

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY  
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

---



# GRADUATION THESIS

SUBMITTED IN PARTIAL FULFILLMENT  
OF THE REQUIREMENTS FOR THE DEGREE OF

## ENGINEER

IN

### INFORMATION TECHNOLOGY

**TODO** TITLE

Author : **Vi Thanh Dat**  
Class : CNTT2.02 K61  
Student ID : 20164803

Supervisor : **TODO** supervisor

HANOI, 5 - 2021

# **Abstract**

Placeholder

# Tóm tắt

Placeholder

# Summary

The thesis is divided into 5 chapters:

- Chapter 1 describes the problem of training neural networks at scale, as well as the object detection problem that would be used as a case study.
- Chapter 2 presents the theoretical background as well as related works, including machine learning, object detection and distributed training.
- Chapter 3 details the design and implementation of BK.Synapse - the proposed framework - and a case study where it is applied to an object detection problem.
- Chapter 4 reports on the results of several experiments and benchmarks.
- Chapter 5 presents our conclusions and lays out plans for future works.

# Contents

<b>List of Figures</b>	<b>1</b>
<b>List of Tables</b>	<b>3</b>
<b>1 Giới thiệu về nhận dạng người nói</b>	<b>1</b>
1.1 Giới thiệu chung về bài toán nhận dạng người nói . . . . .	1
1.2 Nhận dạng người nói trong tiếng Việt . . . . .	4
1.3 Mục tiêu và phạm vi đồ án . . . . .	4
1.4 Bố cục đồ án . . . . .	4
<b>2 Cơ sở lý thuyết</b>	<b>5</b>
2.1 Học máy . . . . .	5
2.1.1 Tổng quan về học máy . . . . .	5
2.1.2 Mạng nơ-ron nhân tạo . . . . .	6
2.2 Học sâu . . . . .	10
2.2.1 Mạng nơ-ron tích chập . . . . .	10
2.2.2 Mạng nơ-ron residual . . . . .	14
2.3 Hàm mất mát . . . . .	15
2.4 Trích xuất thông tin . . . . .	17
<b>3 Nhận dạng người nói tiếng Việt</b>	<b>18</b>
3.1 Nghiên cứu liên quan . . . . .	18

3.2 Mô hình cơ sở . . . . .	20
3.2.1 Biểu diễn khung âm thanh bằng mạng ResNet . . . . .	20
3.2.2 Tổng hợp thống kê tập trung . . . . .	21
3.2.3 Hàm mất mát Angular Prototypical . . . . .	23
3.3 Đề xuất cải tiến mô hình . . . . .	24
3.3.1 Transfer learning . . . . .	25
3.3.2 SGD với momentum khái quát hoá tốt hơn Adam . . . . .	26
3.3.3 Hàm mất mát Angular Margin Prototypical . . . . .	28
<b>4 Thực nghiệm và đánh giá</b>	<b>31</b>
4.1 Cơ sở dữ liệu . . . . .	31
4.1.1 Cải thiện chất lượng bộ dữ liệu . . . . .	32
4.1.2 Bộ dữ liệu thực nghiệm . . . . .	36
4.2 Chi tiết cài đặt thực nghiệm . . . . .	36
4.3 Kết quả thực nghiệm và đánh giá . . . . .	37
<b>5 Conclusions</b>	<b>41</b>
<b>Bibliography</b>	<b>42</b>

# List of Figures

1.1 Nhận định người nói và xác minh người nói . . . . .	2
1.2 Tổng quan hệ thống nhận diện người nói . . . . .	3
2.1 Cấu tạo nơ-ron sinh học <sup>1</sup> . . . . .	7
2.2 Cấu trúc của mạng nơ-ron nhân tạo <sup>2</sup> . . . . .	7
2.3 Phương pháp gradient descent <sup>3</sup> . . . . .	9
2.4 Các thành phần cơ bản của mạng tích chập <sup>4</sup> . . . . .	11
2.5 Các đặc trưng học được trong lớp tích chập <sup>5</sup> . . . . .	11
2.6 Một ví dụ của lớp tích chập <sup>6</sup> . . . . .	12
2.7 Một ví dụ của tổng hợp cực đại và tổng hợp trung bình [48] . . .	13
2.8 Skip connection trong ResNet [39] . . . . .	14
2.9 Các kiến trúc khác nhau của ResNet [39] . . . . .	15
2.10 Kiến trúc hai khối residual trong các kiến trúc mạng ResNet [39] .	15
2.11 Ví dụ hàm mất mát triplet [26] . . . . .	16
2.12 Thuật toán trích xuất MFCCs [5] . . . . .	17
3.1 Tổng quan mô hình nhận diện người nói [37] . . . . .	19
3.2 Không gian embedding với học biểu diễn[44] . . . . .	20
3.3 Tổng quan mô hình cơ sở sử dụng trong đồ án . . . . .	21
3.4 Tổng hợp thống kê tập trung [27] . . . . .	22
3.5 Hàm mất mát Angular Prototypical . . . . .	24

3.6 Sơ đồ mô tả transfer learning sử dụng kiến thức hiện có cho các tác vụ mới . . . . .	25
3.7 Sơ đồ mô tả transfer learning sử dụng kiến thức hiện có cho các tác vụ mới [31] . . . . .	25
3.8 Lợi ích của transfer learning đối với việc huấn luyện mô hình [4] .	26
3.9 Mô tả biểu diễn người nói học bởi hàm softmax trong không gian góc. Đường kẻ chấm đen là đường phân giác giữa 2 tâm . . . . .	28
3.10 Mô tả biểu diễn người nói học bởi hàm AMP-arc trong không gian góc. . . . .	29
4.1 Biểu đồ phân phối số danh tính theo số câu nói của bộ dữ liệu ZaloAI	32
4.2 Ma trận tương đồng cho một tập 10 đoạn âm thanh của một người	33
4.3 Ma trận tương đồng của một danh tính bị loại bỏ . . . . .	34
4.4 Ma trận tương đồng của một danh tính có đoạn âm thanh không hợp lệ . . . . .	35
4.5 . . . . .	36
4.6 Mô tả EER <sup>7</sup> . . . . .	37

# List of Tables

3.1 placeholder . . . . .	21
4.1 placeholder . . . . .	35
4.2 EER trên tập kiểm tra với dữ liệu trước và sau khi cải thiện chất lượng . . . . .	38
4.3 EER trên tập kiểm tra với các phương pháp huấn luyện và dữ liệu khác nhau . . . . .	39
4.4 EER trên tập kiểm tra của mô hình huấn luyện với dữ liệu còn nhiều âm thanh và đã khử tạp âm . . . . .	39
4.5 EER trên tập kiểm tra của mô hình huấn luyện với Adam và SGD	39
4.6 EER trên tập kiểm tra của mô hình huấn luyện với hàm mất mát AP, AMP-cos và AMP-arc với các giá trị phạt khác nhau . . . . .	40

# Chapter 1

## Giới thiệu về nhận dạng người nói

Trong chương này, tác giả giới thiệu tổng quan về bài toán nhận dạng người nói, điểm qua các nghiên cứu liên quan và giới thiệu về bài toán trong đồ án.

### 1.1 Giới thiệu chung về bài toán nhận dạng người nói

Nhận dạng người nói (speaker recognition) là quá trình tự động nhận dạng người đang nói bằng cách sử dụng thông tin riêng biệt của người nói đó có trong tín hiệu âm thanh. Nhận dạng người nói được ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau, tuy nhiên cũng gặp không ít khó khăn khi triển khai trong thực tế. Do vậy, nghiên cứu bài toán nhận dạng người nói rất được quan tâm bởi nhiều nhà khoa học trên thế giới. Một số ứng dụng của bài toán có thể kể đến như:

- Bảo mật cho các hệ thống tài chính, ngân hàng: người dùng dùng giọng nói kết hợp với các lớp bảo mật khác cho xác thực để tăng tính bảo mật khi giao dịch.
- Tăng trải nghiệm khách hàng trong tổng đài chăm sóc khách hàng.
- Xác định danh tính tội phạm trong an ninh khi thu được dữ liệu giọng nói.
- Kết hợp với các hệ thống nhận dạng tiếng nói để xây dựng ứng dụng gõ bằng cuộc họp.
- ...

Dựa vào ứng dụng, nhận dạng người được phân loại thành nhận định người nói (speaker identification) và xác minh người nói (speaker verification) (Hình 1.1).

Trong nhận định người nói, một đoạn tiếng nói từ một người không xác định được phân tích và so sánh với mô hình giọng nói của những người đã biết. Người này được nhận định là người mô hình giọng nói phù hợp nhất với câu nói đầu vào. Trong xác minh người nói, một người lạ xác nhận một danh tính đã biết; đoạn tiếng nói của người này được so sánh với mô hình giọng nói của danh tính đang được xác nhận. Nếu điểm tương đồng đủ tốt, nghĩa là trên một ngưỡng nào đó, danh tính của người lạ được chấp nhận. Ngưỡng cao khiến những kẻ mạo danh khó được chấp nhận bởi hệ thống, nhưng có nguy cơ chối nhầm người dùng hợp lệ. Ngược lại, ngưỡng thấp cho phép chấp nhận người dùng hợp lệ một cách nhất quán, nhưng có nguy cơ chấp nhận những người giả mạo.

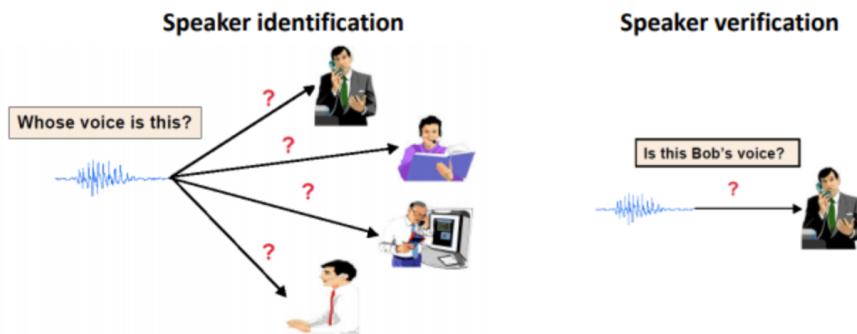


Figure 1.1: Nhận định người nói và xác minh người nói

Thông thường, hệ thống nhận dạng người nói được chia thành 3 pha như mô tả trong Hình 1.2

- Pha 1: phát triển (development). Trong pha này, mô hình có khả năng biểu diễn đặc trưng người nói được huấn luyện và tối ưu trên một cơ sở dữ liệu các đoạn tiếng nói.
- Pha 2: ghi danh (enrollment). Trong pha ghi danh, biểu diễn của người dùng mới được trích xuất bằng mô hình phát triển ở pha 1 và lưu trữ trong cơ sở dữ liệu để phục vụ cho pha 3.
- Pha 3: kiểm tra (testing). Với bài toán nhận định người nói, vec-tơ biểu diễn của câu nói đầu vào được so sánh với tất cả biểu diễn trong cơ sở dữ liệu để tìm ra người dùng tương đồng nhất. Trong bài toán xác thực người nói, được cung cấp danh tính đầu vào, hệ thống chỉ so sánh đoạn tiếng nói đầu vào và đoạn của danh tính trong hệ thống để đưa ra quyết định.

**TODO** Vẽ lại hình này và include development phase to make it more clear

Trong thực tế, hiệu năng của hệ thống nhận diện người nói bị suy giảm do sự khác biệt của các kênh và phiên giữa tín hiệu giọng nói trong pha ghi danh và

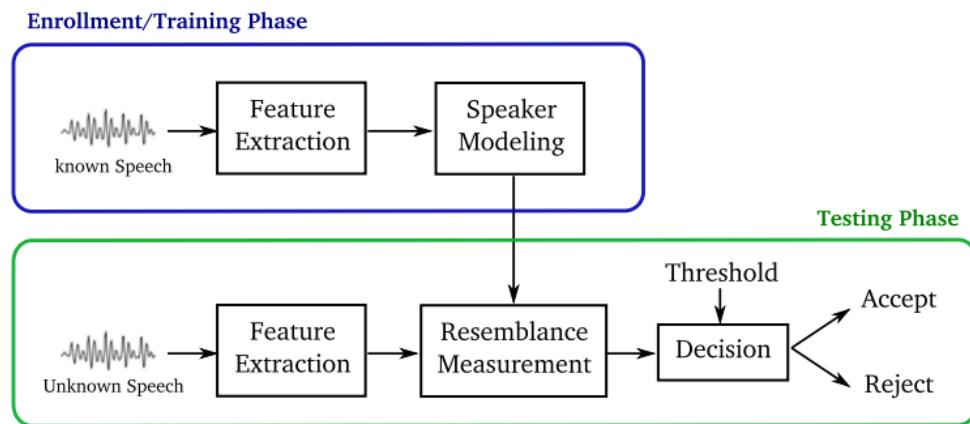


Figure 1.2: Tổng quan hệ thống nhận diện người nói

pha kiểm tra. Các yếu tố làm ảnh hưởng tới tín hiệu giọng nói bao gồm:

- Sử dụng các loại micrô khác nhau khi thu tín hiệu đăng ký và kiểm tra.
- Điều kiện tiếng ồn và độ vang của môi trường.
- Sự khác biệt trong giọng nói của người nói ở các giai đoạn khác nhau của độ tuổi, sức khoẻ, phong cách nói và trạng thái cảm xúc.
- Các kênh truyền như các loại điện thoại di động khác nhau, micrô, giao thức truyền âm thanh qua internet có thể làm thay đổi giọng nói.

Dựa vào sự tương đồng của các câu nói đầu vào, phương pháp giải quyết bài toán nhận dạng người nói còn có thể chia thành nhóm dựa vào văn bản (text-dependent speaker recognition - TDSR) và nhóm không dựa vào văn bản (text-independent speaker recognition - TISR). Các hệ thống TDSR yêu cầu người nói cung cấp các đoạn tiếng nói có nội dung theo một từ hoặc câu được định sẵn; nội dung các câu nói phải được giữ nhất quán trong cả quá trình huấn luyện và nhận dạng. Ngược lại, TISR không yêu cầu người dùng phải thu theo bất cứ một văn bản nào. Với các đoạn tiếng nói ngắn, các hệ thống TDSR đã có thể đạt được hiệu suất nhận diện cao, trong khi các TISR yêu cầu các câu nói dài để huấn luyện các mô hình đáng tin cậy và đạt được hiệu suất tốt. Do sự bất tiện của các hệ thống TDSR, trong vài năm gần đây, cộng đồng nghiên cứu tập trung phát triển các mô hình học sâu end-to-end nhằm mục đích biểu diễn các đoạn tiếng nói ngắn.

## **1.2 Nhận dạng người nói trong tiếng Việt**

## **1.3 Mục tiêu và phạm vi đồ án**

## **1.4 Bố cục đồ án**

## Chapter 2

# Cơ sở lý thuyết

Chương 2 trình bày cơ sở lý thuyết về học máy và học sâu. Trong chương này, tác giả trình bày về các hàm mất mát phân loại phổ biến và phương pháp trích xuất đặc trưng từ tín hiệu âm thanh. Đây là nền tảng để xây dựng mô hình trong chương 3.

### 2.1 Học máy

#### 2.1.1 Tổng quan về học máy

Học máy (Machine learning - ML) là một nhánh con của trí tuệ nhân tạo. Nghiên cứu học máy tập trung phát triển và xây dựng các thuật toán và kỹ thuật để giúp chương trình máy tính có thể "học" thi thực một tác nào đó từ kinh nghiệm. Trong [23], nhà tiên phong về học máy Tom M. Mitchell định nghĩa như sau: Một chương trình máy tính được nói là học từ kinh nghiệm  $E$  để thực thi một tác vụ  $T$  nếu khả năng của chương trình được đo bằng độ đo  $P$  tiến bộ với kinh nghiệm  $E$ . Rõ ràng hơn, học máy là những chương trình máy tính có thể tự học được dữ liệu mà không cần được lập trình một cách cụ thể.

