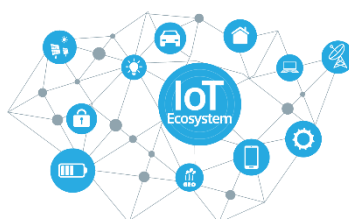




**TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN**



BÁO CÁO ĐỒ ÁN PBL5 - KỸ THUẬT MÁY TÍNH



HỆ THỐNG ĐIỂM DANH LỚP HỌC BẰNG NHẬN DIỆN KHUÔN MẶT

**CÁN BỘ DOANH NGHIỆP HƯỚNG DẪN: Trần Phương Nam
GIẢNG VIÊN ĐỒNG HƯỚNG DẪN: TS. Ninh Khánh Duy**

| STT NHÓM: 01 HỌ VÀ TÊN SINH VIÊN | LỚP HỌC PHẦN ĐỒ ÁN |
|---|-------------------------------|
| Huỳnh Văn Quân | 18N13A |
| Trịnh Xuân Phúc | 18N13A |
| Lê Trường Sanh | 18N13A |
| Phan Anh Tuấn | 18N13A |

ĐÀ NẴNG, 06/2021

TÓM TẮT ĐỒ ÁN

Trong quá trình học tập và tìm hiểu, chúng em nhận thấy việc điểm danh lớp học ở trường chưa thật sự hiệu quả và tối ưu. Do đó, chúng em đề xuất và tiến hành xây dựng đồ án với đề tài “Hệ thống điểm danh lớp học bằng nhận diện khuôn mặt” với mong muốn giúp việc điểm danh diễn ra tự động, không mất nhiều nguồn lực. Nhóm đã ứng dụng các công nghệ AI trong việc điểm danh, API trong việc giao tiếp giữa ứng dụng và server, Raspberry Pi 3 và webcam để chụp ảnh, xây dựng ứng dụng điện thoại để hiển thị kết quả quá trình và sử dụng điện toán đám mây để thiết lập server. Sau khi nghiên cứu và thử nghiệm, hệ thống đã hoạt động tốt. Tuy nhiên, còn một số điểm thiếu sót nên hệ thống vẫn chưa hoàn hảo. Nhóm sẽ tiếp tục phát triển và hoàn thiện trong tương lai.

BẢNG PHÂN CÔNG NHIỆM VỤ

| Sinh viên | Nhiệm vụ | Hoàn thành |
|------------------------|---|-------------------|
| Huỳnh Văn Quân | Phân công công việc, đảm bảo tiến độ đồ án | ✓ |
| | Tìm hiểu, triển khai phần phát hiện khuôn mặt | ✓ |
| | Tìm kiếm và chuẩn bị phần cứng | ✓ |
| | Hỗ trợ xây dựng cơ sở dữ liệu | ✓ |
| | Hỗ trợ xây dựng server | ✓ |
| | Ghép nối và thử nghiệm sản phẩm | ✓ |
| | Viết báo cáo | ✓ |
| | Làm slide | ✓ |
| Trịnh Xuân Phúc | Tìm hiểu, triển khai phần nhận diện khuôn mặt | ✓ |
| | Hỗ trợ xây dựng cơ sở dữ liệu | ✓ |
| | Hỗ trợ xây dựng server | ✓ |
| | Ghép nối và thử nghiệm sản phẩm | ✓ |
| | Viết báo cáo | ✓ |
| Lê Trường Sanh | Xây dựng ứng dụng mobile app | ✓ |
| | Xây dựng cơ sở dữ liệu | ✓ |
| | Hỗ trợ xây dựng server | ✓ |
| | Ghép nối và thử nghiệm sản phẩm | ✓ |
| | Viết báo cáo | ✓ |
| Phan Anh Tuấn | Thiết lập và triển khai server | ✓ |
| | Viết API | ✓ |
| | Tìm kiếm và chuẩn bị phần cứng | ✓ |
| | Hỗ trợ xây dựng cơ sở dữ liệu | ✓ |
| | Ghép nối và thử nghiệm sản phẩm | ✓ |
| | Viết báo cáo | ✓ |

MỤC LỤC

| | |
|--|----|
| 1. Giới thiệu | 1 |
| 1.1. Thực trạng sản phẩm | 1 |
| 1.2. Các vấn đề cần giải quyết | 1 |
| 1.3. Đề xuất giải pháp tổng quan..... | 1 |
| 2. Giải pháp | 2 |
| 2.1. Giải pháp về phần cứng | 2 |
| 2.1.1. Sơ đồ tổng quan hệ thống | 2 |
| 2.1.2. Sơ đồ hoạt động tổng quan | 2 |
| 2.1.3. Linh kiện sử dụng | 3 |
| 2.2. Truyền thông..... | 3 |
| 2.2.1. Restful API | 3 |
| 2.2.2. Giới thiệu về Django..... | 4 |
| 2.2.3. Kiến trúc Django | 5 |
| 2.3. Giải pháp phát hiện khuôn mặt | 5 |
| 2.3.1. Giới thiệu về YOLOv4 | 5 |
| 2.3.2. Kiến trúc YOLOv4 | 5 |
| 2.3.3. Xương sống (Backbone) | 6 |
| 2.3.4. Cổ (Neck) | 8 |
| 2.3.5. Head | 9 |
| 2.3.6. Hàm kích hoạt Mish | 9 |
| 2.3.7. Batch Normalization | 9 |
| 2.4. Giải pháp nhận diện khuôn mặt | 9 |
| 2.4.1. Công thức sử dụng (MobileFaceNet + Arcface) | 9 |
| 2.4.2. Mobile Facenet..... | 10 |
| 2.4.3. Loss Function ArcFace..... | 11 |
| 2.5. Giải pháp ứng dụng di động..... | 13 |
| 2.5.1. Phát triển bài toán | 13 |
| 2.5.2. Công nghệ sử dụng | 13 |
| 2.5.3. Biểu đồ usecase hệ thống..... | 13 |
| 3. Kết quả..... | 15 |
| 3.1. Phát hiện khuôn mặt | 15 |
| 3.1.1. Tập dữ liệu | 15 |
| 3.1.2. Huấn luyện | 15 |
| 3.1.3. Kết quả giải pháp phát hiện khuôn mặt..... | 16 |
| 3.2. Giải pháp nhận diện khuôn mặt | 19 |
| 3.2.1. Dữ liệu..... | 19 |
| 3.2.2. Kết quả giải pháp nhận diện khuôn mặt..... | 20 |
| 3.3. Server | 22 |
| 3.3.1. API | 22 |
| 3.3.2. Tốc độ thực thi hệ thống | 24 |
| 4. Ứng dụng di động | 25 |

Báo cáo đồ án PBL5 - Kỹ thuật máy tính

| | |
|---|-----------|
| 5. Kết luận | 25 |
| 5.1. Đánh giá..... | 25 |
| 5.2. Hướng phát triển | 26 |
| 6. Danh mục tài liệu tham khảo | 26 |

1. Giới thiệu**1.1. Thực trạng sản phẩm**

Hiện nay, trên thế giới đã xuất hiện nhiều các sản phẩm về điểm danh nhận diện khuôn mặt. Trong đó, có nhiều loại với các nghiệp vụ khác nhau như đặt camera trên cao, đặt camera trước cửa, điểm danh online, ... Nhiều sản phẩm có hình ảnh thu được rất rõ nét và khả năng nhận diện đã đạt đến độ chính xác cao. Tuy nhiên, chi phí cho một hệ thống như vậy là cực kỳ lớn. Do đó, chúng em tiến hành thử nghiệm với đề tài này nhằm tìm ra giải pháp tốt với chi phí thấp hơn.

1.2. Các vấn đề cần giải quyết

- Cần có các thiết bị phần cứng để thu nhận dữ liệu.
- Phát hiện và nhận diện nhiều khuôn mặt trong lớp học, tiến hành điểm danh.
- Cần ứng dụng tương tác để giảng viên có thể điểm danh, kiểm tra kết quả.
- Hệ thống chạy theo thời gian thực.

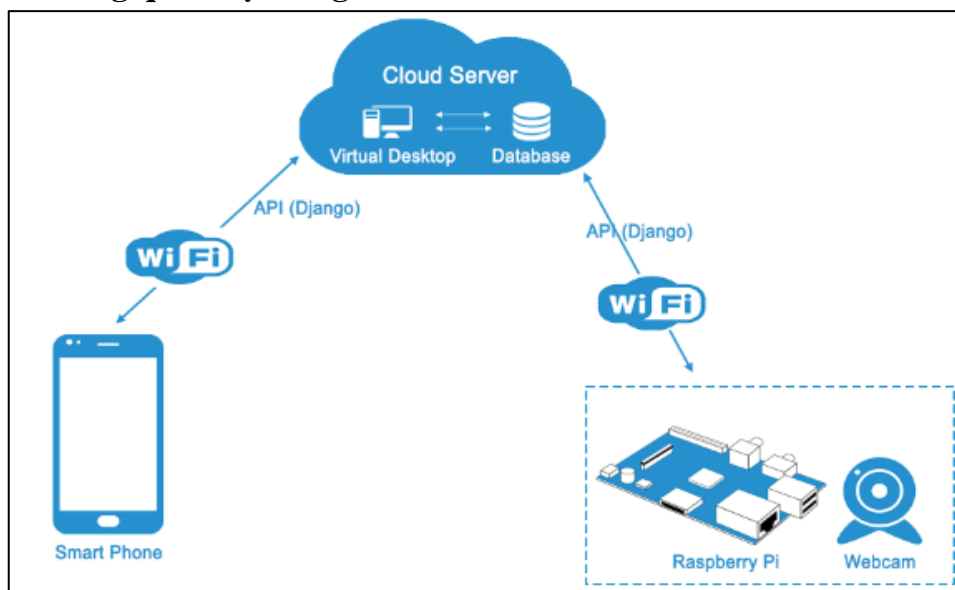
1.3. Đề xuất giải pháp tổng quan**Bảng 1: Đề xuất giải pháp tổng quan**

| Vấn đề | Giải pháp đề xuất |
|---------------------|---|
| Phần cứng | Raspberry Pi 3. Webcam Logitech C270. Điện thoại thông minh. Máy chủ ảo |
| Phát hiện khuôn mặt | Xây dựng và huấn luyện model phát hiện khuôn mặt. Thử nghiệm với các model: Yolo, Facenet,... Huấn luyện trên Google Colab. |
| Nhận diện khuôn mặt | Xây dựng và huấn luyện model nhận diện khuôn mặt. Thử nghiệm với các model: Facenet, ArcFace,... Huấn luyện trên Google Colab. |
| Ứng dụng | Xây dựng ứng dụng điện thoại. Giảng viên có thể đăng nhập. Có chức năng điểm danh tự động, điểm danh thủ công. Hiển thị kết quả điểm danh. |
| Server | Viết API bằng Django |

