

DẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC - KỸ THUẬT MÁY TÍNH

LUẬN VĂN TỐT NGHIỆP ĐẠI HỌC



**Giảm nhiễu tiếng ồn
bằng học sâu**

Ngành: Khoa học máy tính

HỘI ĐỒNG: Khoa học máy tính 2
GVHD: TS. Nguyễn An Khương
TS. Nguyễn Tiên Thịnh
KS. Nguyễn Tân Đức
KS. Nguyễn Hữu Hồng Huy
ThS. Mai Xuân Toàn

—o0o—

SVTH: Nguyễn Long Vũ (1814816)

Thành phố Hồ Chí Minh, 6/2022

Lời cam đoan

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi dưới sự hướng dẫn của TS. Nguyễn An Khương, TS. Nguyễn Tiến Thịnh, KS. Nguyễn Tấn Đức, KS. Nguyễn Hữu Hồng Huy. Nội dung nghiên cứu và các kết quả đều là trung thực và chưa từng được công bố trước đây. Các số liệu được sử dụng cho quá trình phân tích, nhận xét được chính tôi thu thập từ nhiều nguồn khác nhau và sẽ được ghi rõ trong phần tài liệu tham khảo. Ngoài ra, tôi cũng có sử dụng một số nhận xét, đánh giá và số liệu của các tác giả khác, cơ quan tổ chức khác. Tất cả đều có trích dẫn và chú thích nguồn gốc. Nếu phát hiện có bất kỳ sự gian lận nào, tôi xin hoàn toàn chịu trách nhiệm về nội dung luận văn tốt nghiệp của mình. Trường đại học Bách Khoa thành phố Hồ Chí Minh không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện.

KÝ TÊN: _____ NGÀY: _____

Lời cảm ơn

Trong suốt thời gian học tập và rèn luyện tại Trường Đại học Bách Khoa Thành phố Hồ Chí Minh đến nay, tôi đã nhận được rất nhiều sự quan tâm, giúp đỡ của quý thầy cô và bạn bè. Với lòng biết ơn sâu sắc và chân thành nhất, tôi xin gửi đến quý thầy cô ở Khoa Khoa Học và Kỹ Thuật Máy Tính - Trường Đại học Bách Khoa Thành phố Hồ Chí Minh, đã cùng với tri thức và tâm huyết của mình để truyền đạt vốn kiến thức quý báu cho tôi trong suốt thời gian học tập tại trường. Đặc biệt tôi xin gửi lời cảm ơn chân thành đến thầy Nguyễn An Khương. Người thầy đã tận tâm hướng dẫn, theo dõi và hỗ trợ tôi trong suốt quá trình thực hiện luận văn tốt nghiệp. Ngoài những lời khuyên và kiến thức về chuyên môn, học thuật đầy kinh nghiệm của thầy, trong quá trình làm việc cùng thầy một thời gian dài tôi còn học được những đức tính tốt, những kỹ năng cần thiết để trở một người làm khoa học thật thụ như khả năng tư duy phản biện, tư duy sáng tạo, sự cẩn cù, sự trung thực và sự cẩn thận chính xác.

Bên cạnh đó, tôi xin gửi cảm ơn đến thầy Nguyễn Tiến Thịnh, anh Nguyễn Tấn Đức và anh Nguyễn Hữu Hồng Huy đã cùng tham gia hướng dẫn, hỗ trợ tôi thực hiện luận văn tốt nghiệp đề tài “Giảm nhiễu bằng học sâu” suốt thời gian vừa qua. Những kinh nghiệm, kiến thức về đại số, xử lý dữ liệu, những điều cơ bản nhất về trí tuệ nhân tạo và học máy mà tôi có được từ các thầy và các anh trong quá trình nghiên cứu này đã giúp tôi trang bị cho mình những điều cần thiết để hoàn thành luận văn này. Tôi cũng xin gửi lời cảm ơn của mình tới anh Nguyễn Hải Triều Anh và anh Cao Văn Nghĩa vì những lời góp ý sâu sắc các anh dành cho tôi.

Sau cùng, tôi muốn dành những tình cảm sâu sắc trân trọng nhất gửi đến ba mẹ tôi, những người đã hi sinh rất nhiều vì tôi, lo lắng mọi thứ cho tương lai của tôi, tạo cho tôi mọi cơ hội học tập ở những môi trường tốt nhất. Ba mẹ luôn là nguồn động lực to lớn thôi thúc tôi vượt qua những rào cản của bản thân mà tiến về phía trước. Con cảm ơn ba mẹ rất nhiều.

Tóm tắt đề tài

Từ lâu, các hoạt động trực tuyến đã trở nên phổ biến, đặc biệt là trong tình hình dịch bệnh hiện nay. Nhưng hiệu quả của các hoạt động trực tuyến rất dễ bị ảnh hưởng bởi tiếng ồn từ môi trường xung quanh người hoạt động. Để hạn chế điều này, mọi người thường sử dụng các sản phẩm phần mềm được thương mại hóa và các tai nghe lọc nhiễu đắt tiền. Nhưng không phải ai cũng có thể tiếp cận được những sản phẩm này. Do đó, trong đề tài này, chúng tôi đặt ra vấn đề giải quyết tiếng ồn lẫn bên trong âm thanh, loại bỏ chúng trong khi vẫn phải đảm bảo chất lượng giọng nói đầu ra. Bằng cách sử dụng học sâu, chúng tôi mong muốn có thể loại bỏ được phần nào những khó khăn trong hoạt động nghe nhìn trực tuyến của mọi người.

Để có thể lọc được nhiễu ra khỏi giọng nói, chúng tôi tiến hành tham khảo một số mô hình đạt giải cao trong cuộc thi Deep Noise Suppression (DNS) [14] của Microsoft. Dựa trên các đặc điểm của mô hình và dữ liệu mà chúng tôi quan sát được, chúng tôi đề xuất ra hàm tăng cường măt măt và mô hình trong luận văn của mình. Mô hình này sau đó được huấn luyện trên hàm măt măt tăng cường của chúng tôi và sau đó được sử dụng vào ứng dụng. Chúng tôi so sánh kết quả độ đo của mô hình với các mô hình khác, so sánh sự ổn định của mô hình với các ngôn ngữ khác nhau và cả các ứng dụng đã được đi vào sử dụng cả mă nguồn mở và thương mại. Mô hình này sau đó được sử dụng vào ứng dụng của chúng tôi và đáp ứng được kì vọng về độ trễ cũng như chất lượng của giọng nói đầu ra.

Với các công trình nghiên cứu và ứng dụng được thực hiện trong luận văn này, chúng tôi mong muốn góp một phần công sức của mình vào việc cải thiện chất lượng học tập, vui chơi trực tuyến của mọi người.

Mục lục

Lời cam đoan	ii
Lời cảm ơn	iii
Tóm tắt đề tài	iv
Mục lục	v
Danh sách hình vẽ	ix
Danh sách bảng	xv
1 Giới thiệu	1
1.1 Âm thanh	1
1.2 Đặt vấn đề	4
1.3 Mục tiêu và nhiệm vụ đề tài	7
1.4 Kết quả mong đợi	8
1.5 Ý nghĩa của đề tài	9
1.6 Cấu trúc luận văn	9
2 Xử lý tín hiệu số	11
2.1 Phép biến đổi Fourier thời gian ngắn	11
2.1.1 Định nghĩa	11
2.1.2 Hàm cửa sổ	13
2.1.3 Phương pháp Overlap Add	15
2.2 Biến đổi Fourier rời rạc	22
2.2.1 Định nghĩa	22

2.2.2	Biến đổi Fourier nhanh	23
3	Âm học	31
3.1	Âm thanh	31
3.1.1	Sự hình thành sóng âm	31
3.1.2	Các biểu diễn của âm thanh	33
3.1.3	Dãy khoảng tám và dãy khoảng tám lẻ	36
3.1.4	Tần số cảm nhận	37
3.1.5	Thang đo độ to của âm thanh	39
3.2	Dánh giá chất lượng âm thanh	45
3.2.1	Các nhân tố ảnh hưởng tới chất lượng âm thanh	45
3.2.2	Thang đo Mean Opinion Score	46
3.2.3	Short-time Objective Intelligibility Measurement	48
3.2.4	Perceptual Evaluation of Speech Quality	50
3.2.5	A Non-Intrusive Perceptual Objective Speech Quality Metric	54
4	Học sâu	57
4.1	Mạng hồi quy	57
4.2	Long short term memory	61
5	Các công trình nghiên cứu liên quan	65
5.1	Cách tiếp cận cổ điển	65
5.2	Tiếp cận theo bằng học sâu	67
5.2.1	Mask số thực trên miền tần số thời gian	69
5.2.2	Mask số phức trên miền tần số thời gian	72
5.3	Đề xuất hàm mắt mát và mô hình	75
5.3.1	Hàm mắt mát đề xuất	76
5.3.2	Mô hình đề xuất	85
6	Phân tích thiết kế hệ thống	95
6.1	Yêu cầu hệ thống	95
6.2	Hệ thống lọc nhiễu tĩnh	97
6.3	Hệ thống lọc nhiễu động	103
7	Thực nghiệm và kết quả	115

7.1	Chuẩn bị dữ liệu	115
7.2	Một số kết quả của mô hình lọc nhiễu	121
7.3	Một số kết quả của ứng dụng lọc nhiễu	133
8	Tổng kết	135
8.1	Kết quả đạt được	135
8.2	Hạn chế và hướng phát triển	136
Tài liệu tham khảo		137
A Biến đổi Fourier		143
A.1	Phép biến đổi Fourier	143
A.1.1	Định nghĩa và ví dụ	143
A.1.2	Bản chất của biến đổi Fourier	145
A.1.3	Phép biến đổi Fourier ngược	150
A.2	Biến đổi Fourier rời rạc	151

Danh sách hình vẽ

1.1	Sự truyền âm trong không khí, C (R) là nơi mà mật độ các hạt không khí dày (thưa) tương ứng	2
1.2	Một hệ dao động điều hòa dưới tác dụng của của trọng lực và lực kéo từ lò xo có độ cứng là K được kéo ra một đoạn ban đầu x_0 so với vị trí cân bằng	3
1.3	Hiện tượng Room Impulse Response (tiếng dội) xảy ra khi âm thanh gốc bị suy giảm một lượng tuyến tính với thời gian truyền đi và lại bị ghi nhận lại sau một khoảng thời gian nhất định	4
1.4	Sáu loại nhiễu phổ biến nhất theo khảo sát được chúng tôi thực hiện	6
2.1	Tác dụng của cửa sổ trong biến đổi Fourier thời gian ngắn	12
2.2	Quá trình của số hóa tín hiệu thời gian ngắn trong biến đổi Fourier thời gian ngắn	12
2.3	Spectrogram biểu diễn với chiều dài cửa sổ T và khoảng cách giữa các tín hiệu τ , được kí hiệu trong hình là (T, τ)	13
2.4	Đồ thị ba loại cửa sổ Hann, tam giác và đều	15
2.5	spectrogram của dữ liệu được cửa sổ hóa bởi ba loại cửa sổ Hann, tam giác và đều, tác dụng của cửa sổ lên spectrogram được thể hiện ở vùng khoanh đỏ.	16
2.6	Sơ đồ ý tưởng đệ quy của biến đổi Fourier nhanh (chia theo thời gian)	25
2.7	Nguyên nhân của hiện tượng đảo bit trong biến đổi Fourier nhanh cơ số 2 (chia theo thời gian), với các bit được sắp xếp dần thể hiện qua các số được khoanh trong ô màu đỏ	26
2.8	Sơ đồ ý tưởng đệ quy của biến đổi Fourier nhanh (chia theo tần số)	28

3.1	Biểu đồ sóng âm thể hiện mật độ không khí thay đổi như thế nào trong quá trình truyền âm thanh	32
3.2	Cấu tạo của tai người	33
3.3	Waveform của câu “eat your raisins out-doors on the porch steps” .	34
3.4	Spectrum minh họa biểu diễn âm thanh trên miền tần số	35
3.5	Spectrogram minh họa biểu diễn âm thanh trên miền tần số thời gian	35
3.6	Biểu đồ biểu diễn tần số Mel theo tần số Hz	38
3.7	Mel filterbank	38
3.8	Tần số Bark và Bark filterbank	39
3.9	Đường mức cùng độ to trong môi trường định hướng	40
3.10	Sự biến thiên độ to theo tần số trong môi trường mở	41
3.11	Biểu đồ biến đổi của tăng giảm cường độ để âm nghe được tăng xấp xỉ 2 lần âm ban đầu	42
3.12	Quá trình định lượng tín hiệu	46
3.13	Mô hình khôi của PESQ bao gồm có bốn giai đoạn: <i>tiền xử lý</i> (qua một bộ lọc tiêu chuẩn để quy chuẩn âm), <i>căn chỉnh âm thanh</i> (do sự truyền đi trong môi trường mạng, các gói tin sẽ mất một thời gian mới đến đích, việc căn chỉnh là để tránh sự chênh lệch này làm ảnh hưởng tới việc đánh giá chất lượng), <i>biến đổi âm</i> (Bark Spectrogram được tính toán và chuẩn hóa ở bước này) và <i>tính toán độ giàn đoạn</i>	51
3.14	Liên hệ giữa độ kích ứng và Bark, bên trái thể hiện Bark Spectrum, ở giữa thể hiện cho độ kích ứng được thử nghiệm ra, cuối cùng là độ to được tính toán dựa trên tích phân của độ kích ứng theo từng Bark	53
3.15	Khảo sát được tham khảo từ ITU-T P.808	55
4.1	Mô hình của một neural (toán học) mô phỏng lại cấu tạo của neural (sinh học) bên trong não bộ con người	58
4.2	Lớp FC với 7 đầu vào và 5 đầu ra	59
4.3	Sơ đồ khôi của hai mạng neural	59
4.4	Quá trình duỗi mạng RNN theo thời gian	60
4.5	Cấu tạo của một neural trong RNN, để đảm bảo tính tương đồng với LSTM, chúng tôi sử dụng gạch chân <i>c</i> cho các trọng số trong RNN	60
4.6	Cấu tạo của RNN có phần nhớ	62

4.7	Mô hình LSTM tiêu chuẩn và mở rộng	64
5.1	Quá trình lọc âm nhiễu theo hướng tiếp cận cổ điển	68
5.2	Kiến trúc mạng Dual signal Transformation LSTM	70
5.3	Ở miền tần số thời gian, “mask” mà mô hình học được có khả năng xác định những phần nhiễu không liên quan tới giọng nói, ở bước này, chủ yếu mô hình học cách để loại bỏ các nhiễu mạnh làm cho giọng nói trong hơn bằng cách tổng hợp các sóng cosine lại với nhau thông qua biến đổi Fourier nghịch	71
5.4	Ở bước này, mô hình học cách biểu diễn âm thanh đã được lọc qua ở phần trên sang một miền mới và tách các nhiễu ở đó, kì vọng với cách tiếp cận này, mô hình có thể loại bỏ hầu hết các nhiễu bên trong âm thanh mà làm mất mát ít thông tin nhất	71
5.5	Kiến trúc của mô hình Fullband and Subband Fusion Model	73
5.6	Kiến trúc của mô hình Subband	75
5.7	Ý tưởng hình thành hàm tăng cường mất mát	80
5.8	Nhiều vẫn còn tồn tại bên trong spectrogram âm sạch mà không bị lọc đi hoàn toàn (phần được khoanh trong hình chữ nhật màu trắng)	80
5.9	Một số kết quả được kết quả huấn luyện của mô hình DTLN thu gọn bị cắt một nửa tần số, nhiều vẫn còn khá nhiều trong spectrogram sau khi đã lọc	87
5.10	Một số kết quả được kết quả huấn luyện của mô hình Post, kết quả được cải thiện đáng kể so với mô hình DTLN thu gọn	87
5.11	Một số kết quả được kết quả huấn luyện của mô hình kết hợp giữa DTLN thu gọn và Post	88
5.12	Tần số thay đổi dần dần theo thời gian, tuy biên độ có lúc mạnh lúc nhẹ nhưng tần số vẫn thay đổi đều đặn không bị đột ngột cho đến khi kết thúc	89
5.13	Ý tưởng sử dụng lookahead để dự đoán kết quả hiện tại của chúng tôi	90
5.14	Minh họa tính chất của tích chập khi sử dụng kernel 2 lên kết quả tích chập của kernel 1 lên hình đầu vào sẽ cho phép phát hiện ra nhiều khuôn mẫu tương tự nhau	92
6.1	Kiến trúc của bộ quản lý trạng thái	100

6.2	Trạng thái ban đầu của tác vụ lọc nhiễu (dữ liệu được tải lên thẻ hiện dưới dạng spectrogram trong khung màu đỏ)	101
6.3	Chuỗi các hành động được tạo ra bởi sự kiện lọc nhiễu	101
6.4	Tác vụ đang trong quá trình xử lý, phần bị khóa (khung màu xanh) khóa các tương tác giữa ứng dụng với người dùng lại. Trạng thái (khung màu đỏ) hiển thị về cho người dùng thẻ hiện trạng thái của tác vụ mới nhất (tác vụ lọc nhiễu đang thực thi)	102
6.5	Tác vụ hoàn thành, kết quả lọc (khung màu xanh dương) được hiển thị ra cho người dùng. Trạng thái (khung màu đỏ) được cập nhật và tác vụ hủy đăng ký với bộ quản lý. Các vùng bị khóa (khung màu xanh lá) được phép tiếp tục tương tác với người dùng	102
6.6	Luồng di chuyển của dữ liệu trong ứng dụng lọc nhiễu động	104
6.7	Cấu tạo của microphone (phần mềm), microphone thật truyền dữ liệu vào hàng đợi (hình màu xanh dương) này thông qua write head (mũi tên màu xanh) và ứng dụng đọc dữ liệu được ghi vào thông qua read head (mũi tên cam)	106
6.8	Cấu tạo của microphone (phần mềm), thông qua cơ chế truy xuất vùng nhớ trực tiếp (DMA) cho phép giảm thiểu thời gian truyền tải dữ liệu âm thanh với độ trễ thấp	107
6.9	Tổng độ trễ xảy ra trong hệ thống lọc nhiễu động, tổng độ trễ bao gồm độ trễ của microphone thật (vùng màu xám), độ trễ của đường ống lọc nhiễu (vùng màu xanh lá) và độ trễ của microphone ảo (vùng màu cam)	107
6.10	Luồng di chuyển của dữ liệu trong phần Input	108
6.11	Hàng đợi đầu vào của Input	109
6.12	Luồng di chuyển của dữ liệu trong phần Inference	110
6.13	Luồng di chuyển của dữ liệu trong phần Output	110
6.14	Sơ đồ hoạt động của phần Output	111
6.15	Lưu đồ hoạt động của đường ống lọc nhiễu động	113
7.1	Kết quả spectrogram của việc trộn nhiễu tương quan	116
7.2	Kết quả waveform của việc trộn nhiễu tương quan	116
7.3	Kết quả spectrogram của việc trộn nhiễu không tương quan	117
7.4	Kết quả waveform của việc trộn nhiễu không tương quan	117

7.5	Luôn tồn tại khoảng cách từ miệng người nói tới microphone	119
7.6	Một số hình ảnh của ứng dụng lọc nhiễu tĩnh	133
7.7	Một số hình ảnh của ứng dụng lọc nhiễu động	133
A.1	Đồ thị hàm $f(t) = \sin(t) + \sin(2t) + \sin(3t)$	144
A.2	Đồ thị hàm $f(t)$ sau khi nhân với $e^{-\omega t i}$, điểm đó là khối tâm của vật thể này, các giá trị trả về lúc này đều là số phức vì vậy không gian của chúng tôi sử dụng không còn là Oft nữa mà là $O\Re\Im$	145
A.3	Đồ thị biên độ của trọng tâm (spectrum) trả về theo ω	146
A.4	Đồ thị theo thời gian của f_s và đồ thị theo tần số của F_s	152

Danh sách bảng

1.1	Giới hạn phạm vi đề tài	3
1.2	Các âm nhiễu được chọn từ các loại nhiễu phổ biến	7
1.3	Yêu cầu về cấu hình cần thiết để chạy ứng dụng	8
2.1	Bảng đề xuất cửa sổ sử dụng trong một số trường hợp	15
2.2	Bảng định nghĩa các hàm cửa sổ	17
3.1	Bảng thang đo Mean Opinion Score (MOS)	47
4.1	Các hàm mât mát thường được sử dụng trong mô hình hồi quy . .	61
5.1	Bảng các tiêu chí để thiết kế mạng học sâu	86
5.2	Cấu trúc của mô hình Post sơ khởi	88
5.3	Cấu trúc của mô hình Post sử dụng lookahead	90
5.4	Cấu trúc của bộ nén dữ liệu ở cài tiến thứ 3	92
5.5	Cấu trúc của mô hình Post sử dụng bộ nén dữ liệu cài tiến . .	93
5.6	Cấu trúc của mô hình lọc nhiễu tĩnh	93
5.7	Cấu trúc của mô hình lọc nhiễu động	94
6.1	Mô tả yêu cầu về mặt dữ liệu của hệ thống	96
6.2	Yêu cầu phi chức năng chung đối với cả hai hệ thống	96
6.3	Yêu cầu phi chức năng riêng biệt cho từng hệ thống	96
6.4	Yêu cầu chức năng đối với hệ thống	97
6.5	Usecase load file âm thanh của hệ thống lọc nhiễu tĩnh	97
6.6	Usecase ghi âm của hệ thống lọc nhiễu tĩnh	98
6.7	Usecase xuất file âm thanh của hệ thống lọc nhiễu tĩnh	98
6.8	Usecase lọc nhiễu của hệ thống lọc nhiễu tĩnh	99

6.9	Usecase điều khiển hệ thống lọc nhiễu động của hệ thống lọc nhiễu tĩnh	99
6.10	Usecase chuyển đổi bộ lọc của hệ thống lọc nhiễu động	103
6.11	Các mã lệnh cố định và phạm vi định nghĩa bổ sung của mã điều khiển dịch vụ	105
7.1	Các tập dữ liệu tiếng Anh được sử dụng trong luận văn	118
7.2	Các tập dữ liệu tiếng Việt được sử dụng trong luận văn	119
7.3	Metric trung bình trên từng bộ kiểm thử tiếng Việt ứng với các SNR khác nhau	120
7.4	So sánh các mô hình thông qua các metrics trên bộ kiểm thử tiếng Việt	122
7.5	Độ trễ của pipeline lọc nhiễu động trong các trường hợp kiểm thử .	123
7.6	So sánh các metrics giữa các mô hình trên bộ kiểm thử tiếng Việt .	124
7.7	So sánh các metrics giữa mô hình đề xuất và các ứng dụng Sox phiên bản 14.4.2, CrystalSound phiên bản 0.15.2 (dòng trên) và phiên bản 1.4.0.0 (dòng dưới) trên bộ kiểm thử tiếng Việt	125
7.8	Metric trên từng bộ kiểm thử đa ngôn ngữ	126
7.9	Metrics mô hình đề xuất với các loại ngôn ngữ	127

Chương 1

Giới thiệu

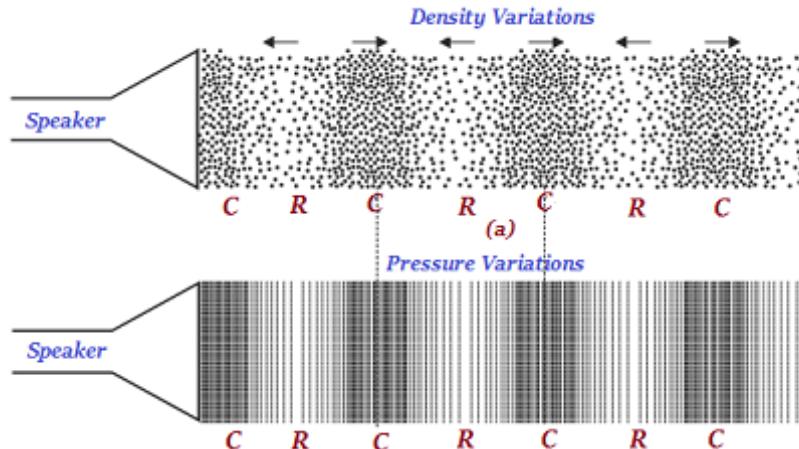
Trong chương này, chúng tôi giới thiệu về lý do chọn đề tài, phạm vi thực hiện, mục đích cũng như ý nghĩa khi chúng tôi tiến hành thực hiện đề tài này. Tuy nhiên trước hết chúng tôi sẽ trình bày lại một số khái niệm căn bản về âm thanh đã được trình bày trong sách giáo khoa Vật Lý lớp 12.

