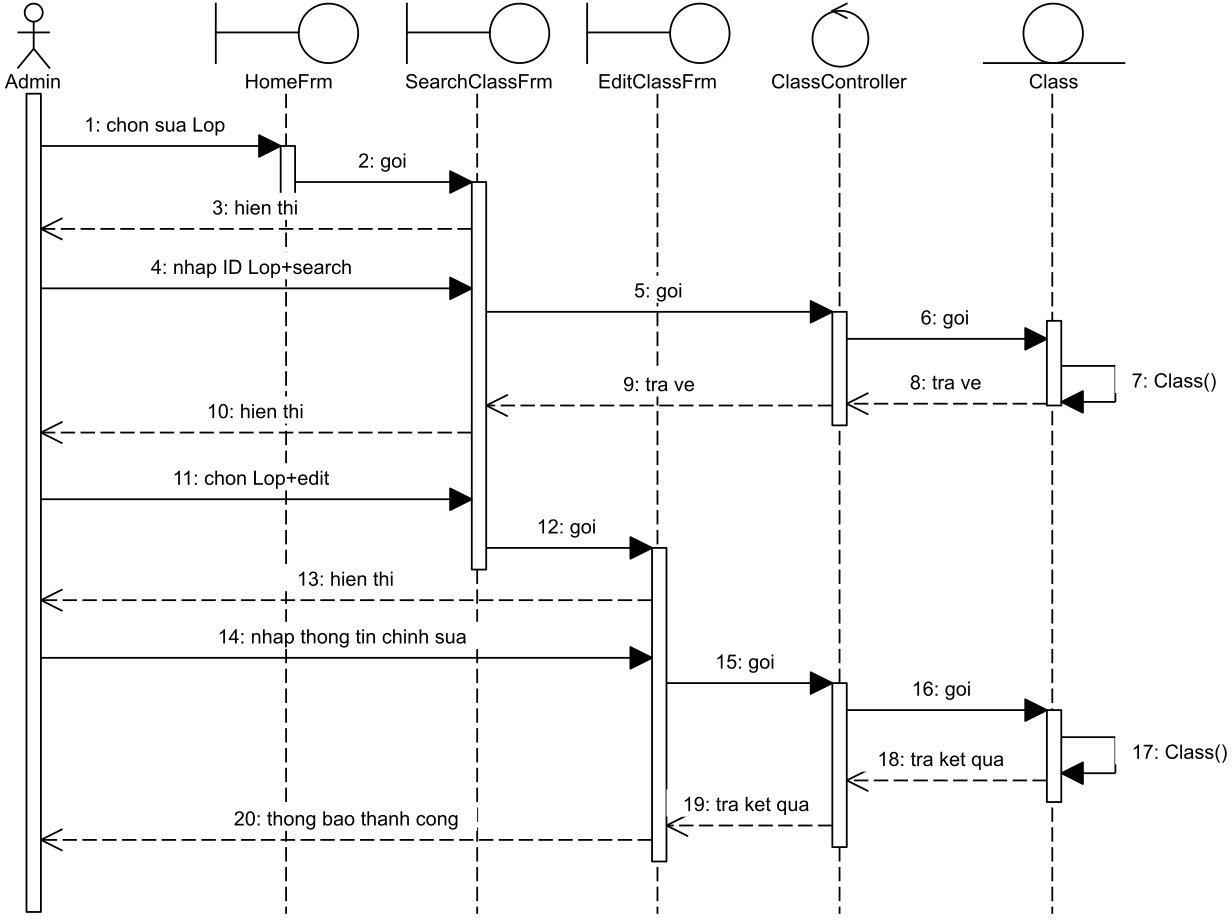
*Hình 3.10. Biểu đồ ca sử dụng xóa lớp học*

* Chỉnh sửa thông tin Lớp

| **Use Case** | Sửa thông tin Lớp |
| --- | --- |
| **Actor** | Admin |
| **Tiền điều kiện** | Đăng nhập thành công vào hệ thống |
| **Hậu điều kiện** | Sửa thông tin Lớp thành công |
| **Chuỗi sự kiện** | 1. Amin chọn chức năng Quản lý thông tin Lớp trên giao |

| **chính** | diện chính của hệ thống.   1. Hệ thống hiển thị các chức năng: Thêm Lớp, Sửa Lớp, Xóa Lớp. 2. Admin chọn chức năng Sửa Lớp. 3. Hệ thống hiển thị giao diện Sửa Lớp. 4. Admin nhập vào mã của Lớp và click chọn Lớp. 5. Hệ thống hiển thị giao diện thông tin của Lớp. 6. Admin thực hiện sửa thông tin của Lớp rồi click Sửa. 7. Hệ thống lưu lại thông tin Lớp vào CSDL và thông báo thành công. |
| --- | --- |
| **Ngoại lệ** | Mã Lớp nhập vào không tồn tại |

