The Ear as a Biometric

Lê Duy Mẫn, Phan Thanh Tuyển, Nguyễn Tấn Chữ, Nguyễn Thanh Tân

Nội dung

Nội dung

Giới thiệu

- Tiềm năng của nhận dạng bằng tai đã được công nhận và khuyến khích trong khoảng thời gian dài trước đó vào năm 1890 bởi nhà tội phạm học người Pháp Alphonse Bertillon.

Alphonse Bertillon's Quotes

The ear, thanks to these multiple small valleys and hills which furrow across it, is the most significant factor from the point of view of identification. Immutable in its form since birth, resistant to the influences of environment and education, this organ remains, during the entire life, like the intangible legacy of heredity and of the intra-uterine life.

Giới thiệu

- Nhận dạng tai không được chú ý nhiều như những phương pháp nhận dạng mặt, mắt, vân tay. Tuy nhiên, tai đã đóng vai trò quan trọng trong khoa học pháp y trong suốt nhiều năm, đặc biệt tại Hoa Kỳ có 1 hệ thống phân loại tai dựa trên cách đo thủ công được phát triển bởi Alfred Victor Iannarelli, hệ thống này đã được sử dụng trong hơn 40 năm.

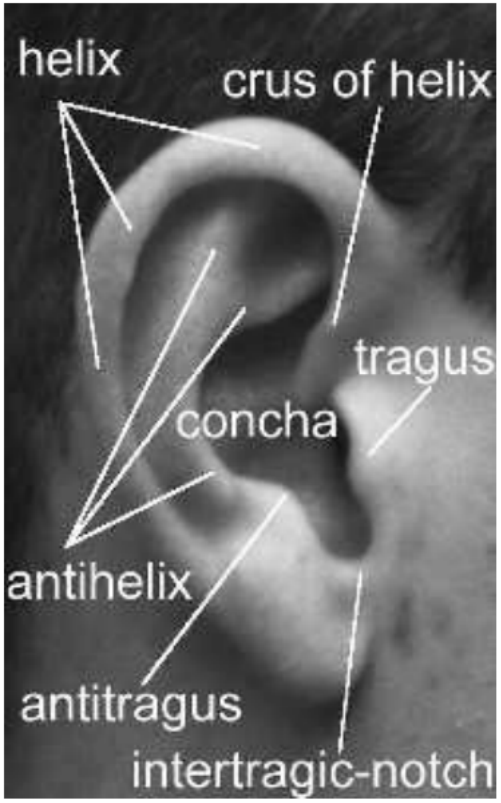
Sơ lược về một vài kỹ thuật và nghiên cứu

- M. Burge, W. Burger và cộng sự là những người đầu tiên mô tả tiềm năng nhận dạng của tai bằng cách sử dụng 1 biểu đồ Voronoi của đường cong được trích xuất từ bản đồ biên Canny.
- B. Moreno, A. Sanchez và cộng sự: dùng neural network và tỉ lệ nhận dạng đúng lên đến 93% trong bộ dữ liệu chứa 168 ảnh bằng kỹ thuật neural network 2 giai đoạn.
- Chang và cộng sự đã sử dụng PCA để nhận dạng tai và đạt được 90.9%.
- H. Zhang, Z. Mu, W. Qu, L. Liu, and C. Zhang đã phát triển một hệ thống kết hợp phân tích thành phần độc lập (Independent Components Analysis - ICA) với một mạng hàm cơ sở (a Radial Basis Function RBF) để nhận dạng và cho thấy ICA đạt hiệu suất tốt hơn cả PCA.

Nội dung

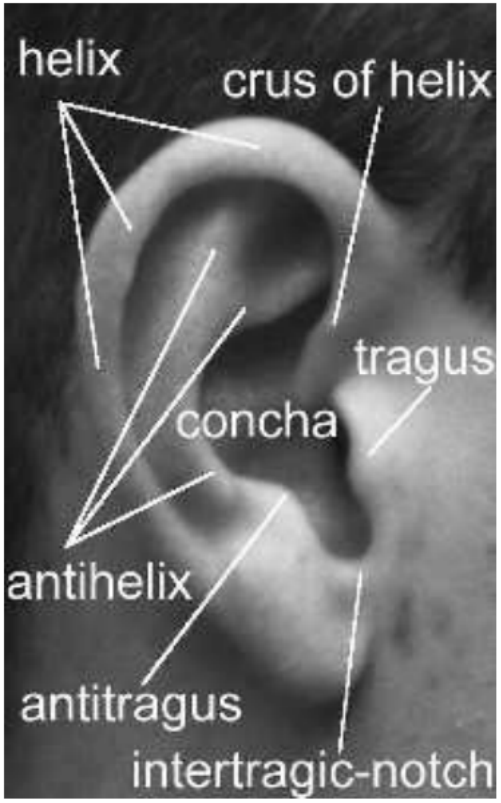
Cấu trúc của tai

Ngoài vành ngoài của tai (helix) và dái tai (lobe), tai còn có những đặc điểm nổi bật khác như vành trong (antihelix), khu vực trung tâm concha.



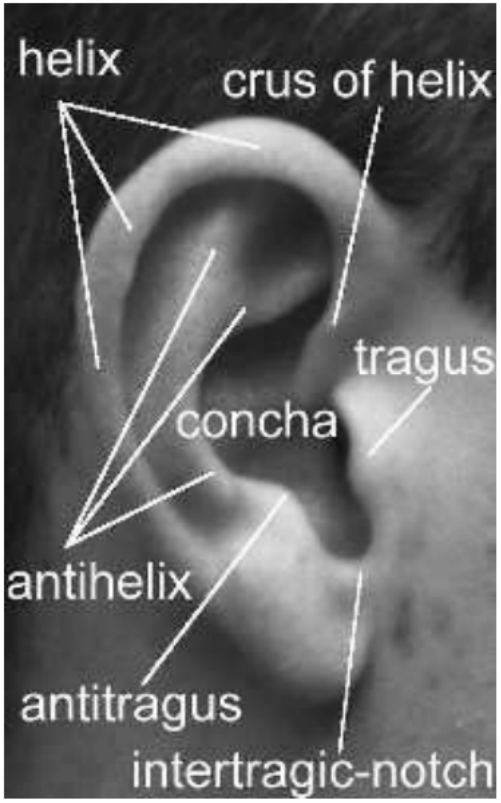
Cấu trúc của tai

- Cấu trúc của tai không phải là ngẫu nhiên mà là có cấu trúc xác định như khuôn mặt.
- Hình dáng của tai thường bị chi phối bởi vành ngoài của tai (helix) và hình dạng của thùy tai (lobe).
- Vành trong của tai (inner helix hay antihelix) chạy song song với vành ngoài của tai (helix) nhưng chia thành 2 nhánh ở đầu trên.



Cấu trúc của tai

Phía dưới của concha hợp nhất vào rãnh giữa tai (intertragic-notch) và có độ cong rất sâu ở dưới cùng có thể tạo thành một điểm tham chiếu hữu ích cho mục đích sinh trắc học. Crus of helix là nơi helix giao với nhánh dưới của antihelix. Đây là một trong những điểm được Iannarelli sử dụng làm điểm tham chiếu cho hệ thống đo lường của ông, điểm tham chiếu khác là antitragus hoặc cái bướu nhỏ ở bên trái của rãnh giữa tai (intertragic-notch).



Cấu trúc của tai

- Một số tai có thùy tai rõ ràng, số khác lại gần như không có thùy được gọi là "thùy gắn kết"$\rightarrow$ đo chiều dài tai trở nên khó khăn.
- Helix và antihelix thường chạy song song, nên có một mức độ tương quan khá cao giữa chúng, làm giảm giá trị sinh trắc học của tai.
- Khu vực concha chỉ là khoảng trống còn lại sau khi các phần khác đã được xác định, do đó đóng góp ít thông tin độc lập.

