

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN



**XÂY DỰNG MÔ HÌNH SPEECH-TO-TEXT TRÊN
NỀN TẢNG NGÔN NGỮ TIẾNG VIỆT**

BÁO CÁO MÔN HỌC MÁY
Ngành: Kỹ thuật Điện tử và Tin học

Cán bộ hướng dẫn : TS. Nguyễn Tiến Cường
CN.Vi Anh Quân

Sinh viên thực hiện : Nguyễn Viết Tuấn – 22001363
Nguyễn Minh Hiếu – 22001321
Nguyễn Tiến Đồng – 22001314
Vũ Trung Kiên – 22001325

HÀ NỘI - 2025

LỜI MỞ ĐẦU

Ngôn ngữ đã và đang là một phần quan trọng trong cuộc sống con người, là cầu nối giữa các thế hệ và văn hóa. Trong thời đại số hóa và kỹ thuật số hóa ngày càng phát triển, khả năng tận dụng ngôn ngữ đã trở thành một yếu tố quan trọng cho sự phát triển của xã hội và kinh tế. Việc chuyển đổi từ lời nói thành văn bản, còn được gọi là “Speech-to-Text” đang trở nên ngày càng quan trọng và thú vị, nó đã trở thành một công cụ mạnh mẽ giúp con người tương tác với máy tính và các thiết bị điện tử một cách hiệu quả hơn. Đặc biệt, trong bối cảnh của ngôn ngữ Tiếng Việt, một trong những ngôn ngữ đa dạng và phong phú về ngữ điệu, từ vựng và ngữ pháp, việc phát triển một hệ thống chuyển đổi giọng nói thành văn bản đối với Tiếng Việt đòi hỏi sự nghiên cứu và phát triển công nghệ sâu rộng.

Trong bối cảnh tiến bộ công nghệ thông tin và sự phát triển của ngôn ngữ Việt Nam, đề tài này tập trung vào việc xây dựng một mô hình “Speech-to-Text” đặc biệt dành riêng cho tiếng Việt. Tuy nhiên, Tiếng Việt là một ngôn ngữ đa dạng với hệ thống âm vị phong phú và ngữ pháp phức tạp. Điều này tạo ra nhiều thách thức trong việc hiểu và chuyển đổi lời nói thành văn bản, nó đòi hỏi sự kết hợp giữa các lĩnh vực như xử lý ngôn ngữ tự nhiên, học máy và xử lý tín hiệu âm thanh để tạo ra một hệ thống mạnh mẽ có khả năng chuyển đổi đoạn hội thoại bằng tiếng Việt thành văn bản một cách chính xác.

Đề tài này xoay quanh việc xây dựng một mô hình Speech-to-Text trên nền tảng ngôn ngữ Tiếng Việt, một thách thức đầy triển vọng và tiềm năng. Đây không chỉ là một nhiệm vụ kỹ thuật, mà còn là một bước tiến quan trọng trong việc thúc đẩy sự tiến bộ của công nghệ ngôn ngữ và tạo ra những ứng dụng thú vị trong các lĩnh vực như trí tuệ nhân tạo, giao tiếp, giáo dục và y tế...

Trong báo cáo này, chúng ta sẽ xem xét chi tiết về mục tiêu, phạm vi của việc xây dựng mô hình "Speech-to-Text" trên nền tảng ngôn ngữ tiếng Việt. Chúng ta sẽ khám phá tại sao việc phát triển một mô hình như vậy có ý nghĩa quan trọng, những thách thức cụ thể mà mô hình sẽ phải đối mặt, và sự cần thiết của sự hợp tác đa ngành để đạt được mục tiêu này.

Thành viên	Công việc	Tỷ lệ đóng góp
Nguyễn Viết Tuấn	Tải và train mô hình	25%
Nguyễn Minh Hiếu	Tạo bộ tokenizer, làm giao diện web	25%
Nguyễn Tiến Đồng	Xử lý dữ liệu đầu vào	25%
Vũ Trung Kiên	Kiểm thử	25%

MỤC LỤC

LỜI MỞ ĐẦU	ii
MỤC LỤC	iv
DANH MỤC CÁC BẢNG HÌNH.....	vi
CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI.....	1
1.1. Giới thiệu đề tài.....	1
1.2. Lý do chọn đề tài.....	1
1.3. Mục tiêu của đề tài	2
1.4. Phạm vi đề tài.....	3
1.5. Bố cục báo cáo	3
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT.....	5
2.1. Speech Recognition	5
2.1.1. Preprocessing	6
2.1.2. Feature Extraction.....	7
2.1.3. Classification Model.....	8
2.1.4. Language Model	9
2.2. Bối cảnh	12
2.2.1. Lịch sử hình thành	12
2.2.2. Thách thức	13
2.3. Các phương pháp	14
2.3.1. Hướng tiếp cận.....	14
2.3.2. Conventional Acoustic Models.....	15
2.3.3. Deep Neural Networks	17
2.3.4. DNN-HMM Hybrid Systems.....	19
2.3.5. Advanced Deep Models.....	20
2.3.6. Các phương pháp đánh giá	21
2.4. Các ứng dụng	22
2.5. Wav2vec 2.0	23
2.5.1. Feature encoder.....	23
2.5.2. Biểu diễn theo ngữ cảnh với Transformer	24
CHƯƠNG 3: XÂY DỰNG MÔ HÌNH.....	25
3.1. Chuẩn bị dữ liệu	25
3.2. Xử lý dữ liệu.....	27
3.3. Xây dựng bộ tokenizer	28
3.4. Trích xuất đặc trưng	30

3.5. Tạo Processor cho Wav2Vec.....	31
3.6. Giới hạn độ dài audio	31
3.7. Tạo trình đối chiếu dữ liệu DataCollator.....	32
3.8. Tạo trình tính độ đo metrics	32
3.9. Tạo mô hình huấn luyện	33
CHƯƠNG 4: THỰC NGHIỆM VÀ ĐÁNH GIÁ KẾT QUẢ.....	36
4.1. Kết quả thực nghiệm.....	36
4.2. So sánh với các mô hình nhận diện giọng nói khác	37
CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	39
5.1. Kết luận.....	39
5.2. Hướng phát triển.....	39
TÀI LIỆU THAM KHẢO	41

DANH MỤC CÁC BẢNG HÌNH

Hình 1: Kiến trúc chung của các mô hình nhận dạng âm thanh.....	5
Hình 2: Ví dụ về một vài dữ liệu âm thanh	6
Hình 3: Ví dụ về văn phạm phụ thuộc trong LM	10
Hình 4: Các công nghệ nhận dạng âm thanh.....	15
Hình 5: Kiến trúc Deep Neural Network.....	18
Hình 6: Hàm GELU ($\mu = 0, \sigma = 1$), ReLU, và ELU ($\alpha = 1$)	24
Hình 7: Quy trình xây dựng mô hình nhận diện giọng nói Wav2Vec	25
Hình 8: Phân bố dữ liệu đầu vào	27
Hình 9: Danh sách một vài ký tự sau khi lấy được	29
Hình 10: Mô hình SOTA cho Tiếng Việt trên bộ Common Voice 8.0	38
Bảng 1: Một số mô hình ngôn ngữ [20]	11
Bảng 2: Mô tả một điểm dữ liệu.....	26
Bảng 3: Cài đặt tham số mô hình	34
Bảng 4: Kết quả huấn luyện đợt 1 (epochs 1 - 7).....	36
Bảng 5: Kết quả huấn luyện đợt 2 (epochs 8 - 14).....	36
Bảng 6: Kết quả huấn luyện đợt 3 (epochs 15 - 21).....	37
Bảng 7: So sánh mô hình hiện tại với mô hình SOTA	38

DANH MỤC CÁC TỪ VIẾT TẮT

STT	Từ viết tắt	Ý nghĩa
1	AI	Artificial Intelligence
2	ANN	Artificial Neural Network
3	ASR	Automatic Speech Recognition
4	BLUE	Bilingual Evaluation Uderstudy
5	CTC	Connectionist Temporal Classification
6	DL	Deep Learning
7	DWT	Discrete Wavelet Transform
8	FFT	Fast Fourier Transform
9	GELU	Gaussian Error Linear Unit
10	GMM	Gaussian mixture models
11	HMM	Hidden Makov Model
12	ICA	Independent Component Analysis
13	LLM	Large Language Model
14	LM	Language Model
15	LPC	Linear predictive coding
16	LSTM	Long Short-Term Memory
17	MFCC	Melfrequency cepstral coefficients
18	ML	Machine Learning
19	MLP	Multi-Layers Perceptron
20	PCA	Principal Component Analysis
21	RNN	Recurrent Neural Network
22	SR	Speech Recognition
23	STR	Speech-to-Text Recognition
24	UBM	Universal Background Model
25	WER	Word Error Rate
26	WRR	Word Recognition Rate

CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI

1.1. Giới thiệu đề tài

Tiếng nói là một cách mà con người thể hiện ý nghĩa, tương tác xã hội và chia sẻ kiến thức. Trong cuộc cách mạng công nghiệp 4.0, khi công nghệ thông tin và truyền thông đang trải qua sự biến đổi mạnh mẽ, Speech-to-Text đã trở thành một phần quan trọng của cuộc sống hàng ngày. Đặc biệt, tại Việt Nam, việc phát triển và tối ưu hóa các mô hình chuyển đổi giọng nói thành văn bản trong ngôn ngữ Tiếng Việt đang trở thành một thách thức quan trọng và thú vị.

Tiếng Việt, với sự đa dạng trong cách phát âm và ngữ pháp, đặt ra nhiều thách thức trong việc chuyển đổi lời nói thành văn bản một cách chính xác. Để hiểu và xử lý tiếng Việt, mô hình Speech-to-Text cần kết hợp nhiều lĩnh vực công nghệ, từ xử lý ngôn ngữ tự nhiên đến xử lý tín hiệu âm thanh. Mục tiêu chính của đề tài này là tạo ra một hệ thống mạnh mẽ có khả năng chuyển đổi đoạn hội thoại bằng tiếng Việt thành văn bản với độ chính xác cao.

Sự thành công trong việc xây dựng mô hình Speech-to-Text tiếng Việt sẽ mang lại nhiều lợi ích to lớn. Nó không chỉ mang lại lợi ích cho cá nhân và cộng đồng mà còn có tác động tích cực lớn đến nhiều lĩnh vực và khía cạnh của xã hội và kinh tế. Nó cải thiện khả năng giao tiếp của những người khuyết tật ngôn ngữ hoặc người học ngoại ngữ, nâng cao sản xuất nội dung truyền thông và giáo dục, hỗ trợ ngành y tế trong việc quản lý thông tin bệnh án, và thúc đẩy phát triển các ứng dụng trí tuệ nhân tạo tiếng Việt. Đây là một bước tiến quan trọng trong việc tạo ra một xã hội thông minh và tiện lợi hơn cho mọi người.

1.2. Lý do chọn đề tài

AI đang ngày càng đóng một vai trò quan trọng hơn trong cuộc sống của con người nhờ những tiềm năng và ứng dụng của nó. Đặc biệt, những tiến bộ gần đây trong lĩnh vực này đã thay đổi cách chúng ta tương tác với nhau và với máy móc. Các hệ thống trợ lý ảo AI hàng đầu trên thế giới như Siri của Apple, Google Assistant của Google, Cortana của Microsoft, Alexa của Amazon và Watson của IBM có khả năng tương tác với con người thông qua văn bản hoặc giọng nói. Ngoài ra, còn có những

ứng dụng khác như xe tự hành, tổng đài trợ lý ảo, bản tin nói, và dịch máy tự động. Tất cả những hệ thống này đều được xây dựng dựa trên công nghệ nhận dạng giọng nói [1].

Nhiều ứng dụng là vậy, tuy nhiên mục tiêu cuối cùng của Speech Recognition là chuyển đổi giọng nói thành văn bản, hay Speech-to-Text. Cùng với sự phát triển của công nghệ, nhu cầu về ứng dụng chuyển đổi tiếng nói thành văn bản trong nhiều lĩnh vực, từ ghi chú họp, tạo phụ đề cho video, đến trợ lý ảo và các ứng dụng ghi âm ngày càng lớn. Điều này làm cho việc xây dựng một mô hình Speech-to-Text cho tiếng Việt trở thành một ứng dụng thực tế và có giá trị thương mại cao. Việc nghiên cứu và phát triển mô hình Speech-to-Text cho tiếng Việt có thể giúp thúc đẩy sự phát triển của công nghệ xử lý ngôn ngữ tự nhiên cho ngôn ngữ phức tạp này.

Với sự phát triển mạnh mẽ và tiềm năng của STR, các phương pháp xử lý và hướng tiếp cận khác nhau được các nhà nghiên cứu đẩy mạnh khám phá. Từ các phương pháp ML [2] đến DL [3], hay hệ thống STR trên các ngôn ngữ khác nhau được thực hiện [4]. Tuy nhiên, việc xây dựng hệ thống STR trên ngôn ngữ tiếng Việt vẫn còn nhiều hạn chế. Từ những hạn chế của STR trên nền tảng tiếng Việt, chúng tôi đề ra ý tưởng sử dụng một mô hình Self-Supervised Learning như Wav2vec để nhận dạng giọng nói đạt được hiệu suất cao. Hiện thực hóa ý tưởng xây dựng mô hình và hoàn thành đề tài này là việc cần thiết.

1.3. Mục tiêu của đề tài

Đề tài này đặt ra mục tiêu chính là xây dựng một mô hình AI có khả năng nhận diện và chuyển đổi tiếng nói thành văn bản chất lượng trên nền tảng ngôn ngữ Tiếng Việt. Chúng ta sẽ khám phá cách mà công nghệ này có thể đem lại lợi ích không chỉ cho việc ghi chép và tài liệu hóa ngôn ngữ mà còn cho nhiều lĩnh vực khác nhau như truyền thông, y học, giáo dục và nhiều lĩnh vực khác. Các mục tiêu được xác định cụ thể như sau:

- Đầu tiên, chúng ta sẽ tìm hiểu các khái niệm cũng như định nghĩa chung về Speech Recognition.
- Tìm hiểu nguyên nhân và bối cảnh ra đời của Speech Recognition cũng như Speech-to-Text Recognition.

- Tiếp đến, chúng ta sẽ tổng hợp các phương pháp và các hướng tiếp cận đã có, cùng với đó, nêu ra những ưu điểm và hạn chế cũng như những khó khăn mà lĩnh vực này vẫn đang còn mắc phải.
- Tổng quan về các ứng dụng của các mô hình xử lý tính hiệu và âm thanh.
- Sau đó, ta sẽ tìm hiểu về phương pháp được sử dụng chính cho mô hình nhận dạng âm thanh này – Wav2vec.
- Sau khi có được cơ sở lý thuyết và các nghiên cứu liên quan, ta tiến hành xây dựng mô hình.
- Tối ưu hóa mô hình, kiểm thử và đánh giá mô hình để mô hình có độ chính xác và hiệu suất tốt nhất.

Trong báo cáo này, chúng ta sẽ bắt đầu bằng việc thảo luận về ngữ cảnh và nhu cầu thúc đẩy đề tài này, cùng với những mục tiêu và ứng dụng tiềm năng mà nó mang lại cho cộng đồng ngôn ngữ Việt Nam và xa hơn nữa.

1.4. Phạm vi đề tài

Phạm vi nghiên cứu của đề tài này bao gồm việc phát triển mô hình Speech-to-Text cho tiếng Việt, xử lý đặc trưng của ngôn ngữ, xử lý dữ liệu, đánh giá hiệu suất, đề ra hướng ứng dụng thực tế và phân tích liên quan đến lĩnh vực xử lý tiếng nói và ngôn ngữ tự nhiên cho tiếng Việt.