1.1 Âm thanh

Âm thanh được định nghĩa như là các dao động cưỡng bức dưới tác dụng của một lực nào đó lên các phần tử không khí khiến cho chúng dao động qua lại tạo nên các sóng âm truyền đi trong không khí và cuối cùng là tối được các thành phần có chức năng nhận biết. Nói một cách tổng quát hơn, âm thanh chính là tập hợp rất nhiều dao động cơ học trong đó các phần tử không khí đóng vai trò như một quả lắc. Chúng dao động qua lại, va chạm vào nhau khiến cho năng lượng của sự dao động này được truyền đi trong môi trường vật chất.

Sự dao động qua lại của các phần tử không khí có thể chung quy lại được xem là một hệ dao động điều hòa cưỡng bức dưới tác động của một hệ lực được tạo bởi nguồn phát. Để hình dung sự tương đồng này ta xét một hệ một con lắc đơn dao động điều hòa dưới tác dụng của trọng lực. Hình 1.2 thể hiện hệ cơ học mà chúng tôi đang xét.

Khi đó, giả sử lò xo đó có độ cứng là K và vật có khối lượng là m . Ta có phương



Hình 1.1: Sự truyền âm trong không khí, C (R) là nơi mà mật độ các hạt không khí dày (thưa) tương ứng

trình sau

$$m\vec{a} + m\vec{g} = K(\vec{x} + \delta\vec{x}). \quad (1.1)$$

Nhưng do phương trình cân bằng lực tại vị trí cân bằng của lò xo, chúng tôi đã có $m\vec{g} = K\delta\vec{x}$ nên công thức (1.1) sẽ được biến đổi thành

$$m\vec{a} = K\vec{x}$$

Từ đó suy ra

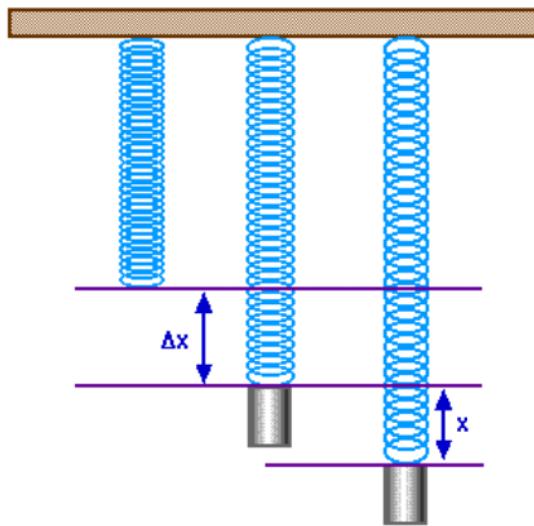
$$m(x)'' = -Kx. \quad (1.2)$$

chiếu phương trình trên lên trên trực đứng của hệ chúng ta có thể bỏ dấu vector của hệ đi, phương trình thu được chính là phương trình đại diện cho dao động của hệ này. Kết quả ta thu được đó chính là biểu thức

$$x = x_0 \cos(\omega(t - t_0)), \quad (1.3)$$

với $\omega = \sqrt{K/m}$, t_0 thể hiện gốc thời gian hệ bắt đầu giao động và x_0 chính là giá trị bị kéo ra của lò xo ở vị trí ban đầu.

Dễ thấy đây chính là một hàm lượng giác với chu kì $T = 2\pi\sqrt{m/K}$. Tuy nhiên,



Hình 1.2: Một hệ dao động điều hòa dưới tác dụng của của trọng lực và lực kéo từ lò xo có độ cứng là K được kéo ra một đoạn ban đầu x_0 so với vị trí cân bằng

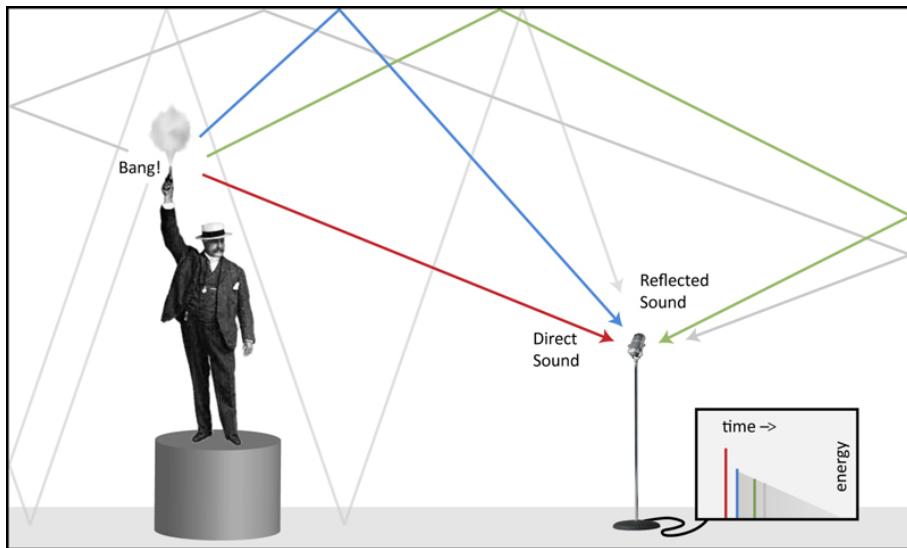
Phạm vi giới hạn

- 1 Chỉ lọc các âm nhiễu được đề ra ở Bảng 1.2
 - 2 Các tiếng vang và dội sẽ không được loại bỏ
 - 3 Chỉ nhắm đến độ hiệu quả khi lọc nhiễu ra khỏi tiếng Việt và tiếng Anh
 - 4 Ứng dụng sẽ chỉ được hiện thực cho hệ điều hành Windows
-

Bảng 1.1: Giới hạn phạm vi đề tài

đây chỉ là phương trình dao động tại thời điểm ban đầu của hệ, để phương trình này vẫn còn thỏa mãn hệ sau khi đã dao động qua một khoảng thời gian t , điều cần thiết phải đặt ra là năng lượng của hệ cơ học không thay đổi hay nói cách khác thì năng lượng của hệ không bị mất mát đi trong quá trình dao động.

Trong thực tế, năng lượng của sóng âm truyền đi này không ổn định, mà chúng sẽ giảm dần theo cả chiều không và thời gian. Sự suy giảm này có thể được nhận thấy rất rõ khi quan sát sự truyền đi các sóng trên mặt nước với điểm phát sóng đi là một nguồn điểm. Tuy nhiên, ở trong luận văn này, ở phương diện của máy tính, khi giọng nói được ghi nhận từ microphone, tín hiệu được giả định là đã ở mức ổn định và chúng tôi sẽ tiến hành xử lý trên đoạn âm thanh được ghi nhận, do vậy các yếu tố vật lý nêu trên sẽ chỉ ảnh hưởng tới quá trình âm thanh truyền từ miệng người sử dụng tới microphone của máy tính. Các tính chất suy giảm biên độ này ảnh hưởng trực tiếp tới quá trình kiểm thử mô hình của chúng tôi và sẽ được chúng tôi trình bày cụ thể hơn ở Mục 7.1.



Hình 1.3: Hiện tượng Room Impulse Response (tiếng dội) xảy ra khi âm thanh gốc bị suy giảm một lượng tuyến tính với thời gian truyền đi và lại bị ghi nhận lại sau một khoảng thời gian nhất định

Tuy nhiên, hiện tượng giảm âm khi truyền đi trong không khí có thể gây ra một số hiệu ứng đặc biệt, một ví dụ điển hình trong thực tế là hiện tượng tiếng vang xuất phát từ giọng nói bị dội lại khi người sử dụng đang ở trong phòng kín (Echo và Reverberation). Có khá nhiều các cuộc thi và các sản phẩm công nghiệp được thương mại hóa khác nhau nhằm tối việc đồng thời khử cả vang và tiếng dội này nhưng trong luận văn của chúng tôi, giới hạn sẽ chỉ nằm ở việc khử nhiễu, chúng tôi sẽ không xử lý cho trường hợp tiếng nói bị vang hay trường hợp người dùng sử dụng trong phòng kín làm cho giọng nói bị dội lại nhiều lần. Hình 1.3 thể hiện cho việc âm thanh sẽ bị dội khi người dùng đang ở trong phòng kín. Bảng 1.1 nêu ra các giới hạn cho phạm vi đề tài của luận văn của chúng tôi.

1.2 Đặt vấn đề

Nhiều có thể được hiểu là các thành phần dư thừa không mong muốn xuất hiện trong dữ liệu thực tế. Các dữ liệu nhiễu này không mang nhiều thông tin có giá trị về ý nghĩa của một câu nói hay một đoạn âm thanh, đôi khi chúng có thể gây hại cho các hệ thống xử lý tín hiệu. Trong thời kì dịch bệnh hoành hành như hiện nay, khiến cho các hoạt động trực tiếp dần được thay thế bởi các hoạt động trực

tuyến. Các cuộc họp, các buổi học online cũng dần trở nên phổ biến hơn, và ngày càng nhanh chóng thích nghi hơn với yêu cầu giao tiếp hiện tại. Nhu cầu giao tiếp trực tuyến nhiều hơn, yêu cầu về chất lượng âm thanh khi giao tiếp cũng cần phải được chú trọng. Nhưng một vấn đề không phải ai cũng có thể đảm bảo được trong quá trình hoạt động trực tuyến, đó chính là môi trường hoạt động của mình. Rất nhiều vấn đề khác nảy sinh từ vấn đề này, nhà có trẻ con đang quấy khóc, nhà gần đường lộ tiếng ồn từ hoạt động giao thông, và các yếu tố môi trường như mưa, gió, quạt bị hư gây ra các tiếng ồn lớn ảnh hưởng rất nhiều tới bản thân người đang hoạt động trực tuyến cũng như những người cùng tham gia vào phiên hoạt động đó. Để làm rõ lý do thực hiện đề tài này, chúng tôi sẽ nêu ra một số định nghĩa. Một đoạn âm thanh bất kì phát ra từ nguồn âm và lan truyền tới tai người nghe có thể được định nghĩa như sau

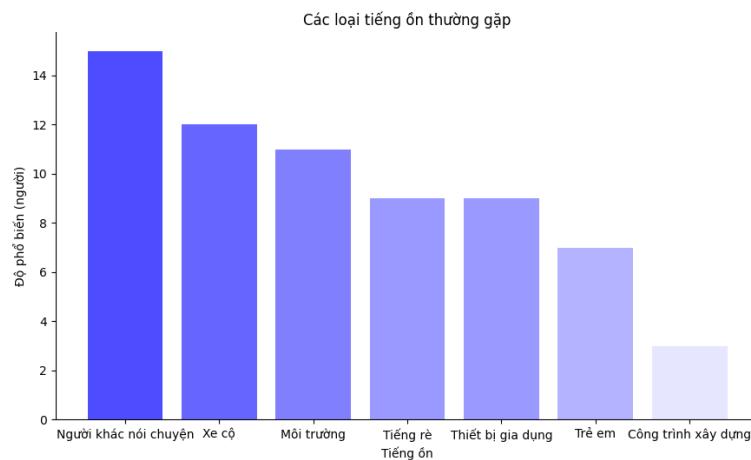
$$y(t) = x(t) + n(t), \quad (1.4)$$

với $y(t)$ là âm thanh hay trong trường hợp này là giọng nói, được ghi âm lại trong thực tế, $x(t)$ chính là thông tin hay giọng nói sạch được ghi lại trong đoạn ghi âm đó, $n(t)$ là các âm nhiễu xuất hiện từ các yếu tố khách quan như môi trường xung quanh người ghi âm. Vậy với một mô hình, chúng tôi gọi là f thì kết quả đầu ra của quá trình lọc được định nghĩa như sau

$$\hat{x}(t) = f(y(t)) \approx x(t), \quad (1.5)$$

với $\hat{x}(t)$ chính là kết quả đầu ra của mô hình. Hay nói cách khác, mô hình chúng tôi cần huấn luyện phải có khả năng tách được các thành phần của giọng nói sạch từ một âm thanh hỗn tạp được cho trước, để làm được như vậy chúng tôi cần xác định mô hình phải chịu được nhiễu từ các nguồn như thế nào.

Do chỉ được định nghĩa một cách rất tổng quát, là các phần dư thừa không mong muốn của dữ liệu, nhiều có thể mang bất cứ hình dạng trong bất cứ ngữ cảnh nào. Nếu đặt trường hợp giao tiếp trong môi trường là quán cà phê, chính các âm thanh được tạo bởi môi trường đó, tiếng cốc nước leng keng, tiếng nhạc phát ra từ các bộ loa đặt xung quanh quán, tiếng người nói chuyện trong quán, đôi khi lẫn cả tiếng xe cộ chạy qua lại trên đường. Chỉ trong một ngữ cảnh đơn giản,



Hình 1.4: Sáu loại nhiễu phổ biến nhất theo khảo sát được chúng tôi thực hiện

đã có thể thấy dù đi đâu, ở trong bất cứ môi trường nào, âm thanh mà tai người nghe được luôn tồn tại một lượng nhiễu nhất định nào đó. Chính vì sự đa dạng như vậy, số lượng âm thanh nhiễu có trong thực tế là rất lớn. Đã có nhiều cuộc thi được tổ chức ở quy mô toàn cầu nhằm tìm kiếm các giải pháp để loại bỏ nhiễu một cách tổng quát và triệt để. Tuy nhiên điều này vẫn phải nhiều khó khăn, từ các loại nhiễu có tần số cấu tạo ổn định theo thời gian tới các âm nhiễu có các tần số cấu tạo biến đổi tương tự giọng nói như giọng thuyết minh được phát ra từ tivi trong quá trình người nói hoạt động trực tuyến. Dù đã được nghiên cứu trong nhiều năm, từ những thuật toán cổ điển như lọc nhiễu thông qua các bộ lọc với giả định nhiễu không có sự biến thiên liên tục về tần số cấu tạo theo thời gian cho tới những cách tiếp cận gần đây như các mô hình học sâu được đề xuất trong cuộc thi Deep Noise Suppression (DNS) [14] của Microsoft, DEMAND [16], Chime [15], ... Nhưng kết quả thu được vẫn còn khá nhiều hạn chế, về cả chất lượng của âm thanh đầu ra và cả khả năng lọc của các cách tiếp cận. Do vậy trong luận văn này, trong số sáu loại nhiễu phổ biến mà chúng tôi đã chọn ra từ khảo sát¹ của mình được thể hiện trong Hình 1.4, từ đó chúng tôi lựa ra nhiều cụ thể cho từng loại. Chúng tôi trình bày các trường hợp được chọn ở Bảng 1.2.

¹Khảo sát tại https://docs.google.com/forms/d/1p_3ndm-ggG1ezMbWb1KGSpMH00cWZ_zSfcOy1XKsnjI

Loại nhiễu	Các nhiễu cụ thể
Người khác nói chuyện	Tiếng thông báo ở sân bay, sân tàu; tiếng trong quán cà phê
Xe cộ	Tiếng xe tải chạy; tiếng đường kẹt xe
Môi trường	Tiếng mưa (mưa to, vừa, nhỏ)
Thiết bị gia dụng	Tiếng quạt (quạt bị hư)
Tiếng rè	Nhiều trăng (white noise)
Trẻ em	Tiếng em bé khóc

Bảng 1.2: Các âm nhiễu được chọn từ các loại nhiễu phổ biến

1.3 Mục tiêu và nhiệm vụ đề tài

Mục tiêu của đề tài này là sử dụng các mô hình học sâu để xử lý loại bỏ nhiễu ra khỏi một đoạn âm thanh đầu vào và sau đó mở rộng ra sử dụng mô hình này cho các cuộc hội thoại thời gian thực. Để thực hiện điều này chúng tôi chia mục tiêu trên thành các nhiệm vụ dưới đây

Nhiệm vụ 1. Tìm hiểu sâu kiến thức về các phép xử lý tín hiệu số

- Biến đổi Fourier
- Biến đổi Fourier thời gian ngắn
- Biến đổi Fourier rời rạc và giải thuật biến đổi Fourier nhanh

Nhiệm vụ 2. Tìm hiểu sâu các khái niệm trong âm học

- Tần số cảm nhận
- Đặc biệt là thang đo độ to, và các metrics đo chất lượng âm thanh

Nhiệm vụ 3. Tìm hiểu sâu về học máy nền tảng

- Tìm hiểu RNN
- Tìm hiểu LSTM

Nhiệm vụ 4. Tìm hiểu, khảo sát các hướng tiếp cận và đề xuất mô hình

- Fullband and Subband Fusion Model
- Deep Complex Convolution RNN
- Discrete Cosine Transform Convolution RNN
- Dual signal Transformation LSTM

Nhiệm vụ 5. Tinh chỉnh mô hình đề xuất được chọn cho phù hợp với yêu cầu bài toán

- Rút giảm số tham số mô hình
- Nén mô hình xuống để phù hợp với thiết bị
- Giảm độ trễ của mô hình trên thiết bị

Nhiệm vụ 6. Sử dụng mô hình vào một sản phẩm ứng dụng thực tế

Ứng dụng cuối cùng sau khi hoàn thành luận văn này, chúng tôi kì vọng sẽ có thể hoàn thành được một ứng dụng chạy trên máy tính cá nhân với cấu hình như Bảng 1.3. Các cấu hình dựa trên khảo sát² về máy tính của chúng tôi, được chọn từ cấu hình hệ thống phổ biến nhất trong khảo sát và quyết định đó sẽ là cấu hình mà chúng tôi sử dụng như yêu cầu cho ứng dụng này.

Yêu cầu tối thiểu	
Hệ điều hành	Windows 10
CPU	Intel i5 (4 cores), 2.1 GHz
RAM	4GB, 2666 MHz
GPU	Không sử dụng

Bảng 1.3: Yêu cầu về cấu hình cần thiết để chạy ứng dụng

1.4 Kết quả mong đợi

Sau luận văn này, ngoài các kiến thức thu được trong quá trình tìm hiểu, chúng tôi mong muốn sẽ có thể đề xuất được một hàm măt măt mới dựa trên các hàm măt măt được sử dụng phổ biến trong xử lý âm thanh, một mô hình vừa có khả năng đáp ứng tính thời gian thực của hệ thống lọc nhiễu được huấn luyện dựa trên hàm măt măt mà chúng tôi đề xuất. Ngoài ra về khía cạnh ứng dụng, chúng tôi mong muốn có thể ứng dụng mô hình của mình vào một ứng dụng thực tế và có khả năng hoạt động trong thời gian thực đối với các dòng máy tương đối yếu (Bảng 1.3) mà vẫn đảm bảo yếu tố chất lượng giọng nói sau khi lọc.

²Khảo sát tại <https://docs.google.com/forms/d/1HzYfNQ-ECDQ-yq-6XoHTLTp5ikJBNd0A5Lds9CA1Mo>

1.5 Ý nghĩa của đề tài

Xuất phát từ nhu cầu giao tiếp thực tế, cùng với thực trạng ô nhiễm tiếng ồn ở nhiều nơi trong thành phố Hồ Chí Minh, đề tài luận văn của chúng tôi mong muốn sẽ có thể góp phần vào cải thiện chất lượng của các cuộc hội họp học tập trực tuyến từ đó có thể nâng cao hiệu quả cũng như chất lượng trong học tập và làm việc trong tình hình dịch bệnh còn nhiều diễn biến phức tạp.

1.6 Cấu trúc luận văn

Luận văn được chia thành các chương như sau

- **Chương 1.** Ở chương này, chúng tôi giới thiệu về đề tài, định nghĩa bài toán, giới hạn phạm vi và đề ra các nhiệm vụ, kết quả mong đợi và cả ý nghĩa thực tiễn của việc thực hiện đề tài.
- **Chương 2.** Chương này sẽ trình bày về các khái niệm cơ bản trong xử lý tín hiệu số.
- **Chương 3.** Chương này sẽ giới thiệu về âm thanh và các tính chất cơ bản của nó.
- **Chương 4.** Chương này giới thiệu về các khái niệm học sâu được chúng tôi sử dụng trong luận văn này.
- **Chương 5.** Đây là chương chúng tôi nghiên cứu về các mô hình từ các cuộc thi lớn từ đó đề xuất ra mô hình và cải tiến hàm mất mát của mình.
- **Chương 6.** Đây là chương chúng tôi phân tích các đặc điểm dữ liệu từ đó nêu các ra các yêu cầu cho ứng dụng. Các thiết kế cũng sẽ được miêu tả chi tiết trong chương này.
- **Chương 7.** Đây là chương chúng tôi tiến hành kiểm thử mô hình của mình và so sánh với các mô hình, ứng dụng được đề xuất trong nghiên cứu và công nghiệp.

- **Chương 8.** Chương này kết luận lại những điều chúng tôi đã thực hiện được, các hạn chế của nó và mục tiêu kế tiếp của chúng tôi cho đê tài này.

Chương 2

Xử lý tín hiệu số

2.1 Phép biến đổi Fourier thời gian ngắn

Từ thực tiễn, một điều dễ thấy là giọng nói sẽ chỉ xuất hiện trong một thời gian ngắn và có xu hướng không ổn định theo thời gian, nó phụ thuộc vào âm độ giọng nói, thông tin người nói muốn truyền đạt, và cả tâm trạng người nói lúc đó nữa và nếu chỉ phân tích trên toàn bộ đoạn âm thanh, thì việc xác định tại thời điểm bắt kì liệu đang có những sóng nào xuất hiện đối với biến đổi Fourier thông thường là không thể. Đó là lí do ta cần phải tìm hiểu về khái niệm **biến đổi Fourier trong thời gian ngắn** hay **short-time Fourier transform**, gọi tắt là STFT.

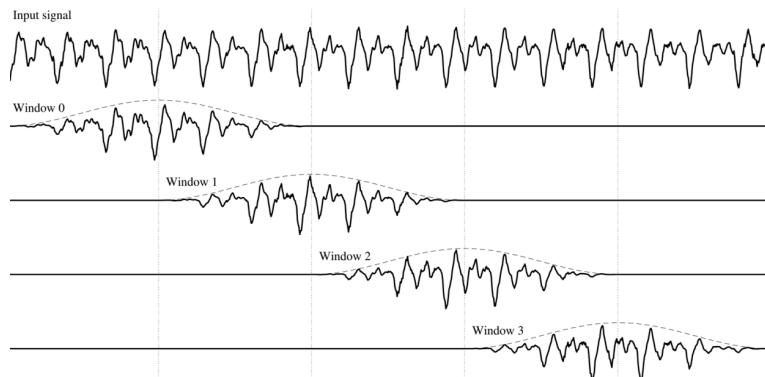
2.1.1 Định nghĩa

Phép biến đổi Fourier trong thời gian ngắn là một biến đổi Fourier nhưng chỉ thực hiện trong khoảng thời gian rất ngắn (thường được chọn là 32ms đối với tần số lấy mẫu là 16000 mẫu/s), và trả về spectrum (đồ thị biến độ của biến đổi Fourier theo tần số) của các tần số cấu tạo nên âm thanh nằm trong khoảng thời gian đó. Về mặt toán học, *Phép biến đổi Fourier trong thời gian ngắn* được định nghĩa như sau

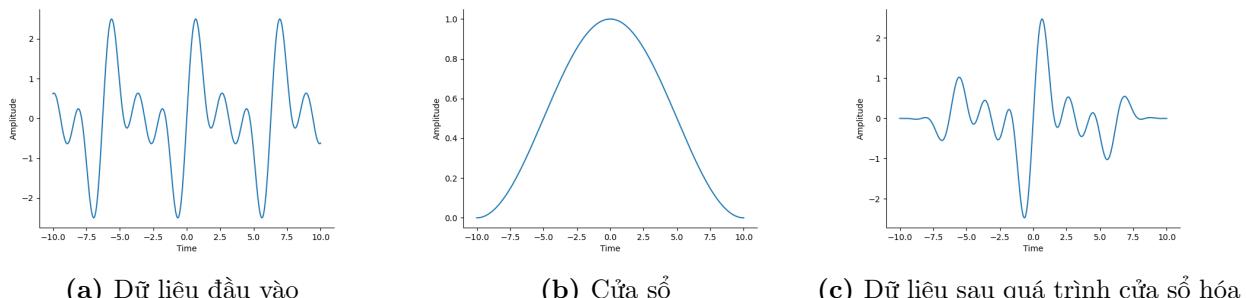
$$F(\tau, \omega) = \int_{-\infty}^{\infty} f(t)w(t - \tau)e^{-\omega t i} dt, \quad (2.1)$$

với $f(t)$ là tín hiệu theo thời gian, $w(t - \tau)$ là cửa sổ dùng để nén dữ liệu, ta sẽ làm rõ chức năng cụ thể của hàm cửa sổ này ở phần tiếp theo, và $F(\tau, \omega)$ là biểu diễn sóng có bước sóng ω tại thời điểm τ trong $f(t)$.