Cấu trúc của tai

- Nhiều loài động vật như ngựa, chó và mèo có thể cử động tai. Ngược lại, tai con người đa số không thể cử động và được giữ nguyên vị trí bởi các mô sụn được gắn chặt vào xương bên của đầu, giúp cho việc thu thập và nhận dạng dễ dàng hơn.
- Khuôn mặt có độ phức tạp hình ảnh tương tự như tai; những thay đổi đơn giản trong các thông số xác định kích thước và hình dạng của mắt, mũi, miệng và xương gò má có thể dẫn đến nhiều diện mạo khuôn mặt khác nhau. Với tai cũng tương tự như vậy khi thay đổi helix, antihelix, lobe,...
- Tai không có đối xứng nên việc trích xuất các đặc trưng từ tai càng đa dạng và riêng biệt hơn.

Nội dung

Động lực nghiên cứu

Iannarelli đã sử dụng 10.000 hình ảnh tai để chứng minh tính độc nhất của tai và kết luận rằng tai có thể phân biệt dựa trên một số đặc điểm hạn chế.

Phân tích chỉ số phân biệt (đo lường sự phân tách giữa điểm số chính xác và điểm số giả mạo cho một hệ thống sinh trắc học) cũng cho thấy tính độc nhất của tai và cho thấy chỉ số phân biệt của tai cao hơn đáng kể so với khuôn mặt, nhưng không lớn bằng chỉ số phân biệt của mống mắt.

Động lực nghiên cứu

Các đặc điểm làm cho sinh trắc học tai trở thành một lựa chọn phổ biến cho nhận dạng con người:

- Hình dạng của tai được cho là ổn định trong phần lớn cuộc đời.
- Không thay đổi hình dạng khi mặt thay đổi biểu cảm. Phân phối màu sắc của tai gần như đồng đều.
- Cải thiện độ chính xác cho hệ thống nhận diện tạo ra bởi các yếu tố ngoại cảnh như ánh sáng, góc chụp nghiêng.
- Tai không bị ảnh hưởng bởi mỹ phẩm và kính áp tròng.
- Quan sát và thu thập dữ liệu dễ dàng.
- Tai có kích thước nhỏ hơn khuôn mặt, giúp hệ thống nhanh và hiệu quả hơn với những ảnh có độ phân giải thấp.

Động lực nghiên cứu

Tính ứng dụng:

- Có thể kết hợp với nhận diện khuôn mặt để tăng độ chính xác trong nhận diện người.
- Sử dụng trong các ứng dụng an ninh bảo mật, truy tìm tội phạm.
- Sử dụng trong lĩnh vực pháp y, y tế để xác định danh tính bệnh nhân khi khuôn mặt bị biến dạng.
- Sinh trắc học tai đã được sử dụng để nhận ra các cặp song sinh giống hệt nhau vì các đặc điểm phân biệt của nó

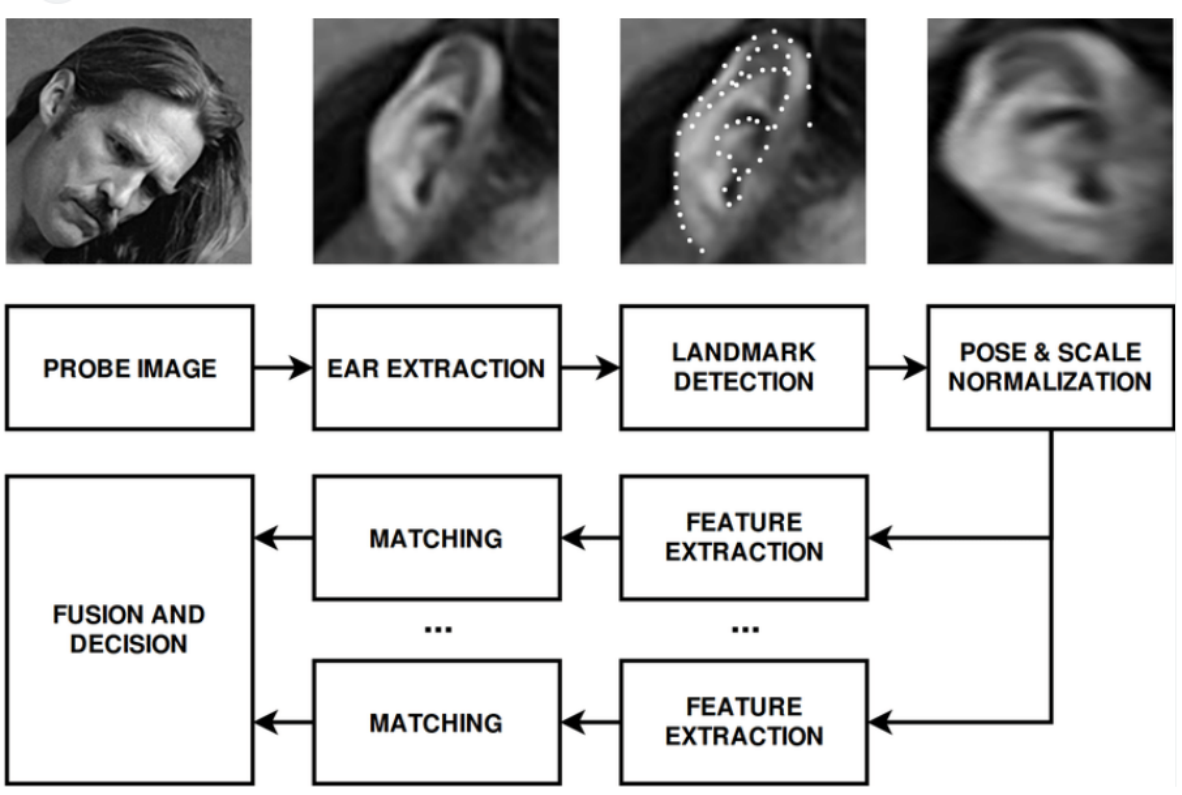
Nội dung

Bài toán

- Loại bài toán: Phát hiện và Nhận diện
- Input: Ảnh 2D chứa 1 đối tượng tai người cần nhận diện.
- Output: ID người phù hợp nhất với đối tượng trong cơ sở dữ liệu.

Bài toán

Một hình ảnh được cung cấp làm đầu vào cho hệ thống. Hệ thống sẽ chỉnh sửa hình ảnh, thực hiện tiền xử lý, trích xuất tính năng, phân loại, huấn luyện và sau đó nhận dạng người dùng.



Nội dung

Ear Recognition Based on Deep Convolutional Network

- Tác giả: Liang Tian, Zhichun Mu
- Tạp chí/ hội nghị: 2016 9th International Congress on Image and Signal Processing, Bio-Medical Engineering and Informatics

Ear Recognition Based on Deep Convolutional Network

Tác giả sử dụng mạng neural tích chập (CNN) để phát triển hệ thống nhận dạng tai. Quá trình huấn luyện đi qua hai pha:

- Lan truyền tiến: Tiếp nhận trọng số của mô hình để tính toán ở các neuron.
- Lan truyền ngược: Tính độ lỗi và cập nhật lại trọng số.

Hệ thống này có thể trích xuất đặc trưng tự động ở mức thấp lẫn mức cao. Những đặc trưng này học tốt hơn đặc trưng thủ công truyền thống.
Điều cần thiết là cách thiết kế một mạng neural và phương pháp huấn luyện phù hợp.

A Review on Biometrics and Ear Recognition Techniques

- Tác giả: Sukhdeep Singh, Dr. Sunil Kumar Singla
- Tạp chí/ hội nghị: IJARCSSE Volume 3, Issue 6, June 2013

A Review on Biometrics and Ear Recognition Techniques

Nền của một tấm ảnh tai người rất dễ đoán vì nó luôn nằm ở một phía của đầu trong khi nhận diện mặt thì cần phải kiểm xoát nền của tấm ảnh để mang lại hiệu quả cao mà thường thì điều này không phải lúc nào cũng đạt được. Còn mống mắt, võng mạc hay vân tay thì cần tiếp xúc gần với công cụ để chụp lại. Trong khi tai thì dễ để chụp ảnh lại hơn và yêu cầu về nền ảnh không quá khắt khe.

