**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**

**KHOA CÔNG NGHỆ THÔNG TIN**

A picture containing text, sign, vector graphics

Description automatically generated

BÁO CÁO BÀI TẬP LỚN

HỌC PHẦN: THỰC TẬP CHUYÊN NGÀNH KỸ THUẬT PHẦN MỀM

**ĐỀ TÀI: NGHIÊN CỨU, ỨNG DỤNG CNN ĐỂ PHÁT HIỆN**

**NGƯỜI KHÔNG ĐEO KHẨU TRANG**

|  |  |
| --- | --- |
| **Giảng viên hướng dẫn:** | **TS. Nguyễn Ngọc Quang** |
| **Mã lớp:** | **20222IT6041002** |
| **Nhóm:** | **18** |
| **Thành viên:** | **Nguyễn Tiến Duy – 2020601607**  **Đinh Tấn Hưng – 2020603858** |
| **Hà Nội – Năm 2023** | |

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**

**KHOA CÔNG NGHỆ THÔNG TIN**

A picture containing text, sign, vector graphics

Description automatically generated

BÁO CÁO BÀI TẬP LỚN

HỌC PHẦN: THỰC TẬP CHUYÊN NGÀNH KỸ THUẬT PHẦN MỀM

**ĐỀ TÀI: NGHIÊN CỨU, ỨNG DỤNG CNN ĐỂ PHÁT HIỆN**

**NGƯỜI KHÔNG ĐEO KHẨU TRANG**

|  |  |
| --- | --- |
| **Giảng viên hướng dẫn:** | **TS. Nguyễn Ngọc Quang** |
| **Mã lớp:** | **20222IT6041002** |
| **Nhóm:** | **18** |
| **Thành viên:** | **Nguyễn Tiến Duy – 2020601607**  **Đinh Tấn Hưng – 2020603858** |
| **Hà Nội – Năm 2023** | |

**MỤC LỤC**

[DANH MỤC CÁC KÝ HIỆU, CHỮ CÁI VIẾT TẮT i](#_Toc137542991)

[DANH MỤC HÌNH ẢNH ii](#_Toc137542992)

[DANH MỤC BẢNG BIỂU iv](#_Toc137542993)

[LỜI CẢM ƠN v](#_Toc137542994)

[MỞ ĐẦU 1](#_Toc137542995)

[CHƯƠNG 1: CƠ SỞ LÝ THUYẾT 3](#_Toc137542996)

[1.1 Tổng quan về Neural Networks 3](#_Toc137542997)

[1.1.1 Định nghĩa cơ bản 3](#_Toc137542998)

[1.1.2 Mô hình của một mạng Neuron 4](#_Toc137542999)

[1.1.3 Mô hình của một Neuron 4](#_Toc137543000)

[1.1.4 Một số loại hàm kích hoạt 5](#_Toc137543001)

[1.1.5 Các loại Neural Networks 7](#_Toc137543002)

[1.1.6 Mạng Perceptron 8](#_Toc137543003)

[1.1.7 Ứng dụng của Neural Networks 10](#_Toc137543004)

[1.2 Convolutional Neural Network (CNN) 12](#_Toc137543005)

[1.2.1 Các lớp cơ bản của một Convolutional Neural Network 12](#_Toc137543006)

[1.2.2 Cách Convolutional Neural Network hoạt động 16](#_Toc137543007)

[1.2.3 Phân loại CNN 18](#_Toc137543008)

[1.2.4 Một số mạng CNN nổi tiếng 19](#_Toc137543009)

[1.2.5 Sử dụng CNN vào bài toán nhận diện đối tượng 23](#_Toc137543010)

[1.3 Giới thiệu về Tensorflow và Keras 24](#_Toc137543011)

[1.3.1 Thư viện Tensorflow 24](#_Toc137543012)

[1.3.2 Framework Keras 25](#_Toc137543013)

[CHƯƠNG 2: XÂY DỰNG ỨNG DỤNG 27](#_Toc137543014)

[2.1 Lựa chọn thuật toán 27](#_Toc137543015)

[2.1.1 Tổng quan về ý tưởng giải quyết đề tài 27](#_Toc137543016)

[2.1.2 Trích xuất khuôn mặt 27](#_Toc137543017)

[2.1.3 Xây dựng mô hình (model) 29](#_Toc137543018)

[2.1.4 Kết hợp 30](#_Toc137543019)

[2.2 Các bước thực hiện 30](#_Toc137543020)

[2.2.1 Thu thập dữ liệu 30](#_Toc137543021)

[2.2.2 Xây dựng chương trình 31](#_Toc137543022)

[2.2.3 Kết quả đạt được 42](#_Toc137543023)

[2.2.4 Tổng kết, kiến nghị 46](#_Toc137543024)

[TÀI LIỆU THAM KHẢO 48](#_Toc137543025)

# DANH MỤC CÁC KÝ HIỆU, CHỮ CÁI VIẾT TẮT

|  |  |  |
| --- | --- | --- |
| **STT** | **Viết tắt** | **Dịch nghĩa** |
| 1 | CNN | Convolutional Neural Networ |
| 2 | ReLU | Rectified Linear Unit |
| 3 | ILSVRC | ImageNet Large Scale Visual Recognition Competition |
| 4 | AR | Augmented Reality |
| 5 | ORL | Olivetti Research Laboratory |

# DANH MỤC HÌNH ẢNH

[Hình 1.1 Ví dụ về mô hình của một Neuron 4](#_Toc137542296)

[Hình 1.2 Hàm Relu 5](#_Toc137542297)

[Hình 1.3 Hàm Sigmoid 6](#_Toc137542298)

[Hình 1.4 Hàm Softmax 6](#_Toc137542299)

[Hình 1.5 Feedforward netword với single-layer của neuron 7](#_Toc137542300)

[Hình 1.6 Fully connected feedforward multi-layer network với một layer ẩn và một layer đầu ra 8](#_Toc137542301)

[Hình 1.7 Rosenblatt’s Perceptron 9](#_Toc137542302)

[Hình 1.8 Biểu đồ luồng tín hiệu của một perceptron 10](#_Toc137542303)

[Hình 1.9 Mô tả phép tính tích chập convolution 12](#_Toc137542304)

[Hình 1.10 Ví dụ về các lớp convolution 14](#_Toc137542305)

[Hình 1.11 Pooling operation 15](#_Toc137542306)

[Hình 1.12 Fully Connected Layer 16](#_Toc137542307)

[Hình 1.13 Minh họa kiến trúc của VGG16 19](#_Toc137542308)

[Hình 1.14 Mô hình kiến trúc cơ bản của mạng RESNET 22](#_Toc137542309)

[Hình 1.15 Phần trăm training error và test error của mạng RESNET 22](#_Toc137542310)

[Hình 2.1 Tổng quan về dữ liệu 30](#_Toc137542311)

[Hình 2.2 Lấy dữ liệu 32](#_Toc137542312)

[Hình 2.3 Chuẩn bị kiến trúc VGG16 33](#_Toc137542313)

[Hình 2.4 Tiến hành huấn luyện mô hình 34](#_Toc137542314)

[Hình 2.5 Đồ thị loss trong quá trình huấn luyện 35](#_Toc137542315)

[Hình 2.6 Đồ thị accuracy trong quá trình huấn luyện 35](#_Toc137542316)

[Hình 2.7 Thử lại với 200 ảnh ngẫu nhiên trong tập private test 36](#_Toc137542317)

[Hình 2.8 Hình thu về kết quả không chính xác 37](#_Toc137542318)

[Hình 2.9 Một số hình ảnh mà model đưa ra kết quả không chính xác 38](#_Toc137542319)

[Hình 2.10 Thử nghiệm với camera 39](#_Toc137542320)

[Hình 2.11 Kết quả của mô hình ban đầu trong trường hợp sử dụng khẩu trang giả 40](#_Toc137542321)

[Hình 2.12 Kết quả của mô hình ban đầu trong trường hợp đeo khẩu trang không đúng cách 41](#_Toc137542322)

[Hình 2.13 Một số hình ảnh trong bộ dữ liệu cho trường hợp đeo khẩu trang không đúng cách 42](#_Toc137542323)

[Hình 2.14 Huấn luyện mô hình lần 2 42](#_Toc137542324)

[Hình 2.15 Đồ thị loss trong huấn luyện lần 2 43](#_Toc137542325)

[Hình 2.16 Đồ thị accuracy trong huấn luyện lần 2 43](#_Toc137542326)

[Hình 2.17 Camera - trường hợp không đeo khẩu trang 44](#_Toc137542327)

[Hình 2.18 Camera - trường hợp đeo khẩu trang 45](#_Toc137542328)

[Hình 2.19 Camera - trường hợp đeo khẩu trang không đúng cách 45](#_Toc137542329)

# DANH MỤC BẢNG BIỂU

[Bảng 1.1 Mô tả ký hiệu của biểu đồ luồng tín hiệu của một perceptron 10](#_Toc137542231)

[Bảng 2.1 Bảng so sánh các phương pháp trích xuất khuôn mặt 29](#_Toc137542232)

# LỜI CẢM ƠN

Để có được kết quả là bài báo cáo hoàn chỉnh, nhóm đã nhận được nhiều nhận xét cũng như góp ý kịp thời từ giảng viên TS. Nguyễn Ngọc Quang, cũng như các bạn trong Khoa Công nghệ thông tin Trường Đại học Công nghiệp Hà Nội. Nhóm bày tỏ lòng biết ơn đến các thành viên đã đóng góp và nỗ lực không ngừng trong quá trình thực hiện đề tài.

