BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM

KHOA ĐÀO TẠO CHẤT LƯỢNG CAO

HCMUTE

BÁO CÁO TRÍ TUỆ NHÂN TẠO

NHẬN DIỆN CÁC ĐIỂM MỐC TRÊN MŨI

Mã môn: ARIN337629_21_2_02CLC

GVHD: PGS. TS Nguyễn Trường Thịnh

SVTH: Nguyễn Thái Bình – 19146050

TP. Thủ Đức, ngày 21 tháng 06 năm 2022

MỤC LỤC

PHỤ LỤC HÌNH ẢNH

LỜI CẢM ƠN

Trước tiên, em xin gửi lời cảm ơn chân thành đến các giảng viên của trường Đại học Sư phạm kỹ thuật Thành phố Hồ Chí Minh đã tạo điều kiện thuận lợi cho em trong suốt quá trình học tập và rèn luyện.

Em xin gửi lời cảm ơn đến các cán bộ của Thư viện trường đã hỗ trợ tận tình cũng như tạo điều kiện cho em trong việc tìm kiếm tư liệu nghiên cứu để hoàn thành đề tài môn học này.

Tiếp đến, nguồn động viên tinh thần lớn lao để nhóm có động lực theo đuổi và hoàn thành đề tài báo cáo là nhờ sự ủng hộ của gia đình và sự giúp đỡ, đoàn kết của bạn bè.

Đặc biệt trong quá trình thực hiện, từ những lúc thiếu kinh nghiệm, em đã gặp rất nhiều khó khăn nhưng với sự hướng dẫn tận tình, chu đáo và theo dõi sát sao, đầy tinh thần, trách nhiệm của thầy Nguyễn Trường Thịnh và anh Minh Triều, đã giúp em có những kinh nghiệm quý báu, những kiến thức bổ ích để hoàn thành tốt báo cáo.

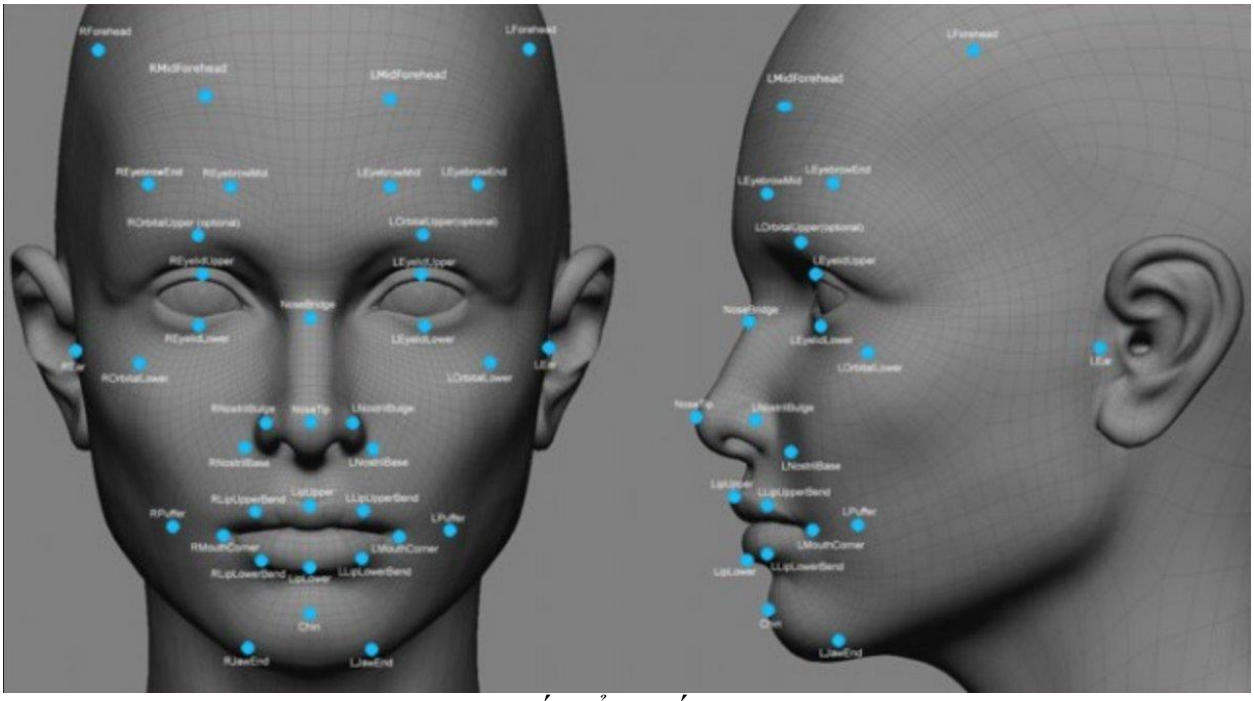
Mặc dù đã cố gắng hoàn thành mô hình, mô phỏng tương đối hoàn chỉnh nhưng không tránh khỏi những thiếu sót. Rất mong nhận được sự góp ý, đánh giá của quý thầy cô để em có thể sửa chữa và hoàn thành tốt hơn.

Lời cuối cùng, kính chúc quý thầy cô, bạn bè cũng như gia đình tràn đầy sức khỏe, bình an.

Dưới đây là bài báo cáo của em vì thời gian học tập và thực hiện báo cáo có hạn, có thể em cũng chưa thật sự nắm rõ một cách sâu sắc nhất, cũng như là số lượng kiến thức của em còn hạn chế nên bài báo cáo sẽ không tránh khỏi sai sót. Mong các thầy cô thông cảm và em mong nhận được sự góp ý của thầy cô để hoàn thiện hơn.

CHƯƠNG 1: GIỚI THIỆU VỀ ĐỀ TÀI

1.1. Tổng quan về đề tài.



Hình 1. Một số điểm mốc trên khuôn mặt

Nhận diện điểm mốc của mũi là bài toán xác định được vị trí mặt người sau đó phát hiện các điểm mốc của mũi trên khuôn mặt và theo dõi chúng kể cả có sự thay đổi tư thế đầu hay nét mặt.

Nhận diện khuôn mặt là một hệ thống tự động xác định và nhận dạng một người dựa trên một bức ảnh kỹ thuật số hoặc một đoạn video từ một nguồn video. Có thể hiểu đơn giản, hệ thống này so sánh các đặc điểm, thông số của một cơ sở dữ liệu về khuôn mặt với một khuôn mặt được chọn trước từ hình ảnh. Một khi khuôn mặt được phát hiện, nó có thể được tìm kiếm các điểm mốc như mắt và mũi.

1.2. Mục đích và phạm vi nghiên cứu.

Đề tài "Nhận diện các điểm mốc trên mũi" được nghiên cứu nhằm giúp cho sinh viên hiểu rõ về các phương pháp học và thuật toán vào việc xây dựng một mô hình nhận diện, từ đó xác định các điểm mốc của khuôn mặt (điểm mốc của mũi) để phân

biệt khuôn mặt của những người khác nhau hoặc phát triển Trí tuệ nhân tạo trong việc chỉnh sửa ảnh chân dung người.

Đề tài được thực hiện với phạm vi nghiên cứu, học hỏi được thực hiện trên nền tảng Google Colab gồm:

Nghiên cứu tổng quan về đề tài.

Phát hiện khuôn mặt.

Xác định điểm mốc của mũi trên khuôn mặt đã được phát hiện.

Kiểm nghiệm bằng phương pháp realtime hoặc video.

1.3. Ý nghĩa thực tiễn.