Ý tưởng chính của biến đổi Fourier thời gian ngắn là thay vì dữ liệu là cả một đoạn âm thanh dài, ta chỉ chọn lấy một khoảng dữ liệu nằm trong $[t_1, t_2]$ và thực hiện biến đổi Fourier lên đoạn dữ liệu này. Bằng việc giới hạn giá trị chỉ khác 0 ở một số khoảng nhất định, hàm cửa sổ $w(t - \tau)$ khiến cho giá trị đầu vào của biến đổi Fourier thay đổi từ trên toàn bộ miền thời gian về chỉ còn tồn tại trong khoảng $[t_1, t_2]$. Hình 2.1 minh họa cho điều mà chúng tôi vừa trình bày. Giá trị τ ở đây được sử dụng như phép dịch của $w(t)$ trên miền thời gian để điều chỉnh khoảng $[t_1, t_2]$ cho đầu vào của biến đổi Fourier, giá trị này được gọi là **hop length** hay **độ dịch của cửa sổ**.



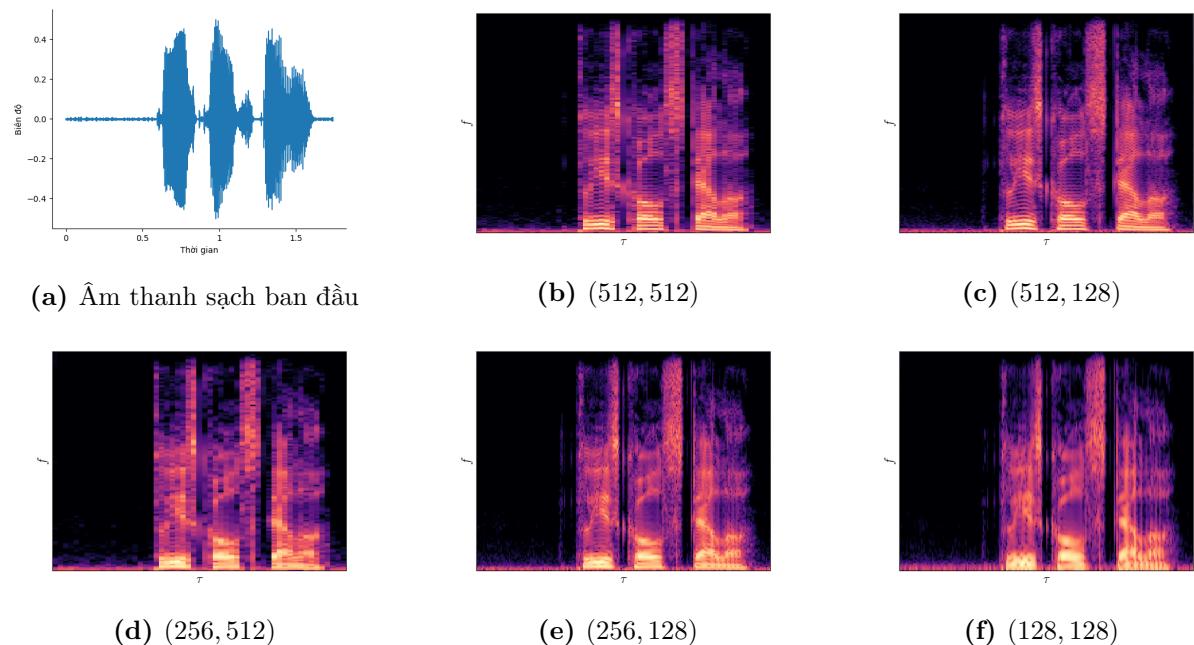
Hình 2.1: Tác dụng của cửa sổ trong biến đổi Fourier thời gian ngắn



Hình 2.2: Quá trình cửa sổ hóa tín hiệu thời gian ngắn trong biến đổi Fourier thời gian ngắn

Việc xác định các giá trị của τ và khoảng cách giữa t_1, t_2 là tùy vào nhu cầu, độ mịn cần thiết của ứng dụng mà tùy chỉnh sao cho phù hợp. Trong luận văn này,

ta sử dụng chiều dài tín hiệu là 32ms (ứng với 512 mẫu được lấy mẫu với tần số 16000 mẫu/s) và khoảng cách giữa hai tín hiệu là 8ms (ứng với 128 mẫu được lấy mẫu với tần số là 16000 mẫu/s). Sau khi đã cửa sổ hóa tín hiệu, sử dụng biến đổi Fourier lên các tín hiệu đã được cửa sổ hóa và lấy kết quả của chúng biểu diễn theo hai trục ω và τ , ta thu được biểu diễn của âm thanh theo miền tần số thời gian và đó cũng là miền thông tin mà sinh viên thực hiện sẽ sử dụng để huấn luyện mô hình của mình. Biểu diễn âm thanh trong miền này bằng các giá trị biên độ số phức gọi là **spectrogram**.



Hình 2.3: Spectrogram biểu diễn với chiều dài cửa sổ T và khoảng cách giữa các tín hiệu τ , được kí hiệu trong hình là (T, τ)

2.1.2 Hàm cửa sổ

Trong công thức biến đổi Fourier thời gian ngắn được nêu ở công thức (2.1). Hàm $w(t - \tau)$ hay trong phần này chúng tôi sẽ gọi là hàm cửa sổ đóng vai trò như một bộ giới hạn tín hiệu về một khoảng thời gian ngắn tùy thuộc vào chiều dài cửa sổ đầu vào. Để định nghĩa cho hàm cửa sổ này, chúng tôi gọi $w_0(t)$ đại diện cho hàm số thể hiện các giá trị mà tại đó hàm cửa sổ này mang giá trị khác 0, như vậy định nghĩa của hàm cửa sổ có thể được chúng tôi thể hiện như sau

$$w(t) = \begin{cases} w_0(t), & \text{nếu } t_1 \leq t \leq t_2, \\ 0, & \text{các trường hợp khác.} \end{cases} \quad (2.2)$$

Dễ thấy, hàm cửa sổ của chúng tôi được chia thành hai nhánh giá trị, một của $w_0(t)$ và 0. Các giá trị nhận từ hàm $w_0(t)$ chỉ được giới hạn trong khoảng $[t_1, t_2]$, nếu giá trị của t rời ra khỏi khoảng này, giá trị hàm cửa sổ trả về sẽ mang giá trị 0. Theo công thức (2.1), $w(t - \tau)$ sẽ làm cho $w(t)$ bị dịch sang bên phải một đoạn τ nên ở đây τ đóng vai trò như một bộ dịch chuyển cửa sổ đi xuyên suốt trên miền thời gian t của dữ liệu $f(t)$ và với chiều dài cửa sổ T là không đổi, giá trị t_1 và t_2 hoàn toàn có thể xác định được.

Sau khi $f(t)$ nhân với cửa sổ $w(t - \tau)$ này, vì các giá trị của cửa sổ chỉ khác 0 khi $t - \tau$ rơi vào khoảng $[t_1, t_2]$ và là 0 với tất cả các điểm khác, điều này làm cho công thức (2.1) được chuyển thành

$$\begin{aligned} F(\tau, \omega) &= \int_{-\infty}^{\infty} f(t)w(t - \tau)e^{-\omega ti}dt \\ &= \int_{-\infty}^{t_1} f(t)w(t - \tau)e^{-\omega ti}dt + \int_{t_1}^{t_2} f(t)w(t - \tau)e^{-\omega ti}dt + \int_{t_2}^{\infty} f(t)w(t - \tau)e^{-\omega ti}dt \\ &= \int_{t_1}^{t_2} f(t)w_0(t - \tau)e^{-\omega ti}dt, \end{aligned} \quad (2.3)$$

và đây chính là điều làm nên tính “*thời gian ngắn*” trong biến đổi Fourier thời gian ngắn.

Cửa sổ đã làm cho các phần dữ liệu nằm bên ngoài cửa sổ bị suy giảm về 0 và biến đổi Fourier chỉ thực sự xảy ra cho các phần dữ liệu nằm bên trong cửa sổ lúc này mang giá trị tương ứng $w_0(t - \tau)f(t)$. Và vì biến đổi Fourier được thực hiện trên dữ liệu đã bị cửa sổ hóa (đã được nhân với hàm cửa sổ) do đó, các loại cửa sổ khác nhau cũng sẽ làm ảnh hưởng tới kết quả của quá trình biến đổi này.

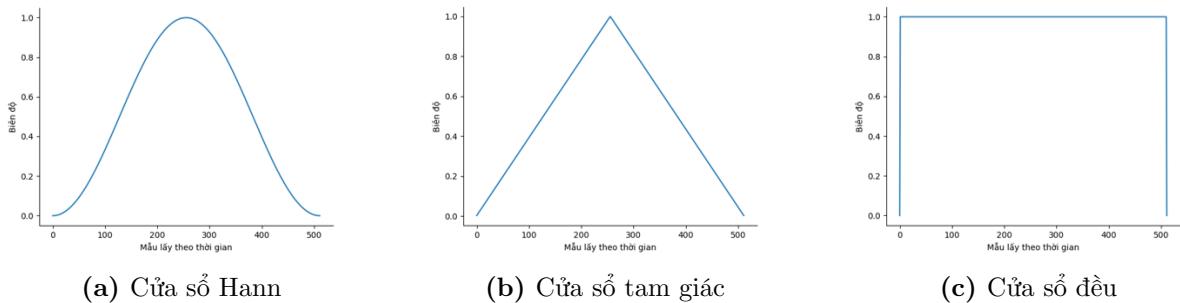
Trong thực tế, có khá nhiều loại cửa sổ có thể được sử dụng, Bảng 2.1 liệt kê một số cửa sổ và dữ liệu chúng thường được sử dụng trong thực tế. Để thể hiện tác dụng khác nhau của các loại cửa sổ lên cùng một loại dữ liệu, chúng tôi lấy dữ liệu từ hàm $f(t) = \sin(t)$ với $t \in [0, 12\pi]$ (tương ứng với sáu chu kỳ của $\sin(t)$).

Kết quả spectrum của biến đổi Fourier được thể hiện ở Hình 2.5 đã cho thấy phần nào tác động của cửa sổ ảnh hưởng lên kết quả cuối cùng của biến đổi Fourier

Loại tín hiệu	Cửa sổ
Tổng hợp các sóng sine và cosine	Hann
Nhiều tín hiệu	Uniform
Sóng sine và cosine có tần số gần nhau	Uniform
Chưa xác định	Hann (phù hợp 95% loại dữ liệu)

Bảng 2.1: Bảng đề xuất cửa sổ sử dụng trong một số trường hợp (nguồn [50])

thời gian ngắn. Cửa sổ Hann và cửa sổ tam giác cho kết quả khá tốt với các giá trị cửa sổ bị tách ra riêng biệt và các vùng lân cận cũng không bị ảnh hưởng nhiều, trái với đó cửa sổ đều trả về kết quả với các vùng tần số lân cận tần số chính bị ảnh hưởng khá nhiều do đó sẽ không phù hợp với dữ liệu dạng sóng sine hay cosine. Trong trường hợp của chúng tôi, các âm thanh đầu vào cấu tạo từ các sóng sine và cosine và ảnh hưởng trực tiếp tới thiết kế của mô hình do vậy cửa sổ Hann là phù hợp nhất với yêu cầu của đề tài này.

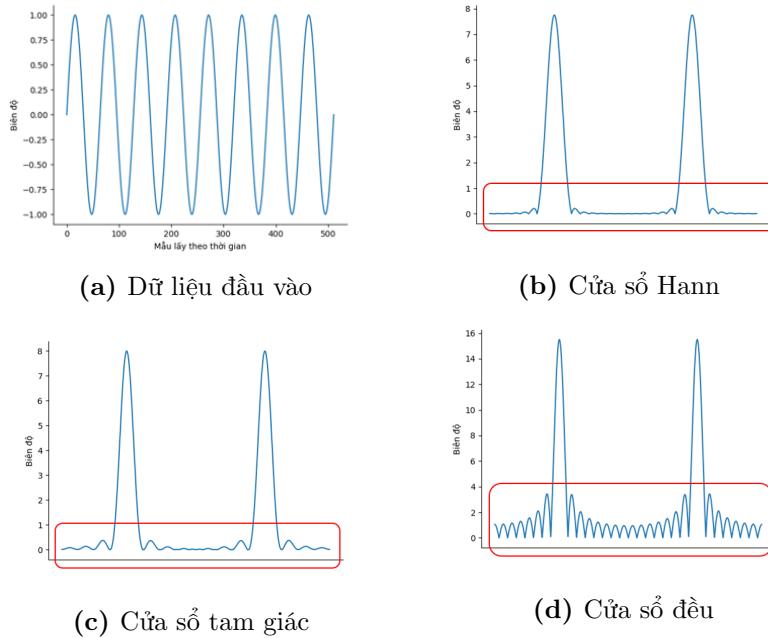


Hình 2.4: Đồ thị ba loại cửa sổ Hann, tam giác và đều

2.1.3 Phương pháp Overlap Add

Xuất phát từ bài toán tìm lại hàm $f(t)$ từ spectrogram thu được từ bước biến đổi thuận được nêu ở công thức (2.1) và các tài liệu [44, 45]. Để bắt đầu vào việc phân tích bản chất của phương pháp này, chúng tôi đặt ra một số định nghĩa:

- $F(\omega, \tau) = \mathcal{F}\{f(t)\}(\omega, \tau)$ là biến đổi Fourier thời gian ngắn ứng với tốc độ góc ω và độ dịch cửa sổ τ .
- $f(t)$ là tín hiệu đầu vào.
- $w(t)$ là hàm cửa sổ.



Hình 2.5: spectrogram của dữ liệu được cửa sổ hóa bởi ba loại cửa sổ Hann, tam giác và đều, tác dụng của cửa sổ lên spectrogram được thể hiện ở vùng khoanh đỏ.

Biến đổi Fourier thời gian ngắn như đã định nghĩa ở công thức (2.1)

$$F(\omega, \tau) = \int_{-\infty}^{+\infty} w(t - \tau) f(t) e^{-\omega t i} dt,$$

sau khi đã biến đổi thuận sang miền tần số thời gian, được thể hiện dưới các giá trị phức của spectrogram và thực hiện các phép biến đổi, phân tích trên miền này, chúng tôi cần một phương pháp để có thể chuyển đổi được spectrogram F này về lại miền thời gian dưới dạng $f(t)$. Phương pháp Overlap Add được sử dụng nhằm mục đích chuyển đổi ngược từ miền tần số thời gian về miền thời gian, có thể được xem mà một bước cần thiết bên trong biến đổi Fourier ngược thời gian ngắn [44, 45]. Overlap Add được chia thành hai trường hợp: *spectrogram không bị biến đổi* và *spectrogram đã bị biến đổi*. Để thuận tiện trong trình bày, chúng tôi gọi $F'(\omega, \tau)$ và $f'(t)$ lần lượt là giá trị của spectrogram sau khi bị biến đổi và giá trị trên miền thời gian sau khi nghịch đảo biến đổi Fourier tương ứng với $F'(\omega, \tau)$.

Spectrogram không bị biến đổi: Biến đổi Fourier ngược của $F(\omega, \tau)$ là

$$\mathcal{F}^{-1}\{F'(\omega, \tau)\}(t) = \mathcal{F}^{-1}\{F(\omega, \tau)\}(t) = w(t - \tau) f(t). \quad (2.4)$$

Cửa sổ $w(t)$	Định nghĩa	T	t_1	t_2
Hann	$w_0(t) = 1 + \cos(t)$	2π	$-\pi$	π
Tam giác	$w_0(t) = \begin{cases} t & \text{nếu } 0 \leq t < 1, \\ -t + 2 & \text{nếu } 1 \leq t \leq 2 \end{cases}$	2	0	2
Đều	$w_0(t) = 1$	1	0	1

Bảng 2.2: Bảng định nghĩa các hàm cửa sổ

Thông qua công thức (2.4), dễ thấy, phương pháp Overlap Add được hiện thực nhằm tìm hàm $f'(t)$ từ dữ liệu bị cửa sổ hóa $w(t - \tau)f(t)$. Trong trường hợp này, Overlap Add sẽ được thực hiện thông qua một tính chất của hàm cửa sổ được trình bày như sau

$$\begin{aligned}
 f'(t) &= \int_{-\infty}^{+\infty} \mathcal{F}^{-1}\{F'(\omega, \tau)\}(t)d\tau \\
 &= \int_{-\infty}^{+\infty} w(t - \tau)f(t)d\tau \\
 &= f(t) \int_{-\infty}^{+\infty} w(t - \tau)d\tau \\
 &= Cf(t).
 \end{aligned} \tag{2.5}$$

Công thức (2.5) thể hiện bản chất của Overlap Add và đây là một tính chất đặc trưng của các hàm cửa sổ. Dễ thấy, khi cộng tất cả các dữ liệu đã bị cửa sổ hóa $w(t - \tau)f(t)$, chúng tôi thu lại được giá trị của $f(t)$ bị nhân thêm cho một hằng C lần. Để lần lượt kiểm chứng tính chất này lên các loại cửa sổ khác nhau, ta sẽ tiến hành tìm giá trị của C trong ba trường hợp cửa sổ thường được sử dụng: *cửa sổ Hann*, *cửa sổ đều* và *cửa sổ tam giác*. Các hàm cửa sổ được định nghĩa như trong Bảng 2.2.

Tính toán chi tiết các cửa sổ:

- i) *Hàm cửa sổ Hann.*

$$\begin{aligned}
 f'(t) &= \int_{t_1}^{t_2} w(t-\tau) f(t) d\tau \\
 &= \int_{-\pi}^{\pi} (1 + \cos(t-\tau)) f(t) d\tau \\
 &= \int_{-\pi}^{\pi} f(t) d\tau + \int_{-\pi}^{\pi} \cos(t-\tau) f(t) d\tau \\
 &= \tau f(t) \Big|_{-\pi}^{\pi} + f(t) \sin(t-\tau) \Big|_{-\pi}^{\pi} \\
 &= 2\pi f(t) + f(t)(\sin(t-\pi) - \sin(t+\pi)) \\
 &= 2\pi f(t).
 \end{aligned}$$

ii) *Hàm cửa sổ tam giác*

$$\begin{aligned}
 f'(t) &= \int_{t_1}^{t_2} w(t-\tau) f(t) d\tau \\
 &= \int_0^1 (t-\tau) f(t) d\tau + \int_1^2 (\tau-t+2) f(t) d\tau \\
 &= \left(\tau t f(t) - \frac{\tau^2}{2} f(t) \right) \Big|_0^1 + \left(\frac{\tau^2}{2} f(t) - \tau t f(t) + 2\tau f(t) \right) \Big|_1^2 \\
 &= t f(t) - \frac{f(t)}{2} + 2f(t) - 2t f(t) + 4f(t) - \frac{f(t)}{2} + t f(t) - 2f(t) \\
 &= 3f(t).
 \end{aligned}$$

iii) *Hàm cửa sổ đều*

$$\begin{aligned}
 f'(t) &= \int_{t_1}^{t_2} w(t-\tau) f(t) d\tau \\
 &= \int_0^1 1 f(t) d\tau \\
 &= f(t).
 \end{aligned}$$

Như vậy, cả ba hàm cửa sổ thông dụng đều thỏa được tính chất của hàm cửa sổ trong phương pháp Overlap Add. Nhưng trong luận văn này, spectrogram của ta sẽ bị biến đổi (được đem nhân với một “mask” số thực, cụ thể sẽ được trình bày trong Chương 5), như vậy tính chất của Overlap Add khi spectrogram không bị biến đổi sẽ không còn đúng nữa, lúc này $F'(\omega, \tau) \neq F(\omega, \tau)$. Do đó, chúng tôi sẽ tiếp tục với phương pháp Overlap Add trong trường hợp spectrogram bị biến đổi.

Spectrogram bị biến đổi: Lại có công thức của Overlap Add được thể hiện dưới biến đổi Fourier ngược của spectrogram $F(\omega, \tau)$ được thay đổi thông qua mask dự đoán $P(\omega, \tau)$

$$f'(t) = \int_{-\infty}^{+\infty} \mathcal{F}^{-1}\{F(\omega, \tau)P(\omega, \tau)\}(t)w^{-1}(t - \tau)d\tau, \quad (2.6)$$

với $\mathcal{F}^{-1}\{\cdot\}(t)$ đại diện cho biến đổi Fourier nghịch của tín hiệu tại thời điểm t , $P(\omega, \tau)$ là “mask” được mô hình chúng tôi dự đoán, mask này ở dạng tổng quát và được chúng tôi xét như một “mask” số phức, $p(t)$ là tín hiệu được tạo ra bởi biến đổi Fourier nghịch của $P(\omega, \tau)$, $w^{-1}(t)$ là hàm cửa sổ nghịch được định nghĩa sau

$$\int_{-\infty}^{+\infty} w(t)w^{-1}(t)dt = 1, \quad (2.7)$$

tương ứng với mỗi hàm cửa sổ $w(t)$, để có thể thực hiện phương pháp Overlap Add và đảm bảo tính liên tục của dữ liệu, chúng tôi nhân thêm vào kết quả sau khi thực hiện biến đổi Fourier nghịch một cửa sổ $w^{-1}(t)$ đảm bảo điều kiện trên và thông qua đó tính toán được kết quả cuối cùng. Để xem xét các tính chất của kết quả tạo ra bởi phương pháp Overlap Add trong trường hợp này và lý do phải tách thành hai trường hợp Overlap Add cho từng loại spectrogram, ta sẽ tiến hành biến đổi từ vấn đề đã được nêu trong công thức (2.6) như sau

$$\begin{aligned} f'(t) &= \int_{-\infty}^{+\infty} \mathcal{F}^{-1}\{F(\omega, \tau)P(\omega, \tau)\}(t)w^{-1}(t - \tau)d\tau \\ &= \int_{-\infty}^{+\infty} ((wf) * p)(t)w^{-1}(t - \tau)d\tau \\ &= \int_{-\infty}^{+\infty} \left(\int_{-\infty}^{+\infty} w(t - \tau - \tau')f(t - \tau')p(\tau')d\tau' \right) w^{-1}(t - \tau)d\tau \\ &= \int_{-\infty}^{+\infty} f(t - \tau')p(\tau') \int_{-\infty}^{+\infty} w(t - \tau - \tau')w^{-1}(t - \tau)d\tau d\tau' \\ &= \int_{-\infty}^{+\infty} f(t - \tau')w'(\tau)p(\tau')d\tau' \\ &= (f * w'p)(t). \end{aligned} \quad (2.8)$$

Sự liên hệ giữa phép tích chập và biến đổi Fourier sẽ được chúng tôi chứng minh

ở Mục 5.1, $w'(\tau)$ là một cửa sổ mới được cấu tạo từ

$$w'(\tau) = \int_{-\infty}^{+\infty} w(t + \tau) w^{-1}(t) dt. \quad (2.9)$$

Để có thể chứng minh được tích phân (2.9) sẽ tạo thành một hàm cửa sổ, chúng tôi xét công thức tổng quát hơn của cửa sổ được tạo từ công thức (2.9), với $w_1(t)$ và $w_2(t)$ như sau

$$w'(\tau) = \int_{-\infty}^{+\infty} w_1(t + \tau) w_2(t) dt, \quad (2.10)$$

với T_1 và T_2 ứng với chiều dài cửa sổ w_1 và w_2 (khoảng thời gian mà trong đó cửa sổ có giá trị khác 0). Lại có

$$w_1(t) = \begin{cases} f(w_1(t + \tau)), & \text{nếu } \frac{-T_1}{2} + \tau \leq t + \tau \leq \frac{T_1}{2} + \tau, \\ 0, & \text{các trường hợp khác,} \end{cases} \quad (2.11)$$

$$w_2(t) = \begin{cases} f(w_2(t)), & \text{nếu } \frac{-T_2}{2} \leq t \leq \frac{T_2}{2}, \\ 0, & \text{các trường hợp khác,} \end{cases} \quad (2.12)$$

với $f(w(t))$ là hàm lấy giá trị cửa sổ $w(t)$ tại thời điểm t mà tại đó $w(t)$ có giá trị khác 0. Vậy công thức tích phân (2.10) có thể được hiểu thành độ trùng lặp giữa cửa sổ $w_2(t)$ và cửa sổ $w_1(t)$ được dịch đi một đoạn τ . Để thấy

$$w_1(t + \tau) w_2(t) = \begin{cases} f(w_1(t + \tau)) f(w_2(t)), & \text{nếu } \max\left(\frac{-T_1}{2} + \tau, \frac{-T_2}{2}\right) \leq t \leq \min\left(\frac{T_1}{2} + \tau, \frac{T_2}{2}\right), \\ 0, & \text{các trường hợp khác.} \end{cases} \quad (2.13)$$

Vậy tích hai cửa sổ được tạo thành bởi $w_1(t + \tau)$ và $w_2(t)$ sẽ chỉ có giá trị khi

$$\max\left(\frac{-T_1}{2} + \tau, \frac{-T_2}{2}\right) \leq t \leq \min\left(\frac{T_1}{2} + \tau, \frac{T_2}{2}\right), \quad (2.14)$$

và nếu như bất đẳng thức trên không được thỏa mãn bởi T_1 , T_2 và τ thì giá trị của tích này sẽ 0 và tích phân ở công thức (2.10) cũng sẽ có giá trị 0. Vậy nên hàm mới được tạo ra bởi tích phân (2.10) sẽ thỏa mãn tính chất của một hàm cửa sổ. Vậy giả định của cửa sổ sẽ quy định giá trị $w'(0)$ này.