Nhóm rất biết ơn tất cả mọi người và hi vọng báo cáo có thể đóng góp ích lợi và cung cấp thông tin, kiến thức về lĩnh vực liên quan.

Nhóm xin chân thành cảm ơn!

# MỞ ĐẦU

1. **Tên đề tài**

Nghiên cứu, ứng dụng CNN để phát hiện người không đeo khẩu trang.

1. **Lý do chọn đề tài**

Ngày nay các kĩ thuật về mạng noron đã được nghiên cứu và phát triển rất rộng rãi. Ứng dụng của CNN vào các vấn đề như nhận diện đối tượng, phân loại đối tượng là rất tốt. Không những thế 2-3 năm gần đây dịch bệnh COVID-19 đã hoành hành và gây ra rất nhiều khó khăn cho toàn cầu. Cách tốt nhất để phòng tránh đại dịch này là thực hiện tốt quy tắc 5k của chính phủ đề ra. Từ đây việc đeo khẩu trang khi ra đường hay vào những nơi làm việc là một việc hết sức cần thiết. Chính vì thế việc phát hiện ra xem có người nào không đeo khẩu trang và tiến hành nhắc nhở là một việc cần được để tâm tới. Vì vậy, xuất phát từ nhu cầu thực tiễn, chúng em chọn đề tài là: “Nghiên cứu, ứng dụng CNN để phát hiện người không đeo khẩu trang”.

1. **Mục tiêu của đề tài**

* Nhận diện được mặt người có đeo khẩu trang hay không.
* Có thể tích hợp vào camera quan sát, đưa ra cảnh báo.

1. **Phương pháp nghiên cứu**

* Sử dụng bộ dữ liệu thu thập từ nhiều nguồn trên Internet.
* Sử dụng kiến thức đã nghiên cứu được để tiến hành viết chương trình, báo cáo.

1. **Đối tượng nghiên cứu**

* Thuật toán CNN áp dụng cho bài toán phân loại.
* Các thuật toán trích xuất gương mặt.
* Thư viện Tensorflow và Framework Keras.

1. **Phạm vi nghiên cứu**

* Dữ liệu là các khuôn mặt người được đánh nhãn tương ứng là đeo khẩu trang và không đeo khẩu trang.
* Thử nghiệm trên thành viên nhóm bằng chương trình chạy camera thời gian thực.

# CƠ SỞ LÝ THUYẾT

## Tổng quan về Neural Networks

### Định nghĩa cơ bản

Neural Networks là một hệ thống phức tạp, phi tuyến tính, song song, bao gồm các đơn vị xử lý nhỏ và đơn giản đơn được kết nối với nhau để có thể thực hiện các phép tính. Neural Networks có thể được sửa đổi bằng cách thay đổi độ mạnh yếu của từng kết nối giữa các đơn vị xử lý để đạt được đầu ra mong muốn thông qua một quá trình gọi là ‘Học’.

Neural Networks tự động trích xuất các tính năng từ các tập data training trong quá trình học. Với khả năng tổng quát hóa, Neural Networks được đào tạo để có thể phân loại được các dữ liệu đầu vào mới vào các lớp đầu ra đã được đào tạo. Có một số thuộc tính của Neural Networks: neural Networks có thể tuyến tính (linear) hoặc phi tuyến tính (nonlinearity). Tầm quan trọng của phi tuyến tính (nonlinearity) là khi dự kiến một Network sẽ tính toán đầu ra mong muốn từ những tín hiệu đầu vào phi tuyến tính như tín hiệu giọng nói. Một Neural Networks thường ánh xạ một tập hợp các đầu vào thành một tập hợp các đầu ra, trong đó một tập hợp các ví dụ huấn luyện bao gồm các tín hiệu đầu vào duy nhất được chỉ định và phản hồi ra một đầu ra mong muốn tương ứng. Một cách ngẫu nhiên, các ví dụ đào tạo được chọn từ một tập hợp và được cung cấp cho Mạng. Độ mạnh của các kết nối giữa các nút của mạng sau đó được sửa đổi để giảm thiểu sự khác biệt giữa phản hồi mong muốn và phản hồi thực tế được tạo ra bởi Mạng. Quá trình này được lặp lại cho đến khi chênh lệch giữa phản hồi mong muốn và phản hồi thực tế bằng 0 hoặc nhỏ nhất, do đó đạt được trạng thái ổn định cho mạng. Neural Networks có tính thích nghi, vì chúng có thể điều chỉnh trọng lượng khớp thần kinh với những thay đổi xung quanh.

### Mô hình của một mạng Neuron

Một đơn vị xử lý đơn giản được kết nói với các đơn vị khác của Neural Network được gọi là một “Neuron”. Neuron bao gồm ba yếu tố cơ bản.

* **Phần tử đầu tiên** là một tập hợp các khớp thần kinh hoặc các liên kết kêt nối, mỗi phần tử được đặc trưng bởi một trọng lượng (weight) hoặc sức mạnh (strength) của riêng nó. Đầu vào là một tập các liên kết kết nối được nhân với độ mạnh của kết nối đó. Trong trường hợp của một artificial neuron, phạm vi giá trị có thể bao gồm giá trị âm hoặc dương.
* **Phần tử thứ hai** là adder, một bộ kết hợp tuyến tính tính tổng mỗi tín hiệu đầu vào tương ứng nhân với cường độ của kết nối.
* **Phần tử thứ ba** là hàm kích hoạt (activation function), được sử dụng để giới hạn biên độ đầu ra của nơ-ron trong khoảng [0,1] hoặc [-1,1], tùy thuộc vào hàm kích hoạt.

### Mô hình của một Neuron

Hình ảnh dưới đây là một ví dụ về neuron.

Diagram

Description automatically generated

Hình 1.1 Ví dụ về mô hình của một Neuron

Chúng ta có thể thấy một neuron cơ bản sẽ có 3 lớp cơ bản là Input layer, Hidden layer và Ouput layer.

### Một số loại hàm kích hoạt

Hàm kích hoạt (Activation function) là những hàm phi tuyến tính được áp dụng vào đầu ra của các neuron trong tầng ẩn (Hidden layer) của một mô hình mạng (Network), đầu ra được chuyển đổi này được sử dụng làm đầu vào của neuron ở layer tiếp theo.

Một số Activation Function thường gặp là:

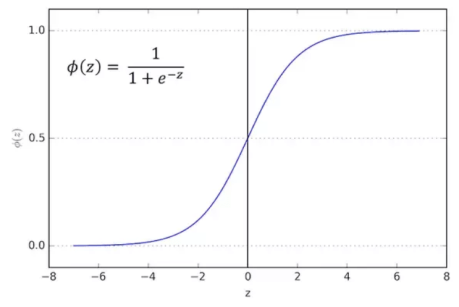
* ReLU: Hàm ReLU cho đầu ra bằng đầu vào nếu đầu vào lớn hơn 0 và bằng 0 nếu đầu vào nhỏ hơn hoặc bằng 0. Hàm ReLU thường được sử dụng trong các lớp nơ-ron ẩn của mạng nơ-ron và có thể giúp tăng tốc độ huấn luyện.

