BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CẦN THƠ
KHOA CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG

BÁO CÁO MÔN HỌC
MÁY HỌC NÂNG CAO

Đề tài

ỨNG DỤNG CNN ĐỂ DỰ DOÁN THÀNH VIÊN NHÓM BẰNG KHUÔN MẶT

Sinh viên thực hiện:

Nguyễn Thanh Tuyền	**B1304519**
Phan Huỳnh Tân	**B1509892**
Nguyễn Tấn Thành	**B1509894**
Nguyễn Hồng Phát	**B1509942**

Cần Thơ, 3/2019

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CẦN THƠ
KHOA CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG

BÁO CÁO MÔN HỌC
MÁY HỌC NÂNG CAO

Đề tài

ỨNG DỤNG CNN ĐỂ DỰ DOÁN THÀNH VIÊN NHÓM BẰNG KHUÔN MẶT

Giáo viên hướng dẫn:
PGS TS.Pham Nguyên Khang

Sinh viên thực hiện:

Nguyễn Thanh Tuyền	**B1304519**
Phan Huỳnh Tân	**B1509892**
Nguyễn Tấn Thành	**B1509894**
Nguyễn Hồng Phát	**B1509942**

Cần Thơ, 3/2019

NHẬN XÉT CỦA GIẢNG VIÊN

MỤC LỤC

PHẦN GIỚI THIỆU

Phần này trình bày vấn đề được đặt ra đối với đề tài, nêu lên lịch sử giải quyết vấn đề, xác định mục tiêu đề tài, xác định đối tượng và phạm vi nghiên cứu, liệt kê phương pháp nghiên cứu và kết quả mà đề tài cần đạt được.

1. Đặt vấn đề

Ngày nay, trí tuệ nhân tạo đang là tâm điểm nghiên cứu của các nhà khoa học máy tính, bởi lợi ích mà trí tuệ nhân tạo mang lại, chúng được ứng dụng rộng rãi trong nhiều lĩnh vực của cuộc sống thường ngày như thương mại, các ứng dụng trợ lý ảo dành cho điện thoại thông minh cho đến các vấn đề mang tính học thuật cao như hệ chuyên gia, người máy học (robotics). Ngoài các lĩnh vực đã liệt kê, không thể không nhắc đến lĩnh vực thị giác máy tính mà ở đó, với trí thông minh nhân tạo, máy tính có thể phân lớp và dự đoán vật thể thông qua hình ảnh từ camera ghi lại.

Trong lĩnh vực này, thì các mô hình mạng nơ-ron của Học sâu (Deep learning) đang được áp dụng vào các sản phẩm tại các công ty công nghệ lớn như: Alex Assistant (Amazon), Alpha Go (Google),... cho đến những ứng dụng đơn giản như nhận dạng và phân loại mà chúng ta vẫn tận dụng hàng ngày. Để bắt kiệp với tốc độ phát triển đó, trong bài báo cáo này ứng dụng mạng nơ-ron tích chập (CNN) sẽ được sử dụng để giải quyết một bài toán cơ sở về nhận dạng người có mặt trong ảnh.

2. Mục tiêu đề tài

- Xây dựng được tập dữ liệu hình ảnh của từng thành viên trong nhóm.
- Tìm hiểu về học sâu và mô hình CNN.
- Phân tích tập dữ liệu đã thu thập và xây dựng được mô hình huấn luyện, dự đoán.
- Viết được ứng dụng dự đoán thành viên nhóm thông qua camera.

3. Đối tượng và phạm vi nghiên cứu

3.1 Đối tượng

Với mục tiêu đã đề ra thì đối tượng nghiên cứu của đề tài là công nghệ xử lý ảnh, thị giác máy tính được ứng dụng thông qua thư viện mã nguồn mở OpenCV, và xâu dựng mô hình dự đoán CNN thông qua gói thư viện Keras trên nền tảng ngôn ngữ Python.

3.2 Phạm vi

- D2 liạm viêu đã đề ra thì đối tượ t liạm viêu đã đề ra thì đố
- Sử dụng mô hình CNN cho tập dữ liệu.

- Được xây dựng bằng gói thư viện Keras.
- Kết quả là ứng dụng trên máy tính.

4. Phương pháp nghiên cứu

- Tìm kiếm, tổng hợp và phân tích tài liệu tham khảo.
- Nghiên cứu các công nghệ, thư viện mã nguồn mở có thể giúp ích cho việc giải quyết các vấn đề gặp phải.
- Tìm kiếm, phát triển giải pháp.
- Xây dựng chương trình.
- Thử nghiệm, đánh giá chương trình.

5. Kết quả đạt được

- Xây dựng được chương trình dự đoán dự đoán thành các thành viên trong nhóm thông qua camera với độ chính xác tương đối cao.
- Quyển báo cáo đề tài.

6. Bố cục quyển báo cáo

Phần giới thiệu

Giới thiệu tổng quát về đề tài, nêu vấn đề mà đề tài cần giải quyết, trình bày lịch sử giải quyết vấn đề ở ngoài nước và cả trong nước, xác định mục tiêu đề tài, nhận định đối tượng và phạm vi nghiên cứu, liệt kê phương pháp nghiên cứu và kết quả đã đạt được khi hoàn thành mục tiêu đề tài đặt ra.

Phần nội dung

Chương 1: Mô tả bài toán.

Chương 2: Thiết kế, cài đặt giải thuật, biểu diễn cơ sở dữ liệu, trình bày các bước xây dựng hệ thống bằng phương pháp lọc cộng tác.

Chương 3: Kiểm thử hệ thống và đánh giá độ chính xác, tốc độ của hệ thống.

Phần kết luận

Trình bày kết quả đạt được và hướng phát triển hệ thống.

PHẦN NỘI DUNG

Phần này sẽ mô tả chi tiết bài toán đặt ra, giới thiệu về các vấn đề liên quan đến bài toán, thư viện Keras, trình bày các vấn đề và giải pháp tương ứng được đề tài sử dụng, thiết kế và cài đặt giải pháp sẽ được giải thích một cách chi tiết.

