TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH

KHOA CƠ KHÍ CHẾ TẠO MÁY

BỘ MÔN CƠ ĐIỆN TỬ

BÁO CÁO CUỐI KÌ

TRÍ TUỆ NHÂN TẠO

FACE DETECTION

FOR IDENTITY VERIFICATION

GVHD: PGS. TS Nguyễn Trường Thịnh

MHP: ARIN337629_22_2_09

Họ và tên: Nguyễn Quang Phúc

MSSV: 20146142

Thành phố Hồ Chí Minh, tháng 5 năm 2023

MỤC LỤC

DANH MỤC CÁC TỪ VIẾT TẮT

Viết tắt	Ý nghĩa
AI	**A**rtificial **I**ntelligent
CNN	**C**onvolutional **N**eural **N**etwork

CHƯƠNG 1. TỔNG QUAN

1.1. Giới thiệu

Trong thời đại hiện nay, thế giới công nghệ ngày càng phát triển, bên cạnh đó không thể không kể đến sự tiến bộ vượt bậc với tốc độ nhanh chóng của AI. Việc ứng dụng AI vào các công việc của cuộc sống, từ những công việc đơn giản nhất, cho đến những công việc đòi hỏi sự phức tạp và trí tuệ, đã và đang ngày càng trở nên phô biến và rộng rãi. Sự kết hợp giữa Trí tuệ nhân tạo và công nghệ nhận diện khuôn mặt đã tạo ra những tiềm năng vô cùng đáng kể trong lĩnh vực bảo mật và quản lý dân sự.

Nhận diện khuôn mặt là quá trình tự động nhận biết và xác định các đặc điểm khuôn mặt của một người thông qua hình ảnh hoặc video. Với việc sử dụng Trí tuệ nhân tạo, đặc biệt là Deep Learning và Mạng Nơ ron tích chập (CNN) đã giúp cho việc phân tích và nhận dạng khuôn mặt nâng cao hiệu suất và độ chính xác. Từ đó, đưa ra những giải pháp mạnh mẽ trong việc xác minh danh tính con người.

Trong lĩnh vực an ninh, công nghệ này giúp phát hiện và ngăn chặn các hoạt động gian lận, xâm phạm an ninh và đảm bảo an toàn cộng đồng. Bên cạnh đó, trong lĩnh vực quản lý nhân sự, việc sử dụng công nghệ nhận diện khuôn mặt giúp đơn giản hóa quá trình điểm danh, kiểm soát truy cập và quản lý dữ liệu nhân viên.

Tuy nhiên, vấn đề Nhận diện khuôn mặt để xác định danh tính cũng đặt ra những thách thức và tranh cãi liên quan đến quyền riêng tư và bảo mật thông tin cá nhân. Điều này yêu cầu việc phát triển và áp dụng các quy định và chính sách bảo vệ quyền riêng tư một cách cẩn thận và minh bạch.

1.2. Lý do chọn đề tài

Như em đã đề cập ở 1.1, trong bối cảnh công nghệ đang ngày càng phát triển, đề tài Nhận diện khuôn mặt để xác định danh tính đã và đang đóng một vai trò quan trọng trong lĩnh vực an ninh và quản lý thông tin. Vậy nên, việc nghiên cứu và phát triển các phương pháp và công nghệ nhận diện khuôn mặt là một lĩnh vực triển vọng, mang lại nhiều lợi ích to lớn và đầy sức hấp dẫn.

Vậy nên, mục tiêu chính của em khi thực hiện đề tài này là xây dựng một mô hình giúp nhận diện được danh tính của một người (giới tính, tên) qua khuôn mặt sử dụng ngôn

ngữ Python. Mô hình sẽ áp dụng các phương pháp có trong Machine Learning và Deep learning để giúp cho việc nhận diện đạt được các kết quả tốt hơn.

1.3. Mục tiêu nghiên cứu

- Xây dựng mô hình nhận diện khuôn mặt bằng giải pháp CNN
- Đưa mô hình lên webapp để mọi người dễ sử dụng

1.4. Phương pháp nghiên cứu

- Thu thập dữ liệu: Chúng ta thu thập dữ liệu chính là ảnh các khuôn mặt của nhiều người khác nhau được chụp từ nhiều góc độ khác nhau. Dữ liệu sẽ được chia thành 2 loại, một cho xác định giới tính, một cho xác định tên người.
- Xây dựng mô hình học máy: Chúng ta sẽ áp dụng thuật toán học máy CNN (Convolutional Neural Network) để xây dựng mô hình nhận dạng giới tính và tên. Chúng ta sẽ xem xét các đặc trưng quan trọng để đào tạo mô hình và cân nhắc việc sử dụng các phương pháp tăng cường dữ liệu để cải thiện độ chính xác.
- Đánh giá hiệu suất và tinh chỉnh mô hình: Sau khi huấn luyện mô hình, chúng ta sẽ đánh giá hiệu suất của mô hình bằng cách sử dụng các phép đo như độ chính xác, độ phủ, độ nhạy và đặc tính. Dựa trên kết quả đánh giá, chúng ta sẽ tinh chỉnh mô hình để đạt được độ chính xác cao nhất có thể.
- Xây dựng Giao diện tương tác: Cuối cùng, chúng ta sẽ xây dựng một giao diện cho phép người dùng có thể đăng ảnh lên để kiểm tra và nhận dạng trong thời gian thực với webcam.

1.5. Nội dung báo cáo

Chương này đã cung cấp một cái nhìn tổng quan về đề tài nhận dạng danh tính bằng Python, bao gồm lý do chọn đề tài, mục tiêu nghiên cứu, phương pháp nghiên cứu và nội dung báo cáo. Trong các chương tiếp theo, chúng ta sẽ đi sâu vào từng phần để trình bày chi tiết về từng bước trong quy trình nhận dạng giới tính và tuổi, bao gồm cách thu thập và tiền xử lý dữ liệu, xây dựng mô hình học máy, đánh giá hiệu suất và xây dựng ứng dụng Python.

Bài báo cáo này được xây dựng với 4 chương, bao gồm:

CHƯƠNG 1: TỔNG QUAN

CHƯƠNG 2: CƠ SỞ LÍ THUYẾT

CHƯƠNG 3: ỨNG DỤNG

CHƯƠNG 4: KẾT LUẬN

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

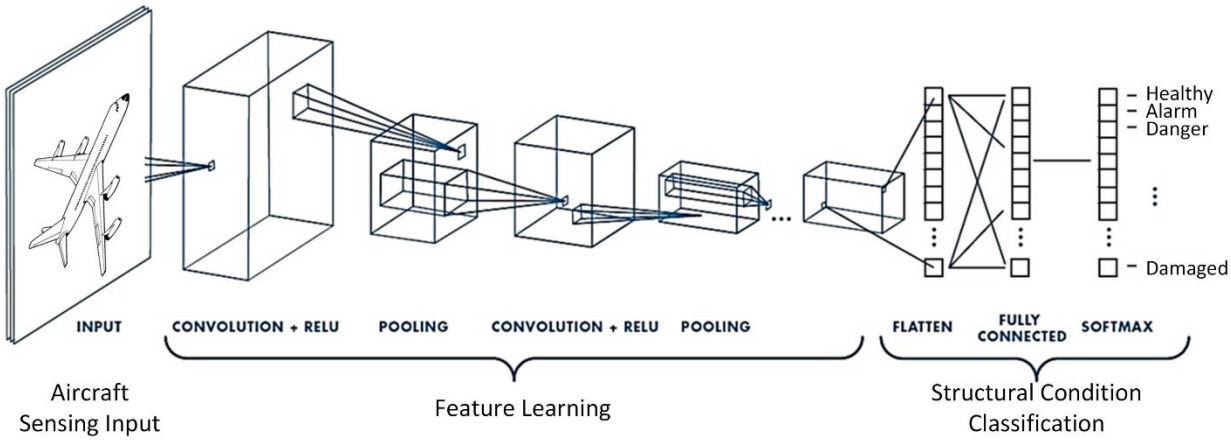
2.1. Thuật toán CNN - Convolutional Neural Network

2.1.1. Convolutional Neural Network là gì

Convolutional Neural Network (CNN – Mạng nơ-ron tích chập) là một trong những mô hình Deep Learning tiên tiến và phổ biến nhất đặc biệt là trong lĩnh vực Machine Vision (Thị giác máy). Bằng cách sử dụng và phát triển từ mô hình Neural Network, CNN giúp cho chúng ta có thể xây dựng được những hệ thống thông minh với độ chính xác cao như hiện nay. CNN được sử dụng nhiều trong các bài toán nhận dạng các vật thể trong ảnh, là một trong các bài toán quan trọng bậc nhất trong lĩnh vực Machine Vision

2.1.2. Cấu trúc của mạng CNN

Khi nhìn sơ qua, chúng ta có lẽ sẽ thấy cấu trúc mạng CNN lúc thì thật đơn giản lúc thì lại rất phức tạp khi có rất nhiều lớp chồng lên nhau. Tuy nhiên, chúng ta có thể tổng hợp lại cấu trúc mạng CNN sẽ bao gồm một số lớp cơ bản sau thường được sắp xếp theo thứ tự sau: Convolutional Layer + Nonlinear Activation => Pooling Layer => Fully Connected Layer. Bên cạnh đó, Trình tự qua các lớp Convolutional Layer + Nonlinear Activation, Pooling Layer hoàn toàn có thể đước lặp lại nhiều lần trước khi tiến tới lớp Fully Connected nhằm trích xuất được những đặc trưng của bức ảnh, điều đó khiến cho cấu trúc mạng CNN nhìn khá phức tạp.