Với các cách định nghĩa  $T$ ,  $P$ , và  $E$  khác nhau, các thuật toán học máy có thể được phân vào các nhóm: học có giám sát, học không giám sát, học bán giám sát, và học tăng cường.

**Học có giám sát** là lớp các thuật toán sử dụng dữ liệu được gán nhãn từ trước để tìm mối liên hệ giữa đầu vào và đầu ra. Mục tiêu của học có giám sát là tạo ra mô hình có thể dự đoán được đầu ra cho các đầu vào mới mà mô hình chưa gặp bao giờ. Tập dữ liệu gán nhãn trên được gọi là dữ liệu huấn luyện. Cho tập dữ liệu huấn luyện gồm  $N$  ví dụ  $D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$  trong đó  $\mathbf{x}_i$  là

vec-tơ đặc trưng của ví dụ thứ  $i$  và  $y_i$  là nhãn tương ứng. Thuật toán học có giám sát tìm hàm ánh xạ  $f : X \longrightarrow Y$  trong đó  $X$  là không gian vec-tơ đầu vào và  $Y$  là tập các nhãn.

Các thuật toán học có giám sát thường được sử dụng để giải quyết hai loại bài toán: phân loại (classification) và hồi quy (regression). Bài toán phân loại có nhãn rời rạc thuộc một tập cho sẵn. Ví dụ: phân loại đồ vật, phân loại phương tiện giao thông, phân loại giới tính dựa trên giọng nói. Khác với bài toán phân loại, bài toán hồi quy lại có nhãn nằm trên một miền liên tục. Ví dụ: dự đoán giá nhà đất, dự đoán thị trường chứng khoán.

Trong **học không giám sát**, khác với học có giám sát, ta chỉ có dữ liệu đầu vào mà không có dữ liệu đầu ra. Mục tiêu chính của các thuật toán là tìm ra cấu trúc ẩn trong dữ liệu hoặc trích xuất đặc trưng chung của tập dữ liệu.

**Học bán giám sát** nằm giữa học có giám sát và không có giám sát. Học bán giám sát thường được sử dụng khi có một lượng lớn dữ liệu không có nhãn và một số ít dữ liệu có nhãn. Phương pháp học bán giám sát nổi tiếng nhất - tự huấn luyện (self-training) sử dụng mô hình huấn luyện trên tập có nhãn để sinh nhãn giả từ tập không nhãn nhằm tăng cường khả năng học của mô hình.

Hệ thống **học củng cố** được xem như là một tác tử trong một môi trường vô định và phức tạp. Với mỗi hành động, tác tử này nhận điểm thưởng hoặc phạt từ môi trường. Mục tiêu của học củng cố là các tác tử thông minh để tối đa điểm thưởng và thực hiện tác vụ của nó. Học củng cố hiện đang là lĩnh vực nghiên cứu tập trung nhiều nguồn lực và công sức với nhiều ứng dụng thực tế liên quan tới điều khiển robot hay vận hành nhà máy một cách tự động.

### 2.1.2 Mạng nơ-ron nhân tạo

Mạng nơ-ron nhân tạo (Artificial neural networks - ANN) là một mô hình tính toán mô phỏng cách nơ-ron trong não người phân tích và xử lý thông tin. Học sâu với nền tảng là ANN cho phép xử lý dữ liệu và thông tin với độ chính xác vượt xa các mô hình xác suất cổ điển và tốc độ vượt trội so với con người.

Nơ-ron sinh học gồm 3 phần chính: dendrite, thân tế bào (soma) và axon (Hình 2.2). Đầu tiên, tín hiệu đầu vào được thu thập bởi các khớp thần kinh của dendrite. Vai trò của soma là xử lý đầu vào và tổng hợp thông tin dựa vào độ quan trọng của tín hiệu. Sau đó, axon sẽ truyền thông tin đã được xử lý tới đầu

---

<sup>1</sup><https://app.biorender.com/biorender-templates/t-5f5b7e6139954000b2bde860-neuron-anatomy>

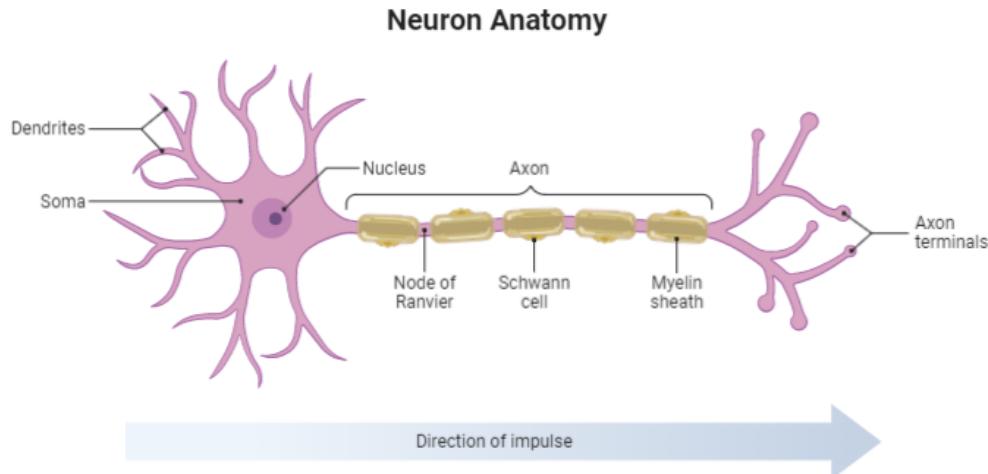


Figure 2.1: Cấu tạo nơ-ron sinh học <sup>1</sup>

ra. Thành phần cấu thành nên ANN là perceptron cũng có cách thức hoạt động tương tự. Perceptron bao gồm lớp đầu vào, tập trọng số cùng hàm kích hoạt để tổng hợp thông tin và lớp đầu ra, tương tự như dendrite, soma và axon.

Mạng nơ-ron là mô hình gồm nhiều lớp chồng lên nhau, mỗi lớp bao gồm nhiều perceptron (hay nơ-ron) tổng hợp thông tin từ lớp trước, mỗi lớp có một tập các nơ-ron độc lập. Lớp đầu tiên trong mạng được gọi là lớp đầu vào (input layer), lớp cuối cùng được gọi là lớp đầu ra (output layer), các lớp ở giữa được gọi là lớp ẩn (hidden layer) (Hình 2.2).

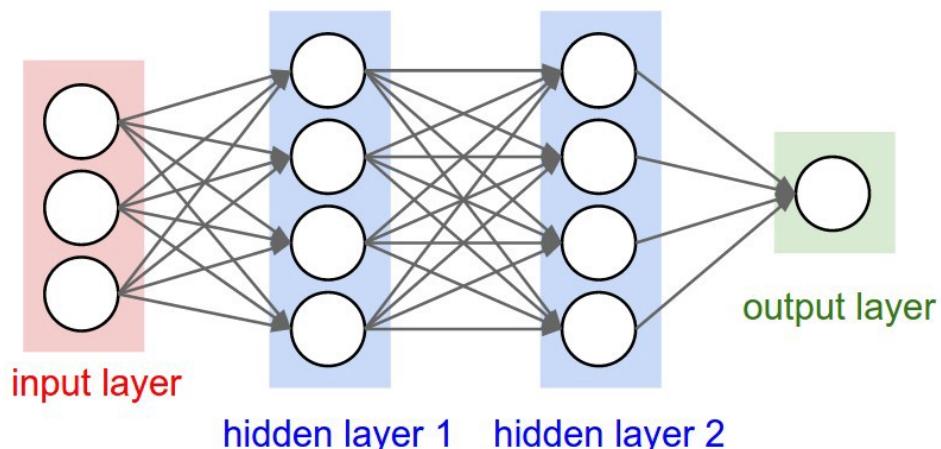


Figure 2.2: Cấu trúc của mạng nơ-ron nhân tạo<sup>2</sup>

Cho một mạng nơ-ron nhân tạo gồm  $L$  lớp, với  $\mathbf{a}^i$  là tập các nơ-ron trong lớp thứ  $i$ , hai lớp nơ-ron liên tiếp có chỉ số  $i$  và  $i + 1$  được kết nối với nhau bằng một ma trận trọng số  $\mathbf{W}^i$  và một vec-tơ bias  $\mathbf{b}^i$ . Ta có thể mô tả quá trình tính

<sup>2</sup><https://towardsdatascience.com/vanilla-neural-networks-in-r-43b028f415>

toán các nơ-ron từ lớp đầu vào và đầu ra dưới dạng toán học như sau:

$$\mathbf{a}^{(i)} = \sigma(\mathbf{W}^i \mathbf{a}^{(i-1)} + \mathbf{b}^i), \quad 0 \leq i \leq L \quad (2.1)$$

Trong đó,  $\sigma$  được gọi là hàm kích hoạt (activation function). Hàm kích hoạt là một thành phần quan trọng không thể thiếu trong mạng nơ-ron nhân tạo. Để mạng nơ-ron nhân tạo có thể xấp xỉ hay học được những biến đổi phức tạp,  $\sigma$  phải là một hàm phi tuyến. Nếu không có hàm kích hoạt hay hàm kích hoạt là tuyến tính thì mạng nơ-ron dù có nhiều lớp đến đâu cũng có thể quy về một hàm tuyến tính và không có khả năng học được nhiều thông tin từ dữ liệu.

Có rất nhiều hàm kích hoạt khác nhau, ví dụ như hàm ReLU, hàm sigmoid, hàm Tanh, hàm softplus,... Trong các mạng nơ-ron hiện đại, hàm ReLU [25] được sử dụng phổ biến nhất do có nhiều lợi ích cho quá trình huấn luyện và tốc độ tính toán nhanh:

$$ReLU(x) = \max(0, x) \quad (2.2)$$

Quá trình lần lượt tính toán mạng nơ-ron từ lớp đầu vào, tới lớp ẩn và cuối cùng là lớp đầu ra được gọi là quá trình lan truyền tiến (feedforward).

Mạng nơ-ron học từ quá trình đối nghịch với lan truyền tiến gọi là lan truyền ngược (backpropagation). Với vec-tơ đặc trưng đầu vào  $\mathbf{x}$  với nhãn tương ứng  $y$ , đầu tiên ta tính giá trị của nơ-ron lớp đầu ra  $\mathbf{a}^{L-1}$  bằng quá trình lan truyền tiến. Sau đó, ta "lan truyền" sai khác giữa  $\mathbf{a}^{L-1}$  và  $y$  tới toàn mạng để điều chỉnh các bộ trọng số  $\mathbf{W}^i$  và  $\mathbf{b}^i$  với mục tiêu để giảm sai khác này.

Để đo đạc sự sai khác trong đầu ra của mạng và nhãn đầu vào, ta sử dụng một hàm mất mát. Dựa vào tác vụ khác nhau của bài toán ta cần sử dụng các hàm mất mát khác nhau. Thông thường, bài toán hồi quy sử dụng hàm trung bình bình phương sai số (mean square error - MSE), còn bài toán phân loại sử dụng hàm entropy chéo (cross entropy - CE). Với tập dữ liệu  $D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$ , giả sử lớp cuối cùng của mạng có một nơ-ron, gọi  $a_i^{L-1}$  là giá trị của nơ-ron đầu ra tương ứng với dữ liệu đầu vào  $\mathbf{x}_i$ . Hàm mất mát MSE được tính như sau:

$$MSE = \frac{1}{N} \sum_{i=1}^N (y_i - a_i^{L-1})^2 \quad (2.3)$$

hàm CE được tính theo công thức:

$$CE = -\frac{1}{N} \sum_{i=1}^N a_i^{L-1} \log(y_i) \quad (2.4)$$

Để điều chỉnh các trọng số, ta cần phải biết giá trị cập nhật cho mỗi trọng số. Dựa trên giá trị mất mát, các tín hiệu mất mát của nơ-ron trong lớp thứ  $i$  được tính toán dựa trên lỗi mà lớp thứ  $i + 1$  gây ra. Tín hiệu mất mát này được lan truyền từ lớp đầu ra về tới lớp đầu vào, sau đó thực hiện cập nhật các trọng số  $\mathbf{W}, \mathbf{b}$  nhằm giảm các giá trị lỗi. Các giá trị lỗi của các bộ trọng số được gọi là gradient.

Sau khi có bộ gradient, ta có thể cập nhật mạng bằng cách đi ngược lại với hướng của gradient. Giá trị mất mát trên bộ dữ liệu sẽ giảm nếu gradient được cập nhật với bước đủ nhỏ. Bằng cách lặp đi lặp lại quá trình lan truyền tiến, lan truyền ngược và cập nhật trọng số bằng gradient, bộ trọng số có thể hội tụ tại một điểm cực tiểu của hàm mất mát. Phương pháp tối ưu lặp này được gọi là gradient descent (Hình 2.3).

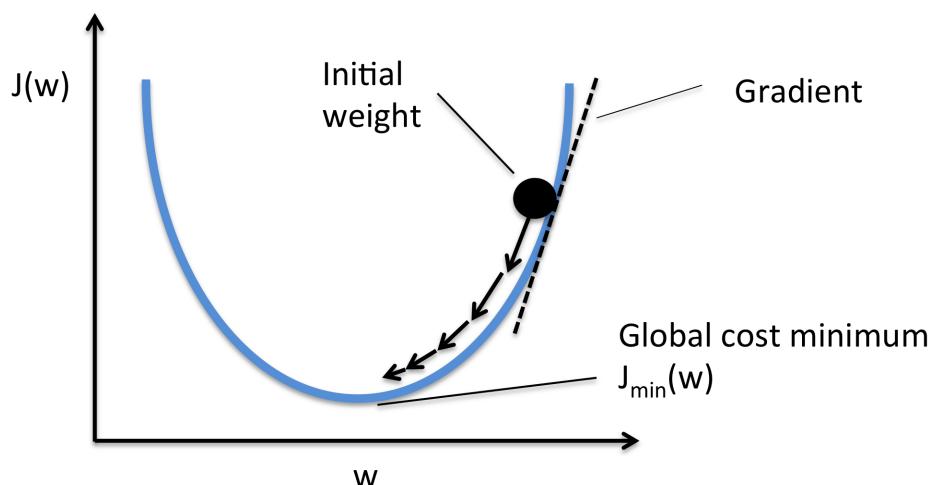


Figure 2.3: Phương pháp gradient descent <sup>3</sup>

Hệ số học (learning rate) được điều chỉnh để kiểm soát bước cập nhật trong gradient descent. Với hệ số học lớn, bước đi của gradient descent trở nên lớn hơn và có thể gặp khó khăn hội tụ khi gần điểm cực tiểu. Hệ số học nhỏ hơn giúp mô hình dễ hội tụ hơn tuy nhiên mất nhiều vòng lặp hơn.

Phương pháp gradient descent truyền thống sử dụng gradient cho cả bộ dữ liệu. Tuy vậy, với các bộ dữ liệu cực lớn, điều này là không thể. Do vậy, ta phải xấp xỉ gradient của cả bộ dữ liệu bằng cách chia nhỏ bộ dữ liệu gốc thành các lô

<sup>3</sup><https://machinelearningnotepad.wordpress.com/2018/04/15/gradient-descent>

nhỏ (mini-batch) và thực hiện việc cập nhật trọng số trên các lô này. Phương pháp xấp xỉ này được gọi là stochastic gradient descent - SGD. Gần đây, nhiều phương pháp biến thể của SGD được phát triển có thể kể đến như RMSprop, Adadelta [49] và Adam [19] giúp tăng tốc độ hội tụ cũng như tăng cường hiệu năng khi huấn luyện mô hình.