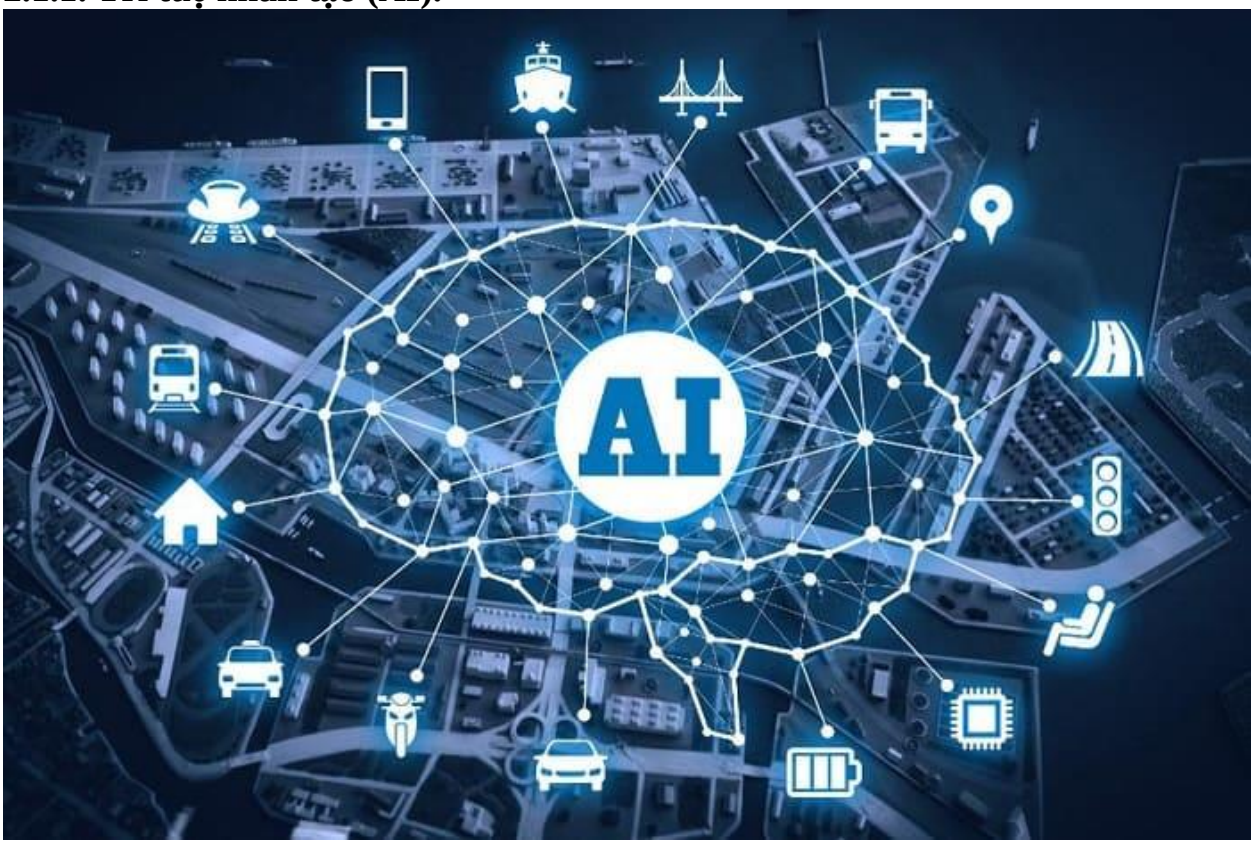
Đề tài tạo tiền đề cho việc phát triển thành sản phẩm ứng dụng vào việc ứng dụng Trí tuệ nhân tạo trong chỉnh sửa ảnh chân dụng hoặc nhận dạng phân biệt người qua điểm mốc của mũi trên khuôn mặt người.

Góp phần xây dựng thêm kiến thức trong học tập cũng như tài liệu tham khảo cho các bạn sinh viên muốn tìm hiểu về Trí tuệ nhân tạo nói chung và nhận diện các điểm mốc nói riêng.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1. Sơ lược về Trí tuệ nhân tạo.

2.1.1. Trí tuệ nhân tạo (AI).



Hình 2. Trí tuê nhân tạo (AI)

Trí tuệ nhân tạo được viết tắt là AI (Arifical Intelligence) là trí thông minh được thể hiện bằng máy móc, trái ngược với trí thông minh tự nhiên của con người. Thông thường, thuật ngữ "trí tuệ nhân tạo" thường được sử dụng để mô tả các máy móc (hoặc máy tính) có khả năng bắt chước các chức năng "nhận thức" mà con người thường phải liên kết với tâm trí, như "học tập" và "giải quyết vấn đề".

Con người thường sử dụng các phán đoán nhanh và trực quan chứ không phải là phép khấu trừ từng bước mà các nghiên cứu AI ban đầu có thể mô phỏng. AI đã tiến triển bằng cách sử dụng cách giải quyết vấn đề "biểu tượng phụ": cách tiếp cận tác nhân được thể hiện nhấn mạnh tầm quan trọng của các kỹ năng cảm biến động đến

lý luận cao hơn; nghiên cứu mạng thần kinh cố gắng để mô phỏng các cấu trúc bên trong não làm phát sinh kỹ năng này.

Trong thế kỷ 21, các kỹ thuật AI đã trải qua sự hồi sinh sau những tiến bộ đồng thời về sức mạnh máy tính, dữ liệu lớn và hiểu biết lý thuyết; và kỹ thuật AI đã trở thành một phần thiết yếu của ngành công nghệ, giúp giải quyết nhiều vấn đề thách thức trong học máy, công nghệ phần mềm và nghiên cứu vận hành.

2.1.2. Học sâu (Machine learning).

Học máy (Machine learning) là một lĩnh vực của trí tuệ nhân tạo liên quan đến việc nghiên cứu và xây dựng các kĩ thuật cho phép các hệ thống "học" tự động từ dữ liệu để giải quyết những vấn đề cụ thể. Ví dụ như các máy có thể "học" cách phân loại thư điện tử xem có phải thư rác (spam) hay không và tự động xếp thư vào thư mục tương ứng.

Học máy vẫn đòi hỏi sự đánh giá của con người trong việc tìm hiểu dữ liệu cơ sở và lựa chọn các kĩ thuật phù hợp để phân tích dữ liệu. Đồng thời, trước khi sử dụng, dữ liệu phải sạch, không có sai lệch và không có dữ liệu giả.

Các mô hình học máy yêu cầu lượng dữ liệu đủ lớn để "huấn luyện" và đánh giá mô hình. Trước đây, các thuật toán học máy thiếu quyền truy cập vào một lượng lớn dữ liệu cần thiết để mô hình hóa các mối quan hệ giữa các dữ liệu. Sự tăng trưởng trong dữ liệu lớn (big data) đã cung cấp các thuật toán học máy với đủ dữ liệu để cải thiện độ chính xác của mô hình và dự đoán.

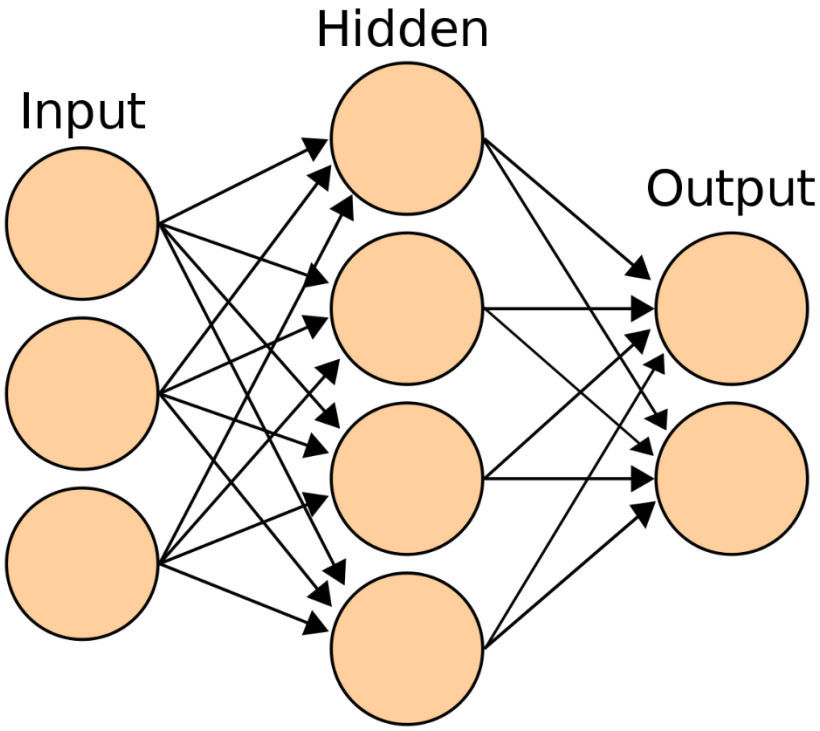
2.1.3. Học sâu (Deep learning).

Học sâu (Deep learning) là một nhánh máy học sử dụng nhiều lớp Neurol network để đưa ra một mô hình toán học trên dữ liệu có sẵn. Học sâu hay Deep Learning thường được nhắc đến cùng với Dữ liệu lớn (Big Data) và Trí tuệ nhân tạo (AI). Đã có nhiều ứng dụng trong thực tế, đang phát triển mạnh theo sự phát triển của tốc độ máy tính đặc biệt là khả năng tính toán trên GPU và sự tăng nhanh của dữ liệu cùng

với các framework (TensorFlow hay Pytorch) làm việc xây dựng model trở nên dễ dàng hơn.

