FOR THE HEADER

CHƯƠNG 1 - MỞ ĐẦU

1.1 Giới thiệu đề tài

1.1.1 Lý do chọn đề tài

1.1.2 Mục tiêu

1.2 Tổng quan hệ thống

1.2.1 Face detection

Face detection là bước đầu tiên trong hệ thống nhận dạng điểm danh, function sử dụng các thuật toán để nhận dạng ra khuôn mặt thông qua các đặc điểm tai, mắt, mũi, miệng, màu da,... Hệ thống ban đầu sử dụng haarcascade của thư viện opencv để nhận dạng khuôn mặt. Nhưng sau đó, việc sử dụng haarcascade không đáp ứng được nhu cầu chạy thời gian thực và việc detect khuôn mặt gặp nhiều sự cố khi camera đặt xa.

Để đáp ứng về độ chính xác và tốc độ detect khuôn mặt cũng như vấn đề chạy thời gian thực, chúng em có tìm hiểu thêm nhiều phương pháp khác nhau và cảm thấy MTCNN là một trong những thuật toán có độ hiệu quả chính xác cao đáp ứng được nhiều nhu cầu của bài toán.

1.2.2 Tracking

Ban đầu hệ thống chỉ có 2 bước là Face detection và Face Recognition nhưng vì việc request và gọi function recognition liên tục với mỗi frame hình gây chậm cho hệ thống và tốn nhiều tài nguyên nên function tracking sinh ra để giảm thiểu hao tốn tài nguyên cho hệ thống. Hệ thống tracking sẽ theo sát khuôn mặt của từng người, mỗi người khi được track sẽ có một id riêng biệt, với mỗi id thì hệ thống chỉ recognition một lần. Hệ thống tracking sử dụng thuật toán SORT (Simple Online Realtime Tracking) để tích hợp vào hệ thống. Sau khi hàm face detection được gọi, SORT sẽ nhận dữ liệu của về khuôn mặt và theo dõi khuôn mặt sau mỗi frame và cập nhật liên tục về vị trí của khuôn mặt và gán id cho từng khuôn mặt . Về hệ thống tracking : https://github.com/nwojke/deep_sort

1.2.3 Face recognition

Đây là bước cuối cùng trong hệ thống , function sẽ nhận input là khuôn mặt của một người và sử dụng các thuật toán học sâu cũng như các model để nhận dạng ra người đó là ai trong data. Sau quá trình tìm hiểu và nghiên cứu, testing nhiều phương pháp khác nhau, training với nhiều phương pháp khác nhau như facenet, thư viện face recognition, RCNN, ... thì phương pháp khả thi nhất là sử dụng mô hình CNN kết hợp với pre-train model vggface2 để train một model với bộ dữ liệu của học sinh, sinh viên trong lớp .

1.2.4 Server

Server sẽ là nơi chứa dữ liệu của các học sinh, sinh viên, các model đã train, nhận request từ client để nhận dạng và điểm danh sinh viên. Việc load model và nhận dạng phải chính xác và liên tục, yêu cầu server phải xử lý dữ liệu liên tục, nên hạn chế những request về server để tăng tốc độ xử lý cũng như tiết kiệm tài nguyên, tránh trường hợp server quá tải .

1.2.5 Client

Hệ thống client sẽ được đổ lên các bo mạch được tích hợp trên từng camera . Các bo mạch sẽ lấy dữ liệu từ camera thông qua opencv và xử lý hình ảnh detect khuôn mặt và tracking . Sau khi lấy được khuôn mặt , client sẽ gửi request lên server để nhận dạng và ghi danh học sinh sinh viên . Vì tích hợp trên bo mạch nhỏ nên việc xử lý hình ảnh phải gọn, code phải tối ưu để giảm lượng tính toán của bo mạch.

1.3 Đặc tả hệ thống

1.3.1 Face detection

- Face detection with MTCNN

MTCNN hoạt động theo 3 bước, mỗi bước có một mạng neural riêng lần lượt là: P-Net, R-Net và O-net. Với mỗi bức ảnh đầu vào, nó sẽ tạo ra nhiều bản sao của hình ảnh đó với các kích thước khác nhau.

Tại P-Net, thuật toán sử dụng 1 kernel 12x12 chạy qua mỗi bức hình để tìm kiếm khuôn mặt. Sau lớp convolution thứ 3, mạng chia thành 2 lớp. Convolution 4-1 đưa ra xác suất của một khuôn mặt nằm trong mỗi bounding boxes, và Convolution 4-2 cung cấp tọa độ của các bounding boxes.

R-Net có cấu trúc tương tự với P-Net. Tuy nhiên sử dụng nhiều layer hơn. Tại đây, network sẽ sử dụng các bounding boxes đc cung cấp từ P-Net và tinh chỉnh là tọa độ. Tương tự R-Net chia ra làm 2 layers ở bước cuối, cung cấp 2 đầu ra đó là tọa độ mới của các bounding boxes, cùng độ tin tưởng của nó.