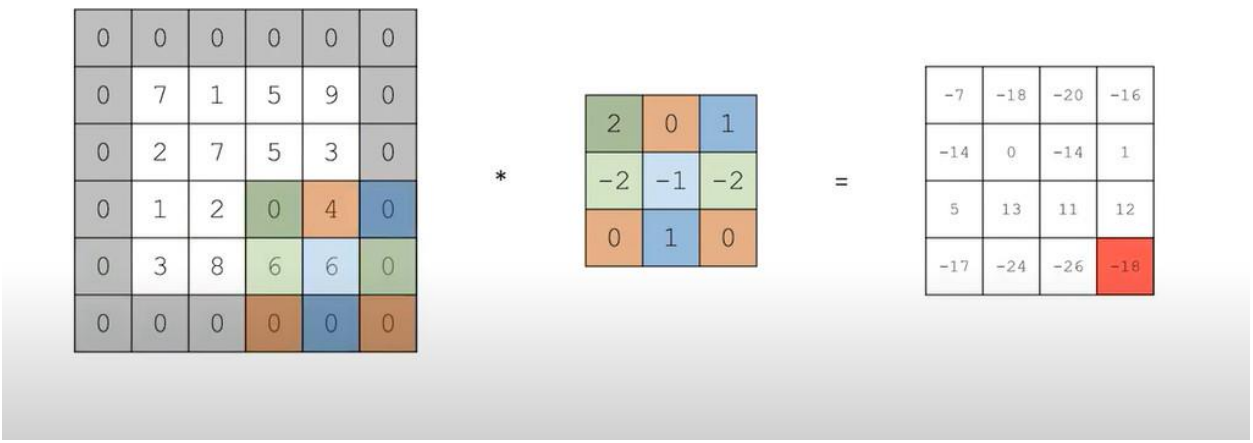
Hình 2.1 Cấu trúc của mô hình CNN

2.1.2.1. Convolutional Layer

Convolutional layer là lớp quan trọng nhất và cũng là lớp đầu tiên của mô hình CNN. Lớp này có chức năng chính là phát hiện các đặc trưng có trong bức ảnh bao gồm các đặc

trưng cơ bản như: góc, cạnh, màu sắc, hoặc các đặc trưng phức tạp hơn như Texture của ảnh.

Convolutional Layer nhận đầu vào là một Tensor (ma trận đa chiều) biểu diễn dữ liệu hình ảnh. Mỗi phần tử trong tensor thể hiện giá trị của một pixel hoặc đặc trưng cụ thể của hình ảnh. Sau đó, Convolutional layer sử dụng một số lượng bộ lọc (filters) để thực hiện phép tích chập trên đầu vào. Mỗi bộ lọc là một ma trận nhỏ chứa các trọng số (weights) có thể học được. Quá trình tích chập được thực hiện bằng cách trượt bộ lọc qua toàn bộ đầu vào theo bước nhảy (stride), tính tích chập (element-wise multiplication) giữa bộ lọc và một phần của đầu vào, sau đó tổng hợp kết quả lại thành một ma trận mới. Convolutional layer sử dụng một số lượng bộ lọc (filters, thông thường sử dụng các filter có kích thước là 3x3) để thực hiện phép tích chập trên đầu vào. Mỗi bộ lọc là một ma trận nhỏ chứa các trọng số (weights) có thể học được. Quá trình tích chập được thực hiện bằng cách trượt bộ lọc qua toàn bộ đầu vào theo bước nhảy (stride), tính tích chập (element-wise multiplication) giữa bộ lọc và một phần của đầu vào, sau đó tổng hợp kết quả lại thành một ma trận mới. Kết quả của phép tích chập là một feature map, đại diện cho việc trích xuất các đặc trưng từ đầu vào. Feature map này giúp nhận biết các cạnh, góc, màu sắc, và các

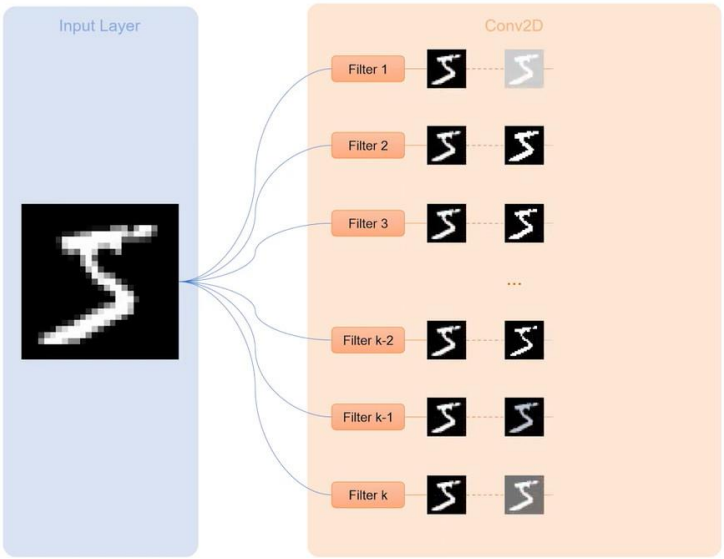


đặc trưng khác trong hình ảnh.

Hình 2.2 Kết quả tính tích chập của 1 kernel

Trong Convolutional Layer, có xuất hiện một thông số là Padding, đó chính là các số 0 xuất hiện xung quanh bức ảnh trên hình 2.2, việc thêm padding giúp cho chúng ta khi nhân tích chập các filter để lấy các đặc trưng của ảnh không làm cho kích thước đầu ra không bị thay đổi so với kích thước ban đầu của ảnh.

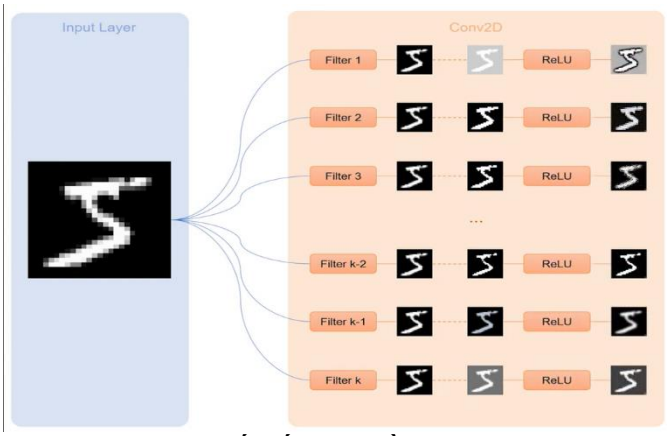
Với mỗi kernel khác nhau ta sẽ học được những đặc trưng khác nhau của ảnh, nên trong mỗi convolutional layer ta sẽ dùng nhiều kernel để học được nhiều thuộc tính của ảnh. Vì mỗi kernel cho ra output là 1 matrix nên k kernel sẽ cho ra k output matrix. Ta kết hợp k output matrix này lại thành 1 tensor 3 chiều có chiều sâu k.



