TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
KHOA CƠ KHÍ CHẾ TẠO MÁY

HCMUTE

BÁO CÁO CUỐI KÌ

FACE RECOGNITION TO RETRIEVE INFORMATION

GVHD: **PGS.TS Nguyễn Trường Thịnh**
SVTH: **Đinh Trần Ngọc Thạch**
MSSV: **20146529**
Lớp: **ARIN337629_22_2_09**

TP. Hồ Chí Minh, tháng 5 năm 2023

BẢN NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

Stt	Nội dung thực hiện	Nhận xét

Nhận xét tổng quát:

.....
.....
.....
.....

TÓM TẮT

Deep Learning là thuật toán dựa trên một số ý tưởng từ não bộ tới việc tiếp thu nhiều tầng biểu đạt, cả cụ thể lẫn trừu tượng, qua đó làm rõ nghĩa của các loại dữ liệu. Deep Learning được ứng dụng trong nhận diện hình ảnh, nhận diện giọng nói, xử lý ngôn ngữ tự nhiên. Hiện nay rất nhiều các bài toán nhận dạng sử dụng Deep Learning, vì nó có thể giải quyết các bài toán với số lượng lớn các biến, tham số kích thước đầu vào lớn với hiệu năng cũng như độ chính xác vượt trội so với các phương pháp phân lớp truyền thống, xây dựng những hệ thống thông minh với độ chính xác cao. Trong báo cáo này, chúng tôi nghiên cứu về đề tài "Nhận diện khuôn mặt truy xuất thông tin" theo sự hướng dẫn của PGS.TS Nguyễn Trường Thịnh

MỤC LỤC

CÁC TỪ VIẾT TẮT

ReLu	Rectified Linear Unit
CNN	Convolutional Neural Network
OpenCV	Open Source Computer Vision
CV	Computer Vision
MLL	Machine Learning Library
CONV	Convolution

CHƯƠNG 1: GIỚI THIỆU

1.1. GIỚI THIỆU

CNN là một trong những mô hình mạng Học sâu phổ biến nhất hiện nay, có khả năng nhận dạng và phân loại hình ảnh với độ chính xác rất cao, thậm chí còn tốt hơn con người trong nhiều trường hợp. Mô hình này đã và đang được phát triển, ứng dụng vào các hệ thống xử lý ảnh lớn của Facebook, Google hay Amazon… cho các mục đích khác nhau, như các thuật toán gắn thẻ tự động, tìm kiếm ảnh hoặc gợi ý sản phẩm cho người tiêu dùng.

Mạng CNN với kiến trúc thay đổi, có khả năng xây dựng liên kết chỉ sử dụng một phần cục bộ trong ảnh kết nối đến node trong lớp tiếp theo thay vì toàn bộ ảnh như trong mạng nơ ron truyền thẳng.

Nhờ sự phát triển mạng CNN, chúng tôi có cơ sở thực hiện đề tài "Nhận diện khuôn mặt truy xuất thông tin".

1.2. MỤC TIÊU ĐỀ TÀI

Đề tài "Nhận diện khuôn mặt truy xuất thông tin" có thể áp dụng trong các hệ thống:

- Hệ thống an ninh
- Phân tích dữ liệu

1.3. GIỚI HẠN ĐỀ TÀI

Phạm vi giới hạn của đề tài:

Đề tài chủ yếu sử dụng các thư viện có sẵn trong lập trình python, đồng thời ứng dụng các mô hình toán học trong mạng nơ ron nhân tạo. Điều kiện đầu vào của mô hình là các tệp hình ảnh có giới hạn, đầu ra phân tích khuôn mặt được đưa vào có độ chính xác tương đối. Đối với chức năng thì có sự giới hạn do hiểu biết về lập trình và cách ứng dụng của mô hình vào các hệ thống lớn.

1.4. PHƯƠNG PHÁP NGHIÊN CỨU

Trong phần này, chúng tôi thiết kế mô hình nhận dạng khuôn mặt dựa trên mạng CNN. Dựa trên các phương pháp nghiên cứu chính như phương pháp phân tích, tham khảo tài liệu, phương pháp tổng hợp tài liệu lý thuyết…

1.5. ĐỐI TƯỢNG VÀ PHẠM VI NGHIÊN CỨU

Các đối tượng cần nghiên cứu và phạm vi nghiên cứu có thể giải quyết được đề tài:

- *Đối tượng nghiên cứu:* Trí tuệ nhân tạo, lập trình python.

- *Phạm vi nghiên cứu:* Mạng CNN, các module trong python: opencv, numpy, os, tensorflow, image.