A picture containing line, diagram, text, plot

Description automatically generated

Hình 1.2 Hàm Relu

* Sigmoid: Hàm sigmoid chuyển đổi đầu vào thành một giá trị trong khoảng từ 0 đến 1. Hàm sigmoid được sử dụng rộng rãi trong các lớp nơ-ron đầu ra và trong các mô hình phân loại nhị phân.



Hình 1.3 Hàm Sigmoid

* Softmax: Hàm softmax thường được sử dụng trong lớp nơ-ron đầu ra của mạng nơ-ron phân loại. Nó chuyển đổi đầu vào thành một phân phối xác suất trên các lớp đầu ra, giúp tạo ra các dự đoán xác suất cho từng lớp.

Diagram

Description automatically generated

Hình 1.4 Hàm Softmax

### Các loại Neural Networks

Có 2 loại Neural Networks là Single-layer feedforward network và Multilayer feedforward network.

* **Single-layer feedforward network**

Một Single-layer network là một loại mạng feedforward network (mạng truyền thẳng) chiếu các đầu vào từ lớp của các nút nguồn trên lớp đầu ra của các neuron (computation nodes). “Single layer” là một phép tính được thực hiện bởi một lớp duy nhất, là lớp đầu ra. Không có tính toán nào được thực hiện bởi lớp đầu vào của các nút nguồn.

Diagram

Description automatically generated

Hình 1.5 Feedforward netword với single-layer của neuron

* **Multilayer feedforward network**

Một Multilayer network là một feedforward netword với một layer của các nút nguồn, một hoặc nhiều layer ẩn của lớp neurons và lớp đầu ra của các reuron (Hình 1.3). Cả lớp ẩn và lớp đầu ra đều bao gồm các nút tính toán. Thuật ngữ lớp “Ẩn” đề cập đến thực tế là lớp như vậy không được nhìn thấy trực tiếp từ đầu ra hoặc đầu vào của mạng và chúng nhận đầu vào từ các nút nguồn hoặc các nút của các lớp ẩn trước đó và kết quả đầu ra của mạng.

Diagram

Description automatically generated

Hình 1.6 Fully connected feedforward multi-layer network với một layer ẩn và một layer đầu ra

### Mạng Perceptron

Mạng Perceptron là dạng đơn giản nhất của một Neural Network bao gồm một neuron duy nhất có trọng số liên kết (synaptic weights) và độ lệch (bias) có thể điều chỉnh được. Một neuron riêng lẻ được sử dụng nghiêm ngặt như một bộ phân loại cho hai lớp của các mẫu có thể phân tách tuyến tính, nằm trên các mặt đối diện của một siêu phẳng. Trong *Hình 1.7*, một mẫu có thể phân tách tuyến tính của hai lớp được phân loại bằng cách sử dụng một Rosenblatt’s Perceptron.

Chart, scatter chart

Description automatically generated

Hình 1.7 Rosenblatt’s Perceptron

Cách mà một Perceptron được sử dụng cho bài toán phân loại hai lớp là lấy các tín hiệu đầu vào khác nhau 𝑥𝑚 và phân loại chúng thành một trong hai lớp, C1 hoặc C2, tùy thuộc vào giá trị đầu ra 𝑦 của nơ-ron, cho dù đó là +1 hay -1 (*Hình 3.2*). Kết quả phân loại mẫu có thể được vẽ trên một mặt phẳng hai chiều để hiểu rõ hơn về hoạt động của perceptron . Hai lớp trên mặt phẳng hai chiều được ngăn cách bởi một ranh giới quyết định, đó là một siêu không gian; đường phân cách hai lớp với nhau.

Diagram

Description automatically generated

Hình 1.8 Biểu đồ luồng tín hiệu của một perceptron

|  |  |
| --- | --- |
| **Ký Hiệu** | **Mô tả** |
|  | Tín hiệu đầu vào cho nơ-ron |
|  | synaptic weights của rơ-ron |
| *b* | bias |
| *v* | Trường cục bộ gây ra hoặc activation potential của rơ-ron |
|  | Activation potential |
| *y* | Tín hiệu đầu ra của rơ-ron |

Bảng 1.1 Mô tả ký hiệu của biểu đồ luồng tín hiệu của một perceptron

### Ứng dụng của Neural Networks

Phân loại mạng nơ-ron, hoặc các ứng dụng dự báo đã được triển khai rộng rãi trong các lính vực công nghiệp, kinh doanh và khoa học với nhiều ứng dụng khác nhau do các đặc tính của mạng nơ-ron là học và tổng quát hóa, bất kể thuật toán là gì, cho dù là backpropagation, radial basis functions, genetic hoặc Kohonen’s learning vector quantization. Mạng rơ-ron đã đáp ứng nhu cầu đa dạng theo ba loại: phân loại mẫu, dự đoán và tối ưu hóa.

#### Phân loại mẫu

Mạng rơ-ron nhân tạo có thể xử lý nhiều tín hiệu đầu vào và suy ra các mối quan hệ phi tuyến tính, chúng được sử dụng để xử lý hình ảnh và nhận dạng các ký tự. Các ứng dụng xử lý hình ảnh bao gồm một loạt các ngành công nghiệp, nhận dạng hình ảnh cho điện thoại thông minh hoặc để chống lại tội phạm bằng cách phát hiện tội phạm bị truy nã từ cộng đồng, cũng như xử lý hình ảnh vệ tinh cho mục đích nông nghiệp để phân biệt các thuộc tính khác nhau của các loại cây trồng khác nhau trên cùng một cánh đồng rộng lớn. Ứng dụng nhận dạng ký tự phù hợp hơn với các ngành ngân hàng và bảo mật, nơi chữ viết tay trên séc có thể được nhận dạng và xử lý thành các số được số hóa. Chữ ký viết tay hợp pháp cũng có thể được phân biệt với những chữ ký giả mạo.

#### Dự đoán

Với khả năng mô hình hóa các mối quan hệ phi tuyến tính mà không thể xác định từ các dữ liệu đầu vào khác nhau, mạng nơ-ron có khả năng dự báo các xu hướng dữ liệu trong tương lai. Có rất nhiều trường hợp sử dụng để dự báo mạng nơ-ron nhân tạo trong các lĩnh vực bán hàng, phân bổ tài chính giữa các sản phẩm, thị trường chứng khoán. Trong phân tích tiếp thị, mạng nơ-ron đã được sử dụng để cắt giảm chi phí của các chiến dịch tiếp thị bằng cách dự báo những khách hàng chưa có khả năng từ những khách hàng tiềm năng dựa trên tập mẫu của tập dữ liệu đã được thu thập từ họ.

#### Tối ưu hóa

Các phương pháp tối ưu hóa được sử dụng để tối đa hóa hoặc giảm thiểu các chức năng, mục tiêu nhất định của hệ thống phi tuyến tính. Mạng nơ-ron nhân tạo giúp tính gần đúng các hàm mục tiêu, do đó cho phép sử dụng các kĩ thuật phù hợp để phát triển các phương trình đa thức nhằm tính toán lời giải cần thiết để đạt được sự tối ưu hóa của hệ thống.

## Convolutional Neural Network (CNN)

### Các lớp cơ bản của một Convolutional Neural Network

#### Convolutional layer

* **Phép tính convolution.**

Ta giả sử ma trận cần tình convolution là ma trận X có kích thước n×m. Và 1 ma trận k có kích thước là x×x. Kí hiệu phép tính convolution (⊗), kí hiệu Y = X ⊗ W. Với mỗi phần tử xij trong ma trận X lấy ra một ma trận có kích thước bằng kích thước của kernel W có phần tử xij làm trung tâm (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 element-wise của ma trận A và ma trận W, rồi viết vào ma trận kết quả Y.

A diagram of a house

Description automatically generated with low confidence

Hình 1.9 Mô tả phép tính tích chập convolution

Dễ thấy với phép tính như trên thì shape của ma trân Y sẽ nhỏ hơn shape (kích thước) của ma trận X đầu vào. Với những trường hợp cần ma trận Y có cùng khích thương với ma trận X chúng ta thêm 1 hệ số được gọi là padding vào ma trận X rồi thực hiện phép tính convolution như bình thường.