*Bảng 3.10. Kịch bản ca sử dụng sửa thông tin lớp học Sequence diagram*



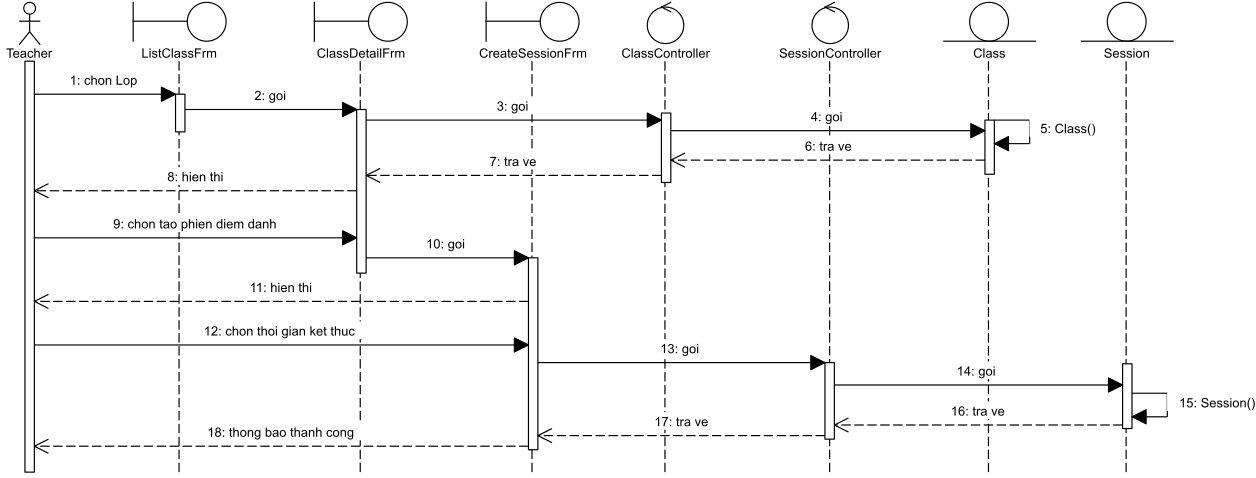
*Hình 3.11. Biểu đồ ca sử dụng sửa thông tin lớp học*

**Các Use Case của Giảng viên**

* Tạo phiên điểm danh

| **Use Case** | Tạo phiên điểm danh |
| --- | --- |
| **Actor** | Giảng viên |
| **Tiền điều kiện** | Đăng nhập thành công vào hệ thống |
| **Hậu điều kiện** | Tạo phiên điểm danh cho lớp học thành công |
| **Chuỗi sự kiện chính** | 1. Giảng viên chọn lớp học vào click tạo phiên điểm danh. 2. Hệ thống thông báo tạo phiên điểm danh thành công và gửi đến các SV trong lớp học. |
| **Ngoại lệ** | GV điểm danh cho SV không thực hiện điểm danh được trong lớp học. |

*Bảng 3.11. Kịch bản tạo phiên điểm danh*

*Sequence diagram*

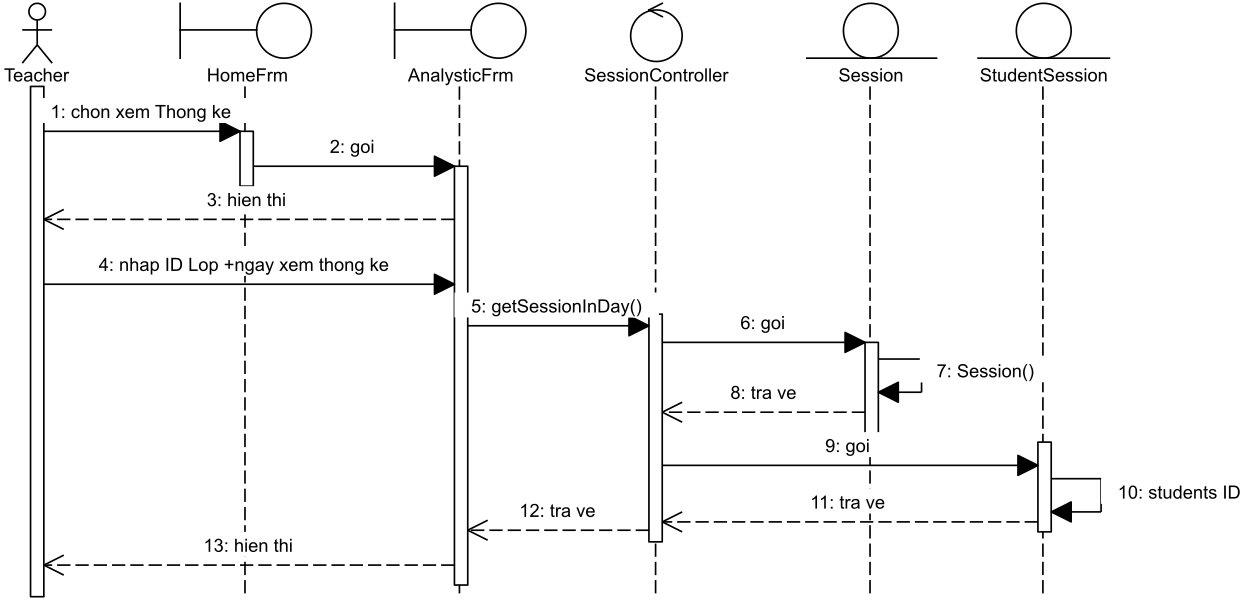
*Hình 3.12. Biểu đồ ca sử dụng tạo phiên điểm danh*