1.6. BỐ CỤC QUYỂN BÁO CÁO

Nội dung báo cáo:

CHƯƠNG 1: GIỚI THIỆU

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

CHƯƠNG 3: XÂY DỰNG CHƯƠNG TRÌNH

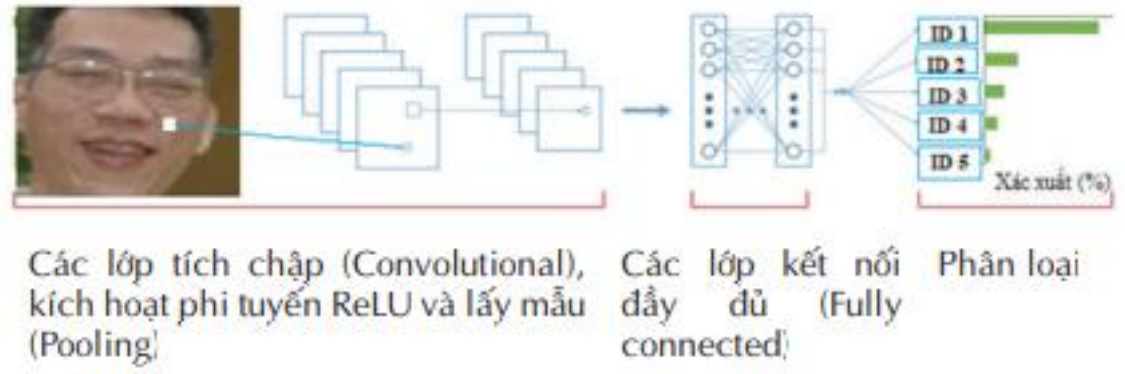
CHƯƠNG 4: KẾT QUẢ THÍ NGHIỆM

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1. GIỚI THIỆU MẠNG CNN

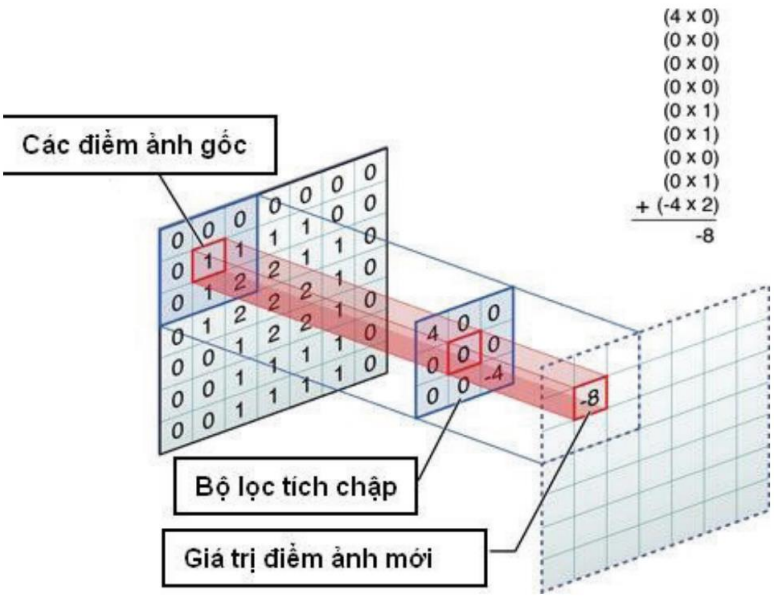
Hình 2.1 trình bày một kiến trúc mạng CNN, các lớp cơ bản trong một mạng CNN bao gồm: lớp tích chập (Convolutional); lớp kích hoạt phi tuyến ReLU (Rectified Linear Unit); lớp lấy mẫu (Pooling); lớp kết nối đầy đủ (Fully connected) được thay đổi về số lượng và cách sắp xếp để tạo ra các mô hình huấn luyện phù hợp cho từng bài toán khác nhau. Các lớp tích chập (Convolutional), kích hoạt phi tuyến ReLU và lấy mẫu (Pooling) Các lớp kết nối đầy đủ (Fully connected) Phân loại Hình 1. Kiến trúc cơ bản của một mạng CNN.

Lớp tích chập: đây là thành phần quan trọng nhất trong mạng CNN, thể hiện sự liên kết cục bộ thay vì kết nối toàn bộ các điểm ảnh. Các liên kết cục bộ được tính toán bằng phép tích chập giữa các giá trị điểm ảnh trong một vùng ảnh cục bộ với các bộ lọc filters có kích thước nhỏ.



Hình 2.1 Kiến trúc cơ bản của mạng CNN

Lớp tích chập: đây là thành phần quan trọng nhất trong mạng CNN, thể hiện sự liên kết cục bộ thay vì kết nối toàn bộ các điểm ảnh. Các liên kết cục bộ được tính toán bằng phép tích chập giữa các giá trị điểm ảnh trong một vùng ảnh cục bộ với các bộ lọc filters có kích thước nhỏ.



Hình 2.2 Bộ lọc tích chập được sử dụng trên ma trận điểm ảnh

Bộ lọc tích chập được sử dụng trên ma trận điểm ảnh. Trong hình 2.2, bộ lọc được sử dụng là một ma trận có kích thước 3x3, bộ lọc này dịch chuyển lần lượt qua từng vùng ảnh đến khi hoàn thành quét toàn bộ bức ảnh, tạo ra một bức ảnh mới có kích thước nhỏ hơn hoặc bằng với kích thước ảnh đầu vào. Kích thước này được quyết định tùy theo kích thước các khoảng trắng được thêm ở viền bức ảnh gốc và được tính theo công thức sau:

$$O = \frac{i + 2 * p - k}{s} + 1 \qquad (2.1)$$

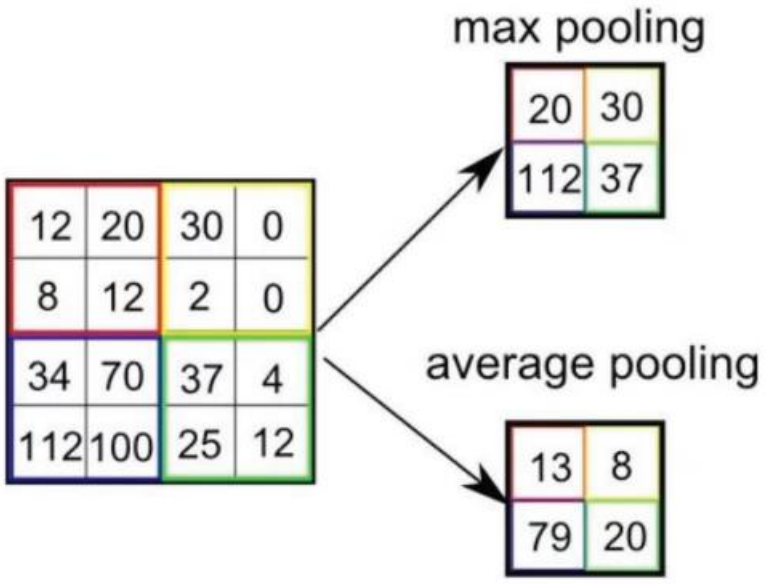
Trong đó: O: kích thước ảnh đầu ra; i: kích thước ảnh đầu vào; p: kích thước khoảng trắng phía ngoài viền của ảnh gốc; k: kích thước bộ lọc; s: bước trượt của bộ lọc. Như vậy, sau khi đưa một bức ảnh đầu vào cho lớp tích chập nhận được kết quả đầu ra là một loạt ảnh tương ứng với các bộ lọc đã được sử dụng để thực hiện phép tích chập. Các trọng số của các bộ lọc này được khởi tạo ngẫu nhiên trong lần đầu tiên và sẽ được cập nhật trong quá trình huấn luyện. - Lớp kích hoạt phi tuyến ReLU: được xây dựng để đảm bảo tính phi tuyến của mô hình huấn luyện sau khi đã thực hiện một loạt các phép tính toán tuyến tính qua các lớp tích chập. Lớp kích hoạt phi tuyến sử dụng các hàm kích hoạt phi tuyến như ReLU hoặc sigmoid, tanh… để giới hạn phạm vi biên độ cho phép của giá trị đầu ra. Trong số

các hàm kích hoạt này, hàm ReLU được chọn do cài đặt đơn giản, tốc độ xử lý nhanh mà vẫn đảm bảo được tính toán hiệu quả. Phép tính toán của hàm ReLU chỉ đơn giản là chuyển tất cả các giá trị âm thành giá trị 0. Lớp ReLU được áp dụng ngay phía sau lớp tích chập, với đầu ra là một ảnh mới có kích thước giống với ảnh đầu vào, các giá trị điểm ảnh cũng hoàn toàn tương tự, trừ các giá trị âm đã bị loại bỏ.

$$f(x) = max(0, x) \qquad (2.2)$$

Lớp lấy mẫu: được đặt sau lớp tích chập và lớp ReLU để làm giảm kích thước ảnh đầu ra trong khi vẫn giữ được các thông tin quan trọng của ảnh đầu vào. Việc giảm kích thước dữ liệu có tác dụng làm giảm được số lượng tham số cũng như tăng hiệu quả tính toán. Lớp lấy mẫu cũng sử dụng một cửa sổ trượt để quét toàn bộ các vùng trong ảnh như lớp tích chập, và thực hiện phép lấy mẫu thay vì phép tích chập, sẽ chọn lưu lại một giá trị duy nhất đại diện cho toàn bộ thông tin của vùng ảnh đó.

Hình 3 thể hiện các phương thức lấy mẫu thường được sử dụng nhất hiện nay, đó là Max Pooling (lấy giá trị điểm ảnh lớn nhất) và Avarage Pooling (lấy giá trị trung bình của các điểm ảnh trong vùng ảnh cục bộ).



Hình 2.3 Phương thức Avarage Pooling và Max Pooling

Như vậy, với mỗi ảnh đầu vào được đưa qua lấy mẫu sẽ thu được một ảnh đầu ra tương ứng, có kích thước giảm xuống đáng kể nhưng vẫn giữ được các đặc trưng cần thiết cho quá trình tính toán và nhận dạng.

Lớp kết nối đầy đủ: được thiết kế tương tự như trong mạng nơ ron truyền thống, tất cả các điểm ảnh được kết nối đầy đủ với node trong lớp tiếp theo.

So với mạng nơ ron truyền thống, các ảnh đầu vào của lớp này đã có kích thước được giảm bớt rất nhiều, đồng thời vẫn đảm bảo các thông tin quan trọng của ảnh cho việc nhận dạng. Do vậy, việc tính toán nhận dạng sử dụng mô hình truyền thẳng đã không còn phức tạp và tốn nhiều thời gian như trong mạng nơ ron truyền thống.

2.2. OPENCV VÀ TINKTER

2.2.1. OpenCV là gì?

Opencv viết tắt của từ Open Source Computer Vision Library.