Hình 2.3 Minh họa một số kết quả đầu ra qua lớp Conv2D

2.1.2.2. Nonlinear Activation

ReLU (Rectified Linear Units, f = max(0, x)) là hàm kích hoạt phổ biến nhất cho CNN tại thời điểm của bài viết, được giới thiệu bởi Geoffrey E. Hinton năm 2010. Trước khi hàm ReLU được áp dụng thì những hàm như sigmoid hay tanh mới là những hàm được sử dụng phổ biến. Hàm ReLU được ưa chuộng vì tính toán đơn giản, giúp hạn chế tình trạng vanishing gradient, và cũng cho kết quả tốt hơn. ReLU cũng như những hàm kích hoạt khác, được đặt ngay sau tầng convolution, ReLU sẽ gán những giá trị âm bằng 0 và giữ nguyên giá trị của đầu vào khi lớn hơn 0.

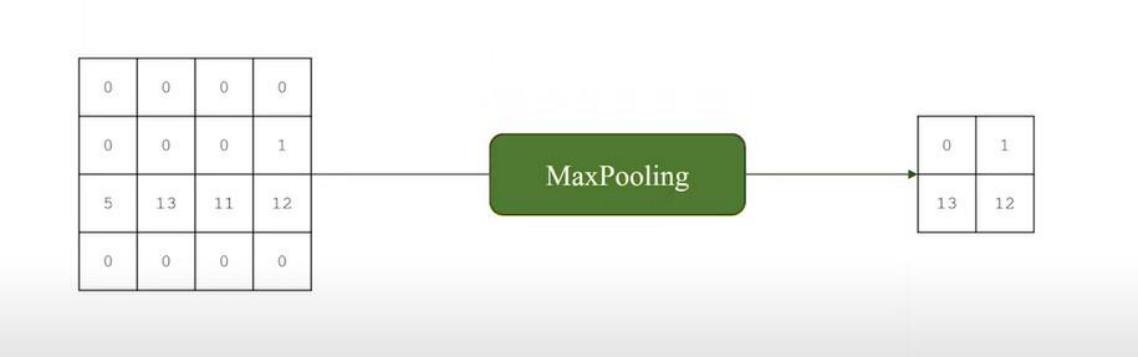


Hình 2.4 Minh họa một số kết quả đầu ra khi sử dụng hàm kích hoạt ReLU

ReLU cũng có một số vấn đề tiềm ẩn như không có đạo hàm tại điểm 0, giá trị của hàm ReLU có thể lớn đến vô cùng và nếu chúng ta không khởi tạo trọng số cẩn thận, hoặc khởi tạo learning rate quá lớn thì những neuron ở tầng này sẽ rơi vào trạng thái chết, tức là luôn có giá trị < 0.

2.1.2.3. Pooling Layer

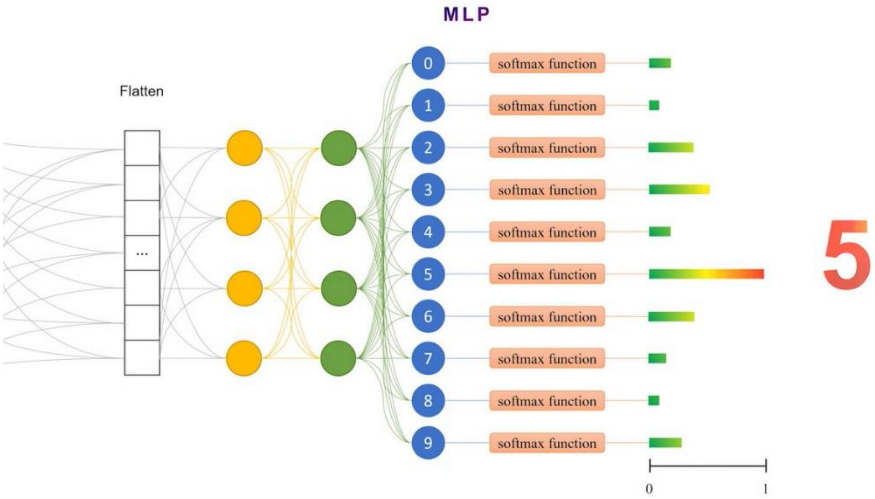
Sau hàm kích hoạt, thông thường với cấu trúc đơn giản chúng ta sẽ đến với Pooling Layer. Một số loại Pooling layer phổ biến như là max-pooling, average pooling, với chức năng chính là giảm kích thước của đầu ra lớp trước đó nhưng vẫn giữ được các nét đặc trưng nổi bật của Feature map đó, nhờ đó, giúp giảm thời gian huấn luyện và hạn chế xảy ra overfit. Với một pooling có kích thước m x m, chúng ta cần phải trượt filter m x m này trên những vùng ảnh có kích thước tương tự rồi sau đó tính max, hay average cho vùng ảnh đó.



Hình 2.5 Pooling Layer

2.1.2.4. Fully Connected Layer

Sau khi ảnh được truyền qua nhiều convolutional layer và pooling layer thì model đã học được tương đối các đặc điểm của ảnh thì tensor của output của layer cuối cùng sẽ được là phẳng thành vector và đưa vào một lớp được kết nối như một mạng nơ-ron. Với FC layer được kết hợp với các tính năng lại với nhau để tạo ra một mô hình. Cuối cùng sử dụng softmax hoặc sigmoid để phân loại đầu ra.



Hình 2.6 Fully Connected Layer

2.2. Thư viện Tensorflow

Thư viện TensorFlow là một thư viện phần mềm mã nguồn mở được sử dụng trong lĩnh vực học máy và trí tuệ nhân tạo. Nó được phát triển bởi nhóm Google Brain với mục tiêu huấn luyện và ứng dụng các mạng nơ-ron sâu (deep neural networks)

TensorFlow cung cấp một cấu trúc và công cụ mạnh mẽ để xây dựng và triển khai các mô hình học máy và mạng nơ-ron sâu. Nó hỗ trợ các tác vụ như xử lý dữ liệu, xây dựng mạng nơ-ron, huấn luyện mô hình, và triển khai vào sản phẩm thực tế như:

- TensorFlow hỗ trợ các ngôn ngữ lập trình như Python và C++. Đặc biệt, Python là ngôn ngữ phổ biến được sử dụng rộng rãi trong việc phát triển các ứng dụng học máy với TensorFlow.
- TensorFlow cho phép xây dựng các mô hình học máy và mạng nơ-ron sâu thông qua việc tạo ra các biểu đồ tính toán (computational graphs). Biểu đồ này mô tả cách dữ liệu chảy qua các phép tính và quy trình huấn luyện mô hình.
- TensorFlow cung cấp một loạt các API cho việc xây dựng và huấn luyện mô hình học máy. Các API này bao gồm Keras, Estimator, và low-level TensorFlow API, cho phép người dùng lựa chọn phù hợp với nhu cầu và kỹ năng của mình.
- TensorFlow cung cấp công cụ hỗ trợ như TensorBoard để giúp quản lý, theo dõi và hình dung quá trình huấn luyện mô hình. TensorBoard cho phép hiển

thị biểu đồ, đồ thị quá trình huấn luyện, và phân tích các chỉ số đánh giá mô hình.

- TensorFlow được sử dụng rộng rãi trong cả nghiên cứu và ứng dụng thực tế. Nó đã trở thành một trong những thư viện học máy phổ biến nhất và được sử dụng bởi nhiều tổ chức, từ các công ty công nghệ lớn cho đến các nhà nghiên cứu và các nhà phát triển độc lập.

Dù đem đến nhiều lợi ích nhưng thực ra nguyên lý hoạt động của Tensorflow khá đơn giản. Về cơ bản Tensorflow sẽ giúp người dùng tạo ra các biểu đồ luồng dữ liệu hoặc những cấu trúc mô tả. Đây cũng là lý do tại sao Tensorflow được coi như là một framework. Những khung sườn này sẽ hướng dẫn dữ liệu làm cách nào để đi qua một biểu đồ hoặc một series nodes đang được xử lý. Lúc này, mỗi nodes sẽ đại diện cho một hoạt động toán học cần xử lý. Còn mỗi kết nối hoặc mỗi edge sẽ được coi như một tensor hoặc một mảng dữ liệu đa chiều.

2.3. Thư viện Keras

Keras chạy trên các thư viện máy mã nguồn mở như TensorFlow, Theano hoặc Bộ công cụ nhận thức (CNTK). Theano là một thư viện python được sử dụng cho các tác vụ tính toán số nhanh. TensorFlow là thư viện toán học biểu tượng nổi tiếng nhất được sử dụng để tạo mạng nơ-ron và mô hình học sâu. TensorFlow rất linh hoạt và lợi ích chính là tính toán phân tán. CNTK là khung học sâu được phát triển bởi Microsoft. Nó sử dụng các thư viện như Python, C #, C ++ hoặc các bộ công cụ học máy độc lập. Theano và TensorFlow là những thư viện rất mạnh nhưng khó hiểu để tạo mạng nơ-ron.