Đề tài được thực hiện trong thời gian 3 tháng từ tháng 05-2023 đến tháng 09-2023.

Một phần quan trọng là xử lý dữ liệu tiếng nói tiếng Việt để sử dụng trong quá trình luyện tập và đánh giá mô hình. Đánh giá hiệu suất của mô hình Speech-to-Text cho tiếng Việt là thứ không thể thiếu.

Việc nghiên cứu và phát triển mô hình wav2vec cho bài toán speech-to-text tiếng Việt có ý nghĩa quan trọng trong việc phát triển các ứng dụng xử lý ngôn ngữ tự nhiên tiếng Việt. Mô hình wav2vec có thể được sử dụng để cải thiện độ chính xác của các ứng dụng này, giúp chúng trở nên hữu ích và tiện lợi hơn cho người dùng.

1.5. Bố cục báo cáo

Với mục đích nghiên cứu về Speech Recognition, tìm hiểu các phương pháp, hướng tiếp cận và ứng dụng của nó trong thực tế. Hơn thế nữa là xây dựng một hệ

thống có thể nhận diện tiếng nói và chuyển giọng nói thành văn bản với hiệu suất cao. Báo cáo sẽ bao gồm các chương:

Chương 1: Tổng quan đề tài. Trình bày tổng quan chi tiết về các lý thuyết và nghiên cứu trước đó có liên quan đến lĩnh vực Speech Recognition và đề tài Xây dựng mô hình Speech-to-Text trên nền tảng ngôn ngữ Tiếng Việt. Sau đó, đặt ra vấn đề cụ thể nghiên cứu đang hướng đến giải quyết và đề xuất một phương hướng giải quyết thích hợp.

Chương 2: Cơ sở lý thuyết. Phân tích chi tiết hơn về lĩnh vực Speech Recognition. Cụ thể hơn là Speech-to-Text. Giải thích rõ lý do hình thành, tiềm năng phát triển, các phương pháp và ứng dụng thực tế, nêu ưu điểm và những khó khăn thách thức của lĩnh vực này. Cuối cùng, đào sâu vào phương pháp được đề xuất để xây dựng mô hình STR này – Wav2vec 2.0.

Chương 3: Xây dựng mô hình. Tại đây, sẽ là mô tả chi tiết về mô hình được áp dụng, thực hiện thực nghiệm, và thực hiện mô phỏng trong quá trình nghiên cứu..

Chương 4: Thực nghiệm và đánh giá kết quả. Chúng ta sẽ tập trung vào cả quá trình giải quyết vấn đề cụ thể và phân tích kết quả thu được, bao gồm những nhận xét và đánh giá chi tiết về kết quả này

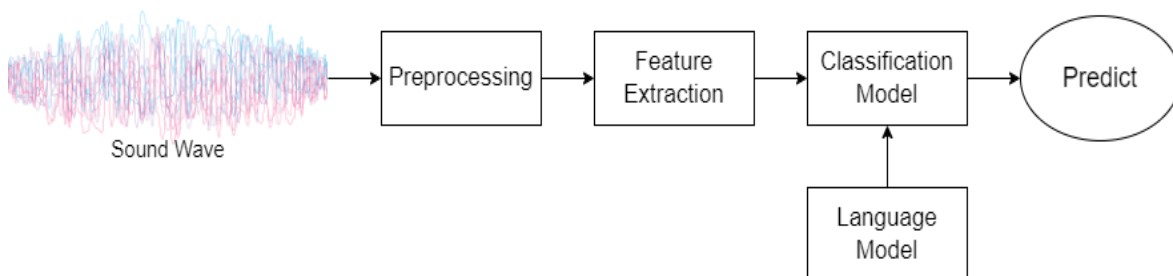
Chương 5: Kết luận và hướng phát triển. Dựa trên mô hình vật lý, thực nghiệm, cài đặt, và mô phỏng được trình bày ở trên, chúng tôi có thể nêu ra một loạt kết luận chung quan trọng, khẳng định những kết quả đạt được, và đề xuất những đóng góp mới cùng với các kiến nghị phát triển trong tương lai.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1. Speech Recognition

Nói sơ qua về nhận dạng giọng nói, Speech Recognition (SR) là một thuật ngữ chung cho công nghệ nhận dạng giọng nói, trong khi Speech-to-Text là một thuật ngữ cụ thể hơn cho quá trình chuyển đổi giọng nói thành văn bản. Công nghệ Speech Recognition có thể được sử dụng cho nhiều mục đích khác nhau, chẳng hạn như tạo bản ghi âm, phiên âm, hoặc điều khiển thiết bị bằng giọng nói. Tuy nhiên, trong tất cả các trường hợp này, mục tiêu cuối cùng là chuyển đổi giọng nói thành văn bản. Vì vậy, có thể nói rằng Speech Recognition là Speech-to-Text. Tuy nhiên, Speech-to-Text là một thuật ngữ cụ thể hơn, chỉ đề cập đến quá trình chuyển đổi giọng nói thành văn bản.

Mặc dù vậy, có thể thấy rõ Speech Recognition là một lĩnh vực rộng hơn, trong đó có chứa Speech-to-Text Recognition (STR) và những thuật ngữ con khác như Automatic Speech Recognition (ASR), Text-to-Speech hay Speech-to-Speech...



Hình 1: Kiến trúc chung của các mô hình nhận dạng âm thanh

Nhiệm vụ của ASR là lấy sóng âm thanh đầu vào và chuyển nó thành dạng văn bản, đầu vào có thể được lấy trực tiếp bằng micrô hoặc dưới dạng tệp âm thanh. Vấn đề này có thể được giải thích theo cách sau: đối với một chuỗi đầu vào X cho trước, trong đó $X = X_1, X_2, \dots, X_n$, với n là độ dài của chuỗi đầu vào, ASR sẽ tìm một chuỗi đầu ra Y tương ứng với độ dài là m , $Y = Y_1, Y_2, \dots, Y_m$. Và chuỗi đầu ra Y có xác suất hậu nghiệm cao nhất $P(Y|X)$, trong đó $P(Y|X)$ có thể được tính bằng công thức:

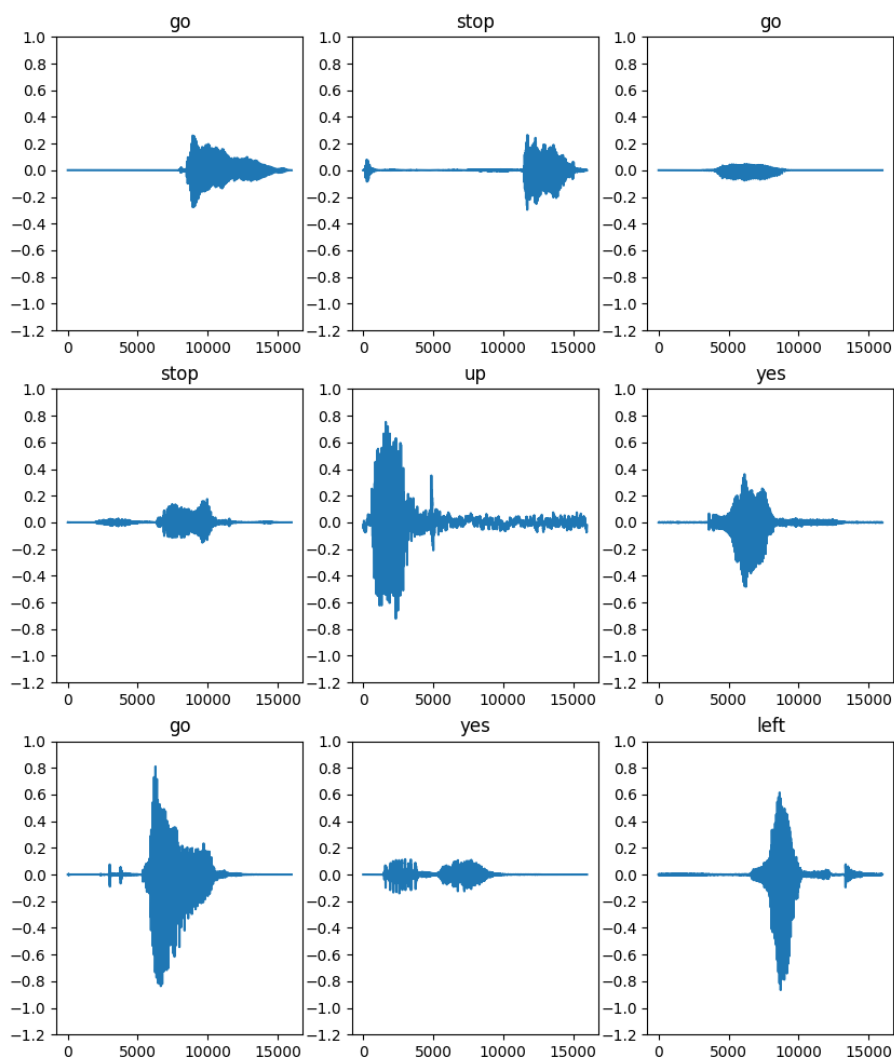
$$W = \operatorname{argmax} P(W|X) = \operatorname{argmax} \frac{P(W)P(X|W)}{P(X)}$$

Trong đó $P(W)$ là xác suất xuất hiện của từ, $P(X)$ là xác suất X có mặt trong tín hiệu và $P(X|W)$ là xác suất xuất hiện tín hiệu âm thanh W tương ứng với từ X [5].

ASR thường có thể được phân chia thành bốn phần chính: tiền xử lý, trích xuất đặc trưng, mô hình phân loại và mô hình ngôn ngữ, như được thể hiện trong Hình #.

2.1.1. Preprocessing

Tín hiệu giọng nói là tín hiệu thay đổi chậm theo thời gian theo nghĩa là khi được kiểm tra trong một khoảng thời gian đủ ngắn (từ 5 đến 100 mili giây), các đặc tính của nó khá ổn định; tuy nhiên, trong khoảng thời gian dài (khoảng 1/5 giây trở lên), các đặc tính tín hiệu thay đổi để phản ánh các âm thanh giọng nói khác nhau được nói. Thông thường, 200 mili giây đầu tiên trở lên (1600 mẫu nếu tốc độ lấy mẫu là 8000 mẫu/giây) của bản ghi giọng nói sẽ tương ứng với khoảng im lặng hoặc tiếng ồn xung quanh vì người nói cần một chút thời gian để đọc khi bắt đầu ghi [6]. Tương tự đối với những mili giây cuối của chuỗi âm thanh.



Hình 2: Ví dụ về một vài dữ liệu âm thanh

Thông thường, dữ liệu đầu vào cho ASR được thu thập thông qua microphone. Điều này đồng nghĩa rằng tiếng ồn có thể kèm theo bên trong dữ liệu âm thanh. Mục tiêu của việc tiền xử lý âm thanh là giảm thiểu tín hiệu so với nhiễu [7]. Có nhiều bộ lọc và phương pháp khác nhau có thể được áp dụng để giảm tiếng ồn trong tín hiệu. Các phương pháp như Framing, normalization [8], end-point detection và pre-emphasis thường được sử dụng để giảm nhiễu trong tín hiệu [6]. Các phương pháp tiền xử lý cũng thay đổi tùy theo thuật toán được sử dụng để trích xuất đặc trưng. Một số thuật toán trích xuất đặc trưng cụ thể yêu cầu việc tiền xử lý đặc biệt được áp dụng cho dữ liệu âm thanh đầu vào của chúng.

2.1.2. Feature Extraction

Sau khi qua giai đoạn tiền xử lý, tín hiệu giọng nói “sạch” sẽ được đưa vào quá trình trích xuất đặc trưng. Hiệu suất và hiệu quả của giai đoạn phân loại phụ thuộc nhiều vào các đặc trưng được trích xuất [9,10]. Có nhiều phương pháp khác nhau để trích xuất các đặc trưng từ tín hiệu tiếng nói. Thông thường, các đặc trưng được biểu diễn dưới dạng hệ số hoặc giá trị được tính toán trước đó thông qua việc áp dụng các phương pháp đa dạng lên tín hiệu giọng nói đầu vào. Mô-đun trích xuất tính năng cần phải đối mặt với các yếu tố biến đổi khác nhau, như hiệu ứng tiếng ồn và tiếng vang. Các phương pháp phổ biến trong trích xuất đặc điểm bao gồm hệ số Mel Frequency cepstral coefficients (MFCC) [11], linear predictive coding (LPC) [12] và discrete wavelet transform (DWT) [9,10, 12].

Mel Frequency cepstral coefficients

Một trong những phương pháp trích xuất đặc trưng phổ biến nhất trong ASR là phương pháp MFCC. Quá trình này bắt đầu bằng việc tính toán phổ FFT hoặc LPC cho từng khung dữ liệu giọng nói, sau đó thực hiện các bước tiếp theo. Đầu tiên, biên độ phổ được chuyển đổi thành đơn vị decibel và loại bỏ pha phổ. Tiếp theo, một tập hợp các bộ lọc hình tam giác được áp dụng, và chúng được đặt cách nhau dựa trên trọng số thang đo cảm nhận mel. Cuối cùng, một FFT nghịch đảo được thực hiện để tạo ra vector phổ cuối cùng [13].

Ưu điểm của phương pháp này là không cần phải đưa ra những quyết định phức tạp để xác định các đặc trưng, chẳng hạn như việc ước tính định dạng hoặc tần số cơ bản (F0), mà thường có thể gây ra sai lệch. Kết quả của ASR sử dụng MFCC thường

cho thấy hiệu suất tốt hơn so với các phương pháp khác như LPC đơn giản hoặc sử dụng ngân hàng bộ lọc thông thường. Các MFCC có thể hiểu như là các đặc trưng gần như không tương quan, nhờ sử dụng FFT nghịch đảo với các hàm cơ sở hình sin trực giao.

Discrete wavelet transform

DWT là một phép biến đổi rời rạc, nghĩa là nó hoạt động trên các tín hiệu thời gian rời rạc. DWT phân tích một tín hiệu thời gian rời rạc thành một số thành phần tần số và thời gian. Các thành phần này được gọi là wavelet coefficients. DWT được thực hiện bằng cách nhân tín hiệu thời gian rời rạc với một bộ sóng. Bộ sóng là một hàm thời gian rời rạc có các đặc điểm được xác định trước. Sản phẩm của phép nhân này được gọi là wavelet transform. Wavelet transform là một tập hợp các wavelet coefficients. Các wavelet coefficients có thể được sử dụng để tái tạo tín hiệu thời gian rời rạc ban đầu. DWT là một công cụ mạnh mẽ có thể được sử dụng trong nhiều ứng dụng khác nhau.

Linear predictive coding

Lời nói của con người được tạo ra trong đường thanh âm có thể được coi gần giống như một ống có đường kính thay đổi. Mô hình mã hóa dự đoán tuyến tính (LPC) dựa trên phép tính gần đúng về mặt toán học của đường phát âm được biểu thị bằng ống có đường kính khác nhau này. Tại một thời điểm cụ thể, t , mẫu giọng nói $s(t)$ được biểu diễn dưới dạng tổng tuyến tính của p mẫu trước đó. Khía cạnh quan trọng nhất của LPC là bộ lọc dự đoán tuyến tính cho phép xác định giá trị của mẫu tiếp theo bằng sự kết hợp tuyến tính của các mẫu trước đó.

Ngoài ra, còn một số phương pháp trích xuất đặc trưng khác như PCA, ICA... ít được sử dụng hơn hay được sử dụng làm nền tảng để phát triển một phương pháp mới.

2.1.3. Classification Model

Phần quan trọng nhất trong các bài toán hay dự án STR là mô hình phân loại, nó được sử dụng để dự đoán văn bản tương ứng với tín hiệu giọng nói đầu vào. Các mô hình phân loại sử dụng đầu vào từ các đặc trưng đã được trích xuất từ giai đoạn trước đó để dự đoán văn bản. Có nhiều phương pháp tiếp cận khác nhau để thực hiện nhiệm vụ nhận dạng giọng nói.