- OpenCV là một thư viện mã nguồn mở phục vụ cho việc nghiên cứu hay phát triển về thị giác máy tính.
- Tối ưu hóa và xử lý ứng dụng trong thời gian thực.
- Giúp cho việc xây dựng các ứng dụng xử lý ảnh, thị giác máy tính ... một cách nhanhhơn.
- OpenCV có hơn 500 hàm khác nhau, được chia làm nhiều phần phục vụ các công việc như: xử lý ảnh, an ninh, camera quan sát, nhận diện, robot...

Thư viện được viết bằng ngôn ngữ C và C++ có thể chạy trên các hệ điều hành Linux, Window và MacOsX. OpenCV được thiết kế để nâng cao hiệu suất tính toán và nhấn mạnh đến hệ thống thời gian thực. OpenCV đưa ra một hệ thống đơn giản, dễ sử dụng giúp mọi người nhanh chóng xây dựng các ứng dụng trong thị giác máy, kể cả các hệ thống kiểm tra trong nhà máy, bức ảnh trong lĩnh vực y học, bảo mật, robot học... Nó chứa các lập trình xử lý ảnh đơn giản, kể cả thực thi các hàm bậc cao như dò tìm khuôn mặt, theo dõi khuôn mặt, nhận dạng khuôn mặt...OpenCV được giới thiệu vào tháng 1/1999, OpenCV đã được sử dụng trong rất nhiều ứng dụng, các sản phẩm và các nghiên cứu như: trong lĩnh vực hàng

không, sử dụng giảm nhiễu trong y học, phân tích đối tượng, an ninh, hệ thống dò tìm, theo dõi tự động và hệ thống bảo mật...., ngoài ra nó còn được sử dụng trong nhận dạng âm thanh. OpenCV còn là một chìa khóa quan trọng trong các robot sử dụng thị giác như Stanford, Asimo.

2.2.2. Tinkter là gì?

Tinkter là một thư viện Python phổ biến được sử dụng để phát triển giao diện đồ họa người dùng (GUI). Được xây dựng dựa trên Tkinter, một toolkit GUI được cung cấp sẵn trong Python, Tinkter cung cấp một cách đơn giản và linh hoạt để tạo ra các ứng dụng desktop đẹp mắt và tương tác.

Với Tinkter, bạn có thể tạo ra các cửa sổ, các nút nhấn, hộp văn bản, hộp chọn, hình ảnh và nhiều thành phần khác để xây dựng giao diện người dùng tùy chỉnh. Thư viện này cũng cung cấp khả năng xử lý các sự kiện và tương tác người dùng, cho phép bạn phản hồi và điều khiển ứng dụng của mình dễ dàng.

Một trong những điểm mạnh của Tinkter là sự tương thích rộng rãi trên các nền tảng, bao gồm cả Windows, macOS và Linux. Điều này cho phép bạn phát triển ứng dụng GUI đa nền tảng mà không cần thay đổi mã nguồn quá nhiều.

Dù Tinkter không có nhiều tính năng phức tạp như một số thư viện GUI khác, nhưng nó vẫn là một lựa chọn tốt cho việc tạo ra các ứng dụng đơn giản và hiệu quả trong Python. Với cú pháp dễ sử dụng và tài liệu phong phú, Tinkter đã trở thành một công cụ phổ biến trong cộng đồng lập trình Python để phát triển các giao diện người dùng hấp dẫn.

CHƯƠNG 3: XÂY DỰNG CHƯƠNG TRÌNH

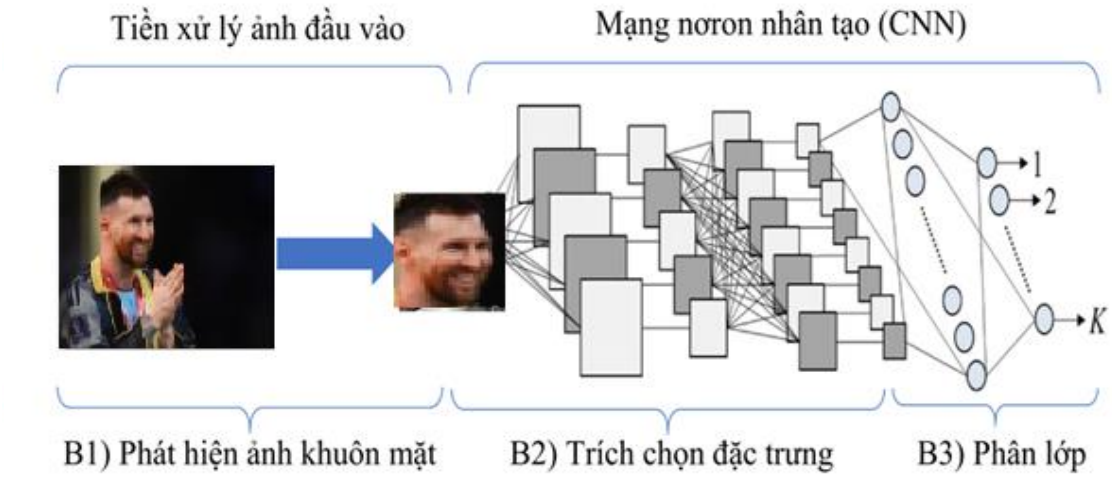
3.1. XÂY DỰNG MÔ HÌNH

Mô hình nhận dạng được chia thành 3 bước chính (Hình 3.1), bao gồm:

Bước 1: Tạo tập dữ liệu.