CHƯƠNG 1: MÔ TẢ BÀI TOÁN

1. Mô tả chi tiết bài toán

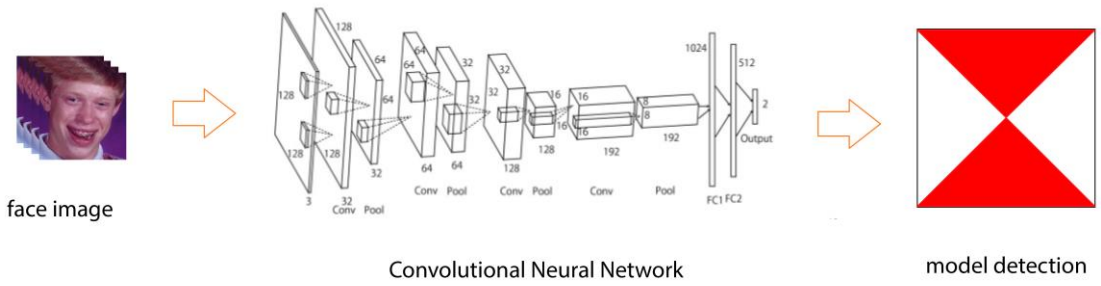
Ứng dụng dựng đoán thành viên thông qua camera sẽ sử dụng camera để chụp ảnh thành viên cần dự đoán và trả về một hình ảnh có chứa mặt thành viên. Sau đó một model nhận diện khuông mặt "*opencv_face_detector*" từ thư viên OpenCV sẽ được gọi và nhận diện khuôn mặt có trong ảnh. Sau khi nhận dạng được khuôn mặt, ứng dụng sẽ tiến hành cắt không mặt ra, chuẩn hóa và sử dụng mô hình đã học để dự đoán thành viên đó là ai trong nhóm.

***Hình 1**: Quá trình dự đoán*

Trong hình trên, để có thể dự đoán được tên của thành viên nhóm, cần phải có một mô hình nhận diện (model detection). Và mô hình nhận diện sẽ được thực hiện qua 2 bước sau:

Đầu tiên, là quá trình thu tập hình ảnh từ những thành viên trong nhóm thông qua camera. Mỗi mỗi thành viên trong nhóm sẽ được chụp với nhiều góc độ khuôn mặt và độ sáng khác nhau với số lượng ảnh nhất định như nhau. Sau mỗi tấm ảnh chụp được, chương trình sẽ sử dụng model nhận diện khuông mặt "*opencv_face_detector*" từ thư viên OpenCV để nhận diện, sau đó tiến hành cắt khuôn mặt và lưu vào một thư mục riêng với tên thư mục chính là tên của thành viên.

Sau khi đã có được tập dữ liệu hình ảnh, bước tiếp theo sẽ chuẩn hóa toàn bộ hình ảnh từ tập dự liệu. Tiến hành thiết kế xây dựng mô hình dự đoán bằng kiến trúc CNN và thực hiện quá trình học để cho ra một mô hình dự đoán.



***Hình 2**: Quá trình tạo mô hình dự đoán*

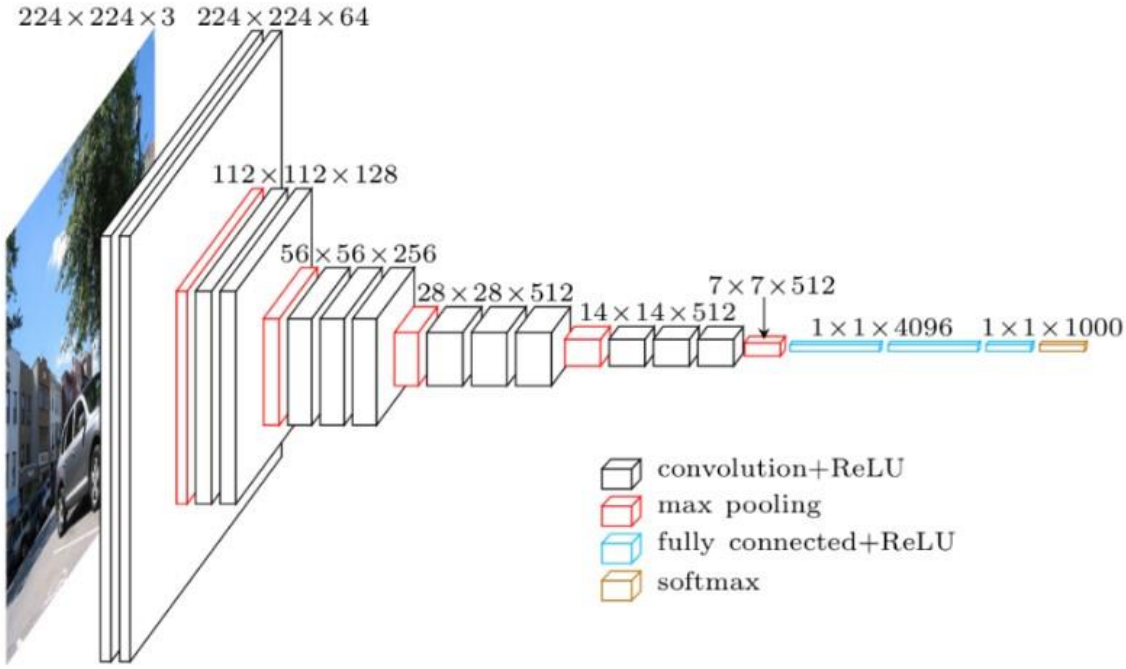
2. Vấn đề và giải pháp liên quan đến bài toán

2.1 Kiến trúc CNN – Convolutional Neural Network

Kiến trúc CNN được xây dựng từ các lớp Convolution chồng lên nhau và sử dụng các hàm nonlinear activation như ReLU để kích hoạt các trọng số trong các node. Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo. Các lớp này liên kết được với nhau thông qua cơ chế convolution. Lớp tiếp theo là kết quả convolution từ lớp trước đó, nhờ vậy mà ta có được các kết nối cục bộ.