* Thống kê kết quả điểm danh theo ngày học

| **Use Case** | Thống kê kết quả điểm danh theo ngày học |
| --- | --- |
| **Actor** | Giảng viên |

| **Tiền điều kiện** | Đăng nhập thành công vào hệ thống |
| --- | --- |
| **Hậu điều kiện** | GV xem được kết quả điểm danh theo ngày học |
| **Chuỗi sự kiện chính** | 1. GV chọn chức năng Xem thống kê điểm danh theo ngày học trên giao diện của hệ thống. 2. Hệ thống hiển thị giao diện nhập thông tin để xem thống kê. 3. GV nhập mã lớp, ngày tháng năm muốn xem kết quả điểm danh và chọn xem thống kê. 4. Hệ thống hiển thị ngày giờ điểm danh của các sinh viên trong lớp trong ngày đã nhập. |
| **Ngoại lệ** |  |

*Bảng 3.12. Kịch bản ca sử dụng thống kê kết quả điểm danh theo ngày học Sequence diagram*



*Hình 3.13. Biểu đồ ca sử dụng thống kê kết quả điểm danh theo ngày học*

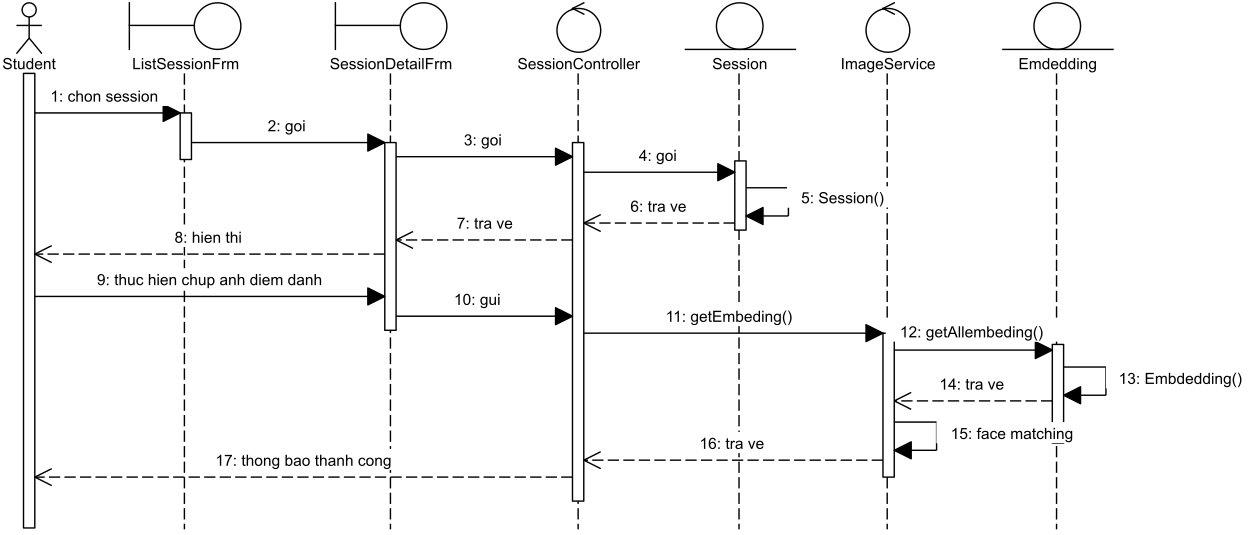
**Các Use Case của Sinh viên**

* Điểm danh

| **Use Case** | Điểm danh |
| --- | --- |
| **Actor** | Sinh viên |
| **Tiền điều kiện** | Đăng nhập thành công vào hệ thống |

| **Hậu điều kiện** | SV thực hiện điểm danh thành công |
| --- | --- |
| **Chuỗi sự kiện chính** | 1. SV click vào thông báo có phiên điểm danh trên giao diện danh sách các lớp học của SV. 2. Hệ thống hiển thị giao diện Camera. 3. SV thực hiện chụp ảnh và gửi lên server. 4. Hệ thống thông báo điểm danh thành công. |
| **Ngoại lệ** | Hệ thống thông báo ảnh giả mạo và SV cần thực hiện chụp lại ảnh. |