* **Lớp tích chập (Convolution)**

Tích chập là lớp đầu tiên để trích xuất các tính năng từ hình ảnh đầu vào. Tích chập duy trì mối quan hệ giữa các pixel bằng cách tìm hiểu các tính năng hình ảnh bằng cách sử dụng các ô vương nhỏ của dữ liệu đầu vào. Nó là 1 phép toán có 2 đầu vào như ma trận hình ảnh và 1 bộ lọc hoặc hạt nhân.

Giả sử input của 1 convolutional layer tổng quát là tensor kích thước H × W × D. Kernel có kích thước F × F × D (kernel luôn có depth bằng depth của input và F thường là số lẻ vì ô vuông lưới chẵn × chẵn thì sẽ không có 1 ô vuông ở tâm đối xứng dẫn tới giảm độ chính xác), stride: S, padding: P. Convolutional layer áp dụng K kernel.

Output sẽ có kích thước:

Ta cũng có thể trồng nhiều lớp convolution lên nhau để lấy được đặc trưng của ảnh. Trước khi output của lớp convolution trước làm input của lớp sau thì ta đưa qua 1 hàm phi tuyến tính.

* **Ý nghĩa của lớp Convolution**

Convolution sẽ giúp làm mờ, làm nét ảnh. Lấy được các đặc trưng của ảnh. Mỗi kernel (ma trận tích chập) khác nhau sẽ đều có những tác dụng khác nhau.

A screenshot of a phone

Description automatically generated with low confidence

Hình 1.10 Ví dụ về các lớp convolution

#### Lớp Pooling

Giống như convolution layer, dữ liệu đầu vào được convoluted với một bộ lọc để tạo thành một ma trận convolution.

Ví dụ trong một Pooling layer một cụm 3×3 được lấy từ dữ liệu đầu và 5×5. Điều này sẽ được miêu tả ở *Hình 1.11*, có hai hàm pooling, average hoặc max còn tùy thuộc vào hàm. Giá trị trung bình của tất cả các phần tử sau đó được đặt vào C11 hoặc giá trị tối đa của cụm được đặt vào C11. Quá trình này được lặp đi lặp lại tùy thuộc vào chức năng, từ phần tử C11 cho đến phần tử C33 cho pooling 3×3.

Diagram

Description automatically generated

Hình 1.11 Pooling operation

Pooling layer thường được dùng giữa các convolutional layer, để giảm kích thước dữ liệu nhưng vẫn giữ được các thuộc tính quan trọng. Việc giảm kích thước dữ liệu giúp giảm các phép tính toán trong model. Bên cạnh đó, với phép pooling kích thước ảnh giảm, do đó lớp convolution học được các vùng có kích thước lớn hơn.

#### Lớp Fully Connected

Bằng nhiều convolution và pooling layer, đầu ra được nhận đến một fully connected layer dưới dạng đầu vào như trong *Hình 1.12*. Một fully connected layer thường được đặt ở phần cuối trong cấu trúc CNN, hoặc ở đâu đó trung gian. Loại layer này rất giống với đa lớp truyền tiếp perceptron (feedforward multilayer perceptron). Trong layer này, tất cả thông tin được lọc, lấy mẫu được thu thập để CNN bắt đầu học. Hầu hết các trọng số cũng nằm trong phần này của mạng.

Diagram

Description automatically generated

Hình 1.12 Fully Connected Layer

### Cách Convolutional Neural Network hoạt động

Đối với bất kì nhiệm vụ nào được yêu cầu thực hiện bởi CNN, các thông số cần phải được điều chỉnh tương tưởng với mỗi nhiệm vụ cần thực thi. Các cơ chế và kĩ thuật khác nhau như khởi tạo trọng số và điều chỉnh mạng được thực hiện để tối ưu hóa mạng Neuron.

#### Khởi tạo Weight

Để huấn luyện CNN một cách ổn định, cần khởi tạo một trọng số chính xác, nếu không đều này sẽ dẫn đến những khó khăn khác nhau trong quá trình học, tùy vào phương pháp học được sử dụng. Ví dụ, trong trường hợp sử dụng phương pháp back-propagation (lan truyền ngược), việc khởi tạo trọng số không chính xác sẽ dẫn đến việc biến mất hoặc bùng nổ của vấn đề gradient (Vanishing and Exploding Gradient). Có kỹ thuật này là khởi tạo Gaussian Random và khởi tạo Uniform Random.

* **Khởi tạo Gaussian Random**

Kĩ thuật khởi tạo này được sử dụng cho các convolution và fully connected layers của CNN với ma trận ngẫu nhiên của các phần tử được lấy mẫu từ Gaussian với giá trị trung bình bằng 0 và độ lệch nhỏ.

* **Khởi tạo Uniform Random**

Kĩ thuật khởi tạo này được sử dụng cho các convulution và fully connected layers của CNN với ma trận ngẫu nhiên của các phần tử được lấy mẫu từ Uniform với giá trị trung bình bằng 0 và độ lệch nhỏ.

#### Quy định Network

CNN được mong đợi sẽ học và khái quát hóa; tuy nhiên, trong một số trường hợp các mạng không thể học và tổng quát hóa vì chúng có xu hướng phù hợp quá mức với dữ liệu đào tạo trong quá trình học. Mạng được kỳ vọng sẽ là học hỏi thay vì ghi nhớ để có thể khái quát hóa và thích ứng tốt hơn với dữ liệu không nhìn thấy được sử dụng để ngăn chặn over-fitting. Data augmentation, dropout, batch normalization, ensemble model averaging, và early stopping là những ví dụ về kỹ thuật chính quy hóa.

CNN có thể phân loại dữ liệu ngẫu nhiên thành các lớp (class) khác nhau. Dữ liệu được phân loại dưa trên các tính năng, đặc điểm và phẩm chất được chia sẻ với các bộ dữ liệu khác trong các danh mục tương ứng. Khi so sánh với các thuật toán phân loại khác, CNN dễ huấn luyện hơn vì nó yêu cầu ít kết nối và tham số hơn, như đã được chứng minh bằng kết quả thu được từ ILSVRC. Trong cuộc thi này deep CNN đã đạt được top 1 về hiệu suất, hoạt động tốt hơn đáng kể so với các thuật toán được sử dụng trước đây. Deep CNN đã phân loại 1,2 triệu hình ảnh có độ phân giải cao trong cuộc thi thành 1000 lớp khác nhau. CNN được sử dụng trong các ứng dụng khác nhau, phân loại các phạm vi dữ liệu khác nhau từ hình ảnh đến văn bản, v…v…. Một số bộ phân loại CNN hiện đại sẽ được nêu ra trong phần này.

### Phân loại CNN

#### Nhận diện khuôn mặt dựa trên CNN (Face recognition)

Trong W. L.-m. Hu Yu-lu, "Hand gesture recognition based on convolutional neural networks," in LIDAR Imaging Detection and Target Recognition 2017, Changchun, 2017, một phương pháp nhận dạng khuyên mặt dựa trên CNN được đề xuất, bao gồm 9 layer, 3 trong số đó là convolution layers, 2 là pooling layers, 2 là fully connected layers và SoftMax regression layer. Mạng này đã được thử nghiệm trên Caffe, một framework Deep Learning có khả năng xử lý hơn 60 triệu hình ảnh mỗi ngày. Framework Caffe đã đào tạo và thử nghiệm hai cơ sở dữ liệu khuôn mặt khác nhau. Cơ sở dữ liệu khuôn mặt ORL bao gồm tổng cộng 400 hình ảnh cho 40 người. Cơ sở dữ liệu khuôn mặt AR chứa tổng cộng 2600 hình ảnh cho 100 người. Sau khoảng 200 lặp lại, mạng đạt tỉ lệ chính xác là 90%. Sau 2000 lần lặp, đạt 97%. Cuối cùng, sau 10.000 lần lặp, tỷ lệ nhận dạng cho tất cả dữ liệu khuôn mặt của AR và ORL đạt độ chính xác lần lượt là 99, 82% và 99,78%. Khi so sánh với các phương pháp khác được sử dụng cho cơ sở dữ liệu AR và cơ sở dữ liệu ORL, CNN được sử dụng đứng đầu vì độ chính xác và mạnh mẽ.