Như vậy, có thể nói mỗi neuron ở lớp kế tiếp sinh ra từ kết quả của filter áp đặt lên một vùng ảnh cục bộ của neuron trước đó. Mỗi một lớp được sử dụng các filter khác nhau, thông thường có hàng trăm hàng nghìn filter và kết hợp kết quả của chúng lại. Ngoài ra có một số layer khác như pooling/subsampling layer dùng để chắt lọc lại các thông tin hữu ích hơn (loại bỏ các thông tin nhiễu).

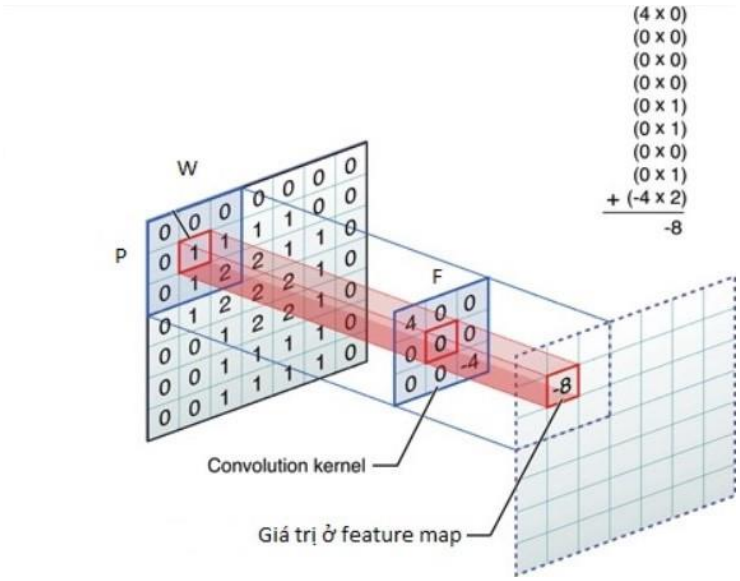
Trong quá trình huấn luyện mạng (traning), CNN tự động học các giá trị qua các lớp filter dựa vào cách thức mà chúng ta thực hiện.



Hình 3: *Một mô hình CNN*

2.1.1 Tầng tích chập (Convolution)

Tầng tích chập là tầng quan trọng nhất trong cấu trúc của CNN. Mục tiêu là thông qua việc lấy tích chập sẽ giúp trích xuất được những đặc trưng (thông tin quan trọng) từ dữ liệu hình ảnh.



Hình 4: *Quá trình tích chập một pixel với kernel 3x3*

Ảnh đầu là một ma trận số sẽ được bộ lọc (kernel) chạy quét qua từng pixel của toàn bộ bức ảnh. Bộ lọc có kích thước là 3x3 (hoặc 5x5) và áp dụng phép tích vô hướng để tính toán, cho ra một giá trị duy nhất của từng pixel trong mảng đầu ra. Đầu ra của phép tích chập là một tập các giá trị ảnh được gọi là mạng- bản đồ đặc trưng (features map).

Công thức tích chập giữa hàm ảnh f(x, y) và bộ lọc k(x, y) có kích thước mxm:

$$k(x,y) * f(x,y) = \sum_{u=-m/2}^{m/2} \sum_{v=-m/2}^{m/2} k(u,v).f(x-u,y-v)$$

Một số khái niệm trong tích chập:

Filter, Kernel hay **Feature Detector** đều là cách gọi của ma trận lọc. Thông thường, ở các lớp đầu tiên của Convolution kernel sẽ có kích thước là [5x5x3].

Convolved Feature, **Activation Map** hay **Feature Map** là đầu ra của ảnh khi cho bộ lọc chạy hết bức ảnh với phép tích vô hướng.

Receptive field là vùng ảnh được chọn để tính tích chập, hay bằng đúng kích thước của bộ lọc.

Depth là số lượng **bộ lọc. Lưu ý:** Depth không phải số lượng kênh màu RBG.

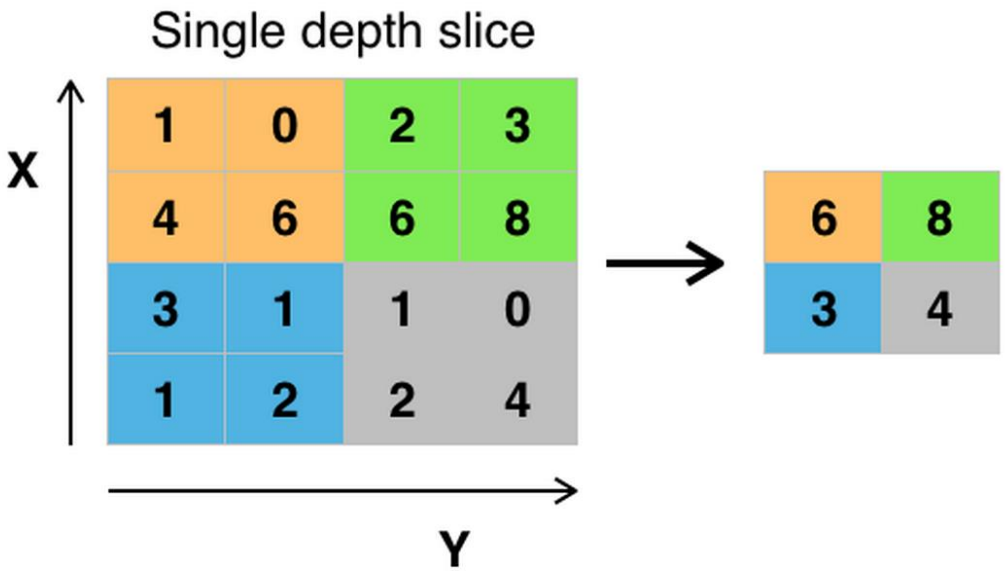
Stride được hiểu là khoảng cách dịch chuyển của bộ lọc sau mỗi lần tính. Ví dụ khi stride=2. Tức sau khi tính xong tại 1 vùng ảnh, nó sẽ dịch sang phải 2 pixel. Tương tự với việc dịch xuống dưới.