2. Giải pháp

2.1. Giải pháp về phần cứng

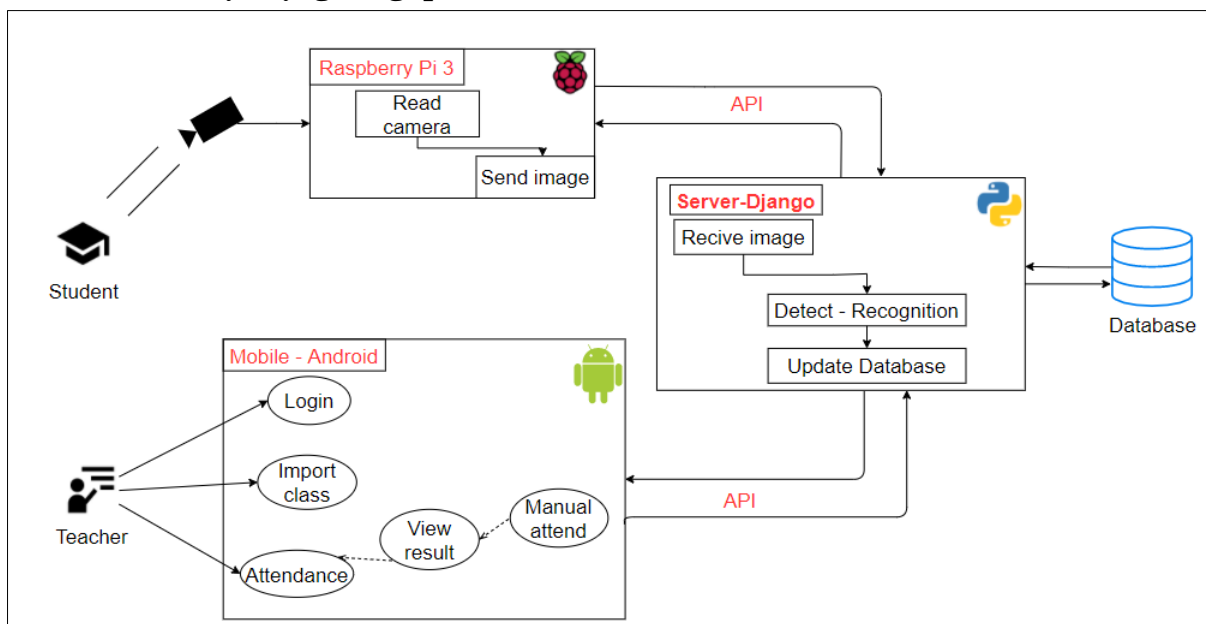
2.1.1. Sơ đồ tổng quan hệ thống



Hình 1: Sơ đồ tổng quan hệ thống



Hệ thống bao gồm Raspberry Pi 3 và webcam dùng để chụp ảnh, thiết bị smart phone dùng tương tác và hiển thị kết quả và Cloud Server để thiết lập Server. Thông qua mạng không dây, điện thoại và Raspberry Pi có thể giao tiếp với Server bằng API. API này được lập trình dựa trên Django Framework.

2.1.2. Sơ đồ hoạt động tổng quan



Hình 2: Sơ đồ hoạt động tổng quan

2.1.3. Linh kiện sử dụng**Bảng 2: Linh kiện sử dụng**

| Tên linh kiện | Hình ảnh | Thông số, hoạt động |
|-----------------------------|--|--|
| Raspberry Pi 3 |  | Thông số kỹ thuật Bộ vi xử lý 64-bit quad-core ARM Cortex-A53 Mạng không dây 802.11 b/g/n Bluetooth 4.1 Bộ nhớ LPDDR2 1GB Nguồn 2.5A cổng MicroUSB 1 cổng Ethernet 10/100 1 cổng HDMI 1 jack cắm RCA 4 cổng USB 2.0 Khe cắm thẻ nhớ MicroSD |
| Webcam Logitech C270 |  | Thông số kỹ thuật Chất lượng video: 1280x720 pixel Độ phân giải hình ảnh: 0.3 – 3.0MP Micro tích hợp, giảm tiếng ồn Cổng USB 2.0 |

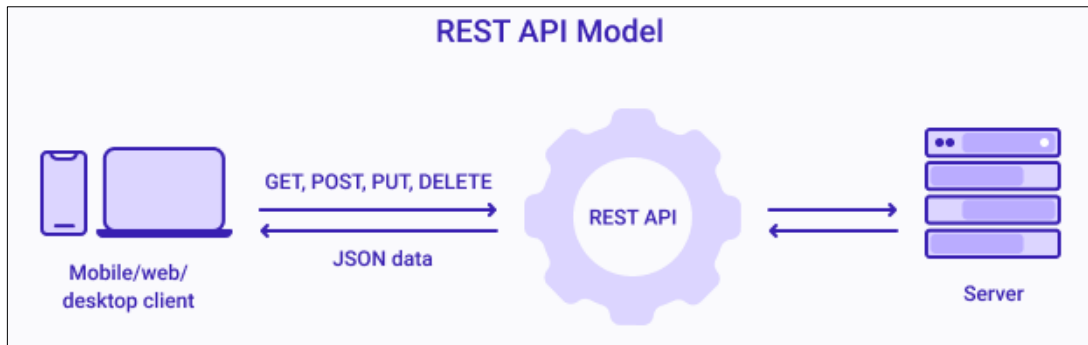
Bảng 3: Bảng kê chi phí đồ án

| Tên linh kiện | Đơn giá | Ghi chú |
|----------------------------------|--------------------------------------|----------------|
| 1 x Raspberry Pi 3 | 930.000 | Mượn |
| 1 x Webcam Logitech C270 | 640.000 | Mượn |
| 1 x Cloud Server | 329.000 | Thuê |
| 1 x Thẻ nhớ 64GB | 80.000 | Mua |
| 1 x Vỏ hộp Raspberry Pi 3 | 120.000 | Mua |
| | Thành tiền: 529.000 | |

2.2. Truyền thông**2.2.1. Restful API**

RESTful API là một tiêu chuẩn được sử dụng trong việc thiết kế API cho các phần mềm, ứng dụng và dịch vụ web để tạo sự thuận tiện cho việc quản lý các resource. Các tài nguyên hệ thống như tệp văn bản, ảnh, video, âm thanh hay dữ liệu di động là mục tiêu mà nó hướng tới, bao gồm các trạng thái tài nguyên được định dạng và truyền tải qua HTTP. Có thể nói, RESTful API không phải là một loại công nghệ. Nó chỉ là một phương thức tạo ra API và nguyên lý tổ chức nhất định.

❖ Các thành phần của RESTful API



Hình 3: Mô hình RESTful API

API (Application Programming Interface): Là tập hợp các quy tắc và cơ chế mà một ứng dụng hay một thành phần nào đó có khả năng tương tác với một ứng dụng với thành phần khác. API sẽ trả về những kiểu dữ liệu phổ biến như JSON hoặc XML mà ứng dụng của bạn cần sử dụng đến.

REST (Representational State Transfer): Là một dạng chuyển đổi cấu trúc hay kiểu kiến trúc để viết API. Nó có khả năng tạo ra sự tương tác giữa các máy với nhau thông qua phương thức HTTP đơn giản. Chức năng của REST là quy định sử dụng các phương thức HTTP và định dạng URL cho ứng dụng web.

❖ Cách thức hoạt động của RESTful API

RESTful API là phương thức tạo ra API và hoạt động dựa trên phương thức HTTP:

- GET (SELECT): Trả về một Resource hoặc một danh sách Resource.
- POST (CREATE): Tạo mới một Resource.
- PUT (UPDATE): Cập nhật thông tin cho Resource.
- DELETE (DELETE): Xoá một Resource.

2.2.2. Giới thiệu về Django

Django là một web framework khá nổi tiếng được viết hoàn toàn bằng ngôn ngữ Python. Nó là một framework với đầy đủ các thư viện, module hỗ trợ các web-developer. Mục tiêu chính của Django là đơn giản hóa việc tạo các website phức tạp có sử dụng cơ sở dữ liệu. Django tập trung vào tính năng “có thể tái sử dụng” và “có thể tự chạy”, tính năng phát triển nhanh, không làm lại những gì đã làm. Một số website phổ biến được xây dựng từ Django là Pinterest, Instagram, Mozilla, và Bitbucket.

Django-Rest-Framework là một framework được cài vào Django, có đầy đủ chức năng, đủ sức mạnh để tạo ra 1 hệ thống dịch vụ API mạnh mẽ nhằm làm cầu nối cho các hệ thống khác nhau như giữa các client với server. Client ở đây là web, mobile, tablet,

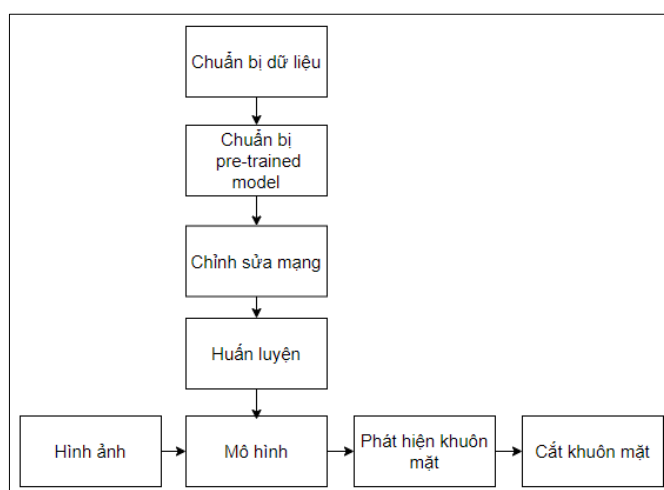
... server ở đây là Django, database MySQL,... Django-Rest-Framework hỗ trợ giao thức truyền tải dữ liệu HTTP thông qua các phương thức như Post, Get, Put, Delete.

2.2.3. Kiến trúc Django

Django sử dụng mô hình **MVT** (Model-View-Template) thay vì sử dụng mô hình **MVC** (Model-View-Controller). Mô hình MVT được sử dụng trong khi tạo một ứng dụng với Tương tác người dùng.

Mô hình này bao gồm code HTML với Django Template Language (DTL). Controller là mã được viết để kiểm soát sự tương tác giữa Model và View và Django để dàng xử lý nó. Bất cứ khi nào người dùng gửi request, nó xử lý request của người dùng đó bằng Model, View và Template. Nó hoạt động như một Controller để kiểm tra xem nó có khả dụng hay không bằng cách ánh xạ URL và nếu URL ánh xạ thành công thì View sẽ bắt đầu tương tác với Model và gửi lại Template cho người dùng dưới dạng response.

2.3. Giải pháp phát hiện khuôn mặt



Hình 4: Quá trình thực hiện giải pháp

2.3.1. Giới thiệu về YOLOv4

Trong phần phát hiện khuôn mặt, nhóm đề xuất sử dụng YOLOv4. YOLOv4 là một mô hình sử dụng cho việc phát hiện, nhận dạng, phân loại đối tượng. YOLOv4 được xây dựng dựa trên mạng nơon tích chập (CNN). So với các phiên bản Yolo trước, YOLOv4 áp dụng một số thuật toán phát hiện vật thể nhanh, tối ưu hóa các phép toán thực hiện song song giúp tăng tốc độ nhận diện và tăng độ chính xác.