#### Nhận dạng cử chỉ tay dựa trên CNN (Hand gesture recognition)

Trong W. L.-m. Hu Yu-lu, "Hand gesture recognition based on convolutional neural networks," in LIDAR Imaging Detection and Target Recognition 2017, Changchun, 2017. Một thuật toán nhận dạng cử chỉ tay dựa trên ConvNet được đề cử để nhận dạng 10 cử chỉ. Mạng đã được đào tạo bằng cách sử dụng 6000 hình ảnh cử chỉ tay và 1100 mẫu thử nghiệm. Mạng đã đạt tỉ lệ nhận dạng là 98%, đứng đầu các thuật toán nhận dạng khác.

### Một số mạng CNN nổi tiếng

#### VGG16

VGG16 là mạng CNN được đề xuất bởi K. Simonyan and A. Zisserman, University of Oxford. Mẫu sau khi huấn luyện bởi mạng VGG 16 đạt độ chính xác 92.7% – top 5 test trong dữ liệu ImageNet (<http://www.image-net.org/>) gồm 14 triệu hình ảnh thuộc 1000 lớp khác nhau. Giờ ta áp dụng kiến thức đã nêu trước đó để phân tích mạng VGG16.

**A picture containing text, screenshot, diagram, line

Description automatically generated**

Hình 1.13 Minh họa kiến trúc của VGG16

Phân tích về VGG 16:

* Convolutional layer: kích thước 3×3, padding=1, stride=1. Mắc định stride=1 và padding để cho output cùng width và height với input.
* Pool/2 : max pooling layer với size 2×2
* 3×3 conv, 64: thì 64 là số kernel áp dụng trong layer đấy, hay depth của output của layer đấy.
* Càng các convolutional layer sau thì kích thước width, height càng giảm nhưng depth càng tăng.
* Sau khá nhiều convolutional layer và pooling layer thì dữ liệu được flatten và cho vào fully connected layer.

#### ResNet

ResNet (Residual Network) được giới thiệu đến công chúng vào năm 2015 và thậm chí đã giành được vị trí thứ 1 trong cuộc thi ILSVRC 2015 với tỉ lệ lỗi top 5 chỉ 3.57%. Không những thế nó còn đứng vị trí đầu tiên trong cuộc thi ILSVRC and COCO 2015 với ImageNet Detection, ImageNet localization, Coco detection và Coco segmentation. Hiện tại thì có rất nhiều biến thể của kiến trúc ResNet với số lớp khác nhau như ResNet-18, ResNet-34, ResNet-50, ResNet-101, ResNet-152,...Với tên là ResNet theo sau là một số chỉ kiến trúc ResNet với số lớp nhất định.

* **Lợi thế của RESNET**

Mạng ResNet (R) là một mạng CNN được thiết kế để làm việc với hàng trăm hoặc hàng nghìn lớp chập. Một vấn đề xảy ra khi xây dựng mạng CNN với nhiều lớp chập sẽ xảy ra hiện tượng Vanishing Gradient dẫn tới quá trình học tập không tốt.

Backpropagation Algorithm (thuật toán lan truyền ngược) là một kĩ thuật thường được sử dụng trong trong quá trình training DNNs. Ý tưởng chung của thuật toán là sẽ đi từ output layer đến input layer và tính toán gradient của cost function tương ứng cho từng parameter (weight) của network. Gradient Descent, sau đó, sẽ được sử dụng để cập nhật các parameter đó.

Quá trình trên sẽ được lặp lại cho tới khi các parameter của network hội tụ. Thông thường chúng ta sẽ có một hyperparameter định nghĩa cho số lượng vòng lặp để thực hiện quá trình trên. Hyperparameter đó thường được gọi là số Epoch (hay số lần mà training set được duyệt qua một lần và weights được cập nhật). Nếu số lượng vòng lặp quá nhỏ, DNN có thể sẽ không cho ra kết quả tốt, và ngược lại thì thời gian training sẽ quá dài nếu số lượng vòng lặp quá lớn. Ở đây ta có một tradeoff giữa độ chính xác và thời gian training.

Tuy nhiên trên thực tế gradients thường sẽ có giá trị nhỏ dần khi đi xuống các layer thấp hơn. Kết quả là các cập nhật thực hiện bởi Gradient Descent không làm thay đổi nhiều weights của các layer đó, khiến chúng không thể hội tụ và DNN sẽ không thu được kết quả tốt. Hiện tượng này được gọi là Vanishing Gradients => ResNet ra đời để giải quyết vấn đề này.

* **Kiến trúc của mạng RESNET**

Giải pháp mà ResNet đưa ra là sử dụng kết nối "tắt" đồng nhất để xuyên qua một hay nhiều lớp. Một khối như vậy được gọi là một Residual Block, như trong *Hình 1.14*:

A picture containing text, diagram, font, line

Description automatically generated

Hình 1.14 Mô hình kiến trúc cơ bản của mạng RESNET

ResNet gần như tương tự với các mạng gồm có convolution, pooling, activation và fully-connected layer. Ảnh bên trên hiển thị khối dư được sử dụng trong mạng. Xuất hiện một mũi tên cong xuất phát từ đầu và kết thúc tại cuối khối dư. Hay nói cách khác là sẽ bổ sung Input X vào đầu ra của layer, hay chính là phép cộng mà ta thấy trong hình minh họa, việc này sẽ chống lại việc đạo hàm bằng 0, do vẫn còn cộng thêm X. Với H(x) là giá trị dự đoán, F(x) là giá trị thật (nhãn), chúng ta muốn H(x) bằng hoặc xấp xỉ F(x).

Theo như nghiên cứu [1] Resnet có thể sử dụng được một số lượng layer rất lớn mà không xảy ra hiện tượng mất đạo hàm (Vanishing Gradients).

A picture containing text, line, diagram, font

Description automatically generated

Hình 1.15 Phần trăm training error và test error của mạng RESNET

### Sử dụng CNN vào bài toán nhận diện đối tượng

Một trong những lĩnh vực quan trọng của Trí tuệ nhân tạo (Artificial Intelligence) là thị giác máy (Computer Vision). Computer Vision là một lĩnh vực bao gồm các phương pháp thu nhận, xử lý ảnh kỹ thuật số, phân tích và nhận dạng các hình ảnh, phát hiện các đối tượng, tạo ảnh, siêu phân giải hình ảnh và nhiều hơn vậy. Object Detection có lẽ là khía cạnh sâu sắc nhất của thị giác máy do số lần sử dụng trong thực tế.

Object Detection đề cập đến khả năng của hệ thống máy tính và phần mềm để định vị các đối tượng trong một hình ảnh và xác định từng đối tượng. Object Detection đã được sử dụng rộng rãi để phát hiện khuôn mặt, phát hiện xe, đếm số người đi bộ, hệ thống bảo mật và xe không người lái. Có nhiều cách để nhận diện đối tượng có thể được sử dụng cũng như trong nhiều lĩnh vực thực hành. Giống như mọi công nghệ khác, một loạt các ứng dụng sáng tạo và tuyệt vời của Object Detection sẽ đến từ các lập trình viên và các nhà phát triển phần mềm.

Bắt đầu sử dụng các phương pháp nhận diện đối tượng hiện đại trong các ứng dụng và hệ thống, cũng như xây dựng các ứng dụng mới dựa trên các phương pháp này. Việc triển nhận diện đối tượng sớm liên quan đến việc sử dụng các thuật toán cổ điển, giống như các thuật toán được hỗ trợ trong OpenCV, thư viện computer vision phổ biến. Tuy nhiên, các thuật toán cổ điển này không thể đạt được hiệu suất đủ để làm việc trong các điều kiện khác nhau.

Việc áp dụng đột phát và nhanh cóng của deep learning vào năm 2012 đã đưa vào sự tồn tại các thuật toán và phương pháp phát hiện đối tượng hiện đại và chính xác cao như R-CNN, Fast-RCNN, Faster-RCNN, RetinaNet và nhanh hơn nhưng rất chính xác như SSD và YOLO. Sử dụng các phương pháp và thuật toán này, dựa trên deep learning và cũng dựa trên việc học máy đòi hỏi rất nhiều kiến thức về toán học và việc học sâu. Có hàng triệu chuyên gia lập trình và các nhà phát triển phần mềm muốn tích hợp và tạo ra các sản phẩm mới sử dụng object detection. Nhưng công nghệ này xa tầm tay của họ và phức tạp để hiểu và sử dụng thực tế của nó.