*Bảng 3.13. Kịch bản ca sử dụng điểm danh*

*Sequence diagram*

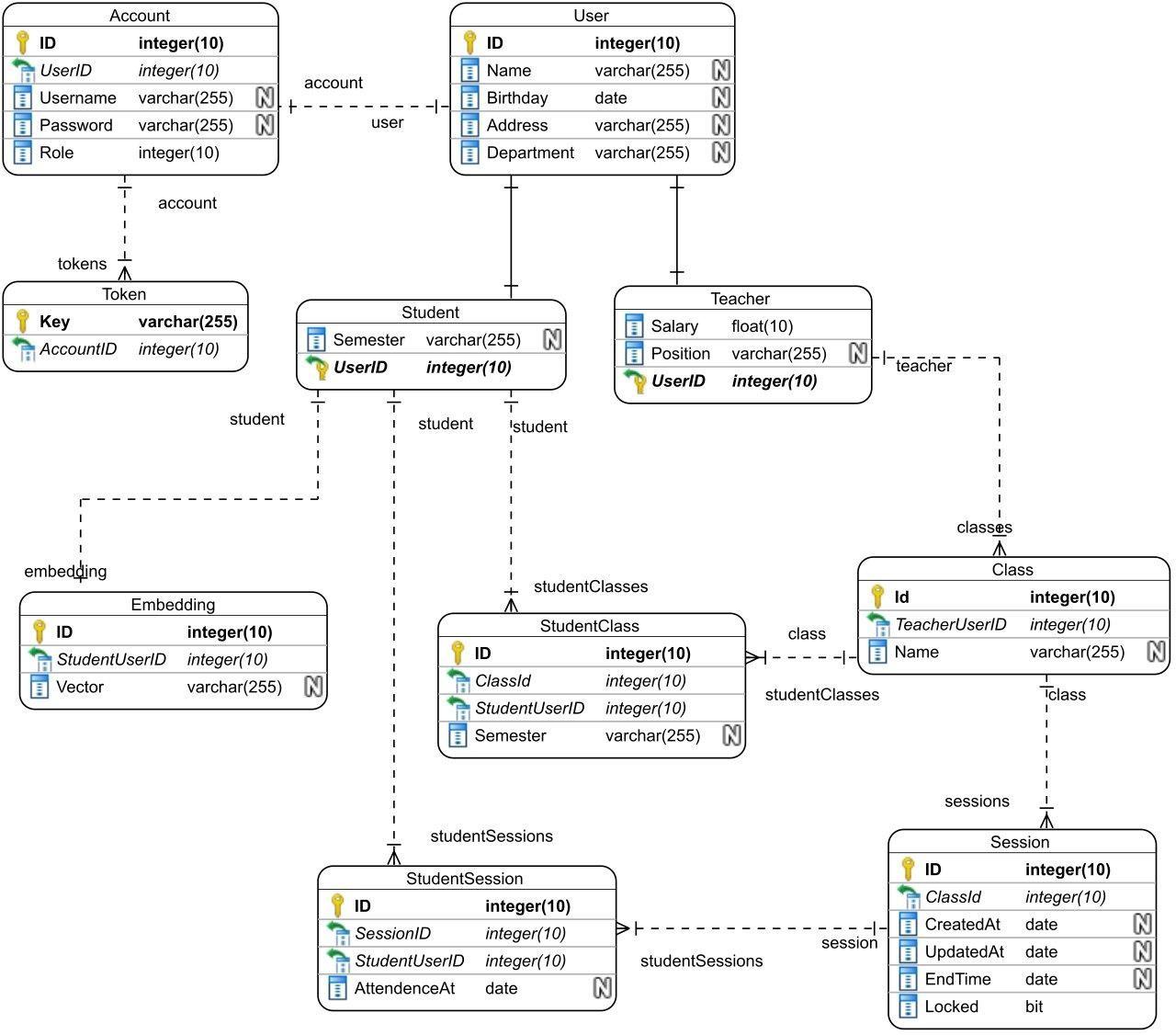
*Hình 3.14. Biểu đồ ca sử dụng điểm danh*