Mục tiêu của quá trình phân loại âm thanh là dự đoán nhãn lớp cho dữ liệu âm thanh đầu vào. Mô hình có thể dự đoán một nhãn lớp duy nhất cho toàn bộ dãy dữ liệu âm thanh hoặc có thể dự đoán nhãn cho từng khung âm thanh riêng lẻ, thường là mỗi khung có độ dài khoảng 20 mili giây. Trong trường hợp này, đầu ra của mô hình là một chuỗi xác suất tương ứng với các nhãn lớp.

Phương pháp đầu tiên sử dụng phân phối xác suất chung, được hình thành thông qua tập dữ liệu huấn luyện, và sau đó sử dụng phân phối xác suất này để dự đoán đầu ra trong tương lai. Cách tiếp cận này được gọi là cách tiếp cận sáng tạo, với mô hình HMM (Hidden Markov Model) [14] và Gaussian mixture models (GMM) [15] là những mô hình phổ biến dựa trên phương pháp này.

Cách tiếp cận thứ hai là tính toán mô hình tham số bằng cách sử dụng tập dữ liệu huấn luyện với các cặp vector đầu vào và đầu ra tương ứng. Cách tiếp cận này được gọi là cách tiếp cận phân biệt đối xử, với các ví dụ như Support Vector Machine (SVM) [16] và Mạng nơ-ron nhân tạo (ANN) [17] là những phương pháp phổ biến trong loại này.

Ngoài ra, có thể sử dụng các phương pháp kết hợp, như việc kết hợp HMM và ANN, để đạt được hiệu suất tốt hơn trong quá trình phân loại [18]. Các phương pháp này sẽ được trình bày chi tiết hơn trong phần sau của báo cáo.

2.1.4. Language Model

Mô hình ngôn ngữ bao gồm nhiều loại quy tắc và cấu trúc ngôn ngữ. Chúng là yếu tố quan trọng trong quá trình nhận dạng âm vị mà bộ phân loại dự đoán và cũng được sử dụng để hình thành các từ và câu bằng cách kết hợp tất cả các âm vị dự đoán từ dữ liệu đầu vào cụ thể. Mặc dù hầu hết các hệ thống nhận dạng giọng nói hiện đại có thể hoạt động mà không cần mô hình ngôn ngữ, nhưng hiệu suất của chúng có thể đạt được sự cải thiện đáng kể khi sử dụng mô hình ngôn ngữ [19].

Trước những năm 1980, ASR chỉ sử dụng thông tin âm thanh để đánh giá các giả thuyết văn bản. Tuy nhiên tại thời điểm đó các mô hình học sâu với khả năng học tập và cho hiệu suất cao chưa phổ biến, sau đó, nhận thấy rằng việc kết hợp kiến thức về văn bản bằng cách khai thác thông tin thừa trong nó sẽ làm tăng đáng kể độ chính xác của ASR. Lời nói thường tuân theo các quy tắc ngôn ngữ như cú pháp và ngữ nghĩa. Tuy nhiên, đôi khi, lời nói chỉ là một chuỗi các từ ngẫu nhiên được rút ra từ một tập

Một cách nói dễ hiểu hơn, LM sử dụng văn phạm phụ thuộc để tối ưu mô hình. Nó dựa trên ý tưởng rằng mỗi từ trong một câu có một vai trò cụ thể, và các từ này phụ thuộc lẫn nhau để tạo thành một câu có ý nghĩa. Language model sử dụng văn phạm phụ thuộc để hiểu mối quan hệ giữa các từ trong một câu. Điều này giúp chúng tạo ra văn bản có ý nghĩa và mạch lạc. Hình # có thể giải thích cụ thể việc sử dụng văn phạm trong nó.

STT	Language Model	Năm
1	N-gram model	1981
2	Cache-based model	1990
3	Class-based models	1990
4	Maximum Entropy model	1992
5	Trigger-based model	1993
6	Skipping model	1993
7	Structured model	1993
8	Mixed order markov model	1997
9	Aggregate Language model	1997
10	Mixture-Based Language model	1997
11	Latent semantic analysis	1997
12	Probability latent semantic analysis	1999
13	Neural probabilistic language model	2000
14	Latent maximum entropy model	2001
15	Latent Dirichlet Allocation	2003
16	Gaussian mixture language model	2007
17	Word Topic model	2007
18	Continuous topic language model	2008
19	Word Vicinity mode	2009
20	Large Language Models	2019

Bảng 1: Một số mô hình ngôn ngữ [20]

2.2. Bối cảnh

2.2.1. Lịch sử hình thành

Trải qua một thời kỳ kéo dài, cộng đồng các nhà nghiên cứu trong lĩnh vực khoa học máy tính đã nỗ lực để tạo ra một máy tính có khả năng tương tác và trò chuyện như con người. Họ đã đặt nỗ lực vào việc làm cho máy tính có khả năng hiểu, giải thích và tái tạo ngôn ngữ và giọng điệu của con người [20] trong những năm đầu tiên.

Cột mốc đầu tiên trong việc nhận dạng giọng nói là hệ thống có tên Audrey, phát triển tại Phòng thí nghiệm Bell năm 1952, có khả năng phân biệt các chữ số khác nhau nói bởi người dùng [5]. Tại Phòng thí nghiệm MIT Lincoln vào năm 1959, họ đã phát triển một hệ thống khác có khả năng phân biệt 10 âm vị mà một người nói [21] sử dụng. Các phương pháp được sử dụng trong giai đoạn này chủ yếu dựa trên các kỹ thuật xử lý tín hiệu truyền thống, như phân tích tần số và phân tích thời gian. Tuy nhiên, độ chính xác của các hệ thống nhận dạng giọng nói trong giai đoạn này còn rất hạn chế, chỉ khoảng 50%.

Trong thập kỷ 1970, lĩnh vực nhận dạng giọng nói đã chứng kiến nhiều nghiên cứu quan trọng. Các nhà nghiên cứu ở Nga đã phát triển một hệ thống có khả năng phân biệt các từ [22]. Ý tưởng sử dụng quy hoạch động [23] và thuật toán nhận dạng mẫu [22] cũng đã được trình bày trong giai đoạn này. Vào đầu những năm 1980, Hidden Markov Model (HMM) đã được giới thiệu [14]. Mặc dù HMM được xem là quá đơn giản để định rõ ngôn ngữ của con người, nhưng chúng đã thay thế được kỹ thuật sử dụng trong việc xử lý thời gian động [5]. Cuối thập kỷ 1980, mô hình n-gram đã được giới thiệu [5].

Trong những năm đầu của thế kỷ 21, HMM đã được kết hợp với Mạng Nơ-ron Nhân Tạo (ANN) để tạo nên sự tiến bộ trong lĩnh vực này [18]. Hiện nay, Long Short-Term Memory (LSTM), một loại Mạng Nơ-ron Hồi quy (RNN), được sử dụng rộng rãi để nhận dạng giọng nói, kết hợp với các kỹ thuật học sâu đa dạng khác [24]. Giai đoạn này, sự phát triển của các kỹ thuật học sâu đã mang lại một bước đột phá cho công nghệ nhận dạng giọng nói. Các hệ thống SR dựa trên học sâu có thể học được các đặc trưng của giọng nói từ một lượng dữ liệu khổng lồ, giúp cải thiện độ chính xác lên đáng kể. Tiêu biểu trong giai đoạn này là nhóm nghiên cứu [25] của GS. Geoffrey Hinton năm 2012 đã phát triển một mô hình mạng nơ-ron sâu có thể nhận

dạng giọng nói với độ chính xác cao và vào năm 2016, TS. Lê Viết Quốc cùng với nhóm nghiên cứu đã phát triển một mô hình mạng nơ-ron sâu có thể nhận dạng giọng nói tốt nhất thời điểm hiện tại [16].

Ở giai đoạn hiện nay, không thể không nhắc đến Transformer, một mô hình “new-state-of-art” ở thời điểm hiện tại. Transformer cho bài toán SR là một mô hình mạng nơ-ron sâu được phát triển vào năm 2018 [26]. Mô hình này đã đạt được những thành tựu đáng kể trong nhiều lĩnh vực, bao gồm cả nhận dạng giọng nói. Trong giai đoạn hiện nay, Transformer là một trong những mô hình được sử dụng phổ biến nhất trong nhận dạng giọng nói. Các mô hình Transformer có thể đạt được độ chính xác cao hơn đáng kể so với các mô hình dựa trên mạng nơ-ron truyền thống.

2.2.2. Thách thức

Một khác biệt quan trọng giữa hệ thống giao tiếp của con người và hệ thống nhân tạo nằm ở sự không đối xứng giữa máy phát và máy thu đối với tín hiệu giọng nói [13]. Trong nhiều hệ thống truyền thông, thường cần thiết thiết kế một bộ thu để đảo ngược toàn bộ quá trình xử lý được thực hiện tại bộ mã hóa. Tuy nhiên, ở con người, thính giác đã phát triển từ rất lâu trước khi khả năng nói xuất hiện; thực tế, hệ thống thính giác của hầu hết các loài động vật có sự tương đồng, trong khi chỉ có con người mới có khả năng nói.

Hệ thống thính giác có thể đã tiến hóa để phòng ngừa các nguy cơ và nguy hiểm. Sau đó, con người đã nhận ra tiềm năng của việc sử dụng âm thanh phức tạp từ hệ thống tiếng nói của họ để truyền đạt thông tin hữu ích. Sự khác biệt lớn về cấu trúc và chức năng giữa hệ thống tạo ra tiếng nói và hệ thống nhận thức trong con người đã làm cho việc phân tích nó trở nên phức tạp. Con người đã học cách sử dụng các cơ quan như lưỡi, môi, hàm và dây thanh âm, kết hợp với hơi thở, để truyền đạt những ý tưởng phức tạp một cách hiệu quả. Tuy nhiên, vì cơ quan tiếng nói chia sẻ chức năng với việc thở và ăn uống, nên chúng không được tạo ra lý tưởng để sản xuất âm thanh phức tạp theo ý muốn cho tai nghe.

Ngoài ra, nhận dạng giọng nói cũng có thể bị ảnh hưởng bởi các yếu tố khác, như môi trường, chất lượng âm thanh và độ dài của đoạn âm thanh cần nhận dạng. Tiếng ồn có thể làm giảm độ chính xác của các hệ thống nhận dạng giọng nói. Các hệ thống này cần phải được thiết kế để có thể loại bỏ tiếng ồn và nhận dạng giọng nói trong

môi trường ồn ào. Các trọng âm và phương ngữ khác nhau có thể khiến các hệ thống nhận dạng giọng nói gặp khó khăn trong việc nhận dạng chính xác. Các hệ thống này cần được đào tạo trên một tập dữ liệu lớn bao gồm các giọng nói từ nhiều nguồn khác nhau. Các từ hiếm và cụm từ có thể không được bao gồm trong tập dữ liệu đào tạo của các hệ thống nhận dạng giọng nói (out-of-vocabulary). Các hệ thống này cần có khả năng học các từ và cụm từ mới để cải thiện độ chính xác. Các hệ thống nhận dạng giọng nói cần có khả năng nhận dạng các giọng nói khác nhau, bao gồm cả giọng nói của trẻ em, người già và người có khuyết tật.

Sử dụng thuật toán học sâu là một trong những phương pháp quan trọng nhất, cho phép hệ thống học các đặc trưng của giọng nói một cách hiệu quả và cải thiện đáng kể độ chính xác của quá trình nhận dạng. Bên cạnh đó, việc áp dụng các bộ lọc âm thanh có khả năng loại bỏ tiếng ồn và tối ưu hóa chất lượng âm thanh có thể giúp hệ thống làm việc tốt hơn trong môi trường ồn ào.

Công nghệ thực tế ảo cũng đang được kết hợp vào các hệ thống nhận dạng giọng nói để cải thiện hiểu biết của hệ thống về môi trường xung quanh, từ đó tăng độ chính xác trong các tình huống phức tạp. Với sự tiến bộ liên tục của công nghệ, các thách thức liên quan đến nhận dạng giọng nói dự kiến sẽ ngày càng được giải quyết, khiến cho công nghệ này trở nên phổ biến và hữu ích hơn trong nhiều lĩnh vực.

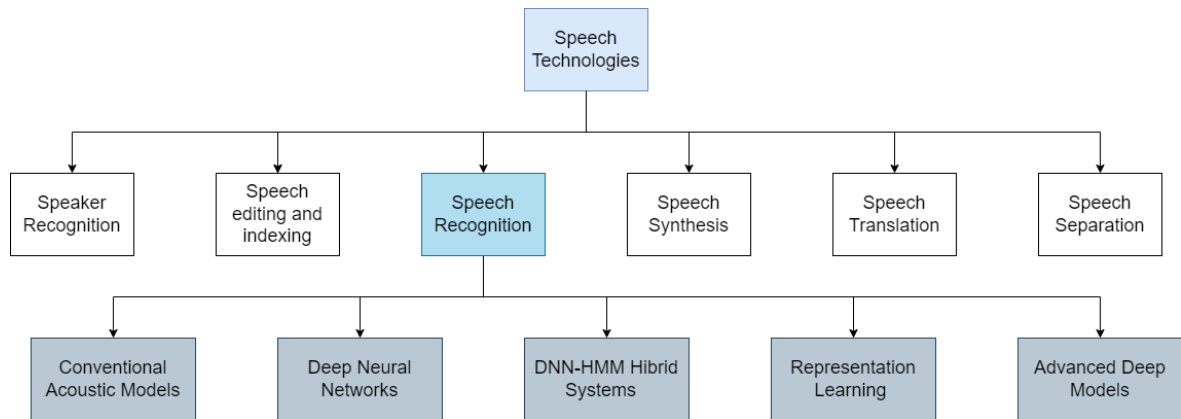
2.3. Các phương pháp

2.3.1. Hướng tiếp cận

Có hai hướng tiếp cận chính cho các bài toán tính hiệu âm thanh: SR với phổ tần số thời gian [27] và không cần phải chuyển về phổ.

Nhận dạng giọng nói bằng quang phổ là một phương pháp nhận dạng giọng nói sử dụng quang phổ để phân tích giọng nói. Phương pháp này dựa trên nguyên tắc rằng mỗi âm thanh có một phổ tần số đặc trưng. Trong phương pháp này, âm thanh được thu nhận bằng một microphone và sau đó được phân tích và trả về một phổ tần số thời gian. Các thành phần tần số này sau đó được sử dụng để xác định âm thanh. Hướng tiếp cận này sẽ chuyển một bài toán xử lý âm thanh về bài toán xử lý ảnh quen thuộc hơn.

Về các phương pháp giải quyết bài toán SR sẽ bao gồm: các mô hình xử lý âm thanh thông thường (Conventional Acoustic Models), các mạng neural học sâu (Deep Neural Networks), các hệ thống DNN-HMM kết hợp (DNN-HMM Hybrid Systems), học đại diện (Representation Learning) và các mô hình học sâu nâng cao (Advanced Deep Models).



Hình 4: Các công nghệ nhận dạng âm thanh

2.3.2. Conventional Acoustic Models

Tiêu biểu trong các phương pháp này là Gaussian mixture models (GMM) [15] và Hidden Markov Model (HMM) [14].