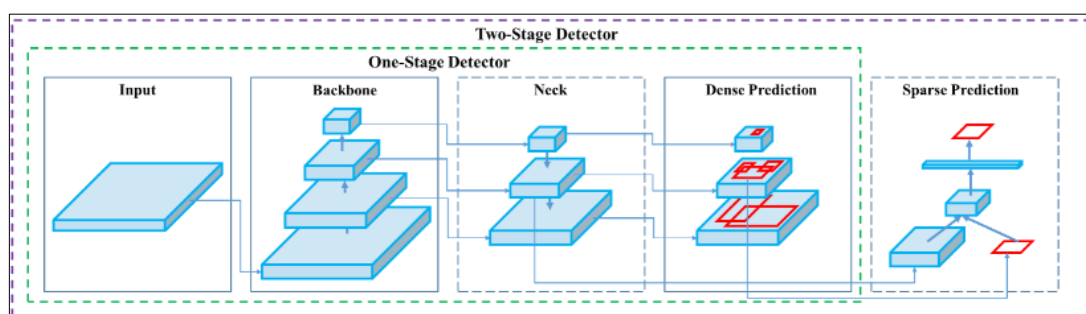
2.3.2. Kiến trúc YOLOv4

Cấu trúc nhận diện vật thể của YOLOv4 có 3 phần:

Backbone: Ở phần Backbone, YOLOv4 sử dụng một mô hình pre-trained của một mô hình học chuyển (transfer learning) để trích rút các đặc trưng. Các mô hình học chuyển thường là VGG16, Resnet-50,... Mô hình học chuyển được áp dụng trong YOLOv4 là CSPDarknet53.

Neck: Ở phần Neck, YOLOv4 sử dụng Spatial Pyramid Pooling (SPP) để đưa các bản đồ đặc trưng về cùng kích thước. Đồng thời, sử dụng Path Aggregation Network (PAN) để thực hiện phân đoạn.

Head: Ở phần Head, YOLOv4 sử dụng thuật toán YOLO để thực dự đoán bounding box và class.



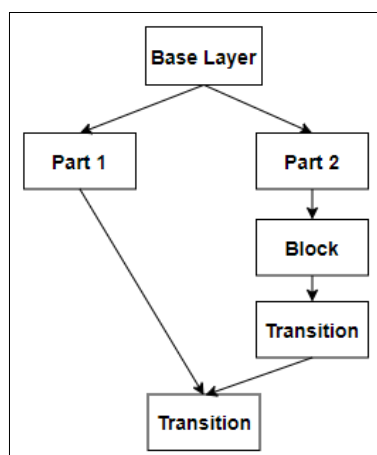
Hình 5: Kiến trúc YOLOv4

2.3.3. Xương sống (Backbone)

Backbone cho một bộ phát hiện đối tượng là bộ phân lớp đã được huấn luyện trước trên tập dữ liệu ImageNet nhằm trích rút đặc trưng. Trong Yolov4, phần Backbone sử dụng mạng CSPDarknet53. CSPDarknet53 là mạng nơ-ron tích chập và là xương sống cho phần phát hiện vật thể. Mạng này được xây dựng dựa trên CSPNet và Darknet53.

❖ CSPNet (Cross Stage Partial Network)

Khi mạng CSP kết hợp với Darknet53, khối block của CSP sẽ được thay bằng Darknet53, tạo thành một mạng mới, giúp tăng cường khả năng học và xử lý của Darknet53 hơn.



Hình 6: Kiến trúc CSPNet

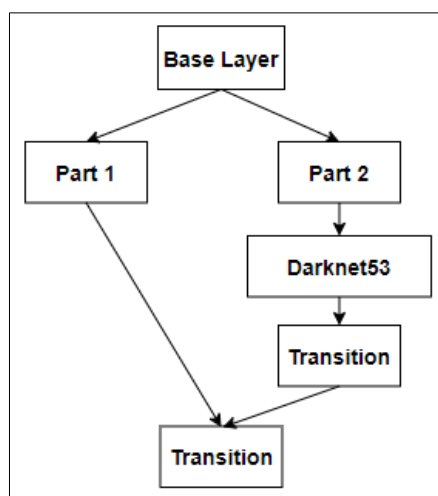
❖ Darknet53

Darknet53 là mạng sử dụng trong trích rút đặc trưng, xây dựng dựa trên mạng CNN. Mạng này được áp dụng trong Yolov3, đáp ứng mục đích phát hiện và nhận diện vật thể. Kiến trúc CSPDarknet53 sử dụng tổng cộng 52 lớp tích chập, 23 lớp residual.

| | Type | Filters | Size | Output |
|----|---------------|---------|------------------|------------------|
| | Convolutional | 32 | 3×3 | 256×256 |
| | Convolutional | 64 | $3 \times 3 / 2$ | 128×128 |
| 1x | Convolutional | 32 | 1×1 | |
| | Convolutional | 64 | 3×3 | |
| | Residual | | | 128×128 |
| | Convolutional | 128 | $3 \times 3 / 2$ | 64×64 |
| 2x | Convolutional | 64 | 1×1 | |
| | Convolutional | 128 | 3×3 | |
| | Residual | | | 64×64 |
| | Convolutional | 256 | $3 \times 3 / 2$ | 32×32 |
| 8x | Convolutional | 128 | 1×1 | |
| | Convolutional | 256 | 3×3 | |
| | Residual | | | 32×32 |
| | Convolutional | 512 | $3 \times 3 / 2$ | 16×16 |
| 8x | Convolutional | 256 | 1×1 | |
| | Convolutional | 512 | 3×3 | |
| | Residual | | | 16×16 |
| | Convolutional | 1024 | $3 \times 3 / 2$ | 8×8 |
| 4x | Convolutional | 512 | 1×1 | |
| | Convolutional | 1024 | 3×3 | |
| | Residual | | | 8×8 |
| | Avgpool | | Global | |
| | Connected | | 1000 | |
| | Softmax | | | |

Hình 7: Kiến trúc Darknet53**❖ CSPDarknet53**

Khi mạng CSP kết hợp với Darknet53, khối block của CSP sẽ được thay bằng Darknet53, tạo thành một mạng mới, giúp tăng cường khả năng học và xử lý của Darknet53 hơn.

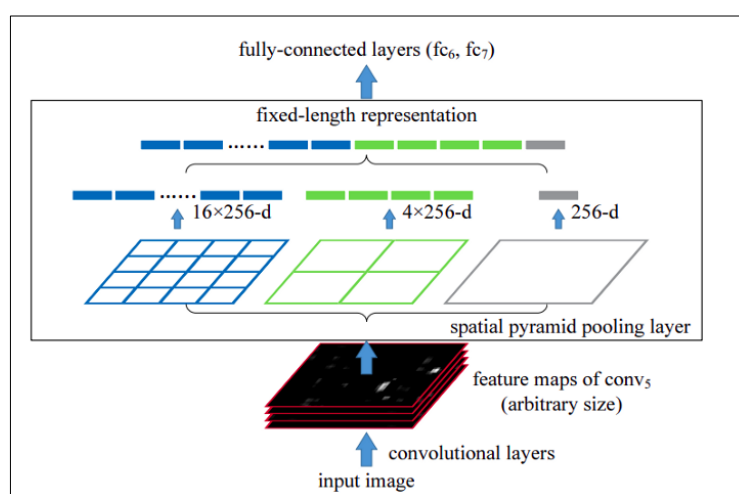
**Hình 8: Kiến trúc CSPDarknet53**

2.3.4. Cổ (Neck)

Trong phần neck, YOLOv4 sử dụng khối SPP và PAN.

❖ Spatial Pyramid Pooling

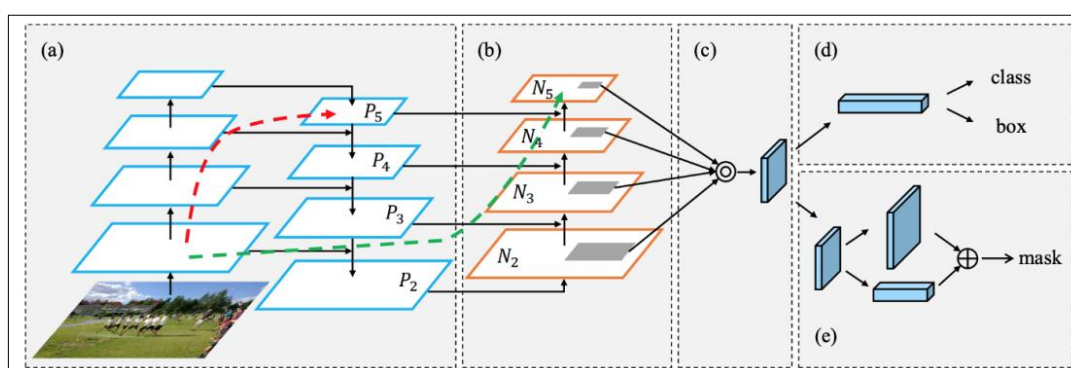
Trong YOLOv4, SPP giúp đưa kích thước các bản đồ đặc trưng về cùng một kích thước cố định. Một lớp tích chập đơn hoặc một tập hợp lớp tích chập lấy một hình ảnh và tạo ra một bản đồ đặc trưng tỉ lệ với một tỉ lệ cụ thể với hình ảnh đầu vào. Nhưng đối với lớp kết nối đầy đủ, đầu vào phải là một vector có chiều dài cố định. Do đó, trong YOLOv4 phải sử dụng SPP để đưa kích thước các bản đồ đặc trưng về cùng một kích thước cố định. Phép gộp mà SPP sử dụng trong YOLOv4 là phép gộp giá trị lớn nhất (Max Pooling).



Hình 9: Nguyên lý hoạt động Spatial Pyramid Pooling

❖ Path Aggregation Network (PAN)

Mạng học sâu càng sâu thì càng làm mất mát thông tin do khi hình ảnh đi qua các lớp của mạng nơ-ron, độ phức tạp của đặc trưng tăng, độ phân giải hình ảnh giảm. Vì vậy, sẽ rất khó để phát hiện được vật thể kích thước nhỏ. Trong YOLOv4, PANet được dùng để phân đoạn vì nó có thể lưu chính xác thông tin không gian, giúp định vị trí chính xác các pixel và tạo thành mặt nạ.



Hình 10: Nguyên lý hoạt động của PAN

2.3.5. Head

Ở phần Head, để thực hiện dự đoán bounding box và class, YOLOv4 được thực hiện giống YOLOv3. Dự đoán cho bounding box: Sử dụng hồi quy logistic để dự đoán. Kết quả dự đoán bằng 1 nếu bounding box có sự trùng lặp so với anchor box tốt nhất trong các bounding box. Phân lớp: Để tăng kết quả dự đoán với bài toán phân loại đa nhãn (multi-label classification), tác giả sử dụng phân lớp logistic độc lập (independent logistic classifier) thay cho hàm softmax.

2.3.6. Hàm kích hoạt Mish

Trong YOLOv4 ta sử dụng các hàm kích hoạt Mish. Mish là một hàm kích hoạt trơn, liên tục, tự điều chỉnh và không đơn điệu, được định nghĩa với công thức:

$$f(x) = x \tanh(\text{softplus}(x)) = x \tanh(\ln(1 + e^x))$$

Hàm Mish sử dụng trong YOLOv4 vì chi phí của nó thấp, nhiều tính chất khác nhau như trơn tru và không đơn điệu, không có chặn trên, có chặn dưới, hội tụ nhanh chóng giúp cải thiện hiệu suất của mô hình hơn so với việc sử dụng hàm ReLU.