Học sâu đã phát triển cùng với thời đại kĩ thuật số, điều này đã mang lại sự bùng nổ dữ liệu dưới mọi hình thức và từ mọi khu vực trên thế giới. Dữ liệu này, gọi đơn giản là dữ liệu lớn, được lấy từ các nguồn như phương tiện truyền thông xã hội, công cụ tìm kiếm trên internet, nền tảng thương mại điện tử hoặc rạp chiếu phim trực tuyến, ...

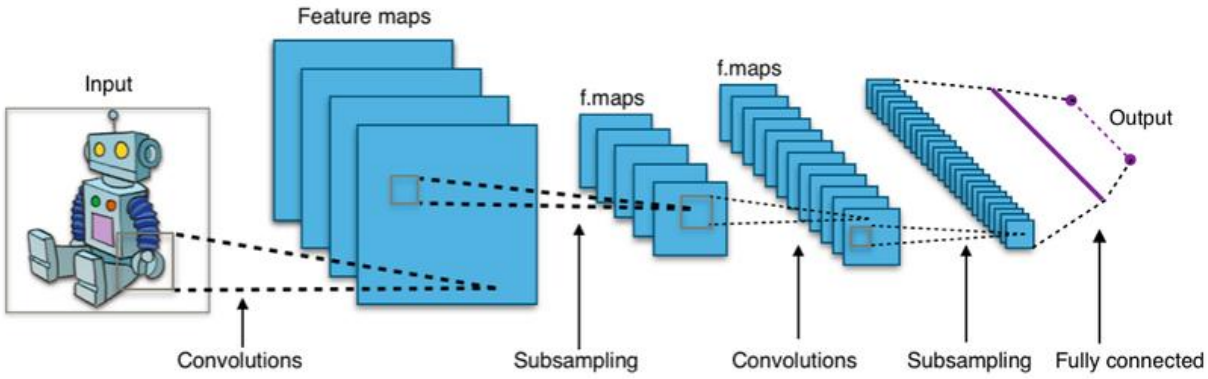
2.2. Mạng nơ-ron tích chập (CNN)



Hình 3. Convolution Neural Network

Mạng nơ-ron tích chập (Convolutional Neural Network) là một trong những mô hình Deep Learning tiên tiến. Nó giúp cho chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao như hiện nay. Áp dụng phổ biến nhất để phân tích hình ảnh trực quan. Mạng còn được gọi là shift invariant (dịch chuyển bất biến) hay mạng thần kinh nhân tạo không gian bất biến (SIANN), dựa trên kiến trúc trọng số được chia sẻ và các đặc tính đối xứng tịnh tiến (translational symmetry).

Đầu tiên, các trọng số cho mỗi filter (kernel) phải giống nhau. Tất cả các nơ-ron trong lớp ẩn đầu sẽ phát hiện chính xác feature tương tự chỉ ở các vị trí khác nhau trong hình ảnh đầu vào. Chúng ta gọi việc map từ input layer sang hidden layer là một feature map. Một convolutional layer bao gồm các feature map khác nhau. Mỗi một feature map giúp detect một vài feature trong bức ảnh. Lợi ích lớn nhất của trọng số chia sẻ là giảm tối đa số lượng tham số trong mạng CNN.



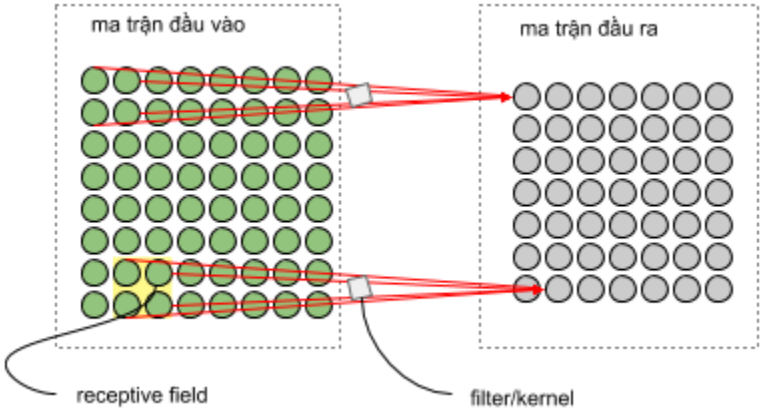
Hình 4. Các lớp trong CNN

2.3. Các lớp cơ bản của mạng CNN.

2.3.1. Convolution Layer.

Convolution layer là lớp quan trọng nhất và cũng là lớp đầu tiên của của mô hình CNN. Lớp này có chức năng chính là phát hiện các đặc trưng có tính không gian hiệu quả. Trong tầng này có 4 đối tượng chính là: ma trận đầu vào, bộ filters, và receptive field, feature map. Conv layer nhận đầu vào là một ma trận 3 chiều và một bộ filters cần phải học. Bộ filters này sẽ trượt qua từng vị trí trên bức ảnh để tính tích chập (convolution) giữa bộ filter và phần tương ứng trên bức ảnh. Phần tưng ứng này trên bức ảnh gọi là receptive field, tức là vùng mà một neuron có thể nhìn thấy để đưa ra quyết định, và mà trận cho ra bới quá trình này được gọi là feature map. Để hình dung, các bạn có thể tưởng tượng, bộ filters giống như các tháp canh trong nhà tù quét lần lượt qua không gian xung quanh để tìm kiếm tên tù nhân bỏ trốn. Khi phát hiện tên tù nhân bỏ trốn, thì chuông báo động sẽ reo lên,

giống như các bộ filters tìm kiếm được đặc trưng nhất định thì tích chập đó sẽ cho giá trị lớn.



Hình 5. Phương pháp tích chập

2.3.2. Nonlinear Layer.

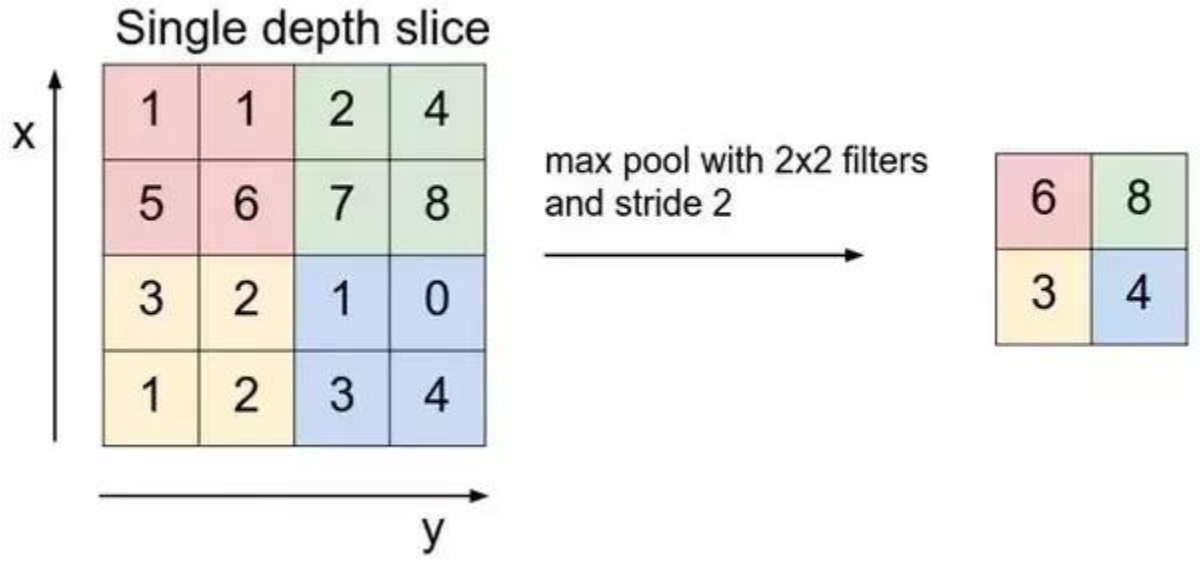
ReLU (Rectified Linear Units, f = max (0, x)) là hàm kích hoạt phổ biến nhất cho CNN tại thời điểm của bài viết, được giới thiệu bởi Geoffrey E. Hinton năm 2010. Trước khi hàm ReLU được áp dụng thì những hàm như sigmoid hay tanh mới là những hàm được sử dụng phổ biến. Hàm ReLU được ưa chuộng vì tính 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.

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.

2.3.3. Pooling Layer.

Sau hàm kích hoạt, thông thường chúng ta sử dụng tầng pooling. Một số loại pooling layer phổ biến như là max-pooling, average pooling, với chức năng chính là

giảm chiều của tầng trước đó. Với một pooling có kích thước 2x2, các bạn cần phải trược filter 2x2 này trên những vùng ảnh có kích thước tương tự rồi sau đó tính max, hay average cho vùng ảnh đó.