Gaussian mixture models

GMM với tư cách là mô hình thống kê cho các đặc điểm giọng nói dựa trên phổ Fourier đóng vai trò quan trọng trong mô hình âm thanh của các hệ thống nhận dạng giọng nói thông thường. Người ta thường nghĩ mô hình hỗn hợp Gaussian và ước lượng tham số liên quan là một vấn đề thiếu dữ liệu. Để hiểu điều này, chúng ta hãy giả sử rằng các điểm dữ liệu đang được xem xét có “tư cách thành viên” hoặc thành phần của hỗn hợp, thuộc một trong các phân bố Gaussian riêng lẻ mà chúng ta đang sử dụng để lập mô hình dữ liệu. Khi bắt đầu, tư cách thành viên này không xác định hoặc bị thiếu. Nhiệm vụ của ước lượng tham số là tìm hiểu các tham số thích hợp cho phân bố, với kết nối tới các điểm dữ liệu được biểu diễn dưới dạng thành viên của chúng trong các phân bố Gaussian riêng lẻ [28].

Khi dạng sóng giọng nói được xử lý thành dạng nén (ví dụ: bằng cách lấy logarit của) cường độ biến đổi Fourier thời gian ngắn hoặc cesstra liên quan, phân bố hỗn hợp Gaussian được thảo luận ở trên được chứng minh là khá phù hợp để phù hợp với

các đặc điểm giọng nói như vậy khi thông tin về thứ tự thời gian bị loại bỏ [28]. Nghĩa là, người ta có thể sử dụng phân bố hỗn hợp Gaussian làm mô hình để biểu diễn các đặc điểm giọng nói dựa trên khung.

GMM (Gaussian Mixture Model) thường được áp dụng để mô hình hóa dữ liệu và thực hiện phân loại dựa trên thống kê. GMM nổi tiếng với khả năng biểu diễn các phân phối phức tạp một cách linh hoạt, cho phép nó mô tả các phân phối dữ liệu với nhiều chế độ khác nhau. Các hệ thống phân loại dựa trên GMM thường hiệu quả khi được áp dụng trong lĩnh vực nghiên cứu giọng nói, đặc biệt là trong việc nhận dạng người nói, xử lý nhiễu và phân loại giọng nói.

Để thực hiện nhận dạng người nói, GMM thường được sử dụng trực tiếp làm mô hình nền phổ quát (UBM - Universal Background Model) để mô phỏng đặc điểm giọng nói được tổng hợp từ tất cả các người nói [28-30]. Trong các ứng dụng như loại bỏ nhiễu hoặc theo dõi tiếng ồn, GMM cũng được áp dụng tương tự và được sử dụng như một phân phối tham chiếu [31].

Hidden Markov Model

Mạng phổ biến nhất cho ASR là mạng Markov bậc nhất hoặc chuỗi Markov bậc nhất, trong đó khả năng ở một trạng thái nhất định chỉ phụ thuộc vào trạng thái ngay trước đó. Vì các âm vị có thứ tự thời gian rõ ràng trong từ, nên mạng này chỉ cho phép chuyển từ trạng thái trước sang trạng thái sau. Do đó, các trạng thái mô hình được sắp xếp theo thứ tự, với các trạng thái ban đầu, trung gian và cuối tương ứng với phần đầu, phần giữa và phần cuối của đơn vị phát ngôn được mô hình hóa. Loại mạng này thường được gọi là mạng Markov ẩn (Hidden Markov Model) vì việc xây dựng các mô hình phụ thuộc vào dự đoán từ các quan sát đầu ra của giọng nói, thay vì dựa trên các quan sát trực tiếp về cấu trúc của bộ phận tạo ra giọng nói [13].

Các trạng thái trong mạng HMM thường không có sự liên kết rõ ràng với các sự kiện âm thanh cụ thể. Thay vào đó, người ta thường lựa chọn số lượng trạng thái sao cho mỗi phân đoạn âm thanh riêng biệt trong đơn vị giọng nói có thể được xấp xỉ bằng một trạng thái. Ví dụ, một mô hình HMM cho một âm vị có thể sử dụng ba trạng thái: một trạng thái đầu tiên biểu diễn phổ chuyển tiếp từ âm vị trước, một trạng thái ổn định biểu diễn phổ âm thanh của chính âm vị đó, và một trạng thái cuối cùng biểu diễn phổ chuyển tiếp sang âm vị tiếp theo.

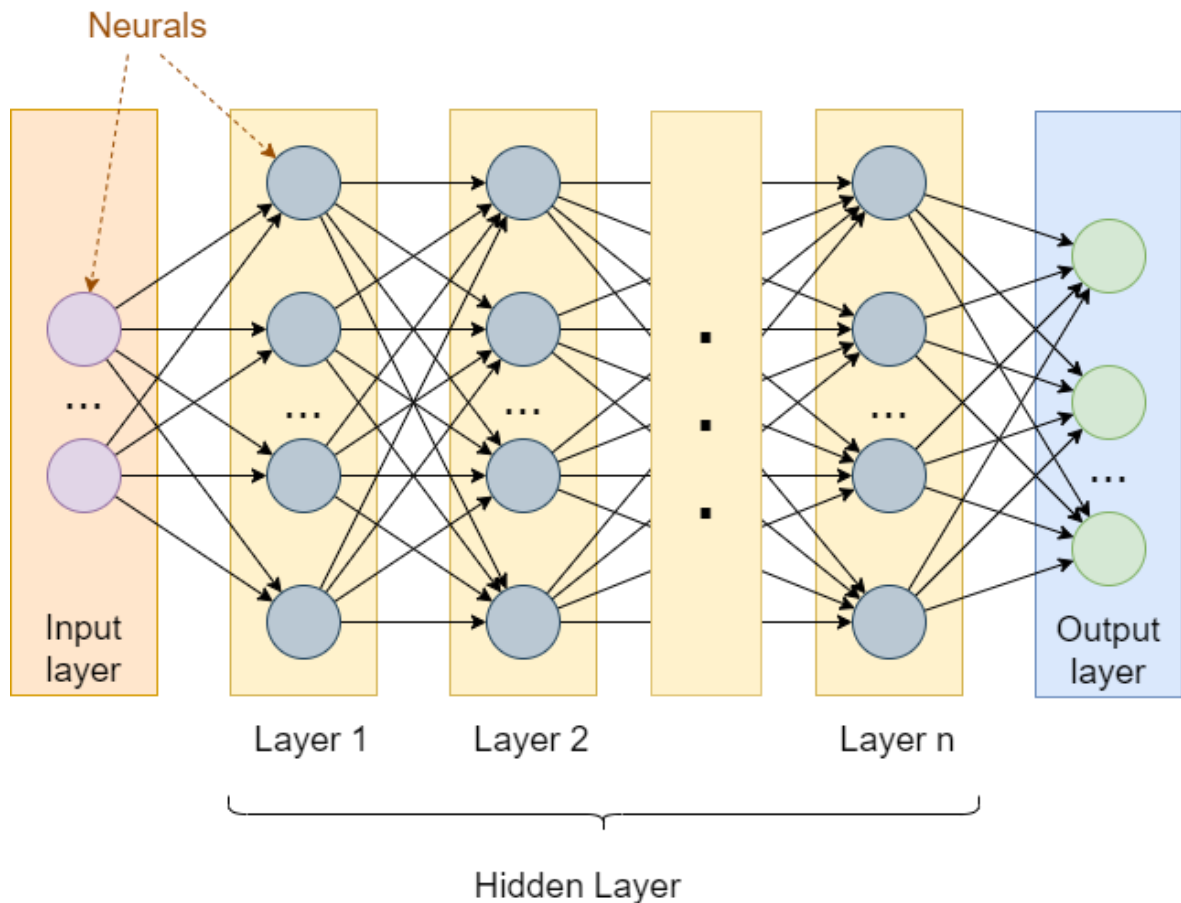
Để giải thích sự thay đổi trong giọng nói, ví dụ như cách phát âm, tốc độ nói, hoặc giọng điệu khác nhau của các người nói, mỗi trạng thái được biểu thị bằng một bản PDF (Probability Density Functions) quang phổ chứ không phải một phổ đơn lẻ [13].

Các tham số trong mỗi HMM được huấn luyện thông qua ước tính ban đầu và sau đó là ước tính lại lặp đi lặp lại. Một phương pháp tiêu chuẩn là thuật toán expectation – maximization [32], một quy trình tương tự như quy trình thiết kế sách mã lượng tử hóa vector. Điều này sử dụng phương pháp Gradient hoặc leo đồi, mang lại mô hình tối ưu cục bộ. Vector âm thanh nguyên mẫu ban đầu cho mỗi trạng thái phát triển lặp đi lặp lại thành một tập PDF đầy đủ, thông qua quy trình lấy trung bình hoặc phân cụm liên quan đến phần còn lại của dữ liệu huấn luyện. Thông thường, vector ban đầu cung cấp ước tính đầu tiên về giá trị trung bình của PDF. Các quy trình lấy trung bình phổ biến nhất là các biến thể của thuật toán Viterbi [33] (ví dụ: phương pháp Gradient), thuật toán Baum – Welch [34] và thuật Forward – Backward [35].

2.3.3. Deep Neural Networks

Mạng nơ-ron sâu (DNN) là một perceptron đa lớp (MLP) thông thường với nhiều lớp ẩn. Mạng nơ-ron học sâu thường được chia thành ba thành phần chính để thực hiện các tính toán: Input Layer, hidden Layers, Output Layer. Mỗi lớp ẩn bao gồm một số lượng nơ-ron. Mỗi nơ-ron trong mạng nhận đầu vào từ các nơ-ron trong lớp trước và sử dụng một hàm kích hoạt để biến đổi đầu vào thành đầu ra. Các hàm kích hoạt có vai trò quan trọng trong quá trình này và thường được chọn dựa trên yêu cầu của bài toán cụ thể. Một số hàm kích hoạt phổ biến bao gồm hàm sigmoid, hàm ReLU, và hàm leaky ReLU, mỗi loại có cách thức biến đổi đầu vào riêng để tạo ra đầu ra.

Lớp đầu vào chịu trách nhiệm nhận dữ liệu đầu vào, đóng vai trò như cửa sổ chính để thông tin từ thế giới bên ngoài được đưa vào mạng. Các lớp ẩn là nơi thực hiện các tính toán phức tạp để biến đổi dữ liệu đầu vào thành các biểu diễn tương đối phức tạp hơn. Mỗi lớp ẩn bao gồm một tập hợp các nơ-ron, và mỗi nơ-ron trong lớp này nhận đầu vào từ các nơ-ron trong lớp trước đó và tạo ra đầu ra cho các nơ-ron trong lớp tiếp theo. Điều này tạo ra một mạng liên kết phức tạp của các tính toán. Ớp đầu ra là phần cuối cùng của mạng và tạo ra đầu ra của toàn bộ mô hình. Thông qua quá trình tính toán trong các lớp ẩn, thông tin đã được biến đổi và tinh chỉnh để tạo ra đầu ra mong muốn.



Hình 5: Kiến trúc Deep Neural Network

Bởi vì mỗi lớp trong mạng có thể được coi như một bộ trích xuất đặc trưng của lớp trước đó, việc lựa chọn kích thước của mỗi lớp trong mạng là một quá trình quan trọng. Số lượng nơ-ron trong mỗi lớp phải đủ lớn để có khả năng nắm bắt được các mẫu quan trọng trong dữ liệu. Điều này đặc biệt quan trọng đối với các lớp dưới cùng của mạng, bởi vì các tính năng ở lớp đầu tiên thường có sự biến đổi lớn và đòi hỏi nhiều nơ-ron hơn để mô hình hóa chúng so với các lớp sau.

Tuy nhiên, nếu kích thước của mỗi lớp quá lớn, mạng có thể dễ dàng bị Overfitting với dữ liệu huấn luyện, điều này có nghĩa là nó sẽ học nhớ dữ liệu huấn luyện thay vì học cách tổng quát hóa và áp dụng cho dữ liệu mới. Về cơ bản, mạng "rộng" (có nhiều neural) và "nông" (ít lớp ẩn) có thể dễ dàng dẫn đến tình trạng Overfitting, trong khi mạng "sâu" (nhiều lớp ẩn) và "hẹp" (ít neural) có thể dẫn đến Underfitting.

Trong thực tế, một cách tiếp cận thường được sử dụng là tối ưu hóa số lượng neural trong mỗi lớp của mạng một lớp ẩn. Sau đó, nhiều lớp hơn có thể được xếp chồng lên với cùng kích thước lớp ẩn để tạo ra một mạng sâu hơn. Trong các nhiệm vụ

vụ nhận dạng giọng nói, thường thấy rằng sử dụng khoảng 5-7 lớp mạng neural học sâu với 1.000-3.000 neural ở mỗi lớp hoạt động hiệu quả. Việc tìm kiếm cấu hình tốt thường dễ dàng hơn trên mạng rộng và sâu hơn, chứ không phải trên mạng hẹp và nông. Điều này xuất phát từ sự có nhiều tối ưu cục bộ tốt hoạt động tương tự trên các mạng rộng và sâu.

Cơ chế lan truyền ngược (hay còn gọi là backpropagation) là một thuật toán được sử dụng để đào tạo các mạng neural học sâu. Thuật toán này sử dụng đạo hàm của hàm tổn thất để tính toán Gradient của các trọng số trong mạng. Gradient sau đó được sử dụng bởi thuật toán tối ưu hóa để điều chỉnh các trọng số theo hướng giảm thiểu hàm mất mát.

Đầu tiên, thuật toán tính toán hàm mất mát của mạng trên dữ liệu đào tạo. Hàm mất mát là một hàm đo lường mức độ sai lệch giữa đầu ra của mạng và đầu ra mong muốn. Sau đó, thuật toán tính toán Gradient của hàm mất mát theo các trọng số trong mạng. Gradient là một vector cho biết hướng mà hàm mất mát sẽ thay đổi nếu các trọng số được điều chỉnh. Cuối cùng, thuật toán sử dụng gradient để cập nhật các trọng số trong mạng. Quá trình cập nhật được thực hiện bằng cách sử dụng thuật toán tối ưu hóa.

2.3.4. DNN-HMM Hybrid Systems

Mô hình kết hợp DNN-HMM [18] là một mô hình học máy kết hợp các ưu điểm của mạng neural học sâu DNN và HMM. DNN có thể học các mối quan hệ phi tuyến phức tạp, trong khi HMM có thể mô hình hóa các trạng thái ẩn không quan sát được.

Mô hình DNN-HMM được đào tạo bằng quá trình học máy có giám sát. Trong quá trình này, mô hình được cung cấp một bộ dữ liệu đào tạo bao gồm tín hiệu âm thanh và nhãn trạng thái ẩn. Mô hình sau đó được điều chỉnh để tạo ra nhãn trạng thái ẩn chính xác nhất. Quá trình điều chỉnh mô hình được thực hiện bằng cách sử dụng thuật toán tối ưu hóa. Thuật toán tối ưu hóa sẽ tìm các giá trị của các trọng số trong mô hình sao cho mô hình tạo ra nhãn trạng thái ẩn có độ chính xác cao.

Mô hình DNN-HMM có thể học các mối quan hệ phi tuyến phức tạp từ dữ liệu âm thanh, mô hình hóa các trạng thái ẩn không quan sát được và có thể đạt được độ chính xác cao trong các ứng dụng xử lý âm thanh. Tuy nhiên, nó cũng có một vài

nhược điểm như có thể phức tạp và tốn kém để đào tạo và có thể dễ bị thiên vị trong dữ liệu đào tạo.

2.3.5. Advanced Deep Models

Có một số kỹ thuật quan trọng đã được sử dụng để khởi tạo hoặc huấn luyện trước các mô hình mạng thần kinh sâu (DNN). Những kỹ thuật này đã đóng vai trò quan trọng trong giai đoạn đầu của nghiên cứu về deep learning và vẫn có giá trị trong nhiều tình huống khác nhau. Các kỹ thuật này bao gồm: Restricted Boltzmann Machine (RBM), Deep Belief Network (DBN) [28].