2.3.7. Batch Normalization

Batch Normalization là một phương pháp hiệu quả khi huấn luyện một mô hình mạng nơron. Mục tiêu của phương pháp này chính là việc muốn chuẩn hóa các đặc trưng (đầu ra của mỗi layer sau khi đi qua các hàm kích hoạt) về trạng thái zero-mean với độ lệch chuẩn 1. Trong YOLOv4, sau khi các đặc trưng qua hàm Mish sẽ được chuẩn hóa.

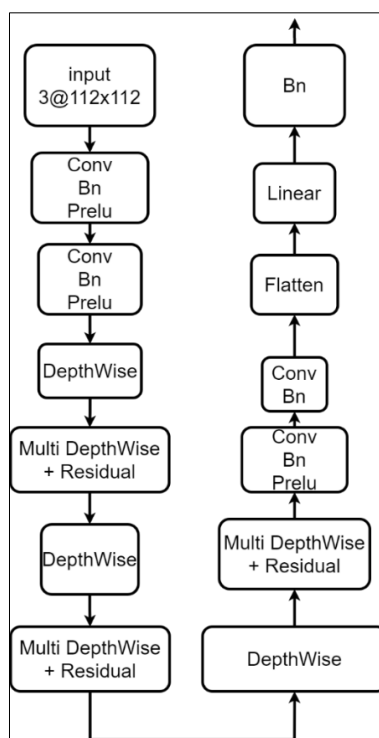
2.4. Giải pháp nhận diện khuôn mặt**2.4.1. Công thức sử dụng (Giải pháp MobileFaceNet + Arcface)****Bảng 4: Các công thức sử dụng**

| Tên công thức | Công thức | Mục đích |
|-----------------------------------|--|---|
| Prelu | $f(x) = \begin{cases} x & x > 0 \\ \alpha(x) & x \leq 0 \end{cases}$ | Activate Function |
| Batch normalization | $y = \frac{x - E[x]}{\sqrt{\text{Var}[x] + \epsilon}} * \gamma + \beta$ | Chuẩn hóa các đầu vào thành một lớp cho mỗi batch, giúp ổn định quá trình học tập và giảm đáng kể số lượng epoch cần thiết để đào tạo |
| Euclidean Distance (norm2) | $\text{norm2}(a, b) = \sqrt{\sum_i \ x_i\ ^2}$ | Khoảng cách giữa 2 điểm bất kì trong không gian Euclid nhiều chiều (xi là hiệu giữa 2 thành phần trên chiều thứ i) |

| | | |
|-----------------------------------|---|---|
| L2_Normalization | $L2_norm(x) = \frac{x}{norm2(x)}$ | Chuẩn hóa dữ liệu |
| Cross entropy loss funtion | $CE(t, s) = - \sum_i^C t_i \log(s_i)$ $= CE = -\log\left(\frac{e^{s_i}}{\sum_j^C e^{s_j}}\right)$ | Dùng để đo lường hiệu suất của mô hình phân loại có đầu ra là giá trị xác suất từ 0 đến 1 |

2.4.2. Mobile Facenet

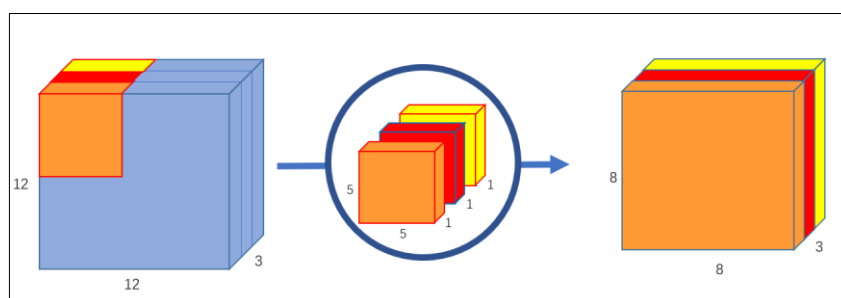
Kiến trúc chi tiết



Hình 11: Kiến trúc chi tiết MobileFaceNet

Depth-wise Convolution

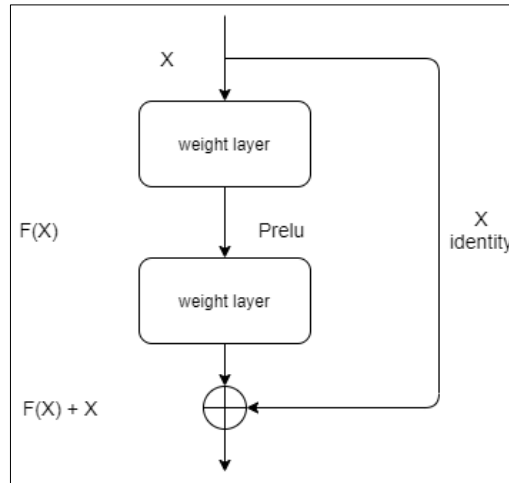
Depth-wise Convolution là tích chập giữa fillter với từng input chanel.



Hình 12: Minh họa Depth-wise Convolution

Residual

Residual là việc thêm input vào output, cũng được gọi là skip-connection, việc này cho phép làm mượt gradient trong back-propagation và có thể đưa những feature quan trọng đến bước cuối cùng.



Hình 13: Minh họa Residual

2.4.3. Loss Function ArcFace

Tóm tắt

Bởi vì tác giả muốn có một đầu ra là xác suất để sử dụng trong cross-entropy loss nên để bắt đầu với loss function Arcface, tác giả đi từ công thức của activate function Softmax:

$$L_1 = -\frac{1}{m} \sum_{i=1}^m \log \left(\frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_{j=1}^n e^{W_{y_i}^T x_i + b_{y_j}}} \right)$$

Để đơn giản, tác giả sửa lại độ lệch (bias) $b_j = 0$, sau đó biến đổi tử số bằng tích vô hướng của 2 vector:

$$W_j^T x_i = \|W_j\| \|x_i\| \cos(\theta_j)$$

Chuẩn hóa norm của W_j (tức là $\|W_j\|$) thành 1 bằng L2 normalization, điều này sẽ làm cho dự đoán chỉ phụ thuộc vào góc giữa feature vector và weight (tức góc θ), L1 được viết lại như sau:

$$L_2 = -\frac{1}{m} \sum_{i=1}^m \log \frac{e^{\|x_i\| \cos(\theta_{y_i})}}{e^{\|x_i\| \cos(\theta_{y_i})} + \sum_{j=1, j \neq y_i}^n e^{\|x_i\| \cos(\theta_j)}}$$

Norm của feature được chuẩn hóa thành 1 bằng cách áp dụng L2 normalization, và re-scale $\|x_i\|$ với hệ số co giãn s, L2 được viết lại như sau:

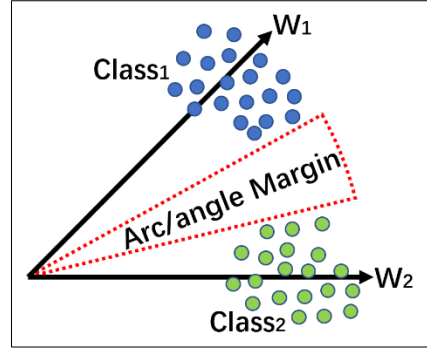
$$L2 = -\frac{1}{m} \sum_{i=1}^m \log \frac{e^{s \cos(\theta_{y_i})}}{e^{s \cos(\theta_{y_i})} + \sum_{j=1, j \neq y_i}^n e^{s \cos(\theta_j)}}$$

Hệ số trừng phạt m (angular margin m) được thêm trực tiếp vào θ_{y_i} , công thức cuối cùng của Arcface được viết như sau:

$$L3 = -\frac{1}{m} \sum_{i=1}^m \log \frac{e^{s (\cos(\theta_{y_i}) + m)}}{e^{s (\cos(\theta_{y_i}) + m)} + \sum_{j=1, j \neq y_i}^n e^{s \cos(\theta_j)}}$$

Với $W_j = \frac{W_j}{\|W_j\|}$, $x_i = \frac{x_i}{\|x_i\|}$, $\cos(\theta_j) = W_j^T x_i$

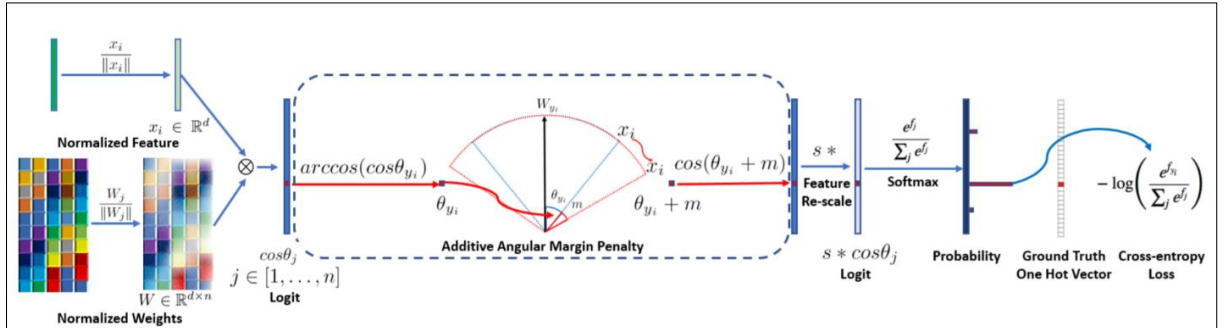
Biểu diễn hình học



Hình 14: Biểu diễn hình học của ArcFace

Minh họa hình học cho Arcface. Gồm 2 class là class1 và class2. Vùng chấm màu xanh dương đại diện cho embedding của class1, vùng màu xanh lục đại diện cho embedding của class2. Vùng trong nét đứt đỏ đại diện cho ranh giới phân chia 2 class (góc $\theta + m$), có thể thấy 2 class phân chia rất rõ ràng.

Mô hình



Hình 15: Mô hình quá trình training mạng DCNN sử dụng Arcface

Mã giả

- Áp dụng chuẩn hóa L2 lên embeddings (có được sau khi input đi qua mobilenet) và weights (ban đầu khởi tạo ngẫu nhiên).
- Tính $\cos(\theta)$ bằng cách tính tích vô hướng của embeddings và weights.
- Tính $\cos(\theta + m)$ bằng cách thêm hằng số trừng phạt m vào góc θ , công thức:
$$\cos(\theta + m) = \cos(\theta) * \cos(m) - \sin(\theta) * \sin(m)$$
- Sử dụng cross entropy loss function trên giá trị $\cos(\theta + m)$ vừa mới tính để tính giá trị loss.

2.5. Giải pháp ứng dụng di động

2.5.1. Phát triển bài toán

Xây dựng app android để tương tác với hệ thống server và raspberry để thực hiện và quản lý điểm danh. Hệ thống bao gồm 2 tác nhân chính đó là giáo viên và học sinh.