2.1.2 Tầng Pooling

Tầng Pooling (hay còn gọi subsampling hoặc downsample) là một trong những thành phần tính toán chính trong cấu trúc CNN. Xét về mặt toán học pooling thực chất là quá trình tính toán trên ma trận, mục tiêu sau khi tính toán là giảm kích thước ma trận nhưng vẫn làm nổi bật lên được đặc trưng có trong ma trận đầu vào. Trong CNN toán tử pooling được thực hiện độc lập trên mỗi kênh màu của ma trận ảnh đầu vào.

Có nhiều toán tử pooling như Sum-Pooling, Max-Pooling, L2-Pooling nhưng Max-Pooling thường được sử dụng. Max-Pooling sẽ giữ lại chi tiết quan trọng hay bằng cách giữ lại pixel có giá trị lớn nhất trong vùng thiết lập.

Ví dụ: Max-Pooling với bộ lọc 2x2 và stride = 2. Bộ lọc sẽ quét qua ảnh, với mỗi vùng ảnh được chọn, sẽ chọn ra 1 giá trị lớn nhất và giữ lại.

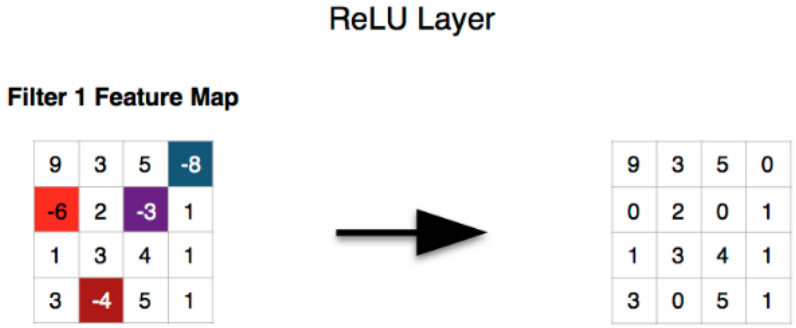


Hình 5: *Max-Pooling với bộ lọc 2x2, stride = 2. Nguồn: Wikipedia*

2.1.3 Tầng ReLU

ReLU layer áp dụng các kích hoạt (activation function) **max(0,x)** lên đầu ra của tầng tích chập, có tác dụng đưa các giá trị âm về thành 0. Tầng này không thay đổi kích thước của ảnh và không có thêm bất kì tham số nào.

Mục đích của lớp ReLu là đưa ảnh một mức ngưỡng, ở đây là 0. Để loại bỏ các giá trị âm không cần thiết mà có thể sẽ ảnh hưởng cho việc tính toán ở các layer sau đó.



Hình 6: *Tầng ReLU trong CNN*

2.1.4 Tầng Fully Connected (FC)

Sau các lớp Convolution và Pooling là lớp Fully connected, để kết nối tất cả các feature đã tìm được ở các lớp trước đó. Tại tầng này, mỗi một nơ-ron của tầng này sẽ liên kết tới mọi nơ-ron của tầng khác.

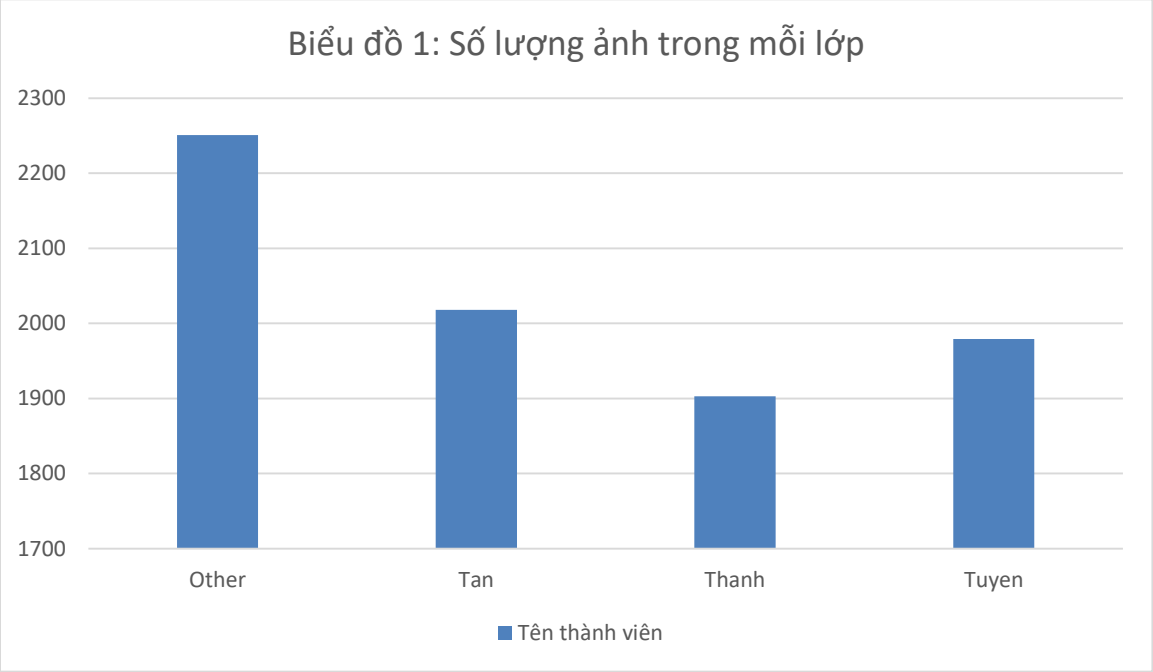
Để đưa ảnh từ các tầng trước vào mạng này, buộc phải dàn phẳng bức ảnh ra thành 1 vector thay vì là mảng nhiều chiều như trước. Tại tầng cuối cùng sẽ sử dụng 1 hàm trong học máy đó là *softmax* để phân loại đối tượng dựa vào vector đặc trưng đã được tính toán của các tầng trước đó.