## Giới thiệu về Tensorflow và Keras

### Thư viện Tensorflow

TensorFlow là một thư viện mã nguồn mở và một khung (framework) mạnh mẽ trong lĩnh vực học máy và deep learning. Nó được phát triển bởi Google Brain Team và được giới thiệu lần đầu vào năm 2015. TensorFlow cung cấp một cách tiếp cận linh hoạt và hiệu quả để xây dựng và huấn luyện các mô hình học máy phức tạp.

Tầm quan trọng của TensorFlow trong việc xây dựng mô hình học máy là như sau:

* Linh hoạt và phong phú: TensorFlow cung cấp một loạt các công cụ và API để xây dựng và triển khai các mô hình học máy. Nó hỗ trợ cả việc tạo ra các mạng nơ-ron truyền thẳng (feedforward neural networks) đơn giản và mô hình học sâu (deep learning) phức tạp. Bạn có thể tạo ra các kiến trúc mạng phức tạp bằng cách kết hợp các tầng, kết nối, và hoạt động khác nhau trong TensorFlow.
* Xử lý hiệu quả: TensorFlow được tối ưu để hoạt động trên nhiều nền tảng tính toán, bao gồm CPU và GPU. Nó cung cấp các biểu đồ tính toán và trình tối ưu hóa để tận dụng tối đa khả năng tính toán song song của GPU, giúp tăng tốc độ huấn luyện và dự đoán của mô hình học máy.
* Khả năng phân phối và triển khai: TensorFlow hỗ trợ phân phối và triển khai các mô hình học máy trên nhiều máy tính, bao gồm cả các hệ thống phân tán. Điều này cho phép bạn huấn luyện và triển khai mô hình trên dữ liệu lớn và môi trường sản xuất thực tế.
* Cộng đồng lớn và hỗ trợ đa dạng: TensorFlow có một cộng đồng đông đảo của các nhà phát triển và nghiên cứu, cung cấp tài liệu phong phú, ví dụ và hỗ trợ cho người dùng. Bạn có thể tìm thấy nhiều nguồn tài liệu, bài viết, mã mẫu và mô hình sẵn có để khám phá và áp dụng trong các dự án của mình.
* Sự phát triển liên tục: TensorFlow được Google và cộng đồng của nó chăm sóc và phát triển tiếp tục. Các phiên bản mới của TensorFlow được phát hành định kỳ, bổ sung các tính năng mới, cải thiện hiệu suất và sửa các lỗi. Điều này đảm bảo rằng TensorFlow luôn cung cấp các công nghệ mới nhất và đáng tin cậy cho việc xây dựng mô hình học máy.

### Framework Keras

Keras là một framework cao cấp (high-level) được xây dựng trên TensorFlow, được phát triển bởi François Chollet. Nó cung cấp một giao diện dễ sử dụng và trực quan để xây dựng và huấn luyện mô hình mạng nơ-ron, đặc biệt là mô hình Convolutional Neural Networks (CNNs).

Điểm mạnh của Keras là sự đơn giản và tập trung vào tính linh hoạt. Nó giúp đơn giản hóa việc xây dựng và huấn luyện mô hình học máy bằng cách cung cấp các lớp và module trừu tượng để xây dựng mạng nơ-ron và các phép toán liên quan. Bằng cách sử dụng Keras, bạn có thể tạo ra các mô hình mạng nơ-ron phức tạp một cách dễ dàng chỉ với một vài dòng mã.

Một số đặc điểm và lợi ích chính của Keras là:

* Dễ sử dụng: Keras cung cấp một API đơn giản và dễ hiểu, cho phép người dùng xây dựng và tùy chỉnh các mô hình mạng nơ-ron một cách dễ dàng. Cú pháp của Keras rõ ràng và cho phép người dùng tập trung vào việc xây dựng mô hình mà không phải lo lắng về các chi tiết cài đặt phức tạp.
* Hỗ trợ đa ngôn ngữ: Keras hỗ trợ nhiều ngôn ngữ lập trình, bao gồm Python, R và Julia. Điều này giúp mở rộng cộng đồng sử dụng và đóng góp vào Keras.
* Tích hợp với TensorFlow: Keras được xây dựng trên TensorFlow, cho phép tận dụng toàn bộ tiềm năng của TensorFlow trong việc xử lý và tính toán. Bạn có thể sử dụng các tính năng và công cụ của TensorFlow một cách dễ dàng trong Keras.
* Hỗ trợ mô hình mạng phổ biến: Keras cung cấp các lớp và mô-đun để xây dựng các mô hình mạng nơ-ron phổ biến như CNNs, Recurrent Neural Networks (RNNs) và mạng nơ-ron tái cấu trúc (autoencoders). Bạn có thể sử dụng các lớp này để xây dựng các mô hình phức tạp và đáp ứng nhu cầu của các bài toán khác nhau.
* Hỗ trợ mạnh mẽ cho huấn luyện mô hình: Keras cung cấp các công cụ hỗ trợ cho việc huấn luyện mô hình, bao gồm các thuật toán tối ưu (optimizer), hàm mất mát (loss function), và các đánh giá (metrics). Các tham số này cũng có thể tùy chỉnh và tinh chỉnh để tối ưu hoá quá trình huấn luyện.

# XÂY DỰNG ỨNG DỤNG

## Lựa chọn thuật toán

### Tổng quan về ý tưởng giải quyết đề tài

Với đề tài này, sau một thời gian tìm hiểu trên internet và tham khảo ý kiến từ nhiều nguồn, nhóm em đã đưa ra được ý tưởng chung như sau.

Sẽ chia bài toán ra làm 2 phần là:

* + Bài toán trích xuất khuôn mặt.
  + Bài toán phân loại các lớp (có đeo khẩu trang, không đeo khẩu trang).

### Trích xuất khuôn mặt

Trích xuất khuôn mặt là bước tiền xử lí dữ liệu cho bài toàn. Mục đích nhằm loại bỏ đi các yếu tố ngoại cảnh. Đầu ra sẽ là một ảnh có khuôn mặt ở chính giữa. Điều này sẽ giúp model dễ dàng nhận biết từ đó tăng hiệu quả, chính xác hơn trong việc học tập và dự đoán của model.

Hiện nay có rất nhiều kĩ thuật trích xuất khuôn mặt bằng cả học máy và học sâu.Python đã tích hợp khá nhiêu thư viện giúp chúng ta làm được điều này. Mỗi phương pháp thì sẽ có ưu và nhược điểm riêng của mình.

Sau đây là một số phương pháp nhóm em đã nghiên cứu được:

|  |  |  |
| --- | --- | --- |
| **Tên** | **Ưu điểm** | **Nhược điểm** |
| **Faster RCNN[6]** | Độ chính xác cao và khả năng phát hiện đa đối tượng.  Có khả năng xác định kích thước và vị trí chính xác của khuôn mặt. | Khó cài đặt.  Đòi hỏi tài nguyên tính toán cao, đặc biệt khi phải xử lý nhiều vùng quan tâm trong hình ảnh.  Thời gian huấn luyện và phát hiện chậm đối với ảnh đầu vào lớn |
| **Dlib sử dụng học máy[7]** | Độ chính xác tương đối cao trong việc phát hiện khuôn mặt.  Có thể hoạt động tốt trong các tình huống khác nhau, bao gồm cả ánh sáng yếu và biến dạng khuôn mặt. | Cần phải có một số lượng lớn dữ liệu huấn luyện để đạt được độ chính xác cao.  Hiệu suất phụ thuộc vào các đặc trưng thủ công được chọn và quá trình huấn luyện. |
| **Dlib sử dụng CNN[7]** | Cải thiện độ chính xác so với Dlib sử dụng học máy thông thường.  Độ chính xác tương đối cao trong việc phát hiện khuôn mặt và khả năng đặc trưng hóa khuôn mặt. | Tốc độ tương đối chậm.  Đòi hỏi tài nguyên tính toán cao trong quá trình huấn luyện và phát hiện.  Thời gian huấn luyện dài và yêu cầu tập dữ liệu lớn để đạt được độ chính xác cao. |
| **MTCNN[8]** | Độ chính xác cao trong việc phát hiện khuôn mặt, điểm mốc mắt và điểm mốc mũi.  Tính ổn định và khả năng hoạt động tốt trong các điều kiện ánh sáng khác nhau và biến dạng khuôn mặt.. | Yêu cầu tài nguyên tính toán cao, đặc biệt là khi xử lý ảnh đầu vào lớn.  Thời gian phát hiện chậm hơn so với một số phương pháp khác. |
| **DNN[9]** | Độ chính xác cao: DNN có khả năng học và mô hình hóa các đặc trưng phức tạp của khuôn mặt, giúp nâng cao độ chính xác của quá trình trích xuất và nhận dạng.  Tính linh hoạt: DNN có thể được đào tạo để nhận dạng và phân loại các biểu hiện khuôn mặt khác nhau, cho phép áp dụng cho nhiều ứng dụng khác nhau. | Độ nhận diện kém đối với ảnh chất lượng thấp, và quá đông người.  Đòi hỏi tài nguyên tính toán cao: DNN có cấu trúc phức tạp và yêu cầu tài nguyên tính toán lớn để huấn luyện và triển khai.  Yêu cầu dữ liệu huấn luyện lớn: DNN cần một lượng lớn dữ liệu huấn luyện để đạt được độ chính xác cao và tránh overfitting. |

Bảng 2.1 Bảng so sánh các phương pháp trích xuất khuôn mặt

🡪Từ những ưu nhược điểm trên, nhóm em quyết định lựa chọn MTCNN để trích xuất khuôn mặt trong khuôn khổ thử nghiệm.