Restricted Boltzmann Machine

RBM [36] là một mạng lưới thần kinh tổng hợp ngẫu nhiên. Đúng như tên gọi của nó, nó là một biến thể của máy Boltzmann. Về cơ bản, nó là một mô hình đồ họa vô hướng được xây dựng từ một lớp neural ngẫu nhiên nhìn thấy được và một lớp neural ẩn ngẫu nhiên. Các neural hiển thị và ẩn tạo thành một biểu đồ lưỡng cực không có kết nối ẩn-ẩn hoặc ẩn-ẩn. Các neural ẩn thường lấy giá trị nhị phân và tuân theo phân phối Bernoulli. Các neural nhìn thấy được có thể nhận giá trị nhị phân hoặc giá trị thực tùy thuộc vào loại đầu vào.

RBM thường được sử dụng cho việc học không giám sát, nhưng chúng cũng có thể được sử dụng cho các nhiệm vụ học có giám sát như phân loại và hồi quy. RBM bao gồm hai lớp đơn vị: lớp hiển thị và lớp ẩn. Lớp hiển thị biểu thị dữ liệu đầu vào và lớp ẩn biểu thị các tính năng tiềm ẩn của dữ liệu đầu vào. Các đơn vị trong lớp hiển thị và lớp ẩn được kết nối với nhau, nhưng các đơn vị trong cùng một lớp không được kết nối.

RBM được đào tạo bằng cách sử dụng một quá trình gọi là phân kỳ tương phản. Trong phân kỳ tương phản, RBM lần đầu tiên được khởi tạo với các trọng số ngẫu nhiên. Sau đó, RBM được cấp một tập dữ liệu đầu vào và được phép lấy mẫu từ phân bố xác suất mà nó đã học được. Dữ liệu được lấy mẫu sau đó được sử dụng để cập nhật trọng số của RBM. Khi RBM đã được huấn luyện, nó có thể được sử dụng để tạo dữ liệu mới hoặc có thể được sử dụng để phân loại hoặc hồi quy dữ liệu mới.

Deep Belief Network. RBM có thể được xem như là một mô hình tổng quát với một số lượng lớp vô hạn, trong đó tất cả các lớp này chia sẻ cùng một ma trận trọng số. Nếu ta loại bỏ lớp dưới cùng khỏi mô hình RBM, các lớp còn lại sẽ hình thành

một mô hình sinh khác, cũng có vô số lớp với ma trận trọng số chung. Các lớp còn lại này tương đương với một RBM khác có lớp hiển thị và lớp ẩn được hoán đổi. Mô hình này được gọi là Deep Belief Network (DBN), trong đó lớp trên cùng là RBM vô hướng và các lớp dưới cùng hình thành một mô hình tạo có hướng [28].

Việc tiền huấn luyện DBN không quan trọng khi chỉ sử dụng một lớp ẩn và nó thường hoạt động tốt nhất với hai lớp ẩn [37, 38]. Khi số lớp ẩn tăng lên thì hiệu quả thường giảm đi. Điều này là do quá trình đào tạo trước DBN sử dụng hai phép tính gần đúng. Đầu tiên, phép tính gần đúng trường trung bình được sử dụng làm mục tiêu tạo khi huấn luyện lớp tiếp theo. Thứ hai, thuật toán phân kỳ tương phản gần đúng được sử dụng để tìm hiểu các tham số mô hình. Cả hai phép tính gần đúng này đều gây ra lỗi mô hình hóa cho mỗi lớp bổ sung. Khi số lượng lớp tăng lên, các lỗi tích hợp sẽ tăng lên và hiệu quả của việc đào tạo trước DBN sẽ giảm đi. Rõ ràng là mặc dù chúng ta vẫn có thể sử dụng mô hình được huấn luyện trước DBN làm mô hình ban đầu cho DNN với các đơn vị tuyến tính được chỉnh lưu, nhưng hiệu quả sẽ bị giảm đi rất nhiều do không có liên kết trực tiếp giữa hai mô hình.

2.3.6. Các phương pháp đánh giá

Việc đánh giá là một phần quan trọng trong quá trình nghiên cứu bởi vì tầm quan trọng của nó không thể bỏ qua. Phần này tập trung vào việc chi tiết hóa các dữ liệu khác nhau mà chúng ta có thể sử dụng để đánh giá hiệu suất của ASR. Hiệu suất của hệ thống nhận dạng giọng nói thường dựa vào hai yếu tố chính: sự chính xác của kết quả đầu ra mà nó tạo ra và tốc độ xử lý của ASR [5].

Khác với các mô hình dịch máy, độ chính xác được đánh giá theo thang điểm BLEU hay các bài toán phân lớp thường thấy được đánh giá theo ma trận hỗn loạn. Độ chính xác của các bài toán xử lý tín hiệu âm thanh được đánh giá theo WER (Word Error Rate), WRR (Word Recognition Rate) hoặc cả hai.

Tính toán độ chính xác của ASR là một thách thức vì đầu ra của ASR có thể có độ dài khác nhau so với độ dài thực tế của văn bản nói. WER thường được sử dụng làm số liệu để đánh giá hiệu suất của ASR, vì nó đo lường các lỗi ở mức từ thay vì ở mức âm vị. Công thức tính WER có thể được biểu thị như sau:

$$WER = \frac{S + D + I}{N}$$

WRR là một biến thể của WER cũng có thể được sử dụng để đánh giá hiệu suất của ASR. Nó có thể được tính bằng công thức sau:

$$WRR = 1 - WER = \frac{N - S - D - I}{N} = \frac{H - I}{N}$$

Trong đó S là số từ bị thay thế trong văn bản đầu ra so với văn bản thực tế. D là số từ bị bỏ sót và I là số từ bị thay thế được thực hiện. N là tổng số từ trong văn bản thực tế và $H = N - (S + D)$ đại diện cho tổng số từ đoán đúng [5].

Hệ số thời gian thực (RTF) là số liệu được sử dụng phổ biến nhất để tính tốc độ của các mô hình được. RTF có thể được tính bằng cách sử dụng công thức sau:

$$RTF = \frac{P}{I}$$

Tại P là thời gian hệ thống xử lý đầu vào và I là thời lượng của âm thanh đầu vào. Nếu RTF bằng 1 thì âm thanh đầu vào được xử lý “Real-time”. RTF là một giá trị phụ thuộc nhiều vào phần cứng và nó không chỉ giới hạn trong việc tính toán tốc độ của mô hình nhận dạng giọng nói. Nó có thể được sử dụng để tính toán tốc độ của bất kỳ mô hình nào có thể xử lý đầu vào âm thanh hoặc video [5].

2.4. Các ứng dụng

Hiện nay, công nghệ nhận dạng giọng nói đã được ứng dụng rộng rãi trong nhiều lĩnh vực như trợ lý ảo [1], nhận diện người nói [6], điều khiển robot [39], và nhiều ứng dụng khác.

Các ứng dụng dịch ngôn ngữ như Google Translate có thể được sử dụng để dịch giọng nói của người nói từ một ngôn ngữ sang ngôn ngữ khác. Các ứng dụng học tập như Duolingo có thể được sử dụng để luyện tập phát âm tiếng Anh bằng giọng nói. Các ứng dụng chăm sóc sức khỏe như Babylon Health có thể được sử dụng để hỏi ý kiến bác sĩ bằng giọng nói.

Nhận dạng giọng nói là một công nghệ đang phát triển nhanh chóng. Với sự phát triển của các mô hình học máy mạnh mẽ, nhận dạng giọng nói đang trở nên chính xác và hiệu quả hơn. Điều này dẫn đến việc mở ra nhiều ứng dụng mới cho công nghệ này.

2.5. Wav2vec 2.0

Wav2vec 2.0 là một framework học tập tự giám sát để biểu diễn giọng nói. Nó được phát triển bởi Meta AI và phát hành vào năm 2020 [40]. Wav2vec 2.0 học từ dữ liệu giọng nói không được gắn nhãn để trích xuất các biểu diễn có thể được sử dụng cho nhiều tác vụ tiếp theo, chẳng hạn như nhận dạng giọng nói tự động, nhận dạng người nói và hiểu ngôn ngữ. Nó được đào tạo trên một tập dữ liệu lớn gồm các tệp âm thanh và văn bản và có thể tạo ra các bản dịch chính xác và tự nhiên hơn các mô hình chuyên đổi trước đó.

Mô hình này mã hóa âm thanh lời nói thông qua mạng nơ ron tích chập nhiều lớp và sau đó che giấu các biểu diễn giọng nói tiềm ẩn thu được, tương tự như Masked Language Modeling [41]. Các biểu diễn tiềm ẩn được đưa vào mạng Transformer để xây dựng các biểu diễn theo ngữ cảnh và mô hình được đào tạo thông qua một nhiệm vụ tương phản trong đó tiềm ẩn thực sự được phân biệt với các yếu tố gây nhiễu.

Wav2vec 2.0 có một số ưu điểm so với các phương pháp học biểu diễn giọng nói tự giám sát trước đây. Đầu tiên, nó có thể tìm hiểu các cách biểu diễn mạnh mẽ hơn từ dữ liệu âm thanh thô mà không cần bất kỳ dữ liệu được gắn nhãn nào. Thứ hai, việc đào tạo sẽ hiệu quả hơn và có thể được mở rộng thành các tập dữ liệu lớn hơn. Thứ ba, nó linh hoạt hơn và có thể được sử dụng cho nhiều nhiệm vụ tiếp theo hơn.

Wav2vec 2.0 vẫn đang được phát triển nhưng nó đã có tác động đáng kể đến lĩnh vực nhận dạng giọng nói. Nó là một công cụ đầy hứa hẹn để xây dựng các hệ thống nhận dạng giọng nói chính xác và hiệu quả hơn cũng như hỗ trợ các ứng dụng mới liên quan đến giọng nói.

2.5.1. Feature encoder

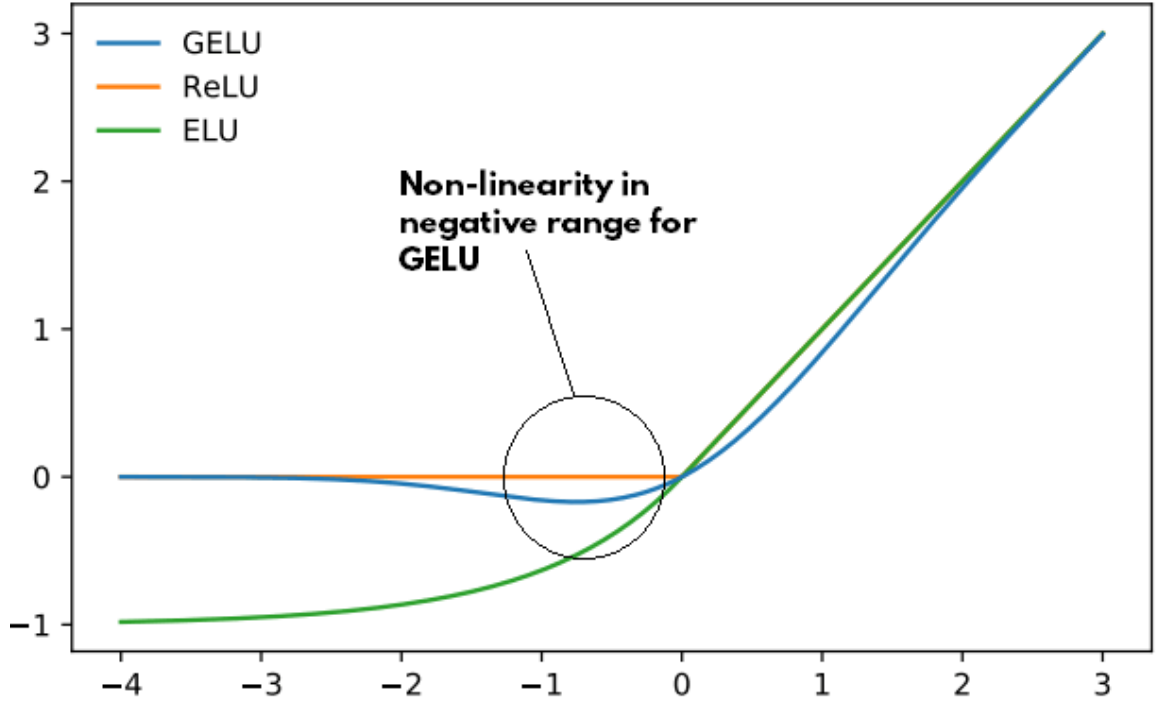
Bộ mã hóa bao gồm một số khối chứa tích chập thời gian, sau đó là lớp normalization và hàm kích hoạt GELU. Đầu vào dạng sóng thô tới bộ mã hóa được chuẩn hóa về giá trị trung bình bằng 0 và phương sai đơn vị. Tổng bước của bộ mã hóa xác định số bước thời gian T được đưa vào Transformer.

GELU là sự kết hợp giữa dropout, zoneout, và ReLU [42]. Quá trình kết hợp được thực hiện bằng cách nhân đầu vào với 0 hoặc một, nhưng các giá trị của mặt nạ 0 – 1 này được xác định ngẫu nhiên đồng thời cũng phụ thuộc vào dữ liệu đầu vào. Phân

phối Bernouli được chọn vì đầu vào neural có xu hướng tuân theo phân phối chuẩn, đặc biệt là với Batch Normalization. Hàm kích hoạt GELU được định nghĩa như sau:

$$GELU(x) = xP(X \leq x) = x\phi(x) = x \cdot \frac{1}{2} \left[1 + \operatorname{erf}\left(\frac{x}{\sqrt{2}}\right) \right]$$

Trong đó, $\phi(x) = P(X \leq x)$, $X \sim N(0, 1)$ là hàm phân phối tích lũy của phân phối chuẩn, erf là hàm lỗi Gauss với $\operatorname{erf}(a) = \frac{2}{\sqrt{\pi}} \int_0^a e^{-t^2} dt$ [43].



Hình 6: Hàm GELU ($\mu = 0$, $\sigma = 1$), ReLU, và ELU ($\alpha = 1$)

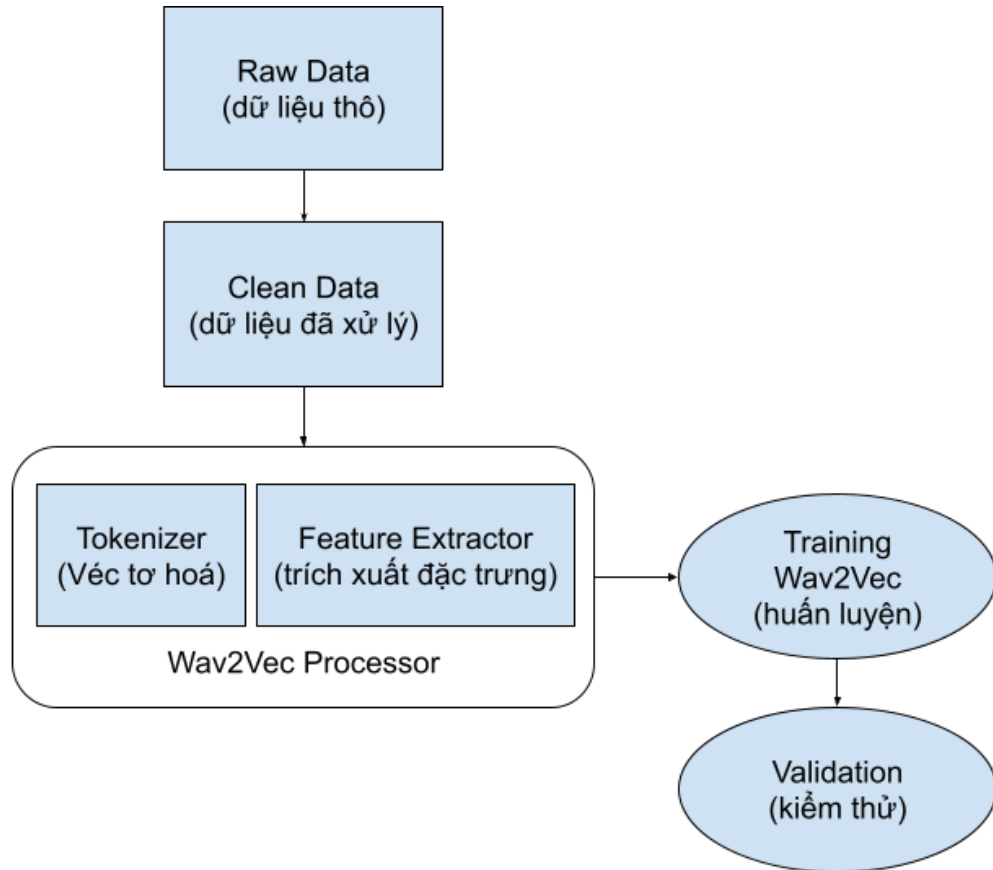
2.5.2. Biểu diễn theo ngữ cảnh với Transformer

Đầu ra của bộ mã hóa tính năng được đưa đến mạng ngữ cảnh theo kiến trúc Transformer [41,44]. Thay vì những vị trí cố định mã hóa thông tin vị trí tuyệt đối, lớp chấp tương tự hoạt động như những vị trí tương đối. GELU được thêm vào tích chập, theo sau là lớp chuẩn hóa Normalization.