2.2 Tập dữ liệu

Tập dữ liệu được thu thập với số lượng ảnh là 8151 bức ảnh của 4 đối tượng với các góc độ khuôn mặt và điều kiện sáng khác nhau. Với số lượng từng ảnh trong mỗi class được mô tả bằng bảng sau:

Other	Tan	Thanh	Tuyen	Tổng
2251	2018	1903	1979	8151

***Bảng 1**: Tổng quan về tập dữ liệu*



Tập dữ liệu được chia thành 2 phần: Thư mục train gồm 7351 ảnh để huấn luyện mô hình và thư mục validation gồm 800 ảnh dùng để xác thực.

2.3 Thư viện Keras

Keras là một thư viện của ngôn ngữ python được phát triển vào năm 2015 bởi Francois Chollet, là một kỹ sư nghiên cứu deep learning tại google. Nó là một open source cho neural network được viết bởi ngôn ngữ python. Keras là một API bậc cao có thể sử dụng chung với các thư viện deep learning nổi tiếng như tensorflow của Google, CNTK của Microsoft, theano của Yoshua Bengio

Cấu trúc dữ liệu chính của Keras là một model, cách bố trí các tầng (layer). Model đơn giản nhất là model ***Sequentail***, và dùng method add để thêm các layer.

2.3.1 Các phương thức:

- **Compile:** biên tập lại toàn bộ model.
- **Fit:** đưa data vào training.
- **Summary:** tổng hợp lại các thông tin của model như: số lượng layer, tổng tham số, shape.
- **Predict:** dự đoán nhãn của biến đầu vào.
- **Evaluate:** độ chính xác của model.

2.3.2 Các tầng (layer) trong Keras:

- **Core layer:** chứa các layer mà hầu như model nào cũng sử dụng đến nó.
    - **Dense layer** này sử dụng như một layer neural network bình thường. Các tham số quan tâm.
        - **units** chiều output.
        - **activation** dùng để chọn activation.
        - **input_dim** chiều input nếu là layer đầu tiên.
        - **kernel_regularizer** regularizer cho coeff.
        - **activity_regularizer** có sử dụng regularizer cho output không.
    - **Activation** dùng để chọn activation trong layer (có thể dùng tham số activation thay thế). Xem phần sau.
    - **Dropout layer** này dùng như regularization cho các layer hạn chế overfiting. Tham số cần chú ý :
        - **rate** tỉ lệ dropout.
    - **Flatten** dùng để lát phẳng layer để fully connection, vd : shape : 20x20 qua layer này sẽ là 400x1.
    - **Input layer** này sử dụng input như 1 layer như vd trước ta đã đề cập.
    - **Reshape** giống như tên gọi của nó, dùng để reshape.
    - **Lambda** dùng như lambda trong Python.
- **Convolutional Layers**: chứa các layer trong mạng nơ ron tích chập ha
    - **Conv1D**, **Conv2D** là convolution layer dùng để lấy feature từ image. tham số cần chú ý:
        - **filters** số filter của convolution layer.

        - **kernel_size** size window search trên image.
        - **strides** bước nhảy mỗi window search.
        - **padding same** là dùng padding, valid là không.
        - **data_format** format channel ở đầu hay cuối.
    - **UpSampling1D**, **UpSampling2D** Ngược lại với convolution layer.

    - **ZeroPadding1D**, **ZeroPadding2D** dùng để padding trên image.
        - **padding** số pixel padding.

- **Pooling Layers:** Chứa các layer dùng trong mạng CNN.

    - **MaxPooling1D**, **MaxPooling2D** dùng để lấy feature nổi bật (dùng max) và giúp giảm parameter khi training.

        - **pool_size** size pooling.

    - **AveragePooling1D**, **AveragePooling2D** giống như maxpooling nhưng dùng Average.

CHƯƠNG 2: THIẾT KẾ VÀ CÀI ĐẶT

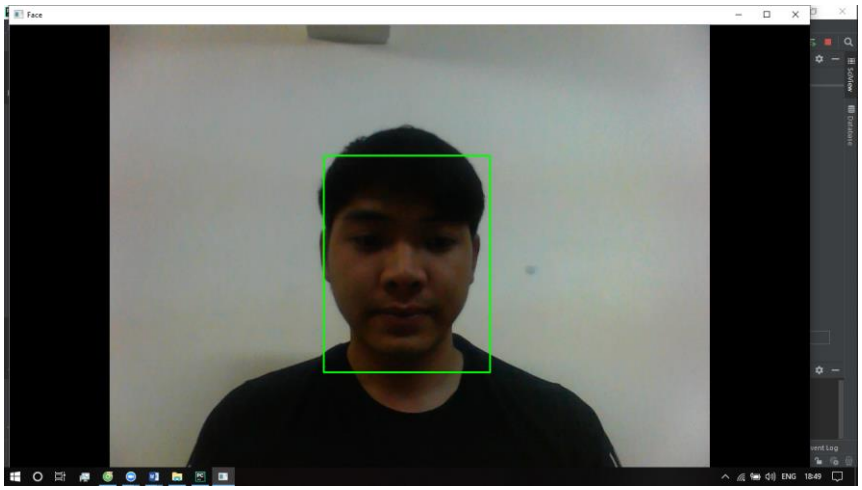
1. Thiết kế hệ thống

Hệ thống gồm 3 modul chính như sau:

Module 1: Thu thập dữ liệu hình ảnh thành viên

Dữ liệu được thu thập từ 2 nguồn chính là **dữ liệu từ camera** kết hợp với nguồn dữ liệu **Open images dataset v4**.