## 2.2 Học sâu

Học sâu là một lớp các mô hình học máy được xây dựng dựa trên mạng nơ-ron nhân tạo. Nếu mạng nơ-ron nhân tạo chỉ gồm vài lớp nơ-ron, các mạng học sâu được thiết kế để mở rộng ra hàng chục, trăm, thậm chí hàng nghìn lớp nơ-ron. Điều này giúp các mô hình học sâu giải quyết được nhiều bài toán phức tạp với độ chính xác ngang bằng con người. Huấn luyện mô hình học sâu yêu cầu một lượng dữ liệu lớn và tài nguyên tính toán lớn. Trong thập kỷ vừa qua với sự bùng nổ của dữ liệu và tài nguyên tính toán, nghiên cứu học sâu trở thành tâm điểm của lĩnh vực trí tuệ nhân tạo.

### 2.2.1 Mạng nơ-ron tích chập

Mạng nơ-ron tích chập (Convolutional neural networks - CNN), hay mạng tích chập, được đề xuất lần đầu vào năm 1989 bởi Yan LeCun [21] để giải quyết bài toán nhận dạng chữ viết tay. Mạng tích chập được thiết kế với mục tiêu xử lý dữ liệu dạng bảng - lưới, ví dụ như dữ liệu chuỗi thời gian có thể biểu diễn dưới dạng bảng 1D hay ảnh là dữ liệu 2D của các điểm ảnh. Năm 2012, Alex Krizhevsky xây dựng một mạng tích chập (AlexNet [20]) và tăng tốc quá trình huấn luyện sử dụng GPU. Mô hình đề xuất của Krizhevsky đứng đầu trong bảng xếp hạng trong cuộc thi phân loại ảnh ImageNet [11] với độ chính xác vượt hơn 10% các đội tham gia. Hiện nay, mạng tích chập được áp dụng xử lý nhiều bài toán phức tạp trong xử lý ảnh, xử lý ngôn ngữ tự nhiên, xử lý tín hiệu âm thanh, ...

Thời điểm hiện tại đã có rất nhiều kiến trúc mạng nơ-ron tích chập khác nhau được xây dựng, tuy nhiên chúng đều được cấu thành từ các lớp thành phần (Hình 2.4), bao gồm: lớp tích chập (convolutional layer), lớp tổng hợp (pooling layer) và lớp kết nối đầy đủ (fully connected layer).

---

<sup>4</sup><https://www.upgrad.com/blog/basic-cnn-architecture>

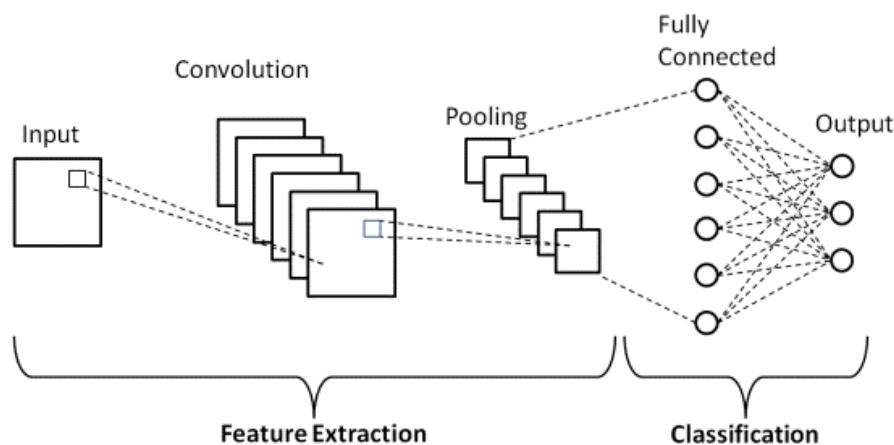


Figure 2.4: Các thành phần cơ bản của mạng tích chập<sup>4</sup>

### Lớp tích chập

Lớp tích chập là lớp được sử dụng nhiều nhất trong mạng nơ-ron tích chập, mục tiêu của lớp tích chập là học các đặc trưng cục bộ từ dữ liệu đầu vào (Hình 2.5). Cái tên lớp tích chập xuất phát từ việc lớp sử dụng phép biến đổi toán học tuyến tính gọi là tích chập (convolution).

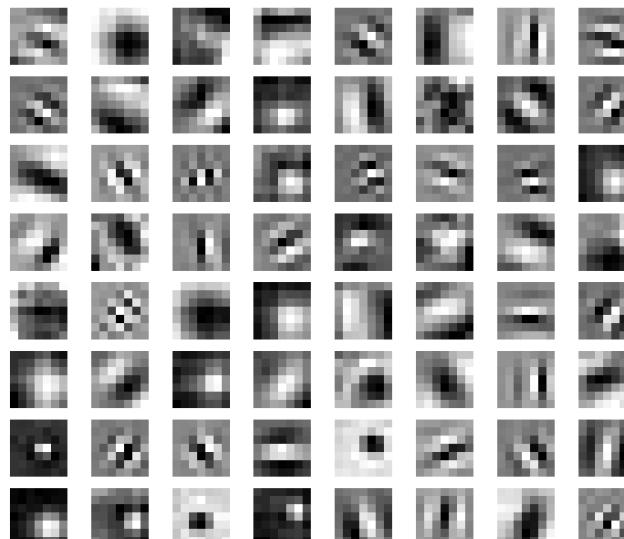


Figure 2.5: Các đặc trưng học được trong lớp tích chập<sup>5</sup>

Ban đầu, phép tích chập được được sử dụng phổ biến trong xử lý tín hiệu. Về sau nguyên lý biến đổi trong tích chập mới được áp dụng trong lĩnh vực xử lý hình ảnh. Công thức tích chập cho 2 hàm  $x$  và  $w$  theo chiều thời gian  $t$  được

<sup>5</sup><https://debuggercafe.com/visualizing-filters-and-feature-maps-in-convolutional-neural-networks-using-pytorch>

tính như sau:

$$(x * w)(t) = \sum x(\alpha)w(t - \alpha) \quad (2.5)$$

trong đó  $(x * w)$  là ký hiệu tích chập. Trong xử lý ảnh, phép tích chập được mở rộng ra 2D hoặc 3D. Trong lớp tích chập,  $x$  được gọi là đầu vào của lớp còn  $w$  được gọi là bộ lọc (filter) (Hình 2.6). Phép toán tích chập dựa trên nơ-ron của lớp trước để tính toán ra giá trị nơ-ron của lớp ngay sau.

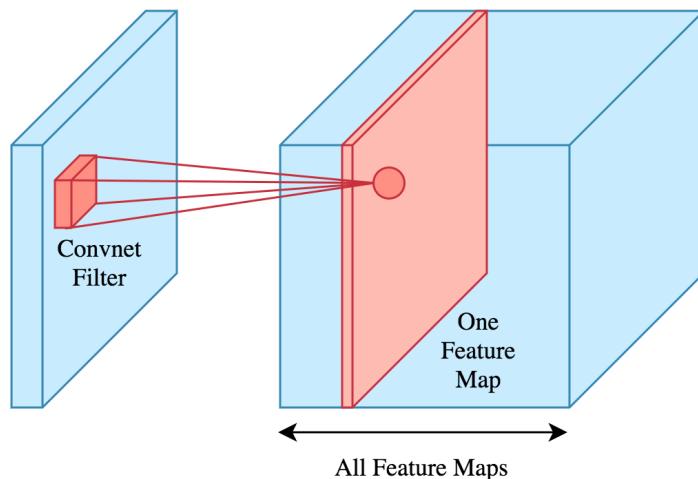


Figure 2.6: Một ví dụ của lớp tích chập <sup>6</sup>

Bộ lọc là một bộ trọng số học được có kích thước nhỏ (ví dụ  $5 \times 5 \times 3$ , dài rộng 5, chiều sâu 3 của khối đầu vào phía trước). Trong quá trình lan truyền tiến, bộ lọc được dịch chuyển trên cả chiều dài và chiều rộng của khối đầu vào để tổng hợp thông tin thành một khối đặc trưng 2D. Cùng với kích thước bộ lọc, lớp tích chập còn có 2 tham số bước nhảy (stride) và đệm (padding). Tham số bước nhảy quy định độ dịch chuyển của bộ lọc, còn tham số đệm là số lượng giá trị 0 được thêm vào xung quanh khối đầu vào để kiểm soát kích thước đầu ra của lớp tích chập.

Có nhiều bộ lọc trong một lớp tích chập để học nhiều dạng thông tin đặc trưng khác nhau như cạnh thẳng, cạnh chéo, đốm màu hay các đặc trưng phức tạp hơn ở các lớp sâu hơn. Các khối đặc trưng 2D của nhiều bộ lọc được xếp chồng lên nhau thành một khối đặc trưng 3D (Hình 2.6).

Điểm khác biệt chính giữa lớp tích chập và lớp ẩn thông thường là tính cục bộ. Mỗi nơ-ron trong lớp tích chập chỉ được kết nối tới một số điểm nhất định trong lớp phía trước trong khi một nơ-ron trong lớp ẩn thông thường được kết nối tới tất cả nơ-ron ở lớp trước.

<sup>6</sup><https://towardsdatascience.com/convolution-neural-networks-a-beginners-guide-implementing-a-mnist-hand-written-digit-8aa60330d022>

Mạng tích chập giả định đặc trưng học được ở một vùng ảnh cũng có thể phát hiện đặc trưng tương tự tại một vùng khác trong ảnh. Trong quá trình tính toán tích chập, bộ trọng số của bộ lọc được sử dụng đi sử dụng lại tại các vùng khác nhau của ảnh. Điều này giúp cho tổng số trọng số của lớp tích chập ít hơn nhiều so với một lớp ẩn trong mạng nơ-ron thông thường mà vẫn học được nhiều đặc trưng ý nghĩa. Trong quá trình lan truyền ngược, gradient của bộ lọc được tổng hợp từ nhiều vùng khác nhau.

### Lớp tổng hợp

Lớp tổng hợp (pooling layer) có chức năng tổng hợp thông tin từ lớp trước, giảm số nơ-ron và trọng số trong mạng từ đó giảm chi phí tính toán. Ngoài ra lớp tổng hợp còn giúp biểu diễn qua các lớp nhất quán hơn với sự thay đổi nhỏ trong đầu vào từ đó giúp mô hình hiệu quả hơn với dữ liệu mới. Cách thức hoạt động của lớp tổng hợp tương đối đơn giản, nó dùng một cửa sổ (thường có kích thước  $2 \times 2$ ) để trượt trên đầu vào. Với mỗi điểm trượt, lớp tổng hợp chọn giá trị lớn nhất trong vùng (tổng hợp cực đại - max pooling) hoặc lấy trung bình của các giá trị (tổng hợp trung bình - average pooling) làm giá trị cho ma trận đầu ra (Hình 2.7).

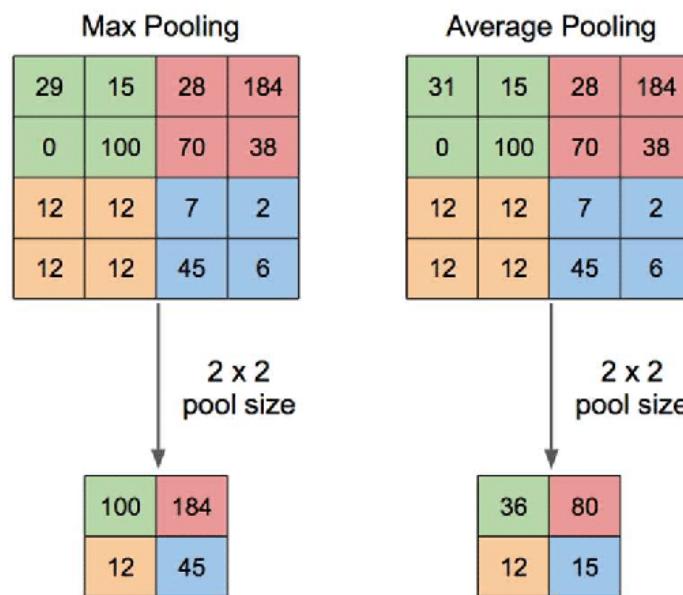


Figure 2.7: Một ví dụ của tổng hợp cực đại và tổng hợp trung bình [48]

### Lớp kết nối đầy đủ

Trong mạng nơ-ron nhân tạo thông thường, mọi lớp trừ lớp đầu vào là lớp kết nối đầy đủ, nghĩa là nơ-ron ở lớp sau được tính toán dựa trên tất cả nơ-ron của

lớp trước. Thông thường trong mạng nơ-ron tích chập, lớp kết nối đầy đủ nằm ở cuối mạng, đóng vai trò là bộ phân loại của mạng. Đầu ra của lớp tích chập hay lớp tổng hợp thường là ma trận 2D hoặc 3D, do vậy cần phải được duỗi thẳng thành một vec-tơ để làm đầu vào cho lớp kết nối đầy đủ.

### 2.2.2 Mạng nơ-ron residual

Các mạng học sâu khi mở rộng càng nhiều lớp càng có khả năng xấp xỉ các hàm phức tạp và giải quyết bài toán phức tạp hơn. Tuy nhiên, mạng rất sâu lại gặp phải vấn đề vanishing/exploding gradients, hiện tượng mà gradient tiến tới 0 hoặc vô hạn trong quá trình lan truyền ngược trong các lớp đầu khiến cho mô hình không được cải thiện.

Mạng nơ-ron residual (residual neural network - ResNet) [39] được ra đời để giải quyết vấn đề vanishing/exploding gradients. Để luồng thông tin tới được các lớp đầu của mạng trong quá trình lan truyền ngược, ResNet sử dụng một cấu trúc đặc biệt gọi là skip connection (Hình 2.8). Skip connection khác với kết nối thông thường ở chỗ nó không kết nối 2 lớp liên tiếp mà kết nối 2 lớp không liền kề nhau. Khi lan truyền ngược, các kết nối này đưa gradient trực tiếp về các lớp phía đầu của mạng và tránh khỏi hiện tượng vanishing/exploding gradients.

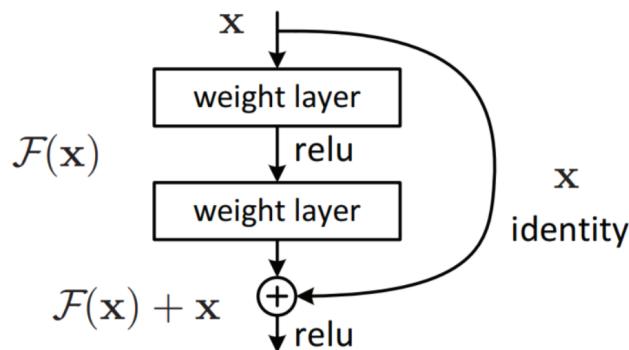


Figure 2.8: Skip connection trong ResNet [39]

ResNet có nhiều kiến trúc với nhiều độ sâu khác nhau, hình 2.9 mô tả các kiến trúc này. Với ResNet-18 và ResNet-34, các khối residual được xây dựng từ các lớp tích chập với bộ lọc kích thước  $3 \times 3$ . Các mô hình nhiều lớp hơn như ResNet-50, ResNet-101, ResNet-152 sử dụng khối nút thắt cổ chai (bottleneck) để giảm chi phí tính toán. Hình 2.10 mô tả kiến trúc hai khối này.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112			7×7, 64, stride 2		
				3×3 max pool, stride 2		
conv2_x	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1			average pool, 1000-d fc, softmax		
FLOPs		$1.8 \times 10^9$	$3.6 \times 10^9$	$3.8 \times 10^9$	$7.6 \times 10^9$	$11.3 \times 10^9$

Figure 2.9: Các kiến trúc khác nhau của ResNet [39]

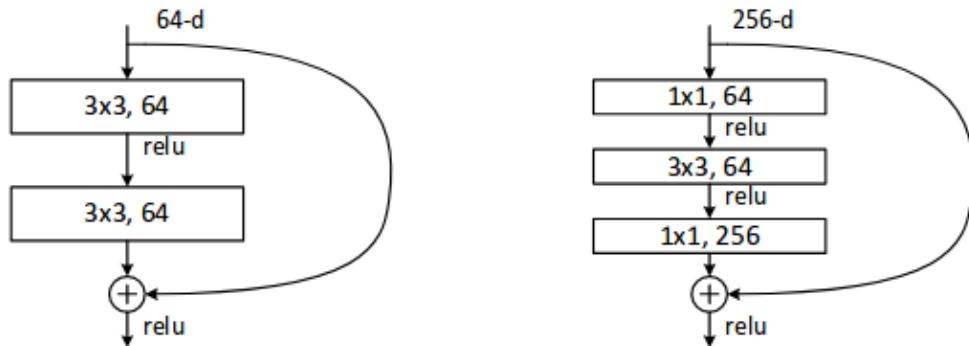


Figure 2.10: Kiến trúc hai khối residual trong các kiến trúc mạng ResNet [39]

## 2.3 Hàm măt mát

Như đã trình bày trong 2.1.2, trong học máy, hàm măt mát đo đặc sự sai khác của đầu ra trong nhãn đầu vào. Điều tương tự cũng áp dụng cho học sâu. Trong học sâu, hàm măt mát có thể chia thành hai nhóm: hàm măt mát phân loại và hàm măt mát phân biệt. Hai hàm măt mát đơn giản và được sử dụng rộng rãi nhất cho 2 nhóm là hàm softmax và hàm triplet.