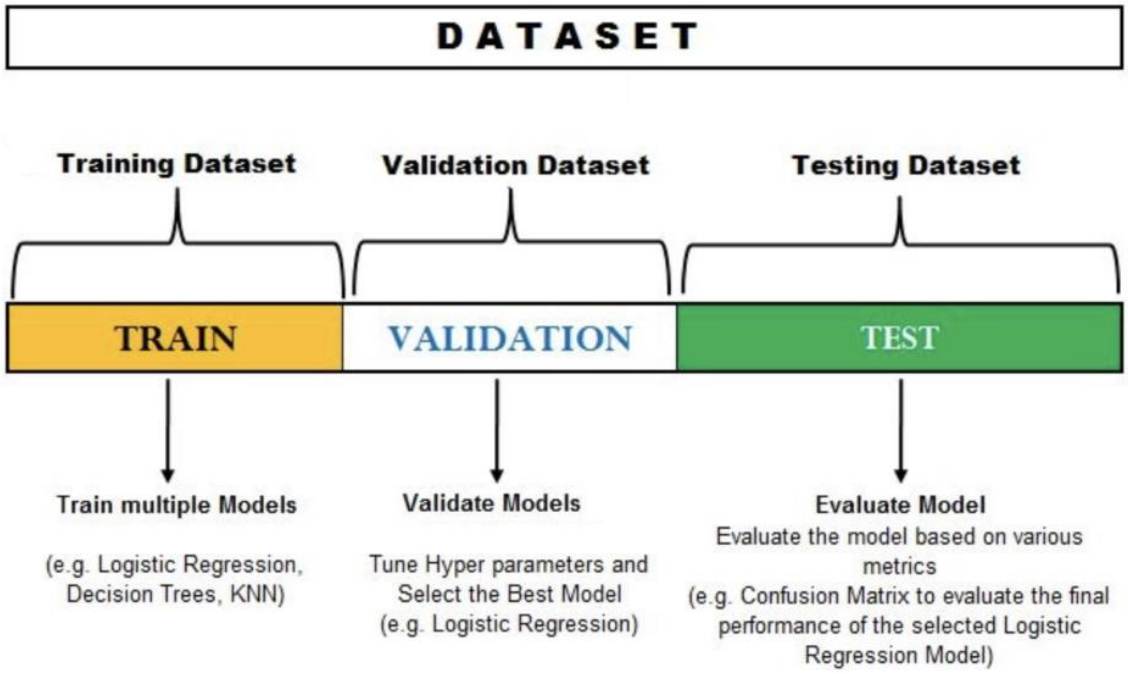
Hình 6. Phương pháp pooling

Ý tương đằng sau tầng pooling là vị trí tuyết đối của những đặc trưng trong không gian ảnh không còn cần cần thiết, thay vào đó vị trí tương đối giữ các đặc trưng đã đủ để phân loại đối tượng. Hơn giảm tầng pooling có khả năng giảm chiều cực kì nhiều, làm hạn chế overfit, và giảm thời gian huấn luyện tốt.

CHƯƠNG 3: QUY TRÌNH THỰC HIỆN ĐỀ TÀI

3.1. Thu thập và xử lý dữ liệu.

Thu thập thông tin là quá trình tập hợp thông tin theo những tiêu chí cụ thể nhằm làm rõ những vấn đề, nội dung liên quan đến lĩnh vực nhất định. Thu thập thông tin là quá trình xác định nhu cầu thông tin, tìm nguồn thông tin, thực hiện tập hợp thông tin theo yêu cầu nhằm đáp ứng mục tiêu đã được định trước.



Hình 7. Các loại dữ liệu

Dữ liệu đào tạo (training data) được sử dụng để xây dựng thuật toán học máy. Các nhà khoa học dữ liệu sẽ cung cấp dữ liệu đầu vào cho thuật toán, tương ứng với đầu ra dự kiến. Dựa trên đó, mô hình sẽ đánh giá dữ liệu nhiều lần để tìm hiểu thêm về đặc tính của dữ liệu và tiếp tục tự điều chỉnh nhằm hướng tới mục đích đã định.

Dữ liệu đánh giá (validation data): Trong quá trình đào tạo, dữ liệu đánh giá sẽ cung cấp thêm dữ liệu mới vào mô hình, nhằm xác định khả năng dự đoán của thuật toán đối với những dữ liệu chưa từng nhìn thấy trước đó. Thực tế, không phải tất cả các nhà khoa học dữ liệu đều sử dụng dữ liệu đánh giá, nhưng nó có thể cung cấp một số thông tin hữu ích để tối ưu hóa các tham số cũng như hiệu suất của thuật toán.

Dữ liệu thử nghiệm (testing data): Dữ liệu thử nghiệm được đưa vào sau khi xây dựng mô hình, nhằm xác nhận hiệu quả của thuật toán. Nếu dữ liệu đào tạo và dữ liệu đánh giá đều đi kèm các nhãn nhằm theo dõi các chỉ số hiệu suất thì dữ liệu thử nghiệm phải được bỏ nhãn. Điều này nhằm đưa mô hình vào môi trường kiểm tra thực tế và xác nhận lần cuối rằng nó có thể hoạt động hiệu quả.

Đề tài sử dụng bộ dữ liệu csv có sẵn được thừa hưởng trên Kaggle với hơn 7000 dữ liệu mỗi điểm mốc và có đến 15 điểm mốc trên cả khuôn mặt. Tuy nhiên với bộ dữ liệu khá lớn và phong phú nhưng chưa phù hợp hoàn toàn với đề tài nên ta cần phải xử lý dữ liệu. Xử lý dữ liệu là hoạt động phân tích, phân loại thông tin theo các nguyên tắc và phương pháp nhất định, trên cơ sở đó đưa ra các biện pháp giải quyết công việc. Là quá trình đối chiếu, chọn lọc, chỉnh lý, biên tập thông tin theo mục đích, yêu cầu xác định. Đây là công việc bắt buộc nhằm nâng cao chất lượng và hiệu quả sử dụng thông tin, tránh sự quá tải, nhiễu thông tin.

```
[ ] Train_Dir = '/content/drive/MyDrive/Ai_project/AI/training.csv'
    Test_Dir = '/content/drive/MyDrive/Ai_project/AI/test.csv'
    lookid_dir = '/content/drive/MyDrive/Ai_project/AI/IdLookupTable.csv'
    train_data = pd.read_csv(Train_Dir)
    test_data = pd.read_csv(Test_Dir)
    lookid_data = pd.read_csv(lookid_dir)
    os.listdir('/content/drive/MyDrive/Ai_project/AI/input')

    []
```

```
[ ] train_data.describe()
```

	left_eye_center_x	left_eye_center_y	right_eye_center_x	right_eye_center_y	left_eye_inner_corner_x	left_eye_inner_corner_y
count	7039.000000	7039.000000	7036.000000	7036.000000	2271.000000	2271.000000
mean	66.359021	37.651234	30.306102	37.976943	59.159339	37.944752
std	3.448233	3.152926	3.083230	3.033621	2.690354	2.307332
min	22.763345	1.616512	0.686592	4.091264	19.064954	27.190098
25%	65.082895	35.900451	28.783339	36.327681	58.039339	36.626283
50%	66.497566	37.528055	30.251378	37.813273	59.304615	37.878040
75%	68.024752	39.258449	31.768334	39.566729	60.519810	39.260348
max	94.689280	80.502649	85.039381	81.270911	84.440991	66.562559

8 rows × 30 columns

Hình 8. Dữ liệu đã được thu thập

https://github.com/NgThaiBinh/Nose_Keypoint_Detecion.git

```
[ ] train_data.isnull().any().value_counts()
```

```
True     28
False     3
dtype: int64
```

```
[ ] train_data.fillna(method = 'ffill',inplace = True)
    # train_data.reset_index(drop = True,inplace = True)
```