Bước 2: Huấn luyện dữ liệu đưa vào và trích chọn các đặc trưng.

Bước 3: Phân loại ảnh khuôn mặt dựa trên đặc trưng được trích chọn và đưa ra kết quả.



Hình 3.1 Sơ đồ quy trình của mô hình nhận dạng khuôn mặt

3.2. CÁC BƯỚC THỰC HIỆN

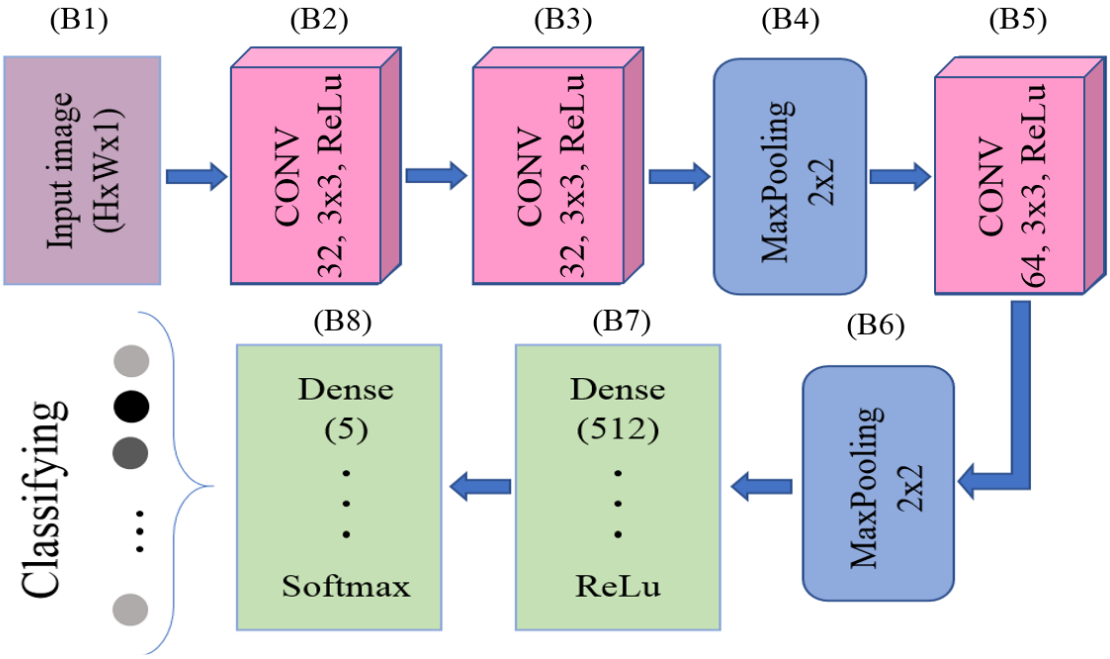
3.2.1. Tạo tập dữ liệu

Các dữ liệu dùng để huấn luyện lấy từ ảnh có sẵn. Lấy ảnh từ internet về lưu vào các tệp để chuẩn bị cho việc training sau này.

3.2.2. Huấn luyện dữ liệu đưa vào Colab

Mô hình CNN được thiết kế gồm hai phần chức năng là trích chọn đặc trưng của ảnh khuôn mặt và phân lớp đối tượng dựa trên đặc trưng đã chọn. Mô hình CNN bao gồm nhiều lớp, số lớp nơron và độ lớn (số nơron) của mỗi lớp ảnh hưởng đến chất lượng cũng như độ phức tạp trong tính toán của mạng nơron. Các nghiên

cứu thường điều chỉnh hai yếu tố này tuỳ theo bài toán ứng dụng để đạt được chất lượng mong muốn và đồng thời đảm bảo sự phức tạp tính toán chấp nhận được.



Hình 3.2 Kiến trúc dạng khối của mô hình CNN

Mỗi lớp nơron trong mô hình CNN lấy một mảng nhiều chiều gồm các số làm đầu vào và tạo ra một mảng số nhiều chiều khác ở đầu ra (sau đó trở thành đầu vào của lớp tiếp theo). Khi phân loại hình ảnh khuôn mặt, đầu vào của lớp nơron đầu tiên là kích thước hình ảnh đầu vào. Kích thước đầu ra của lớp cuối cùng là tập hợp các khả năng của các lớp khác nhau được phân loại cho mỗi ảnh đầu vào. Cả ba loại lớp nơron để xây dựng kiến trúc của CNN bao gồm: 2 lớp tích chập (CONV2D), 2 lớp nơron Maxpooling2D và 2 lớp nơron kết nối đầy đủ để phân loại (gọi là lớp Dense). Mỗi lớp CONV2D được kết nối theo sau nó bởi một lớp Maxpooling, áp dụng cơ chế kích hoạt LeakyReLU (Leaky Rectified Linear Unit, mặc định là max(x,0)) sau mỗi lớp CONV để đảm bảo đầu vào không âm cho lớp nơron kế tiếp. Theo nguyên tắc xếp chồng các lớp nơron và giảm không gian mẫu (downsampling) tại các kết quả đầu ra của chúng, CNN thực hiện trích xuất các đặc trưng ngày càng trừu tượng và phức tạp hơn, đồng thời, là bất biến đối với các phép biến dạng và chuyển đổi.