Keras dựa trên cấu trúc tối thiểu, cung cấp một cách dễ dàng và dễ dàng để tạo các mô hình học sâu dựa trên TensorFlow hoặc Theano. Keras được thiết kế để xác định nhanh các mô hình học sâu. Chà, Keras là một lựa chọn tối ưu cho các ứng dụng học sâu.

Keras tận dụng các kỹ thuật tối ưu hóa khác nhau để làm cho API mạng thần kinh cấp cao dễ dàng hơn và hiệu quả hơn. Nó hỗ trợ các tính năng sau:

- API nhất quán, đơn giản và có thể mở rộng.
- Cấu trúc tối thiểu - dễ dàng đạt được kết quả mà không cần rườm rà.
- Hỗ trợ nhiều nền tảng và backend.

- Thân thiện với người dùng chạy trên cả CPU và GPU.
- Khả năng mở rộng tính toán cao.

Keras năng động, mạnh mẽ và có những ưu điểm sau

- Cộng động lớn hỗ trợ
- Dễ dàng để kiểm tra.
- Mạng nơ-ron Keras được viết bằng Python giúp mọi thứ đơn giản hơn.

Keras hỗ trợ cả mạng convolution và recurrent.

2.4. Thư viện OpenCV

Thư viện OpenCV (Open Source Computer Vision Library), một thư viện mã nguồn mở được sử dụng phổ biến trong lĩnh vực thị giác máy tính và xử lý ảnh. Nó cung cấp các công cụ và chức năng để thực hiện các tác vụ như xử lý ảnh, nhận dạng đối tượng, phân loại, theo dõi và phân tích video. Dưới đây là một số chức năng chính của OpenCV:

• Đọc và ghi ảnh và video: OpenCV cho phép bạn đọc và ghi ảnh và video từ các nguồn dữ liệu khác nhau. Bạn có thể đọc ảnh từ file ảnh, camera hoặc video, và lưu lại ảnh hoặc video sau khi xử lý.

• Xử lý ảnh: OpenCV cung cấp nhiều công cụ xử lý ảnh để thực hiện các thao tác như lọc thông tin, biến đổi hình học, biến đổi màu sắc, phát hiện biên, lấp đầy, cắt và kết hợp ảnh, và nhiều công việc khác.

• Phân loại và nhận dạng đối tượng: OpenCV cung cấp các phương pháp nhận dạng và phân loại đối tượng như phát hiện khuôn mặt, nhận dạng vật thể, phân loại đối tượng và nhận dạng chữ viết tay.

• Xử lý video: OpenCV hỗ trợ xử lý video với các chức năng như trích xuất khung hình, ghi lại video, theo dõi vật thể, phân tích chuyển động và chuyển đổi định dạng video.

• Xử lý camera: OpenCV cho phép truy cập và xử lý dữ liệu từ camera. Bạn có thể chụp ảnh từ camera, thực hiện xử lý trực tiếp trên dữ liệu camera và hiển thị video từ camera.

OpenCV là một thư viện mạnh mẽ và linh hoạt, hỗ trợ nhiều ngôn ngữ lập trình như Python, C++, Java và nhiều nền tảng khác nhau. Nó đã trở thành công cụ không thể thiếu trong nhiều ứng dụng xử lý ảnh và thị giác máy tínhThư viện tập trung vào việc mô hình

hóa dữ liệu. Nó không tập trung vào việc truyền tải dữ liệu, biến đổi hay tổng hợp dữ liệu. Những công việc này dành cho thư viện Numpy và Pandas.

2.5. Streamlit

Streamlit là một thư viện mã nguồn mở được sử dụng trong lĩnh vực Machine Learning và Data Science để tạo ra các ứng dụng web dễ dàng và nhanh chóng. Điểm đặc biệt của Streamlit là khả năng tạo giao diện web tương tự với Jupyter notebook và không hiển thị mã nguồn, giúp người dùng tạo ra các ứng dụng có tính hoàn thiện cao.

Với Streamlit, người dùng không cần có nhiều kiến thức về HTML, CSS và JavaScript để tạo ra các ứng dụng web tương tác. Thay vào đó, người dùng chỉ cần sử dụng Python để tạo giao diện và điều khiển các thành phần của ứng dụng. Streamlit cung cấp các thành phần sẵn có và linh hoạt để tạo ra các trải nghiệm tương tác phong phú chỉ với một vài dòng mã.

Để bắt đầu với Streamlit, người dùng có thể xem các ví dụ và chia sẻ từ cộng đồng Streamlit để tìm hiểu về những gì người khác đã tạo ra và chia sẻ. Thư viện cung cấp một loạt các thành phần có thể được cài đặt chỉ với một dòng mã. Nếu không tìm thấy một thành phần phù hợp, người dùng có thể tự chế tạo thành phần của riêng mình.

Ngoài ra, Streamlit cũng hỗ trợ việc tạo các thành phần hai chiều trạng thái cho phép truyền thông tin trở lại Python từ trình duyệt. Điều này cho phép người dùng tạo các tiện ích con như bộ đếm và khám phá các khả năng tương tác của Streamlit.

CHƯƠNG 3. ỨNG DỤNG

3.1. Xây dựng mô hình

Em sử dụng Google Colab để training model, đồng thời sử dụng GPU được tích hợp trên Google Colab để tăng tốc độ train

3.1.1. Datasets

Trong đề tài này, em chia cơ sở dữ liệu thành 2 phần: Một phần cho việc nhận dạng giới tính, Một phần cho việc nhận dạng tên đối tượng.

Đối với việc nhận dạng giới tính, muốn đạt được một kết quả chính xác cao nhất thì cần có một cơ sở dữ liệu rất lớn. Tuy nhiên, vì nhiều lý do, em chỉ sử dụng tập dữ liệu không quá lớn, nên độ chính xác chỉ ở mức độ tương đối.

Đối với việc nhận dạng tên đối tượng, gồm có dữ liệu là khuôn mặt của 6 bạn trong lớp.

Bảng 3.1. Datasets

	Loại hình	Số lượng ảnh	Tổng
Train	Gender	1604	2004
Validation	Gender	400	
Train	Name	1256	1570
Validation	Name	314	

3.1.2. Xây dựng mô hình

Bước 1: Liên kết Google Drive

```
from google.colab import drive
drive.mount('/content/drive')
```

Bước 2: Khai báo các thư viện cần thiết

```
import tensorflow as tf
from tensorflow import keras
import numpy as np
import os
```

```
from PIL import Image
from keras import layers, models
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D,
Rescaling
from keras.models import Sequential, Model
from keras.callbacks import EarlyStopping, ModelCheckpoint,
ReduceLROnPlateau, Callback
from IPython.display import clear_output
import pathlib
```

Bước 3: Lấy dữ liệu từ drive và chia dữ liệu

Dữ liệu nhận dạng tên

```
dataset_url = r"/content/drive/MyDrive/FinalTermProject/train3"
data_dir = pathlib.Path(dataset_url)

batch_size = 32
img_height = 224
img_width = 224

# Tạo file train với tỉ lệ 80% file data
train_ds = tf.keras.utils.image_dataset_from_directory(
  dataset_url,
  validation_split = 0.2,
  subset = "training",
  seed = 125,
  image_size = (img_height, img_width),
  batch_size = batch_size,
  color_mode= "rgb")

# Tạo file test với tỉ lệ 20% file data
val_ds = tf.keras.utils.image_dataset_from_directory(
  dataset_url,
  validation_split = 0.2,
  subset = "validation",
  seed = 125,
  image_size = (img_height, img_width),
  batch_size = batch_size,
  color_mode= "rgb")
```

```
Found 1570 files belonging to 6 classes.
Using 1256 files for training.
Found 1570 files belonging to 6 classes.
Using 314 files for validation.
```

Dữ liệu nhận dạng giới tính

```
dataset_url = r"/content/drive/MyDrive/FinalProject/Gendertrain2"
data_dir = pathlib.Path(dataset_url)

batch_size = 32
img_height = 100
img_width = 100

# Tạo file train với tỉ lệ 80% file data
train_ds = tf.keras.utils.image_dataset_from_directory(
  dataset_url,
  validation_split = 0.2,
  subset = "training",
  seed = 125,
  image_size = (img_height, img_width),
  batch_size = batch_size,
  color_mode= "rgb")

# Tạo file test với tỉ lệ 20% file data
val_ds = tf.keras.utils.image_dataset_from_directory(
  dataset_url,
  validation_split = 0.2,
  subset = "validation",
  seed = 125,
  image_size = (img_height, img_width),
  batch_size = batch_size,
  color_mode= "rgb")
```

```
Found 2004 files belonging to 2 classes.
Using 1604 files for training.
Found 2004 files belonging to 2 classes.
Using 400 files for validation.
```

Vì dữ liệu của em đã được chia thành các thư mục nhỏ với tên các thư mục tương đương với các nhãn trong một thư mục tổn nên em dùng hàm image_dataset_from_ directory để có thể dễ dàng chia cơ sở dữ liệu thành 2 phần cho train với validation. Đồng thời, vì sử dụng hàm trên nên các nhãn sẽ được nhận diện tự động theo tên các thư mục con và ánh xạ chúng cho mỗi hình ảnh.