CHƯƠNG 3: XÂY DỰNG MÔ HÌNH

3.1. Chuẩn bị dữ liệu

Trước khi tiến hành chúng ta nên nhìn vào bức tranh toàn cảnh để dễ hình dung toàn bộ các bước như sau:



Hình 7: Quy trình xây dựng mô hình nhận diện giọng nói Wav2Vec

Bộ dữ liệu âm thanh được sử dụng ta sẽ trích xuất từ bộ **common-voice** được cung cấp bởi Mozilla Foundation [5]. Đây là một dự án nguồn mở, nơi mọi người có thể đóng góp giọng nói của họ bằng cách đọc một câu cố định hoặc thời gian nhất định, điều này đặc biệt quan trọng để xác minh xem một tệp âm thanh cụ thể có phù hợp với phiên âm tương ứng hay không. Âm thanh được phát hành dưới dạng tệp MPEG-3 16bit, đơn kênh với tốc độ lấy mẫu 48kHz [45], chất lượng âm thanh phù hợp cho các ứng dụng giọng nói. Bộ dữ liệu này là một bộ dữ liệu âm thanh đa ngôn ngữ tuy nhiên ta sẽ chỉ trích xuất ra Tiếng Việt để sử dụng. Bao gồm tập training (2.53k), test (1.24k), validation (248). Mỗi một điểm dữ liệu sẽ có cấu trúc như sau:

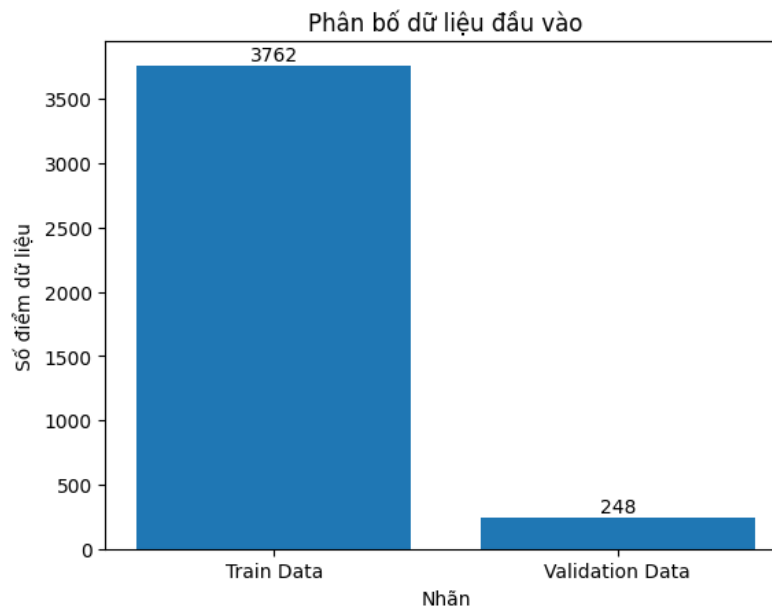
client_id	'c14ad590de005be3a512e187900c5ad5c76921...'
path	'/root/.cache/huggingface/datasets/downloads/extracted/...'
audio	{'path': '/root/.cache/huggingface/datasets/downloads/extracted...}'
sentence	'quả nhiên trúng tuyển vào trường Quốc Lập'
up_votes	2
down_votes	0
age	"
gender	"
accent	"
locale	vi'
segment	"

Bảng 2: Mô tả một điểm dữ liệu

Ở bài toán này, ta chỉ quan tâm tới 3 thuộc tính là **path**, **audio** và **sentence**. Các thuộc tính còn lại không có giá trị sử dụng cho bài toán nên ta sẽ loại bỏ đi để tiết kiệm bộ nhớ. Do dữ liệu được chia ở tập test có số lượng khoảng một nửa tập training, cho nên để tối ưu nhất ta sẽ gộp tập train và test làm 1 rồi dùng validation để kiểm thử ở bước cuối cũng như kiểm tra sự dao động của độ đo sai số trong quá trình huấn luyện.

```
# Lấy dữ liệu common_voice tiếng Việt
ds_train = load_dataset("mozilla-
foundation/common_voice_11_0", "vi", split="train+test")
ds_test = load_dataset("mozilla-
foundation/common_voice_11_0", "vi", split="validation")
```

Việc kết hợp này được áp dụng do data hiện có quá ít. Và trong quá trình thực nghiệm chúng tôi nhận thấy khi kết hợp tập test vào bộ training độ chính xác của mô hình tăng đáng kể so với việc không chia hoặc chia theo tỉ lệ 80/20. Cuối cùng ta sẽ có phân bố dữ liệu như sau.



Hình 8: Phân bố dữ liệu đầu vào

3.2. Xử lý dữ liệu

Đối với dữ liệu âm thanh (audio) ở tập này dữ liệu âm thanh đang được để Sampling Rate là **48.000 mẫu/1 giây**.

Như đã biết thì Sampling Rate càng cao thì số lượng mẫu âm thanh lưu trữ càng nhiều, nhờ vậy mà âm thanh trở nên trong trẻo, rõ ràng và chính xác với thực tế hơn. Tuy vậy để tối ưu trong học máy ta không cần đến mức đó, với bài toán này ta chỉ cần mức Sampling Rate nhỏ vừa đủ để có thể nghe, với lại khi chỉnh nhỏ giúp mô hình huấn luyện nhanh hơn và giảm chi phí tính toán. Vì vậy ta sẽ dùng hàm **cast_column** để chuyển audio có sampling rate từ 48.000 về 16.000.

Sau khi đã xử lý xong trên cột âm thanh ta sẽ tiếp tục xử lý trên dữ liệu văn bản (**sentence**). Ta nhận ra rằng, từ âm thanh chuyển sang văn bản, những dấu câu thường sẽ không có ý nghĩa, vì đơn giản dấu câu dùng để thể hiện trong văn viết, với lại khi thêm dấu câu mô hình sẽ rất khó có thể học được và xác định nó, nếu một vài dấu câu sai cũng sẽ gây rất nhiều phiền toái về ngữ nghĩa làm người đọc hiểu sai. Vì vậy để đơn giản chúng ta sẽ xóa hết những dấu câu này cũng như những ký tự đặc biệt.

```
# Xóa dấu và các ký tự đặc biệt
import re
chars_to_ignore_regex = '[\, \? \. \! \- \; \: \" \' \% \' \' \? ]'
```

```
def remove_special_characters(batch):  
    batch["sentence"] = re.sub(chars_to_ignore_regex, '',  
    batch["sentence"]).lower() + " "  
    return batch
```

Vậy là chúng ta đã xong các bước xử lý dữ liệu cơ bản, tiếp theo chúng ta sẽ tiến hành xây dựng bộ Tokenizer.

3.3. Xây dựng bộ tokenizer

Máy tính không thể hiểu được các văn bản, ta cần chuyển đổi các từ trong văn bản này sang dạng số (encoding) để máy tính có thể tính được xác suất của từng từ một. Thông thường ta có thể làm nhanh bước này bằng cách sử dụng một bộ thư viện có sẵn để chuyển đổi, tuy nhiên ta sẽ làm thủ công để có thể hiểu rõ hơn cách nó hoạt động, với lại đối với bài toán liên quan tới âm thanh, bộ Tokenizer này sẽ đặc biệt hơn một chút.

```
# Lấy ra tất cả ký tự có thể tìm được trong dataset  
def extract_all_chars(batch):  
    all_text = " ".join(batch["sentence"])  
    vocab = list(set(all_text))  
    return {"vocab": [vocab], "all_text": [all_text]}
```

Ta sẽ tiến hành nối tất cả các hàng trong cột sentence thành một hàng văn bản duy nhất rồi sẽ trích xuất các ký tự xuất hiện trong văn bản đó ra. Lưu ý ở đây ta sẽ chỉ lấy ký tự chứ không lấy từ vựng như các bộ tokenizer LLM thông thường. Vì âm thanh không cần dự đoán ngữ nghĩa nhiều, mô hình chỉ cần xác định đúng từ vựng được nói là được, CTC là một phương pháp học máy thường được sử dụng trong các nhiệm vụ như nhận dạng tiếng nói và nhận dạng văn bản. Nó cho phép mô hình dự đoán chuỗi ký tự mà không cần cung cấp độ dài chuỗi cố định. Trong học máy, việc xử lý dữ liệu một cách đúng đắn rất quan trọng để đảm bảo mô hình được đào tạo hiệu quả, đây được gọi là cơ chế CTC. Việc lấy theo ký tự này giúp ta có thể trích lọc ra gần như tất cả các ký tự chữ cái xuất hiện trong Tiếng Việt hiện nay.

```
| {'ứ': 0,
  'e': 1,
  'ồ': 2,
  'y': 3,
  'ặ': 4,
  'z': 5,
  's': 6,
  'à': 7,
  'l': 8,
  'à': 9,
  'ý': 10,
  'ỳ': 11,
  'ạ': 12,
  'ú': 13,
  'ớ': 14,
  'à': 15,
  'ỷ': 16,
  'í': 17,
  'á': 18,
  'ỹ': 19,
```

Hình 9: Danh sách một vài ký tự sau khi lấy được

Việc lấy ký tự cũng giúp ta chỉ cần lưu trữ một lượng nhỏ **94 phần tử** so với số lượng từ vựng đồ sộ trong Tiếng Việt thì lưu ký tự sẽ tối ưu hơn rất nhiều. Tuy nhiên ta cũng sẽ không quên thêm 2 phần tử là [UNK] và [PAD] để cho từ không xác định và padding.

```
# Thêm [UNK] cho từ không xác định
vocab_dict["[UNK]"] = len(vocab_dict)
# Thêm [PAD] để padding cho cùng chiều dài input
vocab_dict["[PAD]"] = len(vocab_dict)
# In chiều dài từ điển
len(vocab_dict)
```

Sau khi xong ta sẽ lưu lại bộ vocab này với định dạng json và tạo tokenizer theo cơ chế CTC mặc định với mô hình Wav2Vec2. Để tạo được bộ tokenizer ta sẽ cần load config từ mô hình gốc, tại đây ta sẽ lấy mô hình **facebook/wav2vec2-large-xlsr-53** và đặt tên cho mô mới là **vietnamese_tts_hus**.

```
# Tên model để load từ HuggingFace
# Model gốc
model_checkpoint = "facebook/wav2vec2-large-xlsr-53"
# Model đã training trước đó từ model gốc
model_checkpoint = "Tuan457/vietnamese_tts_hus"
```

Ta sẽ dùng hàm AutoConfig để lấy các thông tin cài đặt mặc định từ mô hình gốc về nhằm cài lên tokenizer của Tiếng Việt

```
from transformers import AutoConfig
# Load file config.json
config = AutoConfig.from_pretrained(model_wav2vec2)

tokenizer_type = config.model_type if
config.tokenizer_class is None else None
config = config if config.tokenizer_class is not None else
None
```

Cuối cùng ta sẽ gán các giá trị này vào hàm tạo tokenizer của Wav2Vec2.

```
# Tạo bộ tokenizer từ vocab.json
from transformers import Wav2Vec2CTCTokenizer
tokenizer = Wav2Vec2CTCTokenizer.from_pretrained(
    "./",
    config=config,
    tokenizer_type=tokenizer_type,
    unk_token="[UNK]",
    pad_token="[PAD]",
    word_delimiter_token="|",
)
```

3.4. Trích xuất đặc trưng

Đối với trích xuất đặc trưng chúng ta sẽ sử dụng hàm trích xuất mặc định của Wav2Vec, đây là hàm đã được tối ưu hoá để trích lọc đặc trưng dữ liệu cho mô hình Wav2Vec2.

```
# Load Extractor
from transformers import AutoFeatureExtractor
feature_extractor =
AutoFeatureExtractor.from_pretrained(model_wav2vec2)
feature_extractor

Wav2Vec2FeatureExtractor {
  "do_normalize": true,
  "feature_extractor_type": "Wav2Vec2FeatureExtractor",
  "feature_size": 1,
  "padding_side": "right",
  "padding_value": 0,
  "return_attention_mask": true,
  "sampling_rate": 16000 }
```

Lưu ý ở đây các thông số cho hàm trích xuất đặc trưng chúng ta sẽ để mặc định như model gốc padding bên phía bên phải của mảng và các thông số còn lại được mô tả như trên.

3.5. Tạo Processor cho Wav2Vec

Sau khi đã tạo xong **tokenizer** và **feature_extractor**, ta sẽ đưa chúng vào hàm **Wav2Vec2Processor** để tạo bộ xử lý cho dữ liệu.

```
# Load Wav2Vec2Processor
from transformers import Wav2Vec2Processor
processor =
Wav2Vec2Processor(feature_extractor=feature_extractor,
tokenizer=tokenizer)
```

Sau khi đã có bộ xử lý ta sẽ tiến hành áp dụng hàm biến đổi này lên toàn bộ các điểm dữ liệu, trong đó ở từng dòng ta sẽ biến đổi *đầu vào* và *đầu ra* của phần tử, với âm thanh đầu vào hàm processor sẽ trích xuất đặc trưng và biến đổi, với văn bản đầu ra thì sẽ tokenizer.

```
# Hàm map processor lên audio và nhãn của từng điểm dữ liệu
trên tập
def prepare_dataset(batch):
    audio = batch["audio"]

    # batched output is "un-batched"
    batch["input_values"] = processor(audio["array"],
        sampling_rate=audio["sampling_rate"]).input_values[0]
    batch["input_length"] = len(batch["input_values"])

    with processor.as_target_processor():
        batch["labels"] = processor(batch["sentence"]).input_ids
    return batch
```

Sau khi đã hoàn thành, ta sẽ tiến hành mapping nó trên tập train và test của dữ liệu.

3.6. Giới hạn độ dài audio

Để tối ưu tốc độ training cho mô hình ta sẽ giới hạn độ dài audio đầu vào trên dataset bằng cách cắt với độ dài tối đa cho phép trên dữ liệu âm thanh là **15 giây**.

```
# Đặt độ dài tối đa đầu vào là 15s
```

```

max_input_length_in_sec = 15.0
ds_train = ds_train.filter(lambda x: x <
                             max_input_length_in_sec *
                             processor.feature_extractor.sampling_rate,
                             input_columns=["input_length"])
ds_test = ds_test.filter(lambda x: x <
                           max_input_length_in_sec *
                           processor.feature_extractor.sampling_rate,
                           input_columns=["input_length"])

```

3.7. Tạo trình đối chiếu dữ liệu DataCollator

Dữ liệu audio đầu vào và văn bản đầu ra sẽ được padding theo cơ chế CTC và các cài đặt processor chúng ta đã chuẩn bị phía trên thông qua lớp này.

```

# Trình đối chiếu dữ liệu
data_collator =
DataCollatorCTCWithPadding(processor=processor,
                             padding=True)

```

Với các phần ta pad sẽ được điền đầy bằng cách thêm giá trị **-100** vào để bỏ qua tính mất mát (loss).