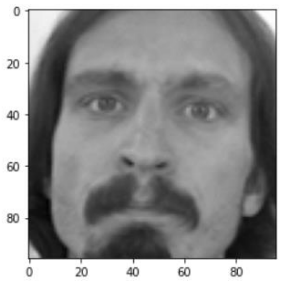
```
[ ] train_data.isnull().any().value_counts()
```

```
False    31
dtype: int64
```

```
[ ] imag = []
    for i in range(0,len(train_data['Image'])-1):
        img = train_data['Image'][i].split(' ')
        img = ['0' if x == '' else x for x in img]
        imag.append(img)
```

```
[ ] image_list = np.array(imag, dtype = 'float')
    X_train = image_list.reshape(-1,96,96,1)
```

```
[ ] plt.imshow(X_train[0].reshape(96,96),cmap='gray')
    plt.show()
```




```
[ ] training = train_data.drop('Image',axis = 1)

    y_train = []
    for i in range(0,len(train_data['Image'])-1):
        y = training.iloc[i,:]
        y_train.append(y)
    y_train = np.array(y_train,dtype = 'float')
```

```
[ ] print(y_train[0])
```

```
[66.03356391 39.00227368 30.22700752 36.4216782  59.58207519 39.64742256
 73.13034586 39.96999699 36.35657143 37.3894015  23.45287218 37.3894015
 56.95326316 29.03364812 80.22712782 32.22813835 40.22760902 29.0023218
 16.35637895 29.64747068 44.42057143 57.06680301 61.19530827 79.97016541
 28.61449624 77.38899248 43.3126015  72.93545865 43.13070677 84.48577444]
```

Hình 9. Dữ liệu đã được xử lý

3.2. Xây dựng mô hình.

Mô hình học máy là một biểu hiện của một thuật toán quét qua hàng núi dữ liệu để tìm ra các mẫu hình hoặc đưa ra dự đoán. Được cung cấp dữ liệu, các mô hình machine learning là "động cơ" toán học của trí tuệ nhân tạo.

https://github.com/NgThaiBinh/Nose_Keypoint_Detecion.git

```
[ ]  model = Sequential()

     model.add(Convolution2D(32, (3,3), padding='same', use_bias=False, input_shape=(96,96,1)))
     model.add(LeakyReLU(alpha = 0.1))
     model.add(BatchNormalization())

     model.add(Convolution2D(32, (3,3), padding='same', use_bias=False))
     model.add(LeakyReLU(alpha = 0.1))
     model.add(BatchNormalization())
     model.add(MaxPool2D(pool_size=(2, 2)))

     model.add(Convolution2D(64, (3,3), padding='same', use_bias=False))
     model.add(LeakyReLU(alpha = 0.1))
     model.add(BatchNormalization())

     model.add(Convolution2D(64, (3,3), padding='same', use_bias=False))
     model.add(LeakyReLU(alpha = 0.1))
     model.add(BatchNormalization())
     model.add(MaxPool2D(pool_size=(2, 2)))

     model.add(Convolution2D(96, (3,3), padding='same', use_bias=False))
     model.add(LeakyReLU(alpha = 0.1))
     model.add(BatchNormalization())

     model.add(Convolution2D(96, (3,3), padding='same', use_bias=False))
     model.add(LeakyReLU(alpha = 0.1))
     model.add(BatchNormalization())
     model.add(MaxPool2D(pool_size=(2, 2)))
```

```
[ ]  model.add(Convolution2D(128, (3,3),padding='same', use_bias=False))
     model.add(LeakyReLU(alpha = 0.1))
     model.add(BatchNormalization())

     model.add(Convolution2D(128, (3,3),padding='same', use_bias=False))
     model.add(LeakyReLU(alpha = 0.1))
     model.add(BatchNormalization())
     model.add(MaxPool2D(pool_size=(2, 2)))

     model.add(Convolution2D(256, (3,3),padding='same',use_bias=False))
     model.add(LeakyReLU(alpha = 0.1))
     model.add(BatchNormalization())

     model.add(Convolution2D(256, (3,3),padding='same',use_bias=False))
     model.add(LeakyReLU(alpha = 0.1))
     model.add(BatchNormalization())
     model.add(MaxPool2D(pool_size=(2, 2)))

     model.add(Convolution2D(512, (3,3), padding='same', use_bias=False))
     model.add(LeakyReLU(alpha = 0.1))
     model.add(BatchNormalization())

     model.add(Convolution2D(512, (3,3), padding='same', use_bias=False))
     model.add(LeakyReLU(alpha = 0.1))
     model.add(BatchNormalization())


     model.add(Flatten())
     model.add(Dense(512,activation='relu'))
     model.add(Dropout(0.1))
     model.add(Dense(30))
     model.summary()
```

Hình 10. Xây dựng mô hình

https://github.com/NgThaiBinh/Nose_Keypoint_Detecion.git

Ảnh được đưa vào có kích thước 96x96 và có 3 kênh màu, ta sử dụng 32 kernel với bộ lọc ma trận 3x3 để trượt qua ảnh gốc giúp làm nổi bật các đặc trưng. Khi một số neuron sẽ chỉ cho ra giá trị là 0 trong quá trình trainning, để giải quyết vấn đề này thì ta dùng các lớp LeakyReLU. Tiếp đến là lớp Batch Normalization giúp chuẩn hóa dữ liệu về một khoảng nào đó, từ đó giảm thiểu overfiting. Ngoài ra, chúng ta sẽ không cần phải sử dụng quá nhiều dropput và điều này rất có ý nghĩa vì chúng ta sẽ không cần phải lo lắng vì bị mất quá nhiều thông tin khi dropout weigths của mạng. Ta sử dụng thêm lớp MaxPool2D để chọn ra những đặc trưng nổi bật làm tăng độ chính xác cũng như làm giảm kích thước ảnh.

Tiếp đến là lớp Dropout, cái tên đã nói lên tất cả lớp này dùng để ẩn 1 số đơn vị trong lớp ẩn trong quá trình huấn luyện mô hình để tránh việc overfitting.

Cuối cùng là Flatten để biến outputs của ảnh lúc này là một ma trận cho quá trình tiến hành phân loại đầu vào.

```
[ ] Model: "sequential"
    _________________________________________________________________
     Layer (type)                Output Shape              Param #
    =================================================================
     conv2d (Conv2D)             (None, 96, 96, 32)        288

     leaky_re_lu (LeakyReLU)     (None, 96, 96, 32)        0

     batch_normalization (BatchN  (None, 96, 96, 32)       128
     ormalization)

     conv2d_1 (Conv2D)           (None, 96, 96, 32)        9216

     leaky_re_lu_1 (LeakyReLU)   (None, 96, 96, 32)        0

     batch_normalization_1 (Batc  (None, 96, 96, 32)       128
     hNormalization)

     max_pooling2d (MaxPooling2D  (None, 48, 48, 32)       0
     )

     conv2d_2 (Conv2D)           (None, 48, 48, 64)        18432

     leaky_re_lu_2 (LeakyReLU)   (None, 48, 48, 64)        0

     batch_normalization_2 (Batc  (None, 48, 48, 64)       256
     hNormalization)

     conv2d_3 (Conv2D)           (None, 48, 48, 64)        36864

     leaky_re_lu_3 (LeakyReLU)   (None, 48, 48, 64)        0

     batch_normalization_3 (Batc  (None, 48, 48, 64)       256
     hNormalization)
```

```
max_pooling2d_1 (MaxPooling  (None, 24, 24, 64)       0
2D)

conv2d_4 (Conv2D)           (None, 24, 24, 96)        55296

leaky_re_lu_4 (LeakyReLU)   (None, 24, 24, 96)        0

batch_normalization_4 (Batc  (None, 24, 24, 96)       384
hNormalization)

conv2d_5 (Conv2D)           (None, 24, 24, 96)        82944

leaky_re_lu_5 (LeakyReLU)   (None, 24, 24, 96)        0

batch_normalization_5 (Batc  (None, 24, 24, 96)       384
hNormalization)

max_pooling2d_2 (MaxPooling  (None, 12, 12, 96)       0
2D)

conv2d_6 (Conv2D)           (None, 12, 12, 128)       110592

leaky_re_lu_6 (LeakyReLU)   (None, 12, 12, 128)       0

batch_normalization_6 (Batc  (None, 12, 12, 128)      512
hNormalization)

conv2d_7 (Conv2D)           (None, 12, 12, 128)       147456

leaky_re_lu_7 (LeakyReLU)   (None, 12, 12, 128)       0

batch_normalization_7 (Batc  (None, 12, 12, 128)      512
hNormalization)

 max_pooling2d_3 (MaxPooling  (None, 6, 6, 128)        0
 2D)

 conv2d_8 (Conv2D)           (None, 6, 6, 256)         294912

 leaky_re_lu_8 (LeakyReLU)   (None, 6, 6, 256)         0

 batch_normalization_8 (Batc  (None, 6, 6, 256)        1024
 hNormalization)

 conv2d_9 (Conv2D)           (None, 6, 6, 256)         589824

 leaky_re_lu_9 (LeakyReLU)   (None, 6, 6, 256)         0

 batch_normalization_9 (Batc  (None, 6, 6, 256)        1024
 hNormalization)

 max_pooling2d_4 (MaxPooling  (None, 3, 3, 256)        0
 2D)

 conv2d_10 (Conv2D)          (None, 3, 3, 512)         1179648

 leaky_re_lu_10 (LeakyReLU)  (None, 3, 3, 512)         0

 batch_normalization_10 (Bat  (None, 3, 3, 512)        2048
 chNormalization)

 conv2d_11 (Conv2D)          (None, 3, 3, 512)         2359296

 leaky_re_lu_11 (LeakyReLU)  (None, 3, 3, 512)         0

 batch_normalization_11 (Bat  (None, 3, 3, 512)        2048
 chNormalization)

 flatten (Flatten)           (None, 4608)              0

 dense (Dense)               (None, 512)               2359808

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 30)                15390

=================================================================
Total params: 7,268,670
Trainable params: 7,264,318
Non-trainable params: 4,352
```