Giáo viên có quyền xem lịch dạy của bản thân, xem thông tin cá nhân, xem lịch sử điểm danh của lớp, xem danh sách lớp, thêm lớp học, lịch sử điểm danh và kết quả điểm danh. Ngoài ra giáo viên còn có thể can thiệp vào kết quả điểm danh bằng tính năng điểm danh tự động. Để sử dụng được các chức năng đó giáo viên phải đăng nhập bằng tài khoản của mình.

Ngoài giáo viên ra còn có học sinh tham gia vào hệ thống. Học sinh có quyền kiểm tra lại kết quả điểm danh của mình trong lớp đó bằng cách đăng nhập mã số học sinh và mã lớp đó.

2.5.2. Công nghệ sử dụng

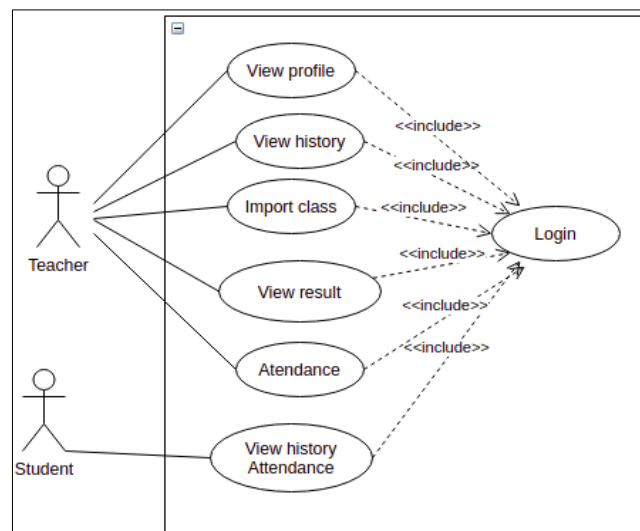
Glide: Dùng để load ảnh vào ImageView.

Paper: để lưu giữ dữ liệu account giúp cho tính năng ghi nhớ điểm danh.

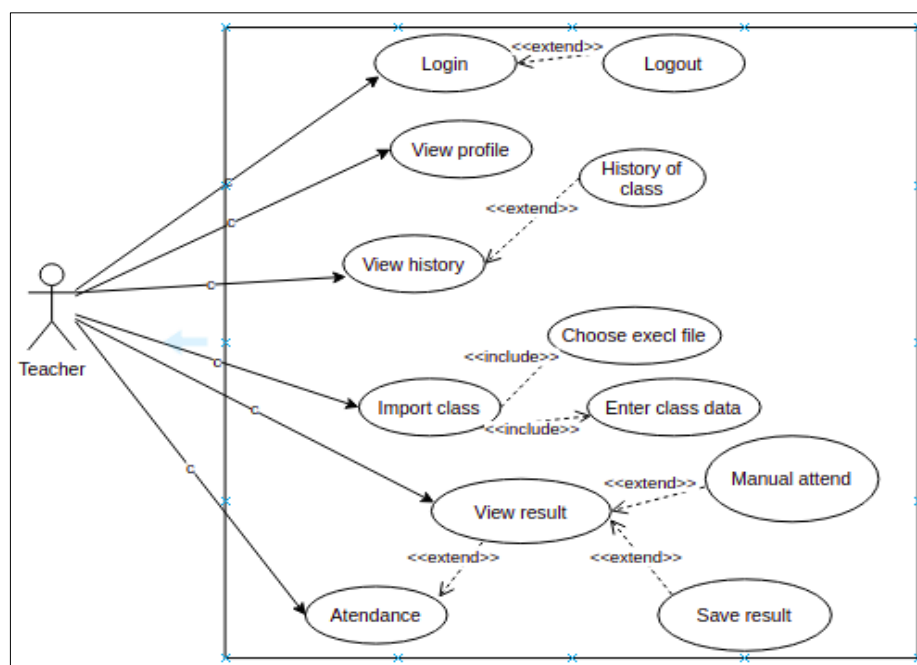
Volley: Dùng để lấy dữ liệu từ server bằng lệnh httprequest.

Rxjava2: Dùng để quản lý luồng. Giúp cho việc trao đổi giữa app và server.

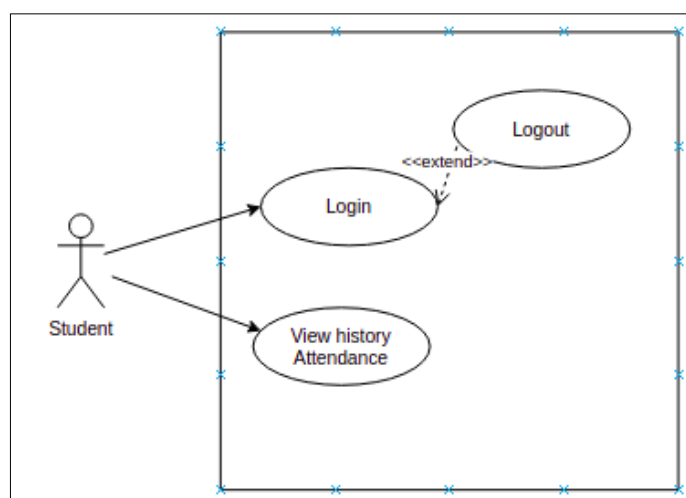
2.5.3. Biểu đồ usecase hệ thống



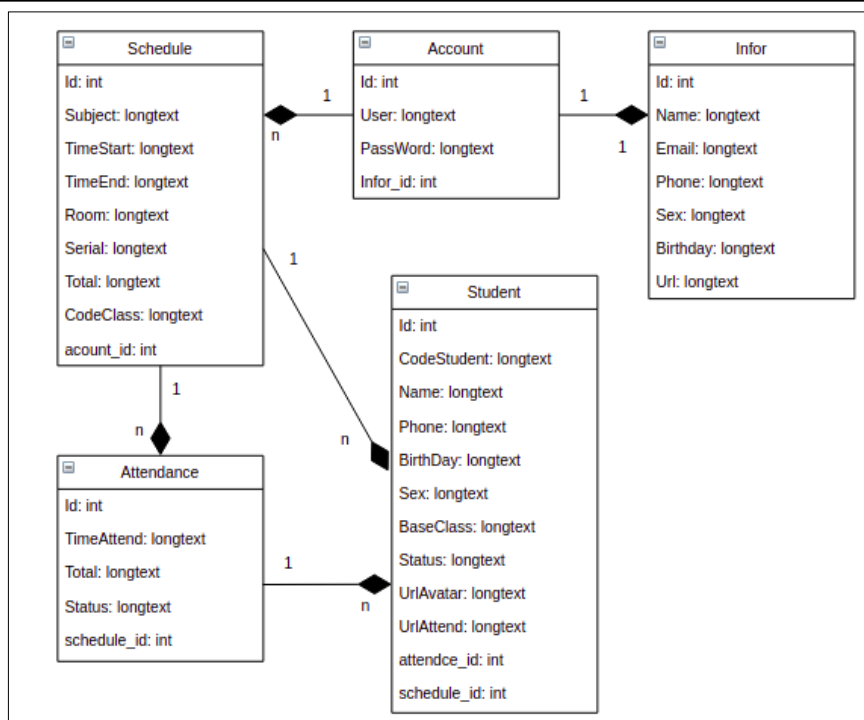
Hình 16: Biểu đồ usecase tổng quát



Hình 17: Biểu đồ usecase tác nhân giảng viên



Hình 18: Biểu đồ usecase tác nhân sinh viên

**Hình 19: Sơ đồ lớp**

3. Kết quả

3.1. Phát hiện khuôn mặt

3.1.1. Tập dữ liệu

Trong phần huấn luyện mô hình phát hiện khuôn mặt, nhóm sử dụng tập dữ liệu Wider Face có sẵn. Thông tin về tập dữ liệu:

Nguồn: <http://shuoyang1213.me/WIDERFACE/>

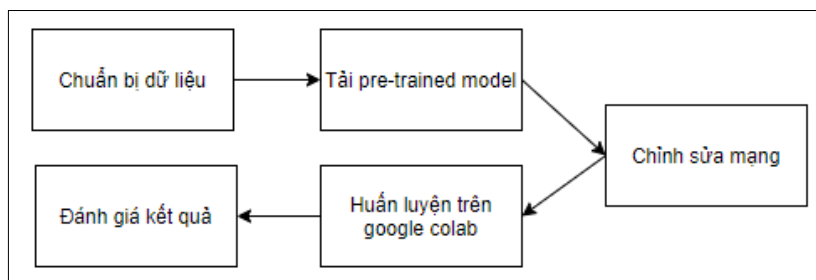
Mô tả: Tập dữ liệu Wider Face là tập dữ liệu chuẩn cho phát hiện khuôn mặt, trong đó hình ảnh được chọn từ tập dữ liệu Wider công khai. Tập này có 32.203 hình ảnh và gán nhãn 393.703 khuôn mặt có mức độ thay đổi cao về tỷ lệ, tư thế. Bộ dữ liệu chứa các hình ảnh về 61 sự kiện khác nhau. Trong đó, tập dữ liệu đã được chia sẵn thành: Train (12880 hình ảnh), Val (3226 hình ảnh), Test (16.097 hình ảnh).

Phân chia dữ liệu: Train: 12.880 (70%), Val: 2.760(15%), Test: 2.760 (15%).

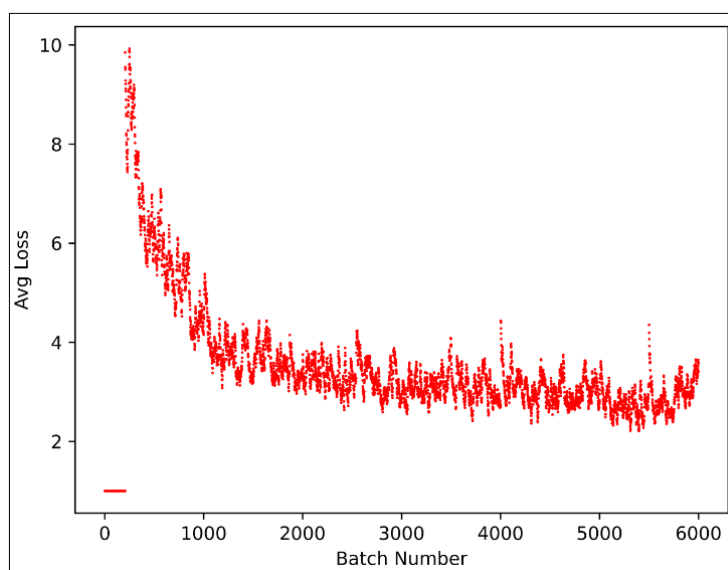
3.1.2. Huấn luyện

Sau khi có được dữ liệu và phân chia dữ liệu theo tỉ lệ phù hợp. Nhóm đã tiến hành tải folder Darknet của AlexeyAB. Darknet là một folder chứa các pre-trained model và các kiến trúc mạng. Từ đây, nhóm chỉnh sửa kiến trúc mạng sử dụng cho Yolo và dùng pretrained-model tương ứng để đi huấn luyện trên Google Colab. Sau khi huấn luyện 6000 batch đã cho ra mô hình phát hiện khuôn mặt. Sau khi huấn luyện nhóm sử dụng AP để tiến hành đánh giá hiệu suất mô hình. Dựa vào hình vẽ ta thấy đồ thị hàm Loss đi từ trên cao xuống và càng về cuối thì giao động trong khoảng 2,4 – 3,6. Từ

đó, cho thấy mô hình chưa hoàn thiện vì Avg Loss chưa hội tụ và cần phải huấn luyện thêm.