O-Net lấy các bounding boxes từ R-Net làm đầu vào và đánh dấu các tọa độ của các mốc trên khuôn mặt. Ở bước này, thuật toán đưa ra 3 kết quả đầu ra khác nhau bao gồm: xác suất của khuôn mặt nằm trong bounding box, tọa độ của bounding box và tọa độ của các mốc trên khuôn mặt (vị trí mắt, mũi, miệng)

- SORT Tracking Tại thời điểm hiện tại SORT là một mã nguồn mở miễn phí tốt nhất để theo dõi. SORT được thiết kế để theo dõi đa đối tượng chạy trên thời gian thực. Việc sử dụng thuật toán SORT cho hệ thống sẽ giảm được yêu cầu request từ client mỗi khi nhận một object cũ. Thay vì mỗi frame với cùng số người như nhau nhưng lúc nào cũng request lên server liên tục để hệ thống nhận dạng làm việc, làm cho server quá tải. Do đó hệ thống ở dưới mỗi camera sẽ có thêm hệ thống tracker, theo dõi mỗi người cho tới khi người đó không còn trong khung hình. SORT sẽ đánh dấu mỗi khuôn mặt là một id riêng biệt, với mỗi id riêng biệt sẽ request lên server một lần duy nhất. Sau khi áp dụng hệ thống SORT vào trong hệ thống, số lần request lên server giảm đáng kể, FPS tăng và độ trễ giảm, đáp ứng được yêu cầu chạy thời gian thực của bài toán.

Mã nguồn mở của SORT.

- Face recognition Bước nhận dạng là bước quan trọng nhất trong hệ thống, mang tính quyết định cho hệ thống tốt hay không tốt, do đó những model có độ chính xác cao như face-net, face-recognition lib, … đều đã được thử qua, nhưng đều không mang tới kết quả tốt với tập dữ liệu của công ty. Do đó chúng tôi quyết định sử dụng CNN để tự build model cho chính tập dữ liệu của mình. Quá trình train model đã sử dụng tensorflow-gpu nhưng vì chưa khai thác hết tài nguyên cũng như công dụng của tensorflow-gpu nên việc training cũng như load model gây độ trễ nhất định.

CHƯƠNG 2 - CHI TIẾT HỆ THỐNG

Dependencies

Để sử dụng hệ thống , về môi trường

- Python 3.8.2
- Opencv-contrib-python

```
Pip install opencv-contrib-python==4.0.2.34
```

- flask

```
Pip install flask
```

- numpy

```bash Pip install numpy==1.19.0
```

 Pillow and request

```bash
Pip install Pillow request
```

- Pytorch và torch-vision

Pytorch.org

Sau khi cài đặt được pytorch , các bạn phải cài đặt facenet-pytorch

- facenet-pytorch

```
Pip install facenet-pytorch
```

Thư mục

```
.
├── aligner.py
├── avatar
├── config
│   └── config
├── Data
├── detector.py
├── face_detect.py
├── get_data.py
├── gui.py
├── icon.jpg
├── linh_api.py
├── model
├── person.py
├── sort
│   ├── LICENSE
│   └── sort.py
└── train_data
```

Lấy data

Trong folder Data sẽ chứa các file video (định dạng (tên người).mp4).Vì đây là video để lấy data mẫu nên yêu cầu trong đoạn clip chỉ xuất hiện 1 người duy nhất.Để lấy data của người mở file lên và chỉnh sửa stream_path ở dòng 12. Ví dụ muốn lấy data của Huy , stream_path sẽ sửa thành

```
stream_path = "./Data/Huy.mp4"
```

Lưu file lại và mở Terminal lên , dẫn vào đường dẫn bạn đang chưa folder và chạy lện python get_data.py . Ví dụ:

```
doanthienthuan@pop-os:~/work/face_recognition_update$ python get_data.py
```

Data lấy được là một loạt các hình ảnh chứa khuôn mặt của một người , được lưu vào folder mang tên người đó và folder đó được lưu trong folder mang tên là train_data Tiếp tục thay đổi stream_path và chạy lại lên terminal trên đến khi lấy được hết tất cả data của mọi người.

Train and save model on google colab

Triển khai hệ thống

- Server

Trước khi chạy server , bạn cần phải có file model đã train được từ google colab , và đưa nó vào folder model.Chắc chắn rằng tên model trong folder với trong file , mở file linh_api.py và chỉnh sửa dòng resnet.load_state_dict(torch.load("model/**name_of_model.pt**")).Ví dụ model sau khi train được lưu tên là modelv4_16_8631_v6_1.pt

```
resnet.load_state_dict(torch.load("model/modelv4_16_8631_v6_1.pt"))
```

Sau đó mở terminal ra và chạy lệnh

```
python linh_api.py
```

Nếu như terminal xuất hiện dòng lệnh như dưới đây thì server đã hoạt động

```
Device: cuda
 * Serving Flask app "linh_api" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5001/ (Press CTRL+C to quit)
```

- Client

Chạy trên command line

Mở file detector.py và chỉnh stream_path thành tên file nếu muốn test trên video hoặc đổi thành 0 hoặc 1 để detect realtime trên cam của máy tính. Với camera ip thì stream_path mỗi cam mỗi khác, cần có đường dẫn cụ thể.

Sau đó chạy lệnh

```
python detector.py
```

Chạy trên GUI

```
python gui.py
```