Lại có

$$\begin{aligned}
 f'(t) &= \int_{-\infty}^{+\infty} \mathcal{F}^{-1}\{F(\omega, \tau)P(\omega, \tau)\}(t)d\tau \\
 &= \int_{-\infty}^{+\infty} ((wf) * p)(t)d\tau \\
 &= \int_{-\infty}^{+\infty} \left(\int_{-\infty}^{+\infty} w(t - \tau - \tau')f(t - \tau')p(\tau')d\tau' \right) d\tau \\
 &= \int_{-\infty}^{+\infty} f(t - \tau')p(\tau') \int_{-\infty}^{+\infty} w(t - \tau - \tau')d\tau d\tau' \\
 &= C \times (f * p)(t).
 \end{aligned} \tag{2.15}$$

Khi sử dụng spectrogram bị biến đổi vào giải thuật Overlap Add với giả định spectrogram không biến đổi, chúng tôi thu được tính chất

$$\begin{aligned}
 \frac{d}{dt}f'(t) &= \frac{d}{dt} \left(\int_{-\infty}^{+\infty} f(\tau)p(t - \tau)d\tau \right) \\
 &= \int_{-\infty}^{+\infty} f(\tau) \left(\frac{d}{dt}p(t - \tau) \right) d\tau,
 \end{aligned}$$

vì đạo hàm $p(t)$ sẽ không liên tục vì dữ liệu này được dự đoán ra bởi mô hình và với giả định của biến đổi Fourier rời rạc (xảy ra khi nghịch đảo Fourier cho từng khoảng) rằng dữ liệu sẽ lặp lại sau từng chu kì của chúng. Do vậy tại các điểm mứt của $p(t)$, sự chuyển tiếp sẽ xảy ra và nếu dữ liệu ở hai điểm đầu và cuối của mỗi chu kì kế tiếp nhau không khớp, chúng sẽ gây ra sự không liên tục và khiến cho đạo hàm của $p(t)$ bị gián đoạn. Nhưng mặt khác

$$\begin{aligned}
 \frac{d}{dt}f'(t) &= \frac{d}{dt} \left(\int_{-\infty}^{+\infty} f(\tau)w'(t - \tau)p(t - \tau)d\tau \right) \\
 &= \int_{-\infty}^{+\infty} f(\tau) \left(\frac{d}{dt}(w'(t - \tau)p(t - \tau)) \right) d\tau,
 \end{aligned}$$

sự không liên tục này lại không xảy ra khi nhân thêm một cửa sổ $w'(t)$ vào $p(t)$, lúc này hai đầu mứt của dữ liệu được biến đổi dần về 0 và sự không liên tục không xảy ra.

Hiện tượng trên được chúng tôi bắt gặp trong lúc hiện thực luận văn của mình. Sự không liên tục này gây ra các tiếng lạ trong âm thanh đầu ra bởi đường ống lọc nhiễu của ứng dụng khiến cho giọng nói sau khi lọc không còn đảm bảo chất

lượng nữa.

Thông qua hai trường hợp của spectrogram bị thay đổi và spectrogram không thay đổi, chúng tôi đã cho thấy phương pháp Overlap Add thỏa mãn được xây dựng lại hàm $f'(t)$ của chúng tôi khi có một tập hợp các dữ liệu được cửa sổ hóa thu được từ spectrogram được mask bởi mô hình. Do đó, phương pháp Overlap Add sẽ được chúng tôi hiện thực ở cuối đường ống lọc nhiễu động (sẽ được chúng tôi thể hiện ở Mục 6.3) như một bộ tổng hợp để trả về kết quả cho người dùng.

2.2 Biến đổi Fourier rời rạc

Trong thực tế, khi máy tính hoạt động việc tính toán biến đổi Fourier liên tục với $f(t)$ vô hạn là không thể bởi những giới hạn về phần cứng máy tính. Do vậy trong phần này, chúng tôi sẽ giới thiệu về phiên bản rời rạc hóa của biến đổi Fourier và các giải thuật được sử dụng để tính toán biến đổi Fourier rời rạc này. Giải thuật **Fast Fourier Transform (biến đổi Fourier nhanh, FFT)** được sử dụng để tính toán biến đổi Fourier được lần đầu giới thiệu bởi Cooley [6] còn được biết đến với cái tên **radix-2 Fast Fourier Transform** và sau đó được cải thiện bởi Rader và Brenner [12, 13]. Ngoài các phiên bản kề trên như radix-2 hay Rader Brenner, còn có rất nhiều phiên bản khác của giải thuật này với các hướng tiếp cận khác nhau như trong [11], tác giả sử dụng giả định chuỗi đầu vào là số thực và cắt giảm số lượng tính toán trong giải thuật xuống một nửa so với giải thuật của Cooley. Trước khi đi chi tiết vào các giải thuật để tính toán biến đổi Fourier rời rạc, chúng tôi sẽ bắt đầu với định nghĩa của biến đổi Fourier rời rạc.

2.2.1 Định nghĩa

Được nêu trong [5], biến đổi Fourier rời rạc được định nghĩa như sau

$$X(j) = \sum_{k=0}^{N-1} x_k e^{-2\pi \frac{jk}{N} i} = \sum_{k=0}^{N-1} x_k W^{jk}, \quad (2.16)$$

với x_k là phần tử thứ k trong vector đầu vào x . Để thấy nếu ta tính toán biến đổi

Fourier rời rạc cho $x \in \mathbb{R}^N$ thì số lượng phép toán tổng quát (ở đây sử dụng là phép cộng và phép nhân) sẽ là $O(N^2)$. Việc này làm cho Fourier rời rạc trở thành nút thắt trong quá trình tính toán với N rất lớn. Để giải quyết vấn đề này, nhiều nghiên cứu đã xuất hiện và trong các nghiên cứu đã được tìm hiểu, ta tìm hiểu các nghiên cứu về giải thuật **biến đổi Fourier nhanh** hay **Fast Fourier Transform**.

2.2.2 Biến đổi Fourier nhanh

Năm 1965, James Cooley [6] đã đề xuất ra giải thuật được sử dụng để tính toán biến đổi Fourier rời rạc của một dãy số phức có chiều dài N . Theo sau [6], hàng loạt các bài báo khác nhầm thể hiện rõ hơn ý tưởng của giải thuật này cũng được xuất bản [7, 10]. Tuy được cho là đã có nhiều công trình khác sử dụng giải thuật này trước khi Cooley công bố kết quả của ông [9], nhưng công trình của Cooley góp phần rất lớn vào việc rút giảm độ phức tạp sinh ra bởi tính toán biến đổi Fourier rời rạc (hiện đang là nút thắt cho rất nhiều công trình khác) còn $O(N \log N)$. Phương pháp của ông được biết đến như giải thuật biến đổi Fourier nhanh cơ số 2 hay còn được gọi là **radix-2 Fast Fourier Transform** (radix-2 FFT).

Biến đổi Fourier nhanh cơ số 2. Biến đổi Fourier nhanh cơ số 2 được chia làm hai loại: *chia theo thời gian (decimation in time)* và *chia theo tần số (decimation in frequency)*. Hai cách chia này về cơ bản khá giống nhau và với giả định $N = 2^p$, giải thuật này thực hiện theo cơ chế chia để trị, thông qua gọi đệ quy của biến đổi Fourier nhanh trên các tập con nhỏ hơn của vector đầu vào x từ đó giảm độ phức tạp xuống $O(N \log N)$.

Chia theo thời gian: Xuất phát từ vector x có độ dài $N = 2^p$ với $p \in \mathbb{Z}^+$, kết quả đầu ra mà chúng tôi mong muốn là một spectrum $X(j)$ với tất cả các giá trị j nguyên có trong $0 \leq j \leq N - 1$. Từ phần trước, chúng tôi đã cho thấy độ phức tạp của việc tính toán biến đổi Fourier rời rạc thông qua định nghĩa sẽ tiêu tốn $O(N^2)$ trên phép toán cộng và nhân số phức. Với

$$\begin{aligned} W^{j+N/2} &= e^{-2\pi \frac{j+N/2}{N} i} \\ &= e^{(-2\pi \frac{j}{N} - \pi)i} \\ &= -W^j, \end{aligned} \tag{2.17}$$

$$\begin{aligned}
 W^{2a(j+N/2)} &= e^{-2\pi \frac{2a(j+N/2)}{N} i} \\
 &= e^{(-2\pi \frac{2aj}{N} - 2\pi a)i} \\
 &= e^{-2\pi \frac{2aj}{N} i} \\
 &= W^{2aj}, \tag{2.18}
 \end{aligned}$$

$$\begin{aligned}
 W^{(2a+1)(j+N/2)} &= W^{j+N/2} W^{2a(j+N/2)} \\
 &= W^{j+N/2} W^{2aj}, \tag{2.19}
 \end{aligned}$$

$$\begin{aligned}
 W^{aN} W^{jN} &= e^{-2\pi \frac{aN}{N} i} \\
 &= e^{-2\pi ai} \\
 &= 1. \tag{2.20}
 \end{aligned}$$

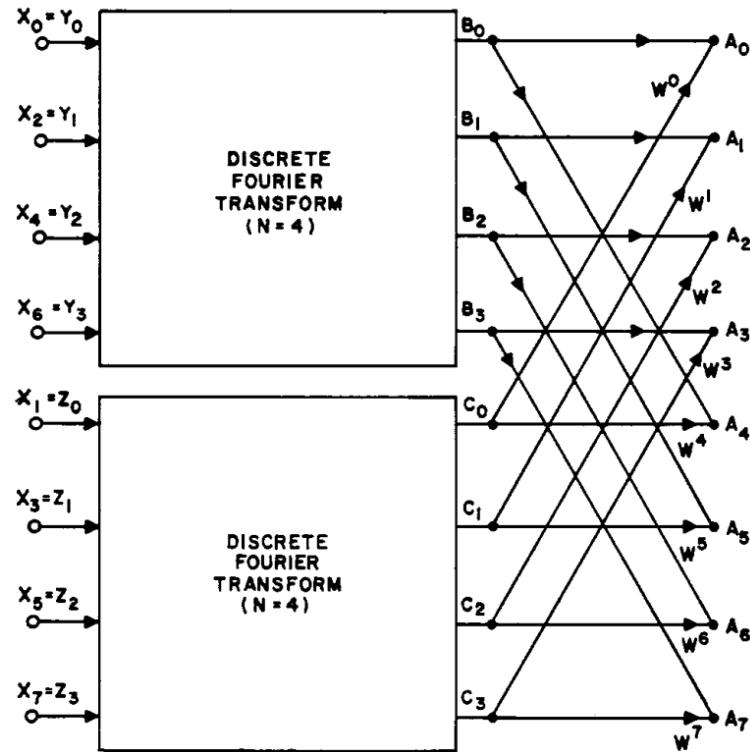
Ta có

$$\begin{aligned}
 X_j &= \sum_{k=0}^{N-1} x_k W^{kj} \\
 &= \sum_{a=0}^{\frac{N}{2}-1} x_{2a} W^{2aj} + \sum_{a=0}^{\frac{N}{2}-1} x_{2a+1} W^{(2a+1)j} \\
 &= \sum_{a=0}^{\frac{N}{2}-1} x_{2a} W^{2aj} + W^j \sum_{a=0}^{\frac{N}{2}-1} x_{2a+1} W^{2aj}, \tag{2.21}
 \end{aligned}$$

$$\begin{aligned}
 X_{j+N/2} &= \sum_{k=0}^{N-1} x_k W^{kj} \\
 &= \sum_{a=0}^{\frac{N}{2}-1} x_{2a} W^{2a(j+N/2)} + \sum_{a=0}^{\frac{N}{2}-1} x_{2a+1} W^{(2a+1)(j+N/2)} \\
 &= \sum_{a=0}^{\frac{N}{2}-1} x_{2a} W^{2aj} + W^{j+N/2} \sum_{a=0}^{\frac{N}{2}-1} x_{2a+1} W^{2aj} \\
 &= \sum_{a=0}^{\frac{N}{2}-1} x_{2a} W^{2aj} - W^j \sum_{a=0}^{\frac{N}{2}-1} x_{2a+1} W^{2aj}. \tag{2.22}
 \end{aligned}$$

Với các tính chất của X_j và $X_{j+N/2}$, mỗi biến đổi Fourier rời rạc với N mẫu sẽ được tách thành hai biến đổi Fourier nhỏ hơn với $N/2$ mẫu cho mỗi biến đổi cộng thêm N phép cộng và $N/2$ phép nhân trên số phức. Hình 2.6 thể hiện ý tưởng chia

để trị trong biến đổi Fourier nhanh, từ một biến đổi lớn thành hai biến đổi con với số lượng phần tử đầu vào bằng một nửa ban đầu.



Hình 2.6: Sơ đồ ý tưởng để quy của biến đổi Fourier nhanh (chia theo thời gian) (nguồn [10])

Từ các nhận định trên, ta thu được hệ thức truy hồi sau, nếu gọi C_N là độ phức tạp khi thực hiện biến đổi Fourier nhanh trên dữ liệu đầu vào kích thước N , ta có

$$\begin{aligned} C_N &= N + \frac{N}{2} + 2C_{N/2} \\ &= 2N + N + 4C_{N/4}. \end{aligned}$$

Tiếp tục khai triển k lần, ta thu được hệ thức truy hồi của C_N như sau

$$C_N = Nk + \frac{kN}{2} + 2^k C_{N/2^k},$$

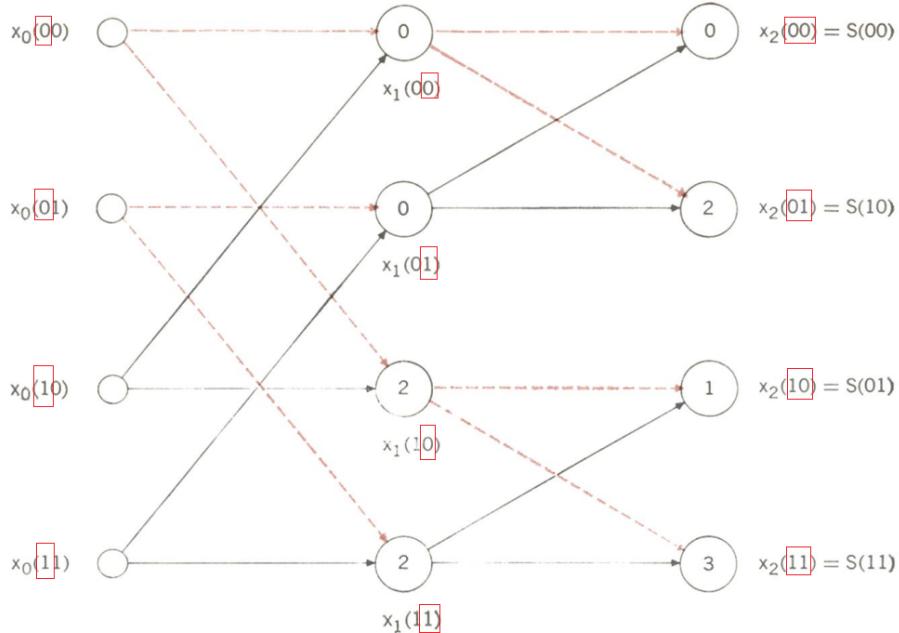
hay với $N = 2^k$, ta lại có

$$C_N = k2^k + \frac{k2^k}{2} + 2^k C_1, \quad (2.23)$$

Khi đó, với N là 1, giải thuật sẽ mất chi phí 1 phép nhân số phức nên $C_1 = 1$ nên do đó ta thu được

$$\begin{aligned} C_N &= k2^k + \frac{k2^k}{2} + 2^k C_1 \\ &= k2^k + \frac{k2^k}{2} + 2^k \\ &\leq \alpha k2^k \\ &= \alpha N \log_2 N \\ &\leq \alpha N \log N \end{aligned}$$

Dễ thấy khi N dần tiến tới vô cùng, chi phí giải thuật C_N sẽ bị chặn bởi $\alpha N \log N$ do vậy ta có giải thuật biến đổi Fourier nhanh có chi phí là $O(N \log N)$. Đây cũng chính là chứng minh cho độ phức tạp của toàn bộ thuật toán biến đổi Fourier rời rạc theo cơ số 2 này, và cũng do sự chia đôi phép tính toán Fourier được định nghĩa ra thành hai phép biến đổi Fourier con mà Cooley đã chứng minh trong bài báo gốc của ông [6].



Hình 2.7: Nguyên nhân của hiện tượng đảo bit trong biến đổi Fourier nhanh cơ số 2 (chia theo thời gian), với các bit được sắp xếp dần theo các số được khoanh trong ô màu đỏ (nguồn [8])

Biến đổi Fourier nhanh theo cơ số 2 sẽ gây ra hiện tượng đảo bit bên trong kết quả X_j nếu là hướng chia theo tần số và đảo bit trong chuỗi đầu vào x_k nếu là

hướng chia theo thời gian. Hiện tượng này được thể hiện trong Hình 2.7 và có thể được giải thích như sau. Gọi

$$k = 2^{b_{i-1}} + 2^{b_{i-2}} + \cdots + 2^{b_0}.$$

Nhờ đó, bằng việc kiểm tra xem giá trị x_k thuộc chẵn hay lẻ tương ứng với việc đang xét tới giá trị b_0 , các số k có b_0 là bit 0 ứng với chẵn sẽ nằm về A_j trong khi các số có b_0 là bit 1 ứng với số lẻ sẽ thuộc về B_j , việc này tiếp tục tới b_1 bởi vì những số có b_1, b_0 ứng với 0 và 0 sẽ nằm ở vị trí chẵn bên trong A_j và nếu số có b_1, b_0 ứng với 1 và 0 sẽ nằm ở vị trí lẻ trong A_j . Tương tự như vậy, các số có b_1, b_0 ứng với 0 và 1 sẽ nằm ở vị trí chẵn và nếu b_1 là 1 sẽ nằm ở vị trí lẻ trong B_j . Cứ như vậy quá trình này tính toán tới giá trị của b_i , và một điều kiện tương tự lại diễn ra. Bằng cách ghi nhận lại thứ tự bit được lấy ra và thứ tự bit lấy ra, chúng tôi dễ dàng quan sát được sự đảo bit bên trong giải thuật biến đổi Fourier rời rạc cơ số 2 này.

Chia theo tần số: Tương tự như cách chia theo thời gian, để thực hiện biến đổi Fourier nhanh chúng tôi cũng bắt đầu từ một cặp giá trị của X , lần này là X_{2j} và X_{2j+1} , với

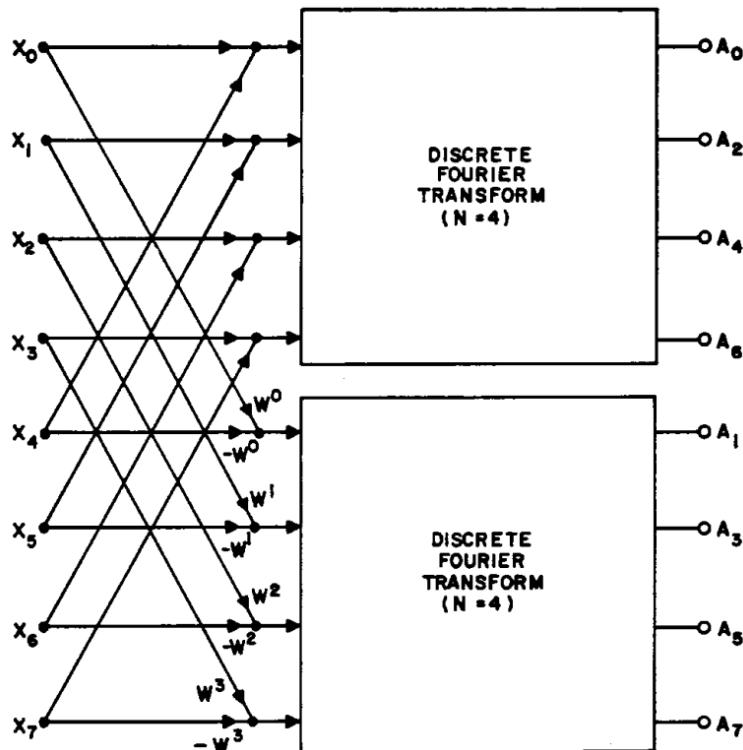
$$W^{jN} = 1. \quad (2.24)$$

có

$$\begin{aligned} X_{2j} &= \sum_{k=0}^{N-1} x_k W^{2jk} \\ &= \sum_{k=0}^{\frac{N}{2}-1} x_k W^{2jk} + \sum_{k=0}^{\frac{N}{2}-1} x_{k+N/2} W^{2j(k+N/2)} \\ &= \sum_{k=0}^{\frac{N}{2}-1} x_k W^{2jk} + \sum_{k=0}^{\frac{N}{2}-1} W^{jN} x_{k+N/2} W^{2jk} \\ &= \sum_{k=0}^{\frac{N}{2}-1} (x_k + x_{k+N/2}) W^{2jk}, \end{aligned} \quad (2.25)$$

$$\begin{aligned}
 X_{2j+1} &= \sum_{k=0}^{N-1} x_k W^{(2j+1)k} \\
 &= \sum_{k=0}^{\frac{N}{2}-1} W^k x_k W^{2jk} + \sum_{k=0}^{\frac{N}{2}-1} W^{k+N/2} x_{k+N/2} W^{2j(k+N/2)} \\
 &= \sum_{k=0}^{\frac{N}{2}-1} W^k x_k W^{2jk} + \sum_{k=0}^{\frac{N}{2}-1} W^{k+N/2} W^{jN} x_{k+N/2} W^{2jk} \\
 &= \sum_{k=0}^{\frac{N}{2}-1} (x_k - x_{k+N/2}) W^k W^{2jk}, \tag{2.26}
 \end{aligned}$$

Bằng các tính chất nêu ở công thức (2.25) và (2.26), chúng tôi có thể xây dựng lại được giải thuật có độ phức tạp tương đương với giải thuật biến đổi Fourier nhanh được chia theo thời gian bằng hệ thức hồi quy của hai biến đổi Fourier rời rạc với dữ liệu mới là $(x_k - x_{k+N/2})$ và $W^k(x_k - x_{k+N/2})$.



Hình 2.8: Sơ đồ ý tưởng đê quy của biến đổi Fourier nhanh (chia theo tần số) (nguồn [10])

Hình 2.8 thể hiện việc chia từ một biến đổi Fourier nhanh với đầu vào có N mẫu về hai biến đổi Fourier nhanh nhỏ hơn với đầu vào có kích thước $N/2$. Hai cách

chia chỉ khác ở thành phần được xét chẵn hay lẻ nằm ở miền thời gian ($x(t)$) hay mở miền tần số ($X(\omega)$).