Hình 20: Sơ đồ quá trình chuẩn bị đến huấn luyện



Hình 21: Đồ thị Avg Loss quá trình huấn luyện

3.1.3. Kết quả giải pháp phát hiện khuôn mặt

Kết quả định tính

- Khi camera có độ phân giải thấp, rất khó phát hiện ở khoảng cách lớn.
- Vẫn còn phát hiện nhầm từ không có khuôn mặt thành có khuôn mặt.
- Trong phạm vi lớp học có thể phát hiện tất cả khuôn mặt nhưng trong phạm vi rộng lớn như sân vận động,... chỉ phát hiện được 50% tổng số khuôn mặt.

Kết quả định lượng

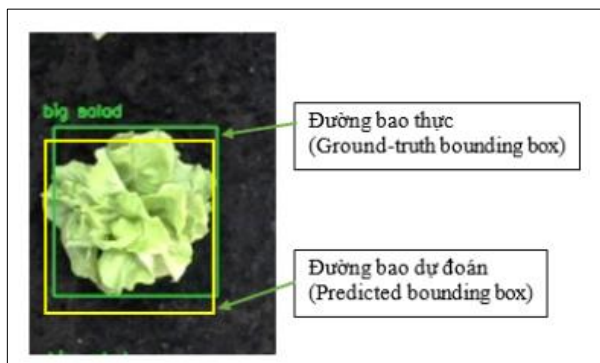
Để tiến hành đánh giá hiệu suất mô hình, nhóm sử dụng mAP (mean Average Precision). mAP là độ đo được sử dụng rất phổ biến cho các bài toán Object Detection. Tuy nhiên, trong mAP còn có nhiều khái niệm khác về Precision, Recall, IOU.

IOU (Intersection over union)

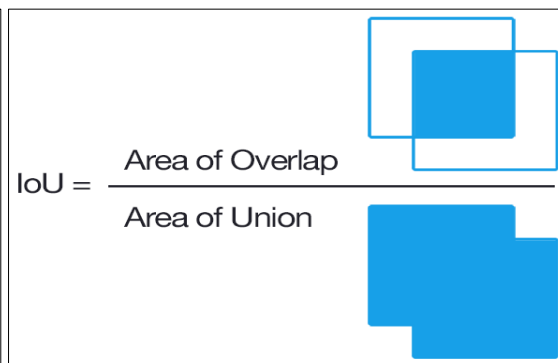
IOU là chỉ số đánh giá được sử dụng để đo độ chính xác của phát hiện đối tượng dựa trên tập dữ liệu cụ thể. Để áp dụng được IOU để đánh giá, cần một đường bao thực là đường bao mà chúng ta gán cho vật thể và một đường bao dự đoán là đường bao dùng

Báo cáo đồ án PBL5 - Kỹ thuật máy tính

model sau khi huấn luyện để nhận dạng. Dựa trên hai đường bao này, ta đo được mức độ giao nhau giữa hai đường bao để xác định khung hình có bị đè chồng lên nhau không. Tỷ lệ này được tính dựa trên phần diện tích giao nhau giữa hai đường bao với phần diện tích không giao nhau giữa chúng.



Hình 22: Đường bao thực, dự đoán



Hình 23: Công thức tính IOU

Độ đo

Precision x Recall curve

Precision x Recall curve là cách tốt nhất đánh giá hiệu suất của một bộ phát hiện khuôn mặt với độ tự tin được thay đổi theo đồ thị đường cong cho mỗi lớp. Một bộ phát hiện khuôn mặt được coi là tốt nếu precision của nó vẫn cao khi recall cao, hoặc là khi thay đổi ngưỡng thì precision và recall vẫn cao.

Precision: Là độ đo đánh giá độ tin cậy của kết luận đưa ra (bao nhiêu % lời kết luận của model là chính xác).

Recall: Là độ đo đánh giá khả năng tìm kiếm toàn bộ các ground truth của mô hình (bao nhiêu % positive sample mà model nhận diện được).

True Positive (TP): Một kết quả phát hiện đúng ($IOU \geq \text{ngưỡng}$).

False Positive (FP): Một kết quả phát hiện sai ($IOU < \text{ngưỡng}$).

False Negative (FN): Một khuôn mặt nhưng không phát hiện ra.

True Negative (TN): Một kết quả không phải là khuôn mặt nhưng phát hiện là khuôn mặt.

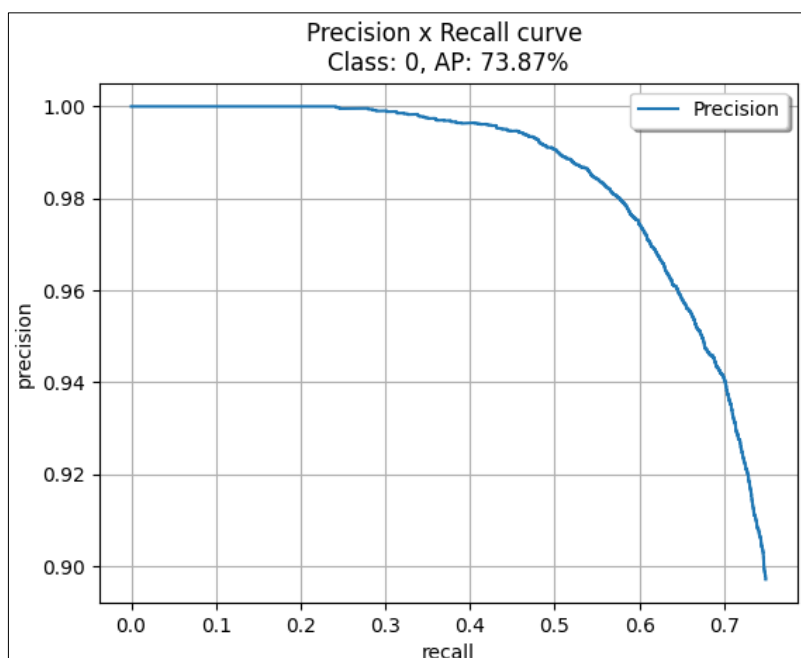
| | |
|----------------------------------|------------------------------|
| $Precision = \frac{TP}{TP + FP}$ | $TP = \text{True positive}$ |
| $Recall = \frac{TP}{TP + FN}$ | $TN = \text{True negative}$ |
| | $FP = \text{False positive}$ |
| | $FN = \text{False negative}$ |

Hình 24: Công thức tính Precision và Recall

Average Precision (AP)

Cách khác để so sánh hiệu suất của các bộ phát hiện khuôn mặt là tính diện tích của đồ thị đường cong (UAC) của Precision x Recall curve. Đường cong AP thường là đường zigzag lên rồi xuống. Trong thực tế, AP là độ chính xác được tính trung bình trên tất cả các giá trị recall giữa 0 và 1.

$$AP = \sum_{k=0}^{k=n-1} [Recalls(k) - Recalls(k+1)] * Precisions(k)$$
$$Recalls(n) = 0, Precisions(n) = 1$$
$$n = \text{Number of thresholds.}$$

Hình 25: Công thức tính AP**Kết quả đo với AP****Hình 26: Kết quả đo mô hình với AP**

Dựa vào hình vẽ ta thấy khi recall tăng lên 0,7, precision vẫn cao (~90%). Điều này cho thấy khả năng phát hiện khuôn mặt trên tổng số các trường hợp phát hiện (Đúng + sai) rất cao, ít có khuôn mặt phát hiện sai. Khả năng phát hiện khuôn mặt trên tổng số các trường hợp phát hiện (Đúng + Không phát hiện ra) cũng rất cao, số khuôn mặt không phát hiện ra rất nhiều. Nguyên nhân nhiều khuôn mặt không bị phát hiện vì trong tập kiểm định có nhiều hình ảnh ở các sự kiện như sân vận động, meeting,.... Có số lượng khuôn mặt lớn trong khung hình (~100 khuôn mặt) nhưng những khuôn mặt rất nhỏ và mờ nên việc sẽ không phát hiện ra hoặc có phát hiện ra nhưng việc định vị các pixel sẽ có độ chênh lệch lớn có thể dẫn đến định vị khuôn mặt sai.

**Bảng 5: Kết quả đo tốc độ thực thi
(Intel Core i5-8250U, RAM: 12GB, Fre: 1,6GHz)**

| Hình | Số khuôn mặt | Số khuôn mặt phát hiện | Tốc độ phát hiện tất cả khuôn mặt (s) |
|------|--------------|------------------------|---------------------------------------|
| 1 | 1 | 1 | 0,80 |
| 2 | 5 | 5 | 0,83 |
| 3 | 10 | 10 | 0,86 |
| 4 | 30 | 30 | 0,92 |
| 5 | 36 | 36 | 0,95 |

Tốc độ thực thi của mô hình nhận diện khuôn mặt rất nhanh (~1s) và chính xác, hoàn toàn có thể đáp ứng cho hệ thống chạy theo thời gian thực. Nguyên nhân thời gian phát hiện trên các hình có số lượng khuôn mặt khác nhau nhưng gần nhau do thời gian tải trọng số chiếm nhiều thời gian và thời gian phát hiện khuôn mặt rất ngắn.

3.2. Giải pháp nhận diện khuôn mặt

3.2.1. Dữ liệu

Face Emores, được nén nhị phân bằng mxnet.