MTCNN cung cấp một tổng thể tốt về độ chính xác và hiệu suất. MTCNN có khả năng phát hiện và xác định khuôn mặt, điểm mốc mắt và điểm mốc mũi một cách chính xác. Nó cũng có khả năng hoạt động tốt trong các điều kiện khác nhau và có tính ổn định. Mặc dù MTCNN yêu cầu tài nguyên tính toán cao và thời gian phát hiện có thể chậm hơn một số phương pháp khác, nhưng với độ chính xác và tính linh hoạt của nó, nó được coi là một lựa chọn hợp lý trong việc trích xuất khuôn mặt.

### Xây dựng mô hình (model)

Như đã trình bày ở phần lý thuyết, kết hợp với kiến thức được giới thiệu trên lớp và kiến trức trên mạng nhóm em đã tìm được một vài kiến trúc của mạng CNN có thể ứng dụng vào bài toán này. Nhóm em đã chọn VGG16 để tiến hành huấn luyện mẫu và thu về kết quả.

### Kết hợp

Sau khi thực hiện xong 2 bước trên. Việc còn lại của nhóm em là kết hợp 2 phần lại với nhau. Xử lý dữ liệu đầu vào bằng trích xuất khuôn mặt sau đó cho vào model đưa ra kết quả dự đoán.

## Các bước thực hiện

### Thu thập dữ liệu

Nhóm em đã đi thu thập dữ liệu là các ảnh mặt người có thể có hoặc không đeo khẩu trang, sau đó đánh nhãn cho các ảnh.

Chart, bar chart

Description automatically generated

Hình 2.1 Tổng quan về dữ liệu

Tổng cộng nhóm thu thập 12.000 ảnh, sau khi đánh nhãn và loại bỏ các ảnh không hợp lệ, nhóm còn lại 11.792 ảnh trong đó 5.883 ảnh đeo khẩu trang và 5.909 ảnh không đeo khẩu trang.

Sau đó nhóm đã chia làm 3 tập:

* Tập train gồm 10.000 ảnh được chia ngẫu nhiên. Có mục đích huấn luyện mẫu và cho mẫu học.
* Tập test (hay public test). Tập này chiếm gần 10% ảnh (992 ảnh) để giúp mẫu kiểm tra trong lúc huấn luyện.
* Tập cuối cùng – private test còn lại có mục đích thử mẫu sau khi huấn luyện xong.

### Xây dựng chương trình

#### Tiền xử lý dữ liệu

Sau khi chia dữ liệu vào các tập đã trình bày như trên. Nhóm em tiến hành xử lý những dữ liệu này.

* Chuyển tất cả dữ liệu về kích thước (224, 224, 3) để huấn luyện mẫu.
* Tiếp đó là chuẩn hóa đầu ra thành dạng vector có kích thước là 2. (Tương ứng với 2 kết quả cần thu về là có hay không đeo khẩu trang)

#### Xây dựng mô hình

Tiến hành code model theo kiến trúc VGG16 sử dụng thư viện tensorflow – framework Keras trên python.

A screenshot of a computer

Description automatically generated with medium confidence

Hình 2.2 Lấy dữ liệu

*A screenshot of a computer

Description automatically generated with medium confidence*

Hình 2.3 Chuẩn bị kiến trúc VGG16

A screenshot of a computer

Description automatically generated with medium confidence

Hình 2.4 Tiến hành huấn luyện mô hình

Sau quá trình huấn luyện với Epoch = 10, nhóm em đã thu về được kết quả là model đã được huấn luyện với độ chính xác khá cao (val\_accuracy = 0.9929)

A picture containing diagram, plot, line, text

Description automatically generated

Hình 2.5 Đồ thị loss trong quá trình huấn luyện

A picture containing diagram, line, plot, text

Description automatically generated

Hình 2.6 Đồ thị accuracy trong quá trình huấn luyện

#### Kiểm tra, đánh giá hiệu quả của mô hình

Từ các thông số thu được, nhóm em đưa ra một số nhận xét. Model học tốt qua tập dữ liệu, không bị hiện tượng quá khớp (overfitting).

Thử lại ngẫu nhiên với tập private test thu được kết quả khá chính xác.

A screenshot of a computer program

Description automatically generated with low confidence

Hình 2.7 Thử lại với 200 ảnh ngẫu nhiên trong tập private test

Hình cho ra kết quả không chính xác là hình ảnh có chất lượng không tốt, không chứa toàn bộ khuôn mặt.

A picture containing timeline

Description automatically generated

Hình 2.8 Hình thu về kết quả không chính xác

Sau nhiều lần thử lại khác nhau, nhóm tìm được thêm một số hình ảnh khác mà model đưa ra kết quả không chính xác.

A close-up of a person wearing a mask

Description automatically generated with medium confidence

Hình 2.9 Một số hình ảnh mà model đưa ra kết quả không chính xác

Ngoài ra, model còn có thể sử dụng với camera thời gian thực sau khi kết hợp với “trích xuất gương mặt”.

Một số hình ảnh với camera:

A person taking a selfie

Description automatically generated

Hình 2.10 Thử nghiệm với camera

Tiếp tục thử nghiệm thêm các trường hợp khác, nhóm phát hiện mô hình hiện tại cho ra kết quả chưa đúng trong một số trường hợp, cụ thể:

* Trường hợp sử dụng khẩu trang giả.