### Hàm măt mát softmax

Về cơ bản, hàm măt mát softmax là một hàm măt mát phân loại đa lớp gồm hàm kích hoạt softmax kết hợp hàm entropy chéo như mô tả trong công thức 2.4. Thông thường trong một mạng học sâu phân loại, lớp đầu ra có số nơ-ron tương ứng với số lớp cần được phân loại (ví dụ mô hình phân loại 6 loại xe có 6 nơ-ron ở lớp đầu ra). Giá trị đại diện cho mỗi lớp ở tầng đầu ra là cái nhận

được khi sử dụng mạng dự đoán cho một ví dụ. Hàm softmax đưa vec-tơ đầu ra về khoảng  $(0, 1)$  và tổng của chúng đúng bằng 1. Bởi vì giá trị của một nơ-ron đại diện cho một lớp, có thể xem đó là xác suất dự đoán của lớp đó. Hàm mất softmax cho một ví dụ có vec-tơ biểu diễn  $\mathbf{z}$  thuộc lớp  $c$  được tính bằng công thức:

$$\mathcal{L}_{\text{softmax}}(\mathbf{z}, c) = -\log\left(\frac{e^{\mathbf{z}_c}}{\sum_j e^{\mathbf{z}_j}}\right) \quad (2.6)$$

### Hàm mất mát triplet

Khác với hàm mất mát phân loại, mục tiêu của hàm mất mát phân biệt là học điểm khác biệt giữa những vật thể hay ví dụ khác nhau. Các hàm phân biệt như triplet giúp mạng học không gian biểu diễn mà trong đó các ví dụ giống nhau sẽ gần nhau và những ví dụ khác nhau nằm xa nhau.

Đầu vào của hàm triplet bao gồm 3 điểm trong không gian biểu diễn. Trong đó có 2 điểm dữ liệu và một điểm có nhãn khác 2 điểm còn lại. Trong mỗi ba điểm, có một điểm được gọi là neo (anchor) kí hiệu A, một điểm dương (positive) kí hiệu P có cùng nhãn với A, và một điểm âm (negative) kí hiệu N khác nhãn với A. Mục tiêu của hàm triplet là kéo điểm P gần hơn tới A trong không gian biểu diễn và đẩy P ra khỏi A sao cho khoảng cách  $A - P$  nhỏ hơn khoảng cách  $A - N$  một khoảng  $m$  gọi là lề (margin) (Hình 2.11). Với  $m$  là tham số lề,  $\mathbf{z}_i, \mathbf{z}_j, \mathbf{z}_k$  lần lượt là vec-tơ A, P, N, giá trị hàm triplet được tính như sau:

$$\mathcal{L}_{\text{triplet}}(\mathbf{z}_i, \mathbf{z}_j, \mathbf{z}_k) = \max\left(0, \|\mathbf{z}_i - \mathbf{z}_j\|_2^2 - \|\mathbf{z}_i - \mathbf{z}_k\|_2^2 + m\right) \quad (2.7)$$

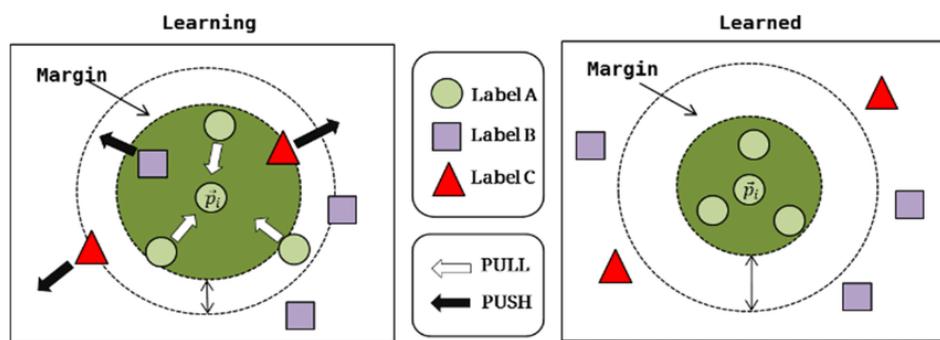


Figure 2.11: Ví dụ hàm mất mát triplet [26]

## 2.4 Trích xuất thông tin

Xử lý tín hiệu đóng vai trò quan trọng trong bất kì hệ thống tiếng nói nào, từ nhận dạng tiếng nói tới nhận dạng người nói. Hai trong những đặc trưng âm thanh phổ biến nhất được sử dụng là filter banks và Mel-Frequency Cepstral Coefficients (MFCCs).

Trích xuất filter banks và MFCCs tuân theo quy trình khá tương tự nhau, tính toán filter banks yêu cầu ít hơn MFCCs một số bước. Quy trình trích xuất MFCCs gồm 6 bước chính (Hình 2.12): nhấn mạnh (pre-emphasis), cắt khung (framing), cửa sổ (windowing), biến đổi Fourier (Fourier transform), filter bank và cuối cùng là biến đổi cô sin rời rạc (Discrete cosine transform) để có kết quả là MFCC.

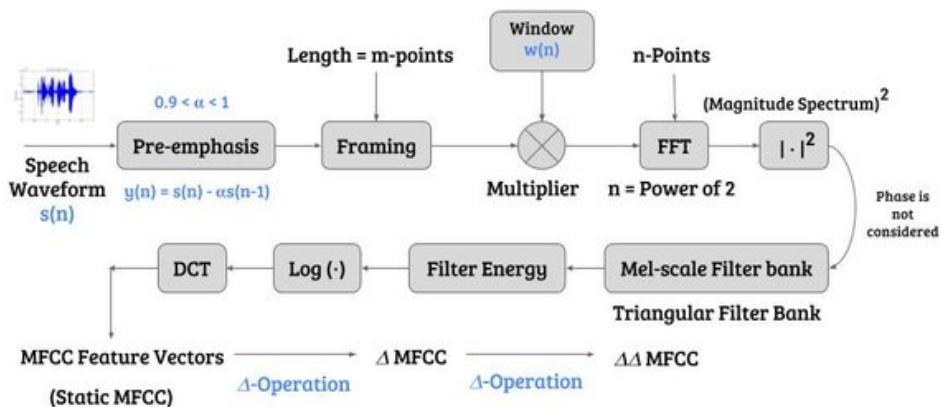


Figure 2.12: Thuật toán trích xuất MFCCs [5]

## Chapter 3

# Nhận dạng người nói tiếng Việt

Trong chương này, tác giả trình bày phương pháp học sâu để giải quyết bài toán nhận diện người nói. Ngoài ra tác giả cũng đề xuất phương pháp để cải tiến mô hình hiện tại để đạt được kết quả cao hơn trên tiếng Việt.

### 3.1 Nghiên cứu liên quan

Các mô hình học sâu hiện đại giải quyết bài toán nhận diện giọng nói thường gồm ba phần chính (Hình 3.1):

- Một mạng nơ-ron làm công cụ trích xuất biểu diễn người nói cho một khung đặc trưng tiếng nói.
- Lớp tổng hợp dữ liệu: lớp này sử dụng biểu diễn của các khung âm thanh từ mạng nơ-ron để tổng hợp ra một vec-tơ duy nhất đại diện cho đoạn tiếng nói đầu vào.
- Một hàm mất mát để tối ưu toàn bộ mô hình.

Trong pha kiểm tra, mạng nơ-ron và lớp tổng hợp dữ liệu được sử dụng để tạo ra vec-tơ biểu diễn của đoạn tiếng nói kiểm tra. Hàm mất mát chỉ được sử dụng trong pha phát triển, và bị loại bỏ trong pha kiểm tra và pha ghi danh.

Phần lớn các nghiên cứu về nhận dạng người nói hướng tới cải thiện hệ thống qua nâng cao hiệu quả của lớp tổng hợp dữ liệu và hàm mất mát. Về mạng nơ-ron trích xuất đặc trưng, các nghiên cứu chủ yếu sử dụng các mạng xương sống thành công trong các bài toán khác như phân loại hình ảnh (Ví dụ: mạng VGG [34] và mạng ResNet [14]) và nhận dạng tiếng nói (mạng TDNN [41] và mạng LSTM [16]).

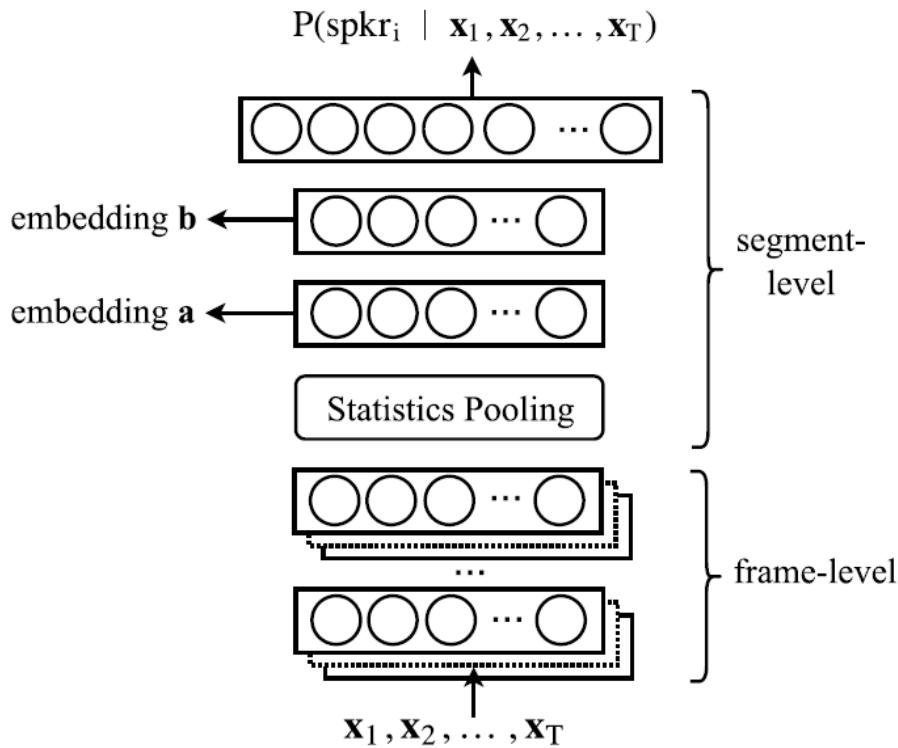


Figure 3.1: Tổng quan mô hình nhận diện người nói [37]

Một vài nghiên cứu nổi bật cho lớp tổng hợp có thể kể đến như: tổng hợp trung bình (average pooling) [39], tổng hợp thống kê (statistical pooling) [37], tổng hợp dựa trên cơ chế tập trung (attentive pooling) [27, 51], hay mã hoá dựa trên từ điển NetVLAD/GhostVLAD trong [46].

Các nghiên cứu tiên phong [24, 35, 37, 39] ứng dụng mạng nơ-ron nhận tạo vào bài toán nhận dạng người nói học không gian biểu diễn bằng các hàm mất mát phân loại (classification loss). Trên cơ sở đó, các nghiên cứu thịnh hành [27, 30, 36] sử dụng hàm softmax để huấn luyện mô hình. Hàm softmax có khả năng học để phân loại người nói một cách hiệu quả, tuy nhiên, nó không được thiết kế để tối ưu tính tương đồng trong không gian embedding.

Để giải quyết điểm yếu của softmax, công trình của Liu và cộng sự [22] đề xuất hàm angular softmax (A-Softmax) cho bài toán nhận diện khuôn mặt bằng cách dùng độ tương đồng cô-sin làm đầu vào của hàm softmax. Các nghiên cứu sau đó áp dụng hàm A-Softmax trên bài toán nhận dạng người nói [38, 40] cho thấy sự vượt trội của hàm này so với softmax thông thường. Các phiên bản cải tiến của hàm A-Softmax như AM-Softmax [42] và AAM-Softmax [12] sử dụng hàm phạt biến trở nên phổ biến cho bài toán nhận dạng người nói do dễ dàng cài đặt và hiệu năng cao. Tuy nhiên, huấn luyện mô hình với AM-Softmax và AAM-Softmax khá khó khăn do hai hàm này rất nhạy cảm với sự điều chỉnh

tham số.

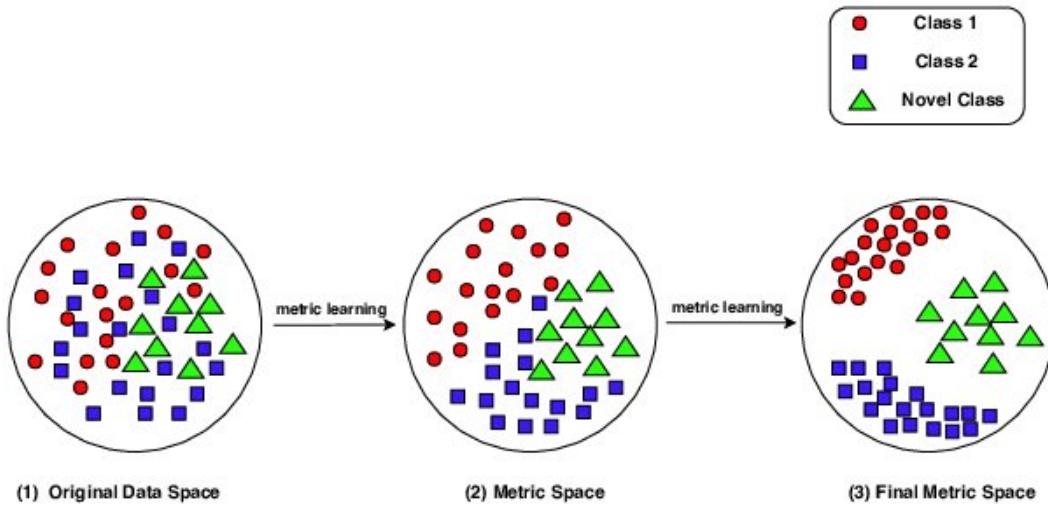


Figure 3.2: Không gian embedding với học biểu diễn[44]

Bên cạnh nhóm hàm mục tiêu phân loại, học biểu diễn (Representation learning hay Metric learning) cũng là lựa chọn phổ biến cho bài toán nhận dạng người nói. Thay vì phân loại, mục tiêu của học biểu diễn là tối ưu không gian embedding 3.2. Trong bài toán nhận dạng người nói, biểu diễn của các câu nói từ cùng một người được kéo lại gần nhau trong không gian embedding, ngược lại câu của những người khác nhau bị đẩy ra xa.

Hai hàm học biểu diễn nổi tiếng là contrastive loss [8] và triplet loss [32] xuất phát từ bài toán nhận diện khuôn mặt cho kết quả hứa hẹn trong bài toán nhận diện người nói [29, 50]. Năm 2020, Chung và cộng sự [9] đề xuất hàm Angular Prototypical cho kết quả vượt trội khi so với các hàm mục tiêu phân loại.