A Review on Biometrics and Ear Recognition Techniques

Toàn bộ công nghệ nhận dạng và xác thực vận hành qua 4 pha:

- Pha chụp: Một mẫu sẽ được chụp lại bằng máy ảnh trong quá trình ghi danh và cả trong quá trình nhận dạng hoặc xác thực. Có thể sử dụng bất kỳ máy ảnh kỹ thuật số nào và dễ sử dụng.
- Pha trích xuất: Một dữ liệu độc nhất được trích xuất từ mẫu bằng cách sử dụng nhiều kỹ thuật khác nhau. Một mẫu hình được tạo ra bằng Matlab và Lab VIEW.
- Pha so sánh: Mẫu hình được so sánh với mẫu gốc.
- Pha khớp/Không khớp: Pha này trả về kết quả mẫu mới khớp hay không khớp.

A Review Paper on Ear Recognition Techniques: Models, Algorithms and Methods

- Tác giả: Khamiss Masaoud, S. Algabary, Khairuddin Omar, Md. Jan Nordin, Siti Norul Huda Sheikh Abdullah
- Tạp chí/ hội nghị: Australian Journal of Basic and Applied Science

A Review Paper on Ear Recognition Techniques: Models, Algorithms and Methods

Bài báo này tổng hợp các công trình nghiên cứu quan trọng.

- Những năm 1980 ở Hoa Kỳ, Iannarelli đã phát triển một hệ thống mô tả đặc trưng tai có ích, phục vụ cho việc nhận dạng cá nhân. Một hệ thống nhận dạng tai giống với hệ thống nhận dạng khuôn mặt và bao gồm 5 thành tố: Tạo cơ sở dữ liệu ảnh, tiền xử lý, trích xuất đặc trưng, huấn luyện mô hình và khớp mẫu hình.
- Alvarez et al. (2005) sử dụng thuật toán contour hiệu chỉnh và mô hình Ovoid để phân biệt tai.

A Review Paper on Ear Recognition Techniques: Models, Algorithms and Methods

- Saleh et al. (2006) thử nghiệm một tập dữ liệu về hình ảnh tai sử dụng một vài bộ phân lớp dựa trên hình ảnh và một vài phương pháp trích chọn đặc trưng. Kết quả cho thấy hiệu suất đạt từ 76.5% đến 94.1% dựa trên từng thực nghiệm của họ.
- Islam et al. (2008) đề xuất phương pháp nhận dạng tai dựa trên thuật toán AdaBoost, mô hình này nhạy cảm với nhiễu và ngoại lai trong dữ liệu.
- Hệ thống do Islam et al (2007) phát triển hệ thống dựa trên đặc trưng Haar-like và sử dụng tập dữ liệu bao gồm nhiều sắc tộc, giới tính, hình dạng, độ nghiêng và độ sáng khác nhau

Training CNN with limited training data for ear recognition in the wild

- Tác giả: Ziga Emersic, Dejan Stepec, Peter Peer
- Tạp chí/ hội nghị: 2017 IEEE 12th International Conference on Automatic Face & Gesture Recognition

Training CNN with limited training data for ear recognition in the wild

Trong bài báo này, vấn đề huấn luyện CNN với dữ liệu hạn chế được nêu ra và họ cố gắng phát triển một mô hình CNN hiệu quả hơn cho bài toán nhận dạng tai. Những cách tiếp cận hiện tại với dữ liệu nhỏ bao gồm:

- Tiếp cận học bằng độ đo, khi tập huấn luyện là các cặp ảnh hay vì các ảnh đơn.
- Tăng cường dữ liệu dựa trên hình học và độ nhiễu màu của tập huấn luyện hoặc là phương pháp tổng hợp mẫu dữ liệu.
- Sử dụng mạng CNN như là một bộ trích xuất đặc trưng.

Smart Augmentation Learning an Optimal Data Augmentation Strategy

- Tác giả: Joseph Lemley, Shabab Bazrafkan, Peter Corcoran

Smart Augmentation Learning an Optimal Data Augmentation Strategy

Nhiều framework học sâu có khả năng tạo ra dữ liệu tăng cường.

- Keras có một phương thức để ngẫu nhiên xoay, lật và thu phóng
- Nhưng không phải phương pháp nào cũng cải thiện hiệu năng và không nên sử dụng một cách mù quáng.
- Sử dụng mạng CNN như là một bộ trích xuất đặc trưng.

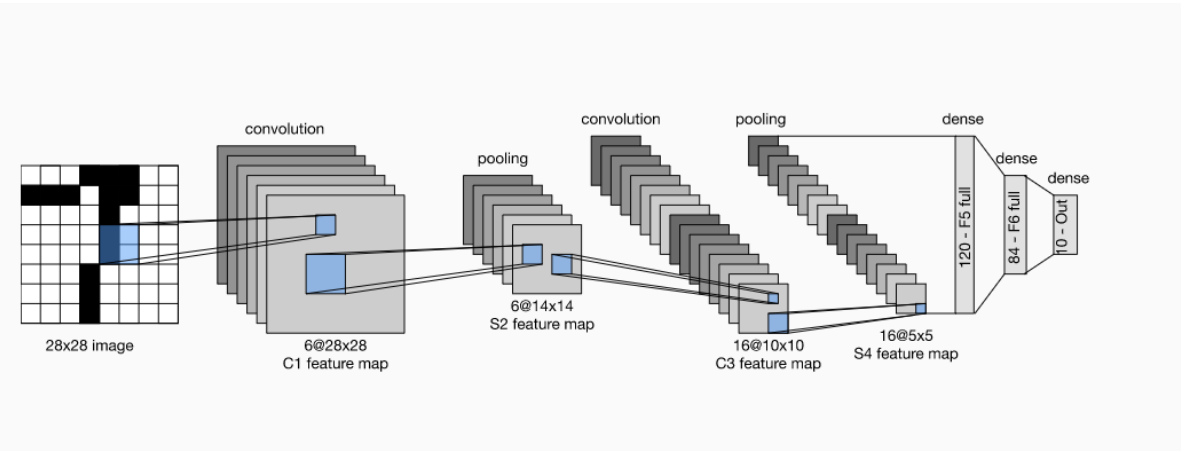
Năm 2014, Srivastava et al. giới thiệu kỹ thuật dropout, với mục tiêu là giảm overfitting, đặc biệt là trong trường hợp không đủ dữ liệu. Dropout hoạt động bằng cách tạm thời loại bỏ một neuron nhân tạo trong mạng và bỏ luôn bất kì liên kết nào đến neuron đó.

Nội dung

Nội dung

Mạng neural tích chập (CNN)

Kiến trúc CNN thông dụng bao gồm Convolution + Hàm kích hoạt ReLU -> Pooling -> Dense -> Softmax loss



Bên trên là mạng LeNet, được giới thiệu lần đầu bởi Yann LeCun cho tác vụ nhận dạng chữ viết tay vào năm 1998.

CNN: Lớp Convolution

Trong bài toán thị giác máy tính:

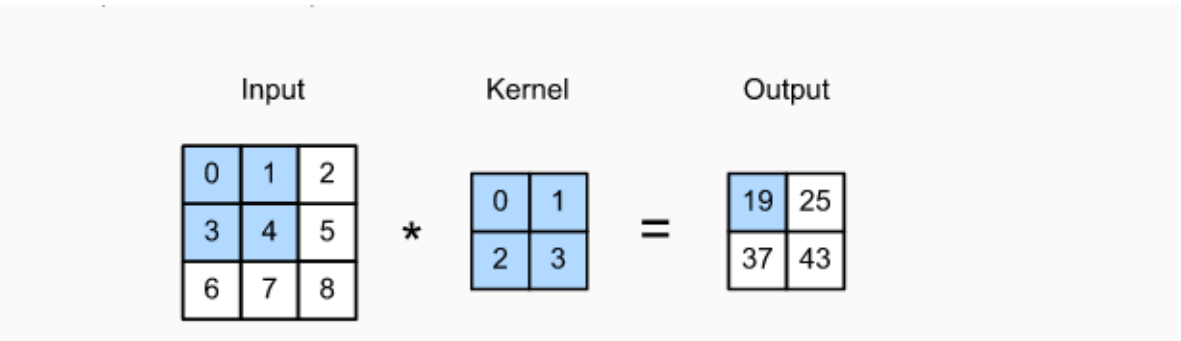
- Lớp dense học các khuôn mẫu toàn cục
- Lớp convolution học các khuôn mẫu cục bộ, thông qua phép tích chập với các filter (còn gọi là kernel) giống như xử lý ảnh (ví dụ như Gabor filter).