## Thiết kế hệ thống

### Biểu đồ lớp của hệ thống

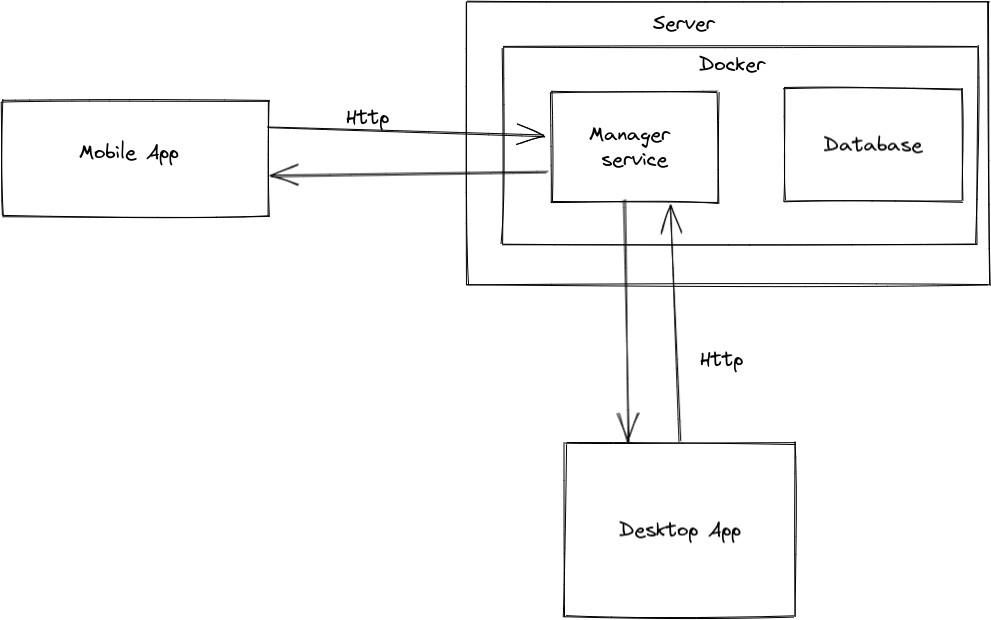
*Hình 3.15. Biểu đồ lớp phân tích của hệ thống*

### Biểu đồ CSDL



*Hình 3.16. Biểu đồ CSDL*

### Thiết kế vật lý của hệ thống



*Hình 3.17. Thiết kế vật lý của hệ thống*

Server

* + *Docker*

Để dễ dàng trong quá trình cài đặt, các dịch vụ của hệ thống được đóng gói lại thành image bằng docker sau đó đẩy lên gitlab container registry.

Docker là một nền tảng cho lập trình viên để develop, deploy và run application với container. Nó cho phép tạo các môi trường độc lập và tách biệt để khởi chạy và phát triển ứng dụng và môi trường này được gọi là container. Khi cần deploy lên bất kỳ server nào chỉ cần run container của Docker thì application sẽ được khởi chạy ngay lập tức.

Lợi ích của việc sử dụng docker

* Khởi chạy nhanh, nhẹ hơn máy ảo.
* Không phụ thuộc nền tảng, không ảnh hưởng đến các phần mềm khác trong hệ thống.
* Dễ dàng thiết lập môi trường làm việc.
  + *Manager service*

Manager service được viết bằng Flask là một web frameworks, nó thuộc loại micro-framework được xây dựng bằng ngôn ngữ lập trình Python. Flask cho phép xây dựng các ứng dụng web từ đơn giản tới phức tạp. Flask

có ưu điểm là nhẹ, ít phụ thuộc vào các cập nhật và dễ tìm kiếm các lỗi bảo mật.

* + *Database*

Sử dụng hệ quản trị cơ sở dữ liệu là MySQL.

**Client:**

Về phía client, hệ thống bao gồm ứng dụng di động chạy trên hệ điều hành Android dành cho giảng viên, sinh viên và ứng dụng trên PC. Ứng dụng đóng vai trò là giao diện để người dùng có thể dễ dàng tương tác với phần server của hệ thống. Ứng dụng Android được viết bằng ngôn ngữ lập trình Java. Ứng dụng Desktop được viết bằng ngôn ngữ PythonQT.

## Tổng quan chức năng điểm danh nằng khuôn mặt

Ứng dụng gồm có 5 phần chính: bộ phát hiện khuôn mặt, bộ căn chỉnh khuôn mặt, bộ chống giả mạo, bộ trích xuất đặc trưng và bộ nhận diện khuôn mặt, cụ thể là:

Bộ phát hiện khuôn mặt:

Sử dụng ảnh RGB làm đầu vào và phát hiện khuôn mặt trong ảnh đó dưới dạng các hộp giới hạn (bounding box) sau đó sẽ làm đầu vào cho các khối chức năng tiếp theo. Có rất nhiều phương pháp phát hiện khuôn mặt, dưới đây là một thống kê độ chính xác (độ đo được sử dụng để đánh giá là AP) phát hiện khuôn mặt của một số mô hình trên tập dữ liệu WIDER FACE (dễ, trung bình, khó):

| **Mô hình** | **WIDER FACE (Dễ)** | **WIDER FACE**  **(Trung bình)** | **WIDER FACE**  **(Khó)** |
| --- | --- | --- | --- |
| **MTCNN** | 85.1% | 82.0 % | 60.7% |
| **CMS-RCNN** | 90.2% | 87.4% | 64.3% |
| **SSH** | 92.7% | 91.5% | 84.4% |
| **PyramidBox** | 95.6% | 94.6% | 90.0% |
| **RetinaFace** | 96.3% | 95.6% | 91.4% |

*Bảng 3.14. Độ chính xác các mô hình phát hiện khuôn mặt trên tập WIDER FACE*

*[8]*

Dựa vào kết quả trên, em sử dụng pre-train model Retina Face và thư viện ncnn-android để thực hiện.