A screenshot of a person with their hand over their mouth

Description automatically generated with medium confidence

Hình 2.11 Kết quả của mô hình ban đầu trong trường hợp sử dụng khẩu trang giả

* Trường hợp đeo khẩu trang không đúng cách

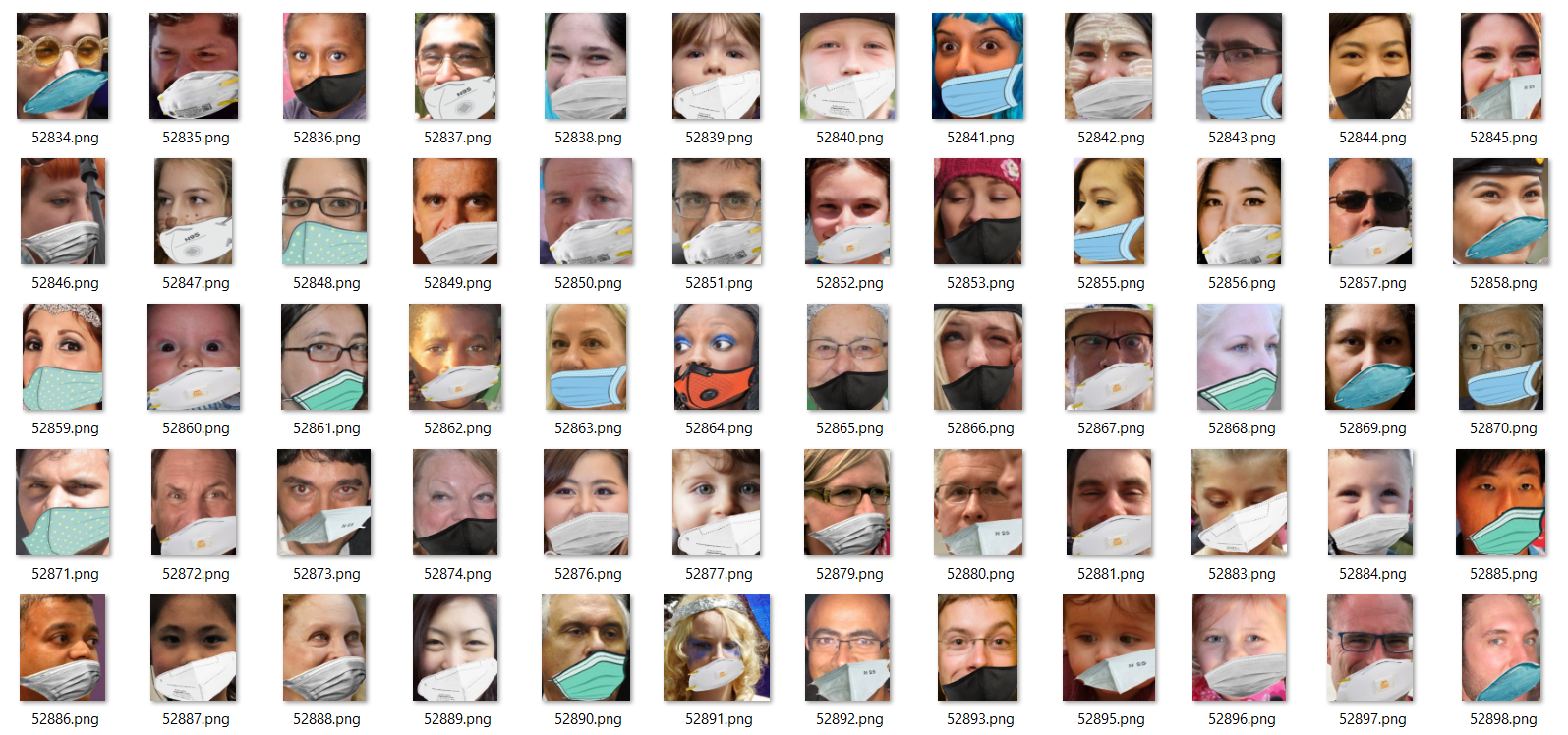
A picture containing person, toddler, human face, wall

Description automatically generated

Hình 2.12 Kết quả của mô hình ban đầu trong trường hợp đeo khẩu trang không đúng cách

#### Tiến hình tối ưu, nâng cấp mô hình

Theo quan sát thực tế, trường hợp đeo khẩu trang không đúng cách xảy ra rất phổ biến, nhóm quyết định tiến hành nâng cấp mô hình, thu thập thêm dữ liệu về trường hợp đeo khẩu trang không đúng cách và tiến hành huấn luyện một mô hình khác có khả năng đưa ra kết quả đúng với trường hợp này.



Hình 2.13 Một số hình ảnh trong bộ dữ liệu cho trường hợp đeo khẩu trang không đúng cách

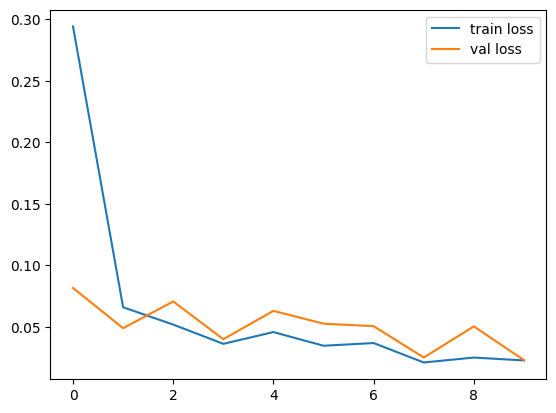
A picture containing text, screenshot, font, number

Description automatically generated

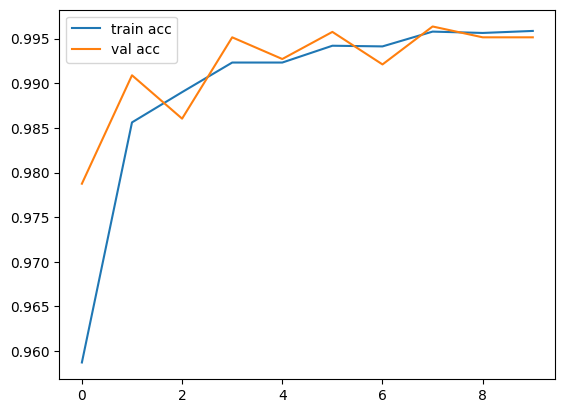
Hình 2.14 Huấn luyện mô hình lần 2

### Kết quả đạt được

Sau khi thực hiện lại việc huấn luyện lại mô hình, kết quả trả về cũng tương đối tốt.



Hình 2.15 Đồ thị loss trong huấn luyện lần 2



Hình 2.16 Đồ thị accuracy trong huấn luyện lần 2

Thử nghiệm lại các trường hợp:

A screenshot of a child

Description automatically generated with medium confidence

Hình 2.17 Camera - trường hợp không đeo khẩu trang

A person wearing a mask

Description automatically generated with low confidence

Hình 2.18 Camera - trường hợp đeo khẩu trang

A person wearing a mask

Description automatically generated with low confidence

Hình 2.19 Camera - trường hợp đeo khẩu trang không đúng cách

### Tổng kết, kiến nghị

#### Tổng kết

Ngày nay khi mà nền kinh tế, đời sống của con người ngày càng phát triển. Các công cụ, và giải phát ngày càng trở nên thông minh trong nền kinh tế 4.0 hiện nay. Việc chuyển đổi số và số hóa các hoạt động bình thường mà phải tốn rất nhiều công sức của nhiều người là điều tất nhiên. Nhu cầu của con người về những công nghệ thông minh cũng ngày càng tăng lên trong những năm gần đây. Nhất là về lĩnh vực trí tuệ nhân tạo. Mỗi ngày có vô số các bài báo được công bố về lĩnh vực này. Không những thế với hậu quả nặng nề của Covid-19 đã gây ảnh hưởng cực lớn đến nền kinh tế và xã hội. Chính vì thế việc nhận diện ra người có đeo khẩu trang hay không khi vào một cơ quan, công ty, nơi đông người là rất cần thiết. Từ những thực tiễn đó nhóm em nghĩ đề tài này có thể được nâng cấp và phát triển nhiều hơn. Từ đó có thể đưa vào thực tiễn để ứng dụng trong cuộc sống.

#### Kiến nghị

Mặc dù đạt được kết quả tương đối tốt khi thử nghiệm xong nhóm cũng đã nhận ra được một số điểm yếu của ứng dụng đề tài này.

* **Điểm yếu**
* Việc nhận diện gương mặt sử dụng MTCNN khá chậm, dẫn đến khi thử nghiệm với camera ta thu được video không được mượt, số khung hình trên giây thấp.
* Nhóm chưa thực hiện huấn luyện mô hình với trường hợp sử dụng khẩu trang giả.
* Model còn tương đối nặng.
* **Hướng phát triển**
* Nghiên cứu cách và thu thập dữ liệu để mô hình có thể phát hiện cả trường hợp sử dụng khẩu trang giả
* Nghiên cứu thêm về các kĩ thuật, thuật toán trích xuất khuôn mặt, Từ đó tăng được tốc độ cũng như độ chính xác của việc trích xuất khuôn mặt.
* Tiếp túc nghiên cứu các kĩ thuật xây dựng mạng Neural Deep learing từ đó nâng cấp model của mình lên chính xác hơn, nhẹ hơn.

# TÀI LIỆU THAM KHẢO

[1] Kaiming He, Xiangyu Zhang, Shaoquing Ren, Jian Sun, Microsoft Research, *Deep Residual Learning for Image Recognition*, 2015.

[2] Karen Simonyan, Andrew Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, v6, 2015*.*

[3] Bách khoa toàn thư Việt Nam, *Học sâu*.

[4] Chi-Feng Wang, *The Vanishing Gradient Problem*, 2019.

[5] Martin Krasser, *Deep face recognition with Keras, Dlib and OpenCV*, 2018.

[6] Iván de Paz Centeno, *MTCNN face detection implementation for TensorFlow, as a PIP package*, 3th Edition, 2021.

[7] Adrian Rosebrock, *Face detection with OpenCV and deep learning*, 2018.

[8] Zhilu Zhang, Mert R. Sabuncu, *Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels*, v4, 2018.