Loại convolution phổ biến trong bài toán xử lý ảnh thường là 2D Convolution, nghĩa là vùng tích chập sẽ "trượt"trên hai chiều ngang và dọc trên ma trận.

CNN: Lớp Convolution

Filter trong lớp Convolution được định nghĩa qua ba tham số:

- F: Kích thước filter, do filter luôn là ma trận vuông nên kích thước sẽ là *FxF*
- S: Kích trước trượt, diễn tả vùng trượt sau mỗi bước sẽ trượt bao nhiêu đơn vị.
- P: Vùng đệm, dùng để bổ sung các con số 0 dọc theo các trục tọa độ cho mục đích riêng.



Về mặt tính toán, đơn giản là thực hiện tích element-wise của vùng trượt với filter, kết quả cho ra một pixel mới. Như vậy, ma trận đầu ra sẽ là ma trận pixel sau khi vùng trượt đi hết ma trận ảnh đầu vào.

CNN: Lớp Pooling

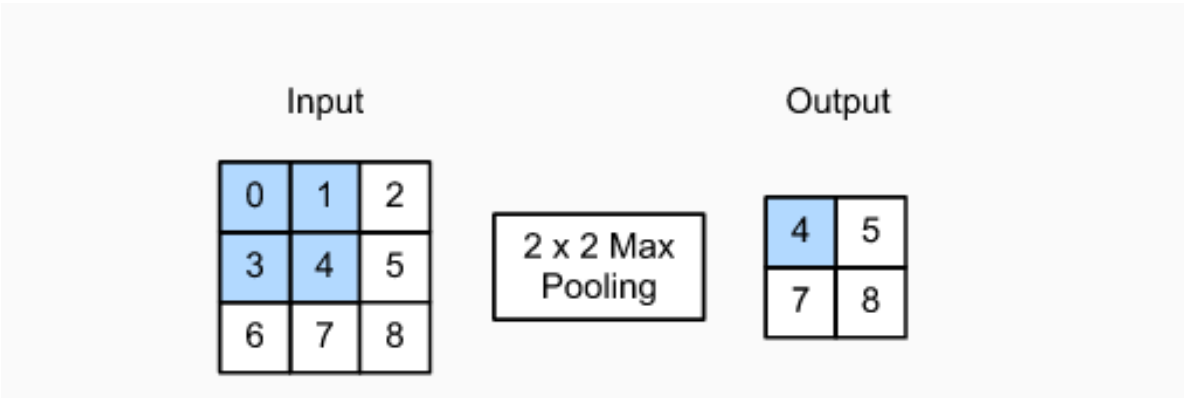
Ý nghĩa của lớp Pooling là để:

- Làm giảm số lượng tham số của mạng.
- Tạo ra sự bất biến với phép tịnh tiến một lượng nhỏ.

Pooling thường có hai dạng là Average Pooling và Max Pooling. Nhưng Max Pooling thường được sử dụng nhiều hơn.

CNN: Max Pooling

- Max Pooling có thể xem là một filter với tác dụng trả về pixel lớn nhất trong một vùng trượt.
- Khác với vùng trượt trong lớp Convolution, vùng trượt trong Max Pooling không chồng lên nhau sau mỗi lần trượt, tức là một pixel chỉ đi qua filter một lần.



CNN: Lớp Dense

- Đưa ra mức độ suy luận cao hơn cho mạng.
- Rút trích đặc trưng toàn cục từ các đặc trưng cục bộ do lớp CNN tạo ra.

Nội dung

Phương pháp

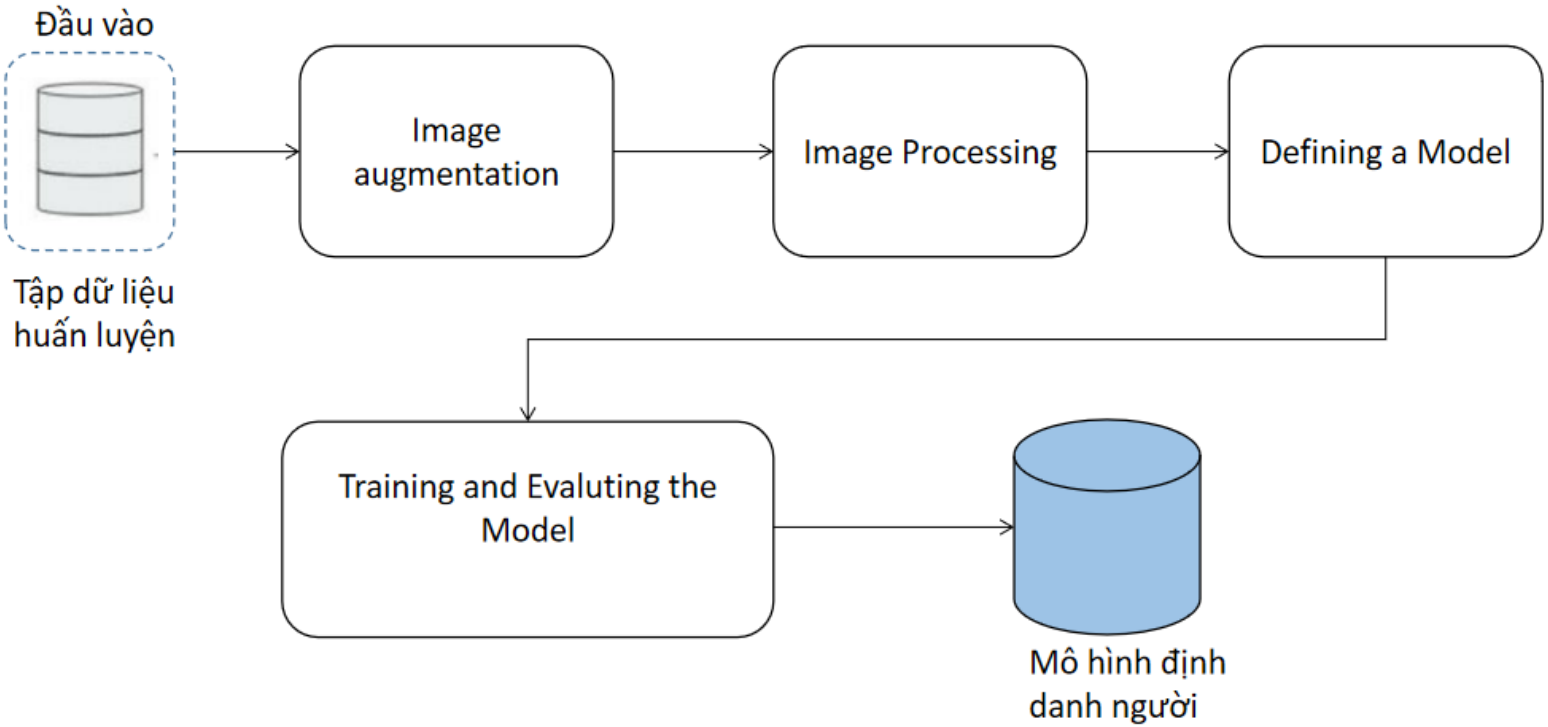
Nhận diện sinh trắc học bằng tai người là tác vụ so khớp 1 đối tượng cần nhận diện với nhiều đối tượng ứng viên trong tập dữ liệu. Việc so khớp này không thực hiện trực tiếp trên các điểm ảnh thô của ảnh đầu vào, thay vào đó thực hiện thông qua vector đặc trưng rút trích từ ảnh đó. Để làm được điều đó cần phải thực hiện 2 giai đoạn: ngoại tuyến và trực tuyến.

Giai đoạn ngoại tuyến

- Giai đoạn ngoại tuyến hay giai đoạn huấn luyện là giai đoạn xây dựng mô hình hay bộ trọng số thích hợp dựa trên đặc trưng của tập input đầu vào để phục vụ cho giai đoạn trực tuyến.
  - Input: Tập dữ liệu ảnh huấn luyện chứa tai người.
  - Output: 1 mô hình có thể rút trích đặc trưng học sâu từ ảnh tai người.