**Bộ căn chỉnh khuôn mặt:**

Mô hình phát hiện khuôn mặt Retina Face cho kết quả là bounding box và 5 điểm trên khuôn mặt: mắt trái, mắt phải, mũi, hai điểm trên môi. Từ đó ta dễ dàng thực hiện căn chỉnh khuôn mặt.

**Bộ chống giả mạo:**

Sử dụng các ảnh chỉ chứa khuôn mặt làm đầu vào, đưa qua mô hình đã được huấn luyện để tiến hành phân loại khuôn mặt đầu vào là thật hay giả mạo. Em xây dựng một mô hình có kiến trúc khai thác điểm mạnh của Depthwise Convolution và các khối residual trong ResNet, train trên Colab và để phát hiện chống giả mạo trên android.

*Chuẩn bị dữ liệu:*

Bộ dữ liệu train gồm 48,134 hình ảnh về khuôn mặt thật và khuôn mặt giả mạo. Các khuôn mặt thật được chụp với các góc mặt khác nhau. Các khuôn mặt giả mạo được tạo ra bằng các màn hình hiển thị khác nhau. Cụ thể:

| Tập dữ liệu | Huấn luyện | Kiểm thử |
| --- | --- | --- |
| Số khuôn mặt thật | 11,025 | 2,945 |
| Số khuôn mặt giả mạo | 21,845 | 12,319 |
| Tổng số khuôn mặt | 32,870 | 15,264 |

*Bảng 3.15. Số lượng hình ảnh khuôn mặt trong tập train*

Bộ dữ liệu test được lấy một phần từ bộ Oulu-NPU và SiW. Bộ SiW được tạo ra từ 165 người khác nhau, có tổng số 1,434,748 khuôn mặt trong tập train, 1,220,369 khuôn mặt trong tập test, ảnh được tạo bởi các góc và cử chỉ khuôn mặt khác nhau. Bộ Oulu-NPU gồm 4950 video về khuôn mặt thật và khuôn mặt giả mạo, có tổng số 241,279 khuôn mặt trong tập train, 179,898 khuôn mặt trong tập test, ảnh được tạo dưới các điều kiện môi trường khác nhau, ảnh giả mạo được thu qua các màn hình và máy in khác nhau. Vì bộ dữ liệu gốc rất lớn nên em chỉ sử dụng một phần làm tập test gọi là Oulu-NPU-mini và SiW-mini, cụ thể:

| Tập dữ liệu (ours) | Oulu-NPU-mini | SiW-mini |
| --- | --- | --- |
| Số khuôn mặt thật | 3,260 | 6,667 |
| Số khuôn mặt giả mạo | 5,840 | 7,742 |
| Tổng số | 9,100 | 14,409 |

*Bảng 3.16. Số lượng hình ảnh khuôn mặt trong tập test Thông số đánh giá:*

Trong bài toán chống giả mạo khuôn mặt, em xác định lớp Positive là khuôn mặt giả mạo, lớp Negative là khuôn mặt thật. Do đó False Positive (FP) là lỗi mô hình dự đoán khuôn mặt giả mạo là khuôn mặt thật và False Negative (FN) là lỗi mô hình dự đoán khuôn mặt thật thành khuôn mặt giả mạo. Từ các định nghĩa trên, em đánh giá mô hình dựa trên các độ đo:

* + APCER: tỉ lệ từ chối sai, có nghĩa là tỉ lệ phân loại nhầm ảnh thật thành ảnh giả, người dùng thật truy cập vào hệ thống nhưng lại bị từ chối. Nếu tỉ lệ này càng lớn sẽ dẫn đến việc người dùng khó đăng nhập vào hệ thống.

APCER = FP / (TN + FP)

* + NPCER: tỉ lệ chấp nhận sai, có nghĩa là tỉ lệ phân loại nhầm ảnh giả thành ảnh thật, người dùng truy cập là giả mạo nhưng hệ thống vẫn chấp nhận. Nếu tỉ lệ này lớn thì hệ thống có khả năng chống giả mạo kém. Đây là một mối nguy hiểm cho các hệ thống bảo mật.

NPCER = FN / (TP + FN)

* + ACER: tỉ lệ lỗi trung bình của hệ thống. Tỉ lệ này giúp ta có cái nhìn tổng quan hơn và đánh giá khách quan hơn về các mức độ từ chối và chấp nhận giữa các phương pháp khác nhau.