Gồm các dataset: agedb_30, calfw, cfp_ff, cfp_fp, cplfw, lfw

Nguồn: <https://github.com/deepinsight/insightface/wiki/Dataset-Zoo>

Bảng 6: Thông tin các dataset được sử dụng trong face emores và các file

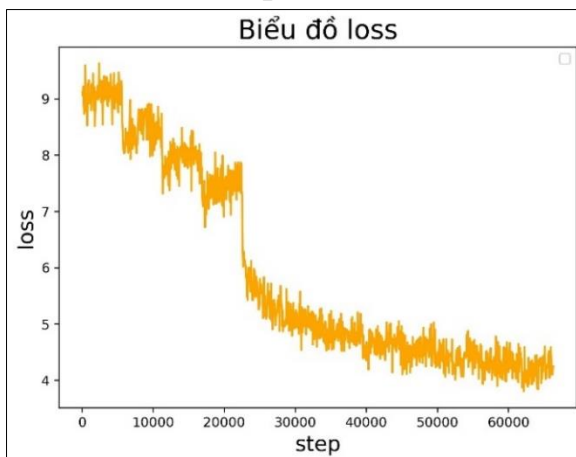
| Tên file | Mô tả | Chức năng | Ghi chú |
|--------------|--|---|--|
| agedb_30.bin | Dữ liệu khuôn mặt thu thập ở các độ tuổi | Valid | 12000 ảnh kích thước 3@112x112 |
| cfp_fp.bin | Dữ liệu chứa khuôn mặt chính diện và khuôn mặt khi xoay đầu 90 độ | Valid | 14000 ảnh kích thước 3@112x112 |
| lfw.bin | Dữ liệu chứa khuôn mặt chủ yếu dùng trong valid | Valid | 12000 ảnh kích thước 3@112x112 |
| property | Chứa thông tin tổng quát của dữ liệu train | Chứa thông tin về số lượng folder (mỗi folder là 1 người) và size ảnh | Nội dung: 85742 112 112 (85742 folder, mỗi ảnh cỡ 112x112) |
| train.idx | Chứa tên và ảnh của tập train, các khuôn mặt được lấy ở nhiều góc độ khác nhau | train | Horizontal resolution: 96 dpi |
| train.rec | | | Vertical resolution: 96 dpi |
| | | | Bit depth: 24 |

3.2.2. Kết quả giải pháp nhận diện khuôn mặt

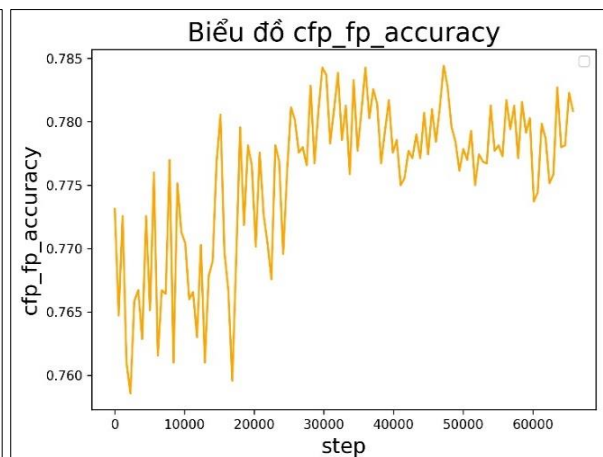
Điều kiện tiến hành thực nghiệm

- Arcface:
 - $m = 0.5$ (tương đương $60^\circ = \arccos(0.5 \text{ rad})$).
 - $s = 64.0$ (giá trị cho kết quả nhận diện tốt nhất theo).
- MobileFaceNet:
 - Huấn luyện 20 epoch.
 - Weight decay parameter = $4e-5$.
 - SGD có momentum = 0.9 để tối ưu model, batch size = 512.
 - Learning rate khởi tại bằng 0.01 và cập nhật bằng cách chia cho 10 ở các epoch 3 7 15 18.
- Infer (nhận diện):
 - Threshold = 1.5 (ngưỡng nhận diện, vượt quá ngưỡng này thì khuôn mặt được đánh dấu là không tồn tại trong cơ sở dữ liệu).

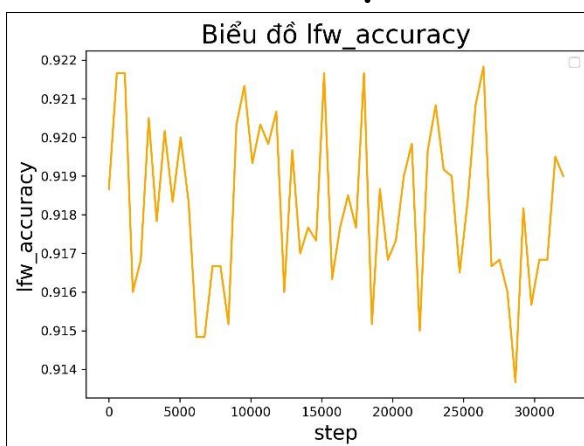
Đồ thị kết quả



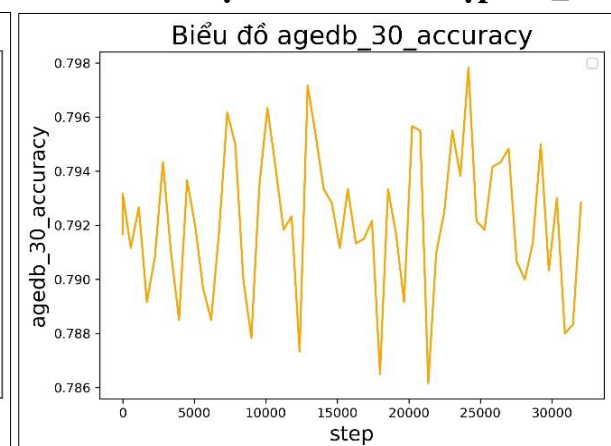
Hình 27: Đồ thị loss



Hình 28: Tỷ lệ chính xác với tập cfb_fb

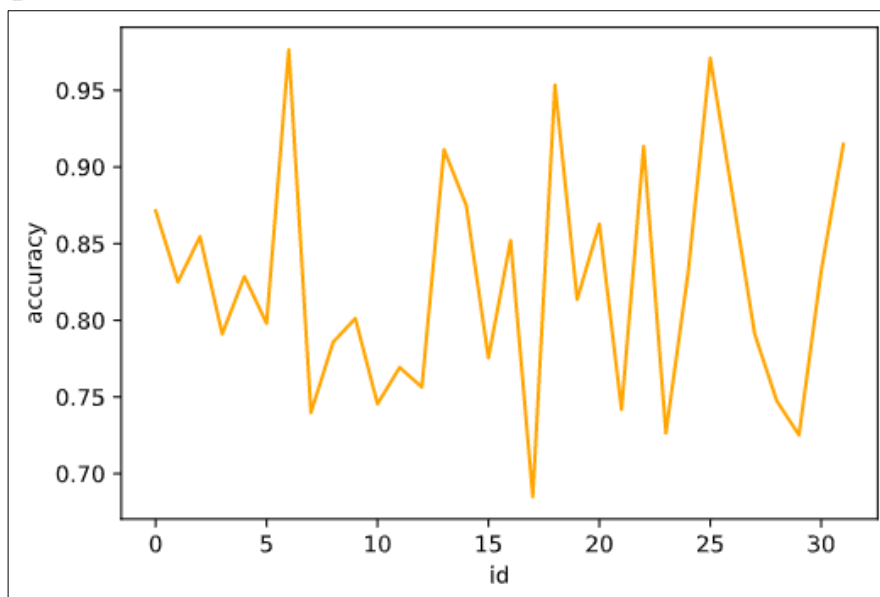


Hình 29: Tỷ lệ chính xác với tập lfw








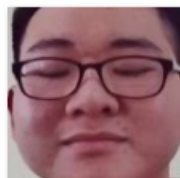



Hình 30: Tỷ lệ chính xác với tập agedb

Kết quả thực tế



Hình 31: Độ chính xác khi thử nghiệm mô hình với dữ liệu một lớp học (82.23%)

Bảng 7: Kết quả thực tế

| Ảnh dự đoán 1 | Ảnh dự đoán 2 | Ảnh gốc | Kết quả |
|--|---|--|----------|
|  102180112_NguyenQuangBao_738 |  102180112_NguyenQuangBao_967 |  im2.png | Khớp 2/2 |
|  102180124_TranQuangHuy_779.jp |  102180123_NguyenQuangHuy_0.7 |  im26.png | Khớp 1/2 |
|  102180139_LeTrungSon_712.jpg |  102180121_LuongMinhHuan_0.71 |  im68.png | Khớp 0/2 |

Tốc độ thực thi

• Điều kiện thực thi

- Tốc độ thực thi được ghi nhận trên cpu intel core i7 gen 8th

- Database chứa 37 dữ liệu đặc trưng khuôn mặt.
- Ảnh chụp trong điều kiện ánh sáng phòng.
- Sử dụng độ đo Euclidean để so khớp.

Bảng 8: Tốc độ thực thi

| Số khuôn mặt | Điều kiện khác | Tốc độ nhận diện |
|--------------|-----------------------------------|------------------|
| 1 | Ảnh chụp chính diện | 0.2s |
| 1 | Ảnh chụp góc nghiêng | 0.72s |
| 35 | Đứng trên bục giảng và nhìn thẳng | 3s |
| 35 | Trong buổi học | 4s |

3.3. Server

3.3.1. API

Account

Bảng 9: Chức năng đăng nhập

| | |
|---------------|--------------------------------------|
| Tên chức năng | Đăng nhập |
| Method | GET |
| URL | /login/<str:username>&<str:password> |
| Parameters | “tên đăng nhập”, “mật khẩu” |

Bảng 10: Chức năng lấy thông tin chi tiết

| | |
|---------------|------------------------|
| Tên chức năng | Lấy thông tin chi tiết |
| Method | GET |
| URL | /info/<str:info_id>/ |
| Parameters | “id chi tiết” |

Bảng 11: Chức năng lấy mã tài khoản

| | |
|---------------|--------------------------|
| Tên chức năng | Lấy mã tài khoản |
| Method | GET |
| URL | /account/<str:username>/ |
| Parameters | “tên tài khoản” |

Student

Bảng 12: Chức năng lấy thông tin sinh viên điểm danh

| | |
|---------------|--|
| Tên chức năng | Lấy thông tin sinh viên điểm danh |
| Method | GET |
| URL | /student/<int:student_id>&<int:codeclass>/ |
| Parameters | “id sinh viên”, “mã học phần” |

Bảng 13: Chức năng lấy kết quả điểm danh của sinh viên

| | |
|---------------|-------------------------------------|
| Tên chức năng | Lấy kết quả điểm danh của sinh viên |
| Method | GET |
| URL | /student/<str:student_id> |
| Parameters | “id sinh viên” |

Bảng 14: Chức năng lấy danh sách sinh viên điểm danh

| | |
|----------------------|-----------------------------------|
| Tên chức năng | Lấy danh sách sinh viên điểm danh |
| Method | GET |
| URL | /student/attend/<str:student_id>/ |
| Parameters | “id sinh viên” |

Bảng 15: Chức năng kiểm tra sinh viên

| | |
|----------------------|---|
| Tên chức năng | Kiểm tra sinh viên |
| Method | GET |
| URL | /checkStudent/<str:codestudent>&<str:schedule_id> |
| Parameters | “mã sinh viên”, ”mã lịch học” |

Bảng 16: Chức năng thêm sinh viên

| | |
|----------------------|------------------|
| Tên chức năng | Thêm sinh viên |
| Method | POST |
| URL | /student/create/ |
| Parameters | None |

Bảng 17: Chức năng chỉnh sửa thông tin sinh viên

| | |
|----------------------|--|
| Tên chức năng | Chỉnh sửa thông tin sinh viên |
| Method | PATCH |
| URL | /student/update/<str:codestudent>&<int:schedule_id>&<int:attendance_id>/ |
| Parameters | “mã sinh viên”, ”mã lịch học”, “mã lịch điểm danh” |

Schedule

Bảng 18: Chức năng lấy lịch học

| | |
|----------------------|---|
| Tên chức năng | Lấy lịch học |
| Method | GET |
| URL | /schedule/<str:schedule_id>&<str:serial>/ |
| Parameters | “id lịch học”, “thứ ngày” |

Bảng 19: Chức năng thêm lịch học

| | |
|----------------------|-------------------|
| Tên chức năng | Thêm lịch học |
| Method | POST |
| URL | /schedule/create/ |
| Parameters | None |