Cải tiến của Rader và Brenner. Như cũng đã thấy rõ trong cả phiên bản chia theo thời gian và chia theo tần số của biến đổi Fourier nhanh cơ số 2, các phép nhân số phức xảy ra trong giải thuật chỉ xuất hiện ở một trong hai biến đổi Fourier con của giải thuật gốc ban đầu. Nhưng so sánh với phép cộng số phức, phép nhân phức tạp hơn nhiều và cũng yêu cầu nguồn tài nguyên tính toán gấp đôi phép cộng số phức do vậy giảm lượng phép nhân trong giải thuật là một điều cần thiết. Nhận thấy điều này, trong [12], Rader và Brenner đã đề xuất ra thêm một số bước biến đổi thực hiện trên cơ sở của giải thuật biến đổi Fourier nhanh cơ số 2 chia theo thời gian để cắt giảm lượng phép nhân trong giải thuật này. Như ta đã thấy ở phần trên, giá trị của $X_{j+N/2}$ sẽ được tính toán như sau

$$\begin{aligned} X_{j+N/2} &= \sum_{a=0}^{\frac{N}{2}-1} x_{2a} W^{2aj} + W^j \sum_{a=0}^{\frac{N}{2}-1} x_{2a+1} W^{2aj} \\ &= B_{2j} - W^j D_{2j}. \end{aligned} \quad (2.27)$$

Dễ thấy phép nhân số phức bắt nguồn từ $W^j D_{2j}$, để có thể giảm đi lượng phép nhân, Rader và Brenner đề xuất một cách biến đổi như sau. Gọi $c_a = x_{2a+1} - x_{2a-1}$, chúng tôi có thể tính được giá trị của biến đổi Fourier rời rạc C_{2j} của các c_k

$$\begin{aligned} C_{2j} &= \sum_{a=0}^{\frac{N}{2}-1} c_a W^{2aj} \\ &= \sum_{a=0}^{\frac{N}{2}-1} x_{2a+1} W^{2aj} - \sum_{a=0}^{\frac{N}{2}-1} x_{2a-1} W^{2aj} \\ &= D_{2j} - W^{2j} \sum_{b=0}^{\frac{N}{2}-1} x_{2b+1} W^{2bj} \\ &= D_{2j}(1 - W^{2j}) \\ &= -W^j D_{2j}(W^j - W^{-j}). \end{aligned} \quad (2.28)$$

Thông qua công thức (2.28), ta dễ dàng thu được công thức của $-W^j D_{2j}$ theo C_{2j}

$$W^j D_{2j} = -\frac{1}{W^j - W^{-j}} C_{2j}. \quad (2.29)$$

Bằng cách thay công thức (2.29) vào công thức (2.27), ta thu được biến đổi Fourier với phần nhân số phức cắt giảm

$$\begin{aligned} X_j &= B_{2j} + W^j D_{2j} \\ &= B_{2j} - \frac{1}{W^j - W^{-j}} C_{2j} \\ &= B_{2j} - \frac{1}{\sin(2\pi j)i} C_{2j} \\ &= \sum_{a=0}^{\frac{N}{2}-1} x_{2a} W^{2aj} - \frac{1}{\sin(2\pi j)i} \sum_{a=0}^{\frac{N}{2}-1} (x_{2a+1} - x_{2a-1}) W^{2aj}. \end{aligned} \quad (2.30)$$

Bằng việc phân tách $W^j D_{2j}$, số lượng phép nhân bây giờ đã được suy giảm còn lại phép chia các số thuần ảo. Nhưng có một vấn đề với $\sin(2\pi j)$, sẽ có một vài điểm mang giá trị 0 nên sẽ gây bất hợp lý cho công thức trên, do vậy ở các điểm giá đoạn, chúng tôi sẽ đặt một số giá trị Q được tính toán từ công thức gốc $W^j D_{2j}$ thay vì sử dụng công thức (2.30).

Chương 3

Âm học

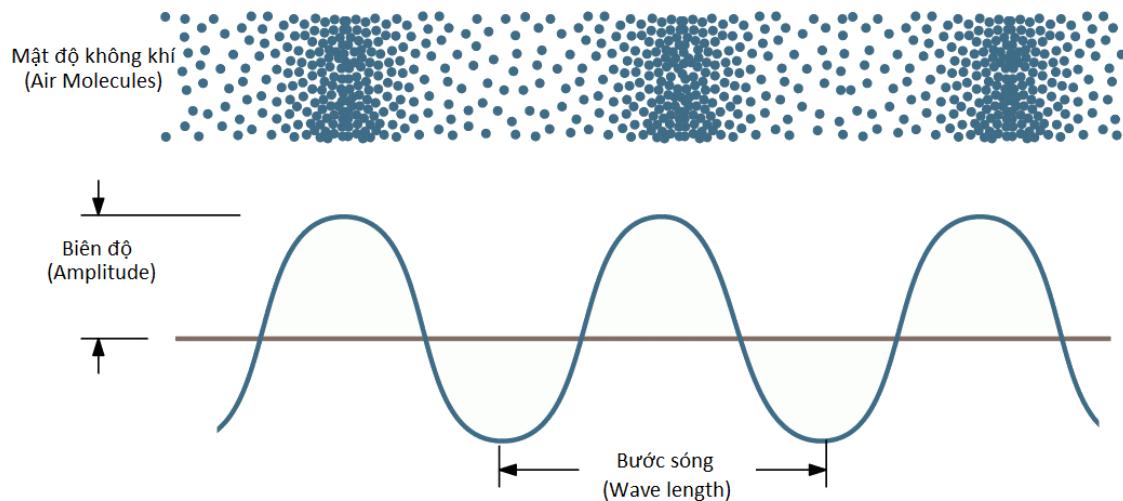
Trong phần này, ta tìm hiểu về âm thanh, cấu tạo của âm thanh và các hình thức biểu diễn của nó. Bên cạnh đó, ta cũng sẽ đề cập các khái niệm nền tảng của âm thanh như tần số cảm nhận và độ to để từ đó đề cập tới các metrics thông dụng sẽ được sử dụng trong luận văn này.

3.1 Âm thanh

Sóng âm hay **âm thanh** về bản chất là một sóng dao động cơ cưỡng bức của các phân tử không khí va chạm vào nhau, xô đẩy tạo nên sự thay đổi về mặt mật độ phân tử, từ đó gây ra hiện tượng lan truyền của sóng âm. Vậy để bắt đầu, ta sẽ tìm hiểu các khái niệm liên quan đến **sóng âm**.

3.1.1 Sự hình thành sóng âm

Sóng âm được biểu diễn bao gồm ba thành phần chính bao gồm: *biên độ A*, *tốc độ góc ω* và *pha ban đầu φ*. Một sóng đơn giản nhất có thể kể đến đó là sóng cosine trong dao động điều hòa $A \cos(\omega t + \phi)$. Biên độ càng lớn, các phân tử không khí dao động càng mạnh tạo nên sự cách biệt giữa các vùng có mật độ không khí thưa và các vùng có mật độ có không khí dày, làm đồ thị sóng âm có biên độ lớn hơn là khi biên độ sóng phát ra từ nguồn thấp.

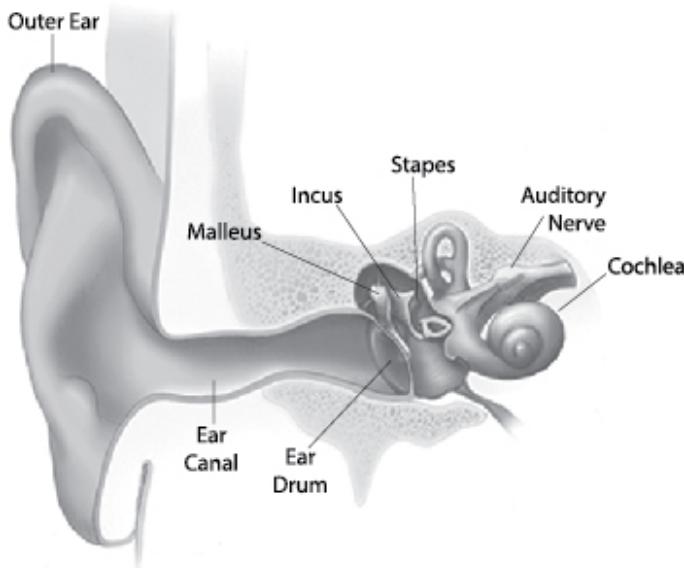


Hình 3.1: Biểu đồ sóng âm thể hiện mật độ không khí thay đổi như thế nào trong quá trình truyền âm thanh

Tương tự như biên độ, tốc độ góc cũng đóng một vai trò rất quan trọng trong sự hình thành của sóng âm. Các âm có tốc độ góc lớn (tương ứng với tần số dao động lớn vì hai đại lượng này quan hệ tỉ lệ thuận với nhau thông qua công thức $\omega = 2\pi f$ với f là tần số) tương ứng với sự dao động nhanh hơn của các phân tử không khí do đó làm cho âm nghe có vẻ cao hơn những âm dao động ở tốc độ góc thấp. Đại lượng cuối cùng quy định tính chất của âm thanh đó chính là góc ban đầu của âm thanh.

Sóng âm thanh mà chúng tôi vừa đề cập là một **sóng âm đơn (pure tone)**, nhưng thực tế, sóng âm thanh mà mọi người nghe được mỗi ngày phức tạp hơn rất nhiều. Các **sóng âm phức tạp (complex tone)** được xem là tổng hợp của rất nhiều sóng âm đơn, sự tổng hợp này có thể được giữ nguyên (trong trường hợp âm thanh ít biến đổi) và cũng có thể biến đổi liên tục (trong trường hợp giọng nói) phụ thuộc vào bản chất của âm thanh này. Tai người từ lâu trong xử lý tín hiệu âm thanh đã luôn là một chuẩn mực để đem đi so sánh với các mô hình và metrics được sử dụng, với lý do này, ta sẽ phân tích cách mà tai người tiếp nhận âm thanh và sự liên hệ với biến đổi Fourier như đã được đề cập ở Chương 2.

Trước tiên, âm thanh sau khi được nguồn âm phát ra, lan truyền đi trong không khí và tới được tai ngoài và từ đó, các phân tử không khí tiếp tục truyền đi bên trong ống tai và va chạm vào màng nhĩ làm cho màng nhĩ dao động. Các dao động



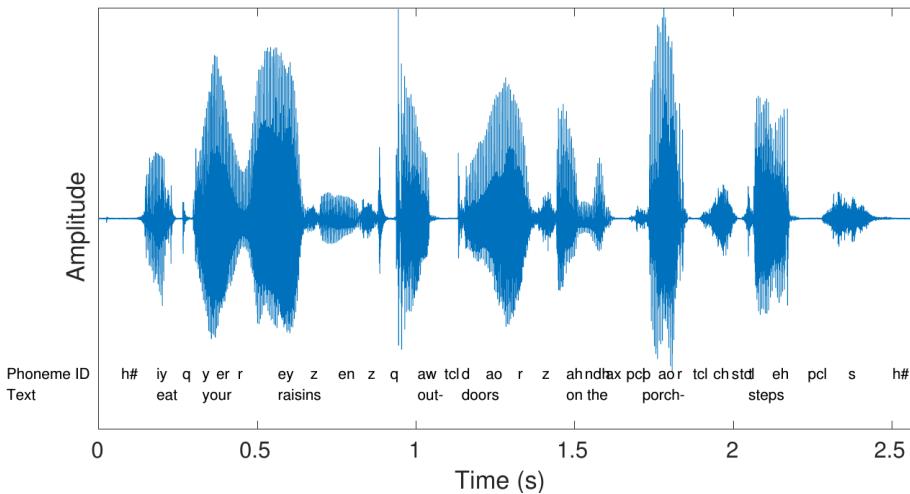
Hình 3.2: Cấu tạo của tai người

từ màng nhĩ làm dao động các xương nhỏ Incus, các xương này dao động và truyền dao động đó qua một ổ chứa dịch đặc là Cochlea. Đây cũng chính là nơi mà sự tương đồng trong cơ chế của biến đổi Fourier được thể hiện. Ổ dịch này được kết nối trực tiếp với dây thần kinh thính giác, và trong môi trường chất lỏng như vậy, các âm có tần số thấp chỉ có khả năng truyền đi trong một khoảng rất ngắn nhưng các âm với tần số cao thì lại khác, các âm này có thể truyền đi rất xa trong môi trường lỏng. Các sóng có tần số cao sẽ đạt tới các dây thần kinh cảm nhận ở sâu bên trong Cochlea, trong khi các sóng có tần số thấp hơn sẽ chỉ đạt tới dây thần kinh cảm nhận ở gần hơn. Thông qua việc tiếp nhận âm thanh như vậy, Cochlea đóng vai trò như một bộ biến đổi Fourier sinh học phân tách sóng âm nhận được từ nguồn âm thành rất nhiều những sóng đơn với tần số khác nhau có biên độ khác nhau. Đó cũng chính cách mà con người cảm nhận được âm thanh.

3.1.2 Các biểu diễn của âm thanh

Ở dạng cơ bản, sóng âm là một hàm theo thời gian $f(t)$ và đây cũng chính là dạng biểu diễn đầu tiên của sóng âm - biểu diễn trong miền thời gian. Ở miền này, sóng âm được thể hiện dưới dạng một biểu đồ sóng theo thời gian được gọi là **waveform**. Hình 3.3 minh họa cho một ví dụ của âm thanh biểu diễn dưới dạng

này.



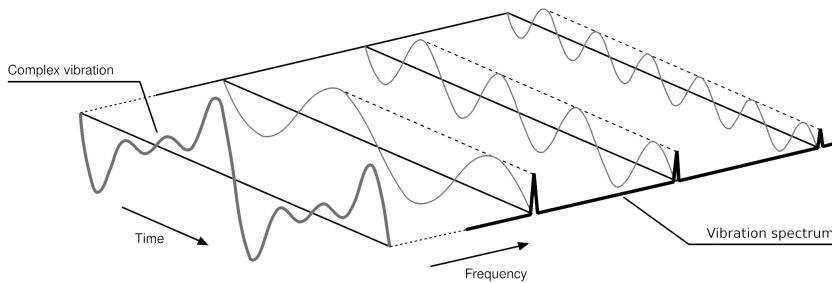
Hình 3.3: Waveform của câu “eat your raisins out-doors on the porch steps”

Như ở Tiêu mục 3.1.1, ta đã tìm hiểu rằng mỗi sóng âm phức tạp (complex tone) luôn là một tập hợp của rất nhiều sóng âm khác

$$f(t) = \sum_j r_j \cos(\omega_j + \phi_j),$$

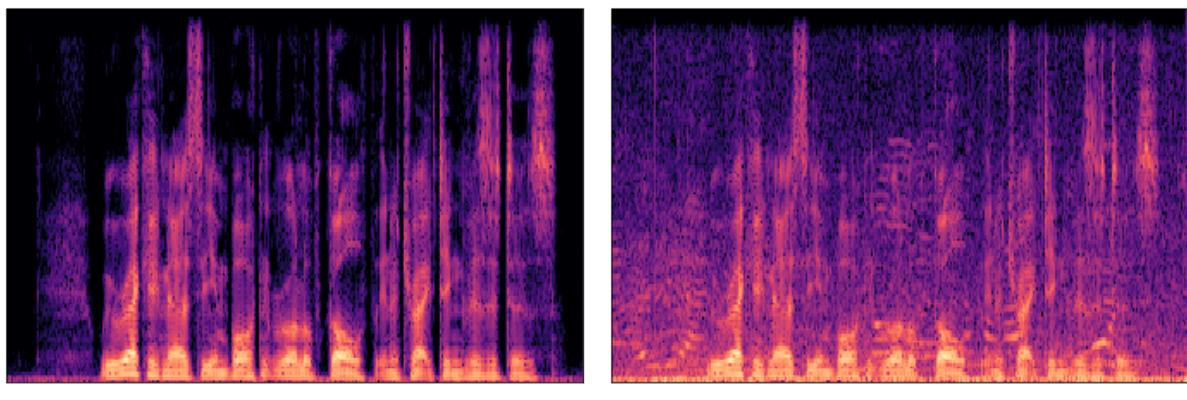
tuy nhiên, sóng âm hiện tại vẫn đang ở dạng trộn lẫn vào nhau, ở dạng thời gian để có thể xác định có những tần số nào đang tồn tại bên trong âm thanh tổng hợp là rất khó. Do đó ta tới với cách biểu diễn tiếp theo của sóng âm, biểu diễn theo tần số. Biểu diễn này chính là biên độ của tâm vật thể tạo bởi $f(t)e^{-\omega t i}$. Thể hiện các biên độ này trên một khoảng ω đủ lớn, kết quả là ta thu được biểu diễn của âm thanh theo miền tần số. Biểu đồ đó được gọi là **spectrum**. Hình 3.4 minh họa cho sự biến đổi qua lại giữa hai miền tần số và thời gian thông qua biến đổi Fourier thuận và nghịch.

Nhưng dễ thấy, bản chất giọng nói không hề có sự ổn định về mặt tần số cấu tạo theo thời gian. Một giọng nói hoàn toàn có thể bị thay đổi, làm giọng cao hơn, trầm đi, các yếu tố này đều chịu sự ảnh hưởng của người nói và thông tin họ muốn truyền tải tại thời điểm đó. Biến đổi Fourier thông thường không chú trọng vào đặc điểm này, biến đổi Fourier xét cấu tạo của âm thanh trên toàn bộ miền thời gian do đó sẽ làm mất đi bản chất biến đổi liên tục của tần số cấu tạo giọng nói. Để khắc phục điều này, thay vì sử dụng biến đổi Fourier trên toàn bộ $f(t)$, ta sẽ



Hình 3.4: Spectrum minh họa biểu diễn âm thanh trên miền tần số

sử dụng phép biến đổi Fourier lên một khoảng thời gian nhỏ trong $f(t)$ và đó cũng chính là cách biểu diễn thứ ba của âm thanh. Miền tần số thời gian, miền này được đặc trưng bởi ***spectrogram***.



(a) Spectrogram của giọng nói sạch

(b) Spectrogram của giọng nói có chứa nhiễu

Hình 3.5: Spectrogram minh họa biểu diễn âm thanh trên miền tần số thời gian

Thông qua Hình 3.5, các tính chất biến đổi của biến độ tần số cấu tạo theo thời gian được thể hiện khá rõ, spectrogram biểu diễn âm thanh khá tốt và có nhiều thông tin hơn về giọng nói cũng như âm nhiễu được pha vào nó, điều này cho phép mô hình tự do hơn trong việc lựa chọn các thuộc tính cần thiết để loại bỏ nhiễu ra khỏi âm thanh này. Đây cũng chính là cách biểu diễn mà sinh viên thực hiện lựa chọn làm dữ liệu đầu vào để huấn luyện mô hình. Ngoài ra cũng có rất nhiều cách biểu diễn khác, một số cho phép mô hình tự chọn cách biểu diễn của riêng mình, một số thì tự quy định ra những thuộc tính có hình dạng đặc biệt như trong biến đổi wavelet hoặc biến đổi Laplace.

3.1.3 Dãy khoảng tám và dãy khoảng tám lẻ

Trong lý thuyết nhạc, khi một khoảng tần số đại diện cho sự di chuyển của tần số một nốt lên một tần số khác gấp đôi nó, ví dụ như từ cao độ nốt Đồ lên nốt Đỗ, khoảng cách cao độ giữa hai nốt này được gọi là một khung nhạc hay **một khoảng tám (octave)**. **Dãy khoảng tám (octave band)** là một tập hợp các khoảng tám như vậy liền kề nhau xuyên suốt khoảng tần số nghe được từ 20 Hz tới 20000 Hz. Trong mỗi khoảng tám, luôn có một tần số trung tâm được gọi là f_c , các tần số biên f_{low} và f_{high} đại diện cho tần số biên dưới và tần số biên trên có thể được tính xấp xỉ theo công thức (3.1).

$$f_c = \sqrt{2} f_{low} = \frac{f_{high}}{\sqrt{2}}. \quad (3.1)$$

Được quy định theo chuẩn ANSI S1.11 [35], mỗi tần số trung tâm f_c trong dãy khoảng tám thứ x có thể được tính toán xấp xỉ bằng công thức

$$f_c = G^{\frac{x-30}{b}} \times 1000, \quad (3.2)$$

với

- b là số tần số lấy trong một dãy khoảng tám, đối với trường hợp $b = 1$, dãy khoảng tám được xét là tần số trung tâm của một khoảng tám, $b = 2$ là của nửa khoảng tám (half octave) và $b = 3$ đối với trường hợp của một phần ba khoảng tám (one-third octave).
- G là một hằng số tùy thuộc vào hệ đang xét, đối với công thức trên của chúng tôi, hệ cơ số 2 đang được xét do đó G trong trường hợp này mang giá trị $G = G_2 = 2$ và đối với trường hợp của cơ số 10, G sẽ mang giá trị $G_{10} = 10^{0.3}$.

Vì hệ cơ số 10 thường được sử dụng hơn nên ta sẽ tiến hành đưa ra công thức của hệ cơ số 10 từ công thức (3.2),

$$\begin{aligned} f_c &= G_2^{\frac{x-30}{b}} \times 1000 \\ &\approx G_{10}^{\frac{x-30}{b}} \times 10^3 \\ &= 10^{0.3 \frac{x-30}{b} + 3}. \end{aligned} \quad (3.3)$$

Đối với trường hợp $b = 3$, công thức tần số trung tâm của một phần ba dãy khoảng tám (one-third octave band) theo hệ cơ số 10 trở thành

$$f_c = 10^{0.1x}, \quad (3.4)$$

đơn giản hơn rất nhiều so với việc tính toán với hệ cơ số 2. Tuy vậy về kết quả xấp xỉ của f_c , hai công thức (3.2) và (3.3) là giống hệt nhau. Các octave bands này được sử dụng phổ biến và là tiền đề cho các lý thuyết về độ cảm nhận âm thanh của tai người, đóng vai trò quan trọng trong các metrics của chúng tôi sử dụng.

3.1.4 Tần số cảm nhận

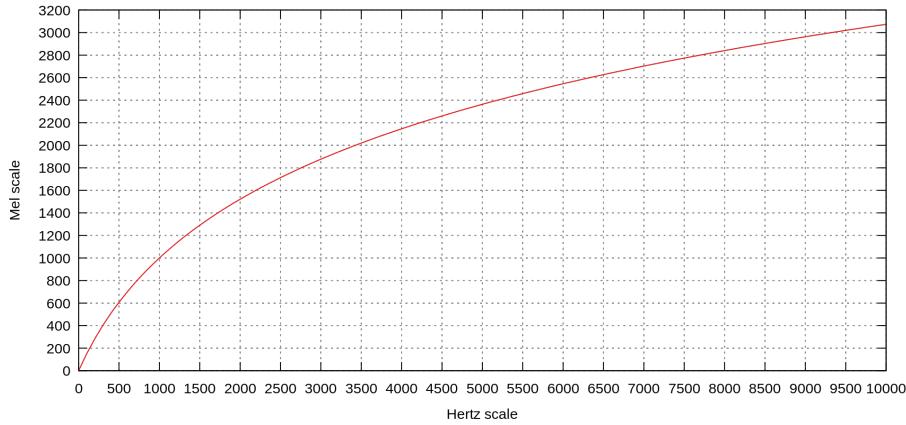
Nhờ vào cơ chế nghe ở tai mà con người có khả năng nhận biết âm thanh và hiểu được nó. Nhưng sự nhạy cảm của tai người đối với các loại âm thanh khác nhau lại tuân theo quy luật phi tuyến. Quy luật phi tuyến này được biết tới như là **tần số cảm nhận**. Tần số cảm nhận có khá nhiều loại, nhưng trong luận văn này, chúng tôi sẽ đề cập tới hai loại tần số được sử dụng trong luận văn là tần số Mel và tần số Bark. Tùy thuộc vào loại tần số cảm nhận mà chúng tôi đang sử dụng, sẽ có những quy luật để tổng hợp các trọng số trong một khoảng về một giá trị tại một tần số đại diện nhất định, giá trị tần số đại diện này được gọi là một băng tần hay một band.

Tần số Mel. Tần số Mel [32] đặc trưng cho sự nhạy cảm của tai trước một âm thanh có tần số bất kì f , sự nhạy cảm này có thể được tính toán bằng

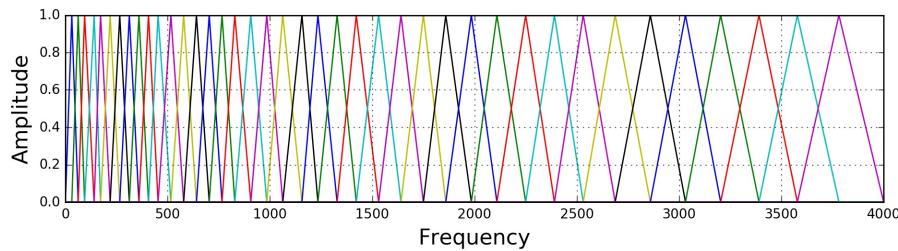
$$m(f) = 1127 \ln \left(1 + \frac{f}{700} \right). \quad (3.5)$$

Càng lên cao, các khoảng Mel cố định sẽ dẫn tới các khoảng dài hơn trên thang tần số Hz, để thuận tiện trong việc hiện thực tổng hợp tần số trong các khoảng tần số Mel, sự biến đổi này có thể được quy đổi thành một tập hợp các bộ lọc được gọi là **Mel filter bank**. Như trong Hình 3.7, các hình tam giác đó đại diện cho những khoảng tần số (phần cạnh đáy tam giác) được quy về chung một giá trị trong tần số Mel (đỉnh của tam giác). Các giá trị tam giác này ứng với sự tổng hợp biên độ của các tần số nằm trong vùng tam giác với trọng số tương ứng để

tạo thành giá trị biên độ cho tần số Mel được quy đổi.