- Từ nguồn camera: Sau khi khởi chạy chương trình thu thập hình ảnh ***create_data.py***, chương trình sẽ yêu cầu nhập vào tên thành viên. Sau khi nhập tên sẽ tiến hành kiểm tra tên đã có tồn tại hay chưa, nếu chưa sẽ tạo một thư mục mới với tên vừa nhận để lưu ảnh. Sau khi đã có thư mục, thư viện OpenCV sẽ được gọi để mở camera và nhận các fram ảnh trả về. Với mỗi fram ảnh, tiến hành áp dụng model "*opencv_face_detection*" để dò tìm khuôn mặt trong ảnh và cắt hình ảnh khuôn mặt lưu vào thư mục với tên vừa nhập vào. Với mỗi một thành viên sẽ có một thư mục lưu ảnh riêng, và có thể mở rộng thêm ảnh.



***Hình 7**: Quá trình thu thập ảnh khuôn mặt thành viên*

- Ngoải ra, tập dữ liệu còn có môt thư mục Other nằm cùng cấp với các thư mục chứa ảnh của mỗi thành viên. Mục đích của thư mục này là chứa các ảnh khuôn mặt không phải là thành viên và cũng được coi như là một nhãn trong tập dự liệu. Các hình ảnh trong thư mục này được lấy từ một phần của tập dữ liệu **Open images dataset v4.**

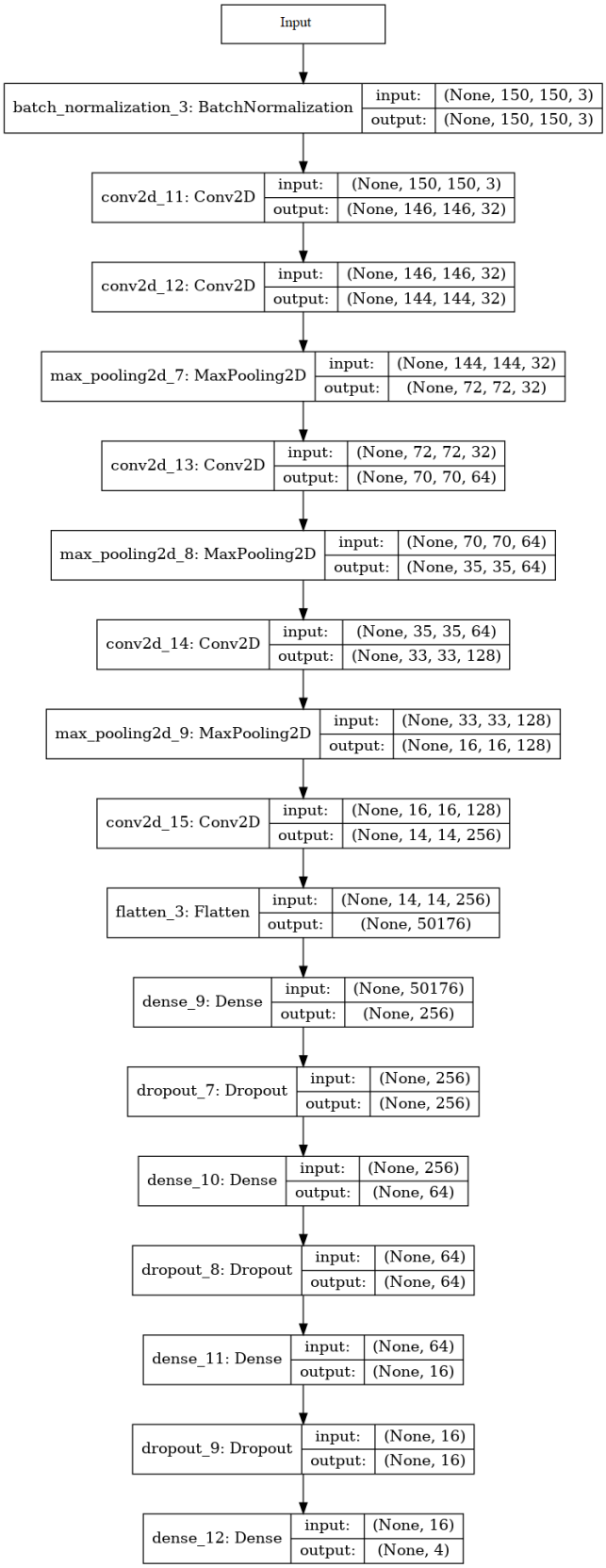
Do số lượng hình ảnh của mỗi thành viên thu được qua camera còn hạn chế, chương trình create_dataset.py được thiết kế để làm giàu số lượng hình ảnh của mỗi thành viên. Chương trình sử dụng thư viện **imageDataGenerator** trong thư viện keras để làm thay đổi góc độ hình ảnh: dịch sang trái, phải, nghiêng 25 độ, xoay ảnh… từ 1 tấm ảnh cho ra nhiều tấm ảnh.

***Hình 8**: Quá trình làm giàu dữ liệu ảnh*

Modul 2: Xây dựng mô hình và tiến hành Train tập dự liệu

Để mô hình hoạt động tốt, tất cả hình ảnh đầu vào sẽ được chuẩn hóa từ 0 – 255 về 0 – 1 và resize lại kích thước 150x150 với 3 kênh màu RGB.

Tiếp đến sẽ cho tập dữ liệu đi qua mô hình được thiết kế với 17 tầng, gồm có 1 tầng BatchNormalization, 5 tầng Conv2D, 3 tầng MaxPoling, 1 tầng Flatten, 3 tầng Dropout, 4 tầng Dense. Và thứ tự sắp xếp các tầng sẽ được biểu diễn ở **Hình 9**.