Attendance

Bảng 20: Chức năng lấy lịch sử điểm danh

| | |
|----------------------|-----------------------|
| Tên chức năng | Lấy lịch sử điểm danh |
| Method | GET |
| URL | /class/ |
| Parameters | None |

Bảng 21: Chức năng lấy lịch sử điểm danh của lớp

| | |
|----------------------|-------------------------------|
| Tên chức năng | Lấy lịch sử điểm danh của lớp |
| Method | GET |
| URL | /class/<str:attendance_id>/ |
| Parameters | “id lịch điểm danh” |

Bảng 22: Chức năng lấy lịch sử điểm danh của giáo viên

| | |
|----------------------|---------------------------------------|
| Tên chức năng | Lấy danh sách điểm danh của giáo viên |
| Method | GET |
| URL | /class/gv/<str:attendance_id>/ |
| Parameters | “id lịch điểm danh” |

Bảng 23: Chức năng điểm danh

| | |
|----------------------|--|
| Tên chức năng | Điểm danh |
| Method | GET |
| URL | /attendance/<str:schedule_id>&<str:attendance_id>/ |
| Parameters | “mã lịch học”, “mã lịch điểm danh” |

Bảng 24: Chức năng thêm lịch điểm danh

| | |
|----------------------|---------------------|
| Tên chức năng | Thêm lịch điểm danh |
| Method | POST |
| URL | /attend/create/ |
| Parameters | None |

Bảng 25: Chức năng chỉnh sửa thông tin lịch điểm danh

| | |
|----------------------|-------------------------------------|
| Tên chức năng | Chỉnh sửa thông tin lịch điểm danh |
| Method | PATCH |
| URL | /attend/update/<str:attendance_id>/ |
| Parameters | “mã lịch điểm danh” |

Raspberry

Bảng 26: Chức năng gửi ảnh

| | |
|----------------------|----------|
| Tên chức năng | Gửi ảnh |
| Method | POST |
| URL | /upload/ |
| Parameters | None |

3.3.2. Tốc độ thực thi hệ thống

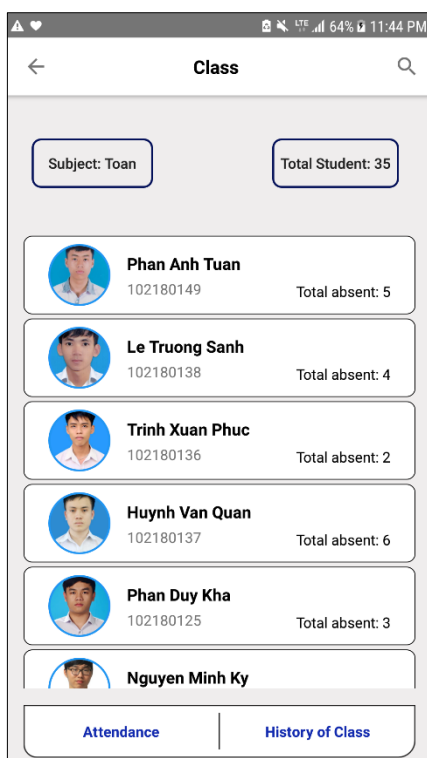
**Bảng 27: Tốc độ thực thi của hệ thống
(RAM 8GB, CPU: 2.5Ghz, Internet: 300Kbs)**

| Số lượng ảnh | Kích thước ảnh (pixels x pixels) | Độ phân giải (dpi) | Số lượng khuôn mặt | Số lượng khuôn mặt phát hiện | Số lượng khuôn mặt điểm danh được | Tổng thời gian chạy (s) | Thời gian app gửi và nhận kết quả |
|--------------|----------------------------------|--------------------|--------------------|------------------------------|-----------------------------------|-------------------------|-----------------------------------|
| 10 | 1280x960 | 96 | 1 | 1 | 1 | 29.62 | 30.21 |
| 10 | 1280x960 | 96 | 1 | 1 | 1 | 29.41 | 30.11 |
| 10 | 1280x960 | 96 | 1 | 1 | 1 | 29.49 | 30.14 |

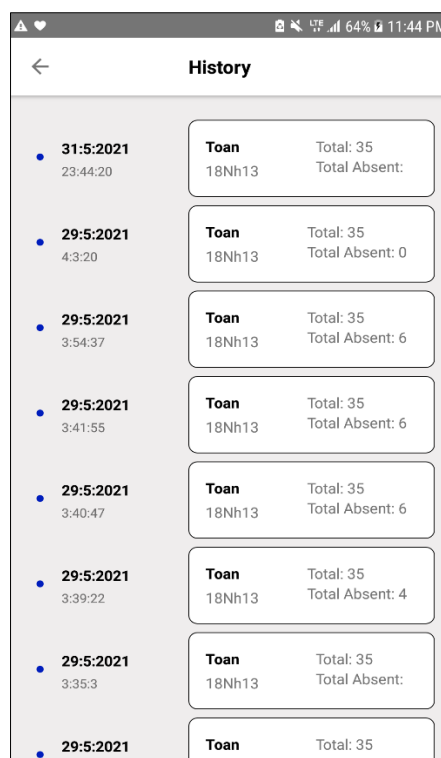
Báo cáo đồ án PBL5 - Kỹ thuật máy tính

| | | | | | | | |
|----|-----------|----|----|----|----|--------|--------|
| 10 | 1280x960 | 96 | 12 | 12 | 12 | 57.69 | 58.71 |
| 10 | 1280x960 | 96 | 12 | 12 | 12 | 61.65 | 62.77 |
| 10 | 1280x960 | 96 | 12 | 12 | 12 | 59.85 | 60.89 |
| 10 | 2560x1920 | 96 | 33 | 33 | 20 | 102.51 | 103.74 |
| 10 | 2560x1920 | 96 | 33 | 33 | 20 | 102.82 | 104.03 |
| 10 | 2560x1920 | 96 | 33 | 33 | 20 | 102.42 | 103.54 |

4. Ứng dụng di động



Hình 32: Giao diện chi tiết lớp học



Hình 33: Lịch sử điểm danh

5. Kết luận

5.1. Đánh giá

Thiết kế: Sản phẩm thiết kế gọn gàng, trọng lượng nhẹ, có thể di chuyển được và có thể cải tiến mở rộng trong tương lai về cả phần cứng và phần mềm.

Chức năng:

Phát hiện khuôn mặt: Mô hình phát hiện khuôn mặt hiện tại chưa hoàn thiện. Tuy nhiên, khả năng phát hiện khuôn mặt đã cho kết quả tốt, có thể phát hiện được tất cả các khuôn mặt trong lớp học. Tốc độ phát hiện khuôn mặt nhanh, đáp ứng hệ thống thời gian thực, đáp ứng đủ yêu cầu đồ án.

Nhận diện khuôn mặt: Mô hình nhận diện khuôn mặt hiện tại chưa hoàn thiện. Độ chính xác nhận diện còn thấp do lệch về tập dữ liệu huấn luyện và hình ảnh kiểm tra,

do độ phân giải chưa tốt,... Thời gian nhận diện rất nhanh, đáp ứng được hệ thống thời gian thực. Đáp ứng yêu cầu đồ án.

Server: Server hệ thống xử lý ổn định, gần như không xảy ra lỗi. Đáp ứng yêu cầu đồ án.

Ứng dụng di động: Ứng dụng có giao diện bắt mắt, dễ sử dụng, đầy đủ các chức năng, phù hợp với yêu cầu đồ án.

5.2. Hướng phát triển

Phát hiện khuôn mặt: Hiện tại mô hình phát hiện khuôn mặt còn nhiều lỗi khi nhận diện. Trong tương lai, nhóm sẽ mở rộng mô hình bằng cách huấn luyện thêm trên tập dữ liệu sinh viên, tập dữ liệu chứa nhiều người trong không gian rộng lớn hơn như sân vận động,... để tăng khả năng phát hiện khuôn mặt và độ chính xác.

Nhận diện khuôn mặt: Nhóm sẽ tiếp tục huấn luyện mô hình để nâng cao độ chính xác nhận diện. Đồng thời, thu thập thêm nhiều ảnh sinh viên hơn nữa phục vụ cho quá trình huấn luyện.

Server: Server hiện tại xử lý còn chậm, xử lý số request còn thấp. Nhóm sẽ phát triển khả năng xử lý của server, xử lý chế độ đa luồng, đa truy cập. Đồng thời, tăng khả năng bảo mật của hệ thống để tránh những trường hợp như sập server, bị tấn công DDOS. Ngoài ra, tiếp tục cải thiện cơ sở dữ liệu để hệ thống được vận hành tốt, xử lý dữ liệu nhanh hơn.

Ứng dụng di động: Hiện tại ứng dụng đã hoàn thiện. Tuy nhiên, vẫn còn nhiều tính năng chưa phát triển cho người dùng. Nhóm sẽ phát triển thêm nhiều tính năng mới, đồng thời sẽ cho lượng lớn người dùng thử nghiệm sản phẩm để nhận phản hồi và cải thiện.

Phần cứng: Nhóm sẽ sử dụng camera có độ phân giải tốt hơn để có những bức hình ở độ phân giải cao. Đồng thời, nâng cấp phần cứng của bộ vi xử lý, hướng đến đặt server trên bộ vi xử lý.

6. Danh mục tài liệu tham khảo

- [1] Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao, “YOLOv4: Optimal Speed and Accuracy of Object Detection”, Apr 23, 2020.
- [2] Zicong Jjang, Liquan Zhao, Shuaiyang Li, Yanfei Jia, “Real-time object detection method for embedded devices”, China.

- [3] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, Jiaya Jia, “Path Aggregation Network for Instance Segmentation”, Sep 18, 2018.
- [4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, “Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition”, Apr 23, 2015.
- [5] Chien-Yao Wang, Hong-Yuan Mark Liao, I-Hau Yeh, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, “CSPNet: A new backbone that can enhance learning capability of CNN”, Nov 27, 2019.
- [6] No name, “[YOLOv4 Discussion Part 2] Network structure”, ProgrammerSought, <https://www.programmersought.com/article/69684985843>.
- [7] Jia Guo, Jiankang Deng. InsightFace: 2D and 3D Face Analysis Project. 2018.
- [8] Wang, Chi-Feng. Depthwise Convolution.
- [9] Sergey Ioffe, Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. Mon, 2 Mar 2015.
- [10] Brownlee, Jason. A Gentle Introduction to Cross-Entropy for Machine Learning. December 22, 2020.
- [11] Gómez, Raúl. Understanding Categorical Cross-Entropy Loss, Binary Cross-Entropy Loss, Softmax Loss, Logistic Loss, Focal Loss and all those confusing names. May 23, 2018.
- [12] Jiankang Deng, Jia Guo, Niannan Xue, Stefanos Zafeiriou. ArcFace: Additive Angular Margin Loss for Deep Face Recognition. 9 Feb 2019.
- [13] Sheng Chen, Yang Liu, Xiang Gao, Zhen Han. MobileFaceNets: Efficient CNNs for Accurate RealTime Face Verification on Mobile Devices. 15 Jun 2018.
- [14] Siriusdemon. Build-Your-Own-Face-Model.