ACER = (APCER + NPCER) / 2

*Kết quả thực nghiệm:*

| Bộ dữ liệu đánh giá | APCER (%) | NPCER (%) | ACER (%) |
| --- | --- | --- | --- |
| Oulu-NPU-mini | 2.4 | 0.56 | 1.49 |
| SiW-mini | 0.49 | 0.5 | 0.49 |

*Bảng 3.17. Kết quả đánh giá mô hình chống giả mạo*

**Bộ trích xuất đặc trưng**

Sau khi phân loại được khuôn mặt ở bộ chống giả mạo, khuôn mặt thật sẽ được đưa vào bộ trích xuất đặc trưng. Dựa trên tiêu chí cân bằng giữa tốc độ và độ chính xác, em lựa chọn mô hình mạng LResNet100E-IR được xây dựng bởi Jiankang Deng làm mô hình trích chọn đặc trưng cho khuôn mặt. Kiến trúc mạng này có backbone là mạng ResNet100 và sử dụng làm loss ArcFace để tối ưu kết quả. Đầu ra của mạng là một vector đại diện cho khuôn mặt có số chiều là 512.

**Bộ nhận diện khuôn mặt**

Với mỗi hình ảnh khuôn mặt đầu vào sẽ được trích xuất thành một vector đặc trưng, sau đó sử dụng độ đo khoảng cách Euclidean để phân khuôn mặt đầu vào là một trong các khuôn mặt nằm trong CSDL khi vượt qua một giá trị ngưỡng cho phép.

# CHƯƠNG 4: THỬ NGHIỆM CÀI ĐẶT VÀ XÂY DỰNG ỨNG DỤNG

## Cài đặt Server

Hệ thống được thuê trên dịch vụ AWS (Amazon Web Services) với cấu hình như sau.

| CPU | Ram | Bộ nhớ |
| --- | --- | --- |
| 2 Core | 4GB | 32GB SSD |

Phiển bản phầm mềm được sử dụng.

* Python 3.9.9 bulleye
* Mysql 5.7
* Docker 19

Container của chứa backend service và data sử dụng chung 1 network ảo và được export 1 port là 8092 để giúp cho client có thể giao tiếp với backend.

## Cài đặt Client

*Android Application*

Em sử dụng trình biên dịch Android Studio Artic Fox để lập trình app Android Cấu hình thiết bị kiểm thử được mô tả ở các bảng dưới:

| Android verion | API version | SoC | RAM |
| --- | --- | --- | --- |
| 10.0 | API 29 | Snapdragon 660 | 4GB |

*Desktop Application*

Được viết trên QT framework với ngôn ngữ lập trình là Python.

QT là được sử dụng để tạo các chương trình GUI không phụ thuộc nền tảng, các phần mềm được tạo ra bởi QT chạy được trên hầu hết các hệ điều hành PC hiện tại. Các chương trình được tạo bởi QT bao gồm các thành phần như QTCore, QWidget, QtUiTools

… QT cung cấp rất nhiều công cụ mạnh mẽ để tạo các chương trình chạy trên desktop. Môi trường phát triển QT là QT Creator.

## Thiết kế giao diện

### Giao diện đăng nhậpGraphical user interface, application Description automatically generatedGraphical user interface, application Description automatically generated

*Hình 4.1. Giao diện đăng nhập*

Người dùng ở đây là Sinh viên hoặc Giảng viên sẽ nhập mã và mật khẩu rồi chọn ĐĂNG NHẬP trên giao diện.

### Giao diện danh sách lớp họcGraphical user interface, application Description automatically generatedGraphical user interface, application Description automatically generated

*Hình 4.2. Giao diện danh sách lớp học*

Sinh viên xem danh sách các lớp học đang cần được điểm danh và chọn ĐIỂM DANH trên giao diện (hình bên trái).

Giảng viên xem danh sách các lớp học của mình và chọn một lớp để tạo phiên điểm danh (hình bên phải).

### Giao diện điểm danhGraphical user interface, application Description automatically generatedGraphical user interface, website Description automatically generated

*Hình 4.3. Giao diện điểm danh*

Sinh viên thực hiện chụp ảnh điểm danh, khi gửi lên hình ảnh giả mạo (hình bên trái) sẽ không được chấp nhận, chỉ khi chụp hình người thật mới được chấp nhận (hình bên phải).

### Giao diện kết quả điểm danhGraphical user interface Description automatically generated

*Hình 4.4. Giao diện kết quả điểm danh*

Giao diện kết quả điểm danh của Sinh viên ở một lớp, những Sinh viên chưa điểm danh được có thể được Giảng viên chọn điểm danh cho.

### Giao diện đăng nhập của Admin

*Hình 4.5. Giao diện đăng nhập của Admin*

Admin nhập tên đăng nhập và mật khẩu trên và chọn đăng nhập trên giao diện.