## 3.2 Mô hình cơ sở

Mô hình tác giả sử dụng trong đồ án tuân theo hệ thống ba pha được mô tả như trong 3.1, đây là sự kết hợp giữa mô hình ResNet-34, lớp tổng hợp thống kê trung (attentive statistic pooling - ASP) và hàm mất mát Angular Prototypical (AP). Kiến trúc của hệ thống được miêu tả trong Hình 3.3.

Trong các mục tiếp theo, tác giả sẽ trình bày chi tiết thành phần của hệ thống.

### 3.2.1 Biểu diễn khung âm thanh bằng mạng ResNet

Mạng ResNet sử dụng trong đồ án là biến thể của mạng ResNet-34 như mô tả trong hình 2.9. Tại mỗi lớp, mạng sử dụng một nửa số bộ lọc so với mạng

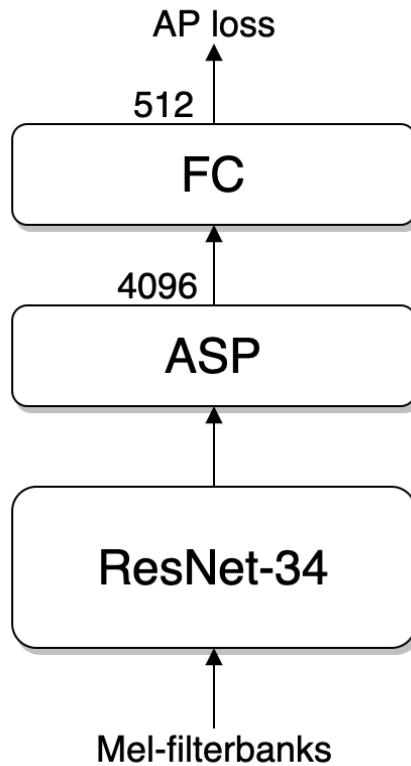


Figure 3.3: Tổng quan mô hình cơ sở sử dụng trong đồ án

ResNet-34 gốc trong các khối ResNet và chứa tổng cộng 8.0 triệu trọng số. Trong lớp tích chập đầu tiên, tham số bước nhảy được chỉnh thành 1 so với 2 trong mạng ResNet-34 gốc khiến đầu vào của các lớp đằng sau lớn hơn từ đó tăng khối lượng tính toán. Chi tiết cấu trúc của mạng được mô tả trong Bảng 3.1.

Table 3.1: placeholder

Layer	Kernel size	Stride	Output shape
Conv1	$3 \times 3 \times 32$	$1 \times 1$	$L \times 64 \times 32$
ResBlock1	$3 \times 3 \times 32$	$1 \times 1$	$L \times 64 \times 32$
ResBlock2	$3 \times 3 \times 64$	$2 \times 2$	$L/8 \times 32 \times 64$
ResBlock3	$3 \times 3 \times 128$	$2 \times 2$	$L/8 \times 16 \times 128$
ResBlock4	$3 \times 3 \times 256$	$2 \times 2$	$L/8 \times 8 \times 256$
Flatten	-	-	$L/8 \times 2048$

### 3.2.2 Tổng hợp thống kê tập trung

Trong thực tế, không phải khung âm thanh nào cũng chứa nhiều thông tin của người nói do độ dài một khung rất ngắn thông thường chỉ 25 mili giây. Ví dụ, một khung có thể chứa nhiều tiếng ồn, hoặc không hề chứa giọng nói. Do vậy, hệ thống sử dụng tổng hợp thống kê tập trung để đánh trọng số cho biểu diễn của các khung với mong muốn tăng thông tin của những khung nhiều ý nghĩa và giảm thông tin của những khung ít ý nghĩa. Từ đó có thể phân biệt giọng

nói một cách hiệu quả hơn.

ASP nhận đầu vào là tập các vec-tơ biểu diễn khung  $\mathbf{h}_t$  ( $t = 1, \dots, T$ ). Vec-tơ biểu diễn của toàn đoạn âm thanh được tính qua 2 bước: đánh trọng số cho từng khung bằng cơ chế tập trung và tổng hợp thông tin thống kê dựa trên trọng số tính được (Hình 3.4).

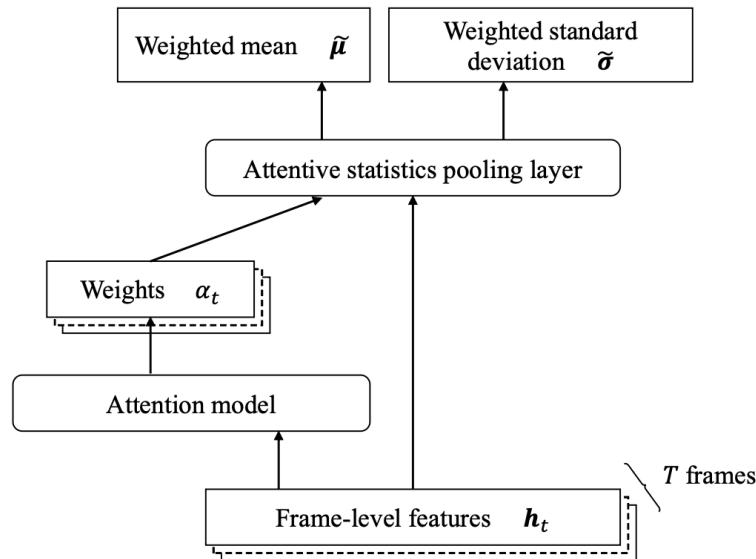


Figure 3.4: Tổng hợp thống kê tập trung [27]

### Cơ chế tập trung

Bằng cơ chế tập trung, trọng số của từng khung có thể được tính như sau:

$$e_t = \mathbf{v}^T f(\mathbf{W}\mathbf{h}_t + b) + k \quad (3.1)$$

$$\alpha_t = \frac{\alpha_t}{\sum_{\rho}^T \alpha_{\rho}} \quad (3.2)$$

Trong công thức 3.1, với  $\mathbf{v}, \mathbf{W}$  là các ma trận trọng số có thể học được,  $f$  là hàm phi tuyến như thanh hoặc ReLu, ta tính được điểm cho mỗi khung  $e_t$ . Sau đó, điểm của mỗi khung được chuẩn hoá trên tất cả các khung để thu được trọng số tập trung bằng hàm softmax như trong 3.2.

### Tổng hợp thống kê

Sau khi có được trọng số của các khung, ta tính vec-tơ trung bình có trọng số:

$$\boldsymbol{\mu} = \sum_t^T \alpha_t \mathbf{h}_t \quad (3.3)$$

Bằng cách tính này, vec-tơ biểu diễn của đoạn âm thanh tập trung hơn vào những khung tiếng nói có ý nghĩa cao. Ngoài ra, các trọng số tập trung còn được sử dụng để tính độ lệch chuẩn có trọng số:

$$\sigma = \sqrt{\sum_t^T \alpha_t \mathbf{h}_t \odot \mathbf{h}_t - \boldsymbol{\mu} \odot \boldsymbol{\mu}} \quad (3.4)$$

Với  $\odot$  là phép nhân Hadamard,  $\boldsymbol{\mu}$  là vec-tơ trung bình có trọng số tính trong công thức 3.3. Sau khi hoàn tất quá trình tính toán vec-tơ trung bình và độ lệch chuẩn có trọng số,  $\boldsymbol{\mu}$  và  $\sigma$  được ghép lại để biểu diễn cho một đoạn tiếng nói. Bằng cách này, mọi đoạn âm thanh dài ngắn đều có vec-tơ biểu diễn với số chiều như nhau, được tổng hợp từ những khung âm thanh có ý nghĩa nhất trong câu.

#### 3.2.3 Hàm mất mát Angular Prototypical

Trong thực tế, ta cần tổng hợp từ một số câu nói nhất định để tạo vec-tơ biểu diễn người nói. Do vậy, Chung và cộng sự [9] đề xuất hàm mất mát AP tối ưu không gian biểu diễn dựa trên nguyên mẫu (prototype) của người nói. Mỗi người nói có một nguyên mẫu và một câu truy vấn, mục tiêu của AP là đẩy xa truy vấn của một người ra xa nguyên mẫu của những người khác và kéo nó lại gần nguyên mẫu của người đó (Hình 3.5).

Xét một mini-batch gồm  $M$  đoạn tiếng nói từ mỗi  $N$  người nói, gọi  $\mathbf{x}_{i,j}$  là vec-tơ biểu diễn của đoạn tiếng nói thứ  $j$  của người thứ  $i$ ,  $1 \leq i \leq N, 1 \leq j \leq M$ . Giả sử truy vấn của một người là câu cuối cùng của người đó  $\mathbf{x}_{i,M}$ , nguyên mẫu của một người nói được tính toán như sau:

$$\mathbf{c}_i = \frac{1}{M-1} \sum_{m=1}^{M-1} \mathbf{x}_{i,m} \quad (3.5)$$

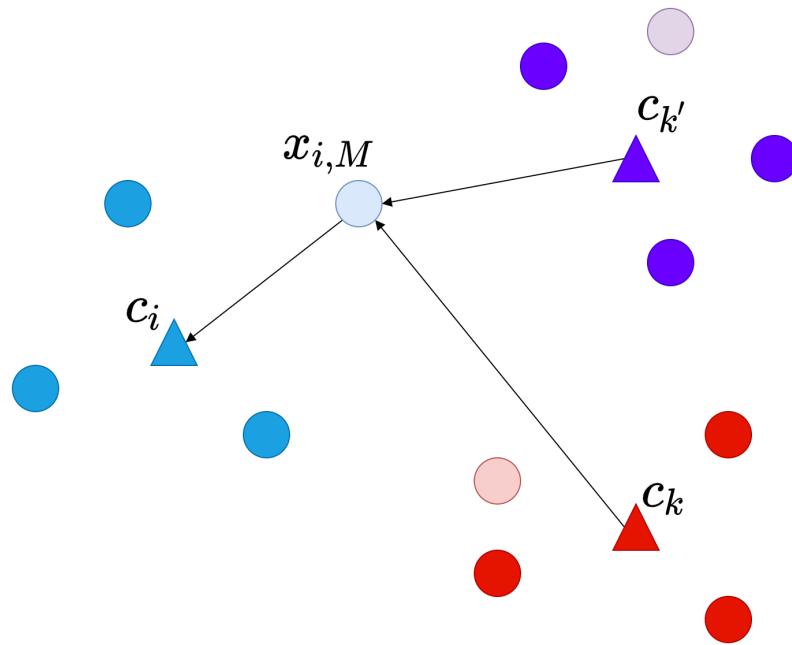


Figure 3.5: Hàm mất mát Angular Prototypical

Trong AP, độ tương đồng cô-sin được sử dụng để làm độ đo. Độ tương đồng được tính theo công thức 3.3.3 với hệ số scale  $w$  và bias  $b$ . Hai hệ số này giúp mô hình hội tụ ổn định hơn và khai quát hóa tốt hơn với thay đổi trong đặc trưng đầu vào [43].

$$S_{i,k} = w \cdot \cos(\mathbf{x}_{i,M}, \mathbf{c}_k) + b \quad (3.6)$$

Trong quá trình huấn luyện, câu truy vấn của mỗi người được phân loại dựa trên độ tương đồng đối với  $N$  nguyên mẫu trong mini-batch:

$$L_{AP} = -\frac{1}{N} \sum_{i=1}^N \log \frac{e^{S_{i,i}}}{\sum_{k=1}^N e^{S_{i,k}}} \quad (3.7)$$

Trong công thức 3.7,  $S_{i,i}$  là độ tương đồng của truy vấn người  $i$  và nguyên mẫu của chính người đó. Bằng việc sử dụng hàm softmax,  $S_{i,i}$  được đẩy gần hơn tới 1 và mô hình bị "phạt" nặng hơn nếu độ tương đồng của truy vấn người  $i$  tới nguyên mẫu của người khác lớn.

### 3.3 Đề xuất cải tiến mô hình

Trong phần này, đề xuất ....

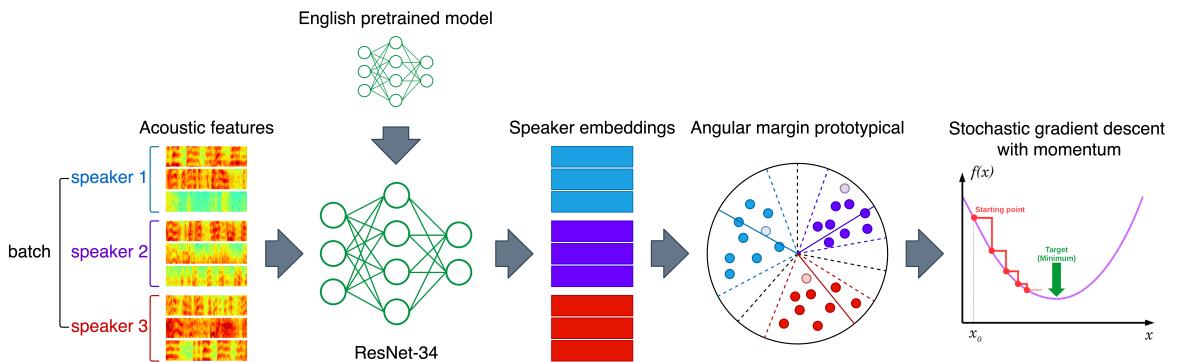


Figure 3.6: Sơ đồ mô tả transfer learning sử dụng kiến thức hiện có cho các tác vụ mới

### 3.3.1 Transfer learning

Transfer learning là một kỹ thuật trong học máy khi mà một mô hình được huấn luyện cho một tác vụ nhất định được sử dụng làm điểm bắt đầu cho một tác vụ khác. Transfer learning cho phép rút ngắn quá trình huấn luyện và gia tăng hiệu năng cho quá trình huấn luyện mô hình trên tác vụ mới với ít dữ liệu hơn đáng kể.

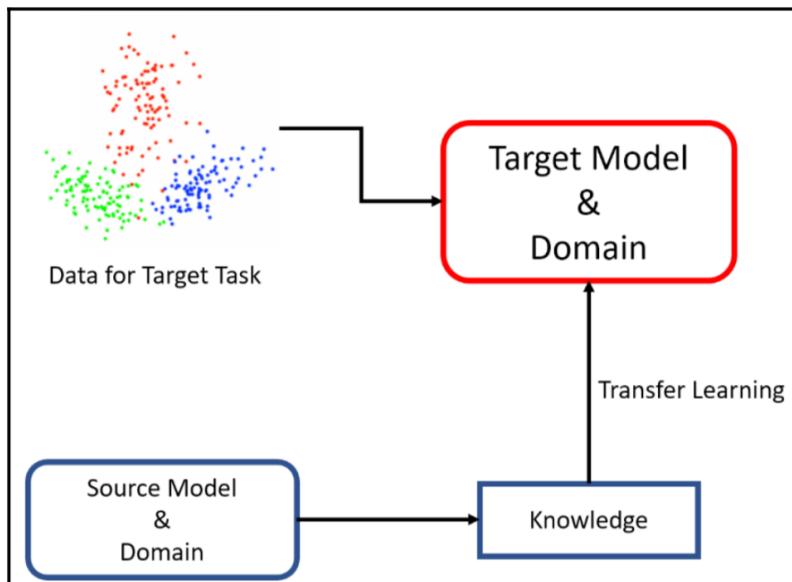


Figure 3.7: Sơ đồ mô tả transfer learning sử dụng kiến thức hiện có cho các tác vụ mới [31]

Cụ thể hơn, transfer learning hỗ trợ việc huấn luyện tác vụ mục tiêu theo những cách sau:

- Hiệu suất cơ sở tốt hơn (higher start): khi ta tăng cường kiến thức của mô hình mới với kiến thức từ mô hình gốc, hiệu suất cơ sở có thể cải thiện nhờ việc chuyển giao kiến thức.
- Thời gian huấn luyện ngắn hơn (higher slope): tốc độ hội tụ của mô hình

mới có thể nhanh hơn dẫn tới thời gian huấn luyện ngắn hơn.

