TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HCM
Khoa Đào Tạo Chất Lượng Cao
Môn học: TRÍ TUỆ NHÂN TẠO
(ARIN337629_21_2_02CLC)

Sinh viên thực hiện:
Nguyễn Thái Bình – 19146050

Giảng viên hướng dẫn:
PSG. TS Nguyễn Trường Thịnh

BÁO CÁO TRÍ TUỆ NHÂN TẠO
ĐỀ TÀI: NHẬN DIỆN CÁC ĐIỂM MỐC TRÊN MŨI

Tổng quan: Với sự phát triển của công nghệ, hễ nhắc tới các thiết bị điện tử hay bất cứ thiết bị nào người ta đều nhắc đến trí tuệ nhân tạo được tích hợp trên thiết bị đó. Vậy trí tuệ nhân tạo là gì và được ứng dụng như thế nào trong việc nhận diện nói chung và nhận diện các điểm mốc của mũi trên khuôn mặt nói riêng? Trong bài báo này, sẽ giải đáp câu hỏi trên cũng như nói về cách xây dựng mô hình nhận diện các điểm mốc trên mũi với Mạng nơ-ron tích chập (Convolutional Neural Network - CNN). Code và mô hình đã được đăng công khai tại:

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

1. Giới thiệu.

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.

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.

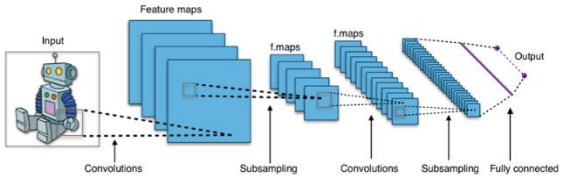
Kiểm nghiệm bằng phương pháp realtime hoặc video.

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.

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.



2. Quy trình thực hiện.

Đầu tiên: Thu thập và xử lý dữ liệu. 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.

Bước hai: Xây dựng mô hình. 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_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)

 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
```

Bước ba: Huấn luyện mô hình. 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.

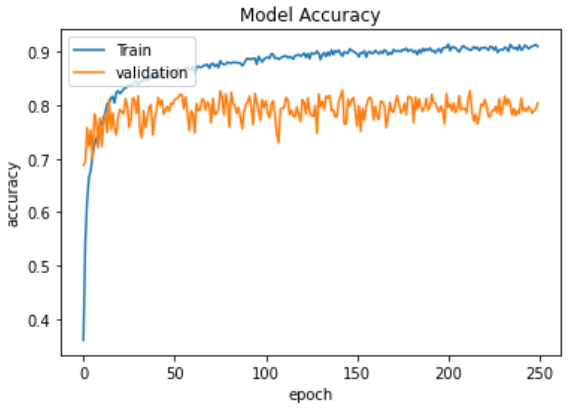
```
[13] model.compile(optimizer='adam',loss='mean_squared_error',metrics=['accuracy'])

[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: 2.0649 - val_accuracy: 0.7645
```

Bước cuối cùng: 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.



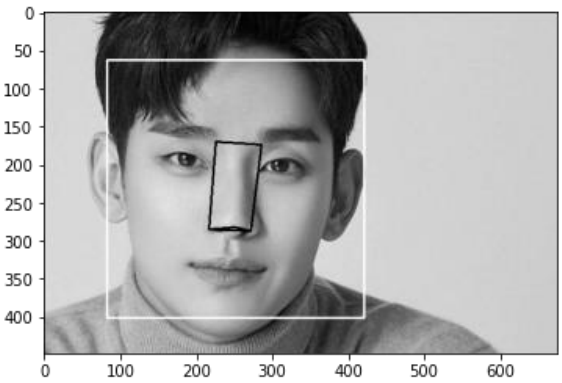
Nhìn vào biểu đề ta có thể thấy mô hình tốt khi không bị overfitting hay underfitting, cũng như bộ dữ liệu đủ lớn và được xử lý sạch trước khi sử dụng

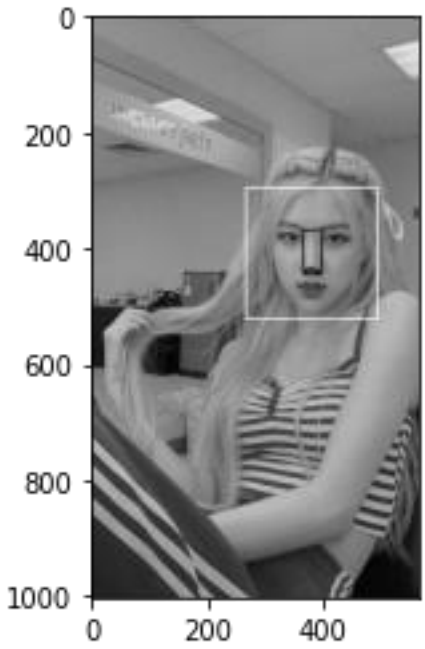
Kiểm nghiệm và đánh giá bằng ảnh.

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.

Kết quả cho ra:

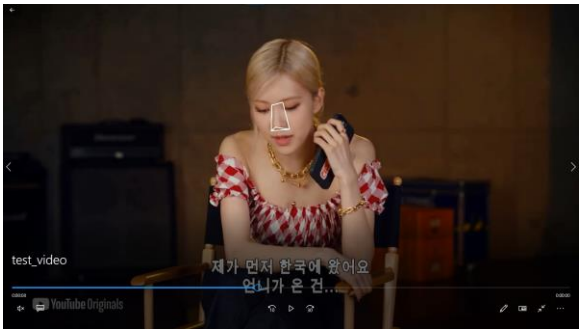




Kiểm nghiệm và đánh giá bằng realtime (video).

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.

Kết quả cho ra:



3. Nhận xét kết quả đề tài và hướng phát triển.

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.

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.

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.

4. 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-ollection: A resource for learning about ML, DL, PyTorch and TensorFlow. Feedback always appreciated :) (github.com)

[4]. OpenCV: face landmark trainer