Giai đoạn ngoại tuyến

Lưu đồ



Giai đoạn ngoại tuyến: Tăng cường dữ liệu

Bộ dữ liệu có sẵn để huấn luyện không đủ để đạt độ chính xác cao. Do đó, cần tìm 1 cách để giải quyết vấn đề này. Một trong những cách đơn giản nhất là Image augmentation (tăng cường hình ảnh) bằng cách tạo thêm nhiều ảnh huấn luyện từ ảnh gốc bằng cách điều chỉnh các thông số như góc quay, độ sáng, lật ngang- dọc . . .

Ý tưởng này mô phỏng những thay đổi của ảnh trong thực tế khi bị tác động bởi các yếu tố ngoại cảnh như độ sáng, góc chụp, biến dạng ảnh . . .

Giai đoạn ngoại tuyến: Tiền xử lý dữ liệu

Vì số lượng ảnh huấn luyện rất lớn nên bước tiền xử lý rất quan trọng để rút ngắn thời gian cũng như tăng hiệu suất huấn luyện. Tiền xử lý gồm:

- Đặt kích thước chuẩn cho hình ảnh là 128x128 để giảm thời gian tải ảnh.
- Chuyển ảnh từ 3 kênh màu RGB sang 1 kênh thang độ xám.
- Gán nhãn theo từng lớp để thực hiện học có giám sát.
- Chia tập huấn luyện thành 2 tập huấn luyện và đánh giá với tỉ lệ 8:2.

Giai đoạn ngoại tuyến: Xây dựng mô hình

Mô hình được xây dựng dựa trên kiến trúc CNN. Mô hình gồm 3 lớp conv, 3 lớp activation, 2 lớp max_pooling và 2 lớp dropout.

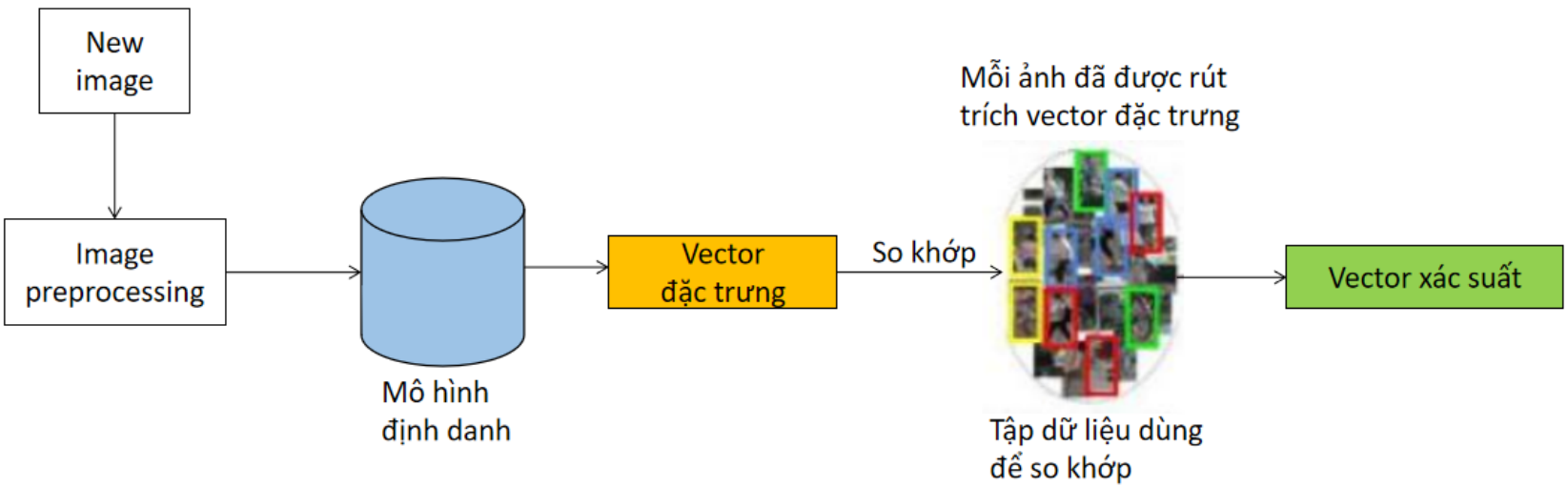
Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 128, 128, 32)	320
activation_1 (Activation)	(None, 128, 128, 32)	0
conv2d_2 (Conv2D)	(None, 126, 126, 32)	9248
activation_2 (Activation)	(None, 126, 126, 32)	0
max_pooling2d_1 (MaxPooling2	(None, 63, 63, 32)	0
dropout_1 (Dropout)	(None, 63, 63, 32)	0
conv2d_3 (Conv2D)	(None, 61, 61, 64)	18496
activation_3 (Activation)	(None, 61, 61, 64)	0
max_pooling2d_2 (MaxPooling2	(None, 30, 30, 64)	0
dropout_2 (Dropout)	(None, 30, 30, 64)	0
flatten_1 (Flatten)	(None, 57600)	0
dense_1 (Dense)	(None, 64)	3686464
activation_4 (Activation)	(None, 64)	0
dropout_3 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 11)	715
activation_5 (Activation)	(None, 11)	0

Giai đoạn ngoại tuyến: Huấn luyện và đánh giá mô hình

Đây là công đoạn dựa vào các đặc trưng học sâu được trích xuất qua các layers, bằng cách lặp lại bước tinh chỉnh bộ trọng số và đánh giá sự mất mát bằng 1 hàm lỗi, quá trình sẽ tiếp tục đến khi quan sát thấy sự mất mát được cải thiện và các giá trị của bộ trọng số có sai số nhỏ nhất.

Giai đoạn trực tuyến

- Input: Ảnh mới cần nhận diện
- Output: Nhãn của ảnh cần nhận diện



Giai đoạn trực tuyến

Giai đoạn trực tuyến gồm:

- Tiền xử lý: Thực hiện các bước tiền xử lý với các quy chuẩn như ở bước ngoại tuyến.
- Rút trích vector đặc trưng: Sử dụng mô hình đã được huấn luyện ở bước ngoại tuyến để rút trích vector đặc trưng học sâu cho ảnh đầu vào.
- So khớp: Vector đặc trưng của ảnh đầu sẽ được so khớp với vector đặc trưng của tập dữ liệu từng đôi một theo một chiến lược nào đó. Kết quả cuối cùng là 1 vector xác suất dự đoán nhãn cho ảnh đầu vào.

Nội dung

Demo

```
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing.image import array_to_img, img_to_array, load_img
import os
from google.colab import drive
```

```
datagen = ImageDataGenerator(
        rotation_range=30,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.2,
        zoom_range=0.2,
        zca_whitening= True,
        horizontal_flip=False,
        rescale=1.2,
        fill_mode='nearest')
```

```
def Data_Augument(link_input,link_output,data_list,number):
    for img_name in data_list:
      img_path = os.path.join(link_input,img_name)
      name = img_name[:-4]
      img = load_img(img_path)  # this is a PIL image
      x = img_to_array(img)  # this is a Numpy array with shape (3, 150, 150)
      x = x.reshape((1,) + x.shape)  # this is a Numpy array with shape (1, 3, 150, 150)
      # the .flow() command below generates batches of randomly transformed images
      # and saves the results to the `preview/` directory
      i = 0
      for batch in datagen.flow(x, batch_size=1,
                                save_to_dir=link_output, save_prefix=name, save_format='bmp'):
        i += 1
        if i >= number:
            break  # otherwise the generator would loop indefinitely

data_path = '/content/drive/MyDrive/Biometric-Ear-Recognition/Images/raw'
data_dir_list = os.listdir(data_path)
Data_Augument(data_path,data_path,data_dir_list,5)
```

```
data_path = '/content/drive/MyDrive/Biometric-Ear-Recognition/Images/raw'
data_dir_list = os.listdir(data_path)
print(len(data_dir_list))
```

```
1089
```