- Kết quả cuối cùng tốt hơn (higher asymptote): hiệu suất cuối cùng cao hơn có thể đạt được bằng việc sử dụng transfer learning.

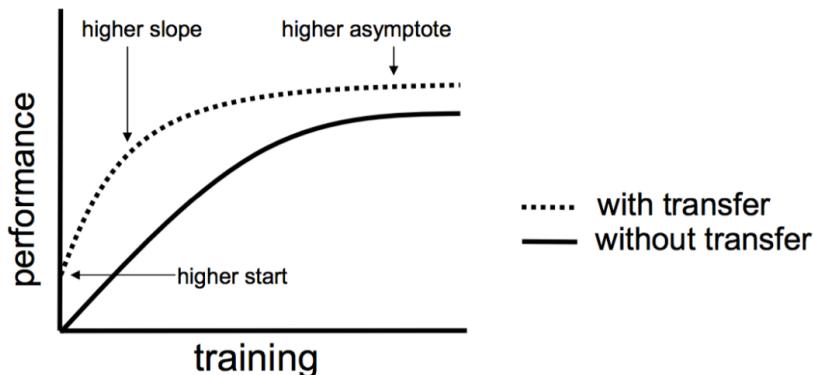


Figure 3.8: Lợi ích của transfer learning đối với việc huấn luyện mô hình [4]

Huấn luyện mô hình học sâu cần một lượng lớn tài nguyên tính toán và dữ liệu, do đó transfer learning được sử dụng rộng rãi trong cộng đồng nghiên cứu học sâu cho bài toán thị giác máy tính hay xử lý ngôn ngữ tự nhiên. Các thư viện học sâu nổi tiếng như Pytorch hay Tensorflow đều công bố các mô hình huấn luyện sẵn trên hàng triệu hay chục triệu ảnh giúp cộng đồng phát triển các mô hình học sâu dễ dàng và hiệu quả hơn với ít dữ liệu. Gần đây, cộng đồng nghiên cứu xử lý ngôn ngữ tự nhiên cũng có thể lợi dụng các mô hình huấn luyện sẵn cực lớn với hàng trăm tỉ trọng số như BERT [13], GPT-2 [28] hay GPT-3 [6] để cải tiến các tác vụ hạ lưu như phân loại nhận dạng thực thể, phân tích cú pháp phụ thuộc, tóm tắt văn bản, ...

Trong nhận dạng người nói, transfer learning cũng được áp dụng **TODO transfer learning in speaker recognition**. Trong đồ án, phương pháp transfer learning từ mô hình huấn luyện trên người nói tiếng Anh được áp dụng để cải thiện mô hình giọng nói tiếng Việt.

### 3.3.2 SGD với momentum khái quát hoá tốt hơn Adam

Adam [19] là một thuật toán tối ưu thích nghi hệ số học. Được công bố vào năm 2014, Adam được trình bày tại một hội nghị rất uy tín cho cộng đồng học sâu - ICLR 2015. Bài báo phát triển một thuật toán rất hứa hẹn, cho thấy sự hội tụ vượt trội so với các thuật toán hiện hành, dẫn đến tăng tốc trong quá trình huấn luyện.

Adam thích nghi hệ số học cho một trọng số của mạng sử dụng giá trị trung bình

trượt của gradient và gradient bình phương của trọng số đó qua các mini-batch. Cho các trọng số  $w^{(t)}$  và hàm mất mát  $L^{(t)}$ , trong đó  $t$  là chỉ số vòng lặp trong quá trình huấn luyện, việc cập nhật trọng số trong Adam như sau:

$$\begin{aligned} m_w^{(t+1)} &\leftarrow \beta_1 m_w^{(t)} + (1 - \beta_1) \nabla_w L^{(t)} \\ v_w^{(t+1)} &\leftarrow \beta_2 v_w^{(t)} + (1 - \beta_2) (\nabla_w L^{(t)})^2 \\ \hat{m}_w &= \frac{m_w^{(t+1)}}{1 - \beta_1^{t+1}} \\ \hat{v}_w &= \frac{v_w^{(t+1)}}{1 - \beta_2^{t+1}} \\ w^{(t+1)} &\leftarrow w^{(t)} - \eta \frac{\hat{m}_w}{\sqrt{\hat{v}_w} + \epsilon} \end{aligned} \quad (3.8)$$

trong đó  $\epsilon$  là một hệ số cực bé (ví dụ  $10^{-6}$ ) để tránh phép chia cho 0,  $\beta_1$  và  $\beta_2$  là hệ số quên cho giá trị trung bình trượt bậc 1 và bậc 2 tương ứng. Phương pháp tối ưu đã được sử dụng trong nhiều ứng dụng do hiệu suất cạnh tranh và khả năng hoạt động tốt mà không cần điều chỉnh các hệ số. Công thức 3.8 cho thấy kích thước bước nhảy của quy tắc cập nhật trong Adam bất biến với độ lớn của gradient.

Tuy nhiên sau một thời gian, cộng đồng bắt đầu nhận thấy trong một số trường hợp, Adam huấn luyện mô hình tệ hơn so với phương pháp tối ưu truyền thống SGD. Bằng thực nghiệm, nghiên cứu [18] cho thấy Adam dù trong những vòng lặp đầu vượt trội so với SGD ở những vòng lặp đầu nhưng nhanh chóng trì trệ trong những vòng lặp sau. Wilson cùng cộng sự trong [45] chỉ ra rằng các phương pháp thích ứng như Adam hay Adadelta không khai quát hoá SGD với momentum sau khi thử nghiệm trên một loạt các tác vụ, không khuyến khích cộng đồng sử dụng các thuật toán thích ứng. Nhìn chung, các nghiên cứu cho thấy rằng tính tổng quát hoá của Adam tệ hơn SGD.

Trong bài toán nhận diện người nói tập mở, tính tổng quát hoá của mô hình là đặc biệt quan trọng. Lý do chính là bởi vì môi trường trong pha kiểm thử khác nhau rất nhiều so với môi trường trong dữ liệu huấn luyện và đăng ký. Do vậy mô hình có tính tổng quát hoá cao, nghĩa là ít nhạy cảm với điều kiện của môi trường cho kết quả tốt hơn.

Vì các lý do kể trên, đồ án đề xuất thay thế sử dụng SGD với momentum thay cho Adam trong mô hình cơ sở.

### 3.3.3 Hàm mất mát Angular Margin Prototypical

Như đã mô tả trong 3.2.3, hàm mất mát AP khuyến khích biểu diễn câu nói của một người gần hơn tới nguyên mẫu  $c_i$  của người đó hơn là các nguyên mẫu khác. Tuy vậy, khoảng cách giữa các câu nói của một người còn khá lớn và các câu khác người nói ở gần biên quyết định hàm softmax có khoảng cách nhỏ. Đây cũng là điểm yếu của hàm softmax mà nhiều nghiên cứu trước đây cũng đã chỉ ra. Biên quyết định yếu có thể gây ra tỉ lệ nhầm lẫn cao khi triển khai mô hình trong thực tế.

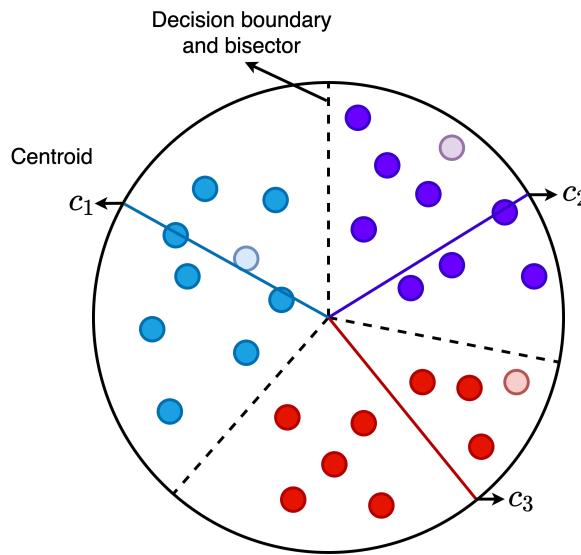


Figure 3.9: Mô tả biểu diễn người nói học bởi hàm softmax trong không gian góc. Đường kẻ chấm đen là đường phân giác giữa 2 tâm

Nhận thấy sự hiệu quả của việc dùng hệ số phạt lè trong các hàm mất mát phân loại như AM-Softmax [42] hay [12], tác giả đề xuất hàm mất mát Angular Margin Prototypical (AMP) thêm hệ số phạt lè vào hàm AP. Hàm AMP có thể được chia thành 2 loại AMP-cos hoặc AMP-arc phụ thuộc vào cách thêm hệ số phạt vào điểm tương đồng cô-sin hay góc giữa điểm biểu diễn và nguyên mẫu.

#### AMP-cos

Trong hàm AMP-cos, hệ số phạt lè được thêm trực tiếp vào điểm tương đồng của 2 câu. Công thức tính điểm tương đồng được thay đổi như sau:

$$S_{i,k} = \begin{cases} w \cdot (\cos(\theta_{x_{i,M}, c_k}) - m) + b, & \text{if } i = k \\ w \cdot \cos(\theta_{x_{i,M}, c_k}) + b, & \text{otherwise} \end{cases} \quad (3.9)$$

trong đó  $\theta_{\mathbf{x}_{i,M}, \mathbf{c}_k}$  là góc giữa truy vấn  $\mathbf{x}_{i,M}$  của người  $i$  và nguyên mẫu  $\mathbf{c}_k$  của người  $k$ ,  $w$  và  $b$  là các trọng số học được, và  $m$  là hệ số phạt lè. Thay vào công thức 3.7, ta được hàm mất mát AMP-cos như sau:

$$L_{AMP-cos} = -\frac{1}{N} \sum_{i=1}^N \log \frac{e^{w \cdot (\cos(\theta_{\mathbf{x}_{i,M}, \mathbf{c}_i}) - m) + b}}{e^{w \cdot (\cos(\theta_{\mathbf{x}_{i,M}, \mathbf{c}_i}) - m) + b} + \sum_{k=1, k \neq i}^N e^{w \cdot \cos(\theta_{\mathbf{x}_{i,M}, \mathbf{c}_k}) + b}} \quad (3.10)$$

### AMP-arc

Hệ số phạt lè được thêm vào góc giữa 2 câu trong hàm AMP-arc. Công thức tính điểm tương đồng được thay đổi như sau:

$$S_{i,k} = \begin{cases} w \cdot \cos(\theta_{\mathbf{x}_{i,M}, \mathbf{c}_k} + m) + b, & \text{if } i = k \\ w \cdot \cos(\theta_{\mathbf{x}_{i,M}, \mathbf{c}_k}) + b, & \text{otherwise} \end{cases} \quad (3.11)$$

Thay công thức trên vào 3.7, ta được hàm mất mát AMP-arc như sau:

$$L_{AMP-arc} = -\frac{1}{N} \sum_{i=1}^N \log \frac{e^{w \cdot \cos(\theta_{\mathbf{x}_{i,M}, \mathbf{c}_i} + m) + b}}{e^{w \cdot \cos(\theta_{\mathbf{x}_{i,M}, \mathbf{c}_i} + m) + b} + \sum_{k=1, k \neq i}^N e^{w \cdot \cos(\theta_{\mathbf{x}_{i,M}, \mathbf{c}_k}) + b}} \quad (3.12)$$

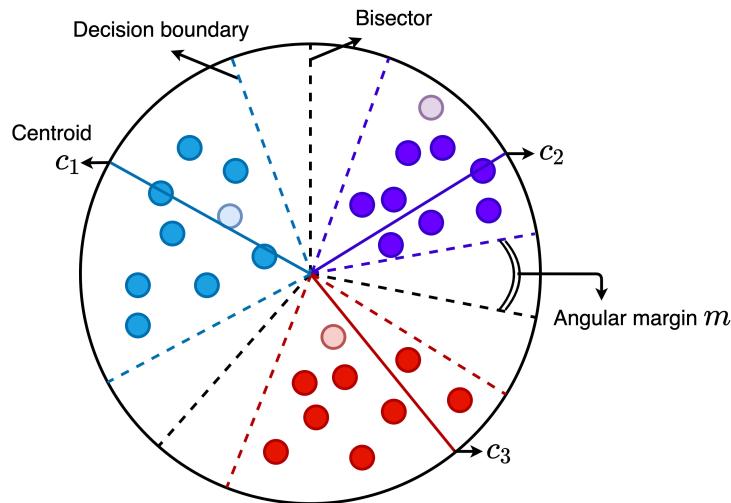


Figure 3.10: Mô tả biểu diễn người nói học bởi hàm AMP-arc trong không gian góc.

Sự co cụm của các câu người nói (intra-class compactness) và khoảng cách với các người nói khác (inter-class separability) là 2 yếu tố chính đóng góp cho khả

năng phân biệt biểu diễn người nói trong không gian vec-tơ. Việc thêm hệ số phạt lè trực tiếp làm tăng khoảng cách giữa các người nói và gián tiếp làm co cụm vùng biểu diễn của một người (Hình 3.10).

Dồ án sử dụng cả

## Chapter 4

# Thực nghiệm và đánh giá

### 4.1 Cơ sở dữ liệu

Để xây dựng cơ sở dữ liệu cho phần thực nghiệm, tác giả tổng hợp và sàng lọc dữ liệu từ ba bộ dữ liệu công khai: ZaloAI, VIVOS Corpus và VLSP ASR 2020.

Bắt đầu từ năm 2018, ZaloAI challenge [3] là một cuộc thi thường niên tập trung vào trí tuệ nhân tạo do Zalo Group, VNG tổ chức. Năm 2020, ZaloAI challenge thử thách các nhà phát triển và kỹ sư học máy Việt Nam với ba bài toán: tóm tắt tin tức, phát hiện biến báo giao thông, và xác thực người nói. Bộ dữ liệu huấn luyện công khai của bài toán xác thực giọng nói bao gồm 400 danh tính tiếng Việt thu thập từ chương trình truyền hình Bạn muốn hẹn hò. Mỗi danh tính có trung bình 26.4 câu nói với phân phối mô tả trong Hình 4.1. Bộ dữ liệu tuy đa dạng về mặt độ tuổi giới tính tuy nhiên vẫn còn vấn đề như: trùng lặp danh tính, nhiều âm thanh như nhạc nền, người nói phía sau, câu nói của danh tính này lẫn vào danh tính kia, ...

VIVOS là tập giọng nói tiếng Việt phục vụ cho bài toán nhận dạng tiếng nói thu thập bởi phòng thí nghiệm khoa học máy tính AILAB từ trường Đại học Khoa học Tự nhiên - Đại học Quốc Gia TP.HCM [2]. Tuy chủ đích của bộ dữ liệu là dành cho nhận dạng tiếng nói nhưng lại có nhãn danh tính cụ thể nên có thể sử dụng cho bài toán nhận dạng người nói. Tập huấn luyện VIVOS bao gồm 40 danh tính với trung bình 253.5 câu nói mỗi người. Tập kiểm thử có 19 danh tính không trùng với tập huấn luyện với trung bình 40 câu nói mỗi người. Chất lượng dữ liệu của VIVOS rất tốt do điều kiện thu âm được kiểm soát nên không yêu cầu xử lý gì thêm.