Bước 4: Kiểm tra tên và nhãn dãn

Dữ liệu nhận dạng tên

```
# lấy tên các thư mục để làm nhãn cho dữ liệu train
class_names = train_ds.class_names
print(class_names)
```

```
# kiểm tra kiểu dữ liệu trong dữ liệu train
for image_batch, labels_batch in train_ds:
  print(image_batch.shape)
  print(labels_batch.shape)
  break
```

```
['Khuong', 'Phuc', 'Thanh', 'Thao', 'TuanKiet', 'VanTrung']

(32, 224, 224, 3)
(32,)
```

Dữ liệu nhận dạng giới tính

```
# lấy tên các thư mục để làm nhãn cho dữ liệu train
class_names = train_ds.class_names
print(class_names)

# kiểm tra kiểu dữ liệu trong dữ liệu train
for image_batch, labels_batch in train_ds:
  print(image_batch.shape)
  print(labels_batch.shape)
  break
```

```
['man', 'woman']

(32, 100, 100, 3)
(32,)
```

Bước 5: Xây dựn mô hình CNN

Mô hình CNN nhận diện tên

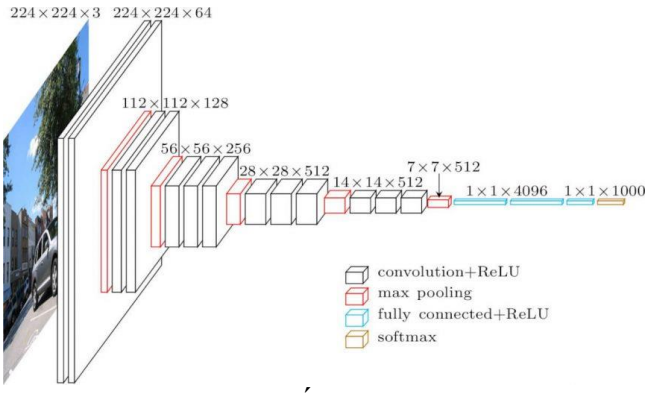
```
num_classes = len(class_names)
# Tạo mô hình Sequential
model = Sequential()
# Chuẩn hóa giá trị pixel từ 0-255 về 0-1
model.add(Rescaling(1./255))

# Tạo lớp Convolution với bộ lọc 3x3, activation là hàm phi tuyến
model.add(Conv2D(64, kernel_size = 3, activation = "relu",
                 input_shape = (224, 224, 3), padding = "same"))
model.add(Conv2D(64, kernel_size = 3, activation = "relu", padding =
"same"))
model.add(MaxPooling2D((2, 2), padding = "same"))

model.add(Conv2D(128, kernel_size = 3, activation = "relu", padding =
"same"))
model.add(Conv2D(128, kernel_size = 3, activation = "relu", padding =
"same"))
model.add(MaxPooling2D((2, 2), padding = "same"))
```

```
model.add(Conv2D(256, kernel_size = 3, activation = "relu", padding =
"same"))
model.add(Conv2D(256, kernel_size = 3, activation = "relu", padding =
"same"))
model.add(Conv2D(256, kernel_size = 3, activation = "relu", padding =
"same"))
model.add(MaxPooling2D((2, 2), padding = "same"))

model.add(Conv2D(512, kernel_size = 3, activation = "relu", padding =
"same"))
model.add(Conv2D(512, kernel_size = 3, activation = "relu", padding =
"same"))
model.add(Conv2D(512, kernel_size = 3, activation = "relu", padding =
"same"))
model.add(MaxPooling2D((2, 2), padding = "same"))


# Chuyển đổi đầu ra các lớp convolution thành mảng 1 chiều
model.add(Flatten())

# Tạo lớp ẩn
model.add(Dense(512, activation = 'relu'))
model.add(Dense(128, activation = 'relu'))
model.add(Dense(num_classes, activation = 'softmax'))

# Biên dịch Mô hình, ở đây sử dụng SparseCategoricalCrossentropy để tính
toán mất mát của mô hình
# dựa trên xác suất dự đoán của mô hình và chỉ số của lớp thực sự của dữ
liệu.
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(),
              metrics=['accuracy'])
```

```
num_classes = len(class_names)
# Tạo mô hình Sequential
model = Sequential()
# Chuẩn hóa giá trị pixel từ 0-255 về 0-1
model.add(Rescaling(1./255))

# Tạo lớp Convolution với bộ lọc 3x3, activation là hàm phi tuyến
model.add(Conv2D(64, kernel_size = 3, activation = "relu",
                 input_shape = (100, 100, 3), padding = "same"))
model.add(Conv2D(64, kernel_size = 3, activation = "relu", padding =
"same"))
model.add(MaxPooling2D((2, 2), padding = "same"))
```

```
model.add(Conv2D(128, kernel_size = 3, activation = "relu", padding =
"same"))
model.add(Conv2D(128, kernel_size = 3, activation = "relu", padding =
"same"))
model.add(MaxPooling2D((2, 2), padding = "same"))

model.add(Conv2D(256, kernel_size = 3, activation = "relu", padding =
"same"))
model.add(Conv2D(256, kernel_size = 3, activation = "relu", padding =
"same"))
model.add(MaxPooling2D((2, 2), padding = "same"))

model.add(Conv2D(512, kernel_size = 3, activation = "relu", padding =
"same"))
model.add(Conv2D(512, kernel_size = 3, activation = "relu", padding =
"same"))
model.add(MaxPooling2D((2, 2), padding = "same"))


# Chuyển đổi đầu ra các lớp convolution thành mảng 1 chiều
model.add(Flatten())

# Tạo lớp ẩn
model.add(Dense(1000, activation = 'relu'))
model.add(Dense(num_classes, activation = 'softmax'))

# Biên dịch Mô hình, ở đây sử dụng SparseCategoricalCrossentropy để tính
toán mất mát của mô hình
# dựa trên xác suất dự đoán của mô hình và chỉ số của lớp thực sự của dữ
liệu.
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(),
              metrics=['accuracy'])
```

Cấu trúc CNN trong đề tài được xây dựng sau khi tham khảo mô hình huấn luyện VGG16 mà cụ thể ta có mô hình VGG16 như sau:



Hình 3.1 Cấu trúc mô hình VGG16

Đối với mô hình nhận diện tên:

- Lớp đầu tiên là lớp Convolution2D với dữ liệu đầu vào có kích thước là (224x224x3), sử dụng 64 bộ lọc (filter). Lớp này có kích thước cửa sổ trượt (Kernel_size) là (3,3) và sử dụng hàm kích hoạt là ReLU
- Lớp tiếp theo tương tự như lớp thứ nhất nhưng không còn thông số kích thước đầu vào.
- Lớp thứ ba là lớp Maxpooling2D, kích thước (pool_size) là (2,2), lớp này có nhiệm vụ giảm kích thước ảnh tuy nhiên vẫn giữ nguyên các đặc trưng của ảnh, từ đó không làm thay đổi đặc tính dữ liệu.
- Lớp thứ ba, tư là lớp Convolution2D sử dụng 128 bộ lọc (filter)., kích thước cửa sổ trượt (kernel_size) là (3,3)
- Lớp thứ năm là lớp Maxpooling2D, kích thước (pool_size) là (2,2)
- Lớp thứ sáu, bảy, tám là lớp Convolution2D sử dụng 256 bộ lọc (filter)., kích thước cửa sổ trượt (kernel_size) là (3,3)
- Lớp thứ chín là lớp Maxpooling2D, kích thước (pool_size) là (2,2)
- Lớp thứ 10, 11, 12 là lớp Convolution2D sử dụng 512 bộ lọc (filter)., kích thước cửa sổ trượt (kernel_size) là (3,3)
- Lớp thứ 13 là lớp Maxpooling2D, kích thước (pool_size) là (2,2)
- Tiếp theo là lớp Falten, lớp này có nhiệm vụ đưa tất cả dữ liệu về kích thước (1,1,Chiều sâu) hay nói cách khác, dữ liệu khi qua lớp này sẽ được trải phẳng và có dạng một chiều.
- Cuối cùng là lớp Fully connected, không có 2 lớp ẩn chứa các giá trị lần lượt là 512 và 128.