Hình 11. Mô hình sau khi được xây dựng

3.3. Huấn luận mô hình.

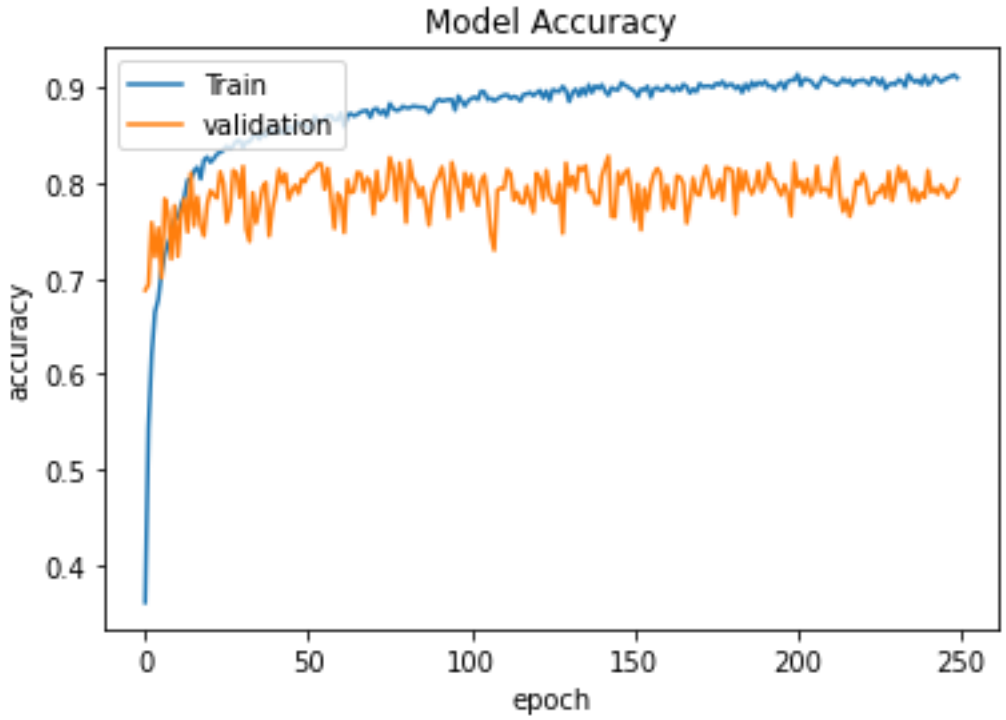
```
[13] model.compile(optimizer='adam',loss='mean_squared_error',metrics=['accuracy'])

[14] history=model.fit(X_train,y_train,epochs = 250,batch_size = 32,validation_split = 0.2)

Epoch 1/250
177/177 [==============================] - 22s 45ms/step - loss: 66.9142 - accuracy: 0.3606 - val_loss: 43.4006 - val_accuracy: 0.6879
Epoch 2/250
177/177 [==============================] - 7s 39ms/step - loss: 16.6431 - accuracy: 0.5433 - val_loss: 13.9089 - val_accuracy: 0.6936
Epoch 3/250
177/177 [==============================] - 7s 39ms/step - loss: 14.4999 - accuracy: 0.6208 - val_loss: 8.9564 - val_accuracy: 0.7589
Epoch 4/250
177/177 [==============================] - 7s 39ms/step - loss: 11.6524 - accuracy: 0.6653 - val_loss: 4.1049 - val_accuracy: 0.7227
Epoch 5/250
177/177 [==============================] - 7s 39ms/step - loss: 11.2438 - accuracy: 0.6781 - val_loss: 8.7269 - val_accuracy: 0.7532
Epoch 6/250
177/177 [==============================] - 7s 40ms/step - loss: 10.2205 - accuracy: 0.7066 - val_loss: 4.4489 - val_accuracy: 0.7007
Epoch 7/250
177/177 [==============================] - 7s 40ms/step - loss: 9.6508 - accuracy: 0.7251 - val_loss: 10.4382 - val_accuracy: 0.7837
Epoch 8/250
177/177 [==============================] - 7s 40ms/step - loss: 8.4901 - accuracy: 0.7373 - val_loss: 4.1353 - val_accuracy: 0.7695
Epoch 9/250
177/177 [==============================] - 7s 40ms/step - loss: 9.6794 - accuracy: 0.7449 - val_loss: 9.2898 - val_accuracy: 0.7206
Epoch 10/250
177/177 [==============================] - 7s 41ms/step - loss: 7.6775 - accuracy: 0.7620 - val_loss: 10.3806 - val_accuracy: 0.7766
Epoch 11/250
177/177 [==============================] - 7s 41ms/step - loss: 7.9222 - accuracy: 0.7643 - val_loss: 4.1349 - val_accuracy: 0.7234
Epoch 12/250
177/177 [==============================] - 7s 40ms/step - loss: 6.9214 - accuracy: 0.7769 - val_loss: 16.9990 - val_accuracy: 0.7681
Epoch 13/250
177/177 [==============================] - 7s 40ms/step - loss: 6.7299 - accuracy: 0.7859 - val_loss: 2.8622 - val_accuracy: 0.7837
Epoch 14/250
177/177 [==============================] - 7s 40ms/step - loss: 6.2952 - accuracy: 0.8033 - val_loss: 3.7558 - val_accuracy: 0.7489
Epoch 15/250
177/177 [==============================] - 7s 40ms/step - loss: 3.0767 - accuracy: 0.8952 - val_loss: 1.8408 - val_accuracy: 0.8071
Epoch 133/250
177/177 [==============================] - 7s 40ms/step - loss: 3.0399 - accuracy: 0.8985 - val_loss: 1.9407 - val_accuracy: 0.7922
Epoch 134/250
177/177 [==============================] - 7s 40ms/step - loss: 3.2032 - accuracy: 0.8915 - val_loss: 2.2142 - val_accuracy: 0.8170
Epoch 135/250
177/177 [==============================] - 7s 40ms/step - loss: 3.0186 - accuracy: 0.8994 - val_loss: 4.2755 - val_accuracy: 0.8135
Epoch 136/250
177/177 [==============================] - 7s 40ms/step - loss: 3.1805 - accuracy: 0.8929 - val_loss: 2.2465 - val_accuracy: 0.8184
Epoch 137/250
177/177 [==============================] - 7s 40ms/step - loss: 3.0668 - accuracy: 0.9032 - val_loss: 3.8286 - val_accuracy: 0.7901
Epoch 138/250
177/177 [==============================] - 7s 40ms/step - loss: 3.2020 - accuracy: 0.8950 - val_loss: 2.3459 - val_accuracy: 0.7915
Epoch 139/250
177/177 [==============================] - 7s 41ms/step - loss: 3.3347 - accuracy: 0.9021 - val_loss: 3.6602 - val_accuracy: 0.7787
Epoch 140/250
177/177 [==============================] - 7s 40ms/step - loss: 3.1747 - accuracy: 0.8991 - val_loss: 2.3053 - val_accuracy: 0.7787
Epoch 141/250
177/177 [==============================] - 7s 40ms/step - loss: 3.1234 - accuracy: 0.9033 - val_loss: 1.9421 - val_accuracy: 0.8078
Epoch 142/250
177/177 [==============================] - 7s 40ms/step - loss: 3.1295 - accuracy: 0.8964 - val_loss: 2.1388 - val_accuracy: 0.8184
Epoch 143/250
177/177 [==============================] - 7s 40ms/step - loss: 3.0746 - accuracy: 0.8922 - val_loss: 2.2722 - val_accuracy: 0.8284
Epoch 144/250
177/177 [==============================] - 7s 40ms/step - loss: 3.0169 - accuracy: 0.9003 - val_loss: 5.7118 - val_accuracy: 0.7652
Epoch 145/250
177/177 [==============================] - 7s 40ms/step - loss: 2.9854 - accuracy: 0.8982 - val_loss: 2.8649 - val_accuracy: 0.7645
```

Hình 12. Mô hình được huấn luyện

Mô hình được huấn luyện sau 250 lần học với độ chính xác cao nhất 82.84% ở lần thứ 143. Ta có thể thấy mô hình khá tốt với độ chính xác khoảng 70 – 80% với phương pháp CNN.