Bộ dữ liệu VLSP ASR 2020 [1] nằm trong chiến dịch đánh giá năm 2020 của

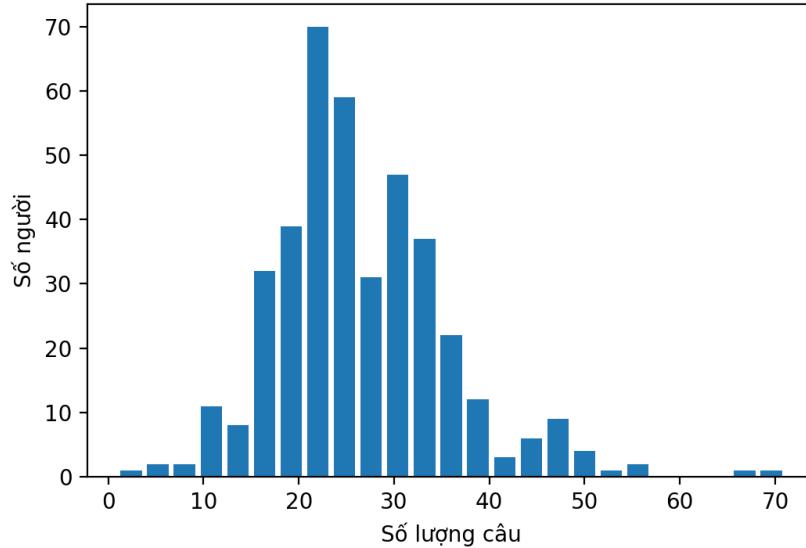


Figure 4.1: Biểu đồ phân phối số danh tính theo số câu nói của bộ dữ liệu ZaloAI

Hiệp hội xử lý Ngôn ngữ và Tiếng nói tiếng Việt. Giống như VIVOS, VLSP ASR 2020 được thu thập và thiết kế cho bài toán nhận diện giọng nói nhưng có nhãn danh tính cho các câu nói. Tổng số danh tính trong VLSP ASR 2020 là 567 người với trung bình 22.3 câu mỗi người. Tuy nhiên, dữ liệu danh tính của bộ dữ liệu lại không được chuẩn xác và có nhiều vấn đề tương tự như bộ ZaloAI. Những vấn đề này được giải quyết bằng phương pháp mô tả trong 4.1.1.

#### 4.1.1 Cải thiện chất lượng bộ dữ liệu

Hai bộ ZaloAI và VLSP có tổng cộng gần 1 nghìn danh tính và hơn 20 nghìn câu. Do vậy, việc kiểm tra dữ liệu rất khó khăn và tốn thời gian. Việc này còn trở nên khó khăn hơn khi đánh giá bằng tai người, ví dụ để phân biệt giọng của 2 người cùng là nam, giọng trầm miền bắc thì cần sự tập trung cao độ để tìm điểm khác biệt. Vì thế, đồ án phân tích ma trận tương đồng của các câu nói để tìm ra sự không nhất quán từ đó thu hẹp phạm vi kiểm tra.

Cho một tập biểu diễn  $n$  đoạn âm thanh đầu vào  $\mathbf{V} = \{\mathbf{v}_0, \mathbf{v}_1, \dots, \mathbf{v}_{n-1}\}$ , sử dụng độ tương đồng cô-sin, ma trận tương đồng cho các đoạn tiếng nói được tính theo công thức 4.1. Ví dụ một ma trận tương đồng trong Hình 4.2, đường chéo chính có giá trị tương đồng là 1 do so sánh mỗi câu với chính câu đó.

$$\mathbf{S}_{i,j} = \cos(\mathbf{v}_i, \mathbf{v}_j) = \frac{\mathbf{v}_i \cdot \mathbf{v}_j}{\|\mathbf{v}_i\| \|\mathbf{v}_j\|}, 0 \leq i, j \leq n \quad (4.1)$$

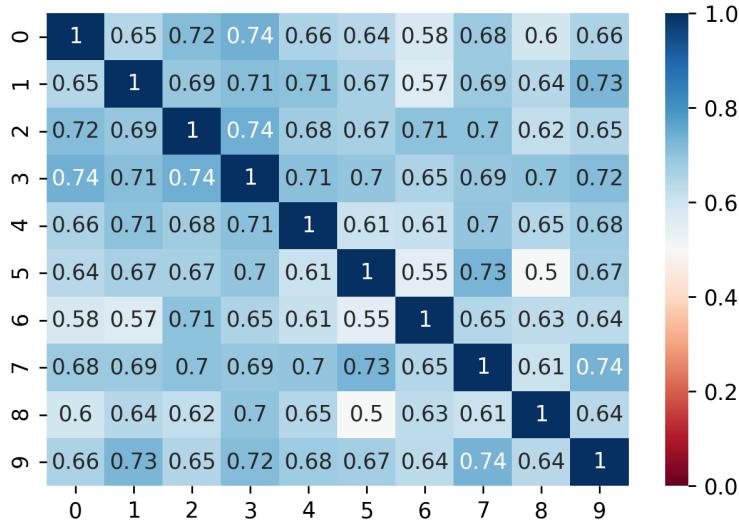


Figure 4.2: Ma trận tương đồng cho một tập 10 đoạn âm thanh của một người

Ma trận tương đồng được áp dụng khá rộng rãi, trong đó, phô biến nhất là phân tích âm nhạc dựa trên nội dung [33], phân tích văn bản [17] và tin sinh [7]. Các kỹ thuật được sử dụng chủ yếu là phân cụm và phân đoạn. Trong đồ án, tác giả chỉ sử dụng phân tích đơn thuần để tìm ra các người nói, câu nói có khả năng bị gán nhãn sai.

#### Loại bỏ người nói không hợp lệ

Việc loại bỏ một danh tính có thể do nhiều lý do: nhiều câu nói không thuộc về người đó, chất lượng âm thanh kém, môi trường xung quanh ồn ào, tệp âm thanh bị hư hại qua đường truyền hoặc thiết bị, ... Các nguyên nhân này dẫn đến việc chất giọng của danh tính không được đảm bảo gây bất lợi cho việc huấn luyện mô hình. Một số danh tính có số lượng câu có vấn đề lớn, làm sạch và loại bỏ từng câu bằng việc nghe rất tốn thời gian và công sức. Do vậy, việc loại bỏ hẳn những danh tính này là cần thiết. Khi nhìn vào ma trận tương đồng của một danh tính, có thể thấy được và loại bỏ những danh tính không hợp lệ. Ma trận tương đồng của một người hợp lệ và bị loại bỏ có thể được thấy trong Hình 4.2 và Hình 4.4 tương ứng.

#### Loại bỏ đoạn tiếng nói không hợp lệ

Các đoạn tiếng nói không hợp lệ bao gồm sai nhầm danh tính, độ dài quá ngắn, tiếng ồn xung quanh quá lớn hay trong một đoạn có giọng của nhiều người khác

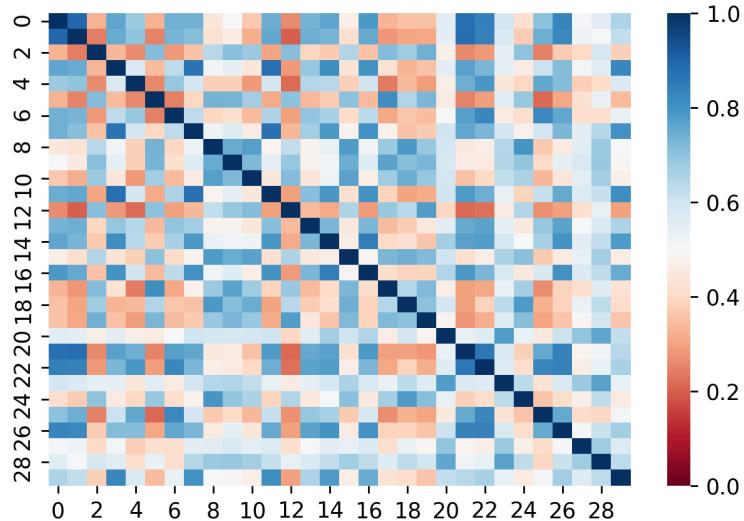


Figure 4.3: Ma trận tương đồng của một danh tính bị loại bỏ

nhau. Các đoạn này làm cho việc huấn luyện mô hình gặp khó khăn và giảm chất lượng của mô hình đầu ra. Lấy ma trận tương đồng của một người có câu nói không hợp lệ (Hình 4.4) làm ví dụ, để lọc ra đoạn có chỉ số 6 khá đơn giản bằng cách lấy một ngưỡng thấp (ví dụ 0.3). Những câu có độ tương đồng so với những câu khác của một danh tính mà dưới ngưỡng này ta sẽ xem là không hợp lệ. Tuy nhiên, cách này không hợp lý với những câu như câu chỉ số 2 trong Hình 4.4, có điểm nằm trong khoảng 0.4 - 0.6. Tuy có điểm tương đồng khá cao nhưng những câu này cũng cần được kiểm tra. Những câu này có thể được tìm thấy bằng cách phát hiện ngoại lệ sử dụng khoảng trong tứ phân vị (Interquartile range) [47].

Với,  $Q1, Q3$  lần lượt là tứ phân vị thứ nhất và thứ ba của tập điểm trung bình của các câu  $a_i = \frac{1}{n} \sum_{j=0, j \neq i}^{n-1} S_{i,j}$ , dựa trên khoảng trong tứ phân vị, đoạn điểm tương đồng hợp lệ cho tập điểm  $\mathbf{a}$  được tính như sau:

$$a_{min} = Q1 - 1.5 * IQR; \quad a_{max} = Q3 + 1.5 * IQR \quad (4.2)$$

$$IQR = Q3 - Q1 \quad (4.3)$$

Các câu có điểm trung bình  $a_i$  nằm ngoài đoạn  $[a_{min}, a_{max}]$  được đánh dấu và cần nghe lại để quyết định có loại bỏ hay không.

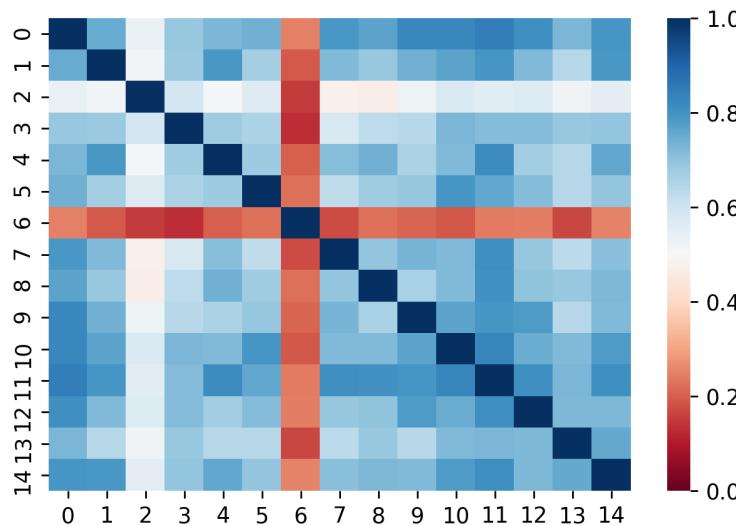


Figure 4.4: Ma trận tương đồng của một danh tính có đoạn âm thanh không hợp lệ

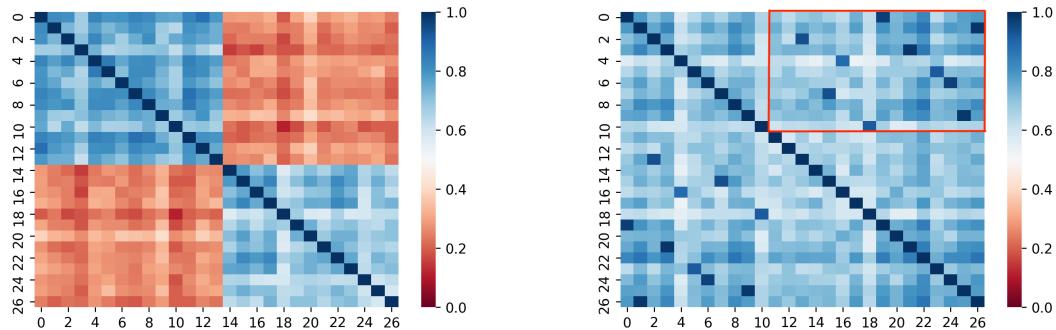
#### Hợp nhất người nói có cùng danh tính

Do các bộ dữ liệu được thu thập một cách độc lập, có khả năng người nói trong bộ dữ liệu này trùng với bộ kia. Hơn nữa, một người đã tồn tại trong cơ sở dữ liệu cũng có khả năng được yêu cầu thu lại. Các cặp người nói trùng danh tính có thể được tìm dựa vào ma trận tương đồng chéo. Từ Hình a mô tả ma trận tương đồng của người nói 52-M-31 và 64-M-30, có thể thấy rõ đây là 2 người khác nhau do ma trận tương đồng chéo (phần màu đỏ) có điểm tương đồng rất thấp. Ngược lại, trong Hình 4.5, 2 người có nhãn khác nhau là 64-M-30 và 636-M-30 thực chất là cùng một người với ma trận tương đồng chéo nằm trong ô màu đỏ. Có nhiều cặp câu điểm tương đồng cao (ô xanh đậm) nhưng không đạt tối 1.0 như trên đường chéo chính do cơ bản có cùng nội dung nhưng khác biệt đến từ sự biến đổi nhất định trong quá trình xử lý. Các cặp người nói có giá trị trung bình của ma trận tương đồng chéo lớn hơn 0.7 yêu cầu được nghe lại và ra quyết định để hợp nhất.

Bảng 4.1 tổng kết thông tin sàng lọc dữ liệu. Số người nói bị loại bỏ chiếm **TODO stat** tổng số người nói, số người nói được hợp nhất chiếm **TODO stat** tổng số người nói, số câu nói bị loại bỏ chiếm **TODO stat** tổng số câu.

Table 4.1: placeholder

Bộ dữ liệu	Số danh tính loại bỏ	Số danh tính hợp nhất	Số câu loại bỏ
ZaloAI	0	51	1,066
VIVOS	0	0	2
VLSP	65	33	549
Tổng	65	84	1,617



(a) Ma trận tương đồng của 52-M-31 và 64-M-30      (b) Ma trận tương đồng của 64-M-30 và 636-M-30

Figure 4.5

#### 4.1.2 Bộ dữ liệu thực nghiệm

Sau khi loại bỏ các danh tính không phù hợp, hợp nhất người nói có cùng danh tính và loại bỏ các câu vấn đề, bộ dữ liệu thực nghiệm đã có chất lượng tương đối tốt. Tổng số lượng người nói là 1110, chia thành 3 tập: tập huấn luyện (training set) gồm 1031 người nói, tập kiểm thử (validation set) gồm 20 người, tập kiểm tra (test set) gồm 59 người nói. Người nói trong tập kiểm thử và 40 người trong tập kiểm tra được lấy ngẫu nhiên trong bộ ZaloAI với điều kiện cân bằng giới tính nam - nữ. 19 người còn lại trong tập kiểm tra là tập kiểm thử của bộ dữ liệu VIVOS.

## 4.2 Chi tiết cài đặt thực nghiệm

### Thông số huấn luyện mô hình

Các thực nghiệm trong mục tiếp theo đều được chạy trên cùng bộ thông số như sau:

- Mạng trích xuất đặc trưng: ResNet
- Lớp tổng hợp: Tổng hợp thống kê tập trung
- Batch size: 100

### Môi trường lập trình

Để cài đặt thực nghiệm, tác giả sử dụng ngôn ngữ lập trình Python kết hợp với thư viện PyTorch phiên bản 1.7.1. PyTorch là thư mã nguồn mở của Facebook

được xây dựng trên ngôn ngữ lập trình Lua. PyTorch cho phép người dùng xây dựng, tuỳ biến mô hình ở cả cấp cao và cấp thấp với thiết kế trực quan.

#### Môi trường thực nghiệm

Để thực hiện huấn luyện các mô hình, tác giả sử dụng Google Colaboratory: Hệ điều hành Ubuntu 18.04, 2vCPU Intel Xeon 2.2 Ghz, RAM 25GB, GPU Tesla T4 15GB.

### 4.3 Kết quả thực nghiệm và đánh giá

Trong phần này tác giả sử dụng tỉ lệ lỗi bằng nhau (Equal error rate - EER) để đánh giá hiệu năng của các mô hình thực nghiệm. EER là điểm nằm cắt nhau giữa đường tỉ lệ chấp nhận giả (False acceptance rate - FAR, tỉ lệ mà người xâm nhập được coi là người dùng hợp lệ) và đường tỉ từ chối giả (False rejection rate - FRR, tỉ lệ mà người dùng hợp lệ bị từ chối) khi điều chỉnh ngưỡng (Hình 4.6). Mô hình nhận dạng người nói càng hiệu quả thì có EER càng nhỏ.