Đối với mô hình nhận diện giới tính:

- Lớp đầu tiên là lớp Convolution2D với dữ liệu đầu vào có kích thước là (100x100x3), sử dụng 64 bộ lọc (filter). Lớp này có kích thước cửa sổ trượt (Kernel_size) là (3,3) và sử dụng hàm kích hoạt là ReLU
- Lớp tiếp theo tương tự như lớp thứ nhất nhưng không còn thông số kích thước đầu vào.

- Lớp thứ ba là lớp Maxpooling2D, kích thước (pool_size) là (2,2), lớp này có nhiệm vụ giảm kích thước ảnh tuy nhiên vẫn giữ nguyên các đặc trưng của ảnh, từ đó không làm thay đổi đặc tính dữ liệu.
- Lớp thứ tư, năm là lớp Convolution2D sử dụng 128 bộ lọc (filter)., kích thước cửa sổ trượt (kernel_size) là (3,3)
- Lớp thứ sáu là lớp Maxpooling2D, kích thước (pool_size) là (2,2)
- Lớp thứ bảy, tám là lớp Convolution2D sử dụng 256 bộ lọc (filter)., kích thước cửa sổ trượt (kernel_size) là (3,3)
- Lớp thứ chín là lớp Maxpooling2D, kích thước (pool_size) là (2,2)
- Lớp thứ 10, 11 là lớp Convolution2D sử dụng 512 bộ lọc (filter)., kích thước cửa sổ trượt (kernel_size) là (3,3)
- Lớp thứ 13 là lớp Maxpooling2D, kích thước (pool_size) là (2,2)
- Tiếp theo là lớp Falten, lớp này có nhiệm vụ đưa tất cả dữ liệu về kích thước (1,1,Chiều sâu) hay nói cách khác, dữ liệu khi qua lớp này sẽ được trải phẳng và có dạng một chiều.
- Cuối cùng là lớp Fully connected, không có 2 lớp ẩn chứa các giá trị lần lượt là 512 và 128.

Có một đặc điểm là ở đây, em sử dụng hàm loss = SparseCategoricalCrossentropy(), là vì đây là hàm được sử dụng để tính toán mất mát giữa các dự đoán của mô hình và nhãn thực tế (labels) trong bài toán phân loại đa lớp. Đặc điểm của nó là không yêu cầu biểu diễn one-hot encoded cho nhãn, mà chấp nhận trực tiếp nhãn dưới dạng số nguyên. Khi sử dụng hàm này, các nhãn nguyên (integer labels) được truyền trực tiếp vào mô hình, và hàm SparseCategoricalCrossentropy() sẽ tự động chuyển đổi chúng thành dạng one-hot encoded để tính toán mất mát. Lý do sử dụng là vì, khi chia dữ liệu bằng hàm image_dataset_from_ directory kết quả em nhận được là các nhãn dưới dạng số nguyên.

Các thông số Input, Ouput cụ thể được thể hiện trong bảng thông số mô hình dưới dây:

Mô hình nhận diện tên:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling (Rescaling)       (None, 224, 224, 3)       0
```

```
 conv2d (Conv2D)             (None, 224, 224, 64)      1792

 conv2d_1 (Conv2D)           (None, 224, 224, 64)      36928

 max_pooling2d (MaxPooling2D  (None, 112, 112, 64)     0
 )

 conv2d_2 (Conv2D)           (None, 112, 112, 128)     73856

 conv2d_3 (Conv2D)           (None, 112, 112, 128)     147584

 max_pooling2d_1 (MaxPooling  (None, 56, 56, 128)      0
 2D)

 conv2d_4 (Conv2D)           (None, 56, 56, 256)       295168

 conv2d_5 (Conv2D)           (None, 56, 56, 256)       590080

 conv2d_6 (Conv2D)           (None, 56, 56, 256)       590080

 max_pooling2d_2 (MaxPooling  (None, 28, 28, 256)      0
 2D)

 conv2d_7 (Conv2D)           (None, 28, 28, 512)       1180160

 conv2d_8 (Conv2D)           (None, 28, 28, 512)       2359808

 conv2d_9 (Conv2D)           (None, 28, 28, 512)       2359808

 max_pooling2d_3 (MaxPooling  (None, 14, 14, 512)      0
 2D)

 flatten (Flatten)           (None, 100352)            0

 dense (Dense)               (None, 512)               51380736

 dense_1 (Dense)             (None, 128)               65664

 dense_2 (Dense)             (None, 6)                 774

=================================================================
Total params: 59,082,438
Trainable params: 59,082,438
Non-trainable params: 0
```

Mô hình nhận dạng giới tính

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling (Rescaling)       (None, 224, 224, 3)       0

 conv2d (Conv2D)             (None, 224, 224, 64)      1792

 conv2d_1 (Conv2D)           (None, 224, 224, 64)      36928

 max_pooling2d (MaxPooling2D  (None, 112, 112, 64)     0
```

```
)

conv2d_2 (Conv2D)            (None, 112, 112, 128)     73856

conv2d_3 (Conv2D)            (None, 112, 112, 128)     147584

max_pooling2d_1 (MaxPooling  (None, 56, 56, 128)       0
2D)

conv2d_4 (Conv2D)            (None, 56, 56, 256)       295168

conv2d_5 (Conv2D)            (None, 56, 56, 256)       590080

conv2d_6 (Conv2D)            (None, 56, 56, 256)       590080

max_pooling2d_2 (MaxPooling  (None, 28, 28, 256)       0
2D)

conv2d_7 (Conv2D)            (None, 28, 28, 512)       1180160

conv2d_8 (Conv2D)            (None, 28, 28, 512)       2359808

conv2d_9 (Conv2D)            (None, 28, 28, 512)       2359808

max_pooling2d_3 (MaxPooling  (None, 14, 14, 512)       0
2D)

flatten (Flatten)            (None, 100352)            0

dense (Dense)                (None, 512)               51380736

dense_1 (Dense)              (None, 128)               65664

dense_2 (Dense)              (None, 6)                 774

=================================================================
Total params: 59,082,438
Trainable params: 59,082,438
Non-trainable params: 0
```

Bước 5: Tạo hàm Callback

```
# Tạo hàm callback, Trong đó
# class chứa câu lệnh được thực hiện khi két thúc 1 epoch
class DisplayCallback(Callback):
  def on_epoch_end(self, epoch, logs=None):
    clear_output(wait=True) # Xóa nội dung hiện tại trên console

# EarlyStopping: dừng quá trình train sớm nếu mô hình không cải thiện độ
chính xác trên tập validation trong một số epoch liên tiếp (patience)
# ReduceLROnPlateau: giảm learning rate nếu độ chính xác trên tập
validation không cải thiện trong một số epoch liên tiếp
# ModelCheckpoint: lưu trữ các check point tốt nhất sau mỗi epoch
callbacks = [DisplayCallback(),
             EarlyStopping(patience=10, verbose=1),
```

```
            ReduceLROnPlateau(patience=5, verbose=1),
            ModelCheckpoint('/content/drive/MyDrive/FinalProject/genderd
etect_final_3.h5',verbose=1, save_best_only=True)]
```

Bước 6: Huấn luyện mô hình

```
# train mô hình
history = model.fit(train_ds,
                    validation_data = val_ds,
                    epochs = 100,
                    callbacks = callbacks)
```

Mô hình được huấn luyện với số lần học 100, ta được kết quả như sau:

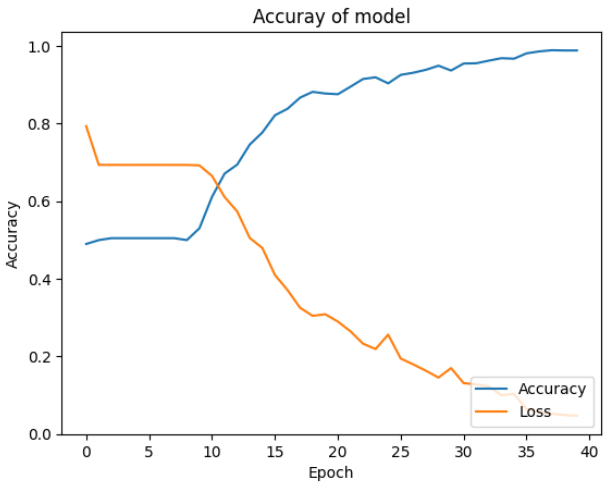
Mô hình nhận diện tên:

```
Epoch 26: val_loss did not improve from 0.00000
40/40 [==============================] - 17s 411ms/step - loss: 6.5203e-08
- accuracy: 1.0000 - val_loss: 4.1761e-09 - val_accuracy: 1.0000 - lr:
1.0000e-05
Epoch 26: early stopping
```

Mô hình nhận diện giới tính:

```
Epoch 40: ReduceLROnPlateau reducing learning rate to 1.0000000656873453e-
06.

Epoch 40: val_loss did not improve from 0.23733
51/51 [==============================] - 5s 93ms/step - loss: 0.0467 -
accuracy: 0.9882 - val_loss: 0.3042 - val_accuracy: 0.8950 - lr: 1.0000e-
05
Epoch 40: early stopping
```

Từ kết quả mô hình huấn luyện, ta có được đồ thị độ chính xác theo số lần lọc:

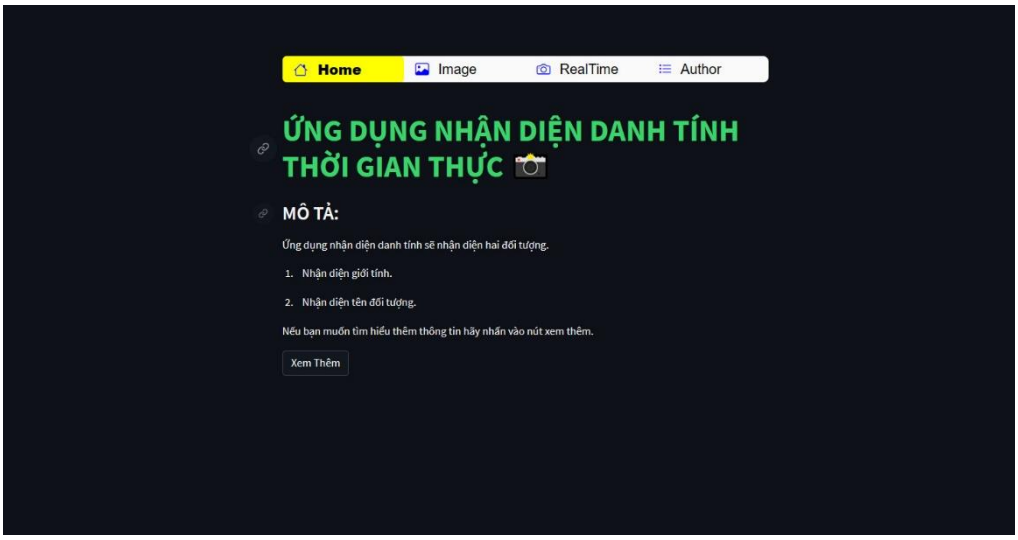


Hình 3.1. Đồ thị độ chính xác của mô hình và giá trị mất mát theo số lần học

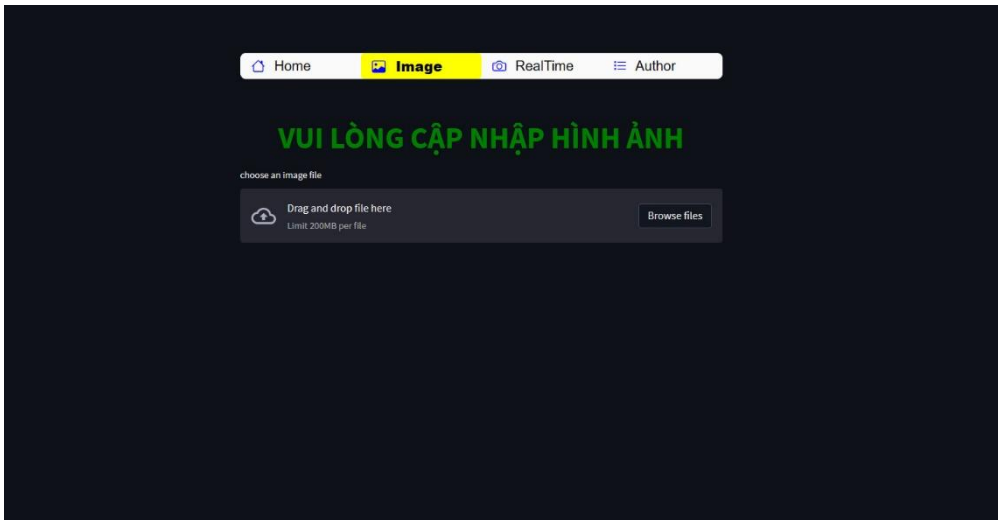
3.2. Xây dựng giao diện với Streamlit

Sau khi đã train xong model, em sẽ lưu model đấy dưới dạng file .h5. Tiếp theo, em xây dựng một giao diện để tương tác với người dùng với thư viện streamlit. Khi hoạt động sẽ lấy mô hình đã được train, tiến hành xử lý hình ảnh và đưa ra dự đoán với các thông số đầu vào như đã training.

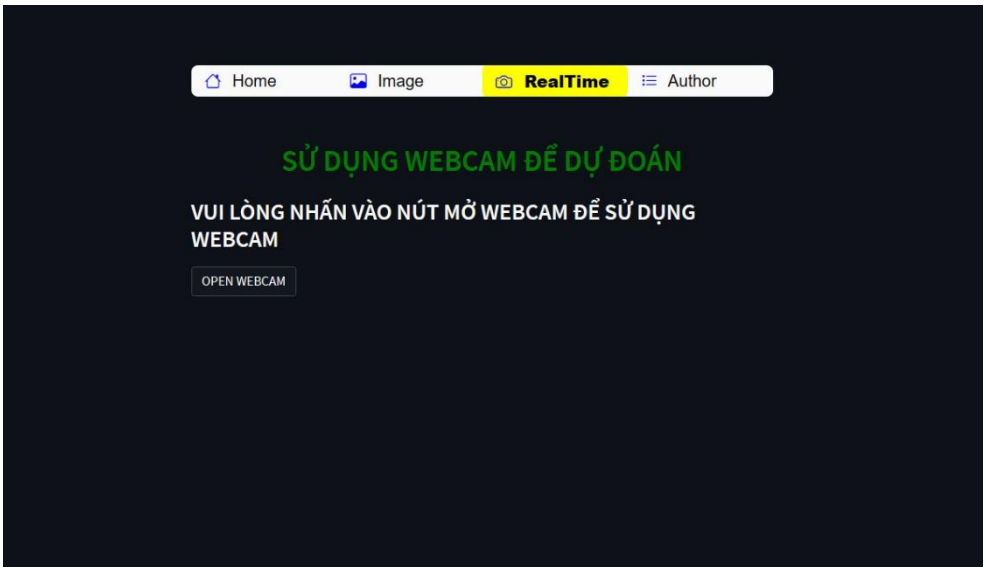
Giao diện của web có 4 taskbar trong đó có task Home, task Image dùng để nhận dạng qua ảnh, task Realtime dùng để nhận diện trong thời gian thực sử dụng webcam và task Author.