### Giao diện thêm mới tài khoản

*Hình 4.6. Giao diện thêm mới tài khoản*

Admin chọn chức năng tạo tài khoản và nhập thông tin cho Sinh viên, Giảng viên.

### Giao diện thêm ảnh vào tài khoản sinh viên

*Hình 4.7. Giao diện thêm ảnh vào tài khoản sinh viên*

Admin thêm ảnh của từng sinh viên vào tài khoản sinh viên để sau này hệ thống kiểm tra và điểm danh cho sinh viên bằng hình ảnh.

### Giao diện sửa tài khoản

*Hình 4.8. Giao diện sửa tài khoản*

Admin chọn sửa tài khoản và thay đổi các thông tin của người dùng.

### Giao diện xóa tài khoản

*Hình 4.9. Giao diện xóa tài khoản*

Admin chọn chức năng xóa tài khoản người dùng trên giao diện.

# KẾT LUẬN

Qua thời gian tìm hiểu, học hỏi và xây dựng đề tài **“Phát triển hệ thống điểm danh trong lớp học sử dụng công nghệ nhận diện khuôn mặt”,** em đã đạt được một số kết quả nhất định:

## Kết quả đạt được

* + Tìm hiểu các cách tiếp cận và phương pháp để giải quyết bài toán.
  + Ứng dụng giải pháp sử dụng mạng nơ-ron tích chập cho bài toán.
  + Nâng cao kỹ năng phân tích, thiết kế hệ thống.
  + Trau dồi thêm kỹ năng lập trình.

## Hướng phát triển

Hiện tại hệ thống chưa đạt kết quả tốt và chưa xử lý hết các trường hợp xảy đến trong quá trình điểm danh. Vì vậy, trong tương lai, em sẽ thay đổi các model để kết quả chính xác hơn, tốc độ xử lý nhanh hơn và có thể triển khai cho thực tế.

**Tiếng Việt**

# TÀI LIỆU THAM KHẢO

1. Nguyễn Thanh Tuấn - Deep Learning Cơ Bản. Copyright 2019.

## Tiếng Anh

1. Shervin Minaee, Ping Luo, Zhe Lin, Kevin Bowyer, “Going Deeper into Face Detection: A Survey”.
2. Meng Yuana, Seyed Yahya Nikoueia, Alem Fitwia, Yu Chena, Yunxi Dong, “Minor Privacy Protection Through Real-time Video Processing at the Edge”.
3. Jiankang Deng, Jia Guo, Niannan Xue, “ArcFace: Additive Angular Margin Loss for Deep Face Recognition”.
4. Zitong Yu, Chenxu Zhao, Zezheng Wang, Yunxiao Qin, Zhuo Su, Xiaobai Li, Feng Zhou, Guoying Zhao, “Searching Central Difference Convolutional Networks for Face Anti-Spoofing”.
5. Peng Zhang, Fuhao Zou1, Zhiwen Wu, Nengli DaiSkarpness Mark, Michael Fu, Juan Zhao, Kai Li, “FeatherNets: Convolutional Neural Networks as Light as Feather for Face Anti-spoofing”.
6. Mei Wang, Weihong Deng, “Deep Face Recognition: A Survey”.
7. Jialiang Zhang, Xiongwei Wu, Jianke Zhu, Steven C.H. Hoi, “Feature Agglomeration Networks for Single Stage Face Detection”.

## Danh mục các website tham khảo

1. “<https://dominhhai.github.io/vi/2018/04/nn-intro/>”.
2. “https://thanhvie.com/tim-hieu-ve-mang-no-ron-tich-chap-convolutional-neural-networks/”.
3. “<https://phamdinhkhanh.github.io/2020/05/31/CNNHistory.html>”.
4. “<https://paperswithcode.com/method/inception-v4>”.
5. “<https://viblo.asia/p/detect-object-su-dung-mo-hinh-ssd-WAyK84rnKxX>”.
6. “[https://ichi.pro/vi/giai-thich-ve-siamese-net-triplet-loss-va-circle-loss-](https://ichi.pro/vi/giai-thich-ve-siamese-net-triplet-loss-va-circle-loss-208539308200695) [208539308200695](https://ichi.pro/vi/giai-thich-ve-siamese-net-triplet-loss-va-circle-loss-208539308200695)”.
7. “[https://viblo.asia/p/nhan-dien-khuon-mat-voi-mang-mtcnn-va-facenet-](https://viblo.asia/p/nhan-dien-khuon-mat-voi-mang-mtcnn-va-facenet-phan-1-Qbq5QDN4lD8) [phan-1-Qbq5QDN4lD8](https://viblo.asia/p/nhan-dien-khuon-mat-voi-mang-mtcnn-va-facenet-phan-1-Qbq5QDN4lD8)”.
8. “<https://viblo.asia/p/distance-measure-trong-machine-learning-ByEZkopYZQ0>”.