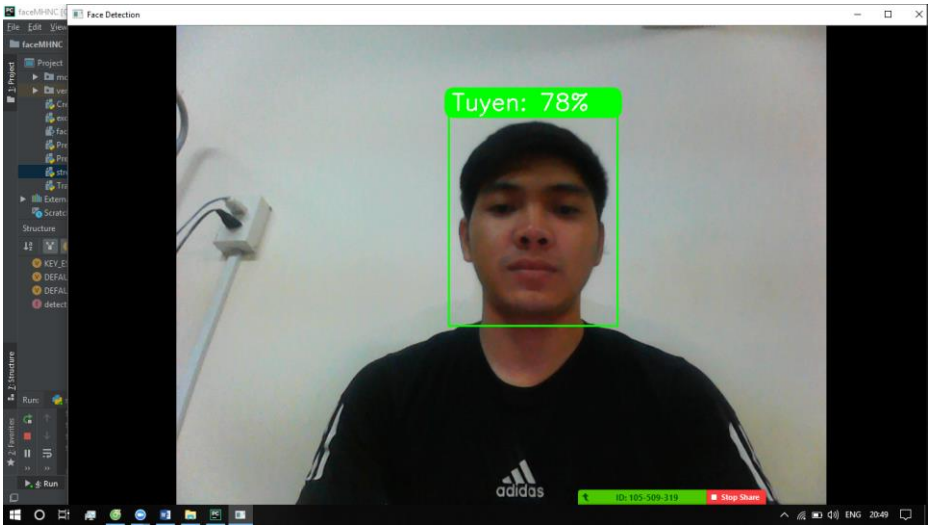
***Hình 9**: Mô hình CNN được thiết kế*

Sau khi toàn bộ tập dữ liệu đi qua mô hình CNN, chương trình sẽ lưu lại 2 file, một file mô tả hệ thống và một file lưu lại tất cả trọng số có được trong quá tình train dữ liệu. Và 2 file này sẽ được sử dụng để dự đoán trong modul thứ 3 của hệ thống.

Modul 3: Nhận diện thành viên nhóm.

Để tiến hành dự đoán, chương trình sẽ kích hoạt camera thông qua thư viện opencv, và trả về các khung hình. Tiếp đến sẽ áp dụng model "*opencv_face_detection*" để dò tìm khuôn mặt trong ảnh và cắt hình ảnh khuôn mặt. Từ hình ảnh khuôn mặt cắt được, sẽ tiến hành chuẩn hóa từ 0 – 255 về 0 – 1 và resize kích thước lại 150x150 với 3 kênh màu RGB.

Sau khi chuẩn hóa hình ảnh, thư viện Keras sẽ được gọi để load 2 file từ modul 2 và tiến hành dự đoán đưa ra nhãn là tên của thành viên hoặc không phải là thành viên sẽ trả về tên là Other.



***Hình 10**: Quá trình dự đoán*

2. Cài đặt giải thuật

Phần này sẽ để cập đến việc cài đặt và huấn luyện mô hình CNN để dự đoán. Từ tập dữ liệu ban đầu tiến hành chuẩn hóa và chia ra làm 2 phần: Tập train với 7351 ảnh khuôn mặt và tập test với 800 ảnh khuôn mặt.

```
# Define
batch_size = 16

train_generator = train_datagen.flow_from_directory(
    'dataset/train',
    target_size=(150, 150),
    batch_size=batch_size,
    class_mode='categorical')

# This is a similar generator, for validation data
validation_generator = test_datagen.flow_from_directory(
    'dataset/validation',
    target_size=(150, 150),
    batch_size=batch_size,
    class_mode='categorical')
```

***Hình 11**: Code chia dữ liệu*

Sau khi đã chia tập dữ liệu, tiến hành sử dụng các hàm trong thư viện Keras để xây dựng mô hình CNN cho tập dữ liệu.

```
# Build modal
model = Sequential()
model.add(BatchNormalization(input_shape=(150, 150, 3)))
model.add(Conv2D(filters=32, kernel_size=(5, 5), activation='relu'))
model.add(Conv2D(filters=32, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(256, (3, 3), activation='relu'))
# Full connect
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(16, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(4, activation='softmax'))
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

***Hình 12**: Cài đặt mô hình CNN*

Tiến hành huấn luyện 50 lần, ứng với mỗi lần ta đặt check point để lưu lại trọng số của model sau mỗi lần huấn luyện.

```
# Check point
cp_callback = keras.callbacks.ModelCheckpoint("./models/weights.{epoch:02d}.h5",
                                               save_weights_only=True,
                                               verbose=1)

history = model.fit_generator(
    train_generator,
    steps_per_epoch=7351 // batch_size,
    epochs=50,
    validation_data=validation_generator,
    validation_steps=800 // batch_size,
    callbacks=[cp_callback])
```

***Hình 13**: Code huấn luyện mô hình*

Và sau quá trình huyến luyện, sẽ lưu lại file mô tả mô hình CNN đã thiết kế để sử dụng cho việc dự đoán.