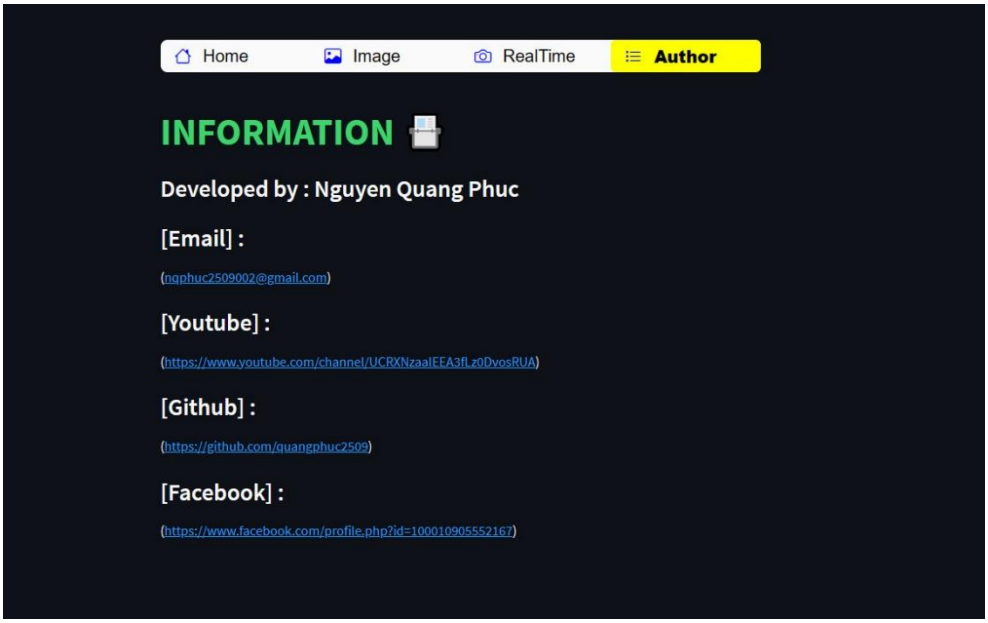
Hình 3.4 Task Home



Hình 3.5 Task Image



Hình 3.6 Task RealTime



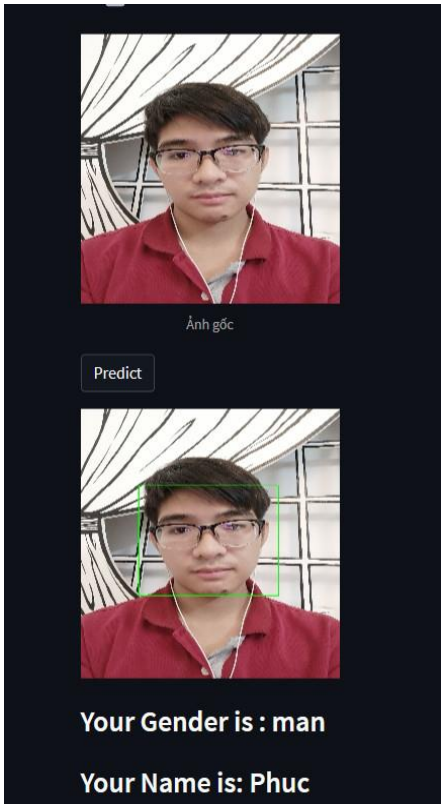
Hình 3.7 Task Author

CHƯƠNG 4. KẾT LUẬN

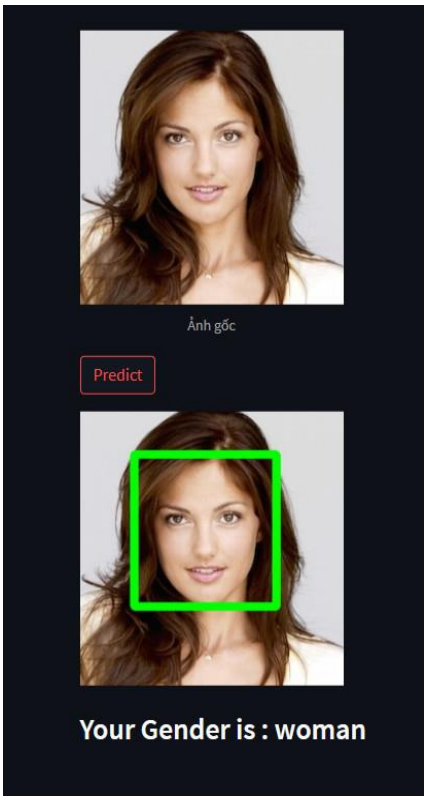
4.1. Kết quả đạt được

- Mô hình nhận dạng tên tuy đó độ chính xác rất cao (99%) nhưng thực tế vẫn nhận diện sai khá nhiều và mô hình nhận dạng giới tính có độ chính xác khoảng 80%

- Kết quả khi nhận dạng ảnh có sẵn

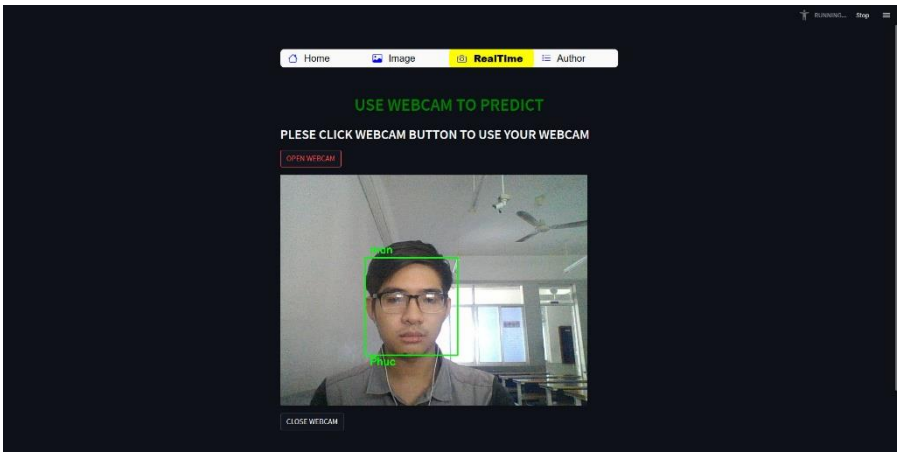


Hình 4.1 Nhận dạng từ ảnh có sẵn



Hình 4.2 Nhận dạng từ ảnh có sẵn

- Kết quả khi nhận dạng thời gian thực (real-time)



Hình 4.3 Nhận dạng thời gian thực

4.2. Nhược điểm

Mặc dù chuẩn đoán mô hình có độ chính xác khá cao nhưng thực tế vẫn còn khá nhiều dự đoán sai. Điều đó đến từ việc dữ liệu thu thập chưa được ổn định và còn ít. Bên cạnh đó, có thể đến từ việc xây dựng mô hình chưa phải là mô hình phù hợp nhất với việc nhận dạng.

4.3. Hướng phát triển

Tiếp tục phát triển mô hình để có thể nâng cao độ chính xác, có thể áp dụng một vài ý kiến sau:

- Tăng cường dữ liệu: Thu thập thêm dữ liệu đồng thời áp dụng các phương pháp tăng cường dữ liệu như xoay, thu phóng, lật ảnh,…

- Sử dụng mô hình phù hợp: xây dựng các mô hình dựa theo cấu trúc các mô hình như ResNet (đem lại độ chính xác khá đáng kể), DenseNet, EffcientNet,…

Xây dựng giao diện mới có khả năng mở khóa bằng khuôn mặt.

TÀI LIỆU THAM KHẢO

[1]. TopDev https://topdev.vn/blog/thuat-toan-cnn-convolutional-neural-network/

[2]. VietNix https://vietnix.vn/cnn-la-gi/

[3]. Thuật toán CNN là gì? Cấu trúc mạng Convolutional Neural Network. [Deep Learning] Tìm hiểu về mang tích chập (CNN) (viblo.asia)

[4]. Thuật toán Convolutional Neural Network (CNN - Phần 1) – Python. https://youtu.be/swguqT77ZLE

[5]. Thuật toán Convolutional Neural Network (CNN - Phần 2) - Python https://youtu.be/BJmAsX_jyj4

[6]. Thuật toán Convolutional Neural Network (CNN - Phần 3) – Python https://youtu.be/mf8YM5XVJw0

PHỤ LỤC

1. Link Datasets:

- Name: https://drive.google.com/drive/folders/1B1hxCSKsE34Y3MyKRbv_xnUC5PyfPfuI?usp=sharing

- Gender: https://drive.google.com/drive/folders/1QGXgDkWNuwSZ0LqgiGWXvWnrNcvcY7pF?usp=sharing

2. link Github: https://github.com/quangphuc2509/Project_Cuoi_Ky

3. link Youtube: https://youtu.be/09RNEb1d5ws