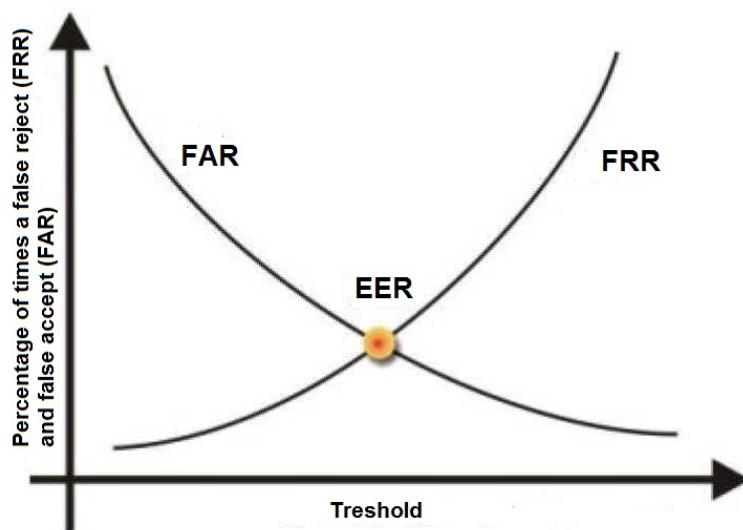


Figure 4.6: Mô tả EER <sup>1</sup>

Có tổng cộng 1626 đoạn âm thanh của 59 người nói trong tập kiểm thử, các cặp câu hợp lệ và câu không hợp lệ được sinh ra từ mọi cặp câu có thể. Hệ thống tính điểm cho một cặp câu bằng việc tính độ tương đồng cô-sin của biểu diễn của 2 đoạn. Hai câu nói của cùng một người nói được coi là hợp lệ; hai câu nói từ 2 người nói khác nhau được coi là câu nói không hợp lệ. Với một ngưỡng cho trước, nếu cặp điểm câu nói hợp lệ dưới ngưỡng này (độ tương đồng thấp), câu

<sup>1</sup><https://wenzwu.com/2019/05/05/which-is-more-important-accuracy-or-acceptability/>

nói đó được tính vào tỉ lệ từ chối giả. Ngược lại, nếu một cặp câu nói không hợp lệ có điểm nằm trên ngưỡng (tỉ lệ tương đồng cao), câu nói được tính vào tỉ lệ chấp nhận giả.

Tác giả thực hiện nhiều trường hợp thực nghiệm khác nhau với mục tiêu huấn luyện mô hình nhận dạng người nói một cách hiệu quả trên tiếng Việt. Thực nghiệm 1 đánh giá hiệu quả của việc làm sạch dữ liệu như mô tả bên trên. Thực nghiệm 2 khảo sát các cách huấn luyện khác nhau so với transfer learning. Thực nghiệm 3 đánh giá hiệu quả trong việc khử nhiễu âm thanh trong. Thực nghiệm 4 so sánh mô hình khi sử dụng phương thức tối ưu Adam và SGD. Thực nghiệm 5 kiểm tra tính hiệu quả của hàm mất mát AMP-cos và AMP-arc với các giá trị hệ số phạt khác nhau.

#### **Thực nghiệm 1: làm sạch dữ liệu**

Bảng 4.2 mô tả kết quả thực nghiệm với dữ liệu ban đầu và dữ liệu đã được sàng lọc như đã trình bày trong 4.1.1. Như có thể thấy, kết quả huấn luyện trên tập đã sàng lọc cải thiện 0.93% EER so với dữ liệu gốc. Do vậy, các thực nghiệm về sau sẽ sử dụng bộ dữ liệu đã qua sàng lọc.

Table 4.2: EER trên tập kiểm tra với dữ liệu trước và sau khi cải thiện chất lượng

Dữ liệu	EER trên tập kiểm tra
Gốc	6.790%
Cải thiện chất lượng	5.860%

#### **Thực nghiệm 2: Phương thức huấn luyện**

Trong thực nghiệm này, tác giả tiến hành khảo sát các phương thức huấn luyện mô hình. Bảng 4.3 mô tả kết quả với các trường hợp khác nhau bao gồm: mô hình huấn luyện sẵn [15], huấn luyện từ đầu trên dữ liệu tiếng Anh và tiếng Việt, huấn luyện từ đầu chỉ trên dữ liệu tiếng Việt, transfer learning trên dữ liệu tiếng Việt.

Do số người nói trong bộ dữ liệu tiếng Việt ít hơn hẳn so với người nói tiếng Anh trong bộ dữ liệu VoxCeleb, huấn luyện không tập trung đủ để tìm ra các đặc trưng hữu ích để phân biệt người nói tiếng Việt, dẫn đến mô hình huấn luyện từ đầu kết hợp hai bộ dữ liệu đạt kết quả tệ hơn. Mô hình cơ sở (mô tả trong phần 3.2) huấn luyện từ đầu trên dữ liệu tiếng Việt đạt kết quả 5.860% EER. Có thể thấy finetune bằng riêng dữ liệu tiếng Việt cho kết quả vượt trội so với huấn luyện từ đầu bằng bộ dữ liệu tiếng Việt hoặc kết hợp VoxCeleb (dữ liệu

tiếng Anh) với EER 4.775%. Kết quả cho thấy các đặc trưng người nói trong tiếng anh góp phần cải thiện mô hình tiếng Việt. Các thử nghiệm phía sau sử dụng phương pháp transfer learning trên bộ dữ liệu tiếng Việt.

Table 4.3: EER trên tập kiểm tra với các phương pháp huấn luyện và dữ liệu khác nhau

Phương pháp huấn luyện	Dữ liệu huấn luyện	EER trên tập kiểm tra
Pretrain	-	10.60%
Huấn luyện từ đầu	Tiếng Việt & VoxCeleb	6.294%
Huấn luyện từ đầu	Tiếng Việt	5.860%
Transfer learning	Tiếng Việt	<b>4.775%</b>

**TODO** name vietnamese dataset for ease in reference

### Thực nghiệm 3: Khử tạp âm trong tín hiệu giọng nói

Do chỉ có VIVOS là được thu thập từ phòng thu âm, tín hiệu giọng nói trong tập dữ liệu còn chứa nhiều tạp âm, ví dụ nhạc nền, tiếng ồn nhỏ xung quanh, âm thanh đường phố xe cộ. Các nhiễu tạp âm có khả năng cản trở mô hình học được chất giọng cần học từ dữ liệu. Do đó, tác giả sử dụng dữ liệu khử tạp âm dùng mô hình do bộ phận nghiên cứu trí tuệ nhân tạo tại Facebook phát triển [10] và đánh giá hiệu quả trong thực nghiệm này. Bảng 4.4 cho thấy việc khử tạp âm có hiệu quả cao với 0.560% EER cải thiện trên tập kiểm tra. Do vậy, trong các thực nghiệm tiếp theo, các mô hình được huấn luyện trên dữ liệu đã lọc tạp âm.

Table 4.4: EER trên tập kiểm tra của mô hình huấn luyện với dữ liệu còn nhiều âm thanh và đã khử tạp âm

Dữ liệu huấn luyện	EER trên tập kiểm tra
Chứa tạp âm	4.775%
Khử tạp âm	<b>4.215%</b>

### Thực nghiệm 4: Phương pháp tối ưu

Table 4.5: EER trên tập kiểm tra của mô hình huấn luyện với Adam và SGD

Dữ liệu	EER trên tập kiểm tra
Adam	4.215%
SGD	<b>2.930%</b>

### Thực nghiệm 5: Hàm mất mát

- Visualizing embedding space with t-SNE (some speakers) - DET curve

Table 4.6: EER trên tập kiểm tra của mô hình huấn luyện với hàm mất mát AP, AMP-cos và AMP-arc với các giá trị phạt khác nhau

Hàm mất mát	EER trên tập kiểm tra
AP	2.930%
AMP-cos (m=0.1)	2.789%
AMP-cos (m=0.2)	2.749%
AMP-cos (m=0.3)	2.754%
AMP-cos (m=0.4)	2.804%
AMP-cos (m=0.5)	2.892%
AMP-arc (m=0.1)	2.782%
AMP-arc (m=0.2)	<b>2.698%</b>
AMP-arc (m=0.3)	2.791%
AMP-arc (m=0.4)	2.790%
AMP-arc (m=0.5)	2.799%

## **Chapter 5**

### **Conclusions**

# Bibliography

- [1] Automatic speech recognition for vietnamese. <https://vlsp.org.vn/vlsp2020/eval/asr>. Truy cập vào: 10-05-2021.
- [2] Vivos corpus. <https://ailab.hcmus.edu.vn/vivos>. Truy cập vào: 10-05-2021.
- [3] Zalo ai challenge. <https://challenge.zalo.ai/>. Truy cập vào: 10-05-2021.
- [4] Yusuf Aytar and Andrew Zisserman. Tabula rasa: Model transfer for object category detection. In *2011 international conference on computer vision*, pages 2252–2259. IEEE, 2011.
- [5] Bidhan Barai, Debayan Das, Nibaran Das, Subhadip Basu, and Mita Nasipuri. An asr system using mfcc and vq/gmm with emphasis on environmental dependency. In *2017 IEEE Calcutta Conference (CALCON)*, pages 362–366. IEEE, 2017.
- [6] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Karpman, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.
- [7] A Bustamam, F Zubedi, and Titin Siswantining. Implementation  $\chi$ -sim co-similarity and agglomerative hierarchical to cluster gene expression data of lymphoma by gene and condition. In *AIP Conference Proceedings*, volume 2023, page 020221. AIP Publishing LLC, 2018.
- [8] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546. IEEE, 2005.
- [9] Joon Son Chung, Jaesung Huh, Seongkyu Mun, Minjae Lee, Hee Soo Heo, Soyeon Choe, Chiheon Ham, Sunghwan Jung, Bong-Jin Lee, and Icksang

- Han. In defence of metric learning for speaker recognition. *arXiv preprint arXiv:2003.11982*, 2020.
- [10] Alexandre Defossez, Gabriel Synnaeve, and Yossi Adi. Real time speech enhancement in the waveform domain. In *Interspeech*, 2020.
- [11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.
- [12] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2019.
- [13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- [14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [15] Hee Soo Heo, Bong-Jin Lee, Jaesung Huh, and Joon Son Chung. Clova baseline system for the voxceleb speaker recognition challenge 2020. *arXiv preprint arXiv:2009.14153*, 2020.
- [16] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [17] Syed Fawad Hussain, Gilles Bisson, and Clément Grimal. An improved co-similarity measure for document clustering. In *2010 Ninth International Conference on Machine Learning and Applications*, pages 190–197. IEEE, 2010.
- [18] Nitish Shirish Keskar and Richard Socher. Improving generalization performance by switching from adam to sgd. *arXiv preprint arXiv:1712.07628*, 2017.
- [19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

- [21] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.
- [22] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 212–220, 2017.
- [23] Tom M Mitchell et al. Machine learning. 1997.
- [24] Arsha Nagrani, Joon Son Chung, and Andrew Zisserman. Voxceleb: a large-scale speaker identification dataset. *arXiv preprint arXiv:1706.08612*, 2017.
- [25] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.
- [26] Jiazhi Ni, Jie Liu, Chenxin Zhang, Dan Ye, and Zhirou Ma. Fine-grained patient similarity measuring using deep metric learning. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1189–1198, 2017.
- [27] Koji Okabe, Takafumi Koshinaka, and Koichi Shinoda. Attentive statistics pooling for deep speaker embedding. *arXiv preprint arXiv:1803.10963*, 2018.
- [28] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.
- [29] FA Rezaur rahman Chowdhury, Quan Wang, Ignacio Lopez Moreno, and Li Wan. Attention-based models for text-dependent speaker verification. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5359–5363. IEEE, 2018.
- [30] Mirco Ravanelli and Yoshua Bengio. Speaker recognition from raw waveform with sincnet. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 1021–1028. IEEE, 2018.
- [31] Dipanjan Sarkar, Raghav Bali, and Tamoghna Ghosh. *Hands-On Transfer Learning with Python: Implement advanced deep learning and neural network models using TensorFlow and Keras*. Packt Publishing Ltd, 2018.
- [32] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the*

- IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.
- [33] Diego F Silva, Chin-Chia M Yeh, Yan Zhu, Gustavo EAPA Batista, and Eamonn Keogh. Fast similarity matrix profile for music analysis and exploration. *IEEE Transactions on Multimedia*, 21(1):29–38, 2018.
  - [34] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
  - [35] David Snyder, Daniel Garcia-Romero, Daniel Povey, and Sanjeev Khudanpur. Deep neural network embeddings for text-independent speaker verification. In *Interspeech*, pages 999–1003, 2017.
  - [36] David Snyder, Daniel Garcia-Romero, Gregory Sell, Alan McCree, Daniel Povey, and Sanjeev Khudanpur. Speaker recognition for multi-speaker conversations using x-vectors. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5796–5800. IEEE, 2019.
  - [37] David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur. X-vectors: Robust dnn embeddings for speaker recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5329–5333. IEEE, 2018.
  - [38] David Snyder, Jesús Villalba, Nanxin Chen, Daniel Povey, Gregory Sell, Najim Dehak, and Sanjeev Khudanpur. The jhu speaker recognition system for the voices 2019 challenge. In *INTERSPEECH*, pages 2468–2472, 2019.
  - [39] Ehsan Variani, Xin Lei, Erik McDermott, Ignacio Lopez Moreno, and Javier Gonzalez-Dominguez. Deep neural networks for small footprint text-dependent speaker verification. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4052–4056. IEEE, 2014.
  - [40] Jesús Villalba, Nanxin Chen, David Snyder, Daniel Garcia-Romero, Alan McCree, Gregory Sell, Jonas Borgstrom, Fred Richardson, Suwon Shon, François Grondin, et al. State-of-the-art speaker recognition for telephone and video speech: The jhu-mit submission for nist sre18. In *Interspeech*, pages 1488–1492, 2019.
  - [41] Alex Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J Lang. Phoneme recognition using time-delay neural networks.

- IEEE transactions on acoustics, speech, and signal processing*, 37(3):328–339, 1989.
- [42] Feng Wang, Jian Cheng, Weiyang Liu, and Haijun Liu. Additive margin softmax for face verification. *IEEE Signal Processing Letters*, 25(7):926–930, 2018.
- [43] Jian Wang, Feng Zhou, Shilei Wen, Xiao Liu, and Yuanqing Lin. Deep metric learning with angular loss. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2593–2601, 2017.
- [44] Zhuoyi Wang, Hemeng Tao, Zelun Kong, Swarup Chandra, and Latifur Khan. Metric learning based framework for streaming classification with concept evolution. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.
- [45] Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nathan Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. *arXiv preprint arXiv:1705.08292*, 2017.
- [46] Weidi Xie, Arsha Nagrani, Joon Son Chung, and Andrew Zisserman. Utterance-level aggregation for speaker recognition in the wild. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5791–5795. IEEE, 2019.
- [47] Jiawei Yang, Susanto Rahardja, and Pasi Fränti. Outlier detection: how to threshold outlier scores? In *Proceedings of the international conference on artificial intelligence, information processing and cloud computing*, pages 1–6, 2019.
- [48] Muhamad Yani et al. Application of transfer learning using convolutional neural network method for early detection of terry’s nail. In *Journal of Physics: Conference Series*, volume 1201, page 012052. IOP Publishing, 2019.
- [49] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.
- [50] Chunlei Zhang, Kazuhito Koishida, and John HL Hansen. Text-independent speaker verification based on triplet convolutional neural network embeddings. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(9):1633–1644, 2018.

- [51] Yingke Zhu, Tom Ko, David Snyder, Brian Mak, and Daniel Povey. Self-attentive speaker embeddings for text-independent speaker verification. In *Interspeech*, volume 2018, pages 3573–3577, 2018.