```
# Save model
model_json = model.to_json()
with open("models/face_model.json", "w") as json_file:
    json_file.write(model_json)
```

***Hình 14**: Code lưu lại file mô tả mô hình CNN*

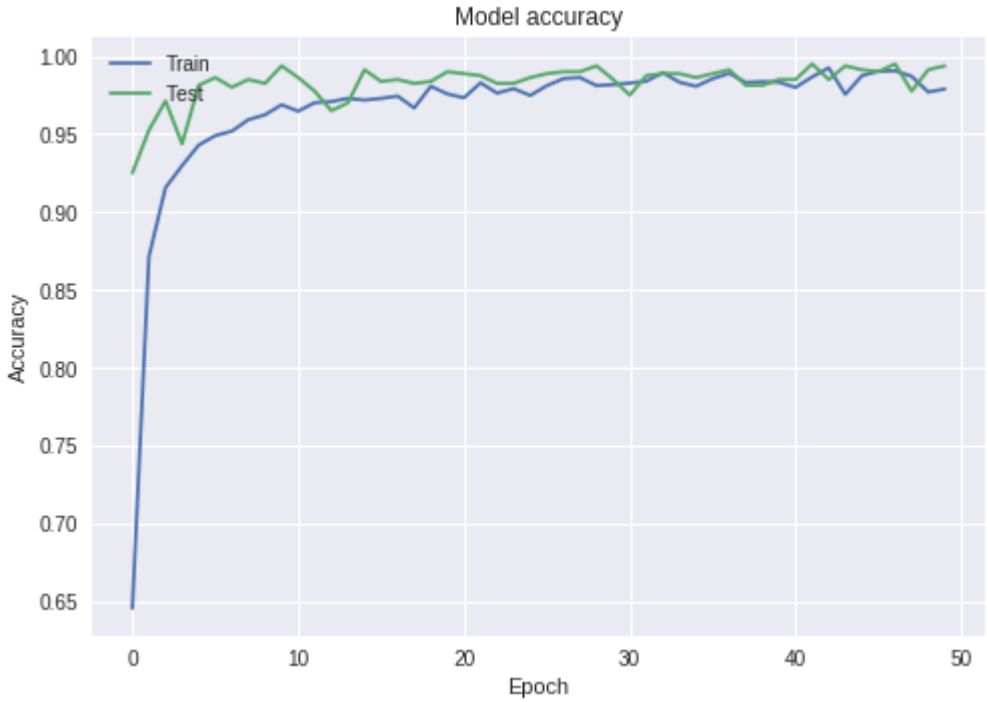
CHƯƠNG 3: KIỂM THỬ VÀ ĐÁNH GIÁ

1. Kết quả kiểm tra

- Kết thúc quá trình huấn luyện, mô hình cho tỷ lệ lỗi tương đối thấp và accuracy trên 98%
- Sau nhiều làn chạy thử nghiệm thì kết quả nhận diện cũng rất khả quan, với tỷ lệ nhận diện đúng thành viên trong nhóm khá cao. Và có thể nhận diện cùng lúc nhiều khuôn mặt trong ảnh.
- Và sau khi dùng tập dữ liệu test là validation, thì accuracy dao động từ 98% đến 99%.

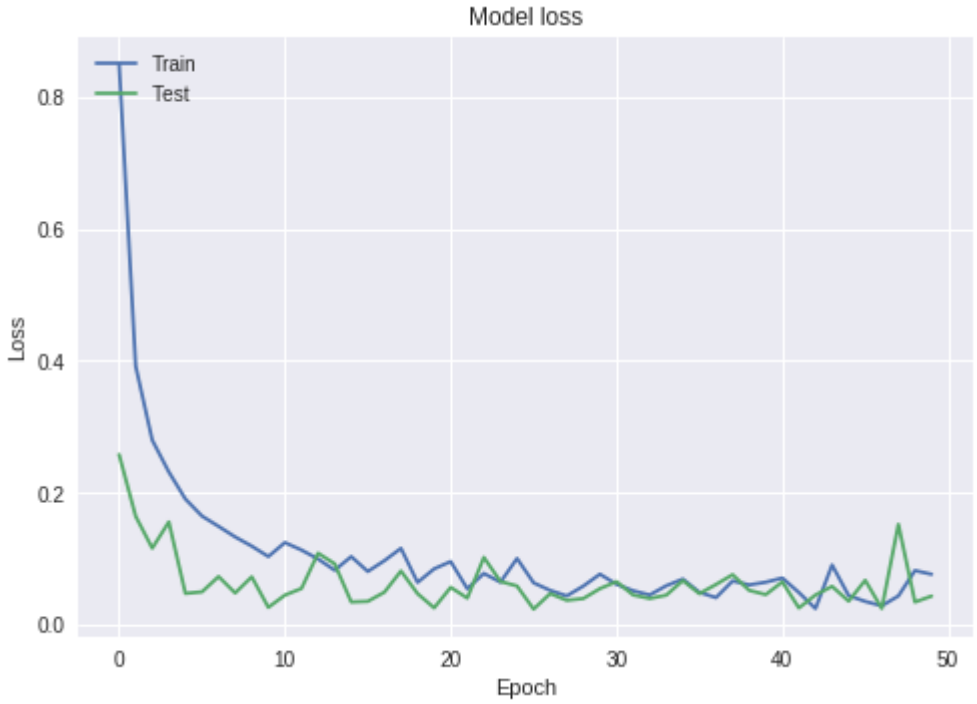
2. Đánh giá

Sau khi huấn luyện 50 lần, ta có được biểu đồ sau:



Biểu đồ 2: Accuracy trong 50 lần huấn luyện

Qua biểu đồ cho thấy mô hình CNN đã thiết kế phù hợp với tập dữ liệu, vì sau 5 lần huấn luyện thì accuracy trên 95%. Và từ lần huấn luyện thứ 25 trở đi, accuracy của train và test không còn thay đổi nhiều và tương đối cao trên 98%.



***Biểu đồ 3**: Loss trong 50 lần huấn luyện*

Tương tự với accuracy, thông qua giá trị loss cho thấy lỗi của mô hình tương đối thấp từ làm huấn luyện thứ 25 trở đi. Như vậy, có thể thấy rằng với tác trọng số từ lần lặp thứ 25 trở đi, ta có thể sự dụng cho quá trình dự đoán với tỷ lệ chính xác cao nhất.

Qua các lần chạy thử nghiệm, model chạy tương đối tốt ở điều kiện ánh sáng bình thường, cải thiện được tình trạng overfitting (học thuộc lòng của model). Song vẫn còn những khuyết điểm như: điều kiện ánh sáng không tốt, đeo kính mát, tóc dài hoặc hai người có khuôn mặt gần giống nhau với các hình ảnh độ phân giải thấp sẽ ảnh hưởng nhiều đến kết quả dự đoán.

PHẦN KẾT LUẬN

1. Kết quả đạt được

- ✓ Hiểu được mô hình hoạt động của CNN và áp dụng vào bài toán.
- ✓ Xây dựng được tập dữ liệu hình ảnh khuôn mặt của các thành viên trong nhóm.
- ✓ Thiết kế được model phù hợp với tập dữ liệu đã thu thập.
- ✓ Đánh giá được kết quả thông qua việc xác thực thành viên trong nhóm.

2. Hướng phát triển

- ✓ Mở rộng thêm tập dữ liệu với hình ảnh của nhiều thành viên trong nhóm.
- ✓ Cải thiện hiệu năng của model bằng cách tinh chỉnh những thông số.
- ✓ Áp dụng thêm các phương pháp xử lý ảnh vào ảnh input để nâng cao độ chính xác khi huấn luyện.
- ✓ Phát triển lên phần mềm điểm danh lớp học thông qua khuôn mặt.