3.2.3. Thiết lập GUI

Thiết kế GUI bằng Tinkter sử dụng một số lệnh đơn giản để thiết kế.

Tổng thể, ứng dụng GUI này cho phép người dùng chọn một ảnh từ máy tính, sau đó sử dụng mô hình đã được tải từ file để dự đoán tên người, tuổi, nghề nghiệp và quốc tịch của người trong ảnh. Kết quả dự đoán được hiển thị trên giao diện người dùng.

CHƯƠNG 4: KẾT QUẢ THÍ NGHIỆM

4.1. KẾT QUẢ SAU KHI HUẤN LUYỆN

Quá trình huấn luyện tập dữ liệu:

```
Model: "sequential_13"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_26 (Conv2D)          (None, 300, 300, 32)      896

 leaky_re_lu_36 (LeakyReLU)  (None, 300, 300, 32)      0

 max_pooling2d_26 (MaxPoolin  (None, 150, 150, 32)     0
 g2D)

 conv2d_27 (Conv2D)          (None, 150, 150, 64)      18496

 leaky_re_lu_37 (LeakyReLU)  (None, 150, 150, 64)      0

 max_pooling2d_27 (MaxPoolin  (None, 75, 75, 64)       0
 g2D)

 flatten_13 (Flatten)        (None, 360000)            0

 dense_26 (Dense)            (None, 512)               184320512

 leaky_re_lu_38 (LeakyReLU)  (None, 512)               0

 dense_27 (Dense)            (None, 2)                 1026

=================================================================
Total params: 184,340,930
Trainable params: 184,340,930
Non-trainable params: 0
```

Hình 4.1 Hơn 184 triệu trọng số được huấn luyện

```
start training
Epoch 1/20
2/2 [==============================] - 4s 1s/step - loss: 63.5981 - accuracy: 0.4500
Epoch 2/20
2/2 [==============================] - 1s 183ms/step - loss: 4.5830 - accuracy: 0.5000
Epoch 3/20
2/2 [==============================] - 1s 902ms/step - loss: 1.1834 - accuracy: 0.7750
Epoch 4/20
2/2 [==============================] - 1s 947ms/step - loss: 0.9630 - accuracy: 0.7500
Epoch 5/20
2/2 [==============================] - 1s 856ms/step - loss: 0.6408 - accuracy: 0.7250
Epoch 6/20
2/2 [==============================] - 1s 180ms/step - loss: 0.3356 - accuracy: 0.8750
Epoch 7/20
2/2 [==============================] - 1s 831ms/step - loss: 0.6645 - accuracy: 0.6750
Epoch 8/20
2/2 [==============================] - 1s 173ms/step - loss: 0.3890 - accuracy: 0.8250
Epoch 9/20
2/2 [==============================] - 1s 817ms/step - loss: 0.4545 - accuracy: 0.8000
Epoch 10/20
2/2 [==============================] - 2s 266ms/step - loss: 0.4490 - accuracy: 0.7250
Epoch 11/20
2/2 [==============================] - 2s 1s/step - loss: 0.3014 - accuracy: 0.8750
Epoch 12/20
2/2 [==============================] - 1s 826ms/step - loss: 0.2991 - accuracy: 0.9000
Epoch 13/20
2/2 [==============================] - 1s 152ms/step - loss: 0.3050 - accuracy: 0.8250
Epoch 14/20
2/2 [==============================] - 1s 179ms/step - loss: 0.2737 - accuracy: 0.9250
Epoch 15/20
2/2 [==============================] - 1s 853ms/step - loss: 0.2361 - accuracy: 0.9500
Epoch 16/20
2/2 [==============================] - 2s 175ms/step - loss: 0.1812 - accuracy: 0.9750
Epoch 17/20
2/2 [==============================] - 1s 179ms/step - loss: 0.1323 - accuracy: 0.9750
Epoch 18/20
2/2 [==============================] - 1s 171ms/step - loss: 0.1097 - accuracy: 0.9500
Epoch 19/20
2/2 [==============================] - 1s 795ms/step - loss: 0.2058 - accuracy: 0.9500
Epoch 20/20
2/2 [==============================] - 1s 908ms/step - loss: 0.2858 - accuracy: 0.9250
```

Hình 4.2 Độ chính và giá trị hàm lỗi theo từng epoch

*Nhận xét:

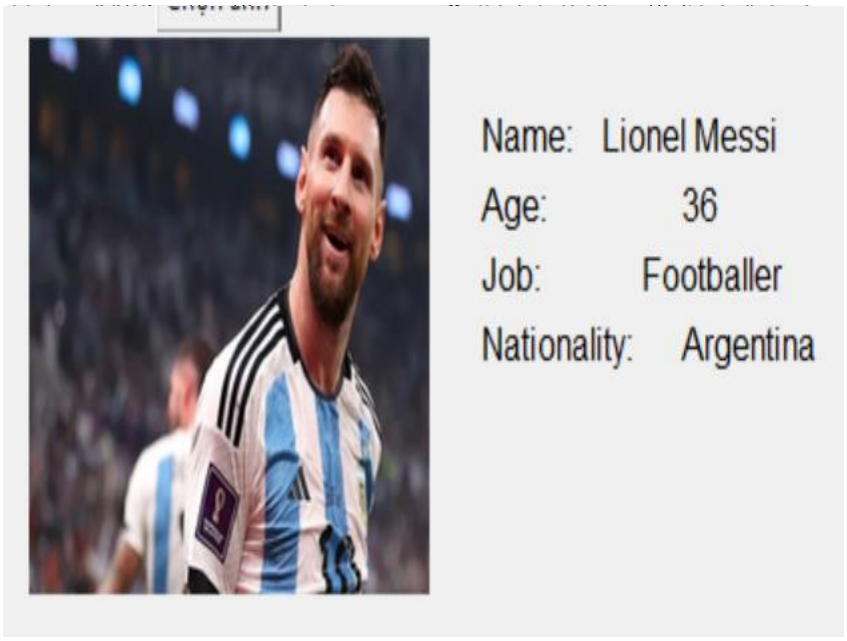
- Giá trị của hàm lỗi giảm dần theo từng epoch. Giá trị của hàm lỗi ở epoch 1 là 62.5981, giá trị hàm lỗi ở epoch 20 giảm xuống còn 0.2858.

- Độ chính xác tăng dần qua từng epoch, tăng lên tối đa 92.5% ở epoch thứ 20.

4.2. ĐƯA HÌNH ẢNH MỘT NGƯỜI CẦN NHẬN DIỆN

- So sánh kết quả trước và sau khi nhận diện 1 một khuôn mặt:

Hình 4.3 Hình ảnh trước khi nhận diện



Hình 4.4 Hình ảnh sau khi nhận diện (Messi)

* Nhận xét:

- Đối với nhận diện một khuôn mặt, kết quả cho ra là chính xác đồng thời phát hiện được khuôn mặt cần nhận diện.

- Tuy nhiên vẫn có nhiều sai sót trong lúc dư đoán

KẾT LUẬN

Trong báo cáo này, chúng tôi đã đề xuất một mô hình dựa trên mạng nơron tích chập (CNN) để nhận dạng khuôn mặt con người. Mô hình này có 2 lớp nơron tích chập (Convolution) và 2 lớp nơron liên kết đầy đủ (Fully Connected), tổng số tham số là khoảng hơn 184 triệu.

Như vậy, có thể khẳng định mô hình của chúng tôi có độ phức tạp ở mức vừa phải, phù hợp với các hệ thống xử lý ở mức trung bình và đem lại tiềm năng khả thi trong ứng dụng thực tiễn.

Mặc dù độ phức tạp của mô hình ở mức thấp so với các mô hình khác, nhưng kết quả thử nghiệm cho thấy tính hiệu quả của phân lớp khá cao. Hiện nay do điều kiện tính toán nên chỉ áp dụng số lần huấn luyện còn thấp, nếu được huấn luyện ở mức độ sâu hơn thì kỳ vọng sẽ đem lại kết quả cao hơn nữa.

Để phát triển thêm cho mô hình, chúng tôi sẽ tìm hiểu và thiết kế một hệ thống thu thập dữ liệu hình ảnh để tạo bộ dữ liệu huấn luyện đa dạng cho mô hình, từ đó xây dựng một ứng dụng cho bài toán thực tiễn như hệ thống điểm danh sinh viên có mặt ở lớp học, hệ thống giám sát cán bộ vào/ra cổng cơ quan, hệ thống theo dõi và định danh liên tục quá trình học tập của người học trực tuyến.

PHỤ LỤC

Chương trình được thực hiện trên môi trường Python và Colab với 2 chương trình như sau:

+ Tạo tập dữ liệu:

```
import matplotlib.pyplot as plt
from keras.layers import LeakyReLU
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout, Conv2D, MaxPooling2D, Flatten
from keras.utils import to_categorical
from keras.models import load_model
from keras.utils import load_img, img_to_array
import os
from keras.preprocessing.image import ImageDataGenerator
```

```
train_data=ImageDataGenerator(rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True)
```

```
train_d=train_data.flow_from_directory('/content/drive/MyDrive/AI cuối kỳ/train',
                 target_size=(300,300), batch_size=32, class_mode='categorical')
```

```
train_d.class_indices
```

```
class1=2
```