3.8. Tạo trình tính độ đo metrics

Đối với bài toán Speech-to-Text thì ngoài sai số loss ra chúng ta sẽ dùng độ đo Word error rate (WER) để ước lượng hiệu suất của mô hình.

```

# Load hàm tính toán chỉ số wer
wer_metric = load_metric("wer")

```

Lưu ý là chỉ số WER này chỉ được tính trên tập dữ liệu kiểm thử (test) để kết quả khách quan nhất có thể. Ngoài ra đối với token có giá trị **-100** tức là pad dùng để điền đầy độ dài văn bản ta đã cài đặt lúc đầu sẽ được chuyển thành pad_token_id của tokenizer.

```

# Hàm tính chỉ số wer trên tập validation
def compute_metrics(pred):
    pred_logits = pred.predictions
    pred_ids = np.argmax(pred_logits, axis=-1)

    pred.label_ids[pred.label_ids == -100] =
        processor.tokenizer.pad_token_id

```

```

pred_str = processor.batch_decode(pred_ids)
# Không nhóm các tokens (group_tokens) khi tính metrics
label_str = processor.batch_decode(pred.label_ids,
                                   group_tokens=False)

wer = wer_metric.compute(predictions=pred_str,
                        references=label_str)

return {"wer": wer}

```

3.9. Tạo mô hình huấn luyện

Ta sẽ tiến hành load mô hình từ **facebook/wav2vec2-large-xlsr-53** về rồi thiết đặt các thông số như dropout và các thông số khác như bên dưới, ta cũng cần phải chuyển mô hình sang device là **cuda** để các trọng số mô hình sẽ được lưu trên GPU, đảm bảo việc tính toán và lưu trữ tận dụng được tối đa sức mạnh của phần cứng.

```

from transformers import AutoModelForCTC
# Tải model wav2vec2 gốc
model = AutoModelForCTC.from_pretrained(
    model_wav2vec2,
    attention_dropout=0.1,
    hidden_dropout=0.1,
    feat_proj_dropout=0.0,
    mask_time_prob=0.05,
    layerdrop=0.1,
    ctc_loss_reduction="mean",
    pad_token_id=processor.tokenizer.pad_token_id,
    vocab_size=len(processor.tokenizer)
).to("cuda")

```

Sau đó ta sẽ đóng băng lớp trích xuất đặc trưng để đảm bảo lớp này không bị thay đổi các giá trị trọng số.

```

# Đóng băng lớp trích xuất đặc trưng để không training lớp này
if hasattr(model, "freeze_feature_extractor"):
    model.freeze_feature_extractor()

```

Sau đó ta sẽ tạo **TrainingArguments** để thiết đặt các siêu tham số cho mô hình này. Mô hình sẽ có các siêu tham số như sau:

Hyper-Parameter	Values
batch_size	30
epochs	7 (thực tế là 21)
learning_rate	0.0003 (3e-4)
save_steps eval_steps logging_steps	50

Bảng 3: Cài đặt siêu tham số mô hình

Ta đặt batch_size trong trường hợp này là 30, mục đích là để phân dữ liệu huấn luyện thành các lô tập hợp 30 điểm mỗi lô training, giúp tăng khả năng hội tụ của mô hình, tiết kiệm thời gian khi xử lý song song cũng như tận dụng tối đa VRAM trên GPU, nếu lớn hơn 30 sẽ bị tràn VRAM. Số epochs được đặt ở đây là 7 để tạm thời, thực tế mô hình sẽ được training qua 21 epochs do thời gian training khá lâu (khoảng 1 giờ 30 phút / 7 epochs) sẽ không thuận tiện cho việc treo máy nên chiến lược chúng ta sẽ là chia ra từng đợt để train, tổng cộng sẽ có 3 đợt mỗi đợt sẽ train 7 epochs, mỗi lần train xong sẽ lưu lại mô hình và khi nào có thời gian thuận tiện việc treo máy sẽ lấy ra train tiếp tục. Ta cũng sẽ lưu lại các thông số đo được trong quá trình này. Các step ta sẽ đặt cứ 50 bước sẽ lưu và tính toán metrics thông báo 1 lần. Còn lại sẽ để theo mặc định như sau.

```
from transformers import TrainingArguments

training_args = TrainingArguments(
    output_dir="wav2vec_vietnamese",
    group_by_length=True,
    per_device_train_batch_size=30,
    gradient_accumulation_steps=2,
    evaluation_strategy="steps",
    num_train_epochs=7,
    gradient_checkpointing=True,
    save_steps=50,
    eval_steps=50,
```

```
logging_steps=50,  
learning_rate=3e-4,  
save_total_limit=2,  
)
```

Cuối cùng ta sẽ tổng hợp lại tất cả rồi đưa vào hàm trainer rồi chạy hàm train() để huấn luyện mô hình.

```
from transformers import Trainer  
  
trainer = Trainer(  
    model=model,  
    data_collator=data_collator,  
    args=training_args,  
    compute_metrics=compute_metrics,  
    train_dataset=ds_train,  
    eval_dataset=ds_test,  
    tokenizer=processor.feature_extractor,  
)
```

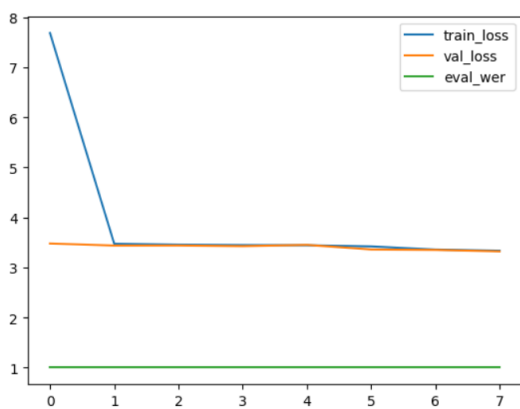
Vậy là chúng ta đã hoàn thành từ quá trình xử lý dữ liệu cho đến khởi tạo và huấn luyện mô hình, tiếp theo chúng ta sẽ tiến hành thực nghiệm để đưa ra kết quả. Toàn bộ link mã nguồn dự án sẽ được đính kèm phía cuối bài báo cáo này.

CHƯƠNG 4: THỰC NGHIỆM VÀ ĐÁNH GIÁ KẾT QUẢ

4.1. Kết quả thực nghiệm

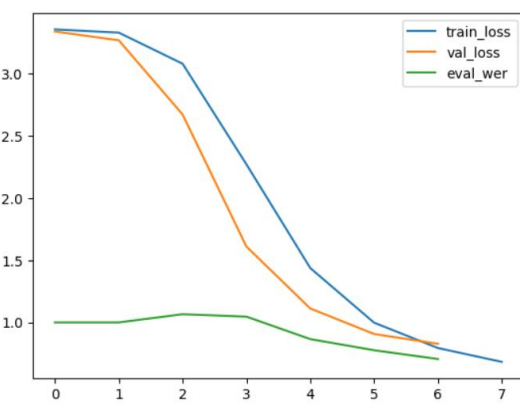
Như đã đề cập ở trên, chiến lược huấn luyện sẽ chia ra làm 3 đợt với mỗi đợt là 7 epochs vì vậy chúng ta sẽ có kết quả từng đợt như sau:

Kết quả Đợt 1:

Biểu đồ minh họa epochs [1 - 7]	Chi tiết thông số																																				
	<div><div></div> [441/441 1:40:20, Epoch 7/7]</div> <table><tr><th>Step</th><th>Training Loss</th><th>Validation Loss</th><th>Wer</th></tr><tr><td>50</td><td>7.683600</td><td>3.479072</td><td>1.000000</td></tr><tr><td>100</td><td>3.475400</td><td>3.438518</td><td>1.000000</td></tr><tr><td>150</td><td>3.456700</td><td>3.439837</td><td>1.000000</td></tr><tr><td>200</td><td>3.448600</td><td>3.426676</td><td>1.000000</td></tr><tr><td>250</td><td>3.444200</td><td>3.451800</td><td>1.000000</td></tr><tr><td>300</td><td>3.420900</td><td>3.360232</td><td>1.000000</td></tr><tr><td>350</td><td>3.358700</td><td>3.351466</td><td>1.000000</td></tr><tr><td>400</td><td>3.334600</td><td>3.322449</td><td>1.000000</td></tr></table>	Step	Training Loss	Validation Loss	Wer	50	7.683600	3.479072	1.000000	100	3.475400	3.438518	1.000000	150	3.456700	3.439837	1.000000	200	3.448600	3.426676	1.000000	250	3.444200	3.451800	1.000000	300	3.420900	3.360232	1.000000	350	3.358700	3.351466	1.000000	400	3.334600	3.322449	1.000000
Step	Training Loss	Validation Loss	Wer																																		
50	7.683600	3.479072	1.000000																																		
100	3.475400	3.438518	1.000000																																		
150	3.456700	3.439837	1.000000																																		
200	3.448600	3.426676	1.000000																																		
250	3.444200	3.451800	1.000000																																		
300	3.420900	3.360232	1.000000																																		
350	3.358700	3.351466	1.000000																																		
400	3.334600	3.322449	1.000000																																		

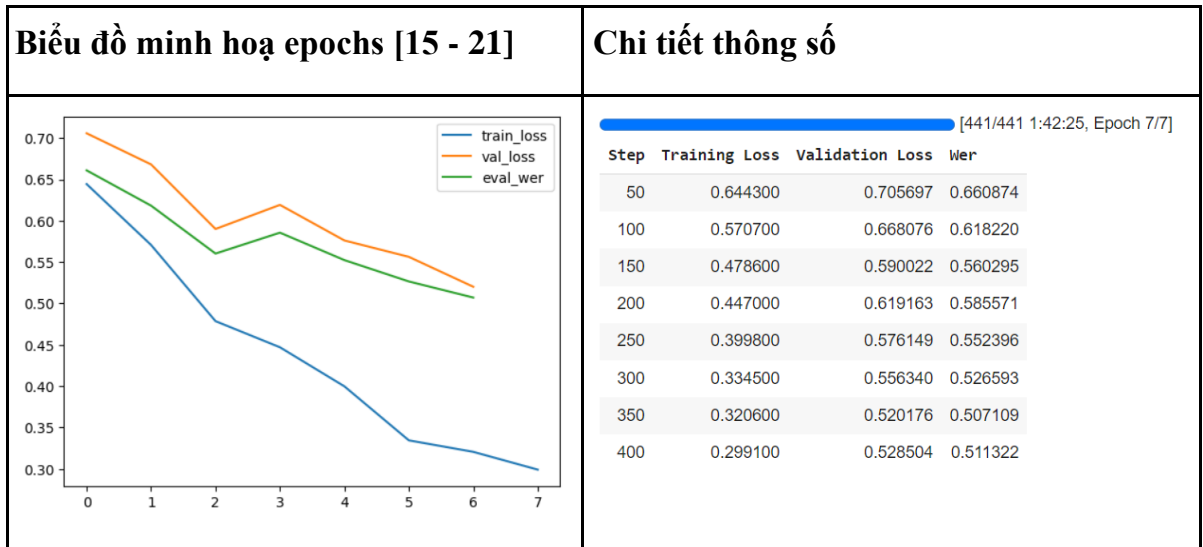
Bảng 4: Kết quả huấn luyện đợt 1 (epochs 1 - 7)

Kết quả Đợt 2:

Biểu đồ minh họa epochs [8 - 14]	Chi tiết thông số																																				
	<div><div></div> [441/441 1:41:28, Epoch 7/7]</div> <table><tr><th>Step</th><th>Training Loss</th><th>Validation Loss</th><th>Wer</th></tr><tr><td>50</td><td>3.356600</td><td>3.339768</td><td>1.000000</td></tr><tr><td>100</td><td>3.330600</td><td>3.268936</td><td>1.000000</td></tr><tr><td>150</td><td>3.080500</td><td>2.672462</td><td>1.065824</td></tr><tr><td>200</td><td>2.271200</td><td>1.610082</td><td>1.046340</td></tr><tr><td>250</td><td>1.437300</td><td>1.111935</td><td>0.865719</td></tr><tr><td>300</td><td>0.997800</td><td>0.906699</td><td>0.776198</td></tr><tr><td>350</td><td>0.793900</td><td>0.828051</td><td>0.705635</td></tr><tr><td>400</td><td>0.682600</td><td>0.742553</td><td>0.672986</td></tr></table>	Step	Training Loss	Validation Loss	Wer	50	3.356600	3.339768	1.000000	100	3.330600	3.268936	1.000000	150	3.080500	2.672462	1.065824	200	2.271200	1.610082	1.046340	250	1.437300	1.111935	0.865719	300	0.997800	0.906699	0.776198	350	0.793900	0.828051	0.705635	400	0.682600	0.742553	0.672986
Step	Training Loss	Validation Loss	Wer																																		
50	3.356600	3.339768	1.000000																																		
100	3.330600	3.268936	1.000000																																		
150	3.080500	2.672462	1.065824																																		
200	2.271200	1.610082	1.046340																																		
250	1.437300	1.111935	0.865719																																		
300	0.997800	0.906699	0.776198																																		
350	0.793900	0.828051	0.705635																																		
400	0.682600	0.742553	0.672986																																		

Bảng 5: Kết quả huấn luyện đợt 2 (epochs 8 - 14)

Kết quả Đợt 3:

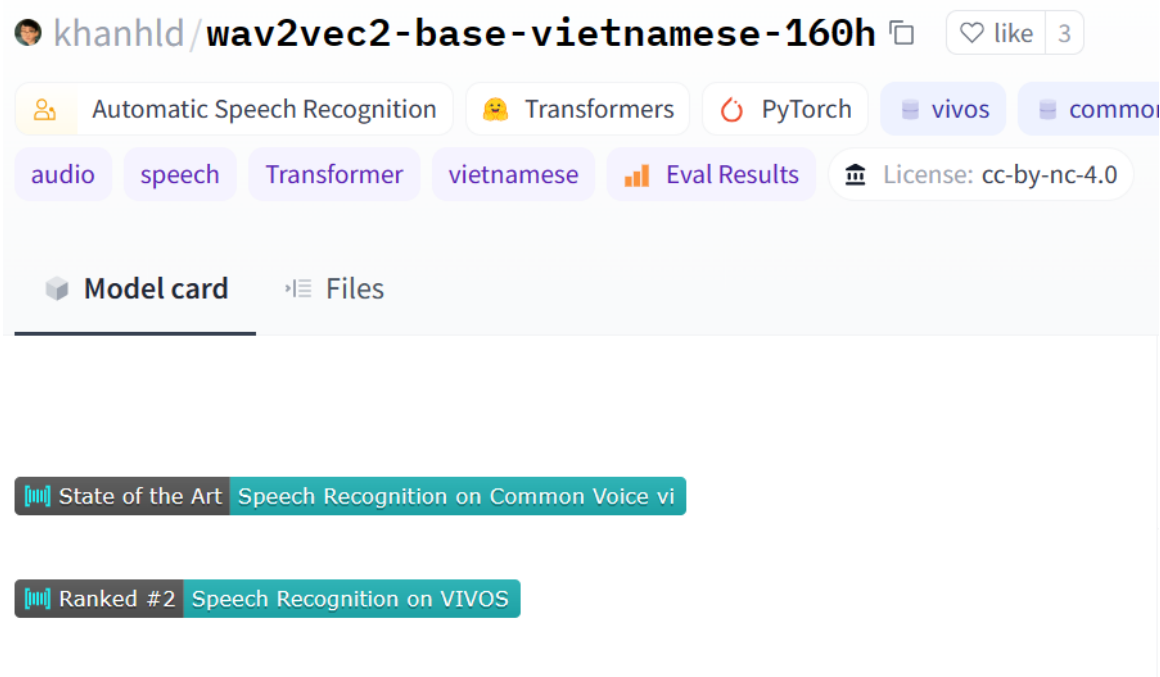


Bảng 6: Kết quả huấn luyện đợt 3 (epochs 15 - 21)

Đánh giá kết quả: Chúng ta nhận thấy rằng, ở 7 epochs đợt đầu tiên mô hình vẫn chưa có sự thay đổi nào ở chỉ số WER nhưng loss trên tập training giảm đáng kể và gần như dừng lại gần bằng loss validation. Tiếp tục ở đợt 2 sự thay đổi rõ ràng hơn ta có thể quan sát được bằng mắt khi loss trên hai tập giảm đều theo thời gian xuống dưới 1 và WER đã có sự thay đổi. Qua đợt 3 ta cũng thấy rằng loss và WER cũng tiếp tục giảm đều và chưa thấy xuất hiện hiện tượng quá khớp (overfitting) trên mô hình. Đây là một kết quả đáng mong đợi tuy nhiên để chính xác hơn ta sẽ cần training trên nhiều bộ dữ liệu khác nhau với từng vùng miền và tăng epochs để mô hình có độ chính xác cao hơn.