Hình 3.6: Biểu đồ biểu diễn tần số Mel theo tần số Hz



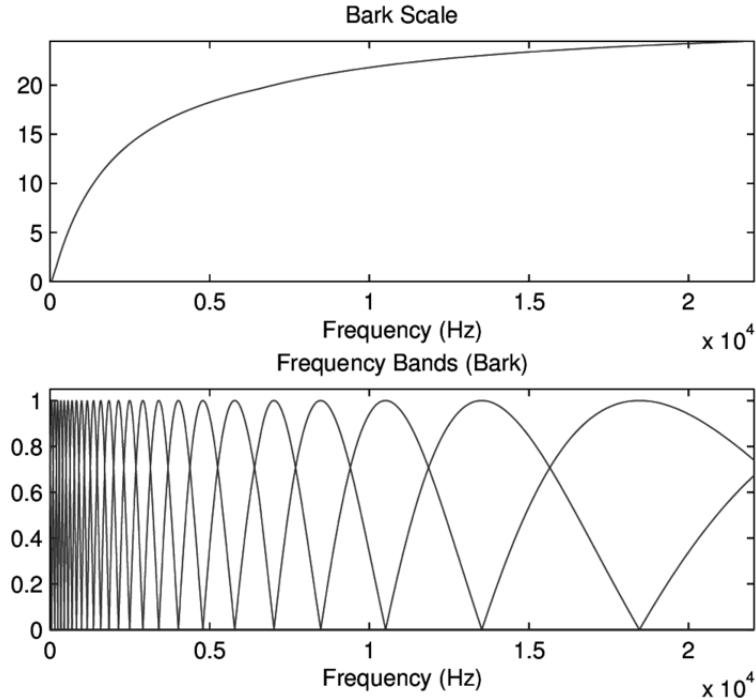
Hình 3.7: Mel filterbank

Tần số Bark. Tần số Bark [36] có thể được xem như một phiên bản khác của tần số Mel. Tần số Bark có thể được chuyển đổi từ tần số Hz theo công thức sau

$$b = 6 \sinh^{-1} \left(\frac{f}{600} \right), \quad (3.6)$$

với f là tần số ở Hz và b là tần số Bark.

Cũng như Mel, trong thực tế, chúng tôi không sử dụng trực tiếp công thức quy đổi Hz về Bark mà thay vào đó sinh viên thực hiện sẽ sử dụng một tập hợp các bộ lọc và thông qua các bộ lọc này tổng hợp các tần số từ một khoảng tần số Hz ứng với một mức nghe tương ứng. Do có mặt hình học khá tương tự như tần số Mel, mà Bark cũng sẽ có những tính chất tương tự. Hình 3.8 thể hiện sự biến đổi giữa Hz và Bark cũng như bộ lọc Bark filter bank.



Hình 3.8: Tần số Bark và Bark filterbank

3.1.5 Thang đo độ to của âm thanh

Xuất phát từ câu hỏi “*Mọi người sẽ đánh giá độ to như thế nào khi ta hỏi họ đánh giá bằng một thang đo định lượng thay vì sử dụng các tính từ?*”, trong [38] do Stevens đặt ra, rất nhiều bài báo khác được phát triển dựa trên nền tảng về “độ to” này như Zwicker [42, 36], Stevens [39], Lochner [40], ... Độ to của âm thanh và cách con người cảm nhận nó lần đầu được định lượng hóa về một thang đo chuẩn năm 1936 bởi Stevens [37]. Trước bài báo này, một thang đo khác dùng để xét về **mức độ to** của âm đơn vị **phon** được đề xuất. Thang đo phon có thể được diễn tả như sau.

Với $G(I, f)$ là hàm độ to (lúc này họ vẫn chưa tìm được định nghĩa của hàm tính độ to này) được sử dụng dựa trên cường độ âm I và tần số f . Vậy, mức độ to L của âm đang xét được định nghĩa là

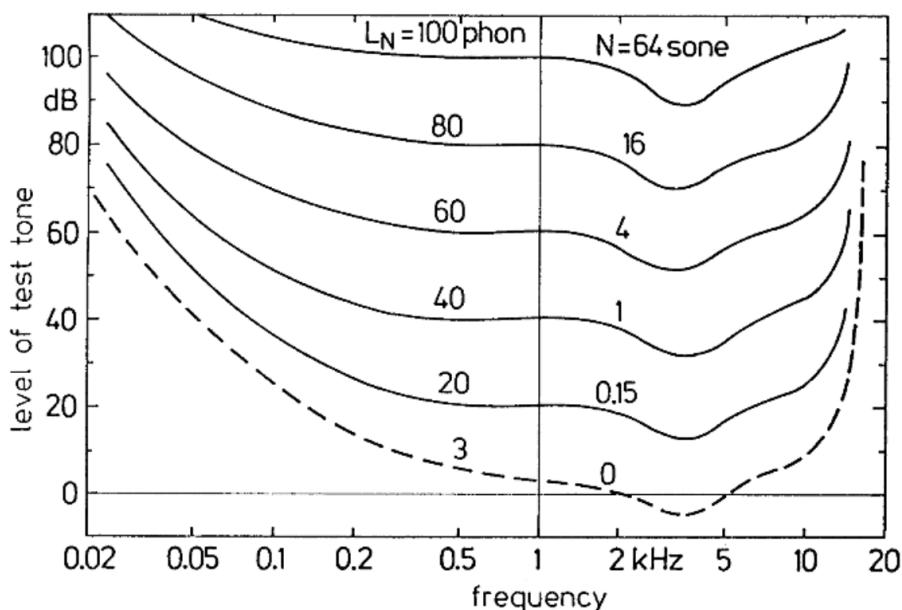
$$G(I, f) = G(L, 1000), \quad (3.7)$$

hay mức độ to của âm này là mức cường độ âm mà tại đó âm thuần (pure tone) có

tần số 1000 Hz nghe “cứng to” như âm đang được xét (cường độ âm của ta đang xét tính theo đơn vị dB được lấy theo giá trị cường độ âm thật I_r theo công thức $10 \log_{10}(I_r/I_0)$ với $I_0 = 10^{-12} W/m^2$). Mỗi phon được quy định là mức độ to của âm thuần 1000 Hz với cường độ âm là 1 dB, nhờ đó mà mức độ to cũng có thể được hiểu là cường độ âm I của âm đang được xét. Cũng từ khái niệm này, họ cũng đề ra khái niệm về **đường mức cùng độ to (equal loudness contour)**, đường này có công thức tổng quát được thể hiện như công thức sau, với $\forall I \in \mathbb{R}, f \in \mathbb{R}^+$ cho trước

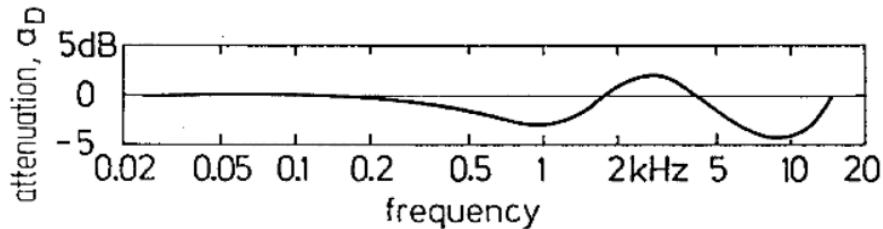
$$G(I, f) = G(L, 1000). \quad (3.8)$$

Hình 3.9 thể hiện đường mức (cùng độ to) được tìm ra từ thực nghiệm. Ở các tần số thấp, đường mức (cùng độ to) này có dạng giống như hàm mũ nhưng khi ở các tần số cao hơn, đặc biệt là ở điểm nhạy cảm của tai tại khoảng từ 2 tới 5 kHz, đường độ to có xu hướng giảm mạnh trở lại trước khi tăng lên tại giới hạn ngưỡng nghe 20 kHz. Các đường mức (cùng độ to) ở Hình 3.9 được thực nghiệm trong điều kiện tiêu chuẩn khi đo đặc độ to, âm thanh được truyền từ khoảng cách 1 mét trước đường trung trực của đường thẳng được nối từ hai tai của người được đo.



Hình 3.9: Đường mức cùng độ to trong môi trường định hướng (nguồn [36])

Tuy nhiên trong môi trường mở, nơi mà âm thanh tiến tới tai từ mọi hướng (diffuse sound field), độ to của âm tại các tần số lớn bị biến đổi so với đường cùng biên độ được trình bày trong Hình 3.9. Sự biến đổi độ to của các âm được thể hiện trong Hình 3.10. Kết hợp đường mức (cùng độ to) và đường biến thiên độ to được nêu, dễ dàng suy ra được đường mức (cùng độ to) cho môi trường mở này.



Hình 3.10: Sự biến thiên độ to theo tần số trong môi trường mở (nguồn [36])

Thang đo độ to được đề xuất bởi Stevens [37] năm 1936 theo đơn vị **sone** được định nghĩa như sau

$$G(40, 1000) = 1. \quad (3.9)$$

Mỗi sone được xem độ to của âm 40 dB với tần số 1000 Hz. Với sự quy chuẩn này, các công trình khác bắt đầu được thực hiện. Trong [38], Stevens đề xuất một hàm xấp xỉ độ to của âm thuần với tần số 1000 Hz với cường độ âm $I > 40$ dB như sau

$$G(I, 1000) = 0.06I_r^{0.3}, \quad (3.10)$$

hay

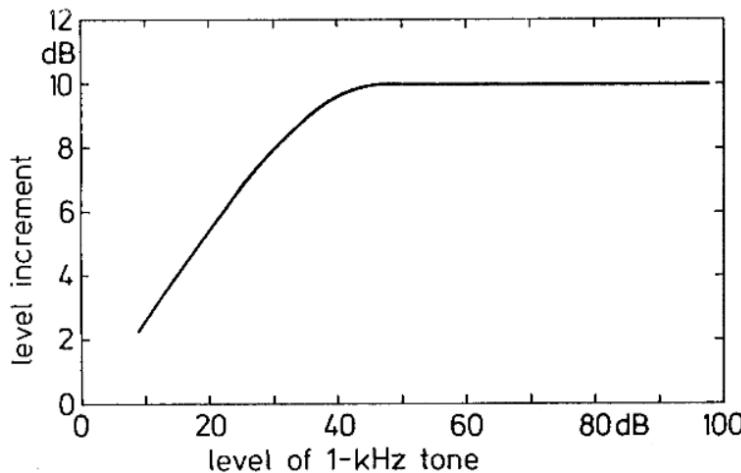
$$\log_{10}(G(I, 1000)) = 0.3L - 1.2. \quad (3.11)$$

Công thức này sau đó, thông qua một thử nghiệm về việc tăng giảm âm lên xấp xỉ hai lần âm thuần 1000 Hz đang được nghe. Sau Steven, Zwicker [36] cũng thực hiện lại thử nghiệm này và cùng thể hiện xấp xỉ chung một kết quả. Những người thực hiện cho thấy một quy luật được thể hiện trong Hình 3.11. Ở khoảng $I \in [0, 40]$ dB, mức tăng cường độ âm tăng mạnh nhưng sau khi vượt ngưỡng 40 dB, các người thực hiện cho thấy mức tăng ổn định ở 10 dB thì sẽ cho kết quả độ

to tăng lên hai lần, qua thực nghiệm trên Zwicker nêu trong [36], công thức độ to với $I > 40$ dB sẽ có thể được xấp xỉ bằng

$$G(I, 1000) = 2^{(L-40)/10} \approx 0.06I_r^{0.3} \quad (3.12)$$

Công thức của Zwicker hoàn toàn có thể được xấp xỉ công thức của Stevens thông qua $L \approx \log_{10}(I)$ từ đó xác nhận nhận định của Stevens của độ to của âm thuần với cường độ âm lớn hơn 40 dB.



Hình 3.11: Biểu đồ biến đổi của tăng giảm cường độ để âm nghe được tăng xấp xỉ 2 lần âm ban đầu (nguồn [36])

Nhưng việc đánh giá độ to của âm thuần 1000 Hz với cường độ âm trong khoảng $[0, 40]$ dB vẫn chưa thể đạt tới kết luận. Theo [40], có rất nhiều dạng hàm số biến thể của công thức (3.10) được đề xuất. Cuối cùng, sử dụng các nghiên cứu của Zwicker, ISO đặt ra tiêu chuẩn để tính toán độ to của âm thuần 1000 Hz được quy định trong ISO 532-1 [43], một hàm phân nhánh được sử dụng

$$G(I, 1000) = \begin{cases} 2^{(L-40)/10}, & \text{nếu } L > 40dB, \\ (L/40 - 0.0005)^{2.86}, & \text{nếu } L \leq 40dB. \end{cases} \quad (3.13)$$

Tuy nhiên, như vậy vẫn chưa đủ, trong thực tế, âm thanh được cấu tạo từ rất nhiều sóng thuần với nhau. Rõ ràng công thức của Zwicker và Stevens chỉ mới thỏa mãn một phần của vấn đề. Xuất phát từ điều này, Stevens [39] đã đề xuất ra một hàm tổng hợp độ to từ các độ to đơn lẻ của các âm thuần. Theo Stevens, hai âm

thuần phát ra riêng biệt sẽ luôn có độ to lớn hơn khi chúng được phát chung với nhau, ông gọi hiện tượng này là **suy giảm kích thích (inhibition stimulus)**. Công thức của ông do vậy cũng được định nghĩa như sau

$$S = S_{max} - k(\sum S_i - S_{max}), \quad (3.14)$$

với

- S_{max} là độ to lớn nhất có giá trị là $S_{max} = \max S_i$.
- S_i là độ to riêng của từng octave band (đôi khi cũng được sử dụng trên half octave band và one third octave band).
- k là hằng số phụ thuộc vào loại băng tần đang xét (octave band, half octave band và one third octave band, k lần lượt là 0.3, 0.2 và 0.13).

Tuy nhiên, với Zwicker, ông lại tiếp cận theo một hướng khác được đề xuất trong [36, 42]. Các kích thích trong tai khi nghe sẽ tỉ lệ với độ to mà con người cảm nhận, xuất phát từ điều này, Zwicker xây dựng công thức tính toán độ to được tổng hợp dựa trên các độ kích thích (excitation level). Bắt đầu từ công thức của đạo hàm của hai vế công thức (3.11), chúng tôi gọi $N = G(L, 1000)$ đại diện cho độ to của âm,

$$\begin{aligned} d(\log_{10} N) &= d(k' \log_{10} I) \\ \frac{dN}{N} &= k' \frac{dI}{I} \\ \approx \frac{\Delta N}{N} &= k' \frac{\Delta I}{I}. \end{aligned} \quad (3.15)$$

Sử dụng dạng được nêu trong công thức (3.15), Zwicker đề xuất một khái niệm về độ to được sử dụng trên mức độ kích thích của một khung tần số trong tần số cảm nhận Bark, N lúc này được ông đổi lại thành N' và gọi là **độ to cụ thể (specific loudness)**. Vậy sau khi đã chuyển đổi dạng của công thức (3.15) sang độ to cụ thể, biểu thức mới lúc này được đề xuất như sau

$$\frac{\Delta N'}{N' + N'_{gr}} = k \frac{\Delta E}{E + E_{gr}}, \quad (3.16)$$

với N'_{gr} và E_{gr} tương ứng với độ to và mức độ kích thích tại ngưỡng nghe của tai

người. Như vậy, lấy lại tích phân và chuyển về dạng hàm số mũ thu được công thức của độ to chi tiết này theo mức độ cảm ứng

$$(N' + N'_{gr}) = C(E + E_{gr})^k. \quad (3.17)$$

Tiếp tục sử dụng điều kiện tại ngưỡng nghe, khi đó cả N' và E đều là 0 nên có

$$N'_{gr} = CE_{gr}^k,$$

hay

$$C = \frac{N'_{gr}}{E_{gr}^k}. \quad (3.18)$$

Do vậy công thức của độ to chi tiết N' lúc này được tính bằng

$$N' + N'_{gr} = \frac{N'_{gr}}{E_{gr}^k}(E_{gr} + E)^k,$$

hay

$$\frac{N'}{N'_{gr}} = \left(\frac{E}{E_{gr}} + 1 \right)^k - 1.$$

Từ đó suy ra

$$N' = N'_{gr} \left(\left(\frac{E}{E_{gr}} + 1 \right)^k - 1 \right), \quad (3.19)$$

với các hệ số N'_{gr} , E_{gr} đều được đo tùy thuộc vào loại môi trường đang được xét, k được cố định ở 0.23 và được xác định bằng thực nghiệm.

Cuối cùng, độ to tổng thể của toàn bộ âm thanh sẽ là tích phân trên toàn bộ miền tần số Bark (chỉ xét tần số trong khoảng 20 - 20000 Hz, tương đương với 24 băng tần trong Bark) được định nghĩa bởi

$$N = \int_{f \in f_{\text{Bark}}} N' df. \quad (3.20)$$

3.2 Đánh giá chất lượng âm thanh

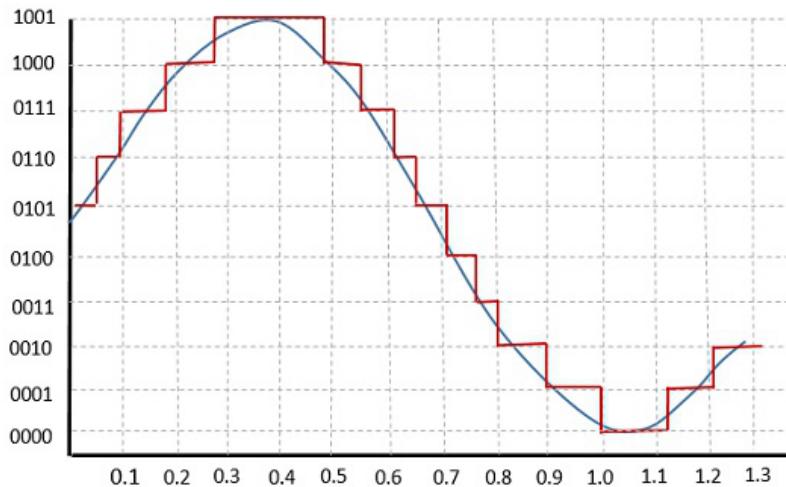
Ở phần này, chúng tôi sẽ đề cập tới một số khái niệm được sử dụng phổ biến trong đánh giá chất lượng âm thanh bao gồm: *thang đo đánh giá chất lượng MOS* [19], *metric đo khả năng thấu hiểu của một đoạn âm thanh STOI* [20], *mức cảm nhận âm thanh PESQ* [21] và *metric mô phỏng lại đánh giá của người sử dụng* [23] [24]. Trước tiên để hiểu thêm về mục đích của các độ đo nêu trên, chúng tôi sẽ đề cập tới những yếu tố có thể làm ảnh hưởng tới chất lượng âm thanh.

3.2.1 Các nhân tố ảnh hưởng tới chất lượng âm thanh

Trong thực tế, có rất nhiều yếu tố có thể ảnh hưởng tới chất lượng âm thanh nói chung, trong phần này để có thể đi sâu với đề tài mà chúng tôi thực hiện, chúng tôi sẽ đề cập tới các yếu tố ảnh hưởng tới chất lượng âm thanh được lưu trữ trong máy tính. Âm thanh được lưu trữ trong máy tính tất cả đều ở dạng các điểm mẫu rời rạc được lấy mẫu với tần số lấy mẫu f_0 từ âm thanh thực tế thông qua các phương tiện ghi như microphone, ..., sau đó lưu xuống đĩa cứng máy tính dưới dạng một mảng số thực. Từ file dữ liệu được ghi nhận trên đĩa cứng này, máy tính xấp xỉ lại hàm cần phát thông qua nội suy giữa hai điểm mẫu kề cận từ đó xấp xỉ lại âm thanh gốc ban đầu.

Trước khi tiến hành xử lý, âm thanh được ghi nhận ở dạng các số 32 bits (hoặc 8, 16, 64 bits), quá trình chuyển tín hiệu từ dạng liên tục trước khi được ghi nhận sang dạng rời rạc sau khi đã được lưu trữ trong máy tính được gọi là phép ***Quantization*** hay phép ***Định lượng***.

Tại mỗi thời điểm t , $f(t)$ lấy một trong các giá trị được chia sẵn thành các ngưỡng nhị phân như Hình 3.12 làm quy đổi cho $f(t)$ sang thành miền rời rạc, từ đó dữ liệu này được lưu xuống file dưới đĩa cứng và cũng là dữ liệu được sử dụng để xấp xỉ $f(t)$ ban đầu. Trong phần lớn các kiểu định lượng được sử dụng hiện nay, dữ liệu được sử dụng là một số thực 32 hoặc 64 bits (cũng có thể là 8 hoặc 16 bits) tùy vào độ phân giải được quy định, các tín hiệu âm thanh cũng được mã hóa dưới dạng này. Quá trình định lượng rõ ràng có tạo ra một sự suy giảm trong chất lượng và sự ảnh hưởng của nó cũng bị phụ thuộc vào tần số lấy mẫu f_0 . Vấn



Hình 3.12: Quá trình định lượng tín hiệu

đè chất lượng bị suy giảm trong quá trình rời rạc hóa như trên được tổng quát lên thành vấn đề rời rạc hóa dữ liệu, trong một số bài báo họ gọi đây là sự suy giảm chất lượng do codecs.

Đó là một trong các yếu tố xuất hiện bên trong hệ thống máy tính, ngoài ra còn có rất nhiều yếu tố khác nữa như thông tin bị mất mát trong quá trình truyền đi, lỗi xuất hiện khi thu âm từ microphone, ... nhưng chung quy lại chúng là những lỗi mất mát không thể tránh được. Ngoài những lỗi nêu trên, trên thực tế, các yếu tố ảnh hưởng tới chất lượng âm thanh nhiều nhất có lẽ là tiếng ồn. Để có thể đánh giá được sự ảnh hưởng của các yếu tố bên trong và các yếu tố bên ngoài, một đại lượng đã được đề xuất ra để lượng hóa cho sự ảnh hưởng của các yếu tố trên lên giọng nói. Đại lượng đó được gọi là **Mean Opinion Score** (tạm dịch là thang đo chất lượng).

3.2.2 Thang đo Mean Opinion Score

“Trung bình một người sẽ đánh giá đoạn âm thanh được cho như thế nào?”. Đây cũng chính là nguồn khởi xướng cho một thang đo tiêu chuẩn của chất lượng âm thanh. **Mean Opinion Score** [19] được chia làm 5 mức tiêu chuẩn được trình bày trong Bảng 3.1.

Mức	Chất lượng giọng nói	Mức độ nhiễu
5	Rất tốt	Không thể nhận biết
4	Tốt	Có thể nhận biết nhưng không gây khó chịu
3	Bình thường	Có thể nhận biết, hơi gây khó chịu
2	Tệ	Nhiều lần át giọng nói, gây khó chịu
1	Rất tệ	Nhiều mạnh không nghe được nội dung, gây khó chịu mạnh

Bảng 3.1: Bảng thang đo Mean Opinion Score (MOS) (nguồn [19])

Các phương pháp kiểm tra chất lượng âm thanh sử dụng thang đo MOS này được chia thành hai loại: *đánh giá chủ quan* (*subjective listening test*) và *đánh giá khách quan* (*objective quality measure*).

Đánh giá chủ quan. Trong cách đánh giá chủ quan, con người được xem là tiêu chuẩn để đánh giá âm thanh. Việc thực hiện đánh giá chủ quan thường phải tuân theo khá nhiều quy chuẩn để tránh đi các yếu tố như tình trạng sức khỏe của người tham gia đánh giá, chất lượng môi trường, thiết bị phần cứng, ... ảnh hưởng lên kết quả đánh giá. Các tiêu chuẩn và một số cách để đánh giá các tiêu chuẩn này được khuyến nghị trong ITU P.808 [25] và ITU P.835 [26]. Sử dụng các kết quả thu được sau các lần đánh giá, một số metrics dùng để xấp xỉ lại đánh giá của con người cũng được áp dụng [23, 24].

Đánh giá khách quan. Với cách đánh giá khách quan, con người không trực tiếp tham gia vào quá trình đánh giá chất lượng âm thanh, thay vào đó các giải thuật hoặc các phép kiểm thử của thống kê sẽ làm việc này. Việc đánh giá một cách chủ quan như vậy tránh đi sự bias trong quá trình đánh giá của con người làm cho kết quả này đáng tin cậy hơn đồng thời cũng đỡ tiêu tốn tài nguyên về thời gian cũng như tiền bạc phải bỏ ra trong quá trình kiểm tra.

Dưới đây, chúng tôi sẽ trình bày hai cách đánh giá theo hướng đánh giá khách quan được sử dụng trong luận văn này là Short-time Objective Intelligibility Measurement (STOI) [20] và Perceptual Evaluation of Speech Quality (PESQ) [21]. Để có một cái nhìn toàn diện hơn về kết quả của mô hình, chúng tôi cũng sử dụng một metric khác tiếp cận theo hướng đánh giá chủ quan và được xem như một trong các metrics chính trong cuộc thi DNS [14] của Microsoft là A Non-Intrusive Perceptual Objective Speech Quality Metric (DNSMOS) [23, 24].

3.2.3 Short-time Objective Intelligibility Measurement

Short-time Objective Intelligibility Measurement (tạm dịch là độ đo thấu hiểu, STOI) [20] là metrics được tính toán trên miền tần số thời gian của hai âm được cho trước, ta lần lượt gọi hai âm này là $x(t)$ và $\hat{x}(t)$ để đại diện âm sạch (referenced sound) và âm được xử lý (degraded sound). Trước khi tràn bày vào metric chính, ta sẽ tìm hiểu về một số khái niệm có liên quan được sử dụng trong metric này.

Dộ đo tương tự Pearson. Độ đo tương tự Pearson của hai vector $a = (a_1, \dots, a_n) \in \mathbb{R}^n$ và $b = (b_1, \dots, b_n) \in \mathbb{R}^n$ được định nghĩa như sau

$$r = \frac{\sum_{i=0}^n (a_i - \bar{a})(b_i - \bar{b})}{\sqrt{\sum_{i=0}^n (a_i - \bar{a})^2 \sum_{i=0}^n (b_i - \bar{b})^2}}, \quad (3.21)$$

với $\bar{a} = \sum_{i=1}^n a_i/n$ và $\bar{b} = \sum_{i=1}^n b_i/n$.

Xem hai vector trên là một chuỗi các giá trị của một biến ngẫu nhiên nào đó được ghi nhận lại, độ đo Pearson có thể được hiểu như sự biến thiên của hai biến ngẫu nhiên có phân phối có trung bình thống kê là 0 và độ lệch chuẩn σ bằng 1. Cụ thể

$$\begin{aligned} r &= \frac{\sum_{i=0}^n (a_i - \bar{a})(b_i - \bar{b})}{\sqrt{\sum_{i=0}^n (a_i - \bar{a})^2 \sum_{i=0}^n (b_i - \bar{b})^2}} \\ &= \frac{n}{n} \frac{\sum_{i=0}^n (a_i - \bar{a})(b_i - \bar{b})}{\sqrt{\sum_{i=0}^n (a_i - \bar{a})^2 \sum_{i=0}^n (b_i - \bar{b})^2}} \\ &= \frac{1}{n} \sum_{i=0}^n \left(\frac{a_i - \bar{a}}{\sigma_a} \right) \left(\frac{b_i - \bar{b}}{\sigma_b} \right) \\ &= \frac{1}{n} \sum_{i=0}^n \text{z-score}(a_i) \text{z-score}(b_i). \end{aligned} \quad (3.22)$$

Hàm z-score có thể được xem là một cách chuẩn hóa phân phối về trung bình thống kê 0 và độ lệch chuẩn là 1. Để xem xét tại sao hàm z-score lại làm được điều này, ta xét một ví dụ với $x \in \mathbb{R}^n$ như sau.

Lần lượt gọi \bar{x}' và $\sigma_{x'}$ là trung bình thống kê và độ lệch chuẩn mới của x sau khi qua z-score, lúc này từ định nghĩa ta đã có

$$x' = \text{z-score}(x) = \frac{x - \bar{x}}{\sigma_x}. \quad (3.23)$$

Từ công thức tính toán trung bình thống kê của x' , ta thu được

$$\begin{aligned} \bar{x}' &= \frac{1}{n} \sum_{i=0}^n x' = \frac{1}{\sigma_x} (\bar{x} - \bar{x}) = 0. \\ \sigma_{x'} &= \sqrt{\frac{\sum_{i=0}^n (x' - \bar{x}')^2}{n}} = \sqrt{\frac{\sum_{i=0}^n x'^2}{n}} \\ &= \sqrt{\frac{\sum_{i=0}^n \left(\frac{x - \bar{x}}{\sigma_x}\right)^2}{n}} = \sqrt{\frac{\sum_{i=0}^n (x - \bar{x})^2}{n} \frac{1}{\sigma_x^2}} \\ &= \sqrt{\frac{\sigma_x^2}{\sigma_x^2}} = 1. \end{aligned}$$

Như vậy, sau khi x được chuẩn hóa bằng z-score, trung bình thống kê và độ lệch chuẩn của x' sẽ lần lượt là 0 và 1. Điều này đưa lại lợi ích rất lớn cho việc so sánh hai biến ngẫu nhiên, vì công thức này sẽ đo giá trị tích vô hướng giữa hai biến đã được chuẩn hóa này.

Short-time Objective Intelligibility Measurement. Trong STOI, sau khi dữ liệu âm thanh được chuyển sang miền tần số thời gian bằng biến đổi Fourier thời gian ngắn, các vùng biên độ trong cùng một khung sẽ được tổng hợp lại theo công thức dưới đây

$$X_j(m) = \sqrt{\sum_{k=f_{low}(j)}^{f_{high}(j)} |F_k(m)|^2}, \quad (3.24)$$

với $f_{low}(j)$ và $f_{high}(j)$ lần lượt là ngưỡng dưới và ngưỡng trên của dãy khoảng tám thứ j , $F_k(m)$ là giá trị thứ k trong spectrum của $f(t)$ ở khung thời gian thứ m . Bằng cách chuyển từ spectrogram của tần số thông thường sang spectrogram của tần số cảm nhận, tác giả giả lập lại cơ chế nghe của tai người từ đó tính ra sự tương quan giữa hai giá trị. Bước tiếp theo sẽ đi tính độ đo tương tự Pearson lên hai miền giá trị tần số thời gian vừa thu được

$$d_j = \frac{\sum_{i=0}^n (X_j(i) - \bar{X}_j)(Y_j(i) - \bar{Y}_j)}{\sqrt{\sum_{i=0}^n (X_j(i) - \bar{X}_j)^2 \sum_{i=0}^n (Y_j(i) - \bar{Y}_j)^2}}. \quad (3.25)$$

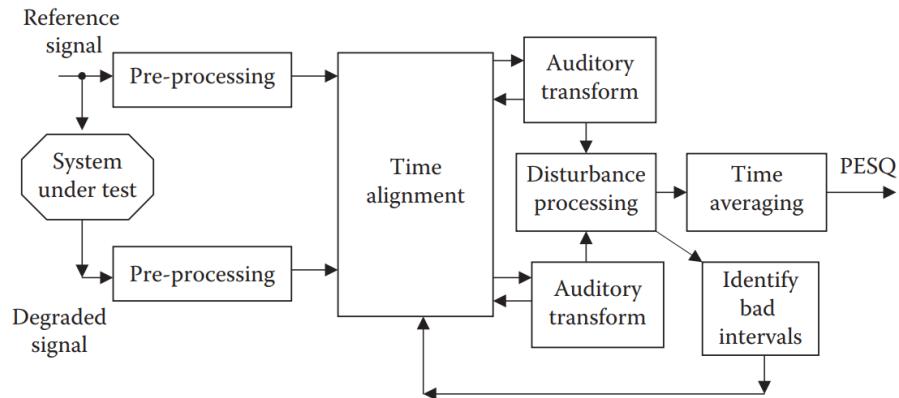
Từ công thức (3.25) thu được một giá trị nằm trong khoảng $[-1, 1]$ đại diện cho sự tương đồng trong khả năng thấu hiểu tần số của tai người tại dãy khoảng tám thứ j , trung bình tất cả kết quả theo thời gian, kết quả cuối cùng chính là độ đo của STOI - đại diện cho toàn bộ khả năng thấu hiểu của tai người trên giữa hai tín hiệu, và đó cũng là lý do đây là độ đo mà chúng tôi sẽ sử dụng để đo khả năng thấu hiểu của tín hiệu được đầu ra bởi mô hình.

3.2.4 Perceptual Evaluation of Speech Quality

Để có thể đánh giá được đúng chất lượng của âm thanh được trả về bởi mô hình, ngoài STOI để đánh giá khả năng nghe được, chúng tôi sử dụng một metric khác để đánh giá chất lượng của âm thanh. Metric này được gọi là **Perceptual Evaluation of Speech Quality** (tạm dịch là **độ đo chất lượng**, PESQ) [21]. Được ra đời dưới như cầu về đánh giá chất lượng âm thanh được truyền đi trong các cuộc gọi hội thoại Voice-over-IP (VoIP), PESQ cho thấy sự tương quan trong đánh giá với chất lượng được người dùng ghi nhận tới 0.92 [21]. Và cũng nhờ vào sự tương quan trong đánh giá của metric và chất lượng thực tế cao như vậy mà PESQ dần trở nên phổ biến trong đánh giá chất lượng giọng nói ở cả nghiên cứu và trong công nghiệp trong một thời gian dài cho tới khi các phương pháp đánh giá chủ quan được ứng dụng.

PESQ bắt đầu được sử dụng như một chuẩn trong ITU-T P.862 được dùng để đánh giá chất lượng âm thanh được truyền đi trong mạng. Trong môi trường mạng, giọng nói người sử dụng nghe được thường bị ảnh hưởng bởi nhiều yếu tố, như bộ codecs của giọng nói, các nhiễu điện tử xuất hiện trong quá trình truyền tải, nhưng hầu hết các yếu tố này chỉ làm xuất hiện tiếng rè và hầu hết sẽ không có sự biến đổi liên tục về tần số cấu tạo theo thời gian. Nhận thấy điều này, trong PESQ, mô hình tính toán độ to chịu giả định rằng các nhiễu này tương tự nhiễu trắng (còn được gọi là white noise, Zwicker gọi các nhiễu này là nhiễu kích thích đều - uniform excitation noise) từ đó thu được công thức chuyển đổi từ Bark spectrogram sang spectrogram độ to. Chúng tôi sẽ giải thích kĩ hơn giả định này trong từng bước xử

lý của PESQ.



Hình 3.13: Mô hình khối của PESQ bao gồm có bốn giai đoạn: *tiền xử lý* (qua một bộ lọc tiêu chuẩn để quy chuẩn âm), *căn chỉnh âm thanh* (do sự truyền đi trong môi trường mạng, các gói tin sẽ mất một thời gian mới đến đích, việc căn chỉnh là để tránh sự chênh lệch này làm ảnh hưởng tới việc đánh giá chất lượng), *biến đổi âm* (Bark Spectrogram được tính toán và chuẩn hóa ở bước này) và *tính toán độ gián đoạn* (nguồn [21])

Hình 3.13 mô tả lại mô hình khối của PESQ. Bắt đầu từ cùng một nguồn phát, âm đã qua xử lý bởi hệ thống $\hat{x}(t)$ và âm sạch $x(t)$ lần lượt đi qua các bước tiền xử lý, căn chỉnh âm thanh, biến đổi âm và tính toán gián đoạn, cuối cùng các thông số về sự gián đoạn lấy trung bình theo thời gian giữa hai âm sẽ được qua một hàm tuyến tính để trả về kết quả dự đoán của PESQ. Chúng tôi liệt kê các bước này như sau:

- Tiền xử lý:** Trong bước này, âm thanh sẽ được đi qua một bộ lọc tương tự như bộ lọc thấp (bộ lọc chỉ cho các tần số thấp hơn đã định đi qua) được sử dụng trong truyền tín hiệu điện thoại, sau đó các âm này được chuẩn hóa lại và đẩy tới bước tiếp theo.
- Căn chỉnh âm thanh:** Âm thanh sau khi đã được chuẩn hóa bởi bước *Tiền xử lý* sẽ được đưa sang để tiến hành tính toán độ trễ. Độ trễ giữa hai âm có thể được ước lượng bằng cách chia các đoạn âm thanh trên thành những khung thời gian ngắn hơn và tính toán sự tương quan theo thời gian giữa các khung này trong hai âm với nhau các đoạn, độ trễ sau đó được chuyển tiếp để tính toán phân chia hai âm trên thành các khung giọng nói tương ứng giữa hai âm (chiều dài các khung này thường vào khoảng 32ms).
- Biến đổi âm thanh:** Âm thanh đã được cắt ra thành các khung sẽ được biến đổi

Fourier thành spectrogram ở tần số Hz với chiều dài khung thời gian là 32ms và 50% overlap giữa các khung, sau đó được chuyển về tần số cảm nhận Bark thành Bark spectrogram (ứng với $x(t)$ và $\hat{x}(t)$ là $B_x(b)$ và $B_{\hat{x}}(b)$, b đại diện cho băng tần Bark thứ b). Sau đó, spectrogram độ to được tính toán dựa trên các băng tần Bark theo công thức của Zwicker [22]

$$S_k(b) = C \left(\frac{P_0(b)}{2} \right)^{0.23} \left(\left(0.5 + 0.5 \frac{B_k(b)}{P_0(b)} \right)^{0.23} - 1 \right), \quad (3.26)$$

với $k \in x, \hat{x}$. Các Spectrogram độ to này cùng với Bark Spectrogram sẽ được chuyển đến bước tiếp theo. Công thức chuyển đổi độ to trên có nguồn gốc từ một công thức được thực nghiệm của Zwicker [36]. Công thức gốc này được định nghĩa như sau

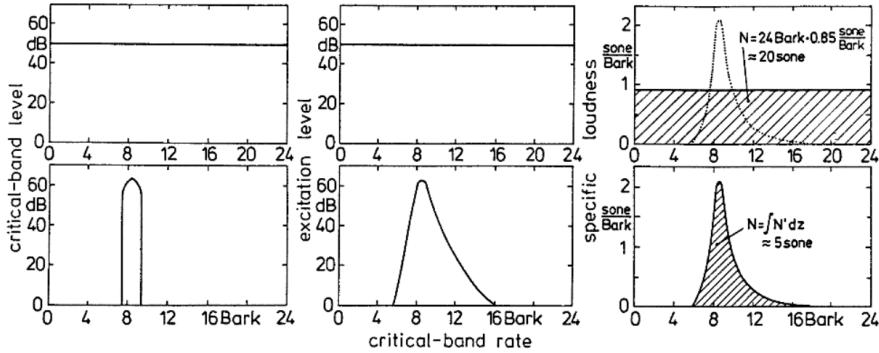
$$N_k = C \left(\frac{E_{TQ}}{E_0} \right)^{0.23} \left(\left(0.5 + 0.5 \frac{E_k}{E_0} \right)^{0.23} - 1 \right), \quad (3.27)$$

với N_k được gọi là độ to cụ thể (được chúng tôi trình bày ở Tiêu mục 3.1.5) và E_k là mức kích thích của âm tại khung thứ k , E_{TQ} là mức kích thích của âm vừa đúng tại ngưỡng nghe của tai người, E_0 là mức kích thích của âm với cường độ âm $I_0 = 10^{-12} W/m^2$. Bằng thực nghiệm, Zwicker cũng phát hiện được “Với nhiều kích thích đều, chỉ có những sự kích thích chính được xảy ra do đó sẽ thu được đồ thị kích ứng giống như bên trái Hình 3.14” [36]. Thông qua thử nghiệm này, rõ ràng tỉ lệ giữa E_k/E_0 sẽ tương ứng với $B_k(b)/P_0(b)$ với $P_0(b)$ là giá trị của tần số Bark thứ b tính tại cường độ âm I_0 . Nhờ đó mà PESQ có thể dựa vào giả định này chuyển hóa mô hình độ to được đề xuất vào các tính toán tiếp theo.

4. *Tính toán gián đoạn:* Tại bước này, các gián đoạn sẽ được xử lý ở đây trước khi được tổng hợp lại và tính ra kết quả dự đoán. Độ gián đoạn $D(b)$ được chia làm hai loại: *gián đoạn dương* (tương ứng $S_{\hat{x}}(b) > S_x(b)$) và *gián đoạn âm* (tương ứng $S_{\hat{x}}(b) < S_x(b)$). Để có sự linh hoạt hơn trong tính toán kết quả cuối, tác giả chọn một ngưỡng gián đoạn nhất định để xác định sự gián đoạn có đang xảy ra hay không, trong tài liệu [22], họ sử dụng

$$T(b) = 0.25 \times \min(S_x(b), S_{\hat{x}}(b)). \quad (3.28)$$

Vậy $D(b)$ được định nghĩa như sau



Hình 3.14: Liên hệ giữa độ kích ứng và Bark, bên trái thể hiện Bark Spectrum, ở giữa thể hiện cho độ kích ứng được thử nghiệm ra, cuối cùng là độ to được tính toán dựa trên tích phân của độ kích ứng theo từng Bark (nguồn [36])

$$D(b) = \begin{cases} (S_x(b) - S_{\hat{x}}(b)) - T(b), & \text{nếu } S_x(b) - S_{\hat{x}}(b) > T(b), \\ 0, & \text{nếu } |S_x(b) - S_{\hat{x}}(b)| \leq T(b), \\ (S_x(b) - S_{\hat{x}}(b)) + T(b), & \text{nếu } S_x(b) - S_{\hat{x}}(b) < -T(b). \end{cases}$$

Trong tài liệu tham khảo [22], tác giả chia gián đoạn ra làm hai loại: *đối xứng* $D_{sym}(b)$ và *bất đối xứng* $D_{asym}(b)$. Hai loại này lần lượt được tính như sau

$$D_{sym}(b) = \left(\sum_{b=1}^{N_b} W_b \right)^{1/2} \left(\left(\sum_{b=1}^{N_b} (|D(b)|W_b)^2 \right) \right)^{1/2},$$

$$D_{asym}(b) = C(b)D(b),$$

với W_b là trọng số của băng tần Bark thứ b , $C(b)$ là hệ số bất đối xứng được tính toán như sau

$$C(b) = \begin{cases} 0, & \left(\frac{B_{\hat{x}}(b)+50}{B_x(b)+50} \right)^{1.2} < 3, \\ 12, & \left(\frac{B_{\hat{x}}(b)+50}{B_x(b)+50} \right)^{1.2} > 12, \\ \left(\frac{B_{\hat{x}}(b)+50}{B_x(b)+50} \right)^{1.2}, & \text{các trường hợp khác.} \end{cases}$$

Cả hai loại gián đoạn này sau đó sẽ được tổng hợp theo 20 khung thời gian (khoảng 320ms do có 50% overlap giữa các khung), việc tổng hợp này sẽ trả về kết quả $D_{sym}(k)$ và $D_{asym}(k)$ với k tương ứng với lần tổng hợp 20 khung thứ k .

Kết quả cuối cùng của hai độ gián đoạn d_{sym} và d_{asym} được tính toán như trung bình theo chuẩn 2 của $D_{sym}(k)$ và $D_{asym}(k)$ trên tất cả các khung thời gian k

$$d_{sym} = \left(\frac{\sum_k (D_{sym}(k)t_k)^2}{\sum_k t_k^2} \right),$$

$$d_{asym} = \left(\frac{\sum_k (D_{asym}(k)t_k)^2}{\sum_k t_k^2} \right).$$

Cuối cùng, kết quả của PESQ sẽ được tổng hợp như một hàm tuyến tính của hai độ gián đoạn nêu trên

$$\text{PESQ} = 4.5 - 0.1d_{sym} - 0.0309d_{asym}. \quad (3.29)$$

PESQ được tính theo công thức (3.29) có giá trị trong khoảng $[-0.5, 4.5]$ nhưng thường, kết quả của phép tính toán này thường rơi vào khoảng $[1, 4.5]$ tương ứng với thang đo MOS. Tuy nhiên, trong [22], tác giả cũng có đề cập tới một số phương pháp để chuyển đổi giữa giá trị của PESQ trong công thức (3.29) (còn gọi là giá trị PESQ nguyên bản) để cho kết quả dự đoán chính xác hơn trên thang đo MOS. Hai phép chuyển đổi này được định nghĩa như công thức (3.30) và (3.31), hai công thức này còn có tên gọi tương ứng là **Wideband PESQ (WBPESQ)** [34] và **Narrowband PESQ (NBPESQ)** [33].

$$\text{WBPESQ} = 0.999 + \frac{4}{1 + e^{-1.3669 \text{ PESQ} + 3.8224}}, \quad (3.30)$$

$$\text{NBPESQ} = 0.999 + \frac{4}{1 + e^{-1.4945 \text{ PESQ} + 4.6607}}. \quad (3.31)$$

Dễ thấy, bị giới hạn bởi bản thân giả định được đặt ra khi mô hình hóa lại độ to của âm đầu vào, PESQ khá kém nhạy cảm với các loại nhiễu không tương tự nhiễu trắng. Với các loại nhiễu này, chỉ số được PESQ dự đoán ra khá kém hiệu quả và đôi khi không phản ánh đúng chất lượng thực sự của âm thanh sau khi lọc. Đây cũng chính là một trong các khó khăn của sinh viên thực hiện khi thực hiện đề tài này.

3.2.5 A Non-Intrusive Perceptual Objective Speech Quality Metric

Tuy STOI và PESQ đã cho ta thấy một cái nhìn khá hoàn chỉnh về giọng nói sau khi lọc bởi mô hình của mình, tuy nhiên cái nhìn này vẫn chưa thực sự phản ánh đúng với chất lượng thực sự của mô hình. Tuy STOI cho ta thấy được cảm nhận

của tai khi nghe nhưng cũng chính vì cơ chế sử dụng độ đo tương tự Pearson, các thành phần có biên độ nhỏ hầu hết sẽ không đóng góp gì vào giá trị của STOI do đó các nhiễu nhỏ thông thường sẽ bị bỏ qua, PESQ lại bị hạn chế khả năng dự đoán bởi giả định nhiễu có trong giọng nói nhiễu là kích thích đều. Các hạn chế này khiến cho việc đánh giá mô hình của chúng tôi gặp nhiều khó khăn và cũng chính từ các hạn chế này, chúng tôi tìm hiểu một metric được áp dụng để mô phỏng lại sự đánh giá của con người với cách tiếp cận chủ quan. Metric mà chúng tôi muốn nói tới là **DNSMOS** [23, 24].

DNSMOS được chia làm hai loại phụ thuộc vào mục tiêu đánh giá của ta như thế nào, đối với bản DNSMOS được đề xuất trong [23], mục tiêu đánh giá là tổng thể toàn bộ đoạn âm thanh theo chuẩn ITU-T P.808 [25], nhưng đối với bản DNSMOS được đề xuất trong [24], mục tiêu trả về đa dạng hơn bao gồm SIG (chất lượng giọng nói), BAK (chất lượng lọc nhiễu) và OVR (chất lượng tổng quan của đoạn âm thanh). Mục tiêu trả về này được quy định theo chuẩn của ITU-T P.835 [26] vậy nên bản DNSMOS này còn được biết tới với cái tên DNSMOS P.835. Đây cũng là bản DNSMOS mà sinh viên thực hiện dùng để đánh giá mô hình của mình.

The screenshot shows a user interface for a DNSMOS survey. It consists of three main sections:

- Section 1:** A question asking "How do you rate the overall quality of the following speech sample?". Below it is a play button and a progress bar showing 00:00 / 00:09. To its right is a table mapping speech quality to scores:

Quality of the speech	Score
Excellent	5
Good	4
Fair	3
Poor	2
Bad	1
- Section 2:** Another question asking "How would you rate the overall quality of the following speech sample?". Below it is a play button and a progress bar showing 00:00 / 00:09. To its right is a table mapping speech quality to scores, identical to Section 1:

Quality of the speech	Score
Excellent	5
Good	4
Fair	3
Poor	2
Bad	1
- Section 3:** A question asking "Listen to the following speech clip." followed by a play button and a progress bar showing 00:00 / 00:08. Below it is a table for describing the speech signal:

the SPEECH SIGNAL in this sample was	Score
Not distorted	5
Slightly distorted	4
Somewhat distorted	3
Fairly distorted	2
Very distorted	1

At the bottom left, there is a "Thanks for your participation." message and a note stating "Your qualification will be assigned in ... minutes/days. Then you are allowed to perform ... jobs."

Hình 3.15: Khảo sát được tham khảo từ ITU-T P.808 (nguồn [25])

DNSMOS P.835 tiếp cận việc đánh giá âm thanh theo góc nhìn học sâu, họ sử dụng một mạng tích chập để học ra các tham số dự đoán cho ba đầu ra SIG, BAK và OVR (đối với chuẩn mô hình theo chuẩn ITU-T P.808 thì chỉ có OVR được dự

đoán). Để có thể huấn luyện được mô hình này họ tiến hành khảo sát và tiến hành thu thập dữ liệu. Khảo sát này được ITU đề xuất trong [25] bao gồm hình thức khảo sát, cách thức tiến hành khảo sát, chọn đối tượng, các biểu mẫu khảo sát và cả những phần thường khuyến khích nên được sử dụng.

Sử dụng các dữ liệu thu được từ khảo sát, mô hình được đề xuất trong [23] và [24] được dùng để dự đoán đánh giá của người nghe. Đối với chuẩn ITU-T P.808, mô hình dự đoán một chỉ số đầu ra đại diện cho chất lượng tổng thể của đoạn âm thanh đầu vào, và đối với chuẩn ITU-T P.835, dự đoán được chia ra làm ba loại tương ứng với ba loại được khuyến nghị bởi ITU. DNSMOS được sử dụng chủ yếu trong đánh giá mô hình cho cuộc thi Deep Noise Suppression (DNS) [14] của Microsoft để xấp xỉ lại đánh giá của người sử dụng nghe đoạn âm thanh được xử lý qua mô hình.

Chương 4

Học sâu

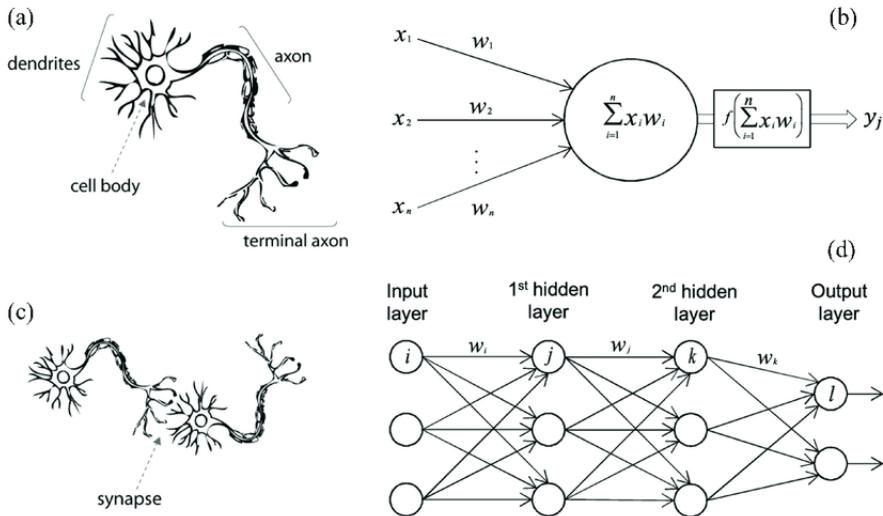
Trong phần này, chúng tôi sẽ đề cập tới một số mạng học sâu mà chúng tôi sẽ sử dụng trong các thí nghiệm. Dữ liệu phụ thuộc thời gian, điển hình là giọng nói, là một dạng dữ liệu mà thông tin hàm chứa bên trong nó thay đổi theo thời gian. Có thể liên quan về các thông tin như biên độ, tần số, đôi khi cũng có thể là các yếu tố như ngữ nghĩa, các thông tin hiện tại thường sẽ có sự phụ thuộc nào đó vào các dữ liệu trước đó. Lấy một ví dụ trong giọng nói, rõ ràng tại một thời điểm, con người không thể nào truyền tải hết toàn bộ thông tin mình muốn nói sang cho người nghe được, và ở phía của người nghe, các sự nhận, thay đổi âm giọng trong quá trình truyền đạt của người nói, người nghe phải nhớ được một số thông tin từ những sự việc đó kết hợp nó với thông tin mới vừa nhận được ở hiện tại và tiếp đó cho tới khi người nói hoàn tất quá trình truyền đạt thì người nghe mới có được một thông tin trọn vẹn. Và đó chính là lý do **mạng hồi quy (recurrent neural network)** sẽ được sử dụng như xương sống cho các mô hình được dùng trong luận văn này.

4.1 Mạng hồi quy

Trong học máy cổ điển, một mạng neural bình thường sẽ bao gồm hai bộ phận chính là *neural* và *sự liên kết giữa các neural*. Bằng cách thay đổi các sự liên kết này, mạng neural đang cố gắng lưu trữ thông tin đang được học lại bên trong mạng bằng những trọng số của nó. Một neural nhân tạo có cấu tạo như Hình 4.1. Gọi

W_t là trọng số của neural này tại thời điểm t , b_t là bias và $f(x)$ là hàm kích hoạt. Kết quả đầu ra của một neural được tính toán bằng công thức

$$h_t = f\left(W_t x_t + b_t\right), \quad (4.1)$$

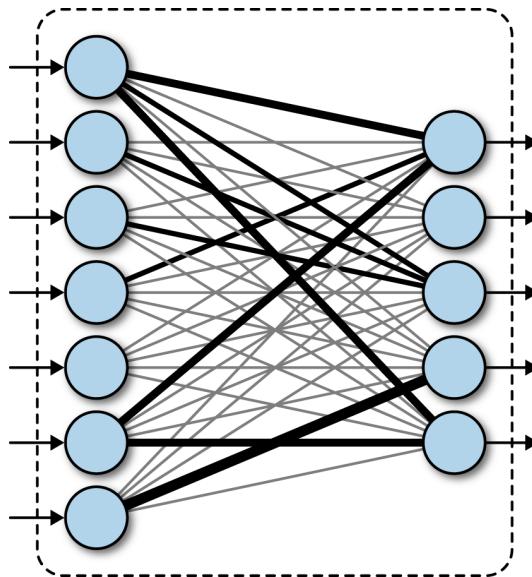


Hình 4.1: Mô hình của một neural (toán học) mô phỏng lại cấu tạo của neural (sinh học) bên trong não bộ con người

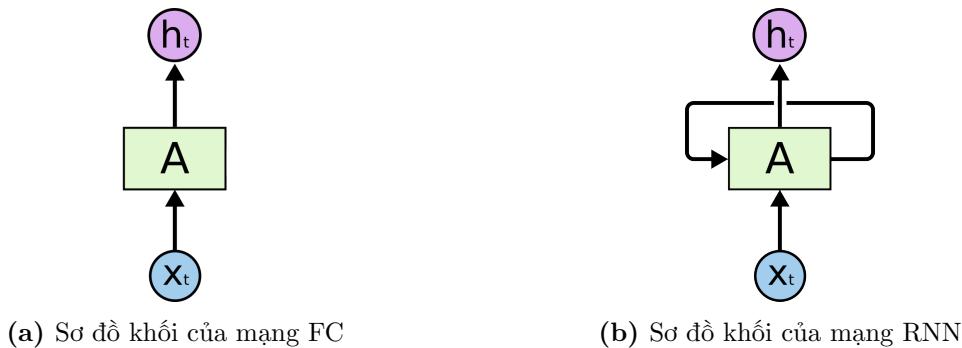
với h_t là giá trị đầu ra của neural ứng với đầu vào x_t thời điểm t . Khi mà các sự liên kết trong neural tiến theo một chiều và tất cả những neural ở phía trước đều liên kết với neural ở sau, nó được gọi là một lớp fully connected (FC).

Nhưng nếu một mạng chỉ toàn bao gồm các lớp FC như thế này, thì việc học ra được sự phụ thuộc về thời gian của dữ liệu gần như là không thể, bởi vì dữ liệu trong một mạng FC (mạng chỉ chứa mỗi FC) dữ liệu chỉ có một chiều đi duy nhất là từ đầu vào ra thẳng đầu ra và nó hoàn toàn không xét tới sự phụ thuộc về mặt thời gian của dữ liệu. Để giải quyết vấn đề này, ta xét một khái niệm mới là **mạng hồi quy (recurrent neural network - RNN)**. Mạng hồi quy là một neural network nhưng ở đây, sự liên kết giữa các neural có một sự khác biệt. Sự khác biệt này được thể hiện thông qua Hình 4.3.

Ở mạng RNN, sự phụ thuộc về mặt thời gian vào đầu ra đối với dữ liệu trước làm cho mô hình có chứa loại mạng này có khả năng học ra được những sự phụ thuộc về thời gian của dữ liệu. Sự lặp lại theo thời gian này có thể được thể hiện



Hình 4.2: Lớp FC với 7 đầu vào và 5 đầu ra



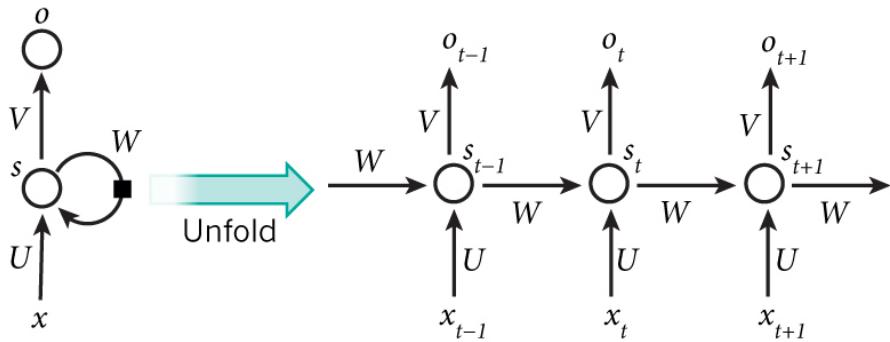
Hình 4.3: Sơ đồ khối của hai mạng neural

rõ hơn bằng cách duỗi network này ra như Hình 4.4.

Hình 4.4 thể hiện luồng chạy của mô hình RNN được mở rộng theo thời gian với x_t là đầu vào tại thời điểm t của mạng, o_t là đầu ra tại thời điểm t của mạng, s_t chính là các thông tin được tổng hợp từ đầu tới thời điểm t hiện tại, các thông tin này sẽ tiếp tục được truyền cho các bước kế tiếp dùng để dự đoán ra đầu ra o của mô hình. Về mặt toán học, kết quả xuất ra tại một neural trong một RNN được tính như sau cho mọi $t > 0$, Hình 4.5 là biểu đồ khái niệm luồng dữ liệu thực thi trên một neural của RNN.

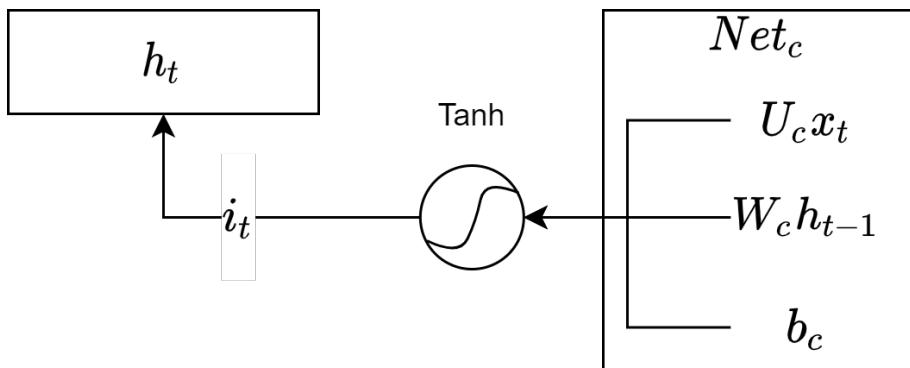
$$h_t = f(U_t x_t + W_t h_{t-1} + b_t). \quad (4.2)$$

Trong trường hợp $t = 0$, h_0 lúc này mang giá trị là 0. Ở công thức (4.2), U_t và



Hình 4.4: Quá trình duỗi mạng RNN theo thời gian

W_t là những trọng số ứng với đầu vào x_t và trạng thái cũ h_{t-1} , f là hàm kích hoạt của neural RNN này (thông thường các neural trong RNN sử dụng hàm tanh như hàm kích hoạt), một số loại hàm kích hoạt thường dùng được liệt kê ở Bảng 4.1.



Hình 4.5: Cấu tạo của một neural trong RNN, để đảm bảo tính tương đồng với LSTM, chúng tôi sử dụng gạch chân c cho các trọng số trong RNN

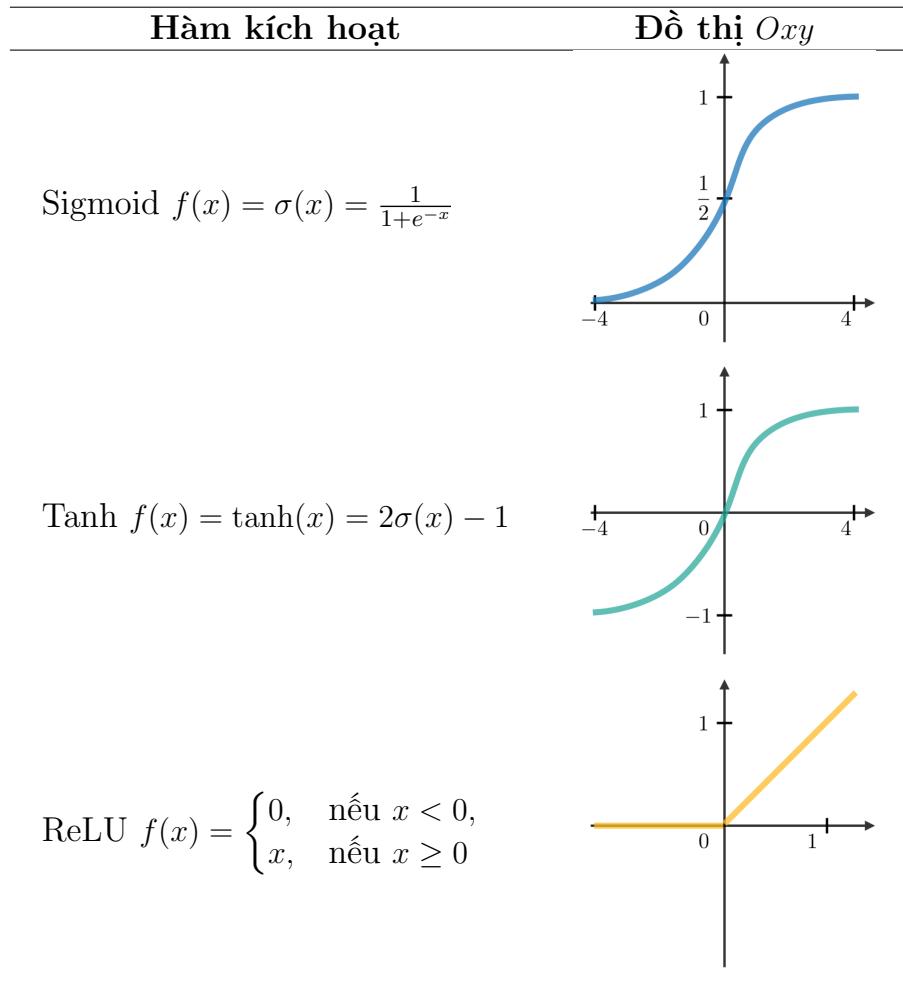
Để có thể huấn luyện mô hình RNN, ta thực hiện phép lấy đạo hàm như sau

$$\frac{d}{dh_{t-1}} h_t = (1 - h_t^2) W_t. \quad (4.3)$$

Tiếp tục lấy đạo hàm hàm h_t theo h_{t-q}

$$\frac{d}{dh_{t-q}} h_t = (1 - h_{t-q}^2) W_{t-q} \prod_{i=t-q+1}^t (1 - h_i^2) W_i, \quad (4.4)$$

vì $(1 - h_t^2)$ luôn nằm trong khoảng $[0, 1]$, do vậy giá trị của đạo hàm trên sẽ càng ngày càng nhỏ, và khi lan truyền đạo hàm này xuống các trọng số, giá trị đạo hàm



Bảng 4.1: Các hàm măt măt thường được sử dụng trong mô hình hồi quy

ở các thời gian với khoảng cách q lớn sẽ gần như về 0 và không còn đóng góp gì vào gradient chung của trọng số nữa. Đây là tình trạng **vanishing gradient** của RNN.

4.2 Long short term memory

Để khắc phục tình trạng “vanishing gradient” được nêu ở trên, **Long-Short Term Memory (tạm dịch là mạng nhớ, LSTM)** [46] sử dụng một cơ chế truyền lỗi hằng bên trong phần nhớ (memory cell) của mô hình. Cơ chế này được gọi là **Constant Error Carousel (CEC)**. Cũng xuất phát từ việc lấy đạo hàm với từ đầu ra của mô hình tại t và $t - q$, có

$$\frac{d}{dh_{t-q}} h_t = \frac{d}{dc_t} h_t \left(\prod_{i=t-q+2}^t \frac{d}{dc_{i-1}} c_i \right) \frac{d}{dh_{t-q}} c_{t-q+1}, \quad (4.5)$$

để có thể khắc phục được tình trạng “vanishing gradient”, chúng tôi cần

$$\frac{d}{dc_{i-1}} c_i = 1.0. \quad (4.6)$$

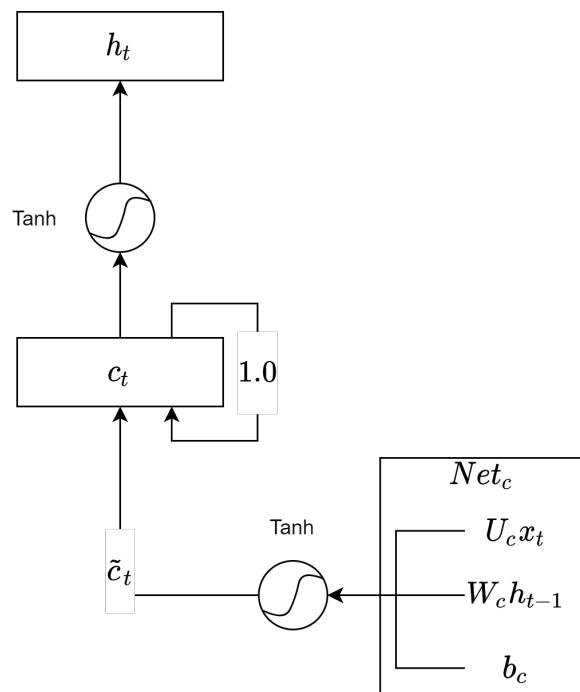
Điều này dẫn tới kết quả

$$c_i = c_{i-1} + C, \quad (4.7)$$

và trong [46], họ chọn C tương ứng với kết quả đầu vào của mô hình.

$$C = \tilde{c}_t = \tanh(U_c x_t + W_c h_{t-1} + b_c). \quad (4.8)$$

Như vậy, mô hình RNN được cải tiến để có thể chống lại phần nào tình trạng “vanishing gradient”. Giá trị c_i lúc này được gọi là phần nhớ của mô hình RNN cải tiến này (tiền đề của LSTM). Hình 4.6 cho thấy cấu trúc mô hình RNN với phần nhớ.



Hình 4.6: Cấu tạo của RNN có phần nhớ

Để tiếp tục xây dựng mô hình LSTM đồi đầu, các tác giả của [46] nhận thấy với các dạng dữ liệu có chứa nhiễu, đặc biệt là các nhiễu xuất hiện nhiễu giữa các điểm dữ liệu, RNN sẽ bị tình trạng **xung đột trọng số** (**weight conflict**). Tình trạng này xảy ra trong quá trình đạo hàm theo thời gian, nhiễu với giá trị mất mát lớn và gần với W, U hơn là các điểm dữ liệu sạch, các gradient sai này sẽ loại bỏ tác dụng của các gradient từ các thời điểm xa hơn và làm cho mô hình trở nên khó học hơn (“*This conflict makes learning difficult*” [46]).

Để khắc phục tình trạng này, các bộ phân loại tại đầu vào và đầu ra của mô hình RNN với phần nhớ được đề xuất. Các bộ phân loại này được gọi là các phần cổng (gate units). Mô hình RNN kết hợp với phần nhớ và các phần cổng này tạo thành mô hình LSTM khởi đầu được đề xuất trong [46]. Mô hình này được thể hiện trong Hình 4.7a.

LSTM khởi đầu này được viết dưới dạng các công thức như sau

$$\begin{aligned} i_t &= \sigma(U_i x_t + W_i h_{t-1} + b_i), \\ o_t &= \sigma(U_o x_t + W_o h_{t-1} + b_o), \\ \tilde{c}_t &= \tanh(U_c x_t + W_c h_{t-1} + b_c), \\ c_t &= c_{t-1} + i_t \tilde{c}_t, \\ h_t &= o_t \tanh(c_t). \end{aligned}$$

Các thông tin từ đầu vào \tilde{c}_t sau khi được chọn bởi cổng i_t để loại bỏ ra các thành phần nhiễu sẽ được tổng hợp tuyến tính với c_{t-1} để tạo thành thông tin mới c_t và sau đó tính toán ra được đầu ra h_t . Nhưng một vấn đề dễ thấy ở đây, tỉ lệ tăng của c_t theo c_{t-1} là 1, do để khắc phục tình trạng “vanishing gradient” như đã nêu ở trên, nhưng điều này khiến cho c_t tăng lên không giới hạn với những chuỗi đầu vào tương đối dài (t lớn).

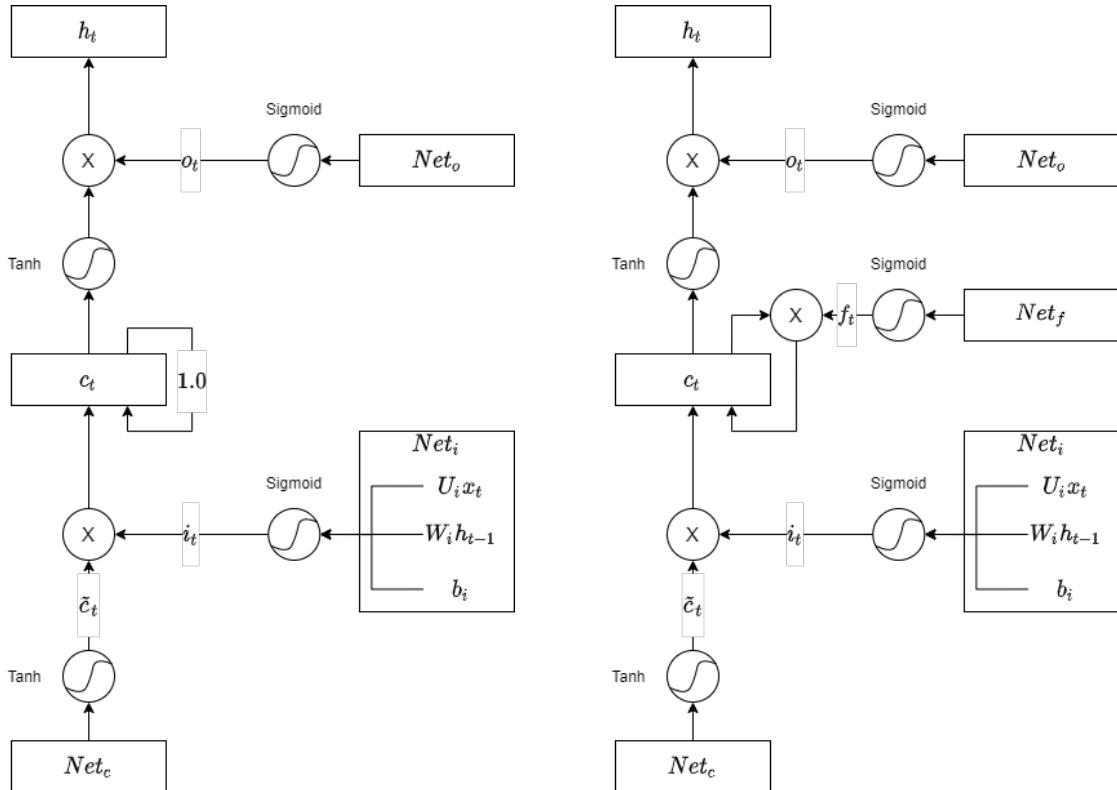
Được nghiên cứu trong [47], LSTM tiêu chuẩn có c_t tăng tuyến tính theo thời gian, do đó khi t đạt tới một mức mà c_t đủ lớn, $\tanh(c_t)$ lúc này xấp xỉ 1 và khiến cho đạo hàm từ h_t về c_t (có giá trị $1 - \tanh^2(c_t)$) chỉ còn xấp xỉ 0 do đó không cho phép gradient từ những thời điểm t này quay về cập nhật lại cho trọng số nữa. Để khắc phục tình trạng này, [47] đề xuất một cổng quên được đặt ở đạo hàm $\frac{d}{dc_{t-1}} c_t$. Vậy công thức đạo hàm của phần nhớ (memory cell) lúc này là

$$\frac{d}{dc_{i-1}} c_i = f_i, \quad (4.9)$$

với các thành phần lúc này được định nghĩa như sau

$$\begin{aligned} f_t &= \sigma(U_f x_t + W_f h_{t-1} + b_f), \\ i_t &= \sigma(U_i x_t + W_i h_{t-1} + b_i), \\ o_t &= \sigma(U_o x_t + W_o h_{t-1} + b_o), \\ \tilde{c}_t &= \tanh(U_c x_t + W_c h_{t-1} + b_c), \\ c_t &= f_t c_{t-1} + i_t \tilde{c}_t, \\ h_t &= o_t \tanh(c_t). \end{aligned}$$

Đây cũng chính là công thức của LSTM thường được sử dụng hiện nay trong công nghiệp và sẽ chính là mô hình chính sẽ được sinh viên thực hiện sử dụng trong luận văn này. Hình 4.7b thể hiện sơ đồ của mô hình LSTM mở rộng.



Hình 4.7: Mô hình LSTM tiêu chuẩn và mở rