

ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG ĐIỆN - ĐIỆN TỬ



ĐỒ ÁN TỐT NGHIỆP

**THIẾT KẾ ỨNG DỤNG SOẠN THẢO VĂN BẢN TIẾNG
VIỆT TỰ ĐỘNG TRÊN SMARTPHONE SỬ DỤNG CÔNG
NGHỆ NHẬN DẠNG TIẾNG NÓI**

LƯU ĐÌNH TÚ
tu.ld213016@sis.hust.edu.vn

Ngành Kỹ thuật Điều khiển và Tự động hóa

Giảng viên hướng dẫn: TS. Trần Thị Anh Xuân

Chữ ký của GVHD

Khoa: Tự động hóa
Trường: Điện - Điện tử

Hà Nội, 07/2025

NHIÊM VỤ
ĐỒ ÁN TỐT NGHIỆP

Họ và tên sinh viên: Lưu Đình Tú

Khóa: K66

Trường: Điện – Điện tử

Ngành: KT ĐK & TĐH

1. *Tên đề tài*

Thiết kế ứng dụng soạn thảo văn bản tiếng Việt tự động trên smartphone sử dụng công nghệ nhận dạng tiếng nói

2. *Nội dung đề tài*

- Nghiên cứu, xây dựng và triển khai hệ thống nhận dạng tiếng nói tiếng Việt trong môi trường tiếng nói sạch
- Tìm hiểu, thiết kế phần mềm ứng dụng soạn thảo văn bản tiếng tiếng Việt trên smartphone có hệ điều hành Android phục vụ chỉ cho chức năng là soạn thảo văn bản tự động
- Tích hợp phần mềm nhận dạng tiếng nói tiếng Việt trong môi trường tiếng nói sạch vào phần mềm ứng dụng soạn thảo văn bản tiếng Việt trên smartphone (Android)
- Thử nghiệm và đánh giá kết quả

3. *Thời gian giao đề tài:* 28/02/2025

4. *Thời gian hoàn thành:* 27/06/2025

Ngày 27 tháng 06 năm 2025

CÁN BỘ HƯỚNG DẪN

Trần Thị Anh Xuân

LỜI CẢM ƠN

Con xin gửi lời cảm ơn sâu sắc đến gia đình – những người luôn là chỗ dựa tinh thần vững chắc, đã hết lòng ủng hộ, tạo điều kiện thuận lợi để con có tâm lý thoải mái trong suốt quá trình học tập và thực hiện đồ án. Sự động viên và khích lệ từ gia đình là nguồn sức mạnh to lớn giúp con vượt qua những lúc khó khăn, bế tắc.

Em xin bày tỏ lòng biết ơn chân thành tới cô Trần Thị Anh Xuân – người đã đồng hành và hướng dẫn em không chỉ trong đồ án tốt nghiệp mà còn từ Đồ án I, Đồ án II trước đó. Trong suốt quá trình thực hiện đề tài, cô luôn tận tình chỉ dẫn, sát sao trong việc định hướng chuyên môn, góp ý chi tiết giúp em điều chỉnh và hoàn thiện đề tài một cách đúng hướng và khoa học. Những nhận xét và góp ý từ cô không chỉ giúp em nâng cao chất lượng đồ án, mà còn rèn luyện cho em tác phong làm việc nghiêm túc, chỉn chu và cầu thị hơn trong quá trình nghiên cứu.

Em cũng xin gửi lời cảm ơn tới Đại học Bách Khoa Hà Nội – ngôi trường em hằng mơ ước từ khi còn nhỏ, nơi đã mang đến cho em môi trường học tập năng động, chuyên nghiệp cùng nhiều cơ hội phát triển bản thân và được làm việc, học hỏi cùng những con người tài năng.

Cuối cùng, em xin gửi lời cảm ơn chân thành đến tất cả những người bạn thân thiết đã luôn đồng hành, chia sẻ và động viên em trong suốt quá trình học tập và thực hiện đề tài. Được bao quanh bởi những người bạn giỏi giang, nhiệt huyết không chỉ là một may mắn mà còn là nguồn động lực to lớn giúp em không ngừng cố gắng và phát triển bản thân.

Em xin trân trọng cảm ơn!

Hà Nội, ngày 26 tháng 06 năm 2025

Sinh viên thực hiện

Lưu Đình Tú

TÓM TẮT ĐỒ ÁN

Đồ án tập trung xây dựng hệ thống nhận dạng tiếng nói tiếng Việt hoạt động ở cả hai chế độ: offline và online (Streaming ASR), có khả năng triển khai thực tế trên nền tảng Android. Bài toán đặt ra là phát triển một hệ thống nhận dạng tiếng nói có độ chính xác cao, độ trễ thấp, có thể vận hành trên thiết bị Android tầm trung.

Hệ thống sử dụng mô hình học sâu kết hợp giữa kiến trúc acoustic model (CTC-based) và mô hình ngôn ngữ n-gram (KenLM) để cải thiện độ chính xác giải mã. Dữ liệu huấn luyện bao gồm các tập VLSP 2020, VIVOS, FPT và dữ liệu sinh viên tự thu. Quá trình huấn luyện thực hiện với TensorFlow, tối ưu hóa tốc độ học bằng linear decay. Ứng dụng triển khai sử dụng Android Studio, mô hình được chuyển sang định dạng TensorFlow Lite.

Kết quả đạt được cho thấy WER trên các tập test dưới 10% khi sử dụng mô hình ngôn ngữ, đảm bảo yêu cầu về độ chính xác. Ứng dụng trên Android hoạt động ổn định, hỗ trợ chế độ Online và xử lý tốt trong môi trường ít nhiễu.

Đồ án thể hiện rõ tính thực tế, có khả năng mở rộng như tích hợp bộ lọc nhiễu, tăng dữ liệu tiếng nói tự thu và cải thiện giao diện người dùng. Qua đồ án, sinh viên đã nâng cao kiến thức về học sâu, xử lý tiếng nói, triển khai hệ thống trên thiết bị di động và kỹ năng lập trình thực tế..

MỤC LỤC

DANH MỤC KÝ HIỆU VÀ CHỮ VIẾT TẮT	i
DANH MỤC HÌNH VẼ	iv
DANH MỤC BẢNG BIỂU	vi
CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI	1
1.1 Tổng quan hệ thống nhận dạng tiếng nói và ứng dụng	1
1.1.1 Thế nào là hệ thống nhận dạng tiếng nói	1
1.1.2 Ứng dụng của nhận dạng tiếng nói	2
1.2 Mục tiêu và phạm vi đề tài	4
1.2.1 Phạm vi đề tài	4
1.2.2 Mục tiêu	4
1.3 Cấu trúc quyển	5
1.4 Kết luận chương	5
CHƯƠNG 2. CƠ SỞ LÝ THUYẾT	6
2.1 Tổng quan hệ thống nhận dạng tiếng nói	6
2.1.1 Pha huấn luyện	6
2.1.2 Pha nhận dạng	7
2.1.3 Phân loại hệ thống nhận dạng tiếng nói	7
2.2 Những khó khăn của hệ thống nhận dạng tiếng nói liên tục	8
2.3 Khối trích xuất đặc trưng	9
2.3.1 Trích xuất đặc trưng Mel-spectrogram	9
2.3.2 Trích xuất đặc trưng Mel frequency cepstral coefficients (MFCC) .	15
2.4 Thuật toán huấn luyện nhận dạng tiếng nói	16
2.4.1 Convolution neural network	16
2.4.2 Residual Connection	17
2.4.3 Thuật toán Transformer	18
2.5 Connectionist Temporal Classification (Phân loại thời gian kết nối) .	24
2.5.1 Lý do sử dụng CTC	24
2.5.2 Đầu ra theo thời gian và đường dẫn xác suất trong CTC . . .	25

2.5.3	Hàm măt măt CTC	26
2.5.4	Giải mă CTC (CTC Decoding)	26
2.6	Mô hình ngôn ngữ (Language model)	26
2.6.1	Mô hình n-gram truyền thông [1]	27
2.6.2	Mô hình ngôn ngữ học sâu (Neural Language Model)	29
2.6.3	Sử dụng mô hình ngôn ngữ với KenLM	29
2.7	Khối giải mă	30
2.7.1	Giải mă GReedy	30
2.7.2	Giải mă Beam Search	33
2.8	Hệ điều hành Android	36
2.8.1	Giới thiệu hệ điều hành Android	36
2.8.2	Một số phần mềm phổ biến để lập trình Android [2]	37
2.9	Kết luận chương	38
CHƯƠNG 3. THIẾT KẾ HỆ THỐNG		39
3.1	Đề xuất sơ đồ khói hệ thống	39
3.2	Thiết kế phần mềm nhận dạng tiếng nói tiếng Việt liên tục	39
3.2.1	Đề xuất sơ đồ khói của phần mềm nhận dạng tiếng nói tiếng Việt liên tục	39
3.2.2	Lựa chọn trích xuất đặc trưng tiếng nói và triển khai	40
3.2.3	Khảo sát và lựa chọn kiến trúc mô hình huấn luyện	42
3.2.4	Mô hình Squeezeformer	46
3.2.5	Xây dựng mô hình ngôn ngữ	53
3.3	Thiết kế phần mềm ứng dụng soạn thảo văn bản tiếng Việt sử dụng công nghệ nhận dạng tiếng nói	55
3.3.1	Công cụ sử dụng	55
3.3.2	Thiết kế giao diện ứng dụng soạn thảo văn bản	57
3.3.3	Tích hợp hệ thống nhận dạng tiếng nói tiếng Việt vào ứng dụng hiển thị văn bản trên smartphone	58
3.4	Thiết kế hệ thống nhận dạng tiếng nói tiếng Việt online	61
3.4.1	Lý do sử dụng	61
3.4.2	Thuật toán Local Agreement	62
3.5	Kết luận chương	63
CHƯƠNG 4. THỬ NGHIỆM VÀ ĐÁNH GIÁ KẾT QUẢ		64

4.1	Cơ sở dữ liệu	64
4.1.1	Vietnamese Language and Speech Processing (VLSP) 2020 [3]	64
4.1.2	FPT Open Speech Dataset	65
4.1.3	VIVOS	65
4.1.4	Bộ dữ liệu tự thu (CStt - Cơ sở tự thu)	65
4.2	Tiền xử lý cơ sở dữ liệu	66
4.2.1	Các phương pháp tiền xử lý	66
4.3	Phân chia dữ liệu sử dụng trong huấn luyện và đánh giá kết quả	67
4.3.1	Sử dụng tập dữ liệu VLSP 2020	68
4.3.2	Sử dụng tập dữ liệu VIVOS	68
4.3.3	Sử dụng tập dữ liệu FPT	68
4.4	Một số thông số đánh giá chất lượng hệ thống	69
4.4.1	Word Error Rate (WER)	69
4.4.2	Real-Time Factor (RTF)	69
4.4.3	Latency (Độ trễ)	70
4.5	Thông số huấn luyện	70
4.6	Thử nghiệm 1: Đánh giá chất lượng của mô hình âm học	71
4.6.1	Mục tiêu thử nghiệm	71
4.6.2	Cách tiến hành	71
4.6.3	Phương án 1: Sử dụng 128 lớp đầu ra cho mô hình âm học . .	72
4.6.4	Phương án 2: Sử dụng 93 lớp đầu ra cho mô hình âm học và mở khóa khối 6 thay cho khối 13 trong encoder	78
4.7	Thử nghiệm 2: Đánh giá vai trò kích thước dữ liệu huấn luyện đối với hệ thống nhận dạng tiếng nói	82
4.8	Thử nghiệm 3: Thử nghiệm tối ưu hóa tham số Alpha và Beta trong Beam Search kết hợp Language Model	87
4.8.1	Mục tiêu thử nghiệm	87
4.8.2	Cách tiến hành	87
4.8.3	Nhận xét:	89
4.8.4	Kết luận	89
4.9	Thử nghiệm 4: Đánh giá vai trò mô hình ngôn ngữ trong hệ thống nhận dạng tiếng nói	89
4.9.1	Mục tiêu thử nghiệm	89
4.9.2	Phương pháp tiến hành	89

4.9.3	Kết quả WER trên các tập dữ liệu khi tích hợp language model 4-gram	90
4.9.4	Nhận xét	91
4.10	Thử nghiệm 5: Đánh giá khả năng tích hợp mô hình âm học vào ứng dụng hiển thị văn bản tiếng Việt trên Android	91
4.11	Thử nghiệm 6: Đánh giá hiệu quả chế độ nhận dạng tiếng nói trực tuyến (Online ASR)	93
4.11.1	Mục đích thử nghiệm	93
4.11.2	Phương pháp tiến hành	93
4.11.3	Trên máy tính xách tay (Laptop)	94
4.11.4	Trên thiết bị Android	95
4.12	Đánh giá chất lượng hệ thống nhận dạng tiếng nói tiếng Việt với các thông số RTF và Latency	96
4.12.1	Hệ thống hoạt động trên máy tính xách tay (laptop)	96
4.12.2	Triển khai hệ thống trên ứng dụng Android	98
4.13	Kết luận chương	100
KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN		101
Kết luận	101	
Hạn chế của hệ thống	101	
Hướng phát triển	101	
TÀI LIỆU THAM KHẢO		103
PHỤ LỤC		106
Phụ lục 1: Thư viện KenLM	106	
Phụ lục 2: Vietnamese Corpus Text	106	
Phụ lục 3: Ngôn ngữ Kotlin	107	
Phụ lục 4: Fine-tuning	107	
Phụ lục 5: Biến đổi Fourier	107	
Phụ lục 6: Tensorflow Lite	108	

DANH MỤC KÝ HIỆU VÀ CHỮ VIẾT TẮT

STT	Ký hiệu / Chữ viết tắt	Ý nghĩa
1	STFT	Short-time Fourier Transform
2	ASR	Automatic Speech Recognition
3	seq2seq	Sequence to Sequence
4	MHA	Multi-head Attention
5	ReLU	Rectified Linear Unit
6	WER	Word Error Rate
7	RTF	Real-Time Factor
8	CTC	Connectionist Temporal Classification
9	FSM	Finite State Machine
10	AM	Acoustic Model
11	LM	Language Model
12	IDE	Integrated Development Environment

DANH MỤC HÌNH VẼ

Hình 1.1. Những ứng dụng của nhận diện tiếng nói trong thực tế	2
Hình 2.1. Sơ đồ khái hoạt động hệ thống nhận dạng tiếng nói	6
Hình 2.2. Hoạt động trích xuất Mel-spectrogram từ tín hiệu tiếng nói thô .	9
Hình 2.3. Hình ảnh trực quan về quá trình windowing và áp dụng biến đổi Fourier cho tín hiệu âm thanh	12
Hình 2.4. Áp dụng biến đổi Fourier rời rạc	13
Hình 2.5. Tạo ra các bands	14
Hình 2.6. Bộ lọc mel	14
Hình 2.7. Hoạt động trích xuất MFCC từ tín hiệu tiếng nói thô	15
Hình 2.8. Kết nối tắt trong mạng phần dư	18
Hình 2.9. Tạo ra 3 vector query, key và value cho từng từ.	20
Hình 2.10. Tính score cho từng từ trong câu đầu vào.	20
Hình 2.11. Tính scaling và softmax của score	21
Hình 2.12. Tính toán đầu ra của self-attention	21
Hình 2.13. Tính toán ma trận đầu ra self-attention	22
Hình 2.14. Tính toán các ma trận Query, Key và Value	22
Hình 2.15. Tính toán ma trận đầu ra cho từng đầu trong multi-head attention	23
Hình 2.16. Tính toán các ma trận Query, Key và value cho từng đầu . . .	23
Hình 2.17. Hợp nhất đầu ra các đầu trong multi-head attention	24
Hình 2.18. Cộng thêm positional encoding vào đầu vào nhúng	24
Hình 2.19. Mô hình hóa nhãn chuỗi bằng FSM trong CTC	26
Hình 2.20. Ví dụ về thuật toán giải mã Greedy	31
Hình 2.21. Ví dụ không tối ưu về giải mã Greedy	32
Hình 2.22. Ví dụ thuật toán beam search với beam width = 3	35
Hình 3.1. Sơ đồ khái hoạt động của toàn bộ hệ thống	39
Hình 3.2. Thiết kế hệ thống nhận dạng tiếng nói tiếng Việt	40
Hình 3.3. Luồng hoạt động của trích xuất Mel-spectrogram từ tín hiệu tiếng nói thô	41
Hình 3.4. Kết quả của wav2vec khi tinh chỉnh trên 960h dữ liệu có nhãn Librispeech	43
Hình 3.5. Kiến trúc mô hình Squeezeformer	47
Hình 3.6. Kiến trúc khối SqueezeFormer, bao gồm hai nhánh CF và MF .	50

Hình 3.7. Cấu trúc module multi-head attention trong SqueezeFormer	51
Hình 3.8. Cấu trúc module feed-forward neural network trong SqueezeFormer	51
Hình 3.9. Cấu trúc module Convolution trong SqueezeFormer	52
Hình 3.10. Môi trường phát triển Android Studio	56
Hình 3.11. Giao diện ứng dụng hiển thị văn bản tiếng Việt	57
Hình 3.12. Đoạn mã thiết kế giao diện ứng dụng hiển thị văn bản tiếng Việt	58
Hình 3.13. Quy trình chuyển đổi mô hình đã huấn luyện sang mô hình định dạng .tflite	59
Hình 3.14. Khởi tạo ứng dụng trên IDE Android Studio	60
Hình 3.15. Di chuyển mô hình định dạng .tflite vào Android Studio . . .	60
Hình 3.16. Mã mẫu tạo dữ liệu đầu vào đúng với kích thước mô hình yêu cầu	61
Hình 3.17. Phương pháp xác nhận từ giữa nhiều phân đoạn	62
Hình 4.1. Phân bố thời lượng tập dữ liệu tự thu	66
Hình 4.2. Thiết lập tốc độ học	71
Hình 4.3. Biểu đồ hàm loss sau 90 epoch khi chỉ unfreeze khối decoder . .	73
Hình 4.4. Biểu đồ WER theo epoch trên tập Valid (Phương án 1 - Trường hợp 1).	73
Hình 4.5. Biểu đồ hàm loss sau 90 epoch	74
Hình 4.6. Biểu đồ WER theo số epoch trên tập validation (Phương án 1 - Trường hợp 2)	75
Hình 4.7. Biểu đồ hàm loss sau 60 epoch	76
Hình 4.8. Biểu đồ WER theo số epoch trên tập validation (Phương án 1 - Trường hợp 3)	76
Hình 4.9. Biểu đồ hàm loss sau 90 epoch	77
Hình 4.10. Biểu đồ WER theo số epoch trên tập validation (Phương án 1 - Trường hợp 4)	78
Hình 4.11. Ảnh hưởng của số tham số học được đến WER trên tập Valid .	79
Hình 4.12. Kết quả huấn luyện sau 135 epoch trên tập Speaker + VIVOS .	80
Hình 4.13. Kết quả sau 200 epochs	83
Hình 4.14. Kết quả huấn luyện sau 175 epoch với dữ liệu mở rộng	84
Hình 4.15. Kết quả sau 175 epochs Thử nghiệm 2 - trường hợp 3	86
Hình 4.16. Cấu trúc tích hợp mô hình KenLM vào Decoder	88
Hình 4.17. Quy trình xử lý và kết quả nhận dạng trên ứng dụng Android .	92
Hình 4.18. Giai đoạn đầu: Hệ thống nhận dạng được từ khóa đầu tiên "hôm nay"	94

Hình 4.19. Giai đoạn tiếp theo: Nhận dạng mở rộng thành "hôm nay là thứ hai trời nắng"	94
Hình 4.20. Tiếp tục suy diễn: Hệ thống hiển thị "ngày mai là thứ ba trời" . .	94
Hình 4.21. Cuối cùng: Câu hoàn chỉnh được nhận dạng đầy đủ	94
Hình 4.22. Quy trình xử lý và kết quả nhận dạng chê độ Online ASR trên ứng dụng Android	95
Hình 4.23. Cấu trúc khung của TensorFlow lite	108

DANH MỤC BẢNG BIỂU

Bảng 3.1. Kết quả WER của wav2vec trên tập dữ liệu tiếng Việt [4]	43
Bảng 3.2. Các phiên bản của Whisper trong nghiên cứu [5, 6]	44
Bảng 3.3. So sánh Word Error Rate (WER) giữa các mô hình ASR trên tiếng Việt[6]	44
Bảng 3.4. So sánh hiệu suất của các phiên bản Squeezeformer trên tập LibriSpeech [7]	45
Bảng 4.1. Tổng quan về các thư mục dữ liệu gốc VLSP 2020	64
Bảng 4.2. Phân chia tập dữ liệu VLSP 2020	68
Bảng 4.3. Tổng quan về tập dữ liệu VIVOS	68
Bảng 4.4. Phân chia tập dữ liệu FPT	69
Bảng 4.5. Dữ liệu Speaker (VLSP 2020) và VIVOS được sử dụng trong thử nghiệm 1	72
Bảng 4.6. Kết quả WER (%) trên Speaker + VIVOS (Thử nghiệm 1 - Phương án 1 - Trường hợp 3)	76
Bảng 4.7. Kết quả WER (%) trên Speaker + VIVOS (Thử nghiệm 1 - Phương án 1 - Trường hợp 4)	78
Bảng 4.8. Kết quả WER (%) trên Speaker + VIVOS (Thử nghiệm 1- Phương án 2)	81
Bảng 4.9. Thông tin chia tập dữ liệu VLSP, VIVOS (Thử nghiệm 2 - Trường hợp 1)	82
Bảng 4.10. Kết quả WER (%) trên Speaker + VIVOS (Thử nghiệm 2 - trường hợp 1)	82
Bảng 4.11. Thông tin chia tập dữ liệu sử dụng VLSP, VIVOS, FPT trong thử nghiệm 2 - trường hợp 2	84
Bảng 4.12. Kết quả WER (%) trên các tập dữ liệu (Thử nghiệm 2 - Trường hợp 2)	85
Bảng 4.13. Thông tin chia tập dữ liệu Database (VLSP 2020)	86
Bảng 4.14. Kết quả WER trên các tập dữ liệu không sử dụng mô hình ngôn ngữ (Thử nghiệm 2- Trường hợp 3)	86
Bảng 4.15. Kết quả thử nghiệm các giá trị Alpha và Beta ảnh hưởng đến WER (300 files)	88
Bảng 4.16. WER trên Speaker + VIVOS với LM 4-gram (TN1-PA2)	90
Bảng 4.17. WER trên Speaker + VIVOS với LM 4-gram (TN2-TH1)	90

Bảng 4.18. WER trên Speaker + VIVOS với LM 4-gram (TN2 – TH2)	90
Bảng 4.19. WER trên các tập dữ liệu với LM 4-gram (TN2 – TH3)	91
Bảng 4.20. Đánh giá Latency và RTF cho các câu nói offline (giới hạn 8 giây)	96
Bảng 4.21. Thống kê Latency và RTF qua 7 lần đo	97
Bảng 4.22. Đánh giá Latency và RTF ở chế độ offline trên các thiết bị Android	99
Bảng 4.23. Đánh giá Latency và RTF ở chế độ online trên các thiết bị Android	100

CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI

Chương này cung cấp cái nhìn tổng quan về nhận dạng tiếng nói, bắt đầu từ định nghĩa cơ bản và quá trình hình thành lĩnh vực này qua từng giai đoạn phát triển công nghệ. Tiếp theo, các ứng dụng thực tế nổi bật sẽ được trình bày nhằm làm rõ vai trò thiết yếu của nhận dạng tiếng nói trong cuộc sống hiện đại. Ngoài ra, chương cũng khái quát định hướng nghiên cứu hiện nay, từ đó dẫn đến mục tiêu và phạm vi của đề tài. Cuối chương, các đầu việc được sắp xếp theo mức độ ưu tiên nhằm đảm bảo hiệu quả triển khai và phù hợp với định hướng xây dựng hệ thống thực tế.

1.1 Tổng quan hệ thống nhận dạng tiếng nói và ứng dụng

Trong thời đại công nghệ số hiện nay, việc tương tác giữa con người và máy tính ngày càng trở nên tự nhiên và thuận tiện hơn, trong đó nhận dạng tiếng nói (Automatic Speech Recognition - ASR) đóng vai trò then chốt. Khả năng chuyển đổi lời nói thành văn bản không chỉ mang lại sự tiện lợi trong giao tiếp mà còn mở ra nhiều hướng ứng dụng đa dạng trong các lĩnh vực như trợ lý ảo, dịch thuật thời gian thực, tự động hóa dịch vụ khách hàng, và các thiết bị điều khiển bằng giọng nói.

1.1.1 Thể nào là hệ thống nhận dạng tiếng nói

Nhận dạng tiếng nói (hay còn được gọi là nhận dạng tiếng nói tự động Automatic Speech Recognition - ASR) là quá trình chuyển đổi tín hiệu tiếng nói thành một chuỗi các từ tương ứng bằng thuật toán được thực hiện như một chương trình máy tính [8].

1.1.1.1 Hệ thống nhận dạng tiếng nói cơ bản

Lời nói là hình thức tự nhiên nhất của giao tiếp, xử lý tiếng nói của con người cũng là một trong những lĩnh vực rất được các nhà nghiên cứu quan tâm. Công nghệ nhận dạng tiếng nói xuất hiện đã giúp máy tính, điện thoại tuân theo các lệnh giọng nói của con người và hiểu được ngôn ngữ con người. Vì mong muốn có cơ chế thực hiện tự động máy móc, vận dụng khả năng ngôn luận của con người để tự động hóa các nhiệm vụ đơn giản đòi hỏi tương tác máy của con người nghiên cứu về nhận dạng tiếng nói tự động đã thu hút rất nhiều sự quan tâm trong hơn nửa thập kỷ. Dựa trên những tiến bộ của nghiên cứu về nhận dạng tiếng nói, chúng đã xuất hiện rộng rãi trong nhiều lĩnh vực yêu cầu giao tiếp giữa người với máy, chẳng hạn như xử lý cuộc gọi tự động trong điện thoại, trợ lý ảo, các hệ thống tự động giải đáp thắc mắc như Chat GPT, Gemini,... Nhưng vẫn còn những vấn đề với nhận dạng tiếng nói cần cải thiện hiện nay như: sự đa dạng vùng miền với một ngôn ngữ, ngôn ngữ ít ngữ liệu, những các phát âm đặc trưng vùng miền,...

1.1.2 Ứng dụng của nhận dạng tiếng nói

Nhận dạng tiếng nói là một trong những thành phần quan trọng trong lĩnh vực trí tuệ nhân tạo, ngày càng được ứng dụng rộng rãi trong nhiều lĩnh vực của đời sống xã hội và sản xuất công nghiệp. Với khả năng chuyển đổi lời nói thành văn bản một cách nhanh chóng và chính xác, công nghệ này đã mở ra nhiều hướng phát triển mới cho các hệ thống thông minh, đặc biệt trong các ứng dụng phục vụ con người. Trong phần này, sinh viên trình bày một số ứng dụng tiêu biểu của nhận dạng tiếng nói cả trong nước và trên thế giới.



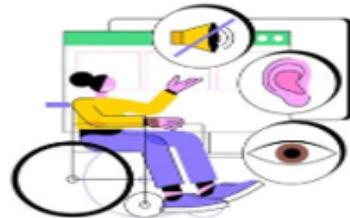
(a) Trợ lý ảo thông minh



Chuyển đổi giọng nói
(b) thành văn bản



(c) Dịch vụ khách hàng



(d) Hỗ trợ người khuyết tật

Hình 1.1. Những ứng dụng của nhận diện tiếng nói trong thực tế

1.1.2.1 Trong nước [9]

Tại Việt Nam, nhận dạng tiếng nói đã được nghiên cứu và phát triển bởi nhiều tập đoàn công nghệ lớn nhằm phục vụ các mục tiêu như nâng cao trải nghiệm người dùng, tự động hóa dịch vụ khách hàng và hỗ trợ các đối tượng đặc biệt trong xã hội.

a, Trợ lý ảo tiếng Việt

Hiện nay, các tập đoàn lớn ở Việt Nam như Zalo, Viettel, FPT.. cũng đã nghiên cứu và phát triển hệ thống trợ lý ảo tự động nhằm phục vụ cho hoạt động chăm sóc khách hàng như: cho phép chuyển đổi lời nói thoại của người dùng sang nội dung text với độ chính xác lên đến 95% và khả năng chống ồn tốt, công nghệ xử lý ngôn ngữ tự nhiên,... Ví dụ thực tế: Viettel Cyber Callbot là tổng đài đầu tiên ở Việt Nam trả lời bằng tiếng Việt, ứng dụng AI do kỹ sư trong nước phát triển.

b, Chuyển đổi tiếng nói thành văn bản

Những đơn vị tiên phong trong lĩnh vực chuyển đổi tiếng nói thành văn bản trong

tiếng Việt có thể kể đến như: Zalo, Vbee, FPT. AI với các ứng dụng phổ biến: chuyển đổi âm thanh trong video/ thuyết trình thành văn bản, hỗ trợ người khiếm thính, ghi biên bản tự động trong cuộc họp.

1.1.2.2 *Nước ngoài* [10]

Trên thế giới, các tập đoàn công nghệ lớn như Google, Amazon, Microsoft,... đã tích hợp công nghệ nhận dạng tiếng nói vào nhiều sản phẩm và dịch vụ, mang lại sự tiện lợi đáng kể cho người dùng. Sinh viên tóm tắt một số ứng dụng tiêu biểu như sau:

a, **Tìm kiếm thông tin bằng giọng nói**

Đây có thể được coi là ứng dụng phổ biến nhất của nhận dạng tiếng nói. Ví dụ thực tế: Khoảng 20% dân số toàn cầu sử dụng tìm kiếm bằng giọng nói, trong khi số lượng trợ lý giọng nói được dùng vào khoảng 8 tỷ, vượt quá tổng dân số thế giới. Tại Hoa Kỳ, 153 triệu cá nhân phụ thuộc vào trợ lý giọng nói, trong đó Google Assistant dự kiến là ứng dụng phổ biến nhất, đạt 92 triệu người dùng vào năm 2025. Các câu hỏi điển hình từ người dùng như "Này Google, thời tiết hôm nay như nào, tôi có thông báo nào không?" sau đó các thiết bị thông tin sẽ truy cập thông tin trực tuyến để đưa ra kết quả chính xác cho người dùng.

b, **Điều khiển các thiết bị trong nhà bằng tiếng nói**

Các thiết bị nhà thông minh sử dụng công nghệ nhận dạng tiếng nói để tự động hóa các công việc gia đình, chẳng hạn như bật đèn, đun nước, điều chỉnh nhiệt độ,...
Ví dụ thực tế:

Amazon Alexa được kết nối với hơn 400 triệu thiết bị nhà thông minh, mà khách hàng sử dụng nhiều lần trong tuần. Alexa hiện hỗ trợ hơn 100.000 thiết bị, tăng từ chỉ 4000 thiết bị vào năm 2017. Người dùng có thể nói các lệnh như "Bật phòng ngủ", "Giảm độ sáng đèn văn phòng", "Bật đèn phòng khách trong năm phút" hoặc "Bật đèn bàn màu xanh lá cây để thay đổi màu sắc".

c, **Điều khiển các chức năng trong ô tô bằng tiếng nói**

Hệ thống nhận dạng tiếng nói trong xe hơi hiện là tiêu chuẩn trong hầu hết các loại xe hiện đại. Lợi ích của nhận dạng tiếng nói trên xe hơi là nó cho phép người lái giữ mắt trên đường và tay trên vô lăng. Các trường hợp sử dụng bao gồm khởi tạo cuộc điện thoại, chọn kênh radiom thiết lập chỉ đường và phát nhạc. Ví dụ thực tế: Tesla đã phát triển các bot tiếng nói cho phép người dùng quản lý khí hậu, giải trí và điều hướng thông qua các lệnh thoại như "Đặt nhiệt độ ở mức 72 độ" hoặc "Điều hướng đến [Điểm đến]"

d, **Hệ thống hỗ trợ người khiếm thị**

Theo nghiên cứu, 80% việc tiếp nhận kiến thức của trẻ em thường thông qua thị giác và động lực chính của chúng là khám phá môi trường xung quanh. Nhận dạng tiếng nói có thể tạo ra một nền tảng học tập công bằng cho trẻ em không có/ mất thị lực.

Ví dụ thực tế: Các công cụ học ngôn ngữ Duoilingo sử dụng nhận dạng tiếng nói để đánh giá cách phát âm ngôn ngữ của người dùng.

e, **Hỗ trợ trong hệ thống chăm sóc sức khỏe**

Trong quá trình khám bệnh, bác sĩ không cần phải lo lắng về việc ghi chép các triệu chứng của bệnh nhân. Ghi chép là một trong những hoạt động tốn nhiều thời gian nhất đối với bác sĩ, làm giảm khả năng khám bệnh nhận của họ. Nhờ công nghệ phiên âm y khoa, bác sĩ có thể giảm thời gian khám trung bình và do đó có thể sắp xếp nhiều bệnh nhân hơn trong lịch trình của họ.

Ví dụ thực tế: Sonda Health đã phát triển các ứng dụng di động cung cấp cho người dùng điểm số về "sức khỏe tinh thần" dựa trên tông giọng, cách dùng từ, năng lượng, sự dao động, nhịp điệu và các yếu tố khác. Nhận diện tiếng nói được sử dụng rộng rãi trong các ứng dụng thực tế ngày nay và đóng vai trò không nhỏ trong việc giúp các hệ thống hoạt động một cách tự động hóa cũng như giúp cuộc sống của con người trở lên tiện lợi và dễ dàng hơn. Dưới đây là những ứng dụng phổ biến của nhận diện tiếng nói trong cuộc sống.

1.2 Mục tiêu và phạm vi đề tài

1.2.1 Phạm vi đề tài

Đề tài được triển khai trong phạm vi cụ thể như sau:

- Ứng dụng được thiết kế và thử nghiệm trong môi trường yên tĩnh (sạch), ít nhiễu âm, phù hợp với điều kiện nhận dạng tiếng nói ổn định
- Ứng dụng được phát triển và triển khai trên thiết bị smartphone sử dụng hệ điều hành Android chỉ yêu cầu với giao diện đơn giản để mô phỏng hoạt động soạn thảo văn bản tự động thực tế
- Chỉ hỗ trợ nhận dạng giọng nói tiếng Việt chuẩn (chủ yếu là người miền Bắc), không bao gồm các phương ngữ địa phương hoặc giọng nói mang tính vùng miền đặc thù
- Chức năng chính giới hạn ở việc chuyển tiếng nói thành văn bản để soạn thảo văn bản cơ bản, chưa tích hợp các chức năng nâng cao như lệnh thoại hay xử lý ngôn ngữ tự nhiên

1.2.2 Mục tiêu

Đề tài được đề xuất để xây dựng một ứng dụng hỗ trợ soạn thảo văn bản tiếng Việt trên smartphone thông qua công nghệ nhận dạng tiếng nói, nhằm giúp người dùng nhập liệu tự động, thuận tiện, đặc biệt trong các tình huống không thuận lợi cho việc gõ phím. Cụ thể, đề tài tập trung vào các mục tiêu sau:

- Tìm hiểu, thiết kế ứng dụng di động trên hệ điều hành Android, với giao diện đơn giản, dễ sử dụng, phục vụ chỉ cho chức năng là soạn thảo văn bản tự động

- Tích hợp hệ thống nhận dạng tiếng nói tiếng Việt vào ứng dụng soạn thảo văn bản, cho phép chuyển tiếng nói tiếng Việt thành văn bản tự động
- Thủ nghiệm ứng dụng và đánh giá kết quả

1.3 Cấu trúc quyển

Để triển khai toàn bộ hệ thống nhận dạng tiếng nói tiếng Việt trên Android, sinh viên trình bày nội dung thực hiện trong quyển báo cáo theo cấu trúc như sau:

- Chương 1: Tổng quan đề tài
- Chương 2: Cơ sở lý thuyết. Trong chương này, sinh viên trình bày về tổng quan hệ thống nhận dạng tiếng nói, quy trình hoạt động hệ thống nhận dạng tiếng nói và hệ điều hành Android
- Chương 3: Thiết kế hệ thống. Trong chương này, sinh viên trình bày chi tiết về thiết kế hệ thống lần lượt từ thiết kế hệ thống nhận dạng tiếng nói tiếng Việt, thiết kế phần mềm ứng dụng trên Android, và hệ thống nhận dạng tiếng nói online.
- Chương 4: Thủ nghiệm và kết quả. Trong chương này, sinh viên trình bày về cơ sở dữ liệu sử dụng, thông số đánh giá chất lượng hệ thống, thông số huấn luyện, môi trường và công cụ. Đồng thời, sinh viên cũng sẽ trình bày toàn bộ thử nghiệm về hệ thống.
- Phần kết luận và hướng phát triển
- Phần phụ lục sinh viên trình bày các khái niệm, định nghĩa có dùng trong nội dung quyển đồ án một cách chi tiết hơn.

1.4 Kết luận chương

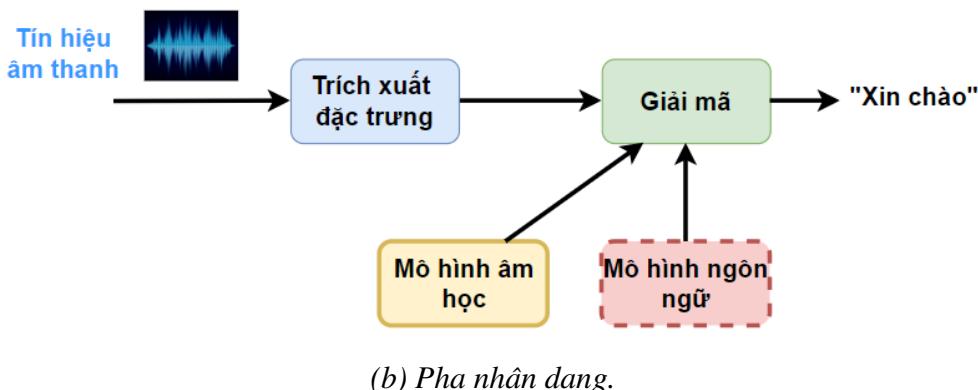
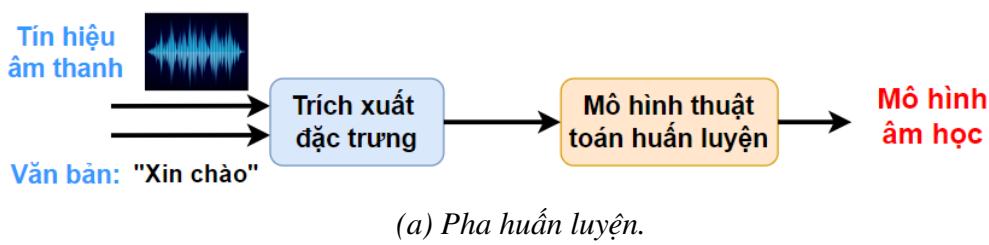
Trong chương này, sinh viên đã trình bày tổng quan về công nghệ nhận dạng tiếng nói và các ứng dụng thực tiễn của nó. Đồng thời, chương cũng nêu rõ mục tiêu, phạm vi của đề tài và cấu trúc của toàn bộ báo cáo. Ở chương 2, báo cáo sẽ đi sâu vào cơ sở lý thuyết của hệ thống nhận dạng tiếng nói, đồng thời trình bày các kiến thức liên quan đến hệ điều hành Android — nền tảng được sử dụng để triển khai ứng dụng trong đề tài.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

Trong Chương 1, sinh viên đã trình bày tổng quan về hệ thống nhận dạng tiếng nói, bao gồm định nghĩa, mục tiêu và phạm vi nghiên cứu của đề tài. Chương này sẽ trình bày các kiến thức nền tảng làm cơ sở cho việc xây dựng hệ thống nhận dạng tiếng nói. Trước hết, báo cáo tập trung mô tả chi tiết các thành phần chính trong một hệ thống nhận dạng tiếng nói, bao gồm: khôi tiền xử lý (pre-processing), khôi trích xuất đặc trưng (feature extraction), và bộ giải mã (decoder). Các nội dung này đóng vai trò quan trọng trong việc hiểu rõ quy trình xử lý tín hiệu âm thanh và triển khai hệ thống trong các chương sau.

2.1 Tổng quan hệ thống nhận dạng tiếng nói

Quá trình nhận dạng tiếng nói trong một hệ thống ASR (Automatic Speech Recognition) hiện đại thường được chia thành hai pha chính: **pha huấn luyện** và **pha nhận dạng**. Sơ đồ khái niệm hoạt động của hệ thống nhận dạng tiếng nói cụ thể như hình 3.2.



Hình 2.1. Sơ đồ khái niệm hoạt động hệ thống nhận dạng tiếng nói

2.1.1 Pha huấn luyện

Như thể hiện trong hình 3.2.a, dữ liệu đầu vào cần chuẩn bị bao gồm các tín hiệu âm thanh và văn bản chú thích tương ứng với từng tệp âm thanh. Dữ liệu này được đưa vào khôi trích xuất đặc trưng, sử dụng các kỹ thuật phổ biến như Mel-spectrogram hoặc MFCC (Mel-Frequency Cepstral Coefficients). Đầu ra của khôi trích xuất đặc

trưng là các đặc trưng quan trọng đại diện cho tín hiệu âm thanh đầu vào. Các đặc trưng này được sử dụng làm đầu vào cho thuật toán huấn luyện, nhằm học cách ánh xạ giữa đặc trưng âm thanh và văn bản tương ứng. Kết quả của quá trình này là một mô hình âm học, đóng vai trò then chốt trong pha nhận dạng.

2.1.2 Pha nhận dạng

Trong pha nhận dạng (Hình 2.1.b), đầu vào chỉ gồm tín hiệu âm thanh. Tín hiệu này được xử lý qua khối trích xuất đặc trưng tương tự như trong pha huấn luyện. Đầu ra của khối trích xuất đặc trưng sẽ được đưa vào mô hình âm học đã được huấn luyện trước đó. Mô hình âm học này sinh ra xác suất các ký tự (hoặc âm vị) tại mỗi bước thời gian.

Đầu ra xác suất từ mô hình âm học là đầu vào của khối giải mã (decoder). Khối này có thể hoạt động theo hai hướng chính:

- **Giải mã không sử dụng mô hình ngôn ngữ:** Các phương pháp như Greedy decoding hoặc Beam Search đơn thuần được sử dụng để tìm chuỗi ký tự có xác suất cao nhất.
- **Giải mã kết hợp với mô hình ngôn ngữ:** Beam Search được tích hợp với mô hình ngôn ngữ (Language Model - LM) để tận dụng thông tin ngữ cảnh, giúp cải thiện độ chính xác của chuỗi từ được nhận dạng.

Cuối cùng, đầu ra của khối giải mã là văn bản tương ứng với tín hiệu âm thanh đầu vào.

2.1.3 Phân loại hệ thống nhận dạng tiếng nói

Trong quá trình xây dựng hệ thống nhận dạng tiếng nói, việc phân loại hệ thống theo các tiêu chí khác nhau là cần thiết để lựa chọn phương pháp xử lý phù hợp với mục tiêu ứng dụng. Hai tiêu chí phân loại phổ biến nhất là: (1) cách thức người dùng phát ngôn (tức hình thức lời nói) và (2) cách hệ thống xử lý dữ liệu theo thời gian. Mỗi tiêu chí đều ảnh hưởng đến kiến trúc, độ phức tạp và độ chính xác của hệ thống.

2.1.3.1 Theo cách tiếng nói được nhận dạng

Phân loại theo hình thức lời nói giúp xác định mức độ tự nhiên và tính phức tạp trong quá trình nhận dạng. Dưới đây là bốn loại hình phổ biến:

- **Lời nói liên tục:** Các hệ thống nhận dạng này cho phép người dùng nói liên tục gần như tự nhiên, và máy tính sẽ tự động xác định nội dung được nói. Đây là loại hệ thống khó phát triển nhất vì cần sử dụng các kỹ thuật đặc biệt để phân biệt ranh giới giữa các từ trong chuỗi lời nói liền mạch.
- **Từ đơn:** Hệ thống nhận dạng từ đơn yêu cầu người dùng phát âm từng từ một cách rõ ràng, với khoảng dừng giữa các lần nói. Các hệ thống này thường hoạt

động theo hai trạng thái: “lắng nghe” và “không lắng nghe”, và chỉ xử lý tín hiệu trong các khoảng tạm dừng. Điều này giúp đơn giản hóa quá trình nhận dạng.

- **Từ khóa:** Các hệ thống nhận dạng từ khóa cho phép người dùng nói liền mạch hơn giữa các từ, không cần dừng rõ ràng sau từng từ như ở hệ thống từ đơn. Điều này giúp tạo cảm giác tự nhiên hơn, gần giống với cách nói thông thường hàng ngày.
- **Lời nói tự phát:** Đây là hình thức lời nói diễn ra một cách tự nhiên, không có sự chuẩn bị hoặc kịch bản từ trước. Hệ thống phải xử lý các câu nói đa dạng, thường gặp trong giao tiếp đời sống hàng ngày, với nhiều chủ đề và cách diễn đạt khác nhau.

2.1.3.2 *Theo cách hệ thống xử lý dữ liệu theo thời gian*

Phân loại theo cách xử lý dữ liệu giúp xác định khả năng đáp ứng thời gian thực của hệ thống. Có hai loại chính như sau:

- **Offline ASR (nhận dạng tiếng nói ngoại tuyến):** Hệ thống yêu cầu toàn bộ tín hiệu âm thanh phải được thu xong trước khi bắt đầu quá trình nhận dạng. Điều này cho phép sử dụng toàn bộ ngữ cảnh và thường cho độ chính xác cao hơn, nhưng không đáp ứng được yêu cầu thời gian thực.
- **Online ASR (nhận dạng tiếng nói trực tuyến):** Hệ thống xử lý tín hiệu âm thanh ngay khi nó đang được thu, cho phép nhận dạng theo thời gian thực với độ trễ thấp. Tuy nhiên, do không có toàn bộ ngữ cảnh phía sau, độ chính xác có thể thấp hơn so với hệ thống offline.

2.2 **Những khó khăn của hệ thống nhận dạng tiếng nói liên tục**

Hệ thống nhận dạng tiếng nói liên tục là một trong những dạng phức tạp nhất của bài toán nhận dạng tiếng nói, do đặc trưng đầu vào là dòng âm thanh không có ngắt quãng rõ ràng giữa các từ. Việc nhận dạng chính xác trong bối cảnh này đòi hỏi hệ thống phải xử lý nhiều thách thức về mặt ngôn ngữ, kỹ thuật và môi trường. Dưới đây là một số khó khăn tiêu biểu thường gặp trong quá trình xây dựng và triển khai hệ thống nhận dạng tiếng nói liên tục:

- **Thiếu thông tin căn chỉnh:** Trong nhận dạng tiếng nói liên tục, người nói có thể phát âm các từ một cách liền mạch, không ngắt quãng rõ ràng giữa các từ. Điều này gây khó khăn cho việc xác định ranh giới giữa các từ trong chuỗi âm thanh đầu vào, đặc biệt khi không có thông tin căn chỉnh thời gian (time-alignment) giữa lời nói và văn bản.

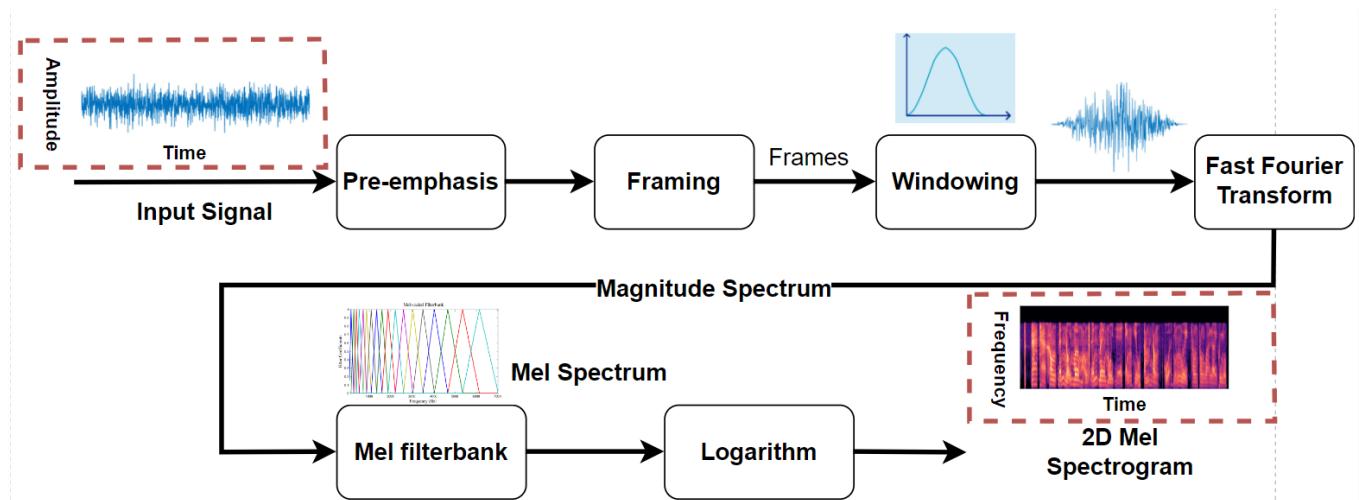
- **Đa dạng về phương ngữ và giọng nói:** Sự khác biệt về phát âm giữa các vùng miền, lứa tuổi, giới tính hoặc cá nhân khiến cho hệ thống khó tổng quát hóa tốt nếu không được huấn luyện với dữ liệu đủ đa dạng.
- **Nhiều và điều kiện thu âm không lý tưởng:** Hệ thống nhận dạng tiếng nói có thể bị ảnh hưởng mạnh bởi tiếng ồn môi trường, độ vang, tạp âm nền hoặc chất lượng thiết bị ghi âm, đặc biệt trong các ứng dụng thực tế.
- **Hiện tượng đồng hóa và nối âm:** Trong lời nói tự nhiên, các hiện tượng như đồng hóa, rút gọn âm hoặc nối âm xảy ra thường xuyên, làm cho tín hiệu âm thanh không còn khớp hoàn toàn với dạng từ chuẩn trong văn bản, từ đó gây khó khăn cho mô hình phân tích và nhận dạng.

2.3 Khôi trích xuất đặc trưng

Như đã trình bày trong sơ đồ khái hoạt động của hệ thống nhận dạng tiếng nói tại Mục 2.1, mỗi khái chức năng đều đóng vai trò quan trọng trong toàn bộ quá trình nhận dạng. Trong đó, khôi trích xuất đặc trưng có nhiệm vụ biến đổi tín hiệu âm thanh thành tập các đặc trưng mang thông tin ngữ âm quan trọng, giúp mô hình học máy dễ dàng xử lý hơn. Trong phần này, sinh viên trình bày cơ sở lý thuyết và nguyên lý hoạt động của hai phương pháp phổ biến nhất hiện nay, đó là Mel spectrogram và MFCC (Mel-Frequency Cepstral Coefficients).

2.3.1 Trích xuất đặc trưng Mel-spectrogram

Mel spectrogram là một phương pháp trực quan và hiệu quả để biểu diễn tín hiệu âm thanh trong miền tần số thời gian, đặc biệt phù hợp với đặc tính nghe của tai người.



Hình 2.2. Hoạt động trích xuất Mel-spectrogram từ tín hiệu tiếng nói thô

2.3.1.1 Vai trò Mel spectrogram

Ví dụ về nhận thức tai người, có 2 mẫu âm thanh:

- Mẫu âm thanh 1: C2 - C4 -> (65 - 262 Hz)
- Mẫu âm thanh 2: G6 - A6 -> (1568 - 1760 Hz)

Hai mẫu âm thanh tưởng chừng có khoảng cách về tần số tương tự nhau đều là 200 Hz nhưng theo cảm nhận của con người, việc phân biệt các tần số khác nhau ở âm thanh 2 sẽ dễ dàng hơn vì con người thường nhạy cảm với tần số cao hơn. Do đó, cảm nhận về cao độ của con người là không tuyến tính theo tần số, nhận thức về cao độ âm thanh của con người là theo logarit. Theo biểu đồ spectrogram thì tần số của âm thanh được biểu diễn tuyến tính theo thời gian, do vậy cần có thang đo khác để đánh giá về cao độ của âm thanh thay vì chỉ dùng thang đo về tần số như thông thường.

Điều kiện của một âm thanh lý tưởng trong học máy:

- Biểu diễn tần số thời gian
- Biểu diễn biên độ có liên quan đến nhận thức
- Biểu diễn tần số có liên quan đến nhận thức

Mel spectrogram có thể giải quyết được vấn đề này, tạo ra đầu vào lý tưởng cho cho các ứng dụng machine learning nói chung và nhận dạng tiếng nói nói riêng.

2.3.1.2 *Đặc tính của thang đo Mel*

Mel là viết tắt của “melody” nghĩa là giai điệu và có liên quan mật thiết đến cao độ và nhận thức âm thanh của con người, nó có các đặc tính đáng chú ý sau:

- Là thang đo theo logarit.
- Liên quan đến mặt nhận thức âm thanh, nghĩa là bằng nhau về khoảng cách trên thang đo sẽ có khả năng nhận thức là tương tự nhau.
- 1000 Hz tương ứng với 1000 trong thang đo Mel.

Công thức chuyển từ tần số sang thang đo mel:

$$m = 2595 \cdot \log \left(1 + \frac{f}{700} \right) \quad (2.1)$$

2.3.1.3 *Quy trình trích xuất đặc trưng Mel-spectrogram*

Quy trình trích xuất Mel-spectrogram gồm các bước chính: tiền nhấn (pre-emphasis), chia khung (framing), áp cửa sổ (windowing), biến đổi Fourier nhanh (FFT), lọc theo thang Mel và lấy log phổ. Dưới đây, sinh viên sẽ trình bày chi tiết từng bước.

Bước 1: Pre-emphasis – Bộ lọc tiền nhấn

Pre-emphasis là một kỹ thuật xử lý tín hiệu phổ biến được sử dụng trong bước đầu tiên của quá trình trích xuất đặc trưng. Mục đích chính của bộ lọc tiền nhấn là tăng cường thành phần tần số cao của tín hiệu âm thanh và giảm thiểu ảnh hưởng của các tần số thấp vốn có biên độ lớn hơn. Điều này giúp cải thiện tỷ lệ tín hiệu trên nhiễu (SNR), làm nổi bật đặc trưng âm thanh quan trọng và nâng cao hiệu quả của các bước xử lý sau:

- Tín hiệu âm thanh tự nhiên thường bị suy giảm năng lượng ở dải tần cao do tính chất vật lý của dây thanh và môi trường truyền âm.
- Ngoài ra, việc tăng cường tần số cao cũng giúp cân bằng phổ và cải thiện khả năng phân biệt giữa các âm tiết tương tự nhau.

Phổ tần sau khi áp dụng pre-emphasis sẽ được làm phẳng hơn, điều này có lợi cho việc trích xuất đặc trưng dựa trên tần số như FFT hay Mel.

Phương trình của bộ lọc tiền nhấn thường là một bộ lọc FIR bậc nhất:

$$y(t) = x(t) - \alpha \cdot x(t-1) \quad (2.2)$$

Trong đó:

- $x(t)$ là tín hiệu đầu vào tại thời điểm t
- $y(t)$ là tín hiệu đầu ra sau khi lọc
- α là hệ số tiền nhấn, thường nằm trong khoảng $[0.95, 0.97]$

Việc lựa chọn α ảnh hưởng đến mức độ khuếch đại của tần số cao. Nếu α quá lớn, tín hiệu có thể bị méo; nếu quá nhỏ, hiệu quả tiền xử lý sẽ giảm. Kết quả của bước này là tín hiệu đã được làm nổi bật các tần số cao, sẵn sàng để bước vào quá trình phân khung và áp dụng biến đổi Fourier.

Bước 2: Windowing

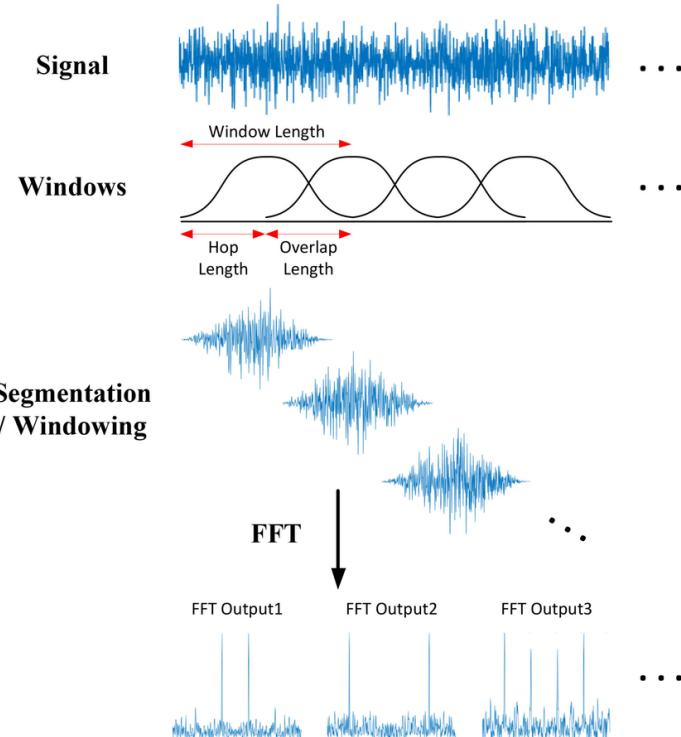
Một cách hiệu quả để tạo ra các phân đoạn được thể hiện như hình 2.3. là áp dụng hàm cửa sổ cho tín hiệu gốc để nhận được các phân đoạn. Dưới đây là công thức áp dụng windowing

$$x_m(k) = x(k) \cdot w(k) \quad (2.3)$$

Những tham số quan trọng khi áp dụng windowing trên hình 2.3.

- Window length: Là số lượng mẫu áp dụng windowing cho frame size. Thông thường window size được lấy chính bằng frame size.

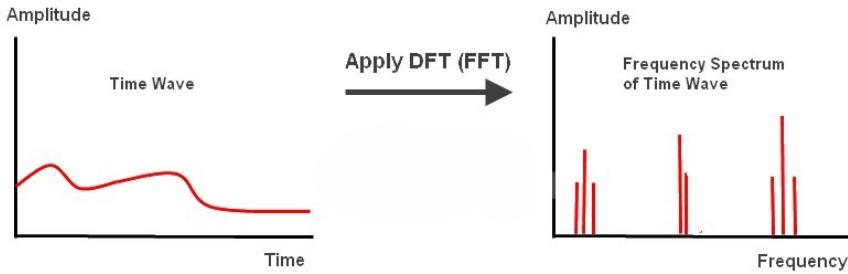
- Hop length: Cho biết có bao nhiêu sample sẽ trượt sang phải khi lấy một khung hình mới. Do các khung hình không thường được xếp độc lập mà thay vào đó chúng chồng chéo lên nhau để đảm bảo khắc phục được rò rỉ quang phổ cũng như tránh làm mất tín hiệu.



Hình 2.3. Hình ảnh trực quan về quá trình windowing và áp dụng biến đổi Fourier cho tín hiệu âm thanh

Bước 3: Trích xuất Fourier rời rạc

Như hình 2.4., khi biểu diễn Fourier rời rạc, ta thấy được sự hiện diện của các thành phần tần số khác nhau trong tín hiệu gốc, đó chính là phổ biên độ. Nó cho thấy được trung bình sự hiện diện của tần số trong toàn bộ thời gian của tín hiệu. Từ đó, ta có thể biết được thành phần tần số nào có trong tín hiệu nhưng lại không biết được tần số đó xảy ra khi nào. Biến đổi Fourier thời gian ngắn có thể giải quyết được vấn đề này bằng việc biến từ 1 hình ảnh tĩnh đơn thuần thành 1 chuỗi các hình ảnh, từ đó chúng ta có thể biết được giá trị của các thành phần tần số theo thời gian.



Hình 2.4. Áp dụng biến đổi Fourier rời rạc

Áp dụng biến đổi Fourier rời rạc cho từng frame

Sau khi áp dụng cửa sổ để phân đoạn tín hiệu ban đầu, ta đã có những phân đoạn nhỏ, sau đó biến đổi Fourier rời rạc sẽ được áp dụng trên từng frame. Công thức tổng quát như sau:

$$S(m, k) = \sum_{n=0}^{N-1} x(n + mH) \cdot w(x) \cdot e^{-i2\pi n \frac{k}{N}} \quad (2.4)$$

Trong đó :

- k: tần số
- m: đại diện cho thời gian, số frame hiện tại
- N: Frame length
- H: Hop length

Đầu ra của biến đổi Fourier rời rạc: Là ma trận phẳng có kích thước (# frequency bins, # frames), biểu diễn các hệ số Fourier theo từng khung thời gian, giúp thu được thông tin đồng thời về tần số và thời gian của tín hiệu âm thanh.

Công thức tính #frequency bins:

$$\# \text{frequency bins} = \frac{\text{frame size}}{2} + 1 \quad (2.5)$$

Công thức tính số lượng frames:

$$\# \text{frame} = \frac{\text{samples} - \text{framesize}}{\text{hopsize}} + 1 \quad (2.6)$$

Bước 4: Chuyển từ tần số sang thang đo mel

Việc chuyển từ tần số sang thang đo mel có thể được tách thành các bước nhỏ hơn như sau:

- **Chọn số lượng mel bands:** Đây là một siêu tham số, cách lựa chọn là dựa trên những tác động của siêu tham số này đến ứng dụng hoặc vấn đề bài toán đang

phải giải quyết. Nó có thể được hiểu đơn giản như số lượng nốt nhạc có trong đàn ghi-ta hoặc piano hoặc những thứ tương tự như vậy, để có thể nhận biết được cao độ của từng âm thanh.

- **Xây dựng bộ lọc Mel, gồm có:**

- Chuyển đổi tần số thấp nhất/ cao nhất của âm thanh đang có sang thang đo Mel bằng công thức 2.1.
- Tạo ra số lượng bands như đã chọn, tạo thành những điểm cách đều nhau. Vì thang đo Mel có thể biểu diễn theo nhận thức, khoảng cách giống nhau sẽ tương tự nhau về mặt nhận thức. Hình 2.5. dưới đây ví dụ một trường hợp chọn 6 mel bands và có khoảng cách đều nhau. Những điểm này là tâm của các dải mel khác nhau, là các tần số trung tâm của các dải.

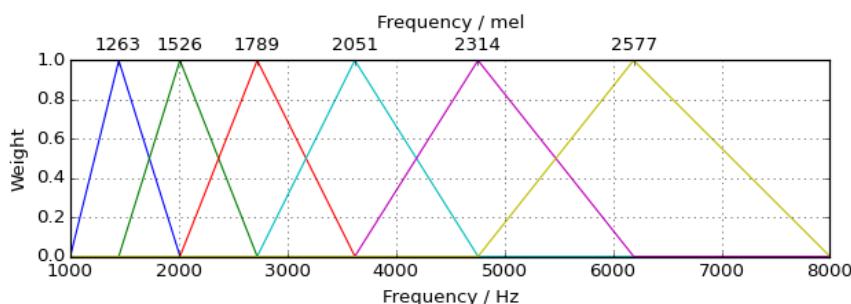


Hình 2.5. Tạo ra các bands

- Chuyển các điểm vừa tạo đó từ thang đo Mel về lại tần số, bằng công thức:

$$f = 700 \left(10^{\frac{m}{2595}} - 1 \right) \quad (2.7)$$

- Làm tròn các điểm này thành tần số gần nhất.
- Cuối cùng, tạo ra bộ lọc tam giác mel như trên hình 2.6., có sáu điểm tương ứng với 6 mel bands.



Hình 2.6. Bộ lọc mel

ứng với 6 Mel bands. Hàng phía trên là thang đo mel, hàng phía dưới là thang đo tần số. Các weight tương trưng cho việc, nếu weight = 1, âm thanh sẽ được giữ nguyên, còn nếu nhỏ hơn một thì sẽ scale lại. Ta có thể thấy các điểm lựa chọn là 1263, 1526, 1789, 2051, 2314 và 2577 trên thang đo Mel, khi gióng xuống thang tần số, ta được các tần số tương ứng chính là

tâm của các dải mel. Sau đó, chỉ cần nối các điểm trên thang đo Mel với các điểm vừa gióng xuống ở thang tần số ta được các bộ lọc tam giác như hình 2.6.. Các điểm được lấy trên thang tần số đó không có khoảng cách đều nhau nhưng sẽ liên quan về mặt nhận thức do các giá trị đó tương ứng ở thang đo Mel có khoảng cách đều nhau.

Kích thước của Mel filter banks:

$$M = (\# \text{ bands}, \text{framsize} / 2 + 1)$$

Kích thước của phổ (spectrogram):

$$Y = (\text{framsize} / 2 + 1, \# \text{frame})$$

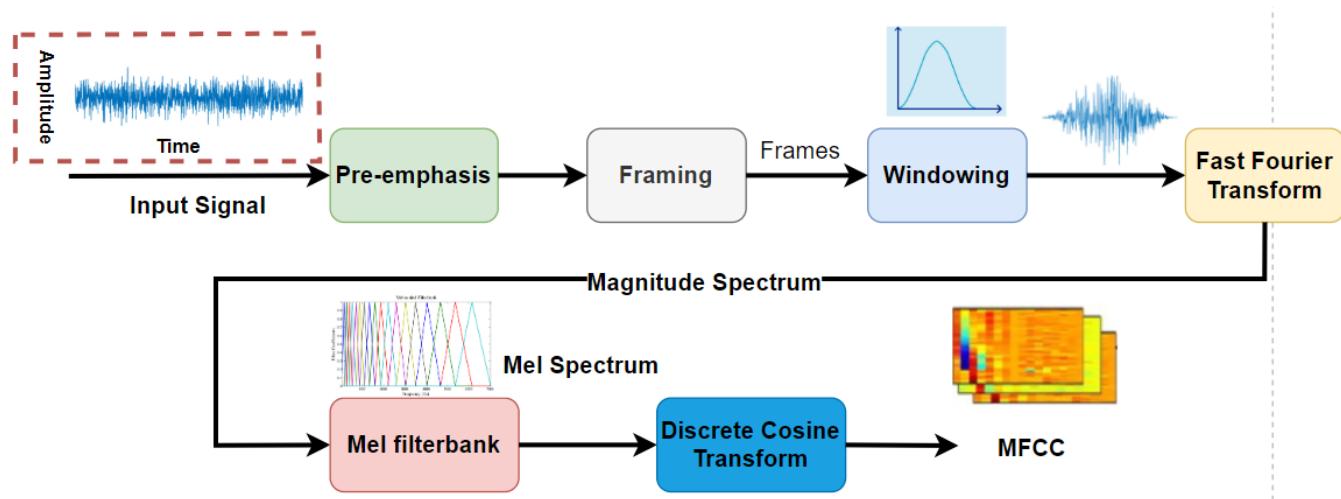
- Áp dụng các bộ lọc Mel cho spectrogram qua 1 phép tích vô hướng

Mel spectrogram = MY, có kích thước là (#bands, #frames)

2.3.2 Trích xuất đặc trưng Mel frequency cepstral coefficients (MFCC)

2.3.2.1 Sơ đồ khái hoạt động trích xuất MFCC

MFCC là một trong những đặc trưng phổ biến trong nhận dạng giọng nói, được thiết kế để mô phỏng cách tai người cảm nhận âm thanh. Đây là đặc trưng dạng nén, giúp giảm nhiễu và loại bỏ thông tin tần số dư thừa, thường được sử dụng trong các hệ thống như HMM-GMM hoặc các mô hình học sâu truyền thống.



Hình 2.7. Hoạt động trích xuất MFCC từ tín hiệu tiếng nói thô

2.3.2.2 Các bước chính để trích xuất MFCC

Quy trình trích xuất đặc trưng MFCC ban đầu **hoàn toàn giống** với quy trình tạo Mel-spectrogram ở các bước đầu, bao gồm:

- Bộ lọc tiền nhấn (Pre-emphasis)
- Phân khung (Framing)
- Áp dụng hàm cửa sổ (Windowing)

- Biến đổi Fourier rời rạc (FFT)
- Áp dụng bộ lọc tam giác trên thang đo Mel (Mel Filterbank)
- Tính log năng lượng (Log-Mel Spectrogram)

Tuy nhiên, thay vì giữ lại Mel-spectrogram 2 chiều như đầu ra cuối cùng, MFCC tiếp tục thực hiện một bước xử lý bổ sung: **áp dụng biến đổi Cosine rời rạc (DCT)** để chuyển đổi log Mel-spectrogram sang miền Cepstral, như hình 2.7. Mục tiêu là loại bỏ tương quan giữa các dải tần và trích xuất các hệ số quan trọng nhất đại diện cho đặc trưng âm thanh. Tiếp theo, sinh viên sẽ trình bày các bước còn lại để trích xuất MFCC gồm có:

- **Tính log Mel-Spectrogram:** như sinh viên đã trình bày ở bước trước, đã có ma trận log năng lượng theo thang đo Mel:

$$\text{Log-Mel Spectrogram} = \log(M \cdot Y + \varepsilon)$$

- **Áp dụng Biến đổi Cosine rời rạc (DCT):** Mục tiêu của DCT là chuyển log Mel Spectrogram từ miền tần số sang miền cepstral (cepstrum), giúp loại bỏ tương quan giữa các dải tần và trích xuất những đặc trưng quan trọng nhất:

$$\text{MFCC}(n) = \sum_{m=1}^M \log(S_m) \cdot \cos\left[\frac{\pi n}{M}(m - 0.5)\right]$$

Trong đó:

- S_m là năng lượng tại Mel band thứ m
- M là tổng số Mel bands
- n là chỉ số hệ số MFCC cần tính (thường lấy 12–13 hệ số đầu)

Kết quả: MFCC là một ma trận có kích thước (#MFCC coefficients, #frames), dùng làm đầu vào hiệu quả cho các mô hình truyền thống và cả một số mạng học sâu.

2.4 Thuật toán huấn luyện nhận dạng tiếng nói

2.4.1 Convolution neural network

Mạng nơ-ron tích chập (CNNs) là một lớp kiến trúc mạng nơ-ron sâu chuyên biệt bao gồm một hoặc nhiều cặp tầng tích chập và tầng gộp (pooling) xen kẽ nhau. Tầng tích chập áp dụng bộ lọc xử lý các phần cục bộ nhỏ của đầu vào, trong đó các bộ lọc này được nhân bản trên toàn bộ không gian đầu vào. Tầng gộp chuyển đổi các kích hoạt của tầng tích chập thành độ phân giải thấp bằng cách lấy giá trị kích hoạt lớn nhất của bộ lọc trong một cửa sổ xác định và chuyển qua bản đồ kích hoạt. CNNs là các biến thể của mạng nơ-ron kết nối đầy đủ được sử dụng rộng rãi để xử lý dữ liệu

có cấu trúc lưới [11].

Biểu diễn spectrogram cho thấy mối tương quan rất mạng theo thời gian và tần số. Do những đặc điểm này của spectrogram, nó là đầu vào phù hợp cho pipeline xử lý CNN yêu cầu bảo toàn tính local trên cả trục tần số và thời gian. Đối với tín hiệu âm thanh, việc mô hình hóa các mối tương quan cục bộ bằng CNNs sẽ rất có lợi. CNNs cũng có thể trích xuất hiệu quả các đặc trưng cấu trúc từ spectrogram và giảm độ phức tạp của mô hình thông qua việc chia sẻ trọng số.

Bởi vì spectrogram là biểu diễn trực quan hai chiều, người ta có thể tận dụng các kiến trúc CNN được sử dụng rộng rãi cho xử lý dữ liệu trực quan (hình ảnh và video) bằng cách thực hiện tích chập theo hai chiều. Phương trình toán học cho tầng tích chập 2D có thể biểu diễn như sau:

$$y_{i,j}^{(k)} = \sigma \left(\sum_{l=1}^L \sum_{m=1}^M x_{i+l-1, j+m-1}^{(l)} w_{l,m}^{(k)} + b^{(k)} \right) \quad (2.8)$$

Trong đó $x_{i,j}^{(l)}$ là giá trị pixel của kênh đầu vào thứ l tại vị trí không gian (i, j) , $w_{l,m}^{(k)}$ là trọng số của bộ lọc thứ m tại kênh thứ l tạo ra bản đồ đặc trưng thứ k , và $b^{(k)}$ là số hạng độ lệch (bias) cho bản đồ đặc trưng thứ k .

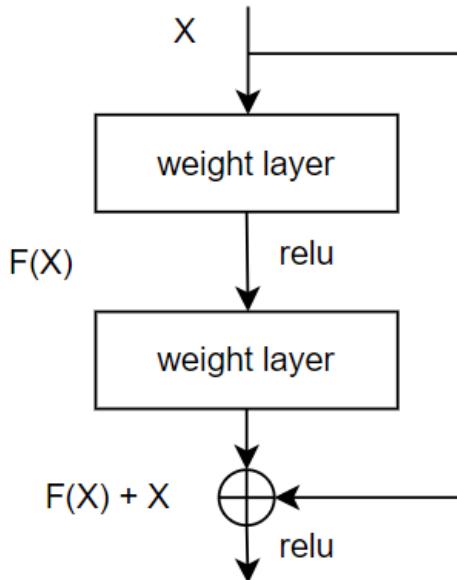
Bản đồ đặc trưng đầu ra $y_{i,j}^{(k)}$ được thu bằng cách tích chập hình ảnh đầu vào với các bộ lọc và sau đó áp dụng hàm kích hoạt σ để đưa vào tính phi tuyến. Phép toán tích chập bao gồm việc trượt cửa sổ bộ lọc trên hình ảnh đầu vào, tính tích vô hướng giữa bộ lọc và các pixel đầu vào tại mỗi vị trí, và tạo ra một pixel đầu ra duy nhất.

Tuy nhiên, có một số nhược điểm khi sử dụng CNN 2D cho xử lý âm thanh. Một trong những vấn đề lớn nhất là tích chập 2D tốn kém về mặt tính toán, đặc biệt đối với các đầu vào lớn. Điều này là do tích chập 2D liên quan đến nhiều phép nhân và cộng, và chi phí tính toán tăng nhanh theo kích thước đầu vào [11].

2.4.2 Residual Connection

Trong những năm gần đây, sự phát triển của các mô hình học sâu đã đóng vai trò quan trọng trong hàng loạt những đột phá về phân loại ảnh cũng như nhận dạng tiếng nói. Các mạng nơ-ron sâu có khả năng tự động học và kết hợp các đặc trưng ở nhiều mức độ, đồng thời cho phép huấn luyện đầu cuối với các bộ phân loại đa lớp. Việc tăng độ sâu của mạng, tức là số lượng lớp xếp chồng, đã chứng minh là một yếu tố quan trọng trong việc nâng cao hiệu suất mô hình [12, 13]. Nhưng khi tăng độ sâu vượt ngưỡng nhất định, mô hình bắt đầu suy giảm hiệu suất - không chỉ trên tập kiểm tra mà ngay cả trên tập huấn luyện - điều này đã được quan sát và xác nhận trong nhiều nghiên cứu [14]. Để khắc phục vấn đề này, nghiên cứu [15] đã đề xuất một khung học tập sâu mới dựa trên **học tàn dư (residual learning)**. Thay vì huấn luyện các lớp nhằm xấp xỉ trực tiếp một ánh xạ mục tiêu $H(x)$, nghiên cứu đã xây dựng mô hình để học một **hàm tàn dư** $F(x) = H(x) - x$, từ đó ánh xạ gốc có thể được

viết lại thành $F(x) + x$. Khái niệm này được hiện thực hóa thông qua các **kết nối tắt (shortcut connections)** không tham số, cho phép tín hiệu đầu vào đi qua trực tiếp, được cộng vào đầu ra của các lớp phi tuyến như hình 2.8.. Các kết nối này không làm tăng số lượng tham số cũng như độ phức tạp tính toán, và có thể dễ dàng tích hợp vào các thư viện học sâu hiện hành.



Hình 2.8. Kết nối tắt trong mạng phần dư

2.4.3 Thuật toán Transformer

2.4.3.1 Vai trò

Transformer là một mô hình được giới thiệu lần đầu vào năm 2017 trong bài báo "Attention is All You Need" của Vaswani và các cộng sự [16]. Kể từ đó, Transformer đã trở thành nguồn cảm hứng cho nhiều nghiên cứu và đột phá trong lĩnh vực học sâu (deep learning). Mô hình này dựa trên kiến trúc encoder-decoder, rất phù hợp với các bài toán dạng chuỗi sang chuỗi (sequence-to-sequence), đặc biệt là trong nhận dạng giọng nói tự động (ASR).

Trong xử lý ngôn ngữ tự nhiên (NLP), Transformer nổi bật nhờ cơ chế attention – giúp mô hình tập trung vào các phần quan trọng của dữ liệu đầu vào để hiểu và xử lý ngữ cảnh hiệu quả hơn. Nhờ đó, các biến thể của Transformer đã được ứng dụng rộng rãi trong nhiều nhiệm vụ, từ dịch máy đến nhận diện tiếng nói.

2.4.3.2 Cơ chế tự tập trung

Điểm nổi bật của transformer là sử dụng cơ chế tự tập trung để khắc phục các nhược điểm của cơ chế tập trung với kiến trúc encoder - decoder đơn giản trước đó [17].

Các vector queries, keys và value

a, Queries vector

Queries vector có vai trò và tầm quan trọng như sau:

- Vai trò: Vector truy vấn đại diện cho tokens hiện đang tính toán attention. Nó xác định tầm quan trọng của các tokens khác trong ngữ cảnh của token hiện tại
- Tầm quan trọng: Vector truy vấn giúp cho mô hình xác định được phần nào của chuỗi cần phải tập trung vào đối với mỗi token cụ thể. Bằng cách tính toán tích vô hướng giữa vector truy vấn và tất cả vector keys, mô hình có thể biết được nên chú ý bao nhiêu vào mỗi token khác
Nó cũng cho biết về mối quan hệ giữa token hiện tại và phần còn lại của chuỗi, nắm bắt được sự phụ thuộc và ngữ cảnh cần thiết

b, Key vector

Key vector có vai trò và tầm quan trọng như sau:

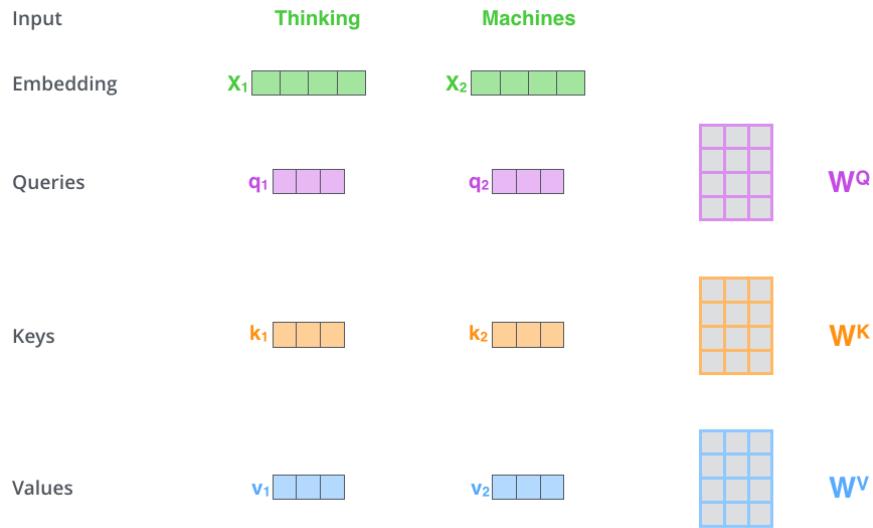
- Vai trò: Vector khoá biểu diễn tất cả các mã thông báo trong chuỗi và được sử dụng để so sánh với các vector truy vấn nhằm tính toán điểm attention.
- Tầm quan trọng: Các keys được so sánh với các truy vấn để đo lường mức độ liên quan hoặc khả năng tương thích của từng mã thông báo với mã thông báo hiện tại. So sánh này giúp xác định mức độ chú ý mà mỗi mã thông báo nêu nhận. Các keys cũng đóng vai trò quan trọng trong việc truy xuất thông tin có liên quan nhất từ chuỗi bằng cách cung cấp cơ sở cho cơ chế attention để tính điểm tương đồng.

c, Value vectors

Dưới đây là vai trò và tầm quan trọng của value vectors:

- Vai trò: Vector giá trị lưu giữ thông tin thực tế sẽ được tổng hợp để tạo thành đầu ra cho cơ chế attention.
- Tầm quan trọng: Các giá trị chứa dữ liệu sẽ được cân nhắc theo điểm attention. Tổng giá trị có trọng số tạo thành đầu ra của cơ chế tự chú ý, sau đó được chuyển đến các lớp tiếp theo trong mạng.
Bằng cách đánh giá các giá trị theo điểm attention, mô hình sẽ bảo toàn và tổng hợp ngữ cảnh có liên quan từ toàn bộ chuỗi, điều này rất quan trọng với các nhiệm vụ seq2seq.

Bước đầu tiên trong việc tính toán sự tập trung là tạo ra ba vector từ mỗi vector đầu vào của bộ mã hoá. Đối với mỗi từ, ta sẽ tạo một vector truy vấn, 1 vector keys và vector giá trị. Những vector này được tạo bằng cách nhân vector nhúng với 3 ma trận đã trained trong suốt quá trình training.



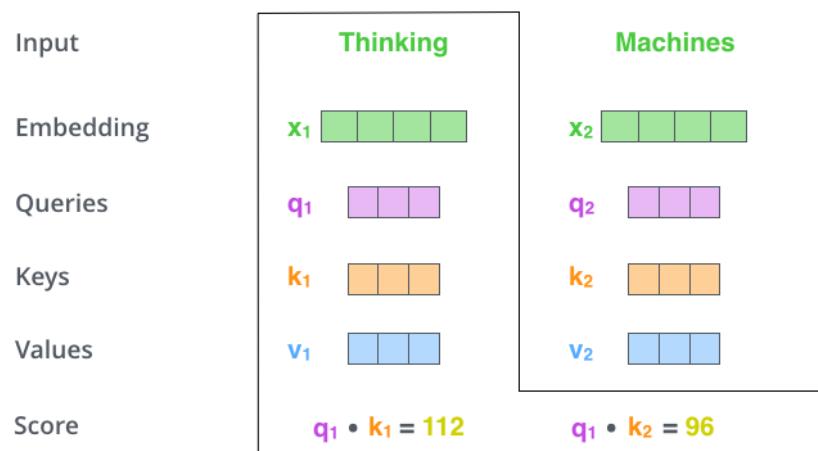
Hình 2.9. Tạo ra 3 vector query, key và value cho từng từ.

Như trình bày ở hình 2.9., Nhân vector nhúng x_1 với W^Q để tạo ra q_1 , Vector "query" q_1 liên kết với từ đó. Tương tự như vậy ta có thể tạo được vector "query" và "value".

Bước thứ hai: tính toán điểm self-attention.

Ví dụ, ta đang tính toán self-attention cho từ đầu tiên trong ví dụ như hình 2.9., "Thinking". Ta cần phải tính điểm của mỗi từ trong chuỗi đầu vào so với từ hiện tại. Điểm đó sẽ xác định cần tạo trung bao nhiêu vào những phần khác của chuỗi đầu vào khi mã hóa 1 từ ở vị trí nhất định.

Điểm được tính bằng cách lấy tích vô hướng vector truy vấn và vector key của từ tương ứng đang muốn tính điểm. Nghĩa là, nếu ta đang xử lý self-attention cho từ ở vị trí #1, điểm đầu tiên sẽ là tích vô hướng của q_1 và k_1 . Điểm thứ hai sẽ là tích vô hướng của q_1 và k_2 , cách tính điểm được biểu diễn trong hình 2.10..



Hình 2.10. Tính score cho từng từ trong câu đầu vào.

Bước thứ ba và thứ tư, chia điểm vừa tính được như trên hình 2.10. cho $\sqrt{d_k}$ trong

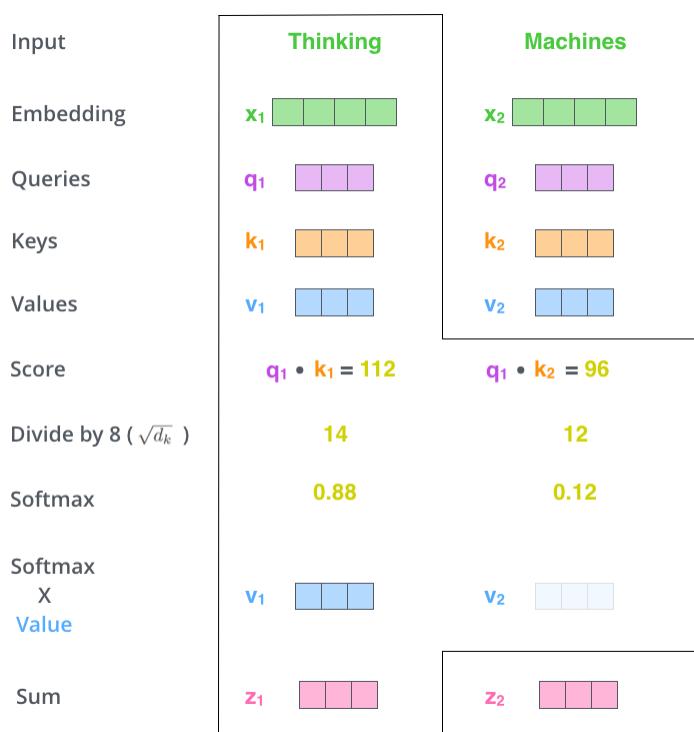
đó d_k là kích thước của key vectors. Bước này còn được gọi là scaling. Nó giúp cho việc có gradients ổn định hơn trong suốt quá trình training, tránh trường hợp bão hòa gradients và biến mất gradients. Các trọng số attention được cân bằng hơn so với trường hợp không sử dụng scaling. Kết quả sau đó được đưa qua toán tử softmax. Softmax chuẩn hóa điểm để tất cả đều dương và có tổng bằng 1.

Score	$q_1 \cdot k_1 = 112$	$q_1 \cdot k_2 = 96$
Divide by 8 ($\sqrt{d_k}$)	14	12
Softmax	0.88	0.12

Hình 2.11. Tính scaling và softmax của score

Kết quả từ toán tử softmax này cho biết mỗi từ sẽ được thể hiện bao nhiêu ở vị trí này. Rõ ràng rằng từ ở vị trí này sẽ có softmax score lớn nhất nhưng đôi khi nó cũng rất hữu ích để có thêm softmax score của những từ khác liên quan đến từ hiện tại để có thể lưu giữ được ngữ cảnh của câu.

Bước thứ năm là nhân mỗi vector value với điểm softmax. Mục đích ở đây là giữ lại những từ mà chúng ta muốn tập trung bao bằng cách nhân với một hệ số lớn, loại bỏ các từ không liên quan bằng cách nhân vector value của từ đó với một hệ số nhỏ, ví dụ 0.001. Bước thứ sau là tổng hợp các vector value có trọng số. Điều này sẽ tạo ra đầu ra của self-attention cho từ ở vị trí này (ở đây là từ đầu tiên), hình 2.10.



Hình 2.12. Tính toán đầu ra của self-attention

Trên hình 2.12., ta có thể thấy từ vector kết quả sẽ được sử dụng để gửi đến để xử lý ở các lớp tiếp theo.

Tính toán ma trận của Self-Attention Bước đầu tiên cũng là tính toán các ma trận Query, Key và Value. Tương tự hình 2.9. nhưng giờ đây các vector nhúng được tổng hợp thành ma trận X. Cuối cùng, tổng hợp từ bước thứ 2 đến 6 như trình bày phần trên để tính toán ma trận đầu ra.

$$\text{softmax}\left(\frac{\begin{array}{|c|c|} \hline Q & K^T \\ \hline \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} & \begin{array}{|c|c|} \hline & \\ \hline & \\ \hline \end{array} \\ \hline \end{array} \times \begin{array}{|c|c|} \hline & \\ \hline & \\ \hline \end{array} \right) \begin{array}{|c|c|c|} \hline & & V \\ \hline & & \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} \\ \hline \end{array}$$

$$= \begin{array}{|c|c|c|} \hline & & Z \\ \hline & & \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} \\ \hline \end{array}$$

Hình 2.13. Tính toán ma trận đầu ra self-attention

$$\begin{array}{ccc} X & \times & W^Q \\ \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} & \times & \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} \\ \hline \end{array} = \begin{array}{|c|c|} \hline & Q \\ \hline & \begin{array}{|c|c|} \hline & \\ \hline & \\ \hline \end{array} \\ \hline \end{array}$$

$$\begin{array}{ccc} X & \times & W^K \\ \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} & \times & \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} \\ \hline \end{array} = \begin{array}{|c|c|} \hline & K \\ \hline & \begin{array}{|c|c|} \hline & \\ \hline & \\ \hline \end{array} \\ \hline \end{array}$$

$$\begin{array}{ccc} X & \times & W^V \\ \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} & \times & \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} \\ \hline \end{array} = \begin{array}{|c|c|} \hline & V \\ \hline & \begin{array}{|c|c|} \hline & \\ \hline & \\ \hline \end{array} \\ \hline \end{array}$$

Hình 2.14. Tính toán các ma trận Query, Key và Value

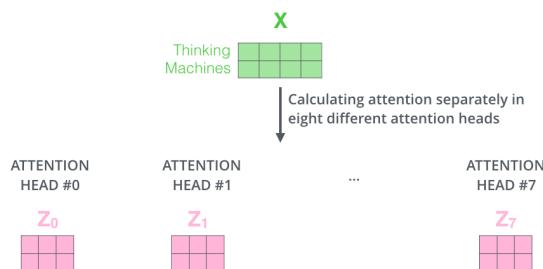
2.4.3.3 Cơ chế tập trung nhiều đầu

Cơ chế tập trung nhiều đầu đã cải thiện cơ chế tự tập trung theo 2 cách:

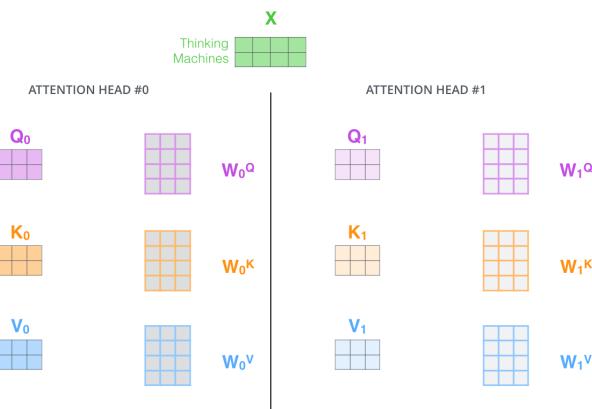
- Mở rộng khả năng của mô hình để tập trung vào nhiều vị trí khác. Ví dụ trong self-attention, z1 có thể chứa một chút thông tin của mọi encoding khác nhưng nó có thể bị chi phối bởi từ thực tế của chính nó.

- Cung cấp cho lớp chú ý "không gian con đại diện". Với sự chú ý nhiều đầu, chúng ta không chỉ có một mà là nhiều bộ ma trận Query, Key và Value. Mỗi bộ này sẽ được khởi tạo ngẫu nhiên. Sau đó, sau khi đào tạo, mỗi bộ được sử dụng để đưa đầu vào nhúng vào một không gian con đại diện khác.

Bước đầu tiên, chia 8 đầu, sau đó nhân ma trận đầu vào X với ma trận trọng số để tạo ra các ma trận Query, Key và Value tương ứng. Thực hiện quy trình tương tự với self-attention ta đã tính toán ở trên, ta có thể thu được từng ma trận đầu ra tương ứng với mỗi đầu.

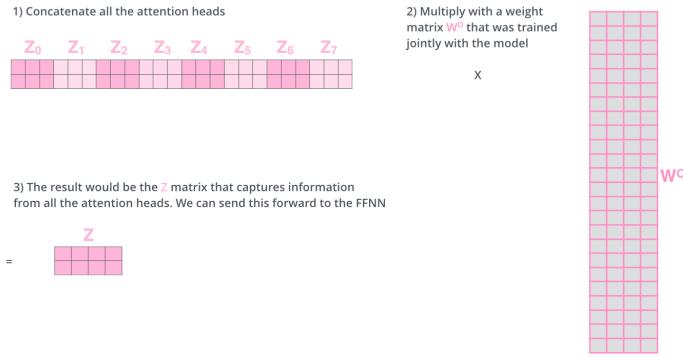


Hình 2.15. Tính toán ma trận đầu ra cho từng đầu trong multi-head attention



Hình 2.16. Tính toán các ma trận Query, Key và value cho từng đầu

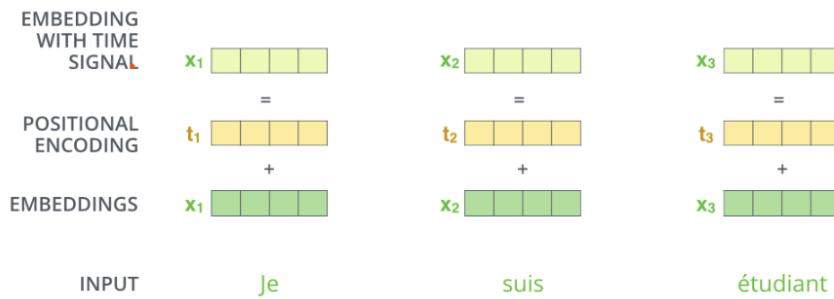
Lớp feed-forward layer mong đợi chỉ một ma trận duy nhất. Do vậy, ta cần phải hợp nhất ma trận đầu ra của 8 đầu thành một ma trận duy nhất sau đó nhân nó với ma trận trọng số bổ sung WO.



Hình 2.17. Hợp nhất đầu ra các đầu trong multi-head attention

2.4.3.4 Mã hóa vị trí

Transformer thêm một vector vào mỗi lần nhúng đầu vào. Các vector này tuân theo một mẫu cụ thể mà mô hình học được, giúp mô hình xác định vị trí của từng từ hoặc khoảng cách giữa các từ khác nhau trong chuỗi.



Hình 2.18. Cộng thêm positional encoding vào đầu vào nhúng

2.5 Connectionist Temporal Classification (Phân loại thời gian kết nối)

2.5.1 Lý do sử dụng CTC

Trong các bài toán nhận dạng tiếng nói, dữ liệu huấn luyện thường bao gồm các cặp (âm thanh, phiên âm). Tuy nhiên, một thách thức lớn là thiếu thông tin căn chỉnh tường minh giữa chuỗi đặc trưng âm thanh và chuỗi ký tự văn bản đầu ra. Cụ thể, không có thông tin về thời điểm cụ thể trong tín hiệu âm thanh mà mỗi ký tự hoặc từ tương ứng xuất hiện. Điều này gây khó khăn cho việc huấn luyện mô hình vì không thể áp dụng các kỹ thuật học có giám sát truyền thống yêu cầu căn chỉnh từng bước đầu vào với nhãn đầu ra. Một cách tiếp cận đơn giản là giả định rằng mỗi ký tự tương ứng với một số bước thời gian cố định (ví dụ: 10 vector đặc trưng trên mỗi ký tự). Tuy nhiên, điều này là phi thực tế vì tốc độ nói và cách phát âm có thể thay đổi đáng kể giữa các cá nhân, làm cho độ dài và phân bố đặc trưng âm thanh rất khác nhau. Việc căn chỉnh thủ công là một lựa chọn khác nhưng không khả thi trong thực tế do chi phí nhân lực cao, đặc biệt với các tập dữ liệu lớn.

Trước những vấn đề nêu trên, Connectionist Temporal Classification (CTC) được giới thiệu bởi Graves et al. như một giải pháp hiệu quả cho việc học từ dữ liệu chưa được căn chỉnh thời gian. Phương pháp này cho phép mô hình học trực tiếp từ các cặp (đầu vào, nhãn mục tiêu) mà không cần thông tin căn chỉnh chi tiết, bằng cách tối đa hóa xác suất sinh ra chuỗi nhãn đầu ra thông qua tất cả các khả năng căn chỉnh hợp lệ [18]. CTC được tích hợp vào pipeline huấn luyện như một lớp đầu ra đặc biệt, cho phép mô hình học mà không cần căn chỉnh frame-level giữa đầu vào và nhãn mục tiêu

2.5.2 Đầu ra theo thời gian và đường dẫn xác suất trong CTC

Trong các hệ thống nhận dạng tiếng nói liên tục, mô hình học sâu (ví dụ RNN, Transformer) thường sinh ra một chuỗi đầu ra có chiều dài T , với mỗi bước thời gian t là một phân phối xác suất trên tập ký tự gồm C ký tự và một ký tự đặc biệt gọi là **blank** (ký hiệu "-").

2.5.2.1 Khái niệm đường dẫn (Path) trong CTC

Một chuỗi đầu ra từ mô hình có thể được hiểu là một *đường dẫn* $\pi = (\pi_1, \pi_2, \dots, \pi_T)$, trong đó tại mỗi thời điểm t , mô hình lựa chọn một ký tự (bao gồm cả blank) từ tập ký tự mở rộng. Xác suất của một đường dẫn π cho chuỗi đầu vào x được tính như sau:

$$P(\pi|x) = \prod_{t=1}^T y_t^{\pi_t} \quad (2.9)$$

trong đó $y_t^{\pi_t}$ là xác suất mô hình dự đoán ký tự π_t tại thời điểm t .

2.5.2.2 Hàm thu gọn B và không gian đường dẫn

Do một nhãn đầu ra (label) có thể được sinh ra từ nhiều đường dẫn khác nhau, CTC sử dụng một hàm thu gọn $B(\pi)$ để ánh xạ một đường dẫn π thành chuỗi ký tự đầu ra cuối cùng thông qua hai bước:

- Loại bỏ các ký tự lặp lại liên tiếp.
- Loại bỏ tất cả các ký tự blank "-".

Ví dụ:

$$B(a - bb --) = ab \quad (2.10)$$

Tập hợp tất cả các đường dẫn sinh ra cùng một chuỗi nhãn l là $\pi | B(\pi) = l$. Điều này cho phép mô hình học ánh xạ từ đặc trưng âm thanh sang chuỗi ký tự mà không cần biết vị trí cụ thể của từng ký tự trong đầu vào.

2.5.3 Hàm măt măt CTC

Hàm măt măt của CTC được định nghĩa là tổng xác suất của tất cả các đường dẫn có thể sinh ra nhăm mục tiêu l :

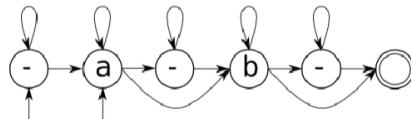
$$p(l|x) = \sum_{\pi \in B^{-1}(l)} P(\pi|x) \quad (2.11)$$

Trong đó $B^{-1}(l)$ là tập các đường dẫn π sao cho $B(\pi) = l$. Việc tính tổng xác suất này được thực hiện hiệu quả nhờ thuật toán Forward-Backward trên biểu đồ trạng thái FSM (Finite State Machine).

Biểu diễn bằng FSM [19].

Một nhăm chuỗi l (ví dụ "ab") được mở rộng bằng cách chèn ký tự blank giữa các ký tự và thêm blank vào đầu/cuối để tạo thành chuỗi l' (ví dụ: "- a - b -"). Mỗi phần tử trong chuỗi mở rộng này tương ứng với một trạng thái trong FSM. FSM cho phép ba loại chuyển trạng thái:

- Lặp lại tại cùng trạng thái.
- Chuyển tiếp sang trạng thái tiếp theo (nếu là blank hoặc ký tự khác).
- Bỏ qua blank giữa hai ký tự khác nhau.



Hình 2.19. Mô hình hóa nhăm chuỗi bằng FSM trong CTC

2.5.4 Giải mã CTC (CTC Decoding)

Sau khi mô hình được huấn luyện, quá trình giải mã (decoding) được sử dụng để chuyển ma trận xác suất đầu ra thành chuỗi ký tự cuối cùng. Hai phương pháp phổ biến nhất là:

- **Best Path Decoding (Greedy Decoding):** Ở mỗi bước thời gian, chọn ký tự có xác suất cao nhất, sau đó áp dụng hàm thu gọn B để loại bỏ blank và gộp lặp.
- **Beam Search Decoding:** Duy trì đồng thời nhiều đường dẫn có xác suất cao nhất (theo beam width), sau đó chọn nhăm có tổng xác suất lớn nhất. Phương pháp này có thể kết hợp với mô hình ngôn ngữ để tăng độ chính xác.

2.6 Mô hình ngôn ngữ (Language model)

Định nghĩa: Mô hình ngôn ngữ (Language Model - LM) là một thành phần quan trọng trong các hệ thống xử lý ngôn ngữ tự nhiên, có nhiệm vụ gán xác suất cho

một chuỗi từ hoặc dự đoán xác suất xuất hiện của từ tiếp theo dựa trên ngữ cảnh trước đó. Cụ thể, mô hình ngôn ngữ học được phân phối xác suất trên các chuỗi từ, từ đó giúp hệ thống đánh giá và lựa chọn các câu, cụm từ có tính tự nhiên và hợp lý hơn về mặt ngữ nghĩa.

Vai trò của mô hình ngôn ngữ trong nhận dạng tiếng nói

Trong bài toán nhận dạng tiếng nói, mô hình ngôn ngữ giữ vai trò then chốt nhằm hiệu chỉnh đầu ra của mô hình âm học (Acoustic Model). Nếu chỉ dựa vào xác suất từng thời điểm của mô hình âm học, hệ thống có thể tạo ra các câu không tự nhiên hoặc sai ngữ pháp. Nhờ mô hình ngôn ngữ, hệ thống có thể ưu tiên các chuỗi từ hợp lý hơn, giảm đáng kể tỉ lệ lỗi (Word Error Rate - WER) và tăng chất lượng nhận dạng. Điều này đặc biệt quan trọng trong tiếng Việt, nơi trật tự từ và ngữ cảnh đóng vai trò lớn trong ý nghĩa câu.

Ví dụ:

- Tôi đang xem đá bóng ở nhà.
- Tôi đang xem ở nhà đá bóng.

Mô hình ngôn ngữ sẽ đánh giá xác suất của hai câu trên và ưu tiên câu đầu vì tự nhiên hơn trong tiếng Việt.

N-gram là một trong những mô hình ngôn ngữ đơn giản và hiệu quả nhất, được sử dụng rộng rãi trong các hệ thống nhận dạng tiếng nói hiện đại nhờ các ưu điểm như: dễ huấn luyện, tốc độ truy xuất nhanh, dễ tích hợp với các thuật toán giải mã như beam search, và không yêu cầu nhiều tài nguyên tính toán. Các công trình nhận dạng tiếng nói gần đây vẫn ưu tiên sử dụng n-gram (đặc biệt là 3-gram, 4-gram) trong các hệ thống lớn hoặc trên thiết bị hạn chế tài nguyên, nhờ khả năng mở rộng trên tập dữ liệu lớn và tính ổn định cao [20].

2.6.1 Mô hình n-gram truyền thống [1]

2.6.1.1 Thế nào là n-gram

Một n-gram là một chuỗi gồm n từ liên tiếp. Ví dụ:

- 1-gram (unigram): (I), (AM), (THE), (KING)
- 2-gram (bigram): (I AM), (AM THE), (THE KING)
- 3-gram (trigram): (I AM THE), (AM THE KING)
- 4-gram: (I AM THE KING)

Với câu tiếng Anh “I AM THE KING”, ta có thể phân tách thành các n-gram như trên.

2.6.1.2 Công thức tính xác suất mô hình n-gram

Xác suất của một n-gram được tính dựa trên tần suất xuất hiện của nó trong tập ngữ liệu huấn luyện. Cụ thể, công thức tính xác suất n-gram như sau:

Unigram (1-gram): Xác suất của một từ w được ước lượng bằng:

$$P(w) = \frac{\text{count}(w)}{N} \quad (2.12)$$

Trong đó, $\text{count}(w)$ là số lần từ w xuất hiện trong tập ngữ liệu, và N là tổng số từ trong tập ngữ liệu.

Bigram (2-gram): Xác suất của từ w_i xuất hiện sau từ w_{i-1} được tính bằng:

$$P(w_i | w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i)}{\text{count}(w_{i-1})} \quad (2.13)$$

Trigram (3-gram): Xác suất của từ w_i xuất hiện sau chuỗi hai từ w_{i-2}, w_{i-1} được tính như sau:

$$P(w_i | w_{i-2}, w_{i-1}) = \frac{\text{count}(w_{i-2}, w_{i-1}, w_i)}{\text{count}(w_{i-2}, w_{i-1})} \quad (2.14)$$

4-gram (4-gram): Xác suất của từ w_i dựa trên ba từ trước đó $w_{i-3}, w_{i-2}, w_{i-1}$ được ước lượng như sau:

$$P(w_i | w_{i-3}, w_{i-2}, w_{i-1}) = \frac{\text{count}(w_{i-3}, w_{i-2}, w_{i-1}, w_i)}{\text{count}(w_{i-3}, w_{i-2}, w_{i-1})} \quad (2.15)$$

Trong các công thức trên, $\text{count}(\cdot)$ biểu thị số lần xuất hiện của n-gram tương ứng trong tập ngữ liệu. Các mô hình n-gram với giá trị n lớn hơn thường cho kết quả chính xác hơn nhưng cũng yêu cầu nhiều dữ liệu hơn và có thể gặp vấn đề về dữ liệu thừa (data sparsity).

2.6.1.3 Ưu và nhược điểm của n-gram:

Mô hình n-gram là một trong những phương pháp truyền thống phổ biến trong xử lý ngôn ngữ tự nhiên và nhận dạng tiếng nói. Dưới đây là một số ưu điểm và hạn chế chính của phương pháp này:

Ưu điểm

- Dễ xây dựng, dễ huấn luyện trên tập dữ liệu lớn.
- Tốc độ truy xuất nhanh, phù hợp với các hệ thống thời gian thực.
- Có thể mở rộng lên 3-gram, 4-gram để tận dụng ngữ cảnh dài hơn mà vẫn giữ được hiệu năng tốt.
- Được hỗ trợ bởi nhiều công cụ tối ưu hóa như KenLM, SRILM, v.v.

- Khả năng tích hợp hiệu quả với beam search trong giải mã CTC.

Nhược điểm

- **Giới hạn về ngữ cảnh:** N-gram chỉ xét được ngữ cảnh ngắn (thường là 3-gram hoặc 4-gram), do đó không thể nắm bắt được các mối quan hệ dài hạn trong câu hoặc đoạn văn. Điều này dẫn đến việc mô hình có thể bỏ qua các cấu trúc ngữ pháp phức tạp hoặc các phụ thuộc xa.
- **Vấn đề dữ liệu thưa (data sparsity):** Khi tăng giá trị n , số lượng n-gram có thể tăng rất nhanh, khiến nhiều n-gram không xuất hiện trong tập huấn luyện, làm giảm độ tin cậy của xác suất dự đoán.

2.6.2 Mô hình ngôn ngữ học sâu (Neural Language Model)

Bên cạnh n-gram, các mô hình ngôn ngữ học sâu như RNN-LM, LSTM-LM, Transformer-LM (ví dụ: GPT, BERT) ngày càng được ứng dụng trong các hệ thống nhận dạng tiếng nói hiện đại. Các mô hình này có khả năng học ngữ cảnh dài, linh hoạt hơn trong việc biểu diễn nghĩa và có thể tổng quát hóa tốt trên các cấu trúc ngôn ngữ phức tạp.

Ưu điểm của mô hình ngôn ngữ học sâu:

- Khả năng mô hình hóa ngữ cảnh dài, vượt xa giới hạn của n-gram truyền thống.
- Học được các đặc trưng ngữ nghĩa sâu, giúp giảm lỗi ngữ pháp và lỗi đồng âm.
- Có thể fine-tune trên các tập dữ liệu đặc thù để nâng cao chất lượng nhận dạng.

Nhược điểm của mô hình ngôn ngữ học sâu :

- **Yêu cầu tài nguyên tính toán lớn:** Các mô hình RNN-LM, LSTM-LM, Transformer-LM đòi hỏi phần cứng mạnh (GPU/TPU), bộ nhớ lớn và thời gian huấn luyện lâu, không phù hợp với các hệ thống nhúng hoặc thiết bị di động.
- **Khó tích hợp vào pipeline decode truyền thống:** Việc tích hợp neural LM vào beam search hoặc các thuật toán giải mã CTC phức tạp hơn n-gram, đòi hỏi tối ưu hóa về tốc độ và bộ nhớ.
- **Cần nhiều dữ liệu và fine-tuning:** Neural LM phát huy hiệu quả khi được huấn luyện trên tập dữ liệu lớn và đa dạng; với dữ liệu hạn chế, mô hình dễ bị overfitting hoặc không tổng quát tốt.

2.6.3 Sử dụng mô hình ngôn ngữ với KenLM

KenLM là một công cụ mạnh mẽ và tối ưu cho việc xây dựng các mô hình ngôn ngữ n-gram, cho phép gán xác suất cho chuỗi các từ dựa trên thống kê từ tập văn bản

huấn luyện. Trong hệ thống nhận dạng tiếng nói, mô hình ngôn ngữ đóng vai trò quan trọng trong giai đoạn giải mã, đặc biệt khi sử dụng thuật toán *beam search* để lựa chọn chuỗi từ đầu ra hợp lý nhất dựa trên ngữ cảnh.

2.6.3.1 Tập dữ liệu huấn luyện

Nguồn dữ liệu dùng để huấn luyện các mô hình ngôn ngữ bao gồm:

- Văn bản từ các tập dữ liệu tiếng Việt có chủ thích như **VLSP 2020, FPT và VIVOS**.
- Dữ liệu trích xuất từ truyện tranh tiếng Việt.
- Văn bản từ sách, báo và các tài liệu tiếng Việt khác được thu thập thủ công.

Toàn bộ dữ liệu được xử lý trước khi huấn luyện, bao gồm chuẩn hóa, loại bỏ ký tự đặc biệt, và tách từ theo chuẩn tiếng Việt nhằm đảm bảo chất lượng và độ chính xác của mô hình ngôn ngữ.

2.7 Khối giải mã

Khi sử dụng mô hình CTC cho bài toán nhận dạng tiếng nói, đầu ra của mạng là một ma trận xác suất $\mathbf{P} \in \mathbf{R}^{T \times |A|}$, trong đó T là số bước thời gian và $|A|$ là kích thước bảng chữ cái (bao gồm cả ký tự đặc biệt *blank*). Mỗi hàng của \mathbf{P} tương ứng với một phân phối xác suất trên bảng chữ cái tại mỗi thời điểm.

Nhiệm vụ của **bộ giải mã (decoder)** là chuyển đổi ma trận xác suất này thành chuỗi ký tự đầu ra có nghĩa. Cụ thể, cần tìm chuỗi y có xác suất hậu nghiệm cao nhất:

$$\hat{y} = \arg \max_y P(y|x) \quad (2.16)$$

trong đó x là đặc trưng âm thanh đầu vào, và $P(y|x)$ là xác suất mô hình dự đoán chuỗi y cho đầu vào x .

Do đặc thù của mô hình CTC, việc giải mã không chỉ đơn thuần là chọn ký tự có xác suất cao nhất tại mỗi thời điểm, mà còn cần áp dụng các quy tắc đặc biệt như loại bỏ các ký tự lặp liên tiếp và ký tự *blank*. Dưới đây là các chiến lược giải mã phổ biến:

2.7.1 Giải mã Greedy

Greedy decoding là phương pháp giải mã đơn giản và phổ biến nhất cho mô hình CTC. Ý tưởng của phương pháp này là tại mỗi bước thời gian t , chỉ chọn ký tự có xác suất cao nhất (tức là “tham lam”, không xét đến các khả năng khác). Sau khi chọn xong toàn bộ các ký tự tại từng bước thời gian, ta tiến hành loại bỏ các ký tự *blank* và gộp các ký tự giống nhau liên tiếp thành một ký tự duy nhất, thu được chuỗi kết quả cuối cùng.

2.7.1.1 Quy trình thực hiện thuật toán

1. Chọn ký tự xác suất lớn nhất tại mỗi bước thời gian:

$$c_t = \arg \max_{c \in A} P(c, t) \quad (2.17)$$

Trong đó, $P(c, t)$ là xác suất ký tự c tại thời điểm t .

2. Tạo chuỗi ký tự tạm thời:

$$S = [c_1, c_2, \dots, c_T]$$

3. Loại bỏ ký tự blank và gộp ký tự lặp liên tiếp:

- Loại bỏ tất cả các ký tự blank (“_”).
- Với các ký tự giống nhau liên tiếp, chỉ giữ lại một ký tự đại diện.
- Chuỗi kết quả cuối cùng y được ký hiệu là:

$$y = \mathcal{B}(S) \quad (2.18)$$

Trong đó, \mathcal{B} là phép biến đổi của CTC (CTC mapping).

2.7.1.2 Ví dụ minh họa

Ví dụ 1

Giả sử đầu ra của mạng tại 10 bước thời gian là:

$$[_, h, h, e, _, l, l, _, o, _]$$

Sau khi loại bỏ blank và gộp ký tự lặp liên tiếp, ta thu được chuỗi kết quả:

"hello"

Ví dụ 2

a	0.01	0.05	0.85	0.7	0.04	0.01	0.01
b	0.01	0.05	0.01	0.05	0.01	0.02	0.03
c	0.15	0.75	0.01	0.05	0.01	0.05	0.01
d	0.01	0.02	0.01	0.05	0.01	0.03	0.05
e	0.01	0.01	0.01	0.05	0.02	0.01	0.02
t	0.01	0.02	0.01	0.05	0.01	0.01	0.78
-	0.8	0.1	0.1	0.05	0.9	0.87	0.1
	1	2	3	4	5	6	7

Hình 2.20. Ví dụ về thuật toán giải mã Greedy

Trong hình 2.20., những giá trị nền vàng là xác suất cao nhất trong mỗi time-step. Với giải thuật này, ta sẽ chọn được chuỗi ký tự "_caa_t" sau đó loại bỏ ký tự trùng nhau và ký tự trống để cho ra chuỗi hoàn chỉnh là "cat".

Ví dụ 3. Xét một ví dụ minh họa đơn giản trong Hình 2.21..

a	0.35	0.2
b	0.05	0.05
-	0.6	0.75
	1	2

Hình 2.21. Ví dụ không tối ưu về giải mã Greedy

Giả sử đầu ra của mô hình CTC tại hai thời điểm $t = 1$ và $t = 2$ có các xác suất tương ứng cho các ký tự a và _ (ký tự blank). Ta muốn tính xác suất để nhận được chuỗi đầu ra là "a" sau khi loại bỏ ký tự lặp và blank theo quy tắc CTC.

Các chuỗi hợp lệ ánh xạ thành "a" gồm:

$$“aa”, “a_”, “_a”$$

Tổng xác suất của "a" là:

$$\begin{aligned} P(“a” | x) &= P(“aa” | x) + P(“a_” | x) + P(“_a” | x) \\ &= 0.35 \times 0.2 + 0.35 \times 0.75 + 0.6 \times 0.2 \\ &= 0.07 + 0.2625 + 0.12 = 0.4525 \end{aligned}$$

Tuy nhiên, nếu sử dụng phương pháp **greedy decoding** — chọn ký tự có xác suất lớn nhất tại mỗi bước — thì ta chọn "_" tại $t = 1$ (xác suất 0.6) và "_" tại $t = 2$ (xác suất 0.75), dẫn đến:

$$P(“_” | x) = 0.6 \times 0.75 = 0.45$$

Sau khi loại bỏ ký tự lặp và blank, chuỗi "_" trở thành chuỗi rỗng "".

Do đó:

$$P(“a” | x) = 0.4525 > 0.45 = P(“” | x)$$

Kết luận: mặc dù chuỗi "a" có xác suất tổng thể cao hơn, phương pháp greedy decoding vẫn chọn ra chuỗi "" do chỉ xét từng bước một. Điều này chứng minh rằng greedy decoding không đảm bảo tìm được chuỗi có xác suất tối đa.

2.7.1.3 Ưu điểm và nhược điểm của giải mã Greedy

Dưới đây là những ưu và nhược điểm của thuật toán giải mã Greedy:

- **Ưu điểm:** Đơn giản, dễ cài đặt, tốc độ xử lý rất nhanh. Yêu cầu tài nguyên tính toán thấp, phù hợp với các thiết bị hạn chế về phần cứng.
- **Nhược điểm:** Không xét đến các khả năng kết hợp ký tự khác, dễ bỏ sót chuỗi có xác suất tổng thể cao hơn. Độ chính xác thường thấp hơn so với các phương pháp giải mã phức tạp hơn như Beam Search, đặc biệt với các chuỗi dài hoặc dữ liệu nhiều nhiễu. Không tận dụng được thông tin ngữ cảnh hoặc mô hình ngôn ngữ.

2.7.2 Giải mã Beam Search

Phương pháp Greedy decoding chỉ chọn ký tự có xác suất cao nhất tại mỗi bước thời gian, dễ bỏ qua các chuỗi có xác suất tổng thể lớn hơn do không xét đến toàn bộ không gian chuỗi. Để khắc phục, **Beam Search** được sử dụng nhằm giữ lại đồng thời nhiều chuỗi ứng viên tốt nhất, giúp tăng khả năng tìm được chuỗi tối ưu.

2.7.2.1 Nguyên lý và quy tắc giải mã Beam Search

Beam Search là thuật toán tìm kiếm xấp xỉ, tại mỗi bước thời gian chỉ giữ lại B chuỗi (beam) có xác suất cao nhất. Ở mỗi bước, mỗi beam được mở rộng bằng cách thêm từng ký tự trong bảng chữ cái, sau đó chỉ chọn lại B beam tốt nhất để tiếp tục. Nhờ đó, thuật toán vừa đảm bảo hiệu quả tính toán, vừa tăng xác suất tìm đúng chuỗi tối ưu.

Đặc thù của CTC:

- Ký tự blank (“_”) dùng để phân tách các ký tự giống nhau liên tiếp.
- Khi chuyển từ chuỗi nhãn tạm thời (có blank và lặp ký tự) sang chuỗi kết quả, ta loại bỏ blank và gộp các ký tự giống nhau liên tiếp thành một.

Khi tính xác suất cho một beam, cần phân biệt hai trường hợp:

- Beam kết thúc bằng blank
- Beam kết thúc bằng ký tự thông thường

2.7.2.2 Công thức cập nhật xác suất Beam

Gọi l là một beam (chuỗi ký tự tạm thời), tại thời điểm t :

$$p_b(l, t) : \text{Xác suất beam } l \text{ kết thúc bằng ký tự blank tại thời điểm } t \quad (2.19)$$

$$p_{nb}(l, t) : \text{Xác suất beam } l \text{ kết thúc bằng ký tự không phải blank tại thời điểm } t \quad (2.20)$$

Tổng xác suất của beam l tại thời điểm t là:

$$p(l, t) = p_b(l, t) + p_{nb}(l, t) \quad (2.21)$$

Quy tắc cập nhật xác suất:

- Khi thêm ký tự blank:

$$p_b(l, t) = p(_, t) \cdot p(l, t - 1) \quad (2.22)$$

- Khi thêm ký tự $c \neq _$:

– Nếu c khác ký tự cuối cùng của l :

$$p_{nb}(l + c, t) = p(c, t) \cdot p(l, t - 1) \quad (2.23)$$

– Nếu c trùng ký tự cuối cùng của l :

$$p_{nb}(l + c, t) = p(c, t) \cdot p_b(l, t - 1) \quad (2.24)$$

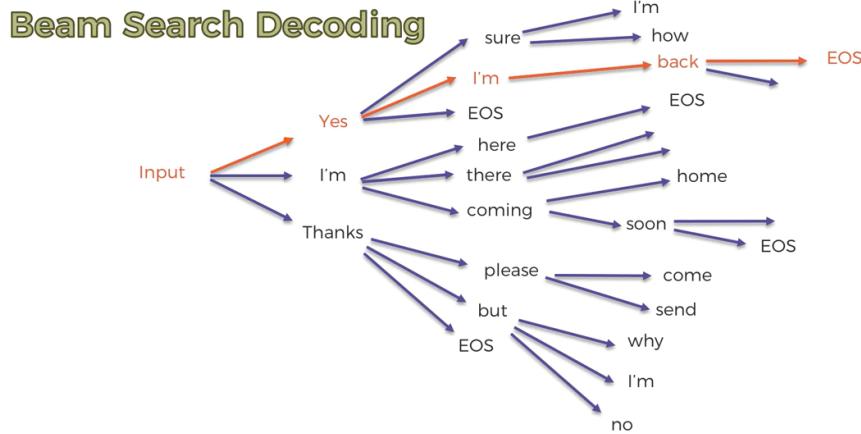
Nếu có nhiều beam tạo thành cùng một chuỗi ký tự sau khi loại bỏ blank và gộp ký tự lặp, ta cộng xác suất của các beam này lại.

2.7.2.3 Quy trình thuật toán Beam Search trong CTC

Dưới đây là quy trình thực hiện thuật toán Beam Search trong CTC:

1. **Khởi tạo:** Tập beam ban đầu $\mathcal{B}_0 = \{\emptyset\}$ (chuỗi rỗng), $p_b(\emptyset, 0) = 1$, $p_{nb}(\emptyset, 0) = 0$.
2. **Lặp với từng bước thời gian** $t = 1, 2, \dots, T$:
 - Với mỗi beam l trong tập beam hiện tại:
 - Mở rộng beam l bằng tất cả các ký tự $c \in A$ (bao gồm blank).
 - Tính toán p_b và p_{nb} cho từng beam mở rộng theo công thức trên.
 - Gộp các beam trùng nhau (sau khi loại bỏ blank và ký tự lặp) bằng cách cộng xác suất.
 - Chọn lại B beam có xác suất tổng lớn nhất để giữ lại cho bước tiếp theo.
3. **Kết thúc:** Sau bước thời gian cuối cùng T , chọn beam có xác suất tổng lớn nhất. Loại bỏ ký tự blank và gộp ký tự lặp liên tiếp để thu được chuỗi kết quả cuối cùng.

Ví dụ minh họa



Hình 2.22. Ví dụ thuật toán beam search với beam width = 3

Giả sử beam width $B = 3$, ở mỗi bước thời gian, ta giữ lại 3 beam có xác suất cao nhất. Mỗi beam được mở rộng bằng tất cả ký tự, tính lại xác suất theo quy tắc trên, sau đó chọn lại 3 beam tốt nhất để tiếp tục cho bước tiếp theo. Hình 2.22. minh họa trực quan quá trình beam search với beam width = 3.

2.7.2.4 *Ưu và nhược điểm giải mã Beam search*

Dưới đây là những ưu điểm và nhược điểm của thuật toán giải mã Beam:

- **Ưu điểm:** Giữ lại nhiều khả năng đầu ra, tránh bỏ sót chuỗi tốt do sai sót ở các bước đầu. Cân bằng giữa hiệu quả tính toán và độ chính xác. Có thể tích hợp thêm mô hình ngôn ngữ (Language Model) để tăng chất lượng giải mã.
- **Hạn chế:** Độ phức tạp tính toán tăng theo beam width. Nếu beam width quá nhỏ, vẫn có thể bỏ sót chuỗi tối ưu.

2.7.2.5 *Giải mã Beam Search kết hợp Language model*

CTC Beam Search có thể được thực hiện với độ rộng chùm B bằng cách tìm B ứng viên có xác suất cao nhất ở mỗi bước thời gian t , dựa trên xác suất tổng hợp từ mô hình âm thanh và mô hình ngôn ngữ (LM) [19]. Phương pháp CTC Beam Search kết hợp Language Model hoạt động bằng cách mở rộng mỗi beam tại từng bước thời gian t , không chỉ dựa trên xác suất âm học P_{AM} mà còn tích hợp thêm xác suất từ mô hình ngôn ngữ P_{LM} . Tại mỗi bước, thuật toán giữ lại B ứng viên tốt nhất (beam width B), với xác suất tổng hợp được tính như sau:

$$\hat{y} = B(\pi^*), \quad \text{với} \quad \pi^* = \arg \max_{\pi} \prod_t P_{AM}(\pi_t | \mathbf{x}) \cdot P_{LM}(y_{\pi_t} | B(\pi_{1:t-1}))^\alpha \quad (2.25)$$

Trong đó:

- $P_{\text{AM}}(\pi_t | \mathbf{x})$: Xác suất âm học tại thời điểm t cho nhãn π_t , do mô hình âm thanh sinh ra.
- $P_{\text{LM}}(y_{\pi_t} | B(\pi_{1:t-1}))$: Xác suất từ mô hình ngôn ngữ dự đoán nhãn tiếp theo dựa trên chuỗi nhãn đã thu gọn đến thời điểm $t - 1$.
- α : Siêu tham số điều chỉnh mức ảnh hưởng của mô hình ngôn ngữ lên tổng xác suất.
- $B(\cdot)$: Hàm thu gọn (collapse) loại bỏ các ký tự lặp và ký tự *blank* trong đường dẫn CTC.

Việc kết hợp mô hình ngôn ngữ trong quá trình beam search giúp giải mã đầu ra trở nên hợp lý và tự nhiên hơn, giảm thiểu các lỗi do mô hình âm học gây ra, đặc biệt trong các trường hợp từ đồng âm hoặc phát âm không rõ ràng. Ngoài ra, mô hình ngôn ngữ còn giúp hệ thống dự đoán chính xác hơn các cụm từ, câu có tính ngữ nghĩa cao. Tuy nhiên, hiệu quả của phương pháp này phụ thuộc đáng kể vào việc lựa chọn giá trị phù hợp cho các siêu tham số như α và độ rộng beam B . Nếu α quá lớn, mô hình ngôn ngữ sẽ lấn át thông tin âm học, dẫn đến các kết quả không sát với tín hiệu đầu vào; ngược lại, nếu α quá nhỏ, hệ thống sẽ không tận dụng được lợi ích của ngữ cảnh ngôn ngữ. Do đó, các tham số này thường được tối ưu hóa thông qua thực nghiệm trên tập dữ liệu kiểm thử.

2.8 Hệ điều hành Android

2.8.1 Giới thiệu hệ điều hành Android

Android là một hệ điều hành mã nguồn mở được phát triển bởi Google, hiện đang thống trị thị trường thiết bị di động toàn cầu. Theo thống kê đến cuối năm 2021, Android chiếm khoảng 71% thị phần trên toàn thế giới và con số này tiếp tục có xu hướng gia tăng trong những năm gần đây. Hệ điều hành này đã được triển khai trên hàng trăm triệu thiết bị di động tại hơn 190 quốc gia, minh chứng cho sự phổ biến và độ phủ rộng lớn của nó trên phạm vi toàn cầu.

Trong khuôn khổ đề tài phát triển ứng dụng nhận dạng tiếng nói, sinh viên lựa chọn Android là nền tảng chính để triển khai, nhờ vào các đặc điểm kỹ thuật và lợi thế vượt trội mà hệ điều hành này mang lại. Cụ thể:

- **Mã nguồn mở và cộng đồng hỗ trợ rộng lớn:** Android được phát hành dưới giấy phép mã nguồn mở, cho phép các nhà phát triển tự do nghiên cứu, tùy biến và cải tiến theo nhu cầu cụ thể. Bên cạnh đó, cộng đồng phát triển Android rất năng động, cung cấp hàng loạt tài nguyên như tài liệu kỹ thuật, diễn đàn, thư viện mã nguồn và các công cụ hỗ trợ, giúp giải quyết nhanh chóng các vấn đề trong quá trình phát triển ứng dụng.

- **Tuân thủ các hướng dẫn thiết kế của Google:** Google cung cấp bộ nguyên tắc thiết kế giao diện (Material Design), giúp các nhà phát triển xây dựng giao diện người dùng nhất quán, thân thiện và dễ sử dụng. Điều này góp phần nâng cao trải nghiệm người dùng, đặc biệt quan trọng đối với các ứng dụng có tính tương tác cao như nhận dạng tiếng nói.
- **Khả năng phân mảnh linh hoạt:** Dù phân mảnh trong hệ sinh thái Android đôi khi được xem là bất lợi, nhưng trong một số trường hợp, nó lại cung cấp khả năng tùy biến mạnh mẽ. Ứng dụng có thể khai thác tính năng đa nhiệm, như chạy đồng thời hai hoạt động (activities) trên cùng một màn hình, từ đó tối ưu hóa hiệu quả sử dụng và khả năng tương tác của người dùng.
- **Quy trình phát hành ứng dụng thuận tiện:** So với các nền tảng di động khác như iOS, quá trình đưa ứng dụng lên Google Play Store thường đơn giản và ít ràng buộc hơn. Nhà phát triển có thể nhanh chóng triển khai, cập nhật và phân phối ứng dụng tới người dùng trên toàn cầu, rút ngắn đáng kể vòng đời phát triển phần mềm.

Với những ưu điểm nổi bật trên, Android là lựa chọn phù hợp và hiệu quả để phát triển và triển khai các ứng dụng nhận dạng tiếng nói trong bối cảnh thực tế hiện nay.

2.8.2 Một số phần mềm phổ biến để lập trình Android [2]

Lập trình Android hiện nay được hỗ trợ bởi nhiều công cụ phát triển đa dạng và mạnh mẽ. Dưới đây là một số phần mềm phổ biến được sử dụng rộng rãi trong cộng đồng phát triển Android:

2.8.2.1 *Android Studio*

Dưới đây là những thông tin cơ bản về phần mềm lập trình ứng dụng điện thoại Android Studio:

- Android Studio là **môi trường phát triển chính thức (official IDE)** do Google phát hành dành cho lập trình ứng dụng Android.
- Dựa trên nền tảng IntelliJ IDEA, Android Studio hỗ trợ mạnh mẽ việc viết mã bằng ngôn ngữ Java và Kotlin.
- Tích hợp sẵn trình giả lập (emulator), công cụ thiết kế giao diện kéo-thả, trình quản lý gói (Gradle), hệ thống debug và phân tích hiệu năng.

2.8.2.2 *Visual Studio Code (VS Code)*

VS Code là một trình soạn thảo mã nguồn phổ biến hiện nay nhờ khả năng mở rộng linh hoạt và giao diện nhẹ. Mặc dù không được thiết kế chuyên biệt cho Android, nhưng VS Code hỗ trợ tốt phát triển ứng dụng đa nền tảng thông qua các phân mảng:

- Visual Studio Code là một trình soạn thảo mã nguồn nhẹ, mã nguồn mở, do Microsoft phát triển.
- VS Code hỗ trợ lập trình Android thông qua các phần mở rộng như Flutter, React Native, hoặc NativeScript.

2.8.2.3 *Eclipse (với Android SDK)*

Eclipse từng là công cụ chính để phát triển ứng dụng Android trong giai đoạn đầu, trước khi Android Studio ra mắt. Dù hiện tại không còn được Google hỗ trợ chính thức, Eclipse vẫn được sử dụng trong một số trường hợp đặc biệt:

- Trước khi Android Studio ra đời, Eclipse từng là công cụ phổ biến để lập trình Android.
- Hiện nay đã không còn được Google hỗ trợ chính thức, nhưng vẫn có thể dùng cho các dự án cũ hoặc mục đích học thuật.
- Cần cài đặt thêm Android Development Tools (ADT) để hỗ trợ phát triển ứng dụng Android.

2.9 Kết luận chương

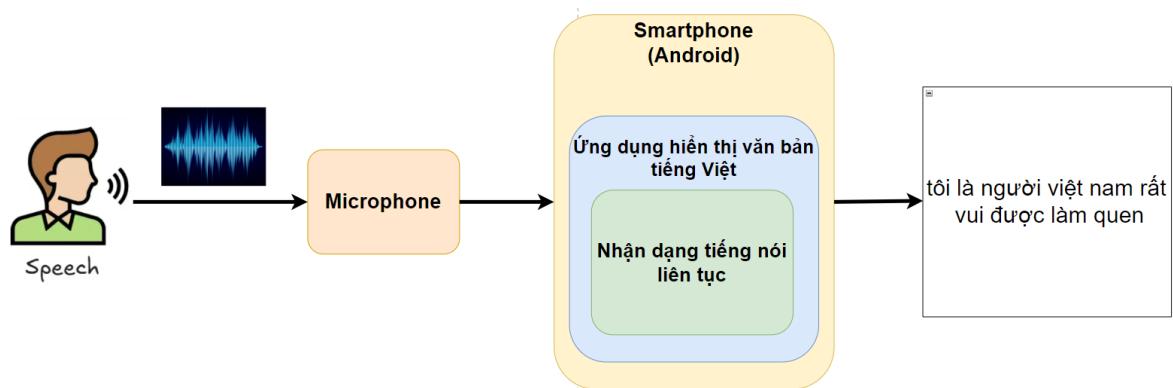
Trong chương này, sinh viên đã trình bày tổng quan về hệ thống nhận dạng tiếng nói, bao gồm hai pha chính là pha huấn luyện và pha nhận dạng. Đồng thời, các cơ sở lý thuyết của từng khối chức năng trong hệ thống cũng đã được phân tích, bao gồm khái trich xuất đặc trưng, thuật toán huấn luyện, mô hình âm học và khái giải mã. Đây là những nền tảng quan trọng làm cơ sở cho việc thiết kế và triển khai hệ thống nhận dạng tiếng nói tiếng Việt. Trong chương 3, sinh viên sẽ trình bày chi tiết về quá trình thiết kế hệ thống nhận dạng tiếng nói tiếng Việt, bao gồm xây dựng pipeline huấn luyện, tích hợp mô hình vào phần mềm ứng dụng trên nền tảng Android, cũng như triển khai chế độ nhận dạng trực tuyến (Online ASR) để đáp ứng yêu cầu thời gian thực.

CHƯƠNG 3. THIẾT KẾ HỆ THỐNG

Ở Chương 2, sinh viên đã trình bày chi tiết cơ sở lý thuyết liên quan đến xử lý tiếng nói, phương pháp trích xuất đặc trưng, thuật toán huấn luyện và các kỹ thuật giải mã. Những kiến thức này đóng vai trò nền tảng cho quá trình thiết kế hệ thống nhận dạng tiếng nói. Trong chương này, sinh viên sẽ trình bày cụ thể quá trình thiết kế và triển khai hệ thống nhận dạng tiếng nói tiếng Việt. Nội dung bao gồm: khảo sát và lựa chọn thuật toán huấn luyện phù hợp, thiết kế kiến trúc hệ thống nhận dạng, phát triển phần mềm ứng dụng trên nền tảng Android, và triển khai chế độ nhận dạng tiếng nói trực tuyến (online ASR) để đáp ứng yêu cầu thời gian thực.

3.1 Đề xuất sơ đồ khái niệm hệ thống

Tiếng nói của người dùng được thu trực tiếp thông qua micro tích hợp trên điện thoại thông minh, thiết bị hiện nay phổ biến và hầu như đều được trang bị sẵn microphone. Tín hiệu âm thanh sau khi được thu sẽ được đưa vào mô hình nhận dạng tiếng nói đã được tích hợp trong ứng dụng Android. Mô hình này thực hiện quá trình xử lý và dự đoán chuỗi văn bản tương ứng, sau đó kết quả được hiển thị lên giao diện cho người dùng, như minh họa trong Hình 3.1..



Hình 3.1. Sơ đồ khái niệm hoạt động của toàn bộ hệ thống

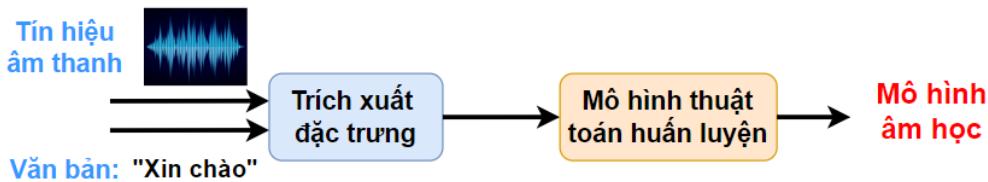
3.2 Thiết kế phần mềm nhận dạng tiếng nói tiếng Việt liên tục

3.2.1 Đề xuất sơ đồ khái niệm của phần mềm nhận dạng tiếng nói tiếng Việt liên tục

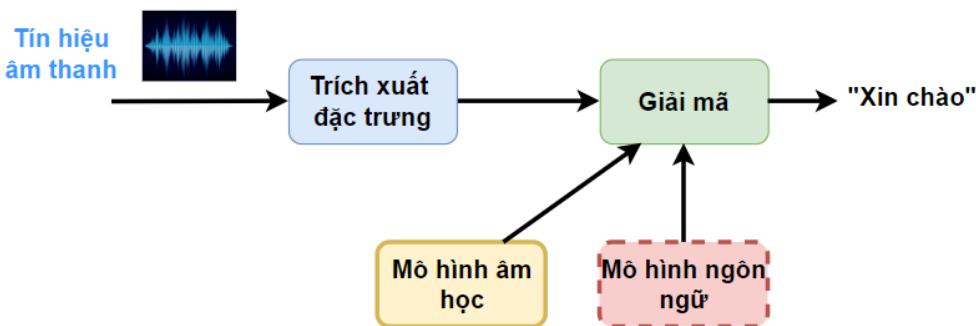
Dựa trên cơ sở lý thuyết về hệ thống nhận dạng tiếng nói đã trình bày ở mục 2.1, sinh viên đề xuất hệ thống nhận dạng tiếng nói tiếng Việt được xây dựng theo hai giai đoạn chính: giai đoạn huấn luyện mô hình âm học và giai đoạn nhận dạng tiếng nói. Quy trình chi tiết được minh họa trong Hình 3.2..

Ở giai đoạn huấn luyện (Hình 3.2.a), hệ thống sử dụng tập dữ liệu gồm các cặp tín hiệu âm thanh và phiên bản văn bản tương ứng. Các tín hiệu âm thanh được chuyển đổi thành đặc trưng thông qua bước trích xuất đặc trưng (feature extraction), sau đó được đưa vào một mô hình học sâu kết hợp với thuật toán huấn luyện và hàm mất mát

CTC (Connectionist Temporal Classification). Kết quả của quá trình này là mô hình âm học có khả năng học được mối liên hệ giữa chuỗi đặc trưng âm thanh và chuỗi ký tự đầu ra, phục vụ cho việc giải mã trong giai đoạn tiếp theo.



(a) Quy trình huấn luyện mô hình trong hệ thống được đề xuất.



(b) Quy trình nhận dạng tiếng nói để xuất

Hình 3.2. Thiết kế hệ thống nhận dạng tiếng nói tiếng Việt

Trong giai đoạn nhận dạng tiếng nói (Hình 3.2.b), tín hiệu âm thanh đầu vào được xử lý tương tự qua bước trích xuất đặc trưng, sau đó được đưa vào mô hình âm học đã được huấn luyện. Đầu ra của mô hình kết hợp với mô hình ngôn ngữ và lớp CTC để thực hiện quá trình giải mã, nhằm tạo ra văn bản tương ứng với chuỗi âm thanh đầu vào.

3.2.2 Lựa chọn trích xuất đặc trưng tiếng nói và triển khai

3.2.2.1 Trích xuất đặc trưng Mel-Spectrogram

Trong quá trình thiết kế hệ thống nhận dạng tiếng nói liên tục tiếng Việt, sinh viên lựa chọn Mel spectrogram làm đặc trưng đầu vào cho mô hình thay vì các đặc trưng khác như MFCC, spectrogram tuyến tính hay filterbank thô, vì các lý do sau:

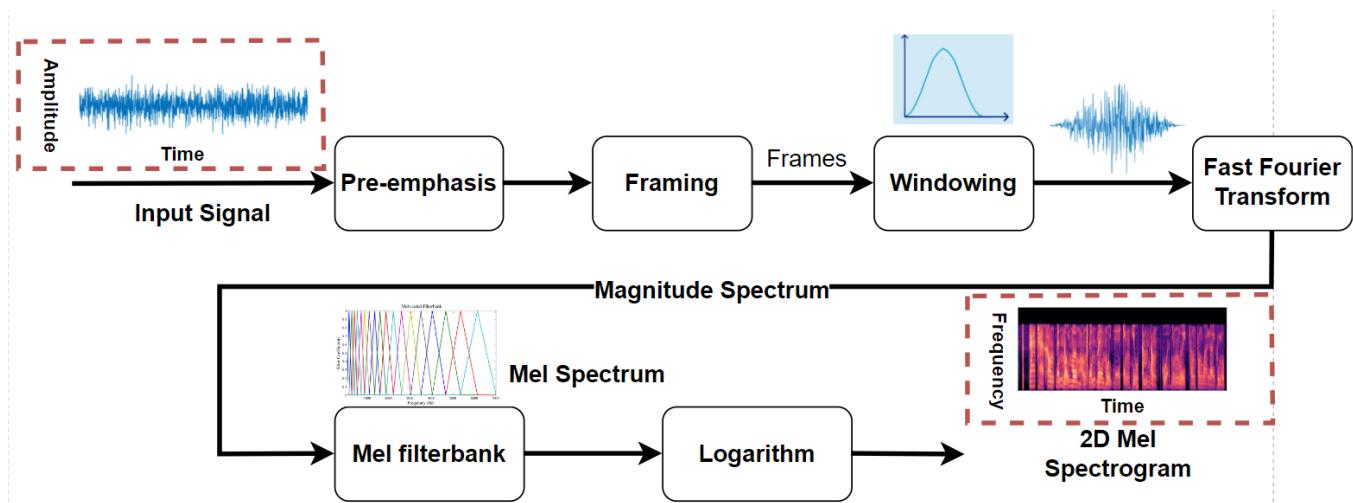
- Phù hợp với đặc trưng ngôn ngữ tiếng Việt:** Tiếng Việt là ngôn ngữ đơn lập, có hệ thống thanh điệu phong phú (6 thanh điệu), do đó thông tin về cường độ và cao độ (tần số) cần được bảo toàn. Mel spectrogram giữ nguyên cấu trúc năng lượng theo thời gian và tần số mà không làm giảm chiều như MFCC, giúp mô hình học được tốt hơn các yếu tố liên quan đến thanh điệu.
- Tính liên tục và trực quan hơn MFCC:** MFCC sử dụng DCT (biến đổi cosin rời rạc) để rút trích hệ số, dẫn đến việc làm mất thông tin cục bộ về tần số. Trong

khi đó, Mel spectrogram giữ lại cấu trúc thời gian – tần số rõ ràng, giúp các mô hình học sâu khai thác hiệu quả hơn.

- **Tương thích tốt với mô hình hiện đại:** Nhiều nghiên cứu gần đây (ví dụ như Conformer, Squeezeformer) sử dụng Mel spectrogram làm đầu vào vì nó tương thích tốt với các mô hình mạng tích chập hoặc attention-based. Nó đóng vai trò như một "bức ảnh" thời gian – tần số, giúp mô hình học đặc trưng trừu tượng một cách trực tiếp. Từ các phân tích trên, Mel spectrogram được chọn là đặc trưng đầu vào trong hệ thống ASR nhằm đảm bảo tính biểu diễn cao, khả năng học sâu tốt và hiệu suất thực nghiệm ổn định trong môi trường tiếng Việt thực tế.

3.2.2.2 Một số thông số cho trích xuất đặc trưng Mel-spectrogram [7]

Để trích xuất đặc trưng Mel-spectrogram một cách hiệu quả, việc lựa chọn bộ tham số phù hợp là yếu tố then chốt, ảnh hưởng trực tiếp đến chất lượng đặc trưng và hiệu suất mô hình.



Hình 3.3. Luồng hoạt động của trích xuất Mel-spectrogram từ tín hiệu tiếng nói thô

Hình 3.3. mô tả luồng hoạt động tổng quát của quá trình trích xuất Mel-spectrogram từ tín hiệu tiếng nói thô. Theo đó, tín hiệu đầu vào sẽ trải qua các bước: chia khung (framing), áp dụng cửa sổ (windowing), biến đổi Fourier nhanh (FFT), ánh xạ sang thang Mel, và cuối cùng là lấy log năng lượng theo từng dải tần. Việc lựa chọn cẩn thận các tham số như kích thước khung, bước nhảy, số lượng dải Mel hay tần số cắt không chỉ giúp biểu diễn tốt hơn các đặc trưng ngữ âm mà còn đảm bảo tính tương thích với kiến trúc của mô hình học sâu được sử dụng, cụ thể như sau:

- **Tần số lấy mẫu (sample rate):** 16000 Hz

Đây là tần số phổ biến trong xử lý tiếng nói, đủ để tái tạo các đặc trưng âm thanh quan trọng của lời nói (theo định lý Nyquist).

- **Kích thước cửa sổ (frame size):** 25 ms và **bước nhảy (stride):** 10 ms
Cấu hình này giúp đảm bảo độ phân giải thời gian – tần số cân bằng, là giá trị chuẩn trong nhiều hệ thống ASR hiện đại.
- **Số lượng dải Mel (num_feature_bins):** 80
Số lượng này giúp biểu diễn đủ chi tiết về tần số trong phổ Mel mà không gây tăng kích thước đầu vào quá lớn.
- **Loại đặc trưng (feature_type):** log_mel_spectrogram
Việc sử dụng thang log giúp làm nổi bật các thông tin âm sắc quan trọng và giảm ảnh hưởng của nhiễu nhỏ trong tín hiệu.
- **Tiền nhấn (pre-emphasis):** 0.97
Hệ số này giúp làm nổi bật các tần số cao trong tín hiệu âm thanh, vốn thường yếu hơn nhưng lại chứa nhiều thông tin phân biệt âm vị.
- **Chuẩn hóa tín hiệu (normalize_signal):** True và **chuẩn hóa đặc trưng (normalize_feature):** True
Giúp mô hình học tốt hơn và giảm sai số do khác biệt năng lượng giữa các đoạn âm thanh.
- **Không chuẩn hóa theo từng khung (normalize_per_frame):** False
Quyết định này nhằm bảo toàn tương quan năng lượng giữa các khung, có ý nghĩa với mô hình học chuỗi.

3.2.3 Khảo sát và lựa chọn kiến trúc mô hình huấn luyện

Một thách thức lớn đối với tiếng Việt — một ngôn ngữ ít tài nguyên — là việc thu thập đủ dữ liệu huấn luyện chất lượng. Để giải quyết vấn đề này, đề tài áp dụng chiến lược fine-tuning trên mô hình pretrained. Cụ thể, mô hình âm học ban đầu được huấn luyện trên một tập dữ liệu lớn, sau đó được tinh chỉnh (fine-tuned) trên tập dữ liệu tiếng Việt nhằm kế thừa các đặc trưng ngôn ngữ và âm học quan trọng đã học trước đó. Cách tiếp cận này không chỉ giúp rút ngắn thời gian huấn luyện, mà còn cải thiện hiệu quả nhận dạng trong điều kiện dữ liệu hạn chế, từ đó nâng cao tính khả thi và hiệu quả của hệ thống nhận dạng tiếng nói tiếng Việt được đề xuất. Nhằm lựa chọn mô hình tiền huấn luyện và thuật toán huấn luyện phù hợp, sinh viên đã tiến hành khảo sát các xu hướng mô hình và phương pháp huấn luyện hiện đại trong lĩnh vực nhận dạng tiếng nói, cụ thể được trình bày dưới đây.

3.2.3.1 Wav2vec [21]

Mô hình Wav2vec lần đầu được giới thiệu năm 2020 trong bài báo "wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations". Trong nghiên cứu này, tác giả lần đầu tiên chứng minh rằng việc học các biểu diễn đặc trưng mạnh mẽ từ âm thanh lời nói không có nhãn, sau đó tinh chỉnh trên một lượng nhỏ dữ

liệu có nhãn, có thể đạt hiệu quả vượt trội so với các phương pháp bán giám sát phức tạp hơn [21]. Mô hình đã thực nghiệm trên Librispeech và LibriVox. Tập LibriVox được xử lý để lấy dữ liệu âm thanh và thu được khoảng 53.2 nghìn giờ âm thanh. Sau đó model được tinh chỉnh trên tập dữ liệu có gán nhãn Librispeech gồm 960h âm thanh. Kết quả được công khai như hình 3.4. Hai phiên bản mô hình được sử

Model	Unlabeled data	LM	dev		test	
			clean	other	clean	other
Supervised						
CTC Transf [51]	-	CLM+Transf.	2.20	4.94	2.47	5.45
S2S Transf. [51]	-	CLM+Transf.	2.10	4.79	2.33	5.17
Transf. Transducer [60]	-	Transf.	-	-	2.0	4.6
ContextNet [17]	-	LSTM	1.9	3.9	1.9	4.1
Conformer [15]	-	LSTM	2.1	4.3	1.9	3.9
Semi-supervised						
CTC Transf. + PL [51]	LV-60k	CLM+Transf.	2.10	4.79	2.33	4.54
S2S Transf. + PL [51]	LV-60k	CLM+Transf.	2.00	3.65	2.09	4.11
Iter. pseudo-labeling [58]	LV-60k	4-gram+Transf.	1.85	3.26	2.10	4.01
Noisy student [42]	LV-60k	LSTM	1.6	3.4	1.7	3.4
This work						
LARGE - from scratch	-	Transf.	1.7	4.3	2.1	4.6
BASE	LS-960	Transf.	1.8	4.7	2.1	4.8
LARGE	LS-960	Transf.	1.7	3.9	2.0	4.1
	LV-60k	Transf.	1.6	3.0	1.8	3.3

Hình 3.4. Kết quả của wav2vec khi tinh chỉnh trên 960h dữ liệu có nhãn Librispeech

dụng là BASE (95m parameters) and LARGE (317m parameters). Đã có nhiều công trình nghiên cứu lấy cảm hứng từ wav2vec để finetuning trên tập dữ liệu tiếng Việt và đạt được hiệu quả tương đối tốt như trong dự án "Vietnamese end-to-end speech recognition using wav2vec 2.0" của tác giả Nguyen Thai Bin năm 2021 [4]. Nghiên cứu này được pre-trained trên 13 000 tiếng của tiếng Việt từ youtube (không gán nhãn) và được tinh chỉnh trên 250 tiếng dữ liệu có nhãn dán từ VLSP trên tốc độ trích mẫu dữ liệu là 16kHz. Tác giả đã sử dụng mô hình wav2vec2 để fine-tuning mô hình. Phiên bản được sử dụng là "Base" với kích thước 95M tham số. Nghiên cứu cũng sử dụng thêm language model 4-grams đã được trained trên 2GB dữ liệu văn bản. Kết quả đạt được như bảng dưới đây:

Bảng 3.1. Kết quả WER của wav2vec trên tập dữ liệu tiếng Việt [4]

Phương pháp	VIVOS	Common Voice VI	VLSP-T1	VLSP-T2
Không dùng LM	10.77	18.34	13.33	51.45
Dùng LM 4-gram	6.15	11.52	9.11	40.81

3.2.3.2 Whisper[5]

Mô hình Whisper được giới thiệu vào năm 2022 trong bài báo [5]. Đây là mô hình hiện đại đã cho thấy khả năng vượt trội so với những mô hình khác và được kế thừa rộng rãi và phổ biến bởi các nhà nghiên cứu hiện nay. Nghiên cứu đã cung cấp 5 phiên bản gồm:

Bảng 3.2. Các phiên bản của Whisper trong nghiên cứu [5, 6]

Phiên bản	Tiny	Base	Small	Medium	Large
Tham số	39M	74M	244M	769M	1550M

Kết quả mô hình đạt được trên tập Librispeech là 5.2% trên Librispeech other với phiên bản large-V2. Đã có những nghiên cứu ứng dụng phát triển trên mô hình Whisper cho tập dữ liệu tiếng Việt, điển hình như "PhoWhisper" [6]. Họ đã fine-tuning mô hình Whisper với cả 5 phiên bản trên 844 tiếng dữ liệu của tiếng Việt.

Bảng 3.3. So sánh Word Error Rate (WER) giữa các mô hình ASR trên tiếng Việt[6]

Model	#params	CMV-Vi	VIVOS	VLSP Task-1	VLSP Task-2
wav2vec2-base-vietnamese-250h	95M	102.04	10.83	21.02	50.35
wav2vec2-base-vi-VLSP 2020	95M	103.71	9.90	16.82	44.91
wav2vec2-large-vi-VLSP 2020	317M	101.41	8.61	15.18	36.75
PhoWhisper _{tiny}	39M	19.05	10.41	20.74	49.85
PhoWhisper _{base}	74M	16.19	8.46	17.00	43.01
PhoWhisper _{small}	244M	11.08	6.33	15.93	32.96
PhoWhisper _{medium}	769M	8.27	4.97	14.12	26.85
PhoWhisper _{large}	1.55B	8.14	4.67	13.75	26.68

3.2.3.3 Squeezeformer

Trong thời gian gần đây, mô hình Conformer đã nổi lên như một kiến trúc backbone tiêu chuẩn cho nhiều hệ thống nhận dạng tiếng nói hiện đại. Conformer kết hợp một cách hiệu quả giữa cơ chế attention toàn cục và mạng tích chập (convolution) cục bộ, cho phép mô hình đồng thời khai thác các đặc trưng ngữ cảnh rộng và cục bộ từ tín hiệu âm thanh đầu vào [22]. Tuy nhiên, theo nghiên cứu của Kim và cộng sự [7], thông qua một chuỗi phân tích và đánh giá có hệ thống, nhóm tác giả đã chỉ ra rằng các lựa chọn thiết kế trong kiến trúc Conformer hiện tại vẫn chưa thật sự tối ưu, đặc biệt về mặt hiệu suất và chi phí tính toán.

Xuất phát từ việc đánh giá lại kiến trúc Conformer ở cả cấp độ vĩ mô (macro-level design) và cấp độ vi mô (micro-level optimization), nhóm tác giả đã đề xuất mô hình SqueezeFormer với một loạt cải tiến về cấu trúc. Những cải tiến này không chỉ giúp giảm độ phức tạp mô hình, mà còn nâng cao độ chính xác nhận dạng tiếng nói, trong khi vẫn duy trì hoặc giảm chi phí tính toán so với Conformer gốc. Nhờ vào những ưu điểm này, SqueezeFormer đang dần trở thành một lựa chọn hấp dẫn cho các hệ thống nhận dạng tiếng nói trong môi trường có giới hạn tài nguyên. **Ở cấp độ vi mô, Squeezeformer đã có những cải tiến như:**

- Tích hợp cấu trúc Temporal U-Net nhằm giảm chi phí tính toán của các mô đun multi-head attention khi xử lý các chuỗi dài.

- Sử dụng cấu trúc khối đơn giản hơn, bao gồm mô đun multi-head attention hoặc convolution kết hợp với một mô đun feed-forward, thay thế cho cấu trúc Macaron phức tạp trong Conformer.
- Đơn giản hóa các hàm kích hoạt trong khối convolution
- Loại bỏ các thao tác chuẩn hóa lớp dư thừa
- Áp dụng một lớp downsampling theo chiều sâu nhằm giảm độ dài chuỗi một cách hiệu quả hơn.

Mô hình Squeezeformer được xuất bản với 6 phiên bản tùy kích thước và được train trên tập dữ liệu 960 Librispeech bằng tiếng Anh. Dưới đây là kết quả WER của các phiên bản mô hình so với các mô hình khác trên tập dữ liệu Librispeech.

Bảng 3.4. So sánh hiệu suất của các phiên bản Squeezeformer trên tập LibriSpeech [7]

Model	dev-clean	dev-other	test-clean	test-other	Params (M)	GFLOPs	Thp (ex/s)
Squeezeformer-XS	3.63	9.30	3.74	9.09	9.5	19.5	763
Squeezeformer-S	2.80	7.49	3.08	7.47	18.6	26.3	602
Squeezeformer-SM	2.71	6.98	2.79	6.89	28.2	42.7	558
Squeezeformer-M	2.43	6.54	2.56	6.50	55.6	72.0	431
Squeezeformer-ML	2.34	6.08	2.61	6.05	125.1	169.2	268
Squeezeformer-L	2.27	5.77	2.57	5.97	236.3	277.9	207

3.2.3.4 Lựa chọn thuật toán huấn luyện

a, Những tiêu chí so sánh

Trong quá trình lựa chọn thuật toán huấn luyện cho hệ thống nhận dạng tiếng nói tự động (ASR), sinh viên có đưa ra một số tiêu chí dựa trên yêu cầu của hệ thống như sau:

- **Hiệu suất và độ chính xác**
- **Kích thước và tốc độ suy luận:** Do mục đích cuối cùng là chạy mô hình trên Android app, do đó mô hình cần phải gọn nhẹ để triển khai trong môi trường có tài nguyên hạn chế. Độ trễ suy luận cũng cần phải thấp để đảm bảo trải nghiệm của người dùng khi sử dụng hệ thống.
- **Khả năng hỗ trợ tiếng Việt:** Ưu tiên những mô hình được pre-trained trên tập dữ liệu tiếng Anh hoặc ngôn ngữ khác có cùng bảng chữ cái La-tinh. Nó sẽ giúp cho việc nắm bắt ngữ cảnh và dự đoán tốt hơn vì tiếng Việt cũng là ngôn ngữ hiện đại sử dụng bảng chữ cái La-tinh.
- Do giới hạn phần cứng nên mô hình lựa chọn cũng không quá lớn vì một phần giới hạn về cả mặt dữ liệu.

- **Trình khả dụng và mức tiêu thụ tài nguyên:** Ưu tiên mô hình có thể tích hợp tốt trên TensorFlow Lite, ... để hướng tới mục đích deploy mô hình chạy offline trên Android app.

b, Lựa chọn thuật toán huấn luyện

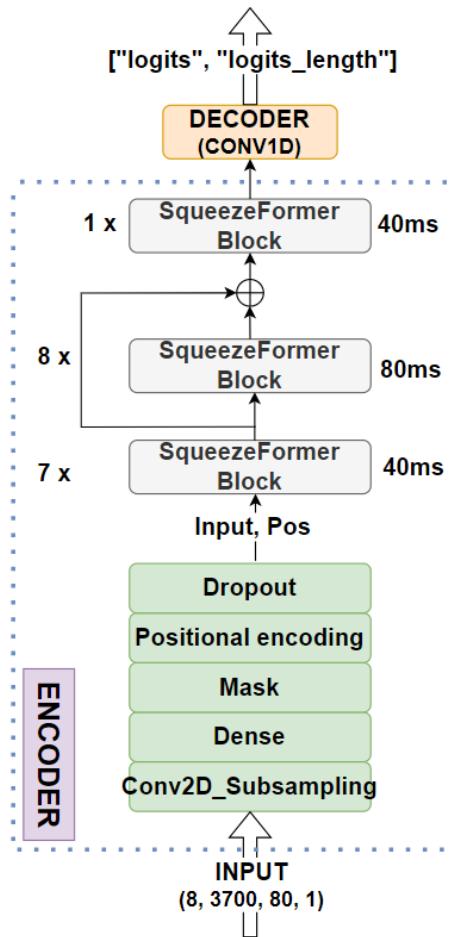
Các mô hình ASR end-to-end gần đây thường bao gồm một encoder, nhận đầu vào là tín hiệu giọng nói (tức là chuỗi các khung giọng nói) và trích xuất các đặc trưng âm học cấp cao, và một decoder, chuyển đổi các đặc trưng được trích xuất từ encoder thành một chuỗi văn bản. Kiến trúc mô hình của encoder quyết định sức mạnh biểu diễn của mô hình ASR và khả năng trích xuất đặc trưng âm học từ tín hiệu đầu vào. Do đó, một kiến trúc mạnh mẽ là rất quan trọng cho hiệu suất tổng thể.

Dựa trên các tiêu chí đánh giá đã trình bày ở phần trước, mô hình **Squeezeformer** được xem là một lựa chọn phù hợp cho bài toán nhận dạng tiếng nói tiếng Việt. Cho đến thời điểm hiện tại, theo tìm hiểu của sinh viên, vẫn chưa có công bố chính thức nào sử dụng Squeezeformer cho nhận dạng tiếng nói tự động (ASR) đối với tiếng Việt. Mô hình này gồm sáu phiên bản khác nhau, trong đó phiên bản nhỏ nhất (XS) chỉ có khoảng **9 triệu tham số**, rất thích hợp để tinh chỉnh (fine-tuning) trên các tập dữ liệu tiếng Việt hạn chế. Hơn nữa, nhờ vào kiến trúc gọn nhẹ, Squeezeformer-XS có tiềm năng lớn trong việc triển khai trên các thiết bị Android phổ thông, cho phép hoạt động hiệu quả trong chế độ ngoại tuyến. Đây là một lợi thế rõ rệt khi so sánh với các mô hình lớn khác vốn đòi hỏi nhiều tài nguyên và khó tích hợp trên nền tảng di động.

3.2.4 Mô hình Squeezeformer

Dựa trên hình 3.5. về kiến trúc mô hình SqueezeFormer phiên bản XS, sinh viên trình bày luồng dữ liệu đầu vào và đầu ra qua từng phần như sau:

- **Input:** Dữ liệu đầu vào có dạng $(B, T_{raw}, F, 1)$, với B là batch size (ví dụ $B = 8$), T_{raw} là số khung thời gian (ví dụ 3700), và $F = 80$ là số đặc trưng Mel.
- **Conv2D Subsampling:** Thực hiện hai lớp tích chập 2D với stride 2 để giảm độ dài chuỗi theo thời gian. Đầu ra có dạng (B, T_{sub}, C) , ví dụ $(8, 232, 256)$.
- **Dense:** Biến đổi đầu ra từ Conv2D sang số chiều phù hợp d_{model} để đưa vào khối Transformer. Đầu ra vẫn giữ dạng (B, T_{sub}, d_{model}) .
- **Mask:** Tạo mặt nạ attention để bỏ qua padding. Đầu ra là một tensor mặt nạ có dạng $(B, 1, T_{sub})$.
- **Positional Encoding:** Thêm thông tin vị trí vào đầu ra của Dense. Kích thước không đổi (B, T_{sub}, d_{model}) .
- **Dropout:** Áp dụng dropout nhằm giảm overfitting. Kích thước đầu ra không thay đổi.



Hình 3.5. Kiến trúc mô hình Squeezeformer

- **Squeezeformer Blocks:** Gồm ba tầng:
 - 7 khối đầu (tầng đầu): giữ nguyên độ phân giải thời gian, tương ứng khoảng 40ms.
 - 8 khối giữa: tăng receptive field (có thể thông qua pooling), tương ứng khoảng 80ms.
 - 1 khối cuối: tinh chỉnh đặc trưng đầu ra, tương ứng khoảng 40ms.
 Tất cả các khối đều có đầu vào và đầu ra dạng (B, T', d_{model}) .
- **Decoder (Conv1D):** Lớp tích chập 1D để ánh xạ đặc trưng thành phân phối xác suất trên không gian nhãm. Đầu ra gồm:
 - logits có dạng (B, T', V) với V là số lượng ký tự hoặc nhãm.
 - logits_length là độ dài thực tế của chuỗi đầu ra.

3.2.4.1 Những cải tiến của mô hình SqueezeFormer

Những cải tiến chính trong kiến trúc Squeezeformer-XS so với mô hình Conformer, như thể hiện trong hình 3.5. bao gồm:

- **Giảm dư thừa tạm thời (temporal redundancy):** Các biểu diễn đặc trưng học được giữa các khung thời gian lân cận thường có tính tương đồng cao, đặc biệt ở các tầng sâu của mạng. Để giảm thiểu chi phí tính toán không cần thiết, Squeezeformer tích hợp một kiến trúc Temporal U-Net, bao gồm một lớp downsampling (giảm tần số lấy mẫu) ở giữa mạng và một lớp upsampling nhẹ ở cuối nhằm khôi phục độ phân giải thời gian, từ đó cải thiện độ ổn định trong huấn luyện.
- **Thiết kế lại khối attention-convolution:** Khối lai giữa multi-head attention (MHA) và convolution được thiết kế lại, trong đó mỗi mô-đun MHA hoặc convolution đều được theo sau bởi một mô-đun feed-forward duy nhất, như minh họa trong Hình 3.6..
- **Tối ưu hóa vi kiến trúc (micro-architecture):**
 - Thay thế các hàm kích hoạt GLU bằng Swish nhằm cải thiện khả năng học phi tuyến.
 - Đơn giản hóa Layer Normalization bằng cách thay thế các lớp pre-Layer Normalization dư thừa bằng các lớp post-Layer Normalization có hệ số học được, từ đó giúp tích hợp vào các lớp khác và giảm chi phí suy luận về 0.
 - Áp dụng depthwise separable convolution cho lớp sub-sampling đầu tiên, giúp giảm đáng kể số phép toán dấu phẩy động (FLOPs) mà không làm giảm hiệu năng mô hình.

3.2.4.2 Kiến trúc Temporal U-Net

Temporal U-Net là một kiến trúc cải tiến dựa trên mô hình Conformer, với mục tiêu giảm độ phức tạp tính toán nhưng vẫn duy trì được khả năng biểu diễn mạnh mẽ và độ chính xác cao. Trong Conformer gốc, các phép toán attention có độ phức tạp bậc hai theo độ dài chuỗi đầu vào, và độ dài này được giữ nguyên trong suốt toàn bộ mô hình. Temporal U-Net khắc phục hạn chế này bằng cách áp dụng cơ chế **temporal downsampling** và **upsampling** tại các vị trí thích hợp trong mạng, nhằm tận dụng tính dư thừa theo thời gian trong các biểu diễn đặc trưng sâu [22].

Phân tích thực nghiệm trên tập *dev-other* của LibriSpeech trong nghiên cứu [7] cho thấy rằng các vector embedding của các khung tiếng nói lân cận có độ tương đồng cosine trung bình rất cao ở các lớp sâu — lên đến 95%. Ngay cả với các khung cách nhau 4 bước thời gian, độ tương đồng vẫn vượt quá 80%. Quan sát này phản ánh mức độ dư thừa cao tại các tầng sâu của mạng, làm phát sinh chi phí tính toán không cần thiết. Dựa trên phát hiện đó, Temporal U-Net thực hiện **downsampling chiều thời gian** sau khối thứ 7 bằng một lớp *depthwise separable convolution* với kernel size = 3 và stride = 2. Việc này tương đương với việc tăng bước thời gian từ 40 ms lên 80 ms,

giúp giảm độ dài chuỗi và từ đó giảm khoảng 4 lần chi phí tính toán của các phép toán attention ở các khối sau, được biểu diễn trên hình 3.5.

Bên cạnh đó, Temporal U-Net còn tích hợp thêm một nhánh **upsampling đối xứng**, tương tự như kiến trúc U-Net truyền thống trong thị giác máy tính [23]. Cụ thể, các đặc trưng từ tầng thấp (với tần số 40 ms) được truyền trực tiếp đến nhánh upsampling thông qua các **skip connections**, nhằm khôi phục lại độ phân giải thời gian ban đầu trước khi đưa tới decoder. Việc khôi phục này đảm bảo rằng các thông tin chi tiết theo thời gian vẫn được giữ lại, giúp mô hình ánh xạ chính xác từng khung tiếng nói đến các nhãn đầu ra tương ứng.

3.2.4.3 Thiết kế khối SqueezeFormer

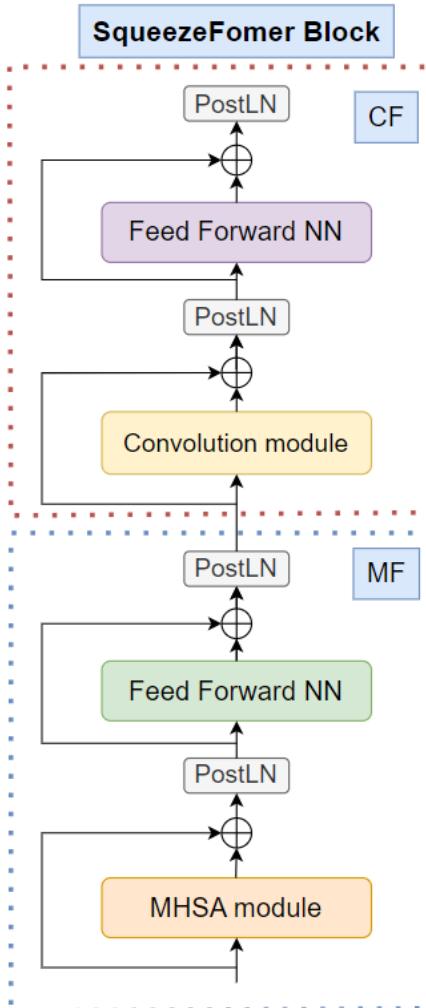
Hình 3.6. minh họa kiến trúc của một khối SqueezeFormer — được thiết kế nhằm giảm trùng lặp chức năng giữa attention và convolution trong mô hình Conformer gốc. Trong Conformer, mỗi khối gồm chuỗi mô-đun theo thứ tự: Feed-forward (F), Multi-head Self-Attention (M), Convolution (C), và thêm một Feed-forward (F), tạo thành cấu trúc F–M–C–F (hay còn gọi là Macaron-style). Thiết kế này giúp khai thác cả phụ thuộc toàn cục (qua attention) và thông tin cục bộ (qua convolution), tuy nhiên dẫn đến chồng lặp tính năng và tăng độ phức tạp. Để giải quyết điều đó, SqueezeFormer tổ chức lại khối thành hai nhánh độc lập:

- **Nhánh CF (Convolution + Feed Forward)**: bao gồm một mô-đun convolution (ví dụ: depthwise separable convolution) và một mạng neural feed-forward. Cả hai đều được nối với chuẩn hóa sau lớp (Post-LN) và kết nối tắt (residual). Nhánh này tập trung học đặc trưng cục bộ theo thời gian.
- **Nhánh MF (Multi-Head Self-Attention + Feed Forward)**: bao gồm mô-đun multi-head self-attention (MHSA) kết hợp với một lớp feed-forward. Cấu trúc cũng sử dụng chuẩn hóa Post-LN và kết nối residual. Nhánh này giúp mô hình hóa các phụ thuộc dài hạn giữa các khung thời gian.

Không giống Conformer sử dụng đồng thời cả MHA và convolution trong cùng một khối, SqueezeFormer chỉ sử dụng một trong hai nhánh (MF hoặc CF) tại một thời điểm, giúp:

- Giảm số lượng tham số và FLOPs.
- Tránh dư thừa biểu diễn.
- Tăng hiệu quả tính toán mà vẫn duy trì khả năng biểu diễn.

Ngoài ra, thiết kế này cũng loại bỏ cấu trúc **Macaron-style FFN**, tức không còn hai mô-đun feed-forward kẹp hai bên attention như trong Transformer truyền thống. Quyết định này dựa trên các kết quả nghiên cứu trong lĩnh vực NLP và thị giác máy



Hình 3.6. Kiến trúc khối SqueezeFormer, bao gồm hai nhánh CF và MF

tính, cho thấy rằng việc sử dụng **một mô-đun feed-forward duy nhất sau attention hoặc convolution** là đủ để đạt hiệu quả cao, đồng thời đơn giản hóa huấn luyện. Các mô hình như BERT [24] và Vision Transformer (ViT) [25] cũng áp dụng chiến lược tương tự.

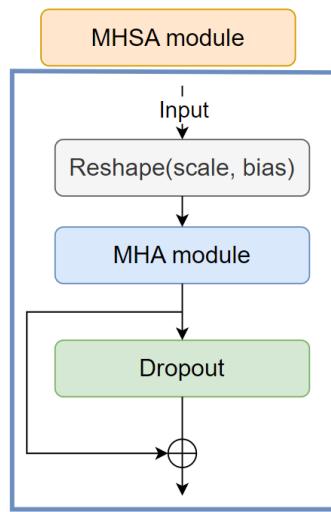
Tổng thể, cách tổ chức theo hướng “nén lại” (squeeze) trong SqueezeFormer mang lại sự cân bằng giữa hiệu suất và chi phí tính toán, giúp mô hình dễ mở rộng hơn cho các hệ thống nhận dạng tiếng nói quy mô lớn.

a, MHSA Module (Attention đa đầu)

Thực hiện phép attention đa đầu, cho phép mô hình học các mối quan hệ dài hạn giữa các vị trí trong chuỗi đặc trưng đầu vào. Cấu trúc cụ thể bao gồm:

- **Reshape (scale, bias):** Thay thế cho pre-LayerNorm truyền thống, giúp chuẩn hóa đầu vào bằng tham số học được, đơn giản hóa mô hình và tiết kiệm FLOPs.
- **Multi-Head Attention:** Trích xuất thông tin toàn cục (global dependencies) giữa các thời điểm khác nhau trong chuỗi âm thanh đầu vào.

- **Dropout + Residual:** Giúp chống overfitting và ổn định lan truyền gradient

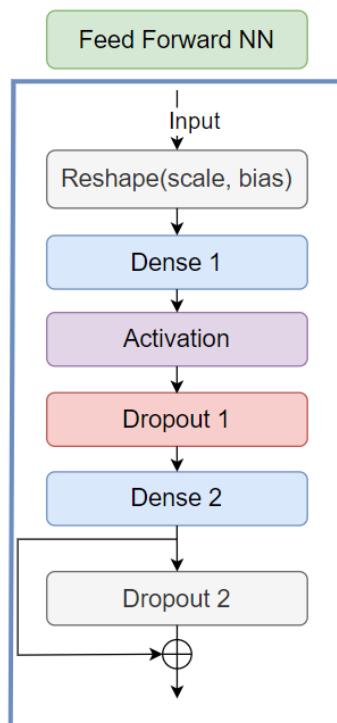


Hình 3.7. Cấu trúc module multi-head attention trong SqueezeFormer

Module này giúp mô hình nắm bắt các quan hệ toàn cục (global context) trong chuỗi âm thanh, rất quan trọng cho nhận dạng tiếng nói liên tục, thay preLN bằng phép scale học được để giảm độ phức tạp.

b, Feed Forward Neural Network (FFN) Module

Module này có vai trò biến đổi phi tuyến đặc trưng đầu vào, tăng khả năng biểu



Hình 3.8. Cấu trúc module feed-forward neural network trong SqueezeFormer

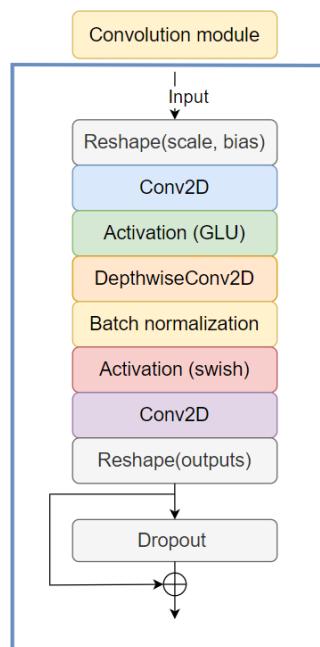
diễn của mô hình. Cấu trúc module gồm có:

- **Reshape (scale, bias):** Chuẩn hóa đầu vào trước khi xử lý.
- **Dense 1:** Mở rộng chiều không gian đặc trưng, cho phép mạng học được biểu diễn phong phú hơn.
- **Activation:** Giúp mô hình học các quan hệ phức tạp hơn, các quan hệ phi tuyến.
- **Dropout 1:** Hạn chế quá khớp
- **Dense 2:** Giảm chiều trở lại kích thước ban đầu
- **Dropout 2 + Residual:** Bổ sung kết nối tắt (residual), giữ nguyên thông tin ban đầu và giúp huấn luyện ổn định hơn với mạng sâu.

FFN là thành phần thiết yếu trong các khối Transformer hiện đại, giúp tăng khả năng học phi tuyến và đa dạng hóa biểu diễn đặc trưng. Trong kiến trúc này, FFN được sử dụng hai lần: trước MHSA và sau Convolution, tạo thành khối FMCF.

c, Convolution Module

Module này khai thác các đặc trưng cục bộ (local features) trong chuỗi âm thanh, bổ sung cho khả năng attention vốn thiên về toàn cục.



Hình 3.9. Cấu trúc module Convolution trong SqueezeFormer

Kiến trúc module Convolution trong khối SqueezeFormer như trên hình 3.9. gồm có:

- **Reshape (scale, bias):** Thay thế pre-LN bằng phép scale học được.
- **Conv2D:** Tích chập 1×1 mở rộng đặc trưng đầu vào.
- **Activation (Swish):** Tăng khả năng phi tuyến, thay thế ReLU truyền thống.

- **DepthwiseConv2D:** Tích chập theo thời gian từng kênh, giúp trích xuất đặc trưng cục bộ hiệu quả hơn với kernel size lớn
- **Batch Normalization:** Chuẩn hóa để ổn định huấn luyện
- **Conv2D:** Thu nhỏ chiều về lại như ban đầu.
- **Reshape(outputs):** Chuyển đổi định dạng đầu ra để phù hợp với tầng kế tiếp.
- **Dropout + Residual:** Giúp cải thiện ổn định và chống overfitting.

Convolution hoạt động như một attention cục bộ (local attention), bổ sung cho MHSA trong việc trích xuất đặc trưng không gian-thời gian.

3.2.5 Xây dựng mô hình ngôn ngữ

3.2.5.1 Công cụ sử dụng

Trong quá trình xây dựng hệ thống nhận dạng tiếng nói, sinh viên không chỉ tập trung vào mô hình âm học mà còn chú trọng đến việc cải thiện giai đoạn giải mã bằng cách tích hợp mô hình ngôn ngữ. Việc lựa chọn công cụ huấn luyện mô hình ngôn ngữ phù hợp đóng vai trò quan trọng nhằm đảm bảo hiệu quả và tốc độ xử lý trong quá trình suy diễn. Sau khi tham khảo và so sánh các giải pháp hiện có, sinh viên quyết định sử dụng **KenLM** với những lý do cụ thể như sau:

- **KenLM** là một trong những công cụ xây dựng mô hình ngôn ngữ n-gram nhanh và hiệu quả nhất hiện nay, được sử dụng rộng rãi trong nhiều hệ thống nhận dạng tiếng nói nhờ khả năng xử lý văn bản lớn và tối ưu hóa cho tốc độ suy diễn (inference).
- Như sinh viên đã trình bày ở mục 2.7, mô hình ngôn ngữ n-gram sẽ được tích hợp vào quá trình giải mã sử dụng phương pháp beam search nhằm nâng cao độ chính xác của hệ thống. Thư viện **pyctcdecode** hỗ trợ kỹ thuật này, với điều kiện là cần cung cấp một mô hình ngôn ngữ n-gram đã được huấn luyện và lưu dưới định dạng nhị phân (.bin), định dạng này có thể được tạo một cách hiệu quả thông qua **công cụ KenLM**.

3.2.5.2 Đề xuất xây dựng cơ sở dữ liệu văn bản tiếng Việt

Trước khi huấn luyện mô hình ngôn ngữ, dữ liệu văn bản cần được tiền xử lý nhằm đảm bảo tính nhất quán, loại bỏ nhiễu và chuẩn hóa định dạng văn bản. Sinh viên đã thực hiện tiền xử lý dữ liệu văn bản phục vụ huấn luyện mô hình ngôn ngữ n-gram như sau:

- **Chuyển đổi chữ hoa thành chữ thường:** Giúp thống nhất cách biểu diễn từ vựng, giảm số lượng từ cần học và tránh phân biệt không cần thiết giữa các dạng viết hoa, viết thường.

- **Loại bỏ dấu câu và ký tự đặc biệt:** Bao gồm việc loại bỏ các dấu câu (như dấu chấm, phẩy, chấm than, dấu ngoặc, v.v.), biểu tượng phi ngôn ngữ (như emoji, ký hiệu đặc biệt) và các ký tự không có giá trị về mặt ngôn ngữ trong ngữ cảnh huấn luyện mô hình.
- **Chuẩn hóa khoảng trắng:** Loại bỏ khoảng trắng dư thừa và đảm bảo mỗi từ cách nhau đúng một dấu cách. Điều này giúp định dạng văn bản sạch hơn và phù hợp với quá trình tokenization.

Nguồn dữ liệu đầu vào phục vụ huấn luyện mô hình ngôn ngữ bao gồm nhiều tập văn bản tiếng Việt có tính đa dạng cao về nội dung và văn phong, cụ thể:

- **Các tập dữ liệu tiếng Việt công khai:** Bao gồm VLSP 2020, FPT Speech Corpus và VIVOS, là những tập dữ liệu tiêu chuẩn trong lĩnh vực xử lý tiếng nói và ngôn ngữ tiếng Việt.
- **Văn bản truyện cổ tích tiếng Việt:** Được thu thập từ các kho truyện trực tuyến, mang tính văn học và truyền thống, giúp mô hình tiếp cận được phong cách ngôn ngữ tự nhiên, đa dạng.
- **Bài báo từ các trang tin tức uy tín:** Văn bản được thu thập từ các nguồn như Báo Nhân Dân, Báo Tuổi Trẻ và Báo 24h, cung cấp lượng lớn dữ liệu chính thống, có cấu trúc rõ ràng và đa dạng về chủ đề.
- **Đoạn hội thoại phi cấu trúc:** Bao gồm các đoạn chat, thảo luận, nhận xét... được trích xuất từ Internet, giúp mô hình thích ứng với ngôn ngữ đời sống hằng ngày, dạng không trang trọng hoặc ít chuẩn hóa.

3.2.5.3 Tao mô hình ngôn ngữ n-gram ở định dạng nhị phân

Dưới đây là quy trình chi tiết mà sinh viên đã thực hiện để tạo mô hình ngôn ngữ n-gram ở định dạng nhị phân sử dụng KenLM.

a) Chuẩn bị môi trường và cài đặt công cụ KenLM

Để huấn luyện mô hình ngôn ngữ n-gram, công cụ KenLM được sinh viên sử dụng. Đây là một thư viện mã nguồn mở được phát triển bằng ngôn ngữ lập trình C++, nổi bật nhờ hiệu năng cao và khả năng xử lý các mô hình ngôn ngữ lớn với tốc độ nhanh. KenLM được sử dụng rộng rãi trong các hệ thống nhận dạng tiếng nói hiện đại. Công cụ này có thể được cài đặt và biên dịch thủ công trên các nền tảng như Linux hoặc Windows (qua WSL hoặc sử dụng CMake và MinGW/MSVC). Các bước cài đặt và biên dịch như sau:

```

git clone https://github.com/kpu/kenlm.git
cd kenlm
mkdir build && cd build
cmake ..
make -j$(nproc)

```

Sau khi biên dịch thành công, hai công cụ chính của KenLM sẽ được tạo ra:

- `lmplz`: Dùng để huấn luyện mô hình n-gram và xuất ra định dạng `.arpa`.
- `build_binary`: Dùng để chuyển đổi mô hình từ định dạng `.arpa` sang định dạng nhị phân để sử dụng hiệu quả hơn trong quá trình giải mã.

b) Huấn luyện mô hình ngôn ngữ n-gram

Sau khi hoàn tất quá trình tiền xử lý dữ liệu văn bản, mô hình n-gram được huấn luyện bằng công cụ `lmplz` với cú pháp sau:

```
bin/lmplz -o 5 < text.txt > model.arpa
```

Trong đó:

- `-o 5` chỉ định bậc của mô hình n-gram (trong ví dụ này là 5-gram).
- `text.txt` là tệp văn bản đầu vào đã được chuẩn hóa và tiền xử lý.
- `model.arpa` là tệp đầu ra chứa mô hình n-gram ở định dạng chuẩn ARPA.

c) Chuyển đổi sang định dạng nhị phân

Để tối ưu hóa tốc độ suy diễn (inference) và giảm mức sử dụng bộ nhớ, mô hình `.arpa` có thể được chuyển sang định dạng nhị phân bằng công cụ `build_binary`, với câu lệnh:

```
bin/build_binary model.arpa model.bin
```

Tệp `model.bin` sau khi được tạo ra có thể được tích hợp trực tiếp vào các decoder, chẳng hạn như `pyctcdecode`, để sử dụng trong quá trình giải mã kết hợp với beam search và mô hình ngôn ngữ n-gram.

3.3 Thiết kế phần mềm ứng dụng soạn thảo văn bản tiếng Việt sử dụng công nghệ nhận dạng tiếng nói

3.3.1 Công cụ sử dụng

3.3.1.1 Phần mềm [26]

Trong quá trình phát triển ứng dụng Android, sinh viên lựa chọn sử dụng phần mềm **Android Studio** – môi trường phát triển tích hợp (IDE) chính thức do Google phát hành, được thiết kế chuyên biệt cho việc phát triển ứng dụng trên nền tảng Android. Android Studio được xây dựng trên nền tảng IntelliJ IDEA, cung cấp các công cụ hỗ trợ lập trình mạnh mẽ như trình chỉnh sửa mã nguồn thông minh, hệ thống kiểm lỗi tự động, công cụ thiết kế giao diện trực quan và khả năng mô phỏng ứng dụng thông qua trình giả lập hoặc thiết bị thực. Hình 3.10. minh họa giao diện của Android Studio.



Hình 3.10. Môi trường phát triển Android Studio

Một số ưu điểm nổi bật của Android Studio mà sinh viên tận dụng trong quá trình phát triển bao gồm:

- Hỗ trợ các ngôn ngữ lập trình phổ biến như **Java**, **Kotlin**, và **C++**, mang lại sự linh hoạt trong lựa chọn công nghệ.
- Trình thiết kế giao diện trực quan (Layout Editor) cho phép sinh viên dễ dàng xây dựng bố cục giao diện bằng thao tác kéo-thả và xem trước thời gian thực.
- Tích hợp hệ thống kiểm thử trên cả trình giả lập và thiết bị vật lý, giúp sinh viên kiểm tra tính đúng đắn và hiệu năng của ứng dụng ngay trong quá trình phát triển.
- Giao diện phát triển thân thiện, hỗ trợ tùy biến cao, phù hợp với nhu cầu học tập và nghiên cứu của sinh viên.

3.3.1.2 Ngôn ngữ lập trình

Sinh viên lựa chọn sử dụng **Kotlin** để xây dựng giao diện ứng dụng. Việc lựa chọn này được đưa ra dựa trên các ưu điểm nổi bật của Kotlin so với Java truyền thống:

- **Khả năng tương thích cao với Java:** Kotlin có thể kết hợp và sử dụng lại các thư viện Java hiện có, hỗ trợ quá trình phát triển ứng dụng một cách linh hoạt.

- **Hỗ trợ trực tiếp cho mô hình học máy:** Kotlin có khả năng tương tác tốt với các thư viện như TensorFlow Lite, phù hợp với hướng tiếp cận ứng dụng nhận dạng giọng nói sử dụng mô hình học sâu.
- **Đa nền tảng:** Kotlin hỗ trợ phát triển ứng dụng trên nhiều nền tảng khác nhau như Windows, macOS, Linux, và cả các thiết bị nhúng như Raspberry Pi.
- **Cú pháp ngắn gọn, an toàn:** Giảm thiểu lỗi lập trình thường gặp như null pointer, đồng thời giúp mã nguồn trở nên súc tích và dễ bảo trì.
- **Cộng đồng hỗ trợ mạnh mẽ:** Với sự phát triển mạnh mẽ của Kotlin, sinh viên dễ dàng tiếp cận tài liệu và nhận được sự hỗ trợ từ cộng đồng lập trình viên.

3.3.2 Thiết kế giao diện ứng dụng soạn thảo văn bản

Giao diện người dùng của ứng dụng được sinh viên thiết kế nhằm đảm bảo tính trực quan, dễ sử dụng và phục vụ đúng chức năng của hệ thống nhận dạng tiếng nói. Các chức năng chính của giao diện bao gồm:

- Cung cấp nút bấm rõ ràng cho phép người dùng bắt đầu quá trình ghi âm giọng nói một cách thuận tiện.
- Hiển thị chính xác và rõ ràng nội dung lời nói đã được nhận dạng (transcript) sau khi xử lý.
- Thiết kế đơn giản, dễ nhìn, thân thiện với người dùng nhằm nâng cao trải nghiệm sử dụng, đặc biệt trên các thiết bị di động.

Với những yêu tố trên, giao diện của ứng dụng hướng tới như hình 3.11.



Hình 3.11. Giao diện ứng dụng hiển thị văn bản tiếng Việt

Giao diện sinh viên thiết kế bao gồm ba thành phần cơ bản:

- Một nút bấm (Button) cho phép người dùng bắt đầu thu âm;
- Một vùng văn bản (TextView) hiển thị kết quả nhận dạng;
- Một thanh tiến trình (ProgressBar) kiểu vòng quay dùng để biểu thị trạng thái đang ghi âm, mặc định ẩn đi và chỉ hiển thị khi quá trình thu âm diễn ra.

```
<Button  
    android:id="@+id/selectBtn"  
    android:layout_width="200dp"  
    android:layout_height="60dp"  
    android:text="Bắt đầu nói"  
    android:textSize="24sp"  
    android:layout_alignParentBottom="true"  
    android:layout_centerHorizontal="true"  
    android:layout_marginBottom="24dp" />  
  
<TextView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:id="@+id/resView"  
    android:text="prediction"  
    android:textSize="20sp"  
    android:layout_marginRight="10dp"  
    android:layout_centerInParent="true"/>  
    <!-- Hiệu ứng đang ghi âm, mặc định ẩn -->  
<ProgressBar  
    android:id="@+id/recordingProgress"  
    style="?android:attr/progressBarStyleLarge"  
    android:visibility="gone"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_centerInParent="true"/>
```

Hình 3.12. Đoạn mã thiết kế giao diện ứng dụng hiển thị văn bản tiếng Việt

3.3.3 Tích hợp hệ thống nhận dạng tiếng nói tiếng Việt vào ứng dụng hiển thị văn bản trên smartphone

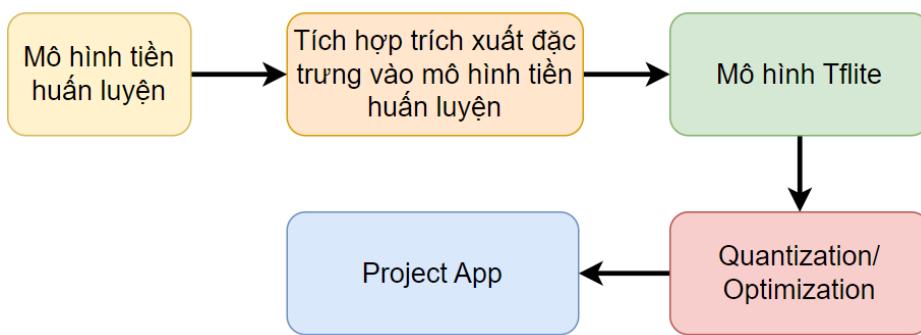
3.3.3.1 Chuyển đổi mô hình đã huấn luyện sang TensorFlow Lite [27]

Mục đích: Đây là một bước quan trọng nhằm tối ưu mô hình học sâu để có thể triển khai hiệu quả trên các thiết bị di động với tài nguyên phần cứng hạn chế. Việc chuyển đổi mô hình sang định dạng .tflite giúp giảm kích thước mô hình, cải thiện tốc độ suy diễn (inference) và giảm mức tiêu thụ bộ nhớ, đồng thời vẫn giữ được độ chính xác chấp nhận được.

Quá trình này được sinh viên thực hiện trên máy tính xách tay, sử dụng công cụ TensorFlow Lite Converter. Đầu vào là mô hình học sâu đã được huấn luyện với định dạng chuẩn (SavedModel hoặc .h5), đầu ra là mô hình đã chuyển đổi sang định dạng .tflite, sẵn sàng để tích hợp vào ứng dụng Android hiển thị văn bản tiếng Việt từ giọng nói.

Quy trình chuyển đổi được mô tả cụ thể trong Hình 3.13.. Dưới đây là các bước chuyển mô hình đã huấn luyện sang định dạng .tflite:

- **Bước 1:** Tích hợp quy trình trích xuất đặc trưng (feature extraction) trực tiếp vào mô hình tiền huấn luyện nhằm đảm bảo tính nhất quán và tối ưu hóa hiệu suất xử lý. Việc trích xuất đặc trưng như Mel-spectrogram nên được thực hiện bằng ngôn ngữ chuyên dụng cho học sâu như Python (với thư viện TensorFlow), thay vì Kotlin. Bởi lẽ, Kotlin không được tối ưu cho các phép toán xử lý tín hiệu số và tính toán ma trận quy mô lớn – những yếu tố then chốt trong trích xuất đặc trưng âm thanh. Nếu thực hiện bước này bằng Kotlin trên thiết bị Android, mô hình có thể bị suy giảm độ chính xác do sai lệch chuẩn hóa, thiếu các kỹ thuật xử lý tín hiệu nâng cao, hoặc hiệu năng không đảm bảo trong thời gian thực.



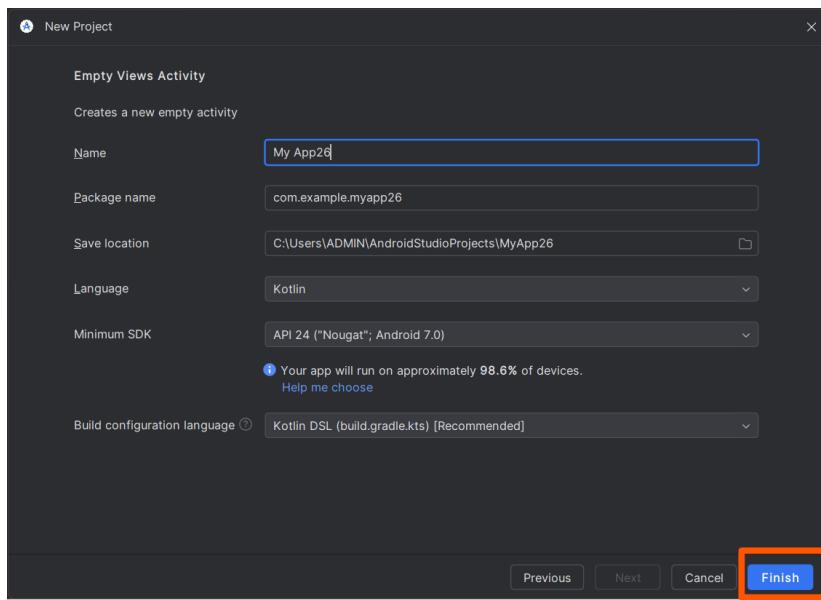
Hình 3.13. Quy trình chuyển đổi mô hình đã huấn luyện sang mô hình định dạng .tflite

- **Bước 2:** Chuyển đổi mô hình TensorFlow sang định dạng TensorFlow Lite. Sau khi mô hình được huấn luyện hoàn chỉnh, cần tiến hành chuyển đổi sang định dạng nhẹ hơn để phù hợp với môi trường tài nguyên hạn chế như thiết bị di động. TensorFlow Lite cung cấp công cụ chuyển đổi ('TFLiteConverter') giúp rút gọn mô hình gốc bằng cách loại bỏ các thành phần không cần thiết và chuẩn hóa kiến trúc mạng. Mô hình sau khi chuyển đổi sẽ có kích thước nhỏ hơn, thời gian khởi tạo nhanh hơn và tiêu tốn ít tài nguyên hơn khi thực thi.
- **Bước 3:** Tối ưu hóa mô hình để nâng cao hiệu suất suy luận. Công cụ TensorFlow Model Optimization Toolkit hỗ trợ các kỹ thuật như lượng hóa (quantization), giúp giảm kích thước mô hình và tăng tốc độ xử lý trên thiết bị di động. Việc tối ưu cần được cân nhắc kỹ để duy trì độ chính xác của mô hình trong khi đáp ứng yêu cầu về tài nguyên hạn chế như bộ nhớ, CPU và pin.
- **Bước 4:** Đầu ra của quy trình này là mô hình đã chuyển đổi sang định dạng .tflite và sẵn sàng tích hợp vào ứng dụng hiển thị văn bản tiếng Việt trên Android sẽ được trình bày trong mục dưới đây.

3.3.3.2 Tích hợp mô hình nhận dạng tiếng nói vào ứng dụng hiển thị văn bản tiếng Việt trên Android

Để triển khai mô hình nhận dạng tiếng nói trên thiết bị di động, sinh viên tiến hành tích hợp mô hình đã huấn luyện dưới định dạng .tflite vào ứng dụng Android. Quy trình bao gồm các bước chính như sau:

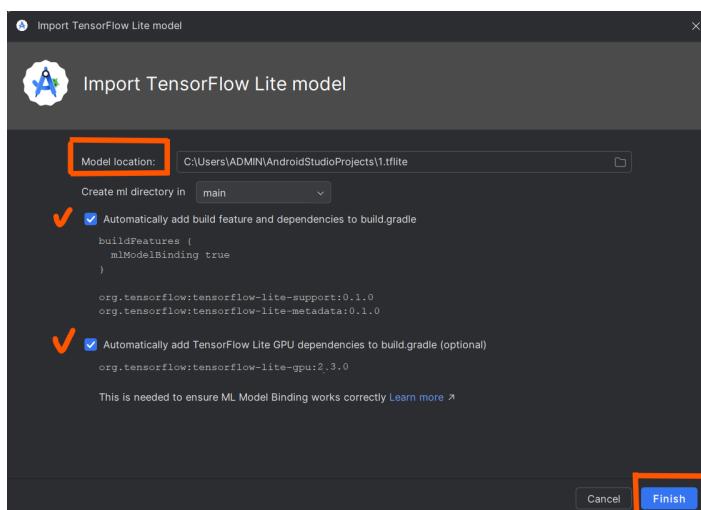
Bước 1: Tạo một dự án mới trên Android Studio.



Hình 3.14. Khởi tạo ứng dụng trên IDE Android Studio

Bước 2: Tích hợp mô hình .tflite vào dự án Android Studio.

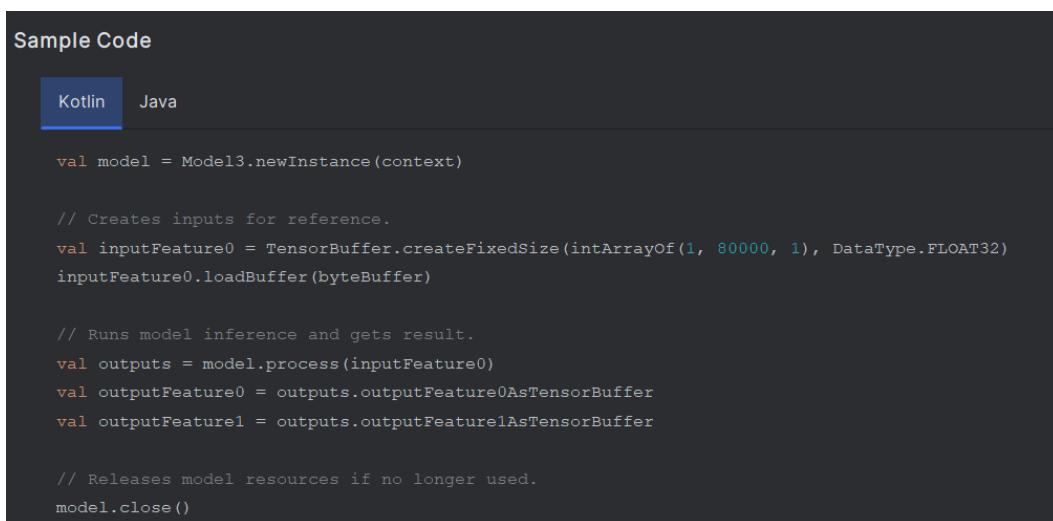
Cách thực hiện: Từ menu chọn File → New → Other → TensorFlow Lite Model, sau đó đưa file .tflite vào vị trí Model location trên giao diện cấu hình.



Hình 3.15. Di chuyển mô hình định dạng .tflite vào Android Studio

Bước 3: Tạo luồng dữ liệu âm thanh từ microphone của điện thoại.

Âm thanh thu được sẽ được chia thành các khung có độ dài phù hợp và chuyển thành định dạng tensor với kích thước đúng như yêu cầu đầu vào của mô hình .tflite, như minh họa ở Hình 3.16..



```
Sample Code

Kotlin Java

val model = Model3.newInstance(context)

// Creates inputs for reference.
val inputFeature0 = TensorBuffer.createFixedSize(intArrayOf(1, 8000, 1), DataType.FLOAT32)
inputFeature0.loadBuffer(byteBuffer)

// Runs model inference and gets result.
val outputs = model.process(inputFeature0)
val outputFeature0 = outputs.outputFeature0AsTensorBuffer
val outputFeature1 = outputs.outputFeature1AsTensorBuffer

// Releases model resources if no longer used.
model.close()
```

Hình 3.16. Mã mẫu tạo dữ liệu đầu vào đúng với kích thước mô hình yêu cầu

Bước 4: Xây dựng thuật toán beam search decoder để giải mã đầu ra của mô hình. Các ký tự hoặc mã ID đầu ra từ mô hình sẽ được chuyển thành văn bản tiếng Việt và hiển thị trên màn hình ứng dụng Android.

3.4 Thiết kế hệ thống nhận dạng tiếng nói tiếng Việt online

3.4.1 Lý do sử dụng

Chế độ hoạt động **online** đóng vai trò quan trọng trong các ứng dụng nhận dạng tiếng nói thời gian thực, đặc biệt là trong các hệ thống hỗ trợ soạn thảo văn bản, nơi yêu cầu độ trễ thấp và phản hồi gần như tức thì. Tuy nhiên, trong quá trình fine-tuning mô hình SqueezeFormer, sinh viên nhận thấy vẫn tồn tại một số hạn chế nhất định khi triển khai ở chế độ online:

- Mô hình SqueezeFormer được tiền huấn luyện theo chế độ **batch ASR**, không được thiết kế tối ưu cho nhận dạng kiểu online. Nếu muốn huấn luyện mô hình theo hướng học cách xử lý đầu vào theo thời gian thực (streaming), cần phải điều chỉnh lại kiến trúc — điều này đồng nghĩa với việc phải huấn luyện lại mô hình từ đầu, gây tổn kém cả về thời gian lẫn tài nguyên.
- Mô hình tiền huấn luyện mà sinh viên lựa chọn không hỗ trợ tính năng **căn chỉnh thời gian (time alignment)** cho từng từ hoặc cụm từ trong chuỗi đầu ra, vốn là một yêu cầu thiết yếu trong nhiều ứng dụng thời gian thực.

Xuất phát từ các bất cập nêu trên, sinh viên đề xuất ứng dụng phương pháp **Local Agreement** nhằm cải thiện khả năng xử lý đầu ra trong chế độ online, đồng thời nâng cao độ chính xác và độ tin cậy của hệ thống khi áp dụng trong các tình huống thực tế.

3.4.2 Thuật toán Local Agreement

3.4.2.1 Phương pháp Local Agreement

Ý tưởng của **phương pháp Local Agreement** được minh họa trong Hình 3.17.. Trong mô hình này, một *audio buffer* được sử dụng để lưu trữ liên tục tín hiệu âm thanh đầu vào. Bộ đệm sẽ được cập nhật định kỳ sau mỗi khoảng thời gian t giây, trong đó t đại diện cho độ dài của một phân đoạn âm thanh. Sau mỗi lần cập nhật, toàn bộ nội dung trong bộ đệm sẽ được đưa vào mô hình nhận dạng tiếng nói để suy diễn văn bản tương ứng [28].



Hình 3.17. Phương pháp xác nhận từ giữa nhiều phân đoạn

Sinh viên triển khai thuật toán bằng cách so sánh các kết quả suy diễn ở các lần lặp liên tiếp nhằm xác định chuỗi văn bản chung dài nhất giữa các lần dự đoán. Chuỗi này được xem như phần “đầu ra xác nhận” — tức là phần ổn định đã được xác minh từ nhiều phân đoạn khác nhau và do đó sẽ được hiển thị cho người dùng. Các phần văn bản đã xác nhận sẽ được cố định, không bị thay đổi trong các lần xử lý tiếp theo. Điều này giúp giảm thiểu hiện tượng thay đổi liên tục hoặc “nhấp nháy” trong kết quả đầu ra — một vấn đề phổ biến trong các hệ thống ASR bán thời gian thực.

Trong trường hợp không còn chuỗi chung nào có thể xác nhận giữa các lần dự đoán liên tiếp, toàn bộ phần văn bản còn lại sẽ được in ra để đảm bảo không mất dữ liệu. Sau đó, quá trình dự đoán sẽ tiếp tục với chu trình mới, bắt đầu từ phân đoạn âm thanh tiếp theo.

Trong hệ thống được đề xuất, sinh viên lựa chọn giá trị $t = 1.75$ giây nhằm đảm bảo đủ độ dài ngữ cảnh để tích hợp mô hình ngôn ngữ, từ đó cải thiện độ chính xác nhận dạng. Quá trình suy diễn một đoạn âm thanh chỉ mất khoảng 0.5 giây, cho phép sinh viên triển khai một kiến trúc đa luồng trong Python — trong đó một luồng thực hiện thu âm liên tục, và một luồng còn lại chịu trách nhiệm xử lý và suy diễn văn bản.

Với cách tiếp cận này, hệ thống có thể mô phỏng hiệu quả hoạt động của một hệ thống nhận dạng tiếng nói trực tuyến (online ASR), đồng thời vẫn tận dụng được sức

mạnh của các mô hình âm học tiền huấn luyện mạnh mẽ, vốn thường được thiết kế cho các tác vụ offline.

3.5 Kết luận chương

Chương này đã trình bày chi tiết quá trình thiết kế một hệ thống nhận dạng tiếng nói hoàn chỉnh, từ việc khảo sát và lựa chọn thuật toán huấn luyện phù hợp đến việc xây dựng các thành phần cốt lõi trong pipeline nhận dạng. Việc lựa chọn Squeezeformer kết hợp với hàm mất mát CTC và bộ giải mã tích hợp mô hình ngôn ngữ KenLM giúp tối ưu hóa độ chính xác nhận dạng. Bên cạnh đó, chương cũng đề cập đến việc triển khai hệ thống trên nền tảng Android và mở rộng sang mô hình hoạt động chế độ online ASR để phục vụ các yêu cầu thời gian thực. Những nội dung này sẽ là nền tảng quan trọng cho quá trình huấn luyện, đánh giá kết quả và chất lượng hệ thống trong chương 4.

CHƯƠNG 4. THỬ NGHIỆM VÀ ĐÁNH GIÁ KẾT QUẢ

Trong Chương 3, sinh viên đã trình bày chi tiết quá trình thiết kế toàn bộ hệ thống, bao gồm: phần mềm nhận dạng tiếng nói tiếng Việt liên tục, ứng dụng hiển thị văn bản tiếng Việt trên nền tảng Android, và hệ thống nhận dạng tiếng nói tiếng Việt hoạt động theo chế độ trực tuyến (online ASR). Chương này sẽ tập trung vào việc thử nghiệm và đánh giá hiệu quả của hệ thống đã được xây dựng. Trước hết, môi trường thực nghiệm và các công cụ hỗ trợ sẽ được giới thiệu nhằm đảm bảo tính khả lặp (reproducibility) và độ tin cậy của các kết quả đánh giá. Tiếp theo, các chỉ số đánh giá chính được sử dụng trong nghiên cứu bao gồm: Word Error Rate (WER), Real-Time Factor (RTF), Latency (độ trễ).

4.1 Cơ sở dữ liệu

Các tập dữ liệu được sinh viên sử dụng trong quá trình huấn luyện và đánh giá bao gồm:

- Các tập dữ liệu công khai: VLSP 2020, FPT, VIVOS
- Tập dữ liệu tự thu

4.1.1 Vietnamese Language and Speech Processing (VLSP) 2020 [3]

Nhằm thúc đẩy nghiên cứu và phát triển các hệ thống xử lý ngôn ngữ và tiếng nói tại Việt Nam, Viện Nghiên cứu Dữ liệu lớn VinBigdata đã đóng góp 100 giờ dữ liệu tiếng nói tiếng Việt cho thử thách ASR (Tự động nhận dạng tiếng nói) thuộc khuôn khổ hội thảo VLSP 2020.

Bộ dữ liệu này được thu thập từ các nguồn mở và được phiên âm thủ công với độ chính xác lên đến 96%. Đây là tập dữ liệu huấn luyện chính thức cho các đội tham gia thử thách ASR 2020, hỗ trợ phát triển mô hình nhận dạng tiếng nói tiếng Việt. Kết quả mô hình sẽ được đánh giá theo chỉ số Word Error Rate (WER) – thước đo phổ biến trong đánh giá hiệu quả của hệ thống ASR và dịch máy.

Cấu trúc tập dữ liệu

Cấu trúc tập dữ liệu VLSP 2020 gồm 3 folder chính được phân tích như bảng 4.1.:

Bảng 4.1. Tổng quan về các thư mục dữ liệu gốc VLSP 2020

Tên thư mục	Mô tả nội dung	Số lượng tệp	Mức độ nhiễu	Dung lượng
Database	Gồm các đoạn âm thanh từ phỏng sự, bản tin và các thể loại khác được trộn lẫn với nhau.	37.000 tệp audio 37.000 tệp văn bản	Có nhiễu nền từ nhỏ đến lớn.	7,7 GB
Speaker	Gồm 760 thư mục, mỗi thư mục tương ứng với một người nói.	14.000 tệp audio 14.000 tệp văn bản 493 nữ, 267 nam	Có nhiễu nền nhẹ, tiếng nói rõ ràng.	2,2 GB
Spkyut	Chứa các câu nói ngẫu nhiên, không theo chủ đề cố định.	5.000 tệp audio 5.000 tệp văn bản	Mức nhiễu cao	1,07 GB

4.1.2 FPT Open Speech Dataset

Bộ dữ liệu này bao gồm 25,921 bài phát biểu tiếng Việt đã được ghi âm (kèm theo bản ghi chép và các mốc thời gian bắt đầu và kết thúc của từng bài phát biểu) được tổng hợp thủ công từ 3 tập dữ liệu con (tổng cộng khoảng 30 giờ) do Tập đoàn FPT công bố công khai vào năm 2018.

Các bài phát biểu được lưu trữ ở định dạng *.mp3 trong khi file bản ghi chép là định dạng *.txt với mã hóa UTF-8.

4.1.3 VIVOS

Tập dữ liệu VIVOS được giới thiệu trong nghiên cứu [29]. Tập dữ liệu được thu âm gồm hơn 50 người nói. Trong **tập huấn luyện**, **46 người** nói (gồm **22 nam** và **24 nữ**) đã tham gia thu âm với tổng thời lượng 15 giờ, tương ứng với 11660 câu nói. Trong khi đó, **tập kiểm thử** được thực hiện với một nhóm người nói khác **gồm 19 người (12 nam và 7 nữ)**, thu âm tổng cộng 760 câu nói với thời lượng khoảng 50 phút. Các buổi ghi âm được thực hiện trong môi trường yên tĩnh với thiết bị ghi âm chất lượng cao. Mặc dù kích thước tập dữ liệu còn khá nhỏ so với tiêu chuẩn hiện nay của các hệ thống nhận dạng tiếng nói tự động (ASR), nhưng các tập dữ liệu có quy mô tương tự vẫn được sử dụng trong các nghiên cứu đánh giá hệ thống nhận dạng tiếng nói tiếng việt như trong [30, 31].

4.1.4 Bộ dữ liệu tự thu (CStt - Cơ sở tự thu)

4.1.4.1 Xây dựng kịch bản

Nhằm đánh giá chất lượng hệ thống nhận dạng tiếng nói một cách khách quan, sinh viên đã tự xây dựng một tập dữ liệu tiếng Việt có tên là **CStt**. Bộ dữ liệu này được thu thập từ 33 người tình nguyện (gồm 21 nam và 12 nữ), với mục tiêu phản ánh sự đa dạng về giọng nói, tốc độ phát âm

Mỗi người tham gia được yêu cầu đọc một đoạn văn bản ngẫu nhiên (có thể là từ sách, báo, truyện,...), với thời lượng từ 3 đến 6 phút. Nội dung được lựa chọn linh hoạt, không giới hạn chủ đề nhằm đảm bảo tính tự nhiên và phong phú của ngôn ngữ. Quá trình thu âm được thực hiện trong môi trường yên tĩnh, ít nhiễu, sử dụng micro tích hợp sẵn của điện thoại thông minh. Tốc độ nói của người tham gia không bị ràng buộc, bao gồm cả chậm, trung bình và nhanh, để phản ánh các tình huống nói khác nhau trong thực tế.

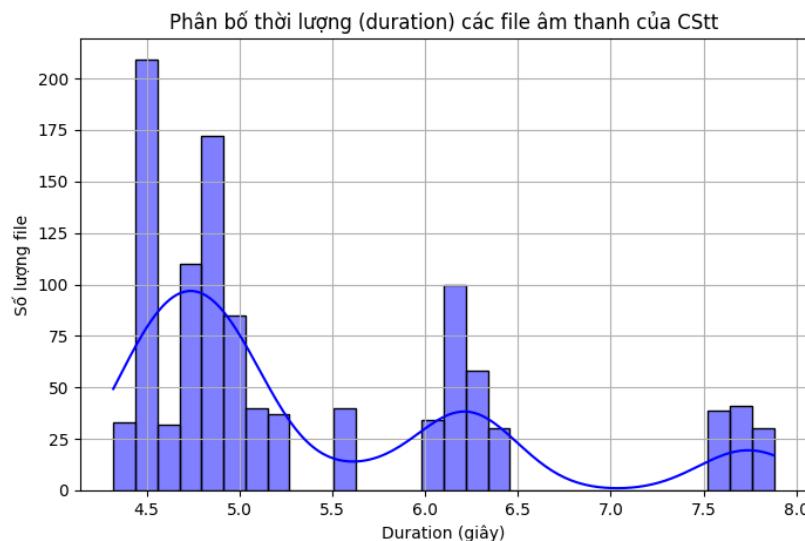
Sau khi thu âm, mỗi đoạn ghi âm được chia nhỏ thành khoảng 30 đến 40 tệp âm thanh định dạng WAV, độ sâu bit 16-bit PCM. Mỗi người có một thư mục riêng chứa các tệp âm thanh đã tách để thuận tiện cho việc xử lý và quản lý. Tổng thời lượng của toàn bộ tập dữ liệu đạt khoảng **98 phút**, được phân bổ đều giữa các đối tượng thu âm.

Về phần văn bản, sinh viên tiến hành đổi chiểu thủ công và đánh máy lại toàn bộ nội dung tương ứng với từng tệp âm thanh, đảm bảo tính chính xác tuyệt đối giữa âm thanh và phiên bản văn bản.

Tóm tắt đặc điểm của tập dữ liệu CSStt được trình bày trong bảng dưới đây:

- **Số lượng người tham gia:** 33 người (21 nam, 12 nữ)
- **Tổng thời lượng dữ liệu:** 98 phút
- **Thời lượng trung bình mỗi người:** 3–5 phút, sau đó được chia thành khoảng 30–40 file âm thanh có độ dài từ 4-8s, tổng có 1190 files
- **Thiết bị và môi trường thu âm:** Micro điện thoại thông minh, môi trường thu âm yên tĩnh, ít tạp âm
- **Nội dung thu âm:** Các đoạn văn bản ngẫu nhiên từ sách, báo, truyện trên Internet
- **Định dạng tệp âm thanh:** WAV, PCM 16-bit
- **Tốc độ nói:** Đa dạng — từ chậm đến nhanh
- **Xử lý văn bản:** Sinh viên tự đánh máy nội dung tương ứng từng file âm thanh (text thủ công)

Hình 4.1. minh họa phân bố thời lượng của tập dữ liệu tự thu:



Hình 4.1. Phân bố thời lượng tập dữ liệu tự thu

4.2 Tiền xử lý cơ sở dữ liệu

4.2.1 Các phương pháp tiền xử lý

Trong quá trình xây dựng hệ thống nhận dạng tiếng nói, chất lượng và độ nhất quán của dữ liệu đều vào đóng vai trò quyết định đến hiệu quả huấn luyện mô hình. Nhằm đảm bảo dữ liệu huấn luyện có chất lượng cao, giảm thiểu nhiễu và phù hợp với

năng lực tính toán của hệ thống, sinh viên đã thực hiện một số bước tiền xử lý quan trọng như sau:

Loại bỏ thư mục *Spkyut* của VLSP 2020:

Thư mục *Spkyut* chứa nhiều tệp âm thanh có mức độ nhiễu cao, bao gồm các đoạn hát karaoke, tiếng ồn từ môi trường như đường phố, xe cộ,... Trong nhiều trường hợp, tiếng ồn còn lấn át cả giọng nói chính, làm giảm chất lượng tín hiệu đầu vào. Ngoài ra, số lượng tệp trong thư mục này không nhiều, nên đóng góp của chúng vào quá trình huấn luyện là không đáng kể. Nhằm hạn chế khả năng mô hình bị ảnh hưởng tiêu cực bởi dữ liệu nhiễu và đảm bảo quá trình huấn luyện diễn ra hiệu quả, sinh viên quyết định loại bỏ toàn bộ các tệp trong thư mục *Spkyut*. Thay vào đó, quá trình chia tập *train*, *validation* và *test* được thực hiện trên hai thư mục còn lại là *Speaker* và *Database* để phục vụ cho quá trình *fine-tuning* và đánh giá hệ thống.

Sử dụng những file âm thanh có độ dài dưới 8 giây:

Do giới hạn về mặt phần cứng, sinh viên không thể sử dụng toàn bộ dữ liệu gốc của tập VLSP 2020 (bao gồm các tệp từ thư mục *Speaker* và *Database*), cũng như các tập dữ liệu khác như FPT, VIVOS cho quá trình huấn luyện mô hình. Cụ thể, toàn bộ các tệp âm thanh cần được xử lý bằng kỹ thuật đệm (*padding*) để đảm bảo có cùng độ dài với tệp có thời lượng dài nhất trong tập dữ liệu. Điều này dẫn đến việc tiêu tốn tài nguyên bộ nhớ lớn, vượt quá khả năng xử lý của hệ thống hiện tại. Để khắc phục, sinh viên quyết định chỉ giữ lại các tệp âm thanh có thời lượng nhỏ hơn hoặc bằng 8 giây trước khi đưa vào huấn luyện. Mốc thời gian 8 giây được lựa chọn vì đây là khoảng thời gian tương đối phù hợp để người dùng có thể biểu đạt một câu nói hoàn chỉnh, đồng thời vẫn đảm bảo tính hiệu quả trong quá trình xử lý và huấn luyện mô hình.

Căn chỉnh lại văn bản trong tập dữ liệu:

- Thay thế các từ phiên âm tiếng Anh bằng từ tương ứng trong tiếng Việt nhằm đảm bảo tính nhất quán và phù hợp với ngữ cảnh sử dụng.
- Loại bỏ toàn bộ dấu câu và các ký tự đặc biệt để đơn giản hóa đầu vào cho mô hình, đồng thời giảm nhiễu trong quá trình huấn luyện.

4.3 Phân chia dữ liệu sử dụng trong huấn luyện và đánh giá kết quả

Nhằm đảm bảo tính khách quan và khả năng tổng quát của mô hình nhận dạng tiếng nói, dữ liệu được phân chia thành ba tập riêng biệt: **tập huấn luyện (train)**, **tập xác thực (validation)**, và **tập kiểm tra (test)**. Việc phân chia này được thực hiện một cách cẩn trọng để đảm bảo rằng không có sự trùng lặp người nói hoặc tệp âm thanh giữa các tập, giúp mô hình học được các đặc trưng âm học và ngôn ngữ một cách độc lập, đồng thời đánh giá chính xác hiệu năng trên những dữ liệu chưa từng được quan sát trong quá trình huấn luyện.

Cụ thể, đối với từng tập dữ liệu sử dụng (bao gồm VLSP 2020, VIVOS, FPT và bộ dữ liệu tự thu CSST), sinh viên tiến hành tách riêng danh sách người nói, sau đó chia thành các nhóm độc lập tương ứng với ba tập train/valid/test. Việc giữ nguyên tính độc lập theo người nói là yếu tố quan trọng để mô hình không bị học “ghi nhớ” giọng nói cụ thể, từ đó đánh giá đúng khả năng khái quát hóa của hệ thống.

Ngoài ra, số lượng tệp trong mỗi tập cũng được phân bố hợp lý để đảm bảo sự cân bằng và ổn định trong quá trình huấn luyện và đánh giá.

4.3.1 Sử dụng tập dữ liệu VLSP 2020

Dữ liệu sử dụng trong tập VLSP 2020 sau khi tiền xử lý được chia như sau:

Bảng 4.2. Phân chia tập dữ liệu VLSP 2020

Tập dữ liệu	Loại	Số lượng (files)	Cách chia	Thời lượng
Speaker(VLSP 2020)	Train	11064	600 người	12h51m
	Valid	1412	80 người	1h38m
	Test	1387	80 người	1h35m
Database(VLSP 2020)	Train	15621	Tỷ lệ 8:1:1	20h47m
	Valid	2000		2h23m
	Test	2000		2h21m

4.3.2 Sử dụng tập dữ liệu VIVOS

Do tập dữ liệu VIVOS đã chia folder huấn luyện và thực nghiệm khá rõ ràng, do đó VIVOS được chia train, test, valid độc lập như sau. Mỗi người nói trong tập huấn luyện có khoảng 200-250 files, trong khi đó, mỗi người nói trong tập thực nghiệm có khoảng 40 files.

Bảng 4.3. Tổng quan về tập dữ liệu VIVOS

Tập dữ liệu	Loại	Số lượng (files)	Cách chia	Thời lượng
VIVOS	Train	9875	40 người	10h
	Valid	1698	6 người	1h47m
	Test	757	19 người	45m

4.3.3 Sử dụng tập dữ liệu FPT

Dữ liệu trong tập gốc FPT cũng đã loại bỏ đi những file âm thanh có độ dài lớn hơn 8 giây, được sử dụng như bảng 4.4..

Bảng 4.4. Phân chia tập dữ liệu FPT

Tập dữ liệu	Loại	Số lượng (files)	Cách chia	Thời lượng
FPT	Train	6317	Tỷ lệ 8:1:1	7h21m
	Valid	1000		53m
	Test	1000		52m

4.4 Một số thông số đánh giá chất lượng hệ thống

Trong lĩnh vực nhận dạng tiếng nói tự động (Automatic Speech Recognition – ASR), việc đánh giá chất lượng của hệ thống đóng vai trò quan trọng nhằm phản ánh mức độ hiệu quả và khả năng áp dụng vào thực tiễn. Ba chỉ số thường được sử dụng để đo lường hiệu suất của một hệ thống ASR bao gồm: **Word Error Rate (WER)**, **Real-Time Factor (RTF)** và **Latency (Độ trễ)**. Các chỉ số này không chỉ phản ánh độ chính xác mà còn thể hiện khả năng phản hồi và xử lý theo thời gian thực của hệ thống, đặc biệt trong các ứng dụng triển khai trên thiết bị di động hoặc môi trường có yêu cầu tương tác tức thời.

4.4.1 Word Error Rate (WER)

Word Error Rate (WER) là chỉ số phổ biến và được sử dụng rộng rãi để đánh giá độ chính xác của hệ thống ASR. WER đo lường tỷ lệ lỗi khi chuyển đổi tín hiệu âm thanh thành văn bản, dựa trên so sánh giữa đầu ra của hệ thống và bản phiên âm đúng (reference transcript).

Công thức tính:

$$\text{WER} = \frac{S + I + D}{N} \quad (4.1)$$

Trong đó:

- S (Substitutions): Số từ bị thay thế;
- I (Insertions): Số từ bị chèn thêm không đúng;
- D (Deletions): Số từ bị thiếu hoặc bị xóa;
- N : Tổng số từ trong bản phiên âm đúng.

Ý nghĩa: Giá trị WER càng thấp cho thấy độ chính xác hệ thống nhận dạng tiếng nói càng cao và ngược lại. Với tiếng Việt – một ngôn ngữ tách từ rõ ràng (word-level language), chỉ số WER đặc biệt phù hợp để đánh giá vì mỗi từ thường mang trọn nghĩa và bao gồm nhiều ký tự. WER không chỉ phản ánh hiệu quả tổng thể mà còn giúp phát hiện các lỗi thường gặp trong quá trình nhận dạng.

4.4.2 Real-Time Factor (RTF)

Real-Time Factor (RTF) là chỉ số dùng để đánh giá khả năng xử lý của hệ thống ASR so với thời gian thực. Đây là yếu tố quan trọng trong việc triển khai các

hệ thống nhận dạng tiếng nói trên các thiết bị đầu cuối như smartphone, nơi yêu cầu tốc độ xử lý nhanh chóng và ổn định.

Công thức tính:

$$RTF = \frac{T_{\text{processing}}}{T_{\text{input}}} \quad (4.2)$$

Trong đó:

- $T_{\text{processing}}$: Thời gian hệ thống cần để xử lý đoạn âm thanh;
- T_{input} : Thời lượng thực tế của đoạn âm thanh đầu vào.

Ý nghĩa: Hệ thống được coi là hoạt động hiệu quả trong thời gian thực nếu $RTF < 1$, tức là có thể xử lý âm thanh nhanh hơn thời lượng thực của chính đoạn đó. RTF cao sẽ gây ra độ trễ, ảnh hưởng đến trải nghiệm người dùng và làm giảm tính ứng dụng của hệ thống trong các tình huống thực tế như điều khiển thiết bị bằng giọng nói hoặc giao tiếp trực tuyến.

4.4.3 Latency (Độ trễ)

Latency là một chỉ số đánh giá khả năng phản hồi tức thì của hệ thống ASR. Đây là khoảng thời gian trễ từ khi người dùng bắt đầu phát âm cho đến khi hệ thống đưa ra phản hồi đầu tiên (dù là tạm thời hay hoàn chỉnh).

Công thức tính:

$$\text{Latency} = t_{\text{output}} - t_{\text{start}} \quad (4.3)$$

Trong đó:

- t_{start} : Thời điểm bắt đầu phát ngôn;
- t_{output} : Thời điểm hệ thống trả về kết quả đầu tiên.

Ý nghĩa: Latency thấp giúp tạo cảm giác giao tiếp tự nhiên và liền mạch giữa người và máy. Đặc biệt trong các ứng dụng như trợ lý ảo, dịch trực tiếp, hoặc hệ thống nhúng, độ trễ dưới 500 mili-giây (< 500 ms) được xem là lý tưởng để đảm bảo chất lượng trải nghiệm người dùng.

4.5 Thông số huấn luyện

Tốc độ học: Trong toàn bộ quá trình huấn luyện, sinh viên sử dụng chiến lược *Linear Decay* cho tốc độ học (*learning rate*). Giá trị khởi tạo được thiết lập là **0.001** và được giảm dần tuyến tính sau mỗi epoch, cho đến khi đạt giá trị nhỏ nhất là **1e-5** sau **160 epoch**. Chiến lược này giúp mô hình học nhanh ở giai đoạn đầu và ổn định hơn trong giai đoạn sau, hạn chế tình trạng dao động hoặc quá khớp (overfitting).

```

initial_lr = 0.001
total_steps = 160 * 1279 #here

lr_schedule = tf.keras.optimizers.schedules.PolynomialDecay(
    initial_learning_rate=initial_lr,
    decay_steps=total_steps,
    end_learning_rate=0.00001,
    power=1.0 # Linear decay
)

```

Hình 4.2. Thiết lập tốc độ học

Batch size: Giá trị batch size được lựa chọn là **32**, đây cũng là kích thước lớn nhất mà phần cứng của sinh viên sử dụng có thể xử lý ổn định. Lựa chọn này giúp cân bằng giữa hiệu suất tính toán và khả năng tổng quát hoá của mô hình. Batch size đủ lớn giúp quá trình huấn luyện diễn ra nhanh hơn trên tập dữ liệu lớn, đồng thời vẫn đảm bảo mô hình học được các đặc trưng tổng thể mà không bị lệ thuộc quá nhiều vào nhiễu của từng mẫu nhỏ lẻ.

4.6 Thủ nghiệm 1: Đánh giá chất lượng của mô hình âm học

4.6.1 Mục tiêu thử nghiệm

Mục tiêu của thử nghiệm này là lựa chọn mô hình âm học phù hợp và đạt chất lượng tốt nhất trong bối cảnh giới hạn về thời gian cũng như tài nguyên tính toán. Việc đánh giá tập trung vào khả năng cân bằng giữa hiệu suất mô hình (độ chính xác nhận dạng) và tính khả thi trong quá trình huấn luyện, từ đó xác định cấu hình tối ưu cho quá trình fine-tuning trên tập dữ liệu thực nghiệm.

4.6.2 Cách tiến hành

Trong các thử nghiệm, mô hình tiền huấn luyện được sử dụng là **SqueezeFormer-XS** với khoảng **9 triệu tham số** — một kích thước tương đối lớn so với điều kiện phần cứng hiện có. Để tối ưu hiệu quả huấn luyện trong giới hạn tài nguyên, sinh viên áp dụng chiến lược **fine-tuning có kiểm soát**, với cách thiết lập như sau:

- **Toàn bộ phần decoder** được mở khóa (*unfreeze*) hoàn toàn, cho phép cập nhật tham số trong suốt quá trình huấn luyện.
- **Phần encoder** được mở dần theo từng lớp, bắt đầu từ các lớp cao nhất. Cách làm này nhằm khảo sát ảnh hưởng của số lượng tham số được cập nhật đến chất lượng mô hình, từ đó lựa chọn cấu hình fine-tuning tối ưu.

Chiến lược trên cho phép đánh giá mối quan hệ giữa số lượng tham số được huấn luyện và hiệu suất tổng thể của mô hình, nhằm tìm ra điểm cân bằng hợp lý giữa độ chính xác và khả năng xử lý của phần cứng. Việc huấn luyện và đánh giá được thực

hiện trên tập dữ liệu được mô tả chi tiết trong Bảng 4.5. Trong đó, hai phương án phân loại đầu ra được thử nghiệm:

1. Mô hình âm học với **128 lớp đầu ra**, bao gồm cả các ký hiệu đặc biệt hoặc từ phụ trợ.
 2. Mô hình âm học với **93 lớp đầu ra**, chỉ gồm các ký tự thuộc bảng chữ cái tiếng Việt.

Mục tiêu của so sánh này là đánh giá xem việc sử dụng tập nhãn phong phú hơn (128 lớp) có mang lại cải thiện hiệu suất đáng kể so với tập nhãn rút gọn (93 lớp) hay không, từ đó định hướng tối ưu cho mô hình nhận dạng tiếng nói tiếng Việt. Dữ liệu sử dụng trong thử nghiệm chỉ bao gồm hai tập: **VIVOS** và **Speaker**, được phân chia như trình bày trong Bảng 4.5..

Bảng 4.5. Dữ liệu Speaker (VLSP 2020) và VIVOS được sử dụng trong thử nghiệm 1

Tập dữ liệu	Loại	Số lượng (files)	Cách chia	Thời lượng
Speaker(VLSP 2020)	Train	11064	600 người	12h51m
	Valid	1412	80 người	1h38m
	Test	1387	80 người	1h35m
VIVOS	Train	9875	40 người	10h
	Valid	1698	6 người	1h47m
	Test	757	19 người	45m

4.6.3 Phương án 1: Sử dụng 128 lớp đầu ra cho mô hình âm học

Số lượng class: 128 bao gồm cả các từ phụ bên cạnh các ký tự duy nhất dùng trong tiếng Việt với chiến thuật mã hóa BPE.

“<unk>”, “<s>”, “</s>”, “_t”, “ng”, “_c”, “_d”, “nh”, “_v”, “_th”, “_l”, “_h”, “_m”, “_b”, “_ch”, “_tr”, “_n”, “_k”, “_nh”, “_s”, “_ng”, “_g”, “_kh”, “_p”, “_d”, “_ph”, “_la”, “õng”, “ie”, “_ca”, “_q”, “_qu”, “_gi”, “_và”, “êñ”, “_r”, “uõ”, “_x”, “ói”, “_có”, “_cù”, “_”, “n”, “h”, “t”, “i”, “c”, “g”, “a”, “m”, “u”, “đ”, “à”, “o”, “u”, “v”, “I”, “r”, “á”, “y”, “b”, “p”, “ô”, “k”, “s”, “ó”, “é”, “ä”, “ö”, “ð”, “ë”, “â”, “ê”, “í”, “d”, “â”, “ô”, “ó”, “â”, “o”, “è”, “q”, “ü”, “ë”, “ä”, “í”, “ö”, “î”, “â”, “e”, “x”, “â”, “ü”, “ú”, “ñ”, “ö”, “û”, “â”, “ð”, “ô”, “ü”, “â”, “ñ”, “ð”, “ò”, “û”, “ù”, “â”, “ý”, “í”, “ë”, “ô”, “û”, “â”, “é”, “í”, “ë”, “â”, “â”, “ð”, “ë”, “â”, “e”, “ý”, “ë”, “ý”, “â”, “ë”, “ô”, “ð”, “â”

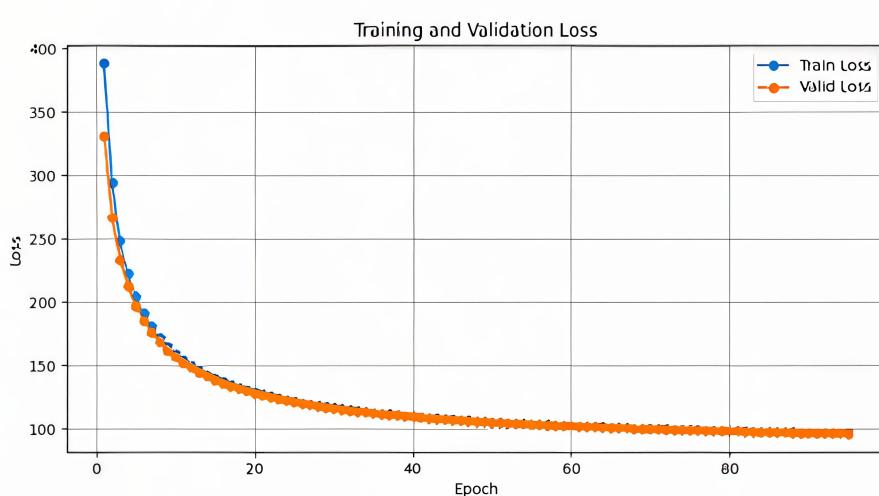
4.6.3.1 Trường hợp 1: Chỉ unfreeze khối Decoder

Vì đây là thử nghiệm đầu tiên trên tập dữ liệu tiếng Việt, sinh viên lựa chọn phương án chỉ mở khóa (unfreeze) khỏi **decoder** của mô hình. Với cấu hình này, tổng số tham số được huấn luyện là khoảng **18.000** trên tổng số **9 triệu** tham số của toàn bộ mô hình.

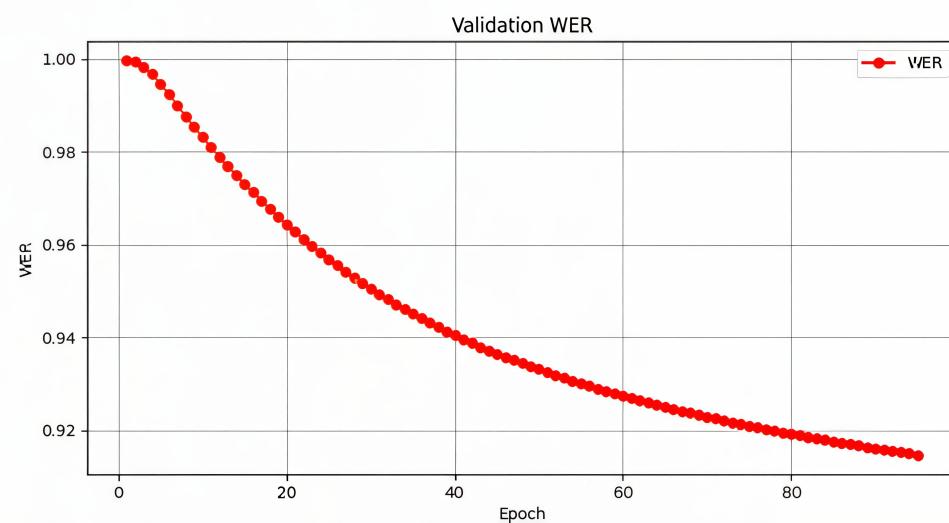
Kết quả:

Biểu đồ diễn biến hàm mất mát (loss) và chỉ số WER trên tập validation được trình bày trong Hình 4.3. và Hình 4.4.

Nhận xét:



Hình 4.3. Biểu đồ hàm loss sau 90 epoch khi chỉ unfreeze khối decoder



Hình 4.4. Biểu đồ WER theo epoch trên tập Valid (Phương án 1 - Trường hợp 1).

- Mặc dù mô hình được huấn luyện trong 90 epoch, chỉ số WER trên tập validation vẫn ở mức rất cao (**91%**), cho thấy hiệu suất nhận dạng rất kém.
- Nguyên nhân chủ yếu có thể đến từ việc số lượng tham số được cập nhật trong quá trình huấn luyện là quá nhỏ (chỉ 18k), khiến mô hình gần như giữ nguyên toàn bộ trọng số đã học trên tập tiếng Anh. Điều này dẫn đến việc mô hình không thích nghi tốt với đặc thù ngôn ngữ tiếng Việt.

Định hướng cải thiện:

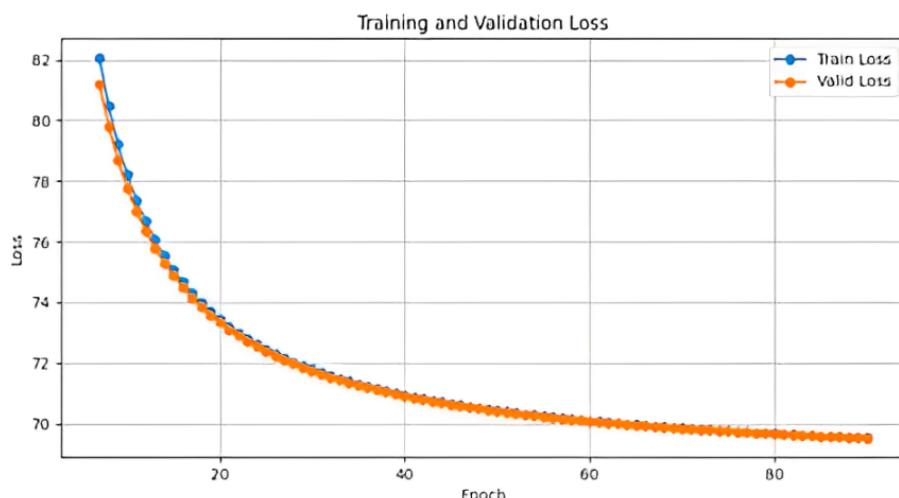
Để nâng cao hiệu quả mô hình, cần tăng số lượng tham số được học bằng cách mở thêm các khối trong kiến trúc mô hình, đặc biệt là các lớp trên cùng trong phần **encoder**. Điều này sẽ giúp mô hình thích nghi tốt hơn với dữ liệu tiếng Việt mà vẫn duy trì được tính ổn định trong quá trình fine-tuning.

4.6.3.2 Trường hợp 2: Unfreeze khối SqueezeFormer Block 15, Dense Encoder và Decoder

Trong trường hợp này, sinh viên giữ nguyên dữ liệu huấn luyện và validation như trường hợp 1, mở khóa thêm khối **SqueezeFormer Block 15** trong phần **encoder**, cùng với các lớp **dense** của cả **encoder** và **decoder**. Việc mở rộng phạm vi fine-tuning này giúp tăng tổng số tham số có thể huấn luyện lên khoảng **560.000 tham số**, so với chỉ 18.000 tham số ở Trường hợp 1.

Kết quả:

Biểu đồ thể hiện diễn biến của hàm mất mát (loss) và chỉ số WER trên tập validation sau 90 epoch được trình bày tại Hình 4.5. và Hình 4.6.:



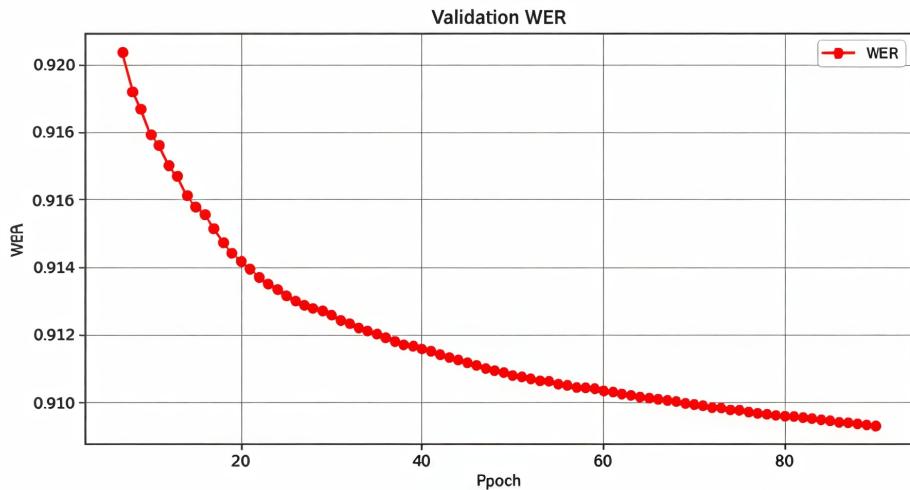
Hình 4.5. Biểu đồ hàm loss sau 90 epoch

Nhận xét:

- Giá trị của hàm loss giảm nhanh và ổn định hơn so với Trường hợp 1, cho thấy mô hình đã bắt đầu học được đặc trưng từ dữ liệu tiếng Việt.
- Chỉ số WER trên tập validation cũng có xu hướng giảm, tuy nhiên mức cải thiện vẫn chưa thực sự đáng kể để đạt yêu cầu sử dụng trong thực tế, vẫn đạt khoảng **90%**

Định hướng cải thiện:

Để tiếp tục cải thiện hiệu suất mô hình, sinh viên đề xuất mở thêm các khối **SqueezeFormer Block** tiếp theo trong phần **encoder**, nhằm tăng số lượng tham số



Hình 4.6. Biểu đồ WER theo số epoch trên tập validation (Phương án 1 - Trường hợp 2)

được cập nhật trong quá trình huấn luyện. Việc mở rộng này cần được thực hiện từng bước, có kiểm soát, nhằm đảm bảo mô hình vẫn giữ được sự ổn định trong fine-tuning và phù hợp với giới hạn phần cứng hiện có.

4.6.3.3 Trường hợp 3: Unfreeze khối SqueezeFormer Block 14–15 và Decoder

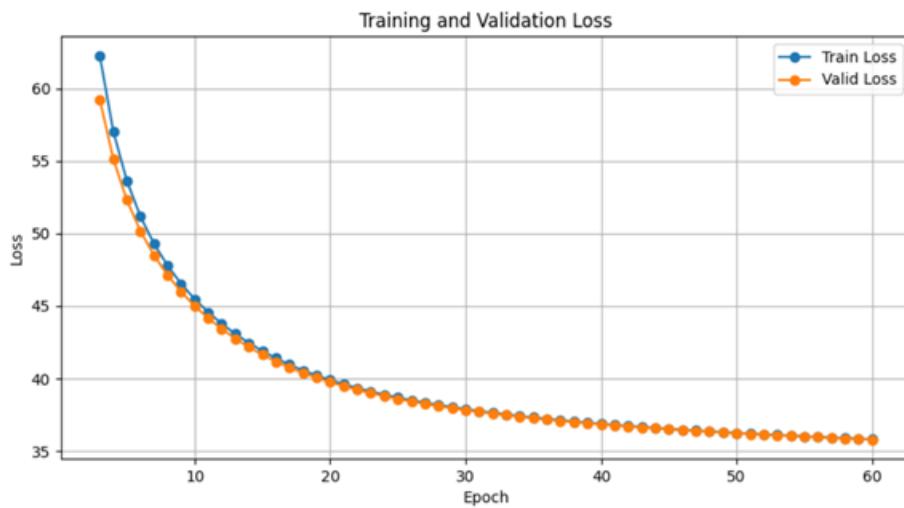
Trong thiết lập thứ ba, sinh viên vẫn giữ nguyên dữ liệu huấn luyện và tiếp tục mở rộng phạm vi fine-tuning bằng cách unfreeze thêm khối **SqueezeFormer Block 14** trong phần **encoder**, cùng với các lớp **dense** và toàn bộ **decoder**. Tổng số tham số được phép cập nhật trong quá trình huấn luyện với cấu hình này vào khoảng **1.100.000 tham số** trên tổng số 9 triệu tham số của mô hình.

Cấu hình unfreeze:

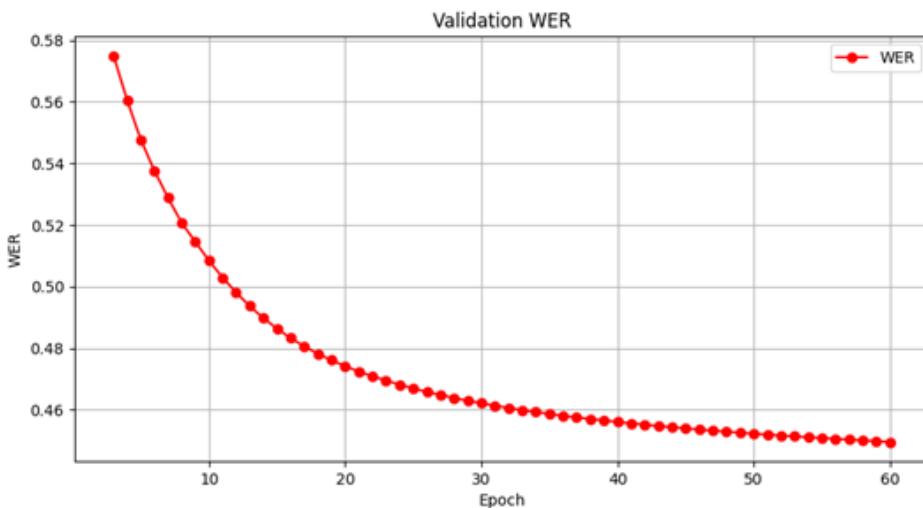
- **Encoder:** SqueezeFormer Block 14, 15 và các lớp dense;
- **Decoder:** Toàn bộ các lớp.

Kết quả:

Biểu đồ diễn biến hàm mất mát (loss) và chỉ số WER trên tập validation được trình bày tại Hình 4.7. và Hình 4.8. dưới đây:



Hình 4.7. Biểu đồ hàm loss sau 60 epoch



Hình 4.8. Biểu đồ WER theo số epoch trên tập validation (Phương án 1 - Trường hợp 3)

Bảng 4.6. Kết quả WER (%) trên Speaker + VIVOS (Thử nghiệm 1 - Phương án 1 - Trường hợp 3)

Tập dữ liệu	Loại	WER no LM (%)
Speaker + VIVOS	Valid Test	44 48

Nhận xét:

- Mô hình đạt WER tốt nhất khoảng **44%** trên tập validation và **48%** trên tập test, chỉ sau 60 epoch huấn luyện — cho thấy sự cải thiện rõ rệt so với các thiết lập trước.

- Việc tăng số lượng tham số được học lên đến hơn 1.1 triệu đã giúp mô hình thích nghi tốt hơn với đặc trưng của ngôn ngữ tiếng Việt, đồng thời vẫn đảm bảo ổn định trong quá trình fine-tuning.

Định hướng cải thiện:

Dựa trên kết quả khả quan đạt được ở trường hợp này, sinh viên định hướng tiếp tục mở rộng phạm vi huấn luyện bằng cách unfreeze thêm các khối trong encoder. Với giới hạn phần cứng hiện tại, tổng số tham số có thể được học tối đa ước tính vào khoảng **1.600.000 tham số**. Đây sẽ là cấu hình thử nghiệm tiếp theo trong quá trình tối ưu hóa mô hình âm học cho hệ thống ASR tiếng Việt.

4.6.3.4 Trường hợp 4: Unfreeze khối SqueezeFormer Block 13–15 và Decoder

Trong cấu hình cuối cùng của loạt thử nghiệm này, sinh viên tiếp tục mở rộng phạm vi fine-tuning bằng cách unfreeze thêm khối **SqueezeFormer Block 13** trong phần **encoder**, bên cạnh các khối 14, 15 và các lớp **dense**. Phần **decoder** vẫn được giữ nguyên toàn bộ ở trạng thái unfreeze như các thử nghiệm trước.

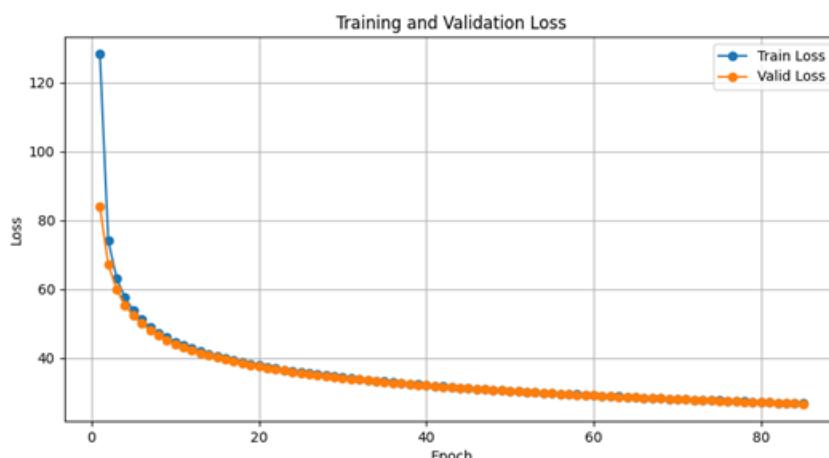
Cấu hình unfreeze:

- **Encoder:** SqueezeFormer Block 13, 14, 15 và các lớp dense;
- **Decoder:** Toàn bộ các lớp.

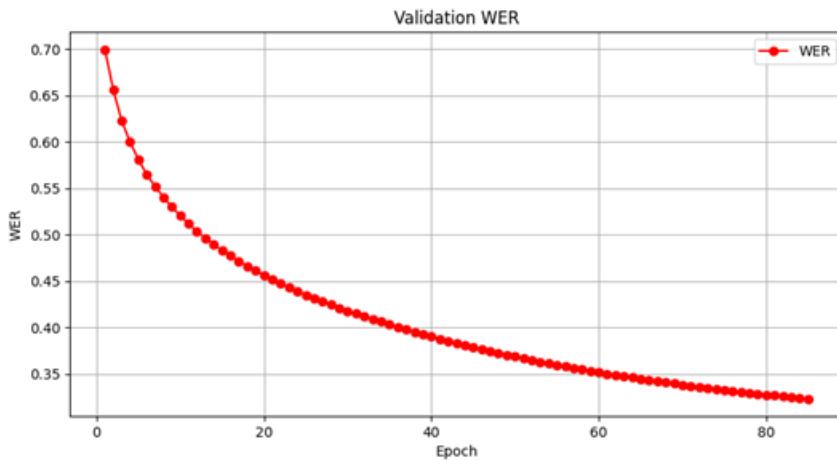
Tổng số tham số có thể học trong thiết lập này là khoảng **1.600.000**, và cũng là ngưỡng tối đa mà phần cứng hiện tại có thể xử lý ổn định với tập dữ liệu tiếng Việt có quy mô tương đối lớn.

Kết quả:

Biểu đồ thể hiện diễn biến của hàm mất mát (loss) và chỉ số WER trên tập validation được trình bày tại Hình 4.9. và Hình 4.10.:



Hình 4.9. Biểu đồ hàm loss sau 90 epoch



Hình 4.10. Biểu đồ WER theo số epoch trên tập validation (Phương án 1 - Trường hợp 4)

Bảng 4.7. Kết quả WER (%) trên Speaker + VIVOS (Thử nghiệm 1 - Phương án 1 - Trường hợp 4)

Tập dữ liệu	Loại	WER no LM (%)
Speaker + VIVOS	Valid	31
	Test	36

Nhận xét:

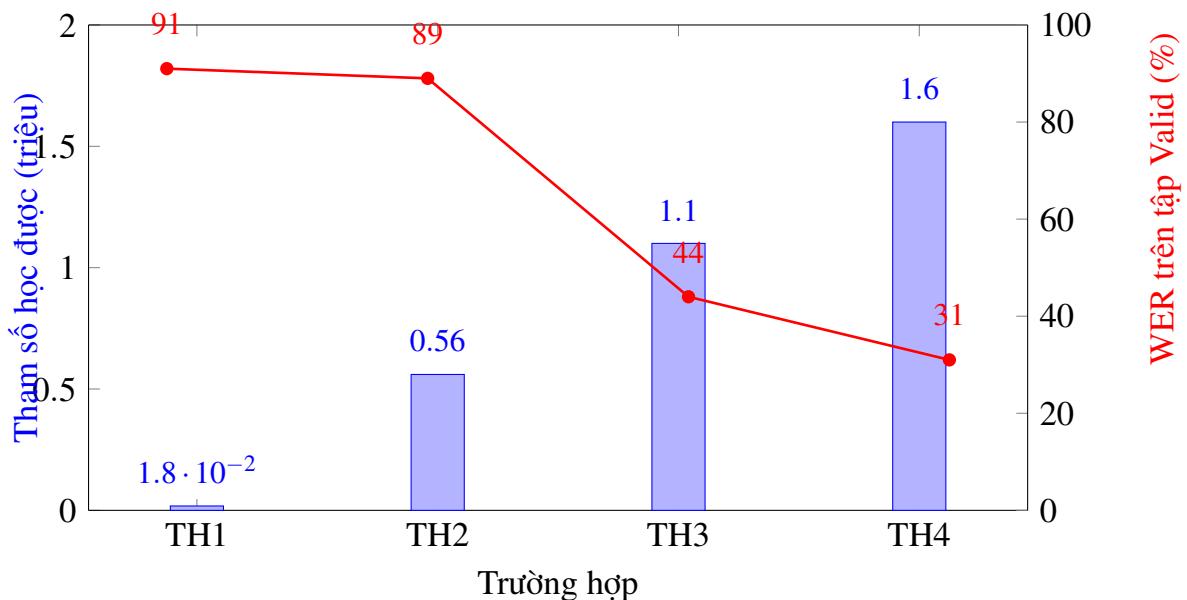
- Mô hình đạt WER tốt nhất khoảng **31%** trên tập validation và **36%** trên tập test, thể hiện sự cải thiện rõ rệt so với các thử nghiệm trước đó.
- Việc mở rộng phạm vi huấn luyện lên đến 1.6 triệu tham số đã giúp mô hình học được nhiều đặc trưng ngữ âm của tiếng Việt hơn, từ đó nâng cao hiệu quả nhận dạng.

Định hướng phát triển: Tại thời điểm này, mô hình đã đạt đến giới hạn tối đa về số lượng tham số có thể học theo điều kiện phần cứng hiện có. Do đó, hướng cải thiện tiếp theo sẽ không tập trung vào mở rộng kiến trúc, mà chuyển sang mở rộng quy mô **dữ liệu huấn luyện**.

4.6.4 Phương án 2: Sử dụng 93 lớp đầu ra cho mô hình âm học và mở khóa khối 6 thay cho khối 13 trong encoder

4.6.4.1 Dữ liệu sử dụng

Dữ liệu huấn luyện và đánh giá trong Phương án 2 giống hoàn toàn với Phương án 1, bao gồm hai tập: **Speaker** và **VIVOS**. Việc chia dữ liệu thành tập huấn luyện, kiểm thử và xác thực được thực hiện tương tự như trong Bảng 4.5..



Hình 4.11. Ảnh hưởng của số tham số học được đến WER trên tập Valid

4.6.4.2 Mục đích thử nghiệm

Thử nghiệm trong Phương án 2 được thiết kế nhằm đánh giá hiệu quả của mô hình khi có đồng thời hai thay đổi so với Phương án 1 – Trường hợp 4:

- **Giảm số lượng lớp đầu ra** từ 128 xuống còn **93 lớp**, chỉ bao gồm các ký tự tiếng Việt đơn lẻ và một số ký hiệu đặc biệt như `<s>` và `</s>`;
- **Thay đổi cấu hình fine-tuning**: thay vì unfreeze các khối 13, 14, 15 trong encoder, Phương án 2 chuyển sang unfreeze các khối 6, 14, 15. Ngoài ra, lớp Dense trong encoder và toàn bộ decoder vẫn được mở khóa như cũ.

Mục tiêu là kiểm tra xem hai thay đổi này – đặc biệt là việc sử dụng khối SqueezeFormer 6 (với độ phân giải thời gian 40ms) thay cho khối 13 (80ms) – có mang lại cải thiện về hiệu quả nhận dạng tiếng nói hay không, trong khi vẫn đảm bảo số lượng tham số học được giữ ở mức giới hạn khoảng **1.600.000 tham số**.

4.6.4.3 Cách tiến hành

Chỉ một mô hình duy nhất được huấn luyện trong phương án này, với kiến trúc SqueezeFormer giữ nguyên như các phương án trước. Tuy nhiên, hai điểm khác biệt chính so với Phương án 1 – Trường hợp 4 là:

- **Đầu ra gồm 93 lớp** thay vì 128 lớp như trước;
- **Unfreeze khối 6 thay cho khối 13**, đồng thời vẫn giữ nguyên unfreeze các khối 14, 15, lớp Dense và decoder.

Hình 3.5. minh họa kiến trúc mô hình SqueezeFormer được sử dụng. Tổng cộng có 16 khối SqueezeFormer trong encoder, được đánh số từ 0 đến 15. Trong đó:

- Các khối từ 0 đến 6 hoạt động với độ phân giải thời gian **40ms**;
 - Các khối từ 7 đến 14 hoạt động ở mức **80ms** do downsampling;
 - Khối 15 quay trở lại mức **40ms**, nhận tín hiệu kết nối tắt từ khối 6.

Việc lựa chọn hai cấu hình khác nhau trong thử nghiệm xuất phát từ giả thuyết rằng: **các khối hoạt động ở mức thời gian 40ms (ví dụ khôi 6)** có khả năng giữ lại thông tin chi tiết hơn so với các khôi 80ms, từ đó có thể ảnh hưởng tích cực đến hiệu quả mô hình khi được fine-tuning.

4.6.4.4 Thiết kế số lượng lớp

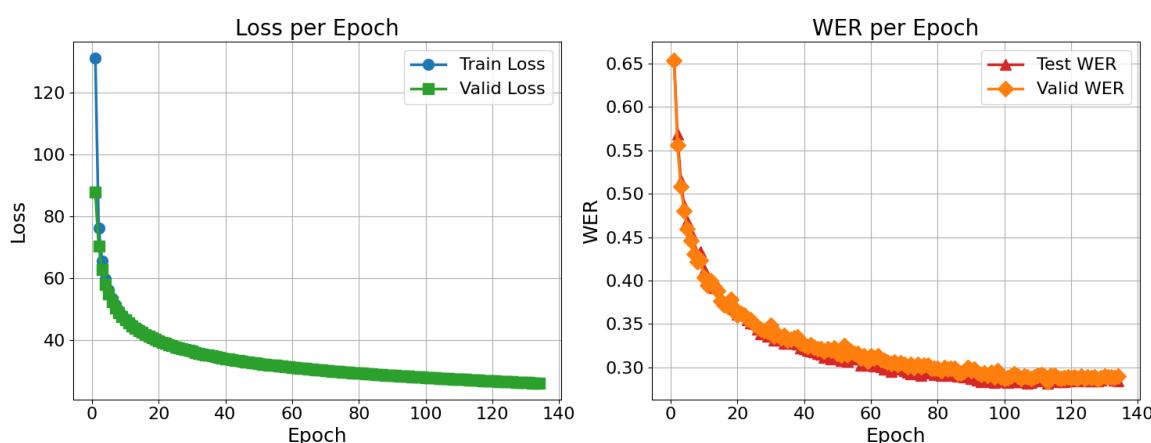
“”, “<s>”, “</s>”, “”, “n”, “h”, “t”, “i”, “c”, “g”, “a”, “m”, “u”, “d”, “à”, “o”, “ú”, “í”, “v”, “r”, “y”, “á”, “b”, “ô”, “p”, “k”, “s”, “ó”, “é”, “á”, “ô”, “ò”, “á”, “ê”, “é”, “d”, “í”, “â”, “ô”, “â”, “ó”, “ó”, “è”, “q”, “û”, “ê”, “ó”, “í”, “á”, “â”, “x”, “í”, “â”, “e”, “ú”, “ñ”, “ü”, “ó”, “â”, “ú”, “ö”, “ö”, “â”, “ü”, “ö”, “ù”, “ö”, “ü”, “ä”, “í”, “ë”, “ý”, “ö”, “ü”, “é”, “â”, “í”, “ë”, “â”, “â”, “ö”, “ë”, “é”, “ë”, “ý”, “è”, “ý”, “ö”, “â”, “ö”, “â”

Việc giảm số lượng lớp đầu ra được thực hiện nhằm đạt được các mục tiêu sau:

- Bộ ký tự 93 là **đủ để biểu diễn toàn bộ từ vựng tiếng Việt** trong tập dữ liệu huấn luyện và thử nghiệm;
 - Giảm số lượng lớp giúp **tăng tốc độ hội tụ và giảm độ phức tạp tính toán**, phù hợp với quy mô dữ liệu và giới hạn phần cứng;
 - Bộ ký tự này được thiết kế để **tương thích với mô hình ngôn ngữ KenLM**, vốn hoạt động tốt nhất khi đầu vào là chuỗi ký tự đơn (character-level), không chồng lấp.

4.6.4.5 Kết quả huấn luyện

Kết quả huấn luyện sau 135 epoch được thể hiện như trong Hình 4.12.:



Hình 4.12. Kết quả huấn luyện sau 135 epoch trên tập Speaker + VIVOS

Bảng 4.8. Kết quả WER (%) trên Speaker + VIVOS (Thử nghiệm I - Phương án 2)

Tập dữ liệu	Loại	WER no LM (%)
Speaker + VIVOS	Test	26.97
	Valid	28.02

Nhận xét:

- Đường cong **train loss** và **validation loss** gần nhau, cho thấy mô hình không bị overfit và vẫn tiếp tục học ổn định trong quá trình fine-tuning.
- Hiện tượng này phù hợp với kỳ vọng, vì mô hình đã được tiền huấn luyện trên tập dữ liệu lớn và giờ chỉ điều chỉnh lại trên một tập nhỏ hơn. Do đó, loss ở cả tập huấn luyện và tập xác thực có xu hướng giảm đồng đều.
- Kết quả cũng chỉ ra rằng mô hình vẫn còn khả năng cải thiện nếu tiếp tục mở rộng quy mô dữ liệu đầu vào.
- Kết quả WER trên tập dữ liệu **Speaker + VIVOS** khi sử dụng **93 lớp đầu ra** không những không bị suy giảm mà còn có xu hướng **cải thiện nhẹ** so với phương án sử dụng 128 lớp. Điều này khẳng định rằng việc rút gọn số lượng lớp đầu ra về 93 ký tự đặc trưng cho tiếng Việt là hoàn toàn khả thi, không gây mất thông tin trong quá trình giải mã.
- Đồng thời, việc thay đổi chiến lược fine-tuning bằng cách **unfreeze khối 6 thay cho khối 13** đã cho thấy kết quả WER tốt hơn. Điều này cung cấp giả thuyết rằng các khối hoạt động ở mức thời gian 40ms như khối 6 có thể bảo toàn nhiều thông tin chi tiết hơn, giúp mô hình học hiệu quả hơn trong giới hạn tham số.

4.6.4.6 Kết luận

Sự kết hợp giữa hai thay đổi: (i) **sử dụng 93 lớp đầu ra** và (ii) **điều chỉnh cấu hình unfreeze encoder** đã mang lại mô hình có hiệu quả tốt nhất tính đến thời điểm hiện tại trong phạm vi thử nghiệm, đồng thời phù hợp với ràng buộc về **giới hạn phần cứng**.

Định hướng phát triển: Để tiếp tục cải thiện hiệu suất mô hình, bước tiếp theo là mở rộng quy mô tập dữ liệu bằng cách thêm **tập database** — một nguồn dữ liệu lớn và phong phú hơn về nội dung. Việc huấn luyện mô hình trên tập dữ liệu đa dạng hơn sẽ giúp mô hình tăng khả năng tổng quát hoá và cải thiện hiệu quả khi triển khai thực tế trong ứng dụng soạn thảo văn bản tiếng Việt.

4.7 Thủ nghiệm 2: Đánh giá vai trò kích thước dữ liệu huấn luyện đối với hệ thống nhận dạng tiếng nói

Như đã trình bày trong Thủ nghiệm 1, mô hình âm học đạt kết quả tối ưu nhất khi áp dụng kỹ thuật đóng băng các tầng **6, 14, 15** thuộc encoder và decoder, đồng thời giảm số lượng lớp đầu ra từ **128** xuống còn **93**. Trong Thủ nghiệm 2, sinh viên tiếp tục sử dụng cấu hình này nhằm đảm bảo tính nhất quán, tuy nhiên mở rộng quy mô huấn luyện để đánh giá ảnh hưởng của kích thước dữ liệu đến chất lượng hệ thống nhận dạng tiếng nói.

4.7.0.1 Trường hợp 1: Mở rộng tập dữ liệu với folder database (thêm 7000 files)

Trong thử nghiệm 2 - trường hợp 1, sinh viên mở rộng quy mô dữ liệu huấn luyện bằng cách kết hợp thêm khoảng 7.000 tệp âm thanh từ thư mục **database (VLSP 2020)** vào các tập Speaker và VIVOS trước đó. Mục tiêu là đánh giá tác động của việc tăng dữ liệu đến hiệu suất của mô hình trong điều kiện các siêu tham số và số trọng số học vẫn giữ nguyên.

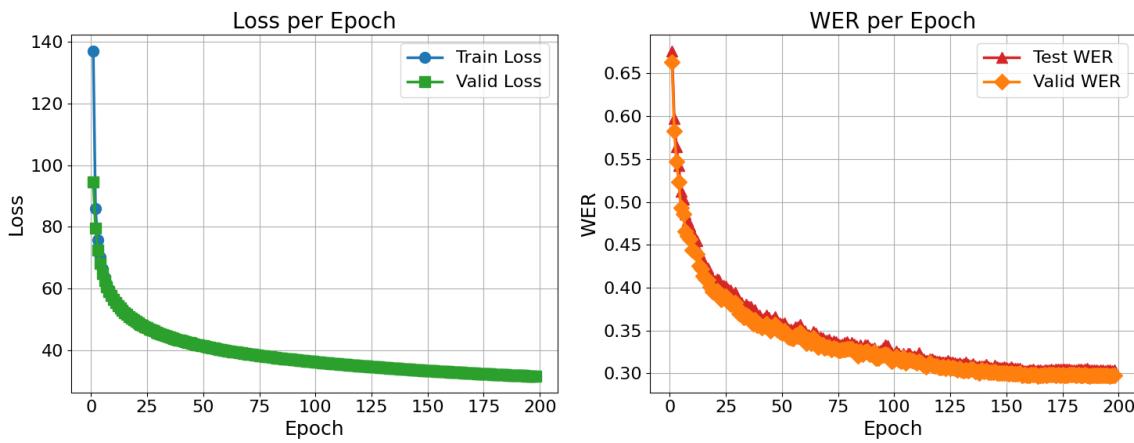
Bảng 4.9. Thông tin chia tập dữ liệu VLSP, VIVOS (Thử nghiệm 2 - Trường hợp 1)

Tập dữ liệu	Loại	Số lượng (files)	Cách chia
Speaker (VLSP 2020)	Train	11064	600 người
	Valid	1412	80 người
	Test	1387	80 người
VIVOS	Train	9875	40 người
	Valid	1698	6 người
	Test	757	19 người
Database 1 (VLSP 2020)	Train	5530	Tỷ lệ 7:1.5:1.5
	Valid	1000	
	Test	1000	

Kết quả được thể hiện trên hình 4.13. Sau khi huấn luyện với tập dữ liệu mở rộng, mô hình được đánh giá lại trên tập Speaker + VIVOS (để đảm bảo tính so sánh với Trường hợp 1). Kết quả được trình bày trong bảng 4.10.:

Bảng 4.10. Kết quả WER (%) trên Speaker + VIVOS (Thử nghiệm 2 - trường hợp 1)

Tập dữ liệu	Loại	WER no LM (%)
Speaker + VIVOS	Train	18.90
	Test	26.40
	Valid	27.23



Hình 4.13. Kết quả sau 200 epochs

Nhận xét:

- Kết quả trên tập test và validation đều được cải thiện rõ rệt so với thử nghiệm 1, chứng minh vai trò quan trọng của việc mở rộng dữ liệu trong quá trình fine-tuning.
- Chỉ số WER trên tập train có tăng nhẹ so với thử nghiệm 1, điều này hợp lý vì dữ liệu đa dạng hơn, mô hình có xu hướng tổng quát hóa tốt hơn thay vì ghi nhớ dữ liệu train.
- Việc thêm dữ liệu từ tập database đã giúp mô hình nhận diện được nhiều kiểu ngôn ngữ nói khác nhau, đặc biệt là các câu dài, đa dạng và nói nhanh.

Hướng phát triển tiếp theo: Trong các thử nghiệm sắp tới, sinh viên sẽ tiếp tục tăng cường tập huấn luyện bằng cách bổ sung thêm dữ liệu từ các nguồn khác như **phân còn lại của folder database** và **dữ liệu từ FPT**. Mục tiêu là cải thiện khả năng tổng quát của mô hình và hướng tới việc triển khai trong thực tế — phục vụ người dùng trong các ứng dụng soạn thảo văn bản bằng giọng nói với chất lượng cao và độ chính xác ổn định.

4.7.0.2 Trường hợp 2: Tăng cường dữ liệu với Database (thêm 11 000 files) và FPT

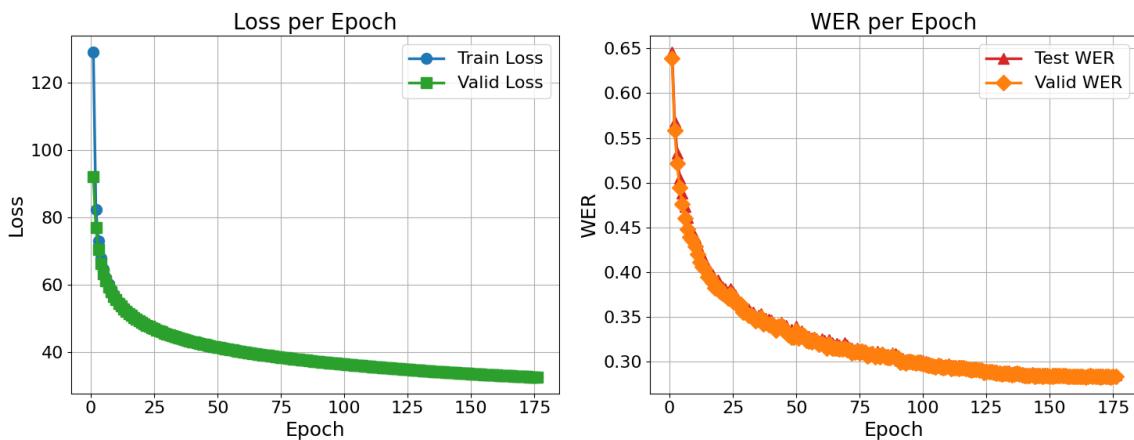
Trong thử nghiệm 2 - trường hợp 2, sinh viên tiếp tục mở rộng quy mô dữ liệu huấn luyện bằng cách bổ sung phần còn lại của tập **Database (VLSP 2020)** (khoảng 11 000 files) cũng như tích hợp thêm **tập dữ liệu tiếng nói từ FPT** nhằm gia tăng độ đa dạng ngữ âm, âm sắc và ngữ cảnh ngôn ngữ trong quá trình huấn luyện. Mục tiêu là đánh giá khả năng tổng quát hóa của mô hình khi huấn luyện với tập dữ liệu lớn và phong phú hơn.

Tập dữ liệu sử dụng: Giữ nguyên các tập Speaker và VIVOS như ở các thử nghiệm trước, và bổ sung thêm dữ liệu theo bảng 4.11.:

Bảng 4.11. Thông tin chia tập dữ liệu sử dụng VLSP, VIVOS, FPT trong thử nghiệm 2 - trường hợp 2

Tập dữ liệu	Loại	Số lượng (files)	Cách chia
Speaker (VLSP 2020)	Train	11064	600 người
	Valid	1412	80 người
	Test	1387	80 người
VIVOS	Train	9875	40 người
	Valid	1698	6 người
	Test	757	19 người
Database 2 (VLSP 2020) - có gồm cả Database 1	Train	9351	Tỷ lệ 8:1:1
	Valid	1000	
	Test	1000	
FPT	Train	6317	Tỷ lệ 7.5:1:1
	Valid	1000	
	Test	1000	

Kết quả huấn luyện: Sau 175 epoch, biểu đồ loss và WER thể hiện trong Hình 4.14. cho thấy xu hướng giảm đều, ổn định, thể hiện mô hình vẫn đang học và chưa xảy ra hiện tượng overfitting.



Hình 4.14. Kết quả huấn luyện sau 175 epoch với dữ liệu mở rộng

Đánh giá mô hình trên tập Speaker + VIVOS:

Định hướng phát triển:

Dựa trên kết quả tích cực của thử nghiệm này, hướng phát triển tiếp theo sẽ tập trung vào:

Bảng 4.12. Kết quả WER (%) trên các tập dữ liệu (Thử nghiệm 2 - Trường hợp 2)

Tập dữ liệu	Loại	WER no LM (%)
Speaker + VIVOS	Train	18.03
	Valid	25.40
	Test	24.36
Database 2 (VLSP 2020)	Train	26.60
	Valid	33.98
	Test	32.16
FPT	Train	23.02
	Valid	28.67
	Test	28.57

- **Bổ sung phần còn lại của tập dữ liệu database (VLSP 2020)** vào quá trình huấn luyện nhằm tận dụng tối đa nguồn dữ liệu đã có.
- Tiếp tục duy trì cấu hình số tham số học (1.6M) đã đạt hiệu quả tốt nhất trong giới hạn phân cứng hiện tại.
- Theo dõi và đánh giá WER trên các tập dữ liệu chuẩn để kiểm tra mức độ cải thiện của mô hình khi tăng dữ liệu huấn luyện.

Nhận xét:

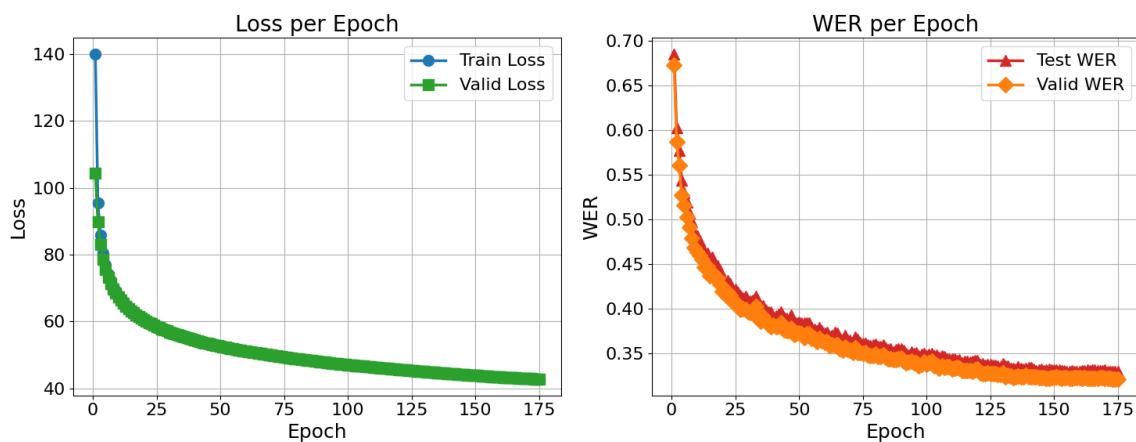
- Hiệu suất mô hình tiếp tục được cải thiện rõ rệt so với các thử nghiệm trước, đặc biệt trên tập test và valid.
- Việc bổ sung dữ liệu từ tập FPT và sử dụng toàn bộ phần còn lại của tập database đã giúp mô hình tiếp xúc với nhiều ngữ cảnh ngôn ngữ hơn, dẫn đến khả năng tổng quát hoá tốt hơn.
- Trong quá trình kiểm tra thực tế, mô hình huấn luyện từ thử nghiệm này cũng cho kết quả tốt hơn rõ rệt khi nhận dạng các câu nói ngẫu nhiên từ người dùng.

4.7.0.3 Trường hợp 3: Bổ sung thêm dữ liệu huấn luyện từ Database (thêm 20 000 files)

Ở trường hợp này, mô hình được huấn luyện trên tập dữ liệu mở rộng hơn so với Trường hợp 2, cụ thể là bổ sung toàn bộ phần còn lại từ tập **Database (VLSP 2020)**. Thông tin chi tiết về cách chia tập dữ liệu được trình bày trong Bảng 4.13.:

Bảng 4.13. Thông tin chia tập dữ liệu Database (VLSP 2020)

Tập dữ liệu	Loại	Số lượng (files)	Cách chia
Database 3 (VLSP 2020) đã bao gồm database 2	Train	15621	Tỷ lệ 8:1:1
	Valid	2000	
	Test	2000	



Hình 4.15. Kết quả sau 175 epochs Thử nghiệm 2 - trường hợp 3

Kết quả WER trên cụ thể từng tập dữ liệu thể hiện trong bảng 4.14.

Bảng 4.14. Kết quả WER trên các tập dữ liệu không sử dụng mô hình ngôn ngữ (Thử nghiệm 2- Trường hợp 3)

Tập dữ liệu	Loại	WER (no LM) (%)
Speaker (VLSP) + VIVOS	TRAIN	19.95
Speaker (VLSP) + VIVOS	TEST	24.50
Speaker (VLSP) + VIVOS	VALID	25.38
FPT	TRAIN	23.23
FPT	TEST	28.12
FPT	VALID	28.61
Database	TRAIN	31.93
Database	TEST	40.72
Database	VALID	40.54
Tự thu	TEST	29.80

Nhận xét: Kết quả cho thấy WER đã được cải thiện rõ rệt trên các tập kiểm thử (Test) và tập xác thực (Valid), nhờ vào việc mở rộng quy mô dữ liệu huấn luyện. Điều này một lần nữa khẳng định vai trò quan trọng của dữ liệu trong việc nâng cao chất lượng mô hình ASR.

Tuy nhiên, kết quả trên tập **FPT** và **Database** vẫn còn thấp hơn so với tập **Speaker** và **VIVOS**. Nguyên nhân có thể đến từ một số yếu tố sau:

- Các tập **Speaker** và **VIVOS** có chất lượng âm thanh cao, ít nhiễu, được thu theo từng người và có cấu trúc rõ ràng.
- Một số tệp trong tập **Database** và **FPT** không khớp hoàn toàn giữa âm thanh và transcript, gây nhiễu cho quá trình học.
- Hai tập này chứa nhiều đoạn âm thanh có mức độ tiếng ồn vừa đến lớn, làm giảm độ chính xác của mô hình.
- Nội dung transcript có nhiều cụm từ hiếm gặp, mang tính cảm thán hoặc không chuẩn cú pháp (ví dụ: “ú òa”, “a a a”, “ảm ương”,...), gây khó khăn cho mô hình trong việc dự đoán chính xác.

4.8 Thủ nghiệm 3: Thủ nghiệm tối ưu hóa tham số Alpha và Beta trong Beam Search kết hợp Language Model

4.8.1 Mục tiêu thử nghiệm

Mục tiêu của thử nghiệm này là tìm ra bộ tham số **Alpha** và **Beta** tối ưu nhằm cải thiện hiệu suất giải mã khi kết hợp giữa mô hình âm học và mô hình ngôn ngữ trong quá trình sử dụng thuật toán *Beam Search*. Hai tham số này ảnh hưởng trực tiếp đến chất lượng nhận dạng đầu ra thông qua việc điều chỉnh tỷ trọng giữa xác suất từ mô hình âm học và mô hình ngôn ngữ trong quá trình tính điểm cho mỗi chuỗi từ được tạo ra.

- **Alpha:** Hệ số khuếch đại trọng số của mô hình ngôn ngữ. Giá trị Alpha càng lớn thì hệ thống càng ưu tiên dựa trên mô hình ngôn ngữ.
- **Beta:** Tham số điều chỉnh phần thưởng hoặc phạt dựa trên độ dài câu (word insertion bonus/penalty), giúp cân bằng giữa việc thêm hoặc lược bỏ từ trong kết quả đầu ra.

4.8.2 Cách tiến hành

Để đánh giá ảnh hưởng của hai tham số này và tìm ra bộ giá trị tối ưu, sinh viên đã thiết lập một loạt thử nghiệm trên tập dữ liệu thử nghiệm bao gồm **300 tệp âm thanh được chọn ngẫu nhiên** từ tập test của hai tập dữ liệu phổ biến là **Speaker (VLSP 2020)** và **VIVOS**.

Các giá trị của Alpha và Beta được điều chỉnh trong khoảng từ 0.2 đến 1.5, với bước nhảy phù hợp (ví dụ: 0.1 hoặc 0.2) để đảm bảo thử nghiệm được thực hiện trên một không gian tham số đủ rộng, qua đó thu được cái nhìn toàn diện về tác động của từng kết hợp tham số lên độ chính xác nhận dạng.

Mỗi cặp giá trị (α, β) được sử dụng để giải mã toàn bộ tập thử nghiệm nói trên, và kết quả được đánh giá dựa trên chỉ số **WER (Word Error Rate)**. Từ đó, sinh viên lựa chọn bộ tham số cho kết quả WER thấp nhất để áp dụng cho các bước triển khai tiếp theo.

```
decoder = build_ctcdecoder(
    labels=labels,
    kenlm_model_path= r"/mnt/f/Luu_Dinh_Tu/Project_2/kenlm/build/model.bin",
    unigrams=unigram_list,
    alpha= 0.8,
    beta= 0.2,
)
```

Hình 4.16. Cấu trúc tích hợp mô hình KenLM vào Decoder

Kết quả một số thử nghiệm được trình bày trong bảng 4.15. dưới đây:

Bảng 4.15. Kết quả thử nghiệm các giá trị Alpha và Beta ảnh hưởng đến WER (300 files)

Alpha	Beta	WER (%)
0.5	0.6	18.41
0.5	0.8	18.41
0.5	1.0	18.31
0.5	1.5	18.09
0.7	0.0	16.95
0.7	0.2	16.90
0.7	0.5	16.90
0.7	0.6	16.85
0.7	0.8	16.70
0.7	1.0	16.80
0.7	1.5	16.53
0.9	1.5	15.54
1.0	1.0	15.49
1.0	1.5	15.49
1.2	1.5	15.24
1.5	1.5	16.18

4.8.3 Nhận xét:

Từ bảng 4.13., có thể nhận thấy rằng khi tăng dần giá trị của Alpha và Beta lên gần mức 1.0 hoặc cao hơn, kết quả WER có xu hướng giảm rõ rệt. Cụ thể, với Alpha = 1.2 và Beta = 1.5, mô hình đạt kết quả tốt nhất với **WER = 15.24%** trên tập thử nghiệm. Điều này chứng minh vai trò quan trọng của việc điều chỉnh các tham số decoder khi kết hợp với mô hình ngôn ngữ.

Tuy nhiên, do điều kiện về mặt thời gian và tài nguyên tính toán, sinh viên chỉ tiến hành thử nghiệm trên một phần nhỏ dữ liệu test. Trong tương lai, việc mở rộng thử nghiệm trên toàn bộ tập test là cần thiết để khẳng định độ ổn định và tổng quát của các giá trị Alpha và Beta đã chọn

4.8.4 Kết luận

Từ các thử nghiệm trên, sinh viên quyết định lựa chọn bộ tham số **Alpha = 1.2** và **Beta = 1,5** làm cấu hình mặc định cho hàm Decoder sử dụng Beam Search kết hợp với mô hình ngôn ngữ (KenLM) cho các thí nghiệm và đánh giá về sau trong quá trình xây dựng hệ thống soạn thảo văn bản tiếng Việt dựa trên nhận dạng tiếng nói.

4.9 Thử nghiệm 4: Đánh giá vai trò mô hình ngôn ngữ trong hệ thống nhận dạng tiếng nói

4.9.1 Mục tiêu thử nghiệm

Mục tiêu của thử nghiệm này là đánh giá vai trò và mức độ đóng góp của mô hình ngôn ngữ trong việc cải thiện chất lượng hệ thống nhận dạng tiếng nói tiếng Việt. Cụ thể, thử nghiệm nhằm đo lường mức độ giảm sai số từ (Word Error Rate - WER) khi tích hợp mô hình ngôn ngữ vào hệ thống nhận dạng.

4.9.2 Phương pháp tiến hành

Trong thử nghiệm này, mô hình âm học được sử dụng là mô hình đã được tối ưu tốt nhất từ Thử nghiệm 2, với dữ liệu huấn luyện đã được tăng cường tối đa. Mô hình này sau đó được tích hợp với mô hình ngôn ngữ n-gram nhằm nâng cao chất lượng nhận dạng tiếng nói. Quá trình huấn luyện mô hình ngôn ngữ n-gram (gồm các phiên bản 3-gram và 4-gram) được thực hiện dựa trên tập dữ liệu văn bản và quy trình huấn luyện đã được trình bày tại Chương Thiết kế hệ thống, mục 3.2.5.

Trong giai đoạn giải mã (decoding), thuật toán beam search được sử dụng kết hợp với mô hình ngôn ngữ. Các hệ số alpha và beta – đại diện cho trọng số giữa xác suất của mô hình âm vị và mô hình ngôn ngữ – được điều chỉnh nhằm tìm ra cấu hình mang lại kết quả WER tối ưu.

Cuối cùng, hệ thống được đánh giá trên các tập dữ liệu kiểm thử đã được xây dựng ở thử nghiệm trước. Kết quả sẽ được phân tích để làm rõ mức độ cải thiện WER khi kết hợp beam search với mô hình ngôn ngữ.

4.9.3 Kết quả WER trên các tập dữ liệu khi tích hợp language model 4-gram

Do các mô hình âm vị sử dụng đầu ra 93 lớp trong Thủ nghiệm 2- trường hợp 3 đã cho thấy hiệu quả nhận dạng vượt trội nhất trong các thử nghiệm, sinh viên lựa chọn sử dụng các mô hình này để kết hợp với mô hình ngôn ngữ dạng 4-gram. Các tham số α , β tối ưu cho thuật toán Beam Search kết hợp Language Model được xác định tại Mục 4.8 sẽ được áp dụng trong quá trình giải mã đầu ra của hệ thống. Mục tiêu của bước thử nghiệm tiếp theo là đánh giá toàn diện chất lượng hệ thống nhận dạng tiếng nói tự động trên các tập dữ liệu đã được chia theo từng trường hợp cụ thể như đã trình bày.

Thử nghiệm 1 - Phương án 2

Bảng 4.16. WER trên Speaker + VIVOS với LM 4-gram (TN1-PA2)

Tập dữ liệu	Loại	WER no LM (%)	WER with LM 4 gram (%)
Speaker + VIVOS	Train	14	2.18
	Test	26.97	8.24
	Valid	28.02	8.47

Thử nghiệm 2 - Trường hợp 1

Bảng 4.17. WER trên Speaker + VIVOS với LM 4-gram (TN2-TH1)

Tập dữ liệu	Loại	WER no LM (%)	WER with LM 4 gram (%)
Speaker + VIVOS	Train	18.90	3.06
	Test	26.40	6.85
	Valid	27.23	7.01

Thử nghiệm 2 - trường hợp 2

Bảng 4.18. WER trên Speaker + VIVOS với LM 4-gram (TN2 – TH2)

Tập dữ liệu	Loại	WER no LM (%)	WER with LM 4 gram (%)
Speaker + VIVOS	Train	18.03	2.97
	Test	24.36	6.17
	Valid	25.40	6.07

Thử nghiệm 2 - trường hợp 3

Bảng 4.19. WER trên các tập dữ liệu với LM 4-gram (TN2 – TH3)

Tập dữ liệu	Loại	WER no LM (%)	WER with LM 4-gram (%)
Speaker + VIVOS	Train	19.95	3.53
Speaker + VIVOS	Test	24.50	5.89
Speaker + VIVOS	Valid	25.38	5.88
FPT	Train	23.23	4.52
FPT	Test	28.12	7.71
FPT	Valid	28.61	8.20
Database	Train	31.93	12.10
Database	Test	40.72	19.02
Database	Valid	40.54	18.72
Tự thu	Test	29.80	7.54

4.9.4 Nhận xét

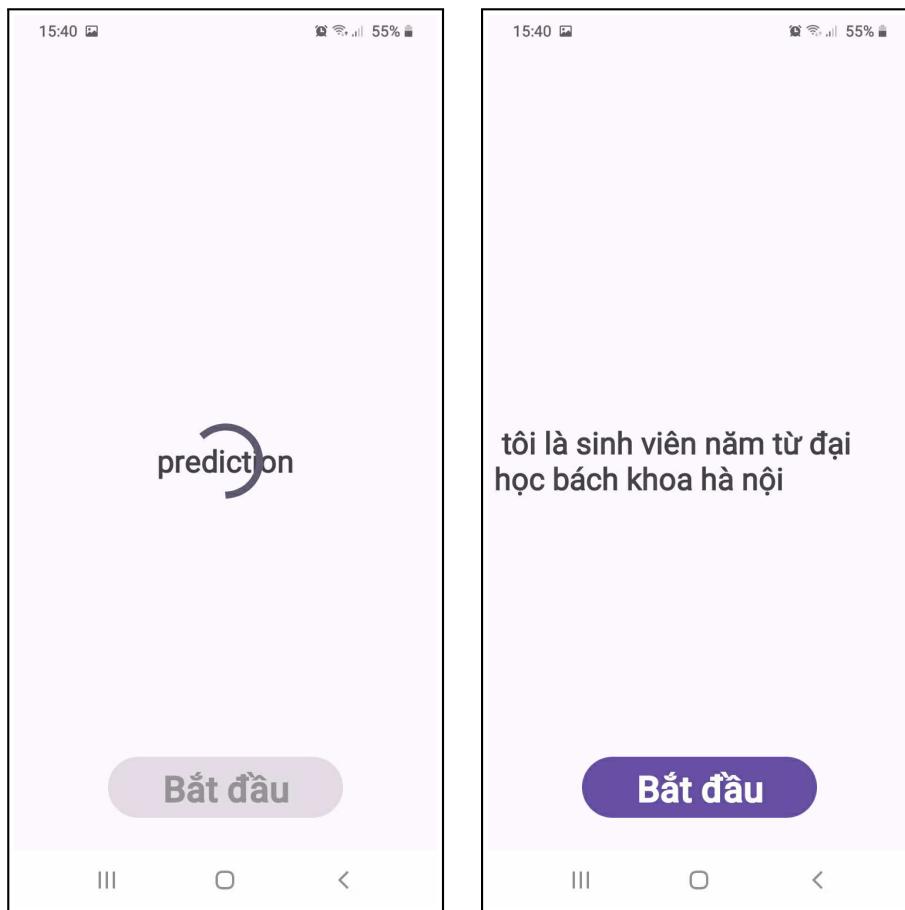
Mô hình ngôn ngữ 4-gram đã góp phần cải thiện rõ rệt chất lượng hệ thống. Các tập dữ liệu như Speaker (VLSP 2020), VIVOS và FPT đều đạt WER dưới 10%, cho thấy hiệu quả của việc kết hợp mô hình ngôn ngữ. Tuy nhiên, kết quả trên tập Database vẫn còn hạn chế do chất lượng âm thanh không đồng đều và văn bản chưa được chuẩn hóa.

Bốn trường hợp thử nghiệm tương ứng với các bảng kết quả cũng cho thấy rằng hiệu quả của mô hình ngôn ngữ phụ thuộc lớn vào chất lượng mô hình âm vị. Khi mô hình âm vị tốt hơn (ở các trường hợp sau của Phương án 2), việc kết hợp mô hình ngôn ngữ 4-gram giúp cải thiện kết quả đáng kể hơn.

4.10 Thủ nghiệm 5: Đánh giá khả năng tích hợp mô hình âm học vào ứng dụng hiển thị văn bản tiếng Việt trên Android

Sau khi hoàn tất quá trình huấn luyện trong Thủ nghiệm 2, mô hình âm học có hiệu suất cao nhất, được huấn luyện với tập dữ liệu đã được tăng cường tối đa, được lựa chọn để triển khai trên nền tảng Android. Việc triển khai này nhằm đánh giá khả năng hoạt động thực tế của hệ thống nhận dạng tiếng nói tiếng Việt trong môi trường ứng dụng di động. Mô hình này đại diện cho cấu hình tối ưu nhất trong toàn bộ quá trình thử nghiệm.

Quy trình triển khai chi tiết được trình bày trong mục 3.3.3. Dưới đây là kết quả sử dụng ứng dụng sau khi tích hợp mô hình:



(a) Quá trình suy luận trên ứng dụng Android

(b) Hiển thị kết quả nhận dạng trên ứng dụng Android

Hình 4.17. Quy trình xử lý và kết quả nhận dạng trên ứng dụng Android

Trong thí nghiệm thực tế, người dùng bấm nút trên giao diện ứng dụng để bắt đầu ghi âm. Âm thanh thu được từ micro điện thoại được xử lý và đưa vào mô hình nhận dạng. Kết quả dự đoán sau suy luận sẽ hiển thị dưới dạng văn bản ngay trên màn hình ứng dụng.

Cụ thể, khi người dùng đọc câu "tôi là sinh viên năm tư đại học bách khoa hà nội", kết quả hiển thị trên ứng dụng là "tôi là sinh viên năm từ đại học bách khoa hà nội", tức là sai tại từ "tư" → "từ". Điều này cho thấy hệ thống hoạt động tương đối chính xác, tuy nhiên vẫn tồn tại một số lỗi nhận dạng.

Đánh giá và phân tích

Mặc dù mô hình hoạt động tốt trên tập dữ liệu kiểm thử trong môi trường huấn luyện, chất lượng nhận dạng khi triển khai trên thiết bị Android vẫn bị ảnh hưởng bởi nhiều yếu tố:

- **Chất lượng micro của thiết bị:** Các thiết bị Android có chất lượng phần cứng micro khác nhau, ảnh hưởng trực tiếp đến chất lượng tín hiệu đầu vào.

- **Môi trường thu âm:** Các tiếng ồn xung quanh (nền, quạt, người nói khác...) không được lọc bỏ có thể làm sai lệch đặc trưng âm thanh đầu vào.
- **Chuyển đổi mô hình:** Mô hình được chuyển từ TensorFlow sang định dạng `.tflite` để giảm kích thước và tương thích với Android. Việc này có thể làm giảm độ chính xác do một số giới hạn về toán tử và độ chính xác tính toán.
- **Giới hạn phần cứng:** Thiết bị di động có tài nguyên tính toán hạn chế hơn nhiều so với môi trường huấn luyện (GPU/TPU). Do đó, tốc độ dự đoán có thể nhanh hơn nhưng có thể ảnh hưởng đến độ sâu tính toán và độ chính xác suy luận.

Kết luận

Việc tích hợp mô hình nhận dạng tiếng nói vào nền tảng Android cho thấy khả năng áp dụng thực tiễn của hệ thống. Dù vẫn còn tồn tại một số sai số nhỏ, kết quả đạt được là khả quan, chứng minh mô hình có thể được triển khai hiệu quả cho các ứng dụng soạn thảo văn bản tiếng Việt dựa trên giọng nói.

4.11 Thủ nghiệm 6: Đánh giá hiệu quả chế độ nhận dạng tiếng nói trực tuyến (Online ASR)

4.11.1 Mục đích thử nghiệm

Thử nghiệm nhằm đánh giá hiệu quả và độ ổn định của hệ thống khi chuyển sang chế độ nhận dạng tiếng nói trực tuyến (online ASR). Việc triển khai mô hình trong môi trường giả lập thời gian thực cho phép kiểm tra khả năng ứng dụng thực tế của hệ thống, đặc biệt trong các tình huống yêu cầu tương tác nhanh và liên tục với người dùng.

4.11.2 Phương pháp tiến hành

Sinh viên đã tiến hành xây dựng cơ chế nhận dạng tiếng nói trực tuyến theo mô tả tại mục 3.4. Hệ thống sử dụng mô hình âm học được huấn luyện trong **Thử nghiệm 2 – Trường hợp 3**, kết hợp với thuật toán giải mã *beam search* và mô hình ngôn ngữ thống kê 4-gram nhằm nâng cao độ chính xác trong quá trình suy diễn đầu ra. Trong chế độ hoạt động trực tuyến, hệ thống được thiết kế gồm hai luồng xử lý song song:

- **Luồng thu âm:** Thực hiện thu tín hiệu âm thanh liên tục từ micro của máy tính xách tay, lưu trữ vào vùng đệm chung.
- **Luồng nhận dạng:** Định kỳ sau mỗi khoảng thời gian 1.75 giây, toàn bộ dữ liệu âm thanh tạm thời được đưa vào mô hình để tiến hành xử lý và nhận dạng.

Khoảng thời gian 1.75 giây được lựa chọn một cách có chủ đích, dựa trên đặc trưng trung bình của các cụm từ có ý nghĩa hoàn chỉnh trong giao tiếp thông thường. Việc lựa chọn mốc thời gian này không chỉ giúp tối ưu hóa độ trễ cảm nhận của người sử dụng, mà còn tạo điều kiện thuận lợi để mô hình ngôn ngữ n-gram phát huy hiệu

quả trong việc tái cấu trúc câu đầu ra một cách tự nhiên, mạch lạc và chuẩn ngữ pháp hơn.

4.11.3 Trên máy tính xách tay (Laptop)

Người dùng nói liên tục câu sau: "**hôm nay là thứ hai trời nắng, ngày mai là thứ ba trời có thể mưa dông**". Trong quá trình nói, hệ thống hiển thị kết quả nhận dạng theo thời gian thực như các hình dưới đây:

```
Nhấn 'r' để bắt đầu ghi âm, nhấn lại 'r' để dừng.  
Đang ghi âm...  
hôm nay[]
```

Hình 4.18. Giai đoạn đầu: Hệ thống nhận dạng được từ khóa đầu tiên "hôm nay"

```
Nhấn 'r' để bắt đầu ghi âm, nhấn lại 'r' để dừng.  
Đang ghi âm...  
hôm nay là thứ hai trời nắng[]
```

Hình 4.19. Giai đoạn tiếp theo: Nhận dạng mở rộng thành "hôm nay là thứ hai trời nắng"

```
Nhấn 'r' để bắt đầu ghi âm, nhấn lại 'r' để dừng.  
Đang ghi âm...  
hôm nay là thứ hai trời nắng ngày mai là thứ ba[]
```

Hình 4.20. Tiếp tục suy diễn: Hệ thống hiển thị "ngày mai là thứ ba trời"

```
Đang ghi âm...  
hôm nay là thứ hai trời nắng ngày mai là thứ ba trời có thể mưa dông  
kết thúc.
```

Hình 4.21. Cuối cùng: Câu hoàn chỉnh được nhận dạng đầy đủ

Nhận xét:

- Hệ thống nhận dạng tiếng nói theo thời gian thực tương đối tốt.
- Tốc độ xử lý và hiển thị văn bản gần như đồng bộ với tốc độ nói thông thường của người dùng.
- Các từ hiển thị được cập nhật dần theo từng cụm từ, đúng với kỳ vọng của chế độ Online ASR.

- Một số cụm từ như “có thể mưa dông” vẫn còn lỗi nhỏ nhưng không ảnh hưởng lớn đến ngữ nghĩa.

Kết luận: Việc triển khai mô hình trên laptop với chế độ trực tuyến hoạt động hiệu quả và có thể mở rộng để ứng dụng trong thực tế như hệ thống ghi biên bản cuộc họp hoặc soạn thảo văn bản tiếng Việt.

4.11.4 Trên thiết bị Android

Để triển khai chế độ nhận dạng tiếng nói trực tuyến (Online ASR) trên ứng dụng Android, sinh viên sử dụng phương pháp tương tự như đã thực hiện trên máy tính xách tay (mục 4.11.3). Mô hình âm vị sử dụng trong quá trình này là mô hình tốt nhất ở Thủ nghiệm 2- trường hợp 3, đã được chuyển sang định dạng TensorFlow Lite (.tflite) nhằm đáp ứng yêu cầu triển khai trên thiết bị di động.

Tuy nhiên, khi chuyển đổi sang định dạng .tflite, kích thước mô hình được giảm xuống đáng kể nhưng kéo theo tốc độ suy luận (inference) cũng bị giảm. Điều này ảnh hưởng đến độ trễ trong chế độ Online, đặc biệt khi ứng dụng chạy trên thiết bị có phần cứng hạn chế.



Hình 4.22. Quy trình xử lý và kết quả nhận dạng chế độ Online ASR trên ứng dụng Android

4.12 Đánh giá chất lượng hệ thống nhận dạng tiếng nói tiếng Việt với các thông số RTF và Latency

4.12.1 Hệ thống hoạt động trên máy tính xách tay (laptop)

4.12.1.1 Chế độ offline ASR

Để đánh giá độ trễ (Latency) và hệ số thời gian thực (RTF – Real-Time Factor) trong chế độ nhận dạng tiếng nói ngoại tuyến, sinh viên đã tiến hành thử nghiệm với 7 câu nói có độ dài khác nhau nhằm phản ánh tính đa dạng của dữ liệu đầu vào.

- **Độ trễ (Latency)** được định nghĩa là khoảng thời gian tính từ khi người nói kết thúc phát âm cho đến khi hệ thống trả về toàn bộ kết quả phiên âm tương ứng. Đại lượng này phản ánh mức độ phản hồi của hệ thống và ảnh hưởng trực tiếp đến trải nghiệm người dùng trong các ứng dụng thời gian thực.
- **Hệ số thời gian thực (RTF)** được tính bằng tỷ số giữa thời gian xử lý thực tế của hệ thống và tổng thời lượng tín hiệu âm thanh được đưa vào xử lý. Cụ thể, RTF =
$$\frac{\text{Thời gian xử lý}}{\text{Thời lượng tín hiệu}}$$
.

Bảng 4.20. Đánh giá Latency và RTF cho các câu nói offline (giới hạn 8 giây)

STT	Câu nói	Thời lượng (s)	Latency (s)	RTF	Dự đoán (LM)
1	hôm nay là thứ hai trời rất nắng ngày mai là thứ ba trời có thể mưa dông	6	0.54	0.09	hôm nay là thứ hai trời rất nắng ngày mai là thứ ba trời có thể mưa dông
2	tôi là sinh viên năm tư đại học bách khoa hà nội	4	0.56	0.14	tôi là sinh viên năm tư đại học bách khoa hà nội
3	doanh nghiệp của tôi đang gấp rất nhiều khó khăn bạn có thể giúp tôi được không	5	0.52	0.104	doanh nghiệp của tôi đang khớp rất nhiều khó khăn bạn có thể giúp tôi được không
4	bạn đã từng đến hà nội chưa hà nội là một điểm đến đẹp và thu hút khách du lịch	6	0.56	0.09	bạn đã từng đến hà nội chưa hà nội là một điểm đến đẹp và thu hút khách du lịch
5	tổng thống mỹ mới sang thăm việt nam ngày hôm qua và ông ấy đi du lịch ở việt nam miền bắc	7	0.57	0.08	tổng thống mỹ mới sang thăm việt nam ngày hôm qua và ông ấy đi du lịch ở việt nam miền bắc
6	bạn có điều gì muốn nói không tôi đang muốn nói điều này tôi cần sự giúp đỡ của người khác	8	0.63	0.08	bạn có điều gì muốn nói không tôi đang muốn nói điều này tôi cần sự giúp đỡ của người khác
7	tôi đang làm đồ án tốt nghiệp và hạnh phúc đến từ những điều đơn giản nhất	6	0.52	0.09	tôi đang làm đồ án tốt nghiệp và hạnh phúc đến từ những điều đơn giản nhất

Nhận xét:

Dựa trên bảng 4.22. kết quả đánh giá với 7 câu nói có độ dài khác nhau (tối đa 8 giây), hệ thống nhận dạng tiếng nói đạt độ trễ trung bình (Latency) khoảng **0.56 giây** và hệ số thời gian thực (RTF) trung bình là **0.097**.

Các giá trị RTF nhỏ hơn 1 cho thấy hệ thống xử lý nhanh hơn thời gian thực, đáp ứng yêu cầu về tốc độ trong các ứng dụng offline. Độ trễ tổng thể vẫn đảm bảo trong giới hạn chấp nhận được, đặc biệt trong bối cảnh xử lý câu nói dài . Kết quả cũng cho

thấy tính ổn định của mô hình, khi độ trễ và RTF không thay đổi đáng kể theo độ dài câu nói, chứng tỏ hệ thống có khả năng mở rộng tốt với các đầu vào đa dạng về nội dung và độ dài. Dựa trên bảng kết quả đánh giá Latency và RTF cho các câu nói ở chế độ offline, có thể nhận thấy rằng: Hầu hết các câu dự đoán đều tương đối chính xác, nội dung được hệ thống mô hình nhận dạng gần như khớp hoàn toàn với câu gốc. Điều này thể hiện rõ ràng khả năng nhận dạng tốt của mô hình trong các trường hợp câu dài và ngữ cảnh đa dạng.

Một số lỗi nhỏ xảy ra ở các từ dễ nhầm lẫn âm vị như:

- "năm tư" bị nhận thành "năm từ"
- "sang thăm Việt Nam" bị nhận thành "sang tâm Việt Nam"
- "điều đơn giản nhất" bị nhận thành "biểu đơn giản nhất"

4.12.1.2 Chế độ online ASR

Dựa trên cách triển khai chế độ online ASR sinh viên đã trình bày ở mục 3.4, cách đánh giá RTF, Latency có chút khác biệt với cách đánh giá ở chế độ offline ASR, cụ thể như sau:

- Latency: Thời gian trễ từ lúc phần audio cuối cùng được thu (tức lúc buffer audio được cập nhật) đến khi kết quả nhận dạng tương ứng được in ra (commit).
- RTF: Tỷ lệ thời gian xử lý nhận dạng chia cho thời lượng audio được xử lý trong lần đó.

Dựa vào cách đánh giá này, sinh viên thử nghiệm trực tiếp với giọng của mình 7 trường hợp và kết quả thống kê ở bảng 4.21.. Bảng 4.21. trình bày kết quả thống kê

Bảng 4.21. Thông kê Latency và RTF qua 7 lần đo

Lần	Latency (s)	RTF
1	0.604	0.214
2	0.571	0.152
3	0.586	0.155
4	0.569	0.150
5	0.622	0.163
6	0.622	0.159
7	0.561	0.146
Trung bình	0.5907	0.1627

Latency và Real-Time Factor (RTF) qua 7 lần đo của hệ thống nhận dạng giọng nói online.

- **Về Latency (độ trễ):**

- Giá trị trung bình khoảng **0.59 giây** là khá tốt cho một hệ thống nhận dạng giọng nói online.
- Latency dưới 1 giây thường được xem là đáp ứng tốt cho các ứng dụng tương tác thời gian thực, giúp người dùng cảm thấy phản hồi nhanh, không bị chậm trễ rõ rệt.
- Các lần đo đều khá đồng đều, dao động trong khoảng 0.56 - 0.62 giây, thể hiện sự ổn định của hệ thống trong việc xử lý và trả kết quả.
- Không có lần đo nào có latency vượt quá 0.7 giây, cho thấy hệ thống không bị trễ đột ngột hoặc quá tải trong các lần thử.

- **Về Real-Time Factor (RTF):**

- Giá trị RTF trung bình khoảng **0.16**, nghĩa là hệ thống xử lý nhanh hơn thời gian thực khoảng 6 lần ($1 / 0.16 \approx 6.25$).
- Đây là con số rất tốt, cho thấy mô hình và phần cứng sử dụng có khả năng nhận dạng nhanh, đủ để chạy online realtime.
- Các lần đo RTF dao động từ 0.146 đến 0.214, cũng khá ổn định, không có sự biến động lớn.

4.12.2 Triển khai hệ thống trên ứng dụng Android

Nhằm đánh giá khả năng triển khai thực tế của hệ thống nhận dạng tiếng nói, sinh viên đã tiến hành thử nghiệm mô hình trong môi trường ứng dụng Android với ba thiết bị có cấu hình phần cứng khác nhau, bao gồm:

- **Samsung M10:** sử dụng vi xử lý *Exynos 7870*, RAM 2GB, tốc độ CPU 1.6 GHz.
- **OPPO A16:** sử dụng vi xử lý *MediaTek Helio G35*, RAM 4GB, CPU gồm 4 nhân 2.3 GHz và 4 nhân 1.8 GHz.
- **Vivo X100s Pro:** sử dụng vi xử lý *MediaTek Dimensity 9300+*, RAM 12GB.

4.12.2.1 Chế độ offline ASR trên ứng dụng nhận dạng tiếng nói (Android)

Phương pháp đo lường độ trễ (Latency) và hệ số thời gian thực (RTF) trên các thiết bị Android được thực hiện tương tự như trong chế độ hoạt động ngoại tuyến trên máy tính xách tay, đã được trình bày ở các mục trước. Cụ thể:

- **Latency** được tính từ thời điểm người dùng bắt đầu phát âm đến thời điểm hệ thống hiển thị bản phiên âm hoàn chỉnh trên giao diện ứng dụng.

- **RTF** được xác định bằng tỷ số giữa thời gian xử lý tín hiệu và tổng độ dài tín hiệu âm thanh đầu vào.

Việc triển khai trên nhiều thiết bị có cấu hình khác nhau nhằm đánh giá mức độ tương thích và hiệu năng của hệ thống trong các điều kiện phần cứng thực tế, từ đó làm cơ sở để đề xuất khả năng ứng dụng rộng rãi trên nhiều dòng điện thoại phổ thông hiện nay.

Bảng 4.22. Đánh giá Latency và RTF ở chế độ offline trên các thiết bị Android

Thiết bị	Thời lượng (s)	Độ trễ (s)	RTF
Samsung M10	5	13.73	1.75
OPPO A16	5	12.80	1.56
Vivo X100s Pro	5	6.10	0.22

Nhận xét

Kết quả trong Bảng 4.22. cho thấy hiệu năng xử lý của hệ thống nhận dạng tiếng nói có sự khác biệt rõ rệt giữa các thiết bị Android. Cụ thể, trên các thiết bị có cấu hình thấp như Samsung M10 và OPPO A16, độ trễ tương đối cao (trên 12 giây) và hệ số thời gian thực RTF vượt quá 1, cho thấy hệ thống xử lý chậm hơn thời gian thực và khó đáp ứng yêu cầu trong các ứng dụng cần phản hồi nhanh.

Ngược lại, trên thiết bị Vivo X100s Pro với phần cứng mạnh hơn, độ trễ giảm xuống đáng kể (6.10 giây) và RTF chỉ còn 0.22, cho thấy hệ thống có khả năng xử lý nhanh hơn thời gian thực và hoàn toàn phù hợp để triển khai trong các ứng dụng nhận dạng tiếng nói ngoại tuyến trên nền tảng di động.

4.12.2.2 Chế độ online ASR trên ứng dụng nhận dạng tiếng nói (Android)

Độ trễ (Latency) và hệ số thời gian thực (RTF) trong chế độ nhận dạng tiếng nói trực tuyến được đánh giá trên ba thiết bị Android đã sử dụng trong phần thử nghiệm chế độ offline. Phương pháp đánh giá được thực hiện như sau:

- **Độ trễ (Latency)** được xác định bằng khoảng thời gian kể từ khi người dùng nhấn nút bắt đầu ghi âm trên giao diện ứng dụng cho đến thời điểm hệ thống hiển thị đoạn văn bản đầu tiên được nhận dạng.
- **Hệ số thời gian thực (RTF)** được tính bằng tỷ số giữa thời gian xử lý (tức là thời điểm hệ thống in ra toàn bộ câu phiên âm cuối cùng trừ đi thời điểm kết thúc ghi âm) và tổng thời lượng âm thanh đầu vào (5 giây). Cụ thể, công thức tính RTF được sử dụng là:

$$RTF = \frac{T_{in_cuối} - T_{kết_thúc_ghi}}{T_{ghi}} \quad (4.4)$$

Trong đó:

- $T_{in_cuối}$ là thời điểm hệ thống in ra đoạn văn bản hoàn chỉnh cuối cùng trên màn hình ứng dụng.
- $T_{kết_thúc_ghi}$ là thời điểm kết thúc quá trình ghi âm (tức là sau 5 giây kể từ khi bắt đầu ghi).
- T_{ghi} là tổng thời lượng ghi âm, được cố định là 5 giây trong các thử nghiệm.

Phương pháp này cho phép đánh giá một cách khách quan cả tốc độ phản hồi tức thời của hệ thống và hiệu quả xử lý trong môi trường thời gian thực khi triển khai trên nền tảng thiết bị di động (Android). **Nhận xét** Kết quả trình bày trong Bảng

Bảng 4.23. Đánh giá Latency và RTF ở chế độ online trên các thiết bị Android

Thiết bị	Độ trễ (s)	RTF
Samsung M10	12.50	2.90
OPPO A16	14.00	3.00
Vivo X100s Pro	5.10	0.96

4.23. cho thấy hiệu năng của hệ thống ASR ở chế độ trực tuyến chịu ảnh hưởng rõ rệt bởi cấu hình phần cứng của thiết bị.

Cụ thể, trên các thiết bị có cấu hình trung bình và thấp như Samsung M10 và OPPO A16, độ trễ dao động trong khoảng 12.5 đến 14 giây, trong khi hệ số thời gian thực (RTF) đều vượt ngưỡng 2. Điều này cho thấy thời gian xử lý kéo dài hơn gấp đôi thời lượng âm thanh đầu vào, gây ra độ trễ đáng kể và ảnh hưởng tiêu cực đến trải nghiệm người dùng. Ngược lại, trên thiết bị Vivo X100s Pro có cấu hình cao, độ trễ giảm còn 5.10 giây và RTF chỉ còn 0.96 – gần bằng thời gian thực. Điều này phản ánh khả năng xử lý hiệu quả của hệ thống khi được triển khai trên phần cứng mạnh, đồng thời khẳng định tính khả thi của mô hình trong các ứng dụng thực tế nếu được tối ưu hóa và triển khai đúng môi trường.

4.13 Kết luận chương

Chương 4 đã tổng hợp và phân tích các kết quả thực nghiệm của hệ thống nhận dạng tiếng nói, với những đánh giá chi tiết thông qua các chỉ số chuẩn như WER, RTF và Latency. Kết quả cho thấy mô hình đạt được độ chính xác cao trên nhiều tập dữ liệu khác nhau và có khả năng hoạt động gần thời gian thực. Việc thay đổi số lượng lớp phân loại từ 128 xuống 93 giúp giảm nhẹ độ phức tạp mô hình nhưng không ảnh hưởng đáng kể đến hiệu suất, đồng thời rút ngắn thời gian suy luận. Những phân tích này là cơ sở quan trọng để rút ra các nhận xét, so sánh và định hướng cải tiến hệ thống trong các nghiên cứu tiếp theo.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Kết luận

Trong khuôn khổ đề tài, sinh viên đã thiết thành công một ứng dụng soạn thảo văn bản tiếng Việt tự động trên smartphone sử dụng công nghệ nhận dạng tiếng nói. Hệ hoạt động hiệu quả ở cả hai chế độ: **offline** và **online**.

Hạn chế của hệ thống

Mặc dù hệ thống nhận dạng tiếng nói tiếng Việt đã đạt được nhiều kết quả khả quan cả về độ chính xác và khả năng triển khai thực tế, tuy nhiên vẫn còn tồn tại một số hạn chế như sau:

- **Xử lý từ láy, từ tượng thanh còn chưa hiệu quả:** Các từ láy như “lập lòe”, “lúng túng”, hay từ tượng thanh như “a a a”, “hú hú” xuất hiện ít trong tập huấn luyện nên mô hình gặp khó khăn trong việc nhận dạng chính xác.
- **Chưa nhận dạng tốt khi người dùng nói nhanh hoặc nuốt âm:** Khi người nói phát âm nhanh, mô hình thường bị mất thông tin, dẫn đến việc nhận dạng thiếu từ hoặc sai ngữ cảnh. Các hiện tượng như nói lướt, nói nỗi âm chưa được xử lý hiệu quả.
- **Chưa xử lý tốt trong môi trường nhiễu lớn:** Hệ thống hiện mới chỉ đạt kết quả tốt trong môi trường yên tĩnh hoặc nhiễu nhẹ.

Hướng phát triển

Trong tương lai, để nâng cao chất lượng và khả năng ứng dụng thực tế của hệ thống nhận dạng tiếng nói tiếng Việt, sinh viên đề xuất một số hướng phát triển như sau:

- **Tăng cường bộ cơ sở dữ liệu huấn luyện:** Tiến hành thu thập thêm dữ liệu tiếng nói từ người dùng thực tế, đặc biệt tập trung vào các cụm từ, từ láy, và câu nói nhanh mà mô hình hiện tại còn gặp khó khăn. Việc mở rộng và đa dạng hóa dữ liệu sẽ giúp mô hình học được các đặc trưng ngôn ngữ phức tạp và cải thiện độ tổng quát.
- **Bổ sung dữ liệu trong môi trường nhiễu và tốc độ nói cao:** Huấn luyện thêm với các dữ liệu có tiếng ồn ở nhiều mức độ khác nhau, đồng thời đưa vào các đoạn âm thanh có tốc độ nói nhanh, giúp mô hình thích nghi tốt hơn trong môi trường thực tế. Ngoài ra, tích hợp thêm các kỹ thuật lọc nhiễu trước khi suy luận sẽ góp phần cải thiện đáng kể chất lượng nhận dạng.
- **Ứng dụng các kỹ thuật tăng cường dữ liệu (data augmentation):** Áp dụng các phương pháp như thay đổi tốc độ, thêm tiếng ồn, biến đổi cao độ,... để

nhân rộng dữ liệu đầu vào, giúp mô hình học được các biến thiên phong phú trong lời nói.

TÀI LIỆU THAM KHẢO

- [1] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*, 3rd ed., 2025, online draft, released January 12, 2025. [Online]. Available: <https://web.stanford.edu/~jurafsky/slp3/>
- [2] A. E. Team, “Top 20 tools for android development,” AltexSoft blog, May 2018, truy cập ngày [ngày hôm nay], từ AltexSoft.
- [3] Viện Nghiên cứu Dữ liệu lớn VinBigdata, “VinBigdata chia sẻ 100 giờ dữ liệu tiếng nói cho cộng đồng,” <https://institute.vinbigdata.org/events/vinbigdata-chia-se-100-gio-du-lieu-tieng-noi-cho-cong-dong/>, Oct. 2020, truy cập ngày ...
- [4] T. B. Nguyen, “Vietnamese end-to-end speech recognition using wav2vec 2.0,” 09 2021. [Online]. Available: <https://github.com/vietai/ASR>
- [5] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, “Robust speech recognition via large-scale weak supervision,” 2022. [Online]. Available: <https://arxiv.org/abs/2212.04356>
- [6] T.-T. Le, L. T. Nguyen, and D. Q. Nguyen, “Phowhisper: Automatic speech recognition for vietnamese,” *arXiv preprint arXiv:2406.02555*, 2024.
- [7] S. Kim, A. Gholami, A. Shaw, N. Lee, K. Mangalam, J. Malik, M. W. Mahoney, and K. Keutzer, “Squeezeformer: An efficient transformer for automatic speech recognition,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 9361–9373, 2022.
- [8] M. Anusuya and S. K. Katti, “Speech recognition by machine, a review,” *arXiv preprint arXiv:1001.2267*, 2010.
- [9] N. Thanh, P. Dung, N. Hay, N.-B. Le, and D. Xuan-Quy, “ĐÁnh giÁ cÁc hÊ thÔng nhÂn dÂng giÓng nÓi tiÊng viÊt (vais, viettel, zalo, fpt vÀ google) trong bÃn tin-tạp chí khoa học giáo dục kỹ thuật,” vol. 63, pp. 28–36, 02 2021.
- [10] AIMultiple, “Voice recognition applications: 10 use cases in 2023,” 2023, accessed: 2025-06-21. [Online]. Available: <https://research.aimultiple.com/voice-recognition-applications/>
- [11] A. Mehrish, N. Majumder, R. Bhardwaj, R. Mihalcea, and S. Poria, “A review of deep learning techniques for speech processing,” 2023. [Online]. Available: <https://arxiv.org/abs/2305.00359>

- [12] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [13] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.
- [14] R. K. Srivastava, K. Greff, and J. Schmidhuber, “Highway networks,” *arXiv preprint arXiv:1505.00387*, 2015.
- [15] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” 2015. [Online]. Available: <https://arxiv.org/abs/1512.03385>
- [16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.
- [17] Y. Kim, C. Denton, L. Hoang, and A. M. Rush, “Structured attention networks,” *arXiv preprint arXiv:1702.00887*, 2017.
- [18] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, “A novel connectionist system for unconstrained handwriting recognition,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 5, pp. 855–868, 2008.
- [19] H. Scheidl, S. Fiel, and R. Sablatnig, “Word beam search: A connectionist temporal classification decoding algorithm,” in *2018 16th International conference on frontiers in handwriting recognition (ICFHR)*. IEEE, 2018, pp. 253–258.
- [20] A. Loubser, P. De Villiers, and A. De Freitas, “End-to-end automated speech recognition using a character based small scale transformer architecture,” *Expert Systems with Applications*, vol. 252, p. 124119, 2024.
- [21] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, “wav2vec 2.0: A framework for self-supervised learning of speech representations,” 2020. [Online]. Available: <https://arxiv.org/abs/2006.11477>
- [22] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and R. Pang, “Conformer: Convolution-augmented transformer for speech recognition,” 2020. [Online]. Available: <https://arxiv.org/abs/2005.08100>
- [23] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” 2015. [Online]. Available: <https://arxiv.org/abs/1505.04597>

- [24] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” in *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, 2019, pp. 4171–4186.
- [25] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, “An image is worth 16x16 words: Transformers for image recognition at scale,” *arXiv preprint arXiv:2010.11929*, 2020.
- [26] Android Developers, “Giới thiệu về android studio,” <https://developer.android.com/studio/intro?hl=vi>, 2024, truy cập ngày 22 tháng 6 năm 2025. [Online]. Available: <https://developer.android.com/studio/intro?hl=vi>
- [27] Google Developers, “Nhận diện đối tượng bằng tensorflow lite trên android,” <https://developers.google.com/codelabs/tflite-object-detection-android?hl=vi#4>, 2024, truy cập ngày 22 tháng 6 năm 2025. [Online]. Available: <https://developers.google.com/codelabs/tflite-object-detection-android?hl=vi#4>
- [28] D. Macháček, R. Dabre, and O. Bojar, “Turning whisper into real-time transcription system,” *arXiv preprint arXiv:2307.14743*, 2023.
- [29] H.-T. Luong and H.-Q. Vu, “A non-expert kaldí recipe for vietnamese speech recognition system,” in *Proceedings of the Third International Workshop on Worldwide Language Service Infrastructure and Second Workshop on Open Infrastructures and Analysis Frameworks for Human Language Technologies (WLSI/OIAF4HLT2016)*, 2016, pp. 51–55.
- [30] K. Le, T. V. Ho, D. Tran, and D. T. Chau, “Chunkformer: Masked chunking conformer for long-form speech transcription,” in *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2025, pp. 1–5.
- [31] T. B. Nguyen, “Vietnamese end-to-end speech recognition using wav2vec 2.0,” *Zenodo*, 2021.
- [32] X. Chen and J. Sheng, “The design of detector to illegal ingredients of kitchen waste based on embedded device,” *Journal of Physics: Conference Series*, vol. 1802, p. 032009, 03 2021.

PHỤ LỤC

Phụ lục 1: Thư viện KenLM

Giới thiệu

KenLM là bộ công cụ mô hình hóa ngôn ngữ được Kenneth Heafield phát triển năm 2011. Đây là phần mềm nguồn mở được thiết kế để xây dựng và truy vấn hiệu quả các mô hình ngôn ngữ, đặc biệt là cho các ứng dụng "Xử lý ngôn ngữ tự nhiên (NLP) như nhận dạng tiếng nói, dịch máy và truy xuất thông tin.

Cách đánh giá KenLM

Các siêu tham số kenlm:

- n: n trong n-gram
- pruning: quyết định cắt tỉa KenLM thông qua việc cắt tỉa, quy trình đạo tạo sẽ loại bỏ các nhánh ít có khả năng xảy ra. Ví dụ, trong mô hình 3-gram, nếu đặt cắt tỉa thành 0 0 1, tức là đang hướng dẫn mô hình chỉ giữ lại xác suất của các tổ hợp 3 từ đã được nhìn thấy ít nhất một lần trong dữ liệu đào tạo.

Kích thước language model có mối tương quan nghịch đảo với tốc độ của nó. Điều này là do trong các mô hình lớn hơn, ta thường sử dụng vốn từ vựng lớn hơn, n lớn hơn. Nhận xét: Mô hình lớn hơn không phải lúc nào cũng hiệu quả hơn mô hình nhỏ hơn. Vì lý do cũng cần cân bằng giữa tốc độ và độ chính xác, mô hình 5-gram thường đem lại kết quả tốt hơn không đáng kể trong dự án này nhưng thời gian suy luận lại tăng lên đáng kể.

Phụ lục 2: Vietnamese Corpus Text

Ngữ liệu văn bản được lấy từ các nguồn sau:

- Văn bản lấy trong tập dữ liệu VLSP 2020, FPT, VIVOS
- Truyện tranh cổ tích
- Báo lấy từ các trang web: báo nhân dân, báo tuổi trẻ, báo 24h
- Các đoạn hội thoại lấy từ Internet

Một số link bài báo, truyện tham khảo (có thể click vào):

- [Báo nhân dân](#)
- [Truyện cổ tích, truyện tranh](#)
- [Các đoạn hội thoại](#)

Phụ lục 3: Ngôn ngữ Kotlin

Kotlin là một ngôn ngữ lập trình hiện đại và là xu hướng phổ biến hiện nay được xuất bản vào năm 2016 bởi JetBrains. Nó đã trở nên rất phổ biến vì nó tương thích với Java (một trong những ngôn ngữ lập trình phổ biến nhất hiện có), nghĩa là mã Java có thể được sử dụng trong chương trình Kotlin.

Phụ lục 4: Fine-tuning

Fine-tuning là một kỹ thuật phổ biến trong học sâu (deep learning), trong đó một mô hình đã được huấn luyện trước (pre-trained model) trên một tập dữ liệu lớn được điều chỉnh và huấn luyện thêm trên một tập dữ liệu mục tiêu nhỏ hơn. Mục tiêu chính là tận dụng kiến thức đã học được từ mô hình ban đầu nhằm cải thiện hiệu suất trên tác vụ mới.

Quy trình fine-tuning thường bao gồm các bước sau:

- **Huấn luyện mô hình nguồn (source model):** Trước tiên, một mô hình mạng nơ-ron sâu được huấn luyện trên một tập dữ liệu lớn và đa dạng, gọi là tập dữ liệu nguồn. Mô hình này học được các đặc trưng khái quát có khả năng áp dụng cho nhiều tác vụ khác nhau.
- **Khởi tạo mô hình mục tiêu (target model):** Một mô hình mới được xây dựng bằng cách sao chép kiến trúc và các tham số (weights) từ mô hình nguồn, ngoại trừ lớp đầu ra. Việc loại bỏ lớp đầu ra là cần thiết vì lớp này phụ thuộc mạnh vào số lượng và loại nhãn của bài toán ban đầu.
- **Giả định:** Các tham số của mô hình nguồn chứa đựng kiến thức tổng quát hữu ích, có thể được tái sử dụng cho bài toán mục tiêu.
- **Thay thế lớp đầu ra:** Một lớp đầu ra mới được thêm vào mô hình mục tiêu với số lượng đơn vị đầu ra tương ứng với số nhãn của bài toán mới. Lớp này được khởi tạo ngẫu nhiên, do không thể kế thừa trực tiếp từ bài toán ban đầu.
- **Fine-tuning mô hình:** Mô hình mục tiêu được huấn luyện trên tập dữ liệu mục tiêu. Trong quá trình này, lớp đầu ra được huấn luyện từ đầu, trong khi các lớp còn lại được tinh chỉnh nhẹ (fine-tuned) dựa trên các tham số đã học từ mô hình nguồn.

Phụ lục 5: Biến đổi Fourier

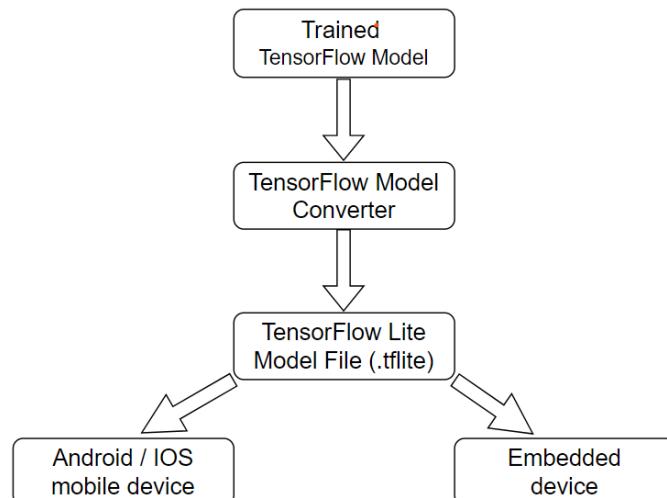
Tín hiệu âm thanh bao gồm một số sóng âm thanh đơn. Khi lấy các mẫu tín hiệu theo thời gian, chúng ta chỉ nắm bắt được biên độ kết quả. Biến đổi Fourier là một công thức toán học cho phép chúng ta phân tách tín hiệu thành tần số riêng lẻ và biên độ của tần số. Nói cách khác, nó chuyển đổi tín hiệu từ miền thời gian thành miền tần số. Kết quả được gọi là phổ (spectrum). Điều này là có thể bởi vì mọi tín hiệu có thể được phân tách thành một tập hợp các sóng hình sin và cosin cộng vào tín hiệu ban đầu.

Phụ lục 6: Tensorflow Lite

Tensorflow lite là một giải pháp học máy nhẹ do Google giới thiệu, chủ yếu hướng đến các thiết bị di động và các thiết bị nhúng. Dựa trên các đặc điểm của thiết bị di động, Tensorflow lite sử dụng nhiều công nghệ để tùy chỉnh và tối ưu hóa lỗi. Tensorflow lite có các đặc điểm sau:

- **Đa nền tảng:** mã lỗi được viết bằng C++ và có thể chạy trên nhiều nền tảng khác nhau;
- **Nhỏ:** Kích thước mã nhỏ, kích thước tệp mô hình nhỏ và bộ nhớ chạy thấp
- Hỗ trợ tăng tốc phần cứng [32].

Cấu trúc khung của Tensorflow được thể hiện như hình 4.23. bên dưới.



Hình 4.23. Cấu trúc khung của TensorFlow lite