Demo

```
[1]  import keras
     import tensorflow as tf

     print(keras.__version__)
     print(tf.__version__)
     from sklearn.utils import shuffle
     from sklearn.model_selection import train_test_split

     from keras import backend as K
     #from tensorflow import Keras as K
     #K.set_image_data_format('tf')
     tf.keras.backend.set_image_data_format('channels_last')
     from tensorflow.keras import regularizers
     from keras.utils import np_utils
     from keras.models import Sequential
     from keras.layers.core import Dense, Dropout, Activation, Flatten
     from keras.layers.convolutional import Convolution2D, MaxPooling2D
     from keras.optimizers import SGD
     import os
     import cv2
     from google.colab.patches import cv2_imshow
     import numpy as np
     from keras.preprocessing.image import ImageDataGenerator
     import matplotlib.pyplot as plt

     2.12.0
     2.12.0
```

Demo

```
PATH = os.getcwd()
# Define data path
data_path = PATH + '/drive/MyDrive/Biometric-Ear-Recognition/Images/raw'
data_dir_list = os.listdir(data_path)
sort_dir_list = sorted(data_dir_list)
print(len(sort_dir_list))
img_rows=128
img_cols=128
num_channel=1
num_epoch=200
```

```
1089
```

Demo

```
[3] img_data_list=[]

    for bmp in sort_dir_list:
        img_path = os.path.join(data_path,bmp)
        input_img= cv2.imread(img_path)
        input_img=cv2.cvtColor(input_img, cv2.COLOR_BGR2GRAY)
        input_img_resize=cv2.resize(input_img,(128,128))
        img_data_list.append(input_img_resize)

    img_data = np.array(img_data_list)
    img_data = img_data.astype('float32')
    img_data /= 255
    print (img_data.shape)
```

```
(1089, 128, 128)
```

```
if num_channel==1:
  if tf.keras.backend.image_data_format()=='th':
    img_data= np.expand_dims(img_data, axis=1)
    print (img_data.shape)
  else:
    img_data= np.expand_dims(img_data, axis=3)
    print (img_data.shape)

else:
  if K.backend.image_data_format()=='th':
    img_data=np.rollaxis(img_data,3,1)
    print (img_data.shape)
```

```
(1089, 128, 128, 1)
```

Demo

```
#%%
# Assigning Labels

# Define the number of classes
num_classes = 51

num_of_samples = img_data.shape[0]

labels = np.ones((num_of_samples,),dtype='int64')

count = 0
precount = count
count_class = 1
for img in  sort_dir_list:
  if int(img[1:3]) == count_class:
    count+=1
  else:
    labels[precount:count] = count_class
    precount = count
    count_class += 1
    count+=1
labels[precount:6681] = count_class
# convert class labels to on-hot encoding
Y = np_utils.to_categorical(labels, num_classes)
print(Y)
```

```
[[0. 1. 0. ... 0. 0. 0.]
 [0. 1. 0. ... 0. 0. 0.]
 [0. 1. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 1.]
 [0. 0. 0. ... 0. 0. 1.]
 [0. 0. 0. ... 0. 0. 1.]]
```

Demo

```
[6] #Shuffle the dataset
    x,y = shuffle(img_data,Y, random_state=2)
    # Split the dataset
    X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=2)
    print(y_test)
```

```
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 1. 0. 0.]]
```

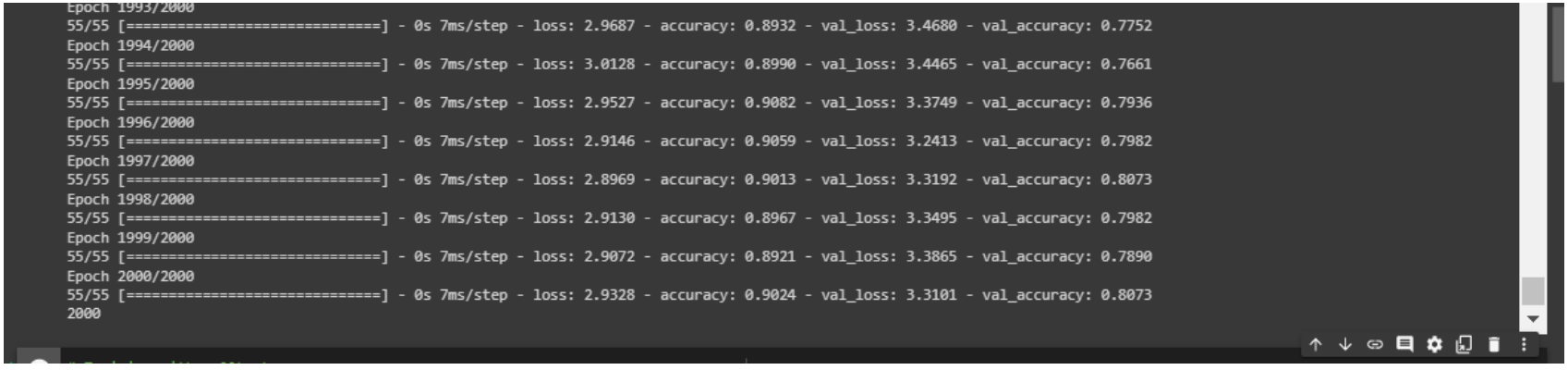
Demo

```
#%%
# Defining the model
# img_data = tf.expand_dims(img_data,axis=-1)
print(img_data.shape)
input_shape=img_data[0].shape
print(input_shape)

model = Sequential()
model.add(Convolution2D(32, 3, padding='same', activation='relu', input_shape= input_shape))
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.25))

model.add(Convolution2D(64, (3, 3), padding='same', activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.5))

model.add(Convolution2D(128, (3, 3), padding='same', activation='relu'))
model.add(Convolution2D(128, (3, 3), padding='same', activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.5))

model.add(Flatten())
model.add(Dense(512, activation='relu', kernel_regularizer=regularizers.l2(0.01)))
model.add(Dropout(0.5))

model.add(Dense(256, activation='relu', kernel_regularizer=regularizers.l2(0.01)))
model.add(Dropout(0.5))

model.add(Dense(num_classes, activation='softmax'))

#sgd = SGD(lr=0.01, decay=5e-4, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer= 'adam',metrics=["accuracy"])
#model.compile(loss='categorical_crossentropy', optimizer='rmsprop',metrics=["accuracy"])

# Viewing model_configuration

model.summary()
model.get_config()
model.layers[0].get_config()
model.layers[0].input_shape
model.layers[0].output_shape
model.layers[0].get_weights()
np.shape(model.layers[0].get_weights()[0])
model.layers[0].trainable
```

Demo

```
(1089, 128, 128, 1)
(128, 128, 1)
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 128, 128, 32)      320

 max_pooling2d (MaxPooling2D  (None, 64, 64, 32)       0
 )

 dropout (Dropout)           (None, 64, 64, 32)        0

 conv2d_1 (Conv2D)           (None, 64, 64, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 32, 32, 64)       0
 2D)

 dropout_1 (Dropout)         (None, 32, 32, 64)        0

 conv2d_2 (Conv2D)           (None, 32, 32, 128)       73856

 conv2d_3 (Conv2D)           (None, 32, 32, 128)       147584

 max_pooling2d_2 (MaxPooling  (None, 16, 16, 128)      0
 2D)

 dropout_2 (Dropout)         (None, 16, 16, 128)       0

 flatten (Flatten)           (None, 32768)             0

 dense (Dense)               (None, 512)               16777728

 dropout_3 (Dropout)         (None, 512)               0

 dense_1 (Dense)             (None, 256)               131328

 dropout_4 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 51)                13107

=================================================================
Total params: 17,162,419
Trainable params: 17,162,419
Non-trainable params: 0
_________________________________________________________________
True
```

Demo


Epoch 1993/2000
55/55 [==============================] - 0s 7ms/step - loss: 2.9687 - accuracy: 0.8932 - val_loss: 3.4680 - val_accuracy: 0.7752
Epoch 1994/2000
55/55 [==============================] - 0s 7ms/step - loss: 3.0128 - accuracy: 0.8990 - val_loss: 3.4465 - val_accuracy: 0.7661
Epoch 1995/2000
55/55 [==============================] - 0s 7ms/step - loss: 2.9527 - accuracy: 0.9082 - val_loss: 3.3749 - val_accuracy: 0.7936
Epoch 1996/2000
55/55 [==============================] - 0s 7ms/step - loss: 2.9146 - accuracy: 0.9059 - val_loss: 3.2413 - val_accuracy: 0.7982
Epoch 1997/2000
55/55 [==============================] - 0s 7ms/step - loss: 2.8969 - accuracy: 0.9013 - val_loss: 3.3192 - val_accuracy: 0.8073
Epoch 1998/2000
55/55 [==============================] - 0s 7ms/step - loss: 2.9130 - accuracy: 0.8967 - val_loss: 3.3495 - val_accuracy: 0.7982
Epoch 1999/2000
55/55 [==============================] - 0s 7ms/step - loss: 2.9072 - accuracy: 0.8921 - val_loss: 3.3865 - val_accuracy: 0.7890
Epoch 2000/2000
55/55 [==============================] - 0s 7ms/step - loss: 2.9328 - accuracy: 0.9024 - val_loss: 3.3101 - val_accuracy: 0.8073
2000

Demo

```
[8]  # Training
     num_epoch = 2000
     hist = model.fit(X_train, y_train, batch_size=16, epochs=num_epoch, verbose=1, validation_data=(X_test, y_test))
     #hist = model.fit(X_train, y_train, batch_size=32, nb_epoch=20,verbose=1, validation_split=0.2)
     print(num_epoch)
```

Demo

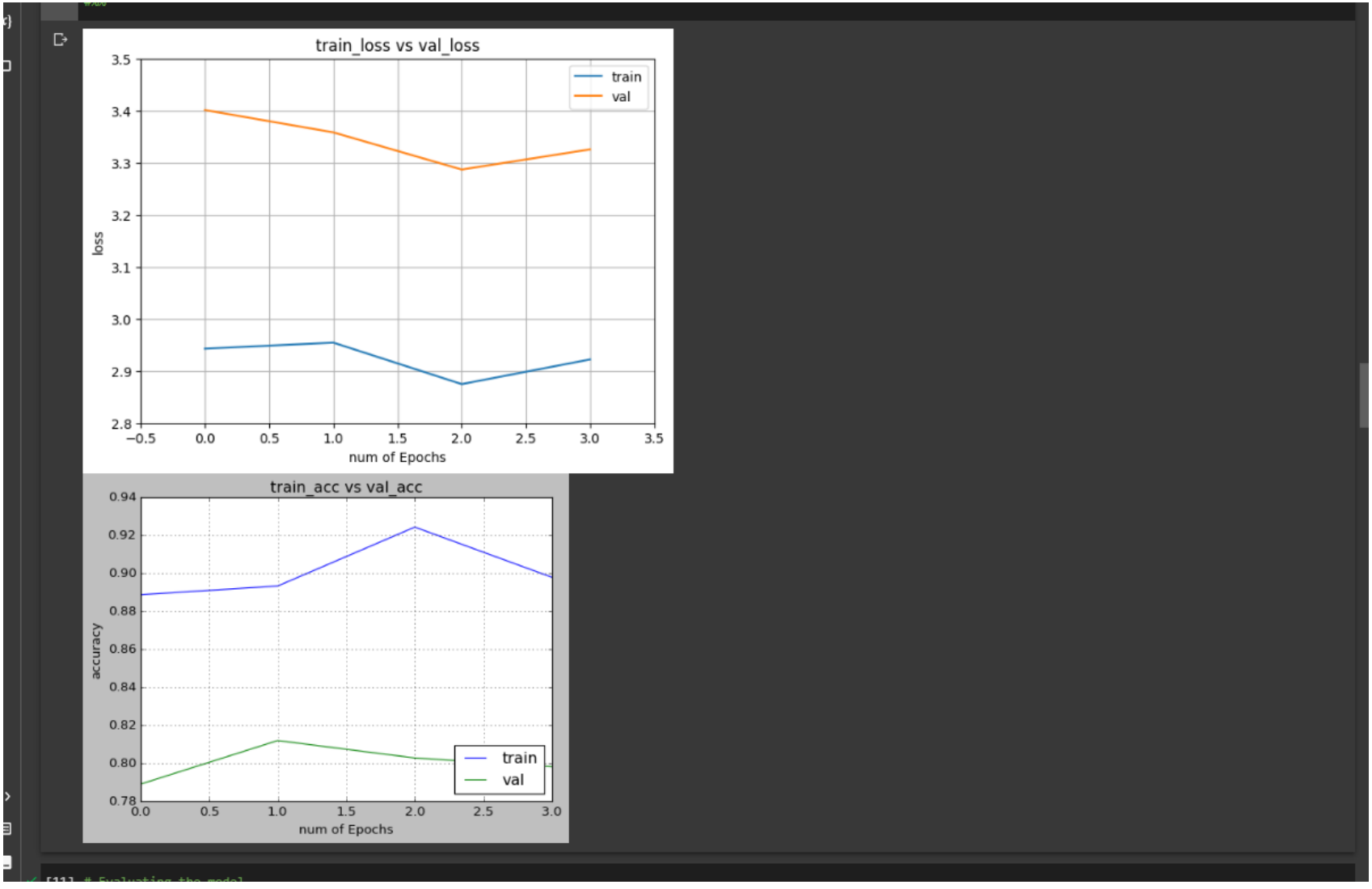
```python
# Training with callbacks
from keras import callbacks

filename='model_train_new.csv'
csv_log=callbacks.CSVLogger(filename, separator=',', append=False)

early_stopping=callbacks.EarlyStopping(monitor='val_loss', min_delta=0, patience=0, verbose=0, mode='min')

filepath="Best-weights-my_model-{epoch:03d}-{loss:.4f}-{accuracy:.4f}.hdf5"

checkpoint = callbacks.ModelCheckpoint(filepath, monitor='val_loss', verbose=1, save_best_only=True, mode='min')

callbacks_list = [csv_log,early_stopping,checkpoint]

hist = model.fit(X_train, y_train, batch_size=16, epochs=num_epoch, verbose=1, validation_data=(X_test, y_test),callbacks=callbacks_list)
```

```
Epoch 1/2000
55/55 [==============================] - ETA: 0s - loss: 2.9427 - accuracy: 0.8886
Epoch 1: val_loss improved from inf to 3.40118, saving model to Best-weights-my_model-001-2.9427-0.8886.hdf5
55/55 [==============================] - 1s 14ms/step - loss: 2.9427 - accuracy: 0.8886 - val_loss: 3.4012 - val_accuracy: 0.7890
Epoch 2/2000
51/55 [==========================>...] - ETA: 0s - loss: 2.9598 - accuracy: 0.8897
Epoch 2: val_loss improved from 3.40118 to 3.35825, saving model to Best-weights-my_model-002-2.9543-0.8932.hdf5
55/55 [==============================] - 1s 12ms/step - loss: 2.9543 - accuracy: 0.8932 - val_loss: 3.3583 - val_accuracy: 0.8119
Epoch 3/2000
51/55 [==========================>...] - ETA: 0s - loss: 2.8784 - accuracy: 0.9240
Epoch 3: val_loss improved from 3.35825 to 3.28684, saving model to Best-weights-my_model-003-2.8746-0.9242.hdf5
55/55 [==============================] - 1s 12ms/step - loss: 2.8746 - accuracy: 0.9242 - val_loss: 3.2868 - val_accuracy: 0.8028
Epoch 4/2000
51/55 [==========================>...] - ETA: 0s - loss: 2.9171 - accuracy: 0.9020
Epoch 4: val_loss did not improve from 3.28684
55/55 [==============================] - 0s 7ms/step - loss: 2.9221 - accuracy: 0.8978 - val_loss: 3.3256 - val_accuracy: 0.7982
```

Demo

```
#%%
# visualizing losses and accuracy
train_loss=hist.history['loss']
val_loss=hist.history['val_loss']
train_acc=hist.history['accuracy']
val_acc=hist.history['val_accuracy']
xc=range(len(train_loss))
plt.figure(1,figsize=(7,5))
plt.plot(xc,train_loss)
plt.plot(xc,val_loss)
plt.xlabel('num of Epochs')
plt.ylabel('loss')
plt.title('train_loss vs val_loss')
plt.grid(True)
plt.legend(['train','val'])
#print plt.style.available # use bmh, classic,ggplot for big pictures
plt.style.use(['classic'])

plt.figure(2,figsize=(7,5))
plt.plot(xc,train_acc)
plt.plot(xc,val_acc)
plt.xlabel('num of Epochs')
plt.ylabel('accuracy')
plt.title('train_acc vs val_acc')
plt.grid(True)
plt.legend(['train','val'],loc=4)
#print plt.style.available # use bmh, classic,ggplot for big pictures
plt.style.use(['classic'])

#%%
```

Demo


train_loss vs val_loss
train
val
loss
num of Epochs
train_acc vs val_acc
accuracy
train
val
num of Epochs

Demo

```
# Evaluating the model

score = model.evaluate(X_test, y_test, verbose=0)
print('Test Loss:', score[0])
print('Test accuracy:', score[1])

test_image = X_test[0:100]
print (test_image.shape)
y_pred = model.predict(test_image)
predicted_classes = np.argmax(y_pred[0])
print(predicted_classes)
print(np.argmax(y_test[0:100]))
```

```
Test Loss: 3.3256077766418457
Test accuracy: 0.7981651425361633
(100, 128, 128, 1)
4/4 [==============================] - 0s 28ms/step
34
34
```

Demo

```
# Testing a new image
# test_image = cv2.imread('raw/002_3_0_1923.bmp')

def predict_new_image(test_link,name):
  link_image = os.path.join(test_link,name)
  test_image = cv2.imread(link_image)
  test_image=cv2.cvtColor(test_image, cv2.COLOR_BGR2GRAY)
  test_image=cv2.resize(test_image,(128,128))
  test_image = np.array(test_image)
  test_image = test_image.astype('float32')
  test_image /= 255

  if num_channel==1:
    if K.image_data_format()=='th':
      test_image= np.expand_dims(test_image, axis=0)
      test_image= np.expand_dims(test_image, axis=0)
      #print (test_image.shape)
    else:
      #test_image= np.expand_dims(test_image, axis=3)
      test_image= np.expand_dims(test_image, axis=2)
      test_image= np.expand_dims(test_image, axis=0)
      #print (test_image.shape)

  else:
    if K.image_data_format()=='th':
      test_image=np.rollaxis(test_image,2,0)
      test_image= np.expand_dims(test_image, axis=0)
      #print (test_image.shape)
    else:
      test_image= np.expand_dims(test_image, axis=0)
      #print (test_image.shape)

  # Predicting the test image
  y_pre = model.predict(test_image)
  temp = y_pre[0]
  predict_classes = np.argmax(temp)
  return (int(name[1:3]),predict_classes)
test_link = '/content/drive/MyDrive/Biometric-Ear-Recognition/Images/test'
test_dir_list = os.listdir(test_link)
acc_test = 0
for img in test_dir_list:
  y,pre_y = predict_new_image(test_link,img)
  print(y,pre_y)
  if y == pre_y:
    acc_test+=1
print(acc_test/186)
```

```
45 49
```

Demo

```
37 37
1/1 [==============================] - 0s 19ms/step
48 48
1/1 [==============================] - 0s 19ms/step
31 31
1/1 [==============================] - 0s 19ms/step
43 22
1/1 [==============================] - 0s 20ms/step
42 42
1/1 [==============================] - 0s 19ms/step
46 46
1/1 [==============================] - 0s 21ms/step
49 49
1/1 [==============================] - 0s 20ms/step
49 42
1/1 [==============================] - 0s 20ms/step
50 50
1/1 [==============================] - 0s 20ms/step
50 26
0.8118279569892473
```

Demo

```python
# Printing the confusion matrix
from sklearn.metrics import classification_report,confusion_matrix
import itertools

Y_pred = model.predict(X_test)
print(Y_pred)
y_pred = np.argmax(Y_pred, axis=1)
print(y_pred)
#y_pred = model.predict_classes(X_test)
#print(y_pred)
target_names = ['true label']

#print(classification_report(np.argmax(y_test,axis=1), y_pred,target_names=target_names))

print(confusion_matrix(np.argmax(y_test,axis=1), y_pred))
```

```
7/7 [==============================] - 0s 3ms/step
[[1.21e-18 1.09e-10 2.94e-13 ... 1.54e-08 3.05e-19 5.39e-15]
 [5.69e-12 3.71e-06 4.12e-09 ... 1.65e-09 9.66e-11 2.02e-10]
 [2.60e-11 9.28e-08 1.22e-06 ... 2.79e-06 6.54e-08 2.19e-11]
 ...
 [2.72e-14 5.03e-07 2.84e-14 ... 1.06e-05 4.32e-01 1.43e-02]
 [3.55e-23 1.07e-08 1.67e-08 ... 1.66e-13 1.21e-20 1.83e-12]
 [1.02e-25 1.64e-17 9.08e-14 ... 1.00e+00 8.24e-15 3.00e-11]]
[34 35 13 17 29 44  7 27 15 27 15 33 12 20 13 17 14 25 34 31 47 36  1 14
 14 41 39 22 47  6 18 29  2 18 35 24 44 37 32 37 42  2 29  4 10 31 23 42
  5 30 28  8 13 21  6 20 31  2 12 32 39 30 30 50 10 16 46  2 29  6 47 41
 44  4 39 28 26 39 49 13 31 12 13  9 44 21 40 19 21 33 31 17  8 16  7 33
 22 13 13 20 23 42  4 34  2  1  1  1 28 34 26 49 13 40  2  7  4 20 15 34
 10 47 13 31 41  1  2 22 35 10 24 33 48 12 23  5 21 48 10 14 30  6 50 13
 17 30 13 38  7 44 13  9 13 47 37 36 48 22 30 28  2 20 41 19 32 48 36 20
  2 32  6 39  5 23 14 23 44 27 39 16 50 28 20  3  1 48 46 30 37 33  3  6
 40 23 33 36 48 10 27 13 19  8 25  8 12 37  4 42 34 28 39  1 12 26 42  7
 32 48]
[[6 0 0 ... 0 0 0]
 [0 8 0 ... 0 0 0]
 [0 0 2 ... 0 0 0]
 ...
 [0 0 0 ... 7 0 0]
 [0 0 0 ... 0 2 0]
 [0 0 0 ... 0 0 3]]
```

Demo

```
# Plotting the confusion matrix
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
# Compute confusion matrix
cnf_matrix = (confusion_matrix(np.argmax(y_test,axis=1), y_pred))

np.set_printoptions(precision=2)

plt.figure()

# Plot non-normalized confusion matrix
plot_confusion_matrix(cnf_matrix, classes=target_names,
                      title='Confusion matrix')
#plt.figure()
# Plot normalized confusion matrix
#plot_confusion_matrix(cnf_matrix, classes=target_names, normalize=True,
#                      title='Normalized confusion matrix')
#plt.figure()
plt.show()
```

Demo


Confusion matrix, without normalization
[[6 0 0 ... 0 0 0]
[0 8 0 ... 0 0 0]
[0 0 2 ... 0 0 0]
...
[0 0 0 ... 7 0 0]
[0 0 0 ... 0 2 0]
[0 0 0 ... 0 0 3]]
Confusion matrix
true label
True label
true label
Predicted label
10
8
6
4
2
0

Demo

```
# Saving and loading model and weights
from keras.models import model_from_json
from keras.models import load_model

# serialize model to JSON
model_json = model.to_json()
with open("model.json", "w") as json_file:
    json_file.write(model_json)
# serialize weights to HDF5
model.save_weights("model.h5")
print("Saved model to disk")

# load json and create model
json_file = open('model.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(loaded_model_json)
# load weights into new model
loaded_model.load_weights("model.h5")
print("Loaded model from disk")

model.save('model_new.hdf5')
loaded_model=load_model('model_new.hdf5')
model=load_model('model_new.hdf5')
```

```
Saved model to disk
Loaded model from disk
```

Xin cảm ơn thầy và các bạn đã lắng nghe.