- Để tạo tập dữ liệu huấn luyện lưu các hình ảnh thu thập vào drive

- + Huấn luyện tập dữ liệu:

```
model = Sequential()
model.add(Conv2D(32,kernel_size=(3,3), activation='relu', input_shape=(300,300, 3), padding='same'))
model.add(LeakyReLU(alpha=0.1))
model.add(MaxPooling2D(2,2))

model.add(Conv2D(64,kernel_size=(3,3), activation='relu', padding='same'))
model.add(LeakyReLU(alpha=0.1))
model.add(MaxPooling2D(2,2))

model.add(Flatten())
model.add(Dense(512,activation='relu'))
model.add(LeakyReLU(alpha=0.1))
model.add(Dense(class1,activation='softmax'))

model.compile(loss='categorical_crossentropy',optimizer='Adam',metrics=['accuracy'])
model.summary()

print("start training")
train=model.fit(train_d,epochs=20,batch_size=128,verbose=1)
```

```
Score=model.evaluate(train_d,verbose=0)
print('Train Loss', Score[0])
print('Train Accuracy', Score[1])
```

```
Train Loss 0.08733129501342773
Train Accuracy 0.9750000238418579
```

```
model.save("/content/drive/MyDrive/AI cuối kỳ/khuonmat.h5")
```

```
model_1=load_model('/content/drive/MyDrive/AI cuối kỳ/khuonmat.h5')
```

+ Nhận diện chạy GUI:

```
from tkinter import *
from tkinter import filedialog
import cv2
from PIL import ImageTk, Image
from tkinter import Tk, Label, Button, filedialog
from keras.models import load_model
from keras.utils import load_img, img_to_array
import numpy as np

# Load model từ drive
model_1 = load_model('khuonmat.h5')
class_names = ['Lionel Messi', 'Truong Giang']
class_age=['36','40']
class_job=['Footballer','Artist']
class_nn=['Argentina','Vietnam']
# Build GUI
th = Tk()
th.title("Face Recognition")
th.geometry('450x450')

lbl = Label(th, text="Final Project AI", fg='red', font=("Arial", 25))
lbl.place(x=120, y=0)
lbl1 = Label(th, text="SVTH: ", fg='black')
lbl1.place(x=40, y=50)
lbl3 = Label(th, text="Dinh Tran Ngoc Thach", fg='black')
lbl3.place(x=120, y=50)
lbl2 = Label(th, text="MSSV:", fg='black')
lbl2.place(x=40, y=75)
lbl4 = Label(th, text="20146529", fg='black')
lbl4.place(x=150, y=75)
lbl5 = Label(th, text="Instructors: ", fg='black')
lbl5.place(x=40, y=100)
lbl6 = Label(th, text="PGT.TS Nguyen Truong Thinh", fg='black')
lbl6.place(x=120, y=100)

# Chức năng chọn ảnh
def select_image():
    # Chọn file ảnh từ máy tính
    file_path = filedialog.askopenfilename()

    # Load ảnh
    img = load_img(file_path, target_size=(300, 300))
    img = img_to_array(img)
    img = img.astype('float32')
    img = img / 255
```

```
    img = np.expand_dims(img, axis=0)

    # Dự đoán kết quả
    result = model_1.predict(img).argmax()
    person_name = class_names[result]
    person_job=class_job[result]
    person_nn=class_nn[result]
    person_age=class_age[result]
    # Thay đổi ảnh hiển thị
    image = Image.open(file_path)
    image = image.resize((200, 200), Image.ANTIALIAS)
    image_tk = ImageTk.PhotoImage(image)
    label_img.config(image=image_tk)
    label_img.image = image_tk

    # Cập nhật tên người
    label_name.config(text=person_name)
    label_name1.config(text='Name:')
    label_age.config(text=person_age)
    label_age1.config(text='Age:')
    label_job.config(text=person_job)
    label_job1.config(text='Job:')
    label_nn.config(text=person_nn)
    label_nn1.config(text='Nationality:')
# Nút "Chọn ảnh"
button_select = Button(th, text="Select Picture", command=select_image)
button_select.place(x=90, y=175)

# Hiển thị ảnh mặc định
default_image_path = 'GetArticleImage.jpg'
default_image = Image.open(default_image_path)
default_image = default_image.resize((200, 200), Image.ANTIALIAS)
default_image_tk = ImageTk.PhotoImage(default_image)
label_img = Label(th, image=default_image_tk)
label_img.place(x=25, y=200)

# Hiển thị tên người
label_name = Label(th, text="", font=("Arial", 12))
label_name.place(x=310, y=225)
label_name1 = Label(th, text="", font=("Arial", 12))
label_name1.place(x=250, y=225)
label_age = Label(th, text="", font=("Arial", 12))
label_age.place(x=350, y=250)
label_age1 = Label(th, text="", font=("Arial", 12))
label_age1.place(x=250, y=250)
label_job = Label(th, text="", font=("Arial", 12))
label_job.place(x=330, y=275)
label_job1 = Label(th, text="", font=("Arial", 12))
```

```
label_job1.place(x=250, y=275)
label_nn = Label(th, text="", font=("Arial", 12))
label_nn.place(x=350, y=300)
label_nn1 = Label(th, text="", font=("Arial", 12))
label_nn1.place(x=250, y=300)
th.mainloop()
```

TÀI LIỆU THAM KHẢO

[1] PGS. TS Trương Ngọc Sơn, *Trí tuệ nhân tạo cơ sở và ứng dụng,* Đại học quốc gia TPHCM, 2020.

[2] Tài liệu online.

[3] Q. Zhang, M. Zhang, T. Chen, Z. Sun, Y. Ma, and B. Yu, "Recent advances in convolutional neural network acceleration," *Neurocomputing*, vol. 323, pp. 0–38, 2019, doi: 10.1016/j.neucom.2018.09.038.