3.4. Kiểm nghiệm và đánh giá mô hình.

Trong quá trình xây dựng một mô hình machine learning, một phần không thể thiếu để biết được chất lượng của mô hình như thế nào đó chính là đánh giá mô hình. Để đánh giá mô hình một cách chính xác hơn ta cần phải kiểm nghiệm thực tế và kiểm tra bằng phương pháp realtime hoặc video.



Hình 13. Biểu đồ độ chính xác của mô hình

Nhìn vào biểu đề ta có thể thấy mô hình tốt khi không bị overfitting hay underfitting, cũng như bộ dữ liệu đủ lớn và được xử lý sạch.

https://github.com/NgThaiBinh/Nose_Keypoint_Detecion.git

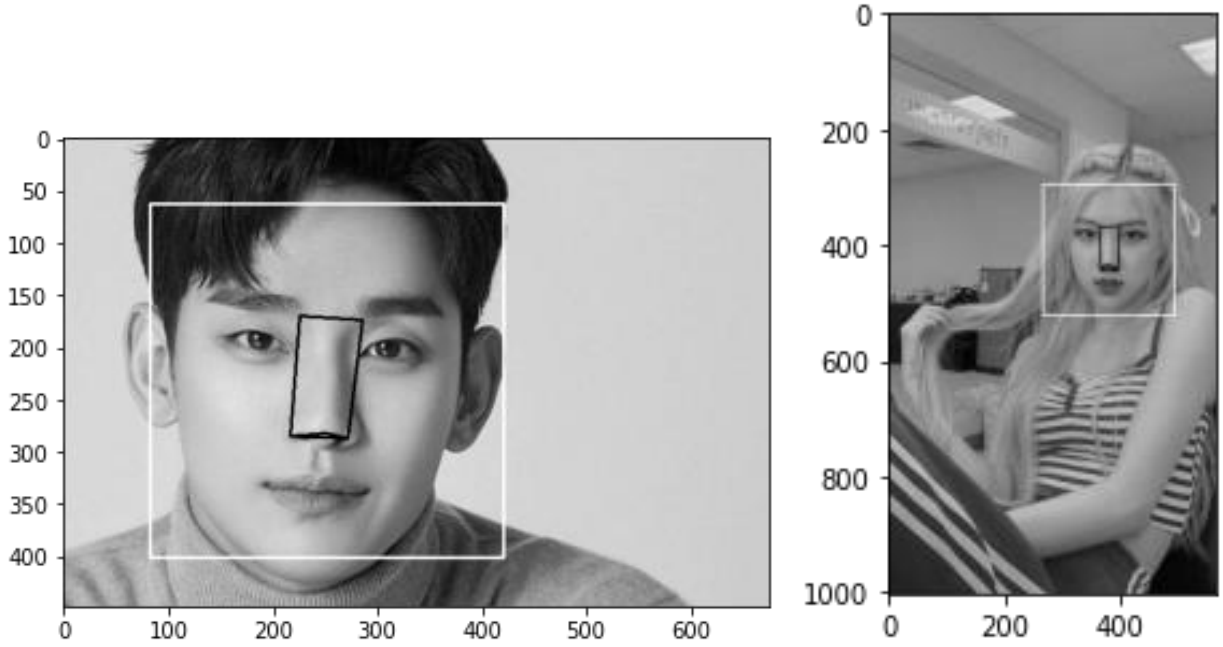
3.4.1. Kiểm nghiệm và đánh giá bằng ảnh.

Mô hình được kiểm nghiệm bằng một bức ảnh mặt người chưa từng có trong bộ dữ liệu trước đó. Ta thấy rằng mô hình hoạt động tốt, xác định được khuôn mặt trong ảnh và đã xác định được điểm mốc của mũi trên khuôn mặt.

```
[50] #anh
     import cv2

     face_cascade = cv2.CascadeClassifier('/content/drive/MyDrive/Ai_project/AI/haarcascade_frontalface_default.xml')

     img = cv2.imread('/content/drive/MyDrive/Ai_project/AI/anh_test/test_mat.jpg')

     img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
     faces = face_cascade.detectMultiScale(img, 1.1, 4)
     maxx = maxy = maxw = maxh = 0
     for (x, y, w, h) in faces:
         if (w*h > maxw*maxh):
           maxx = x
           maxy = y
           maxw = w
           maxh = h
     cv2.rectangle(img, (maxx, maxy), (maxx+maxw, maxy+maxh), (255, 0, 0), 2)
     maxh = maxw = max(maxh, maxw)
     crop_img = img[maxy:maxy+maxh, maxx:maxx+maxw]

     scale = maxw/96.0
     X = cv2.resize(crop_img, (96, 96), interpolation = cv2.INTER_AREA)
     X = np.array([X], dtype = 'float')
     X = X.reshape(-1,96,96,1)

     pred = model.predict(X)

     nose_point = [
                   [pred[0][16],pred[0][17]],
                   [pred[0][12],pred[0][13]],
                   [(pred[0][22]+pred[0][26])/2, (pred[0][13]+96)/2],
                   [pred[0][20],pred[0][21]],
                   [(pred[0][24]+pred[0][28])/2, (pred[0][17]+96)/2],
                   [(pred[0][22]+pred[0][26])/2, (pred[0][13]+96)/2],
                   [(pred[0][24]+pred[0][28])/2, (pred[0][17]+96)/2]
                  ]
     for i in range(0, len(nose_point)):
       nose_point[i][0]=int(nose_point[i][0]*scale)+maxx
       nose_point[i][1]=int(nose_point[i][1]*scale)+maxy
     nose_point = np.array(nose_point, np.int32)
     nose_point = nose_point.reshape((-1, 1, 2))

     img = cv2.polylines(img, [nose_point],
                         True, (0), 2)
     plt.imshow(img, cmap = 'gray')
     plt.show()
```

Hình 14. Kiểm nghiệm bằng ảnh

https://github.com/NgThaiBinh/Nose_Keypoint_Detecion.git



Hình 15. Kết quả kiểm nghiệm bằng ảnh

https://github.com/NgThaiBinh/Nose_Keypoint_Detecion.git

3.4.2. Kiểm nghiệm và đánh giá bằng realtime (video).

Mô hình được kiểm nghiệm bằng một video quay mặt một người lạ với các chuyển động đầu và nét mặt. Mô hình vẫn có thể nhận dạng được tuy nhiên vẫn hơi lệch các điểm mốc và khá lâu hơn so với thực nghiệm bằng ảnh. Vì thực hiện bằng phương pháp CNN và mô hình xác định điểm mốc của từng khung hình trên từng giây (30 hình/ giây) nên như vậy ta có thể cho rằng mô hình đã hoạt động khá tốt.

```
[49] #video
     import cv2
     import numpy as np
     from google.colab.patches import cv2_imshow

     face_cascade = cv2.CascadeClassifier('/content/drive/MyDrive/Ai_project/AI/haarcascade_frontalface_default.xml')

     cap = cv2.VideoCapture('/content/drive/MyDrive/Ai_project/AI/anh_test/test_video.mp4')

     if (cap.isOpened()== False):
       print("Error opening video stream or file")

     count = 0
     # grab the width, height, and fps of the frames in the video stream.
     frame_width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
     frame_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
     fps = int(cap.get(cv2.CAP_PROP_FPS))

     # initialize the FourCC and a video writer object
     fourcc = cv2.VideoWriter_fourcc(*'XVID')
     output = cv2.VideoWriter('output.mp4', fourcc, fps, (frame_width, frame_height))
```

```
     while(cap.isOpened()):
       ret, frame = cap.read()
       if ret == True:
         count += 1
         print(count)
         img = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
         faces = face_cascade.detectMultiScale(img, 1.1, 4)
         maxx = maxy = maxw = maxh = 0
         for (x, y, w, h) in faces:
             if (w*h > maxw*maxh):
```

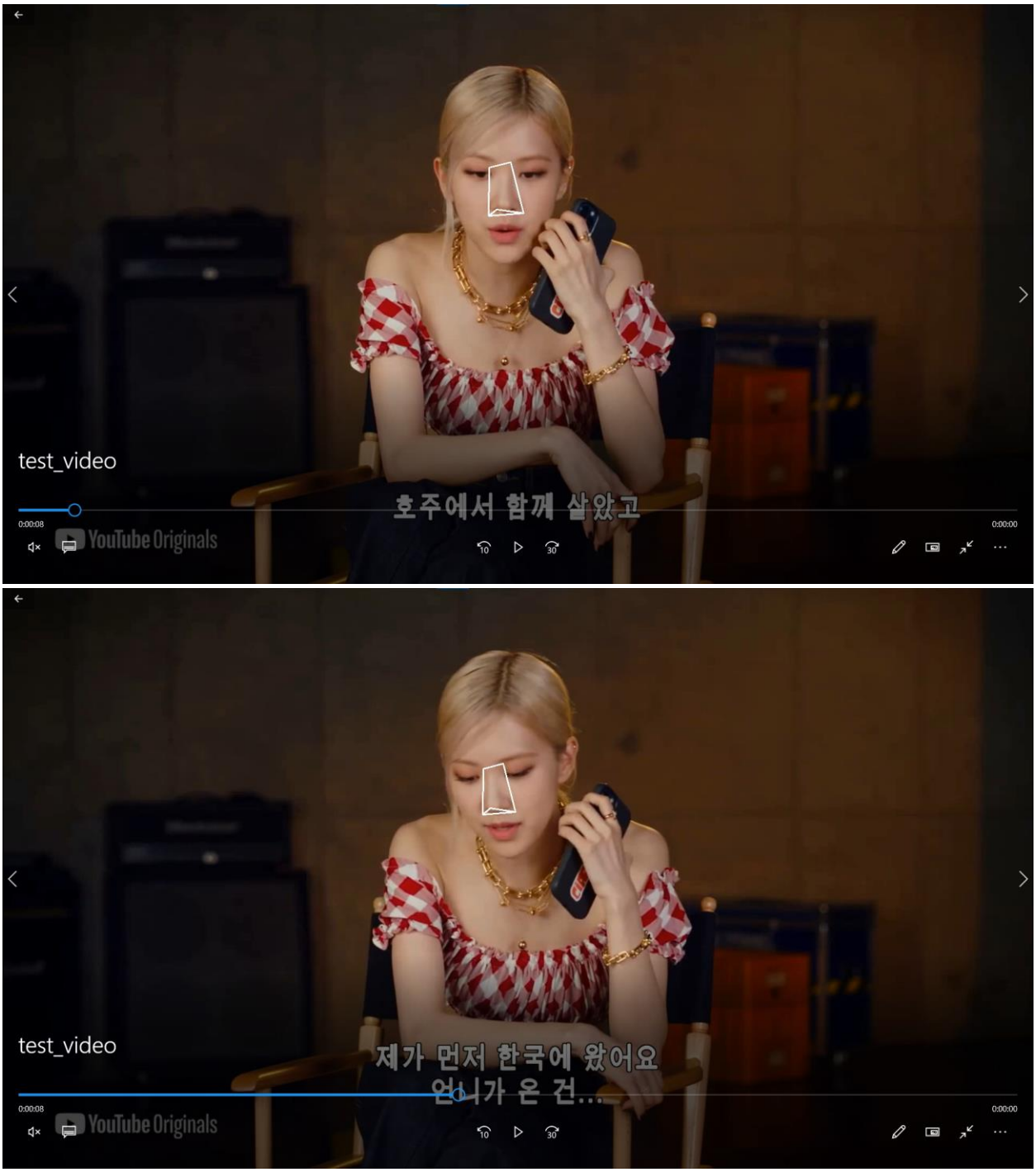

https://github.com/NgThaiBinh/Nose_Keypoint_Detecion.git

```
            maxx = x
            maxy = y
            maxw = w
            maxh = h
        cv2.rectangle(img, (maxx, maxy), (maxx+maxw, maxy+maxh), (255, 0, 0), 2)
        maxh = maxw = max(maxh, maxw)
        crop_img = img[maxy:maxy+maxh, maxx:maxx+maxw]

        scale = maxw/96.0
        #X = cv2.resize(crop_img, (96, 96), interpolation = cv2.INTER_AREA)
        X = np.array([X], dtype = 'float')
        X = X.reshape(-1,96,96,1)

        pred = model.predict(X)

        nose_point = [
                      [pred[0][16],pred[0][17]],
                      [pred[0][12],pred[0][13]],
                      [(pred[0][22]+pred[0][26])/2, (pred[0][13]+96)/2],
                      [pred[0][20],pred[0][21]],
                      [(pred[0][24]+pred[0][28])/2, (pred[0][17]+96)/2],
                      [(pred[0][22]+pred[0][26])/2, (pred[0][13]+96)/2],
                      [(pred[0][24]+pred[0][28])/2, (pred[0][17]+96)/2]
                    ]
        for i in range(0, len(nose_point)):
          nose_point[i][0]=int(nose_point[i][0]*scale)+maxx
          nose_point[i][1]=int(nose_point[i][1]*scale)+maxy
        nose_point = np.array(nose_point, np.int32)
        nose_point = nose_point.reshape((-1, 1, 2))
```

```
        frame = cv2.polylines(frame, [nose_point],
                              True, (255,255,255), 2)
        output.write(frame)
      else:
        break

    cap.release()
    output.release()
```

Hình 16. Kiểm nghiệm bằng video

https://github.com/NgThaiBinh/Nose_Keypoint_Detecion.git



Hình 17. Kết quả kiểm tra bằng video

https://github.com/NgThaiBinh/Nose_Keypoint_Detecion.git

CHƯƠNG 4: NHẬN XÉT VÀ HƯỚNG PHÁT TRIỂN

4.1. Những kết quả đạt được.

Nhận diện được điểm mốc của mũi trên khuôn mặt.

Áp dụng được các phương pháp học và phân lớp vào việc xây dựng và huấn luyện mô hình.

4.2. Hạn chế.

Vẫn còn tồn động những vấn đề về việc tìm điểm mốc chính xác nhất.

Thời gian nhận diện realtime (video) còn lâu.

4.3. Hướng phát triển.

Áp dụng nhiều thuật toán về nhận diện các điểm mốc khuôn mặt và xử lý dữ liệu để có thể rút ngắn thời gian nhận dạng và đạt hiệu suất cao nhất.

Nâng cao thành bài toán dựng 3D từ các điểm mốc đã nhận diện.

Phát triển thực tế thành phần mềm chỉnh sửa ảnh chân dung hoặc phần mềm nhận dạng người qua điểm mốc trên khuôn mặt.

KẾT LUẬN

Qua quá trình nghiên cứu và tìm hiểu dưới sự hướng dẫn tận tình của giảng viên, em đã hoàn thành báo cáo Trí tuệ nhân tạo đề tài "Nhận diện các điểm mốc trên mũi". Qua đó, em có cơ hội làm quen với các công nghệ, kỹ thuật số, hiểu rõ hơn về các thuật toán, các phương pháp xây dựng và huấn luyện một mô hình nhận diện cũng như phát triển kỹ năng tư duy sáng tạo, khả năng học hỏi và giải quyết vấn đề. Với sự nỗ lực của bản thân và sự giúp đỡ của thầy cô, bạn bè em đã hoàn thành báo cáo khá tốt. Tuy nhiên không tránh khỏi những thiết sót rất mong được sự góp của thầy cô và bạn đọc.

Em hy vọng các khóa sau sẽ có cơ hội học tập và phát triển hơn nửa về các kỹ thuật của Trí tuệ nhân tạo cũng như áp dụng được các kiến thức đã học và mô hình vào thực tế.

Trong suốt quá trình học tập và hoàn thành báo cáo này, đã cho em rất nhiều kiến thức và kinh nghiệm về chuyên nghành. Để hoàn thành được báo cáo em đã nhận được sự hướng dẫn, giúp đỡ quý báu của các thầy cô và các bạn.

Em xin chân thành cảm ơn!

TÀI LIỆU THAM KHẢO

[1]. High-Resolution Representations for Labeling Pixels and Regions | Papers With Code

[2]. Facial Keypoint Detection ResNet50 + aug | Kaggle

[3]. aladdinpersson/Machine-Learning-Collection: A resource for learning about ML, DL, PyTorch and TensorFlow. Feedback always appreciated :) (github.com)

[4]. OpenCV: face_landmark_trainer