4.2. So sánh với các mô hình nhận diện giọng nói khác

Hiện tại, ở thời điểm đang viết bài báo cáo này, mô hình được công bố có số điểm WER tốt nhất trên Tiếng Việt là **khanhld/wav2vec2-base-vietnamese-160h**. Mô hình này đạt chỉ số WER trên tập **test** bộ Tiếng Việt của **Common Voice 8.0** là **10.78** đây chính là chỉ số cao nhất từng ghi nhận và được cho là state-of-the-art (SOTA) trên tiếng Việt hiện tại.



Hình 10: Mô hình SOTA cho Tiếng Việt trên bộ Common Voice 8.0

Ta sẽ thử lấy mô hình hiện có của chúng ta để so sánh với mô hình SOTA này, sẽ có kết quả như sau:

Model	WER
khanhld/wav2vec2-base-vietnamese-160h (SOTA)	10.78
Our Model	20.5

Bảng 7: So sánh mô hình hiện tại với mô hình SOTA

Đánh giá: Mặc dù đã đến rất gần nhưng mô hình của chúng ta vẫn chưa thể vượt qua mô hình SOTA hiện có. Nguyên nhân có thể được đưa ra là do giới hạn về phần cứng, thời gian huấn luyện cũng như độ lớn của tập dữ liệu huấn luyện còn hạn chế. Nếu được đầu tư phát triển cũng như được sự đóng góp tích cực từ cộng đồng và kết hợp với nhiều phương pháp mô hình ngôn ngữ khác nhau, mô hình này có thể đạt được kết quả vượt sự kỳ vọng.

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. Kết luận

Nghiên cứu này không chỉ giải quyết một vấn đề quan trọng mà còn đóng góp vào sự phát triển của lĩnh vực và cung cấp một cơ sở cho các nghiên cứu và ứng dụng trong tương lai.

Tóm lại, trong đề án này, chúng ta đã thực hiện thành công quá trình fine tuning model **facebook/wav2vec2-large-xlsr-53** cho tiếng Việt bằng thư viện **Wav2vec2** có CTC. Kết quả độ đo WER trên tập validation là 0.511322. Kết quả này cho thấy hệ thống đã đạt được độ chính xác tương đối ở mức kỳ vọng.

Để đạt được kết quả này, ta đã thực hiện qua các bước bao gồm:

- Tiền xử lý dữ liệu âm thanh tiếng Việt trong bộ **common_voice_11**
- Tạo bộ processor, áp dụng các thay đổi lên trên bộ dữ liệu.
- Finetuning model mô hình với các siêu tham số được tối ưu hóa.
- Đánh giá độ chính xác của model trên tập validation.

Tuy nhiên, vẫn còn một số vấn đề cần được cải thiện, chẳng hạn như:

- Độ chính xác của hệ thống có thể được cải thiện bằng cách tăng kích thước tập dữ liệu huấn luyện.
- Có thể sử dụng các kỹ thuật finetuning khác để cải thiện độ chính xác của hệ thống.
- Có thể sử dụng các mô hình ngôn ngữ lớn (LLM) để cải thiện khả năng hiểu ngôn ngữ tự nhiên của hệ thống.

5.2. Hướng phát triển

Nghiên cứu có thể mở rộng để áp dụng mô hình Speech-to-Text vào các ứng dụng thực tế như tạo phụ đề cho video, triển khai trợ lý ảo hoặc tích hợp vào các sản phẩm và dịch vụ liên quan đến ngôn ngữ tiếng Việt.

Để cải thiện độ chính xác của model, chúng ta có thể thực hiện các bước sau:

- **Tăng kích thước của tập dữ liệu tiếng Việt.** Điều này có thể được thực hiện bằng cách thu thập thêm dữ liệu từ các nguồn khác nhau, chẳng hạn như các cuộc hội thoại, bản ghi âm, v.v.

- **Sử dụng các kỹ thuật tăng cường dữ liệu** để tạo ra thêm dữ liệu từ tập dữ liệu hiện có. Một số kỹ thuật tăng cường dữ liệu phổ biến bao gồm:
 - **Phản chiếu:** Lặp lại các mẫu dữ liệu hiện có theo chiều ngược lại.
 - **Trộn:** Kết hợp các mẫu dữ liệu hiện có với nhau.
 - **Thêm tiếng ồn:** Thêm tiếng ồn vào các mẫu dữ liệu hiện có.
- **Sử dụng các kỹ thuật tối ưu hóa khác nhau.** Các kỹ thuật tối ưu hóa khác nhau có thể được sử dụng để cải thiện độ chính xác của model. Một số kỹ thuật tối ưu hóa phổ biến bao gồm:
 - **AdamW:** Một biến thể của Adam sử dụng trọng số giảm động.
 - **Adagrad:** Một kỹ thuật tối ưu hóa dựa trên độ dốc.
 - **RMSProp:** Một kỹ thuật tối ưu hóa dựa trên bình phương sai số trung bình.

Ngoài ra, chúng ta cũng có thể thử nghiệm các model khác nhau để xem liệu chúng có thể đạt được độ chính xác cao hơn hay không. Hy vọng rằng những ý tưởng này sẽ giúp cải thiện độ chính xác của model và làm cho nó trở nên hữu ích hơn cho các ứng dụng thực tế.

TÀI LIỆU THAM KHẢO

- [1] N. T. M. Thanh, P. X. Dung, N. N. Hay, L. N. Bich, and D. X. Quy, ‘Đánh giá các hệ thống nhận dạng giọng nói tiếng việt (vais, viettel, zalo, fpt và google) trong bản tin’, *Journal of Technical Education Science*, no. 63, pp. 28–35, Apr. 2021.
- [2] T. Đ. Minh and N. T. Luận, ‘Using fuzzy logic in Vietnamese speech recognition’, in *Tạp Chí Khoa Học và Công Nghệ-Đại Học Đà Nẵng*, 2014, pp. 64–70.
- [3] C. Wang *et al.*, ‘fairseq S2T: Fast Speech-to-Text Modeling with fairseq’, *arXiv [cs.CL]*, 11-Oct-2020.
- [4] S. Bansal, H. Kamper, A. Lopez, and S. Goldwater, ‘Towards speech-to-text translation without speech recognition’, *arXiv [cs.CL]*, 13-Feb-2017.
- [5] M. Malik, M. K. Malik, K. Mehmood, and I. Makhdoom, ‘Automatic speech recognition: a survey’, *Multimed. Tools Appl.*, vol. 80, no. 6, pp. 9411–9457, Mar. 2021.
- [6] G. Saha, S. Chakroborty, and S. Senapati, ‘A new silence removal and endpoint detection algorithm for speech and speaker recognition applications’, in *Proceedings of the 11th national conference on communications (NCC)*, 2005, pp. 291–295.
- [7] B. Yegnanarayana and R. N. J. Veldhuis, ‘Extraction of vocal-tract system characteristics from speech signals’, *IEEE Trans. Speech Audio Process.*, vol. 6, no. 4, pp. 313–327, Jul. 1998.
- [8] I. Mporas, T. Ganchev, and M. Sifarakas, ‘Comparison of speech features on the speech recognition task’, *J. Comput. Sci.*, vol. 3, no. 8, pp. 608–616, Aug. 2007.
- [9] W. Alkhalidi, W. Fakhr, and N. Hamdy, ‘Automatic speech/speaker recognition in noisy environments using wavelet transform’, in *The 2002 45th Midwest Symposium on Circuits and Systems, 2002. MWSCAS-2002*, Tulsa, OK, USA, 2003.

- [10] V. V. Krishnan and P. B. Anto, 'Features of wavelet packet decomposition and discrete wavelet transform for malayalam speech recognition', *International Journal of Recent Trends in Engineering*, vol. 1, no. 2, 2009.
- [11] B. Zamani, A. Akbari, B. Nasersharif, and A. Jalalvand, 'Optimized discriminative transformations for speech features based on minimum classification error', *Pattern Recognit. Lett.*, vol. 32, no. 7, pp. 948–955, May 2011.
- [12] S. Ranjan, 'A discrete wavelet transform based approach to Hindi speech recognition', in *2010 International Conference on Signal Acquisition and Processing*, Bangalore, India, 2010.
- [13] D. Oshaughnessy, 'Automatic speech recognition: History, methods and challenges', *Pattern Recognition*, no. 10, pp. 2965–2979, 2008.
- [14] L. Rabiner and B. Juang, 'An introduction to hidden Markov models', *IEEE ASSP Mag.*, vol. 3, no. 1, pp. 4–16, 1986.
- [15] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, 'Speaker verification using adapted Gaussian mixture models', *Digit. Signal Process.*, vol. 10, no. 1–3, pp. 19–41, Jan. 2000.
- [16] K. Aida-zade, A. Xocayev, and S. Rustamov, 'Speech recognition using Support Vector Machines', in *2016 IEEE 10th International Conference on Application of Information and Communication Technologies (AICT)*, Baku, Azerbaijan, 2016.
- [17] E. S. Wahyuni, 'Arabic speech recognition using MFCC feature extraction and ANN classification', in *2017 2nd International conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE)*, Yogyakarta, 2017.
- [18] A. Mohamed and K. N. R. Nair, 'HMM/ANN hybrid model for continuous Malayalam speech recognition', *Procedia Eng.*, vol. 30, pp. 616–622, 2012.
- [19] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, 'Listen, attend and spell: A neural network for large vocabulary conversational speech recognition', in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Shanghai, 2016.

- [20] G. Hemakumar and P. Punitha, ‘Speech recognition technology: a survey on Indian languages’, *International Journal of Information Science and Intelligent System*, vol. 2, no. 4, pp. 1–38, 2013.
- [21] J. W. Forgie and C. D. Forgie, ‘Results obtained from a vowel recognition computer program’, *J. Acoust. Soc. Am.*, vol. 31, no. 6_Supplement, pp. 844–844, Jun. 1959.
- [22] V. M. Velichko and N. G. Zagoruyko, ‘Automatic recognition of 200 words’, *Int. J. Man. Mach. Stud.*, vol. 2, no. 3, pp. 223–234, Jul. 1970.
- [23] H. Sakoe and S. Chiba, ‘Dynamic programming algorithm optimization for spoken word recognition’, in *Readings in Speech Recognition*, Elsevier, 1990, pp. 159–165.
- [24] F. Weninger *et al.*, ‘Speech enhancement with LSTM recurrent neural networks and its application to noise-robust ASR’, in *Latent Variable Analysis and Signal Separation*, Cham: Springer International Publishing, 2015, pp. 91–99.
- [25] G. Hinton *et al.*, ‘Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups’, *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.
- [26] L. Dong, S. Xu, and B. Xu, ‘Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition’, in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Calgary, AB, 2018.
- [27] H. M. Tan, K.-W. Liang, Y.-S. Lee, C.-T. Li, Y.-H. Li, and J.-C. Wang, ‘Speech separation using augmented-discrimination learning on squash-norm embedding vector and node encoder’, *IEEE Access*, vol. 10, pp. 102048–102063, 2022.
- [28] Yu, Dong, and Lin Deng ‘Automatic speech recognition’, *Berlin: Springer*, Vol. 1, 2016.
- [29] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, ‘Front-end factor analysis for speaker verification’, *IEEE Trans. Audio Speech Lang. Processing*, vol. 19, no. 4, pp. 788–798, May 2011.

- [30] D. A. Reynolds and R. C. Rose, ‘Robust text-independent speaker identification using Gaussian mixture speaker models’, *IEEE Trans. Speech Audio Process.*, vol. 3, no. 1, pp. 72–83, 1995.
- [31] M. Fujimoto and Y. A. Riki, ‘Robust speech recognition in additive and channel noise environments using GMM and EM algorithm’, in *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, Montreal, Que., Canada, 2004.
- [32] Moon, Todd K. "The expectation-maximization algorithm.", *IEEE Signal processing magazine*, pp. 47-60, 1996.
- [33] Jelinek, F. ‘Continuous speech recognition by statistical methods’, *Proceedings of the IEEE*, 64(4), pp. 532-556, 1976.
- [34] L. Liporace, ‘Maximum likelihood estimation for multivariate observations of Markov sources’, *IEEE Trans. Inf. Theory*, vol. 28, no. 5, pp. 729–734, Sep. 1982.
- [35] L. R. Bahl, F. Jelinek, and R. L. Mercer, ‘A maximum likelihood approach to continuous speech recognition’, *IEEE transactions on pattern analysis and machine intelligence*, pp. 179-190, 1983.
- [36] P. Smolensky, *Information processing in dynamical systems: Foundations of harmony theory*. 1986.
- [37] F. Seide, G. Li, X. Chen, and D. Yu, ‘Feature engineering in Context-Dependent Deep Neural Networks for conversational speech transcription’, in *2011 IEEE Workshop on Automatic Speech Recognition & Understanding*, Waikoloa, HI, USA, 2011.
- [38] F. Seide, G. Li, and D. Yu, ‘Conversational speech transcription using context-dependent deep neural networks’, in *Twelfth annual conference of the international speech communication association*, 2011.
- [39] K. Zinchenko, C.-Y. Wu, and K.-T. Song, ‘A study on speech recognition control for a surgical robot’, *IEEE Trans. Industr. Inform.*, vol. 13, no. 2, pp. 607–615, Apr. 2017.

- [40] Baevski, Alexei, et al, ‘wav2vec 2.0: A framework for self-supervised learning of speech representations.’, *Advances in neural information processing systems* 33, 2020.
- [41] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, ‘BERT: Pre-training of deep bidirectional Transformers for language understanding’, *arXiv*, 10-Oct-2018.
- [42] D. Hendrycks and K. Gimpel, ‘Gaussian Error Linear Units (GELUs)’, *arXiv [cs.LG]*, 27-Jun-2016.
- [43] Andrews, L. C. ‘Special functions of mathematics for engineers’, *Spie Press*, Vol. 49, pp. 110, 1998.
- [44] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, ‘Roberta: A robustly optimized bert pretraining approach’, *arXiv preprint arXiv:1907.11692*, 2019.
- [45] R. Ardila *et al.*, ‘Common Voice: A massively-multilingual speech corpus’, *arXiv [cs.CL]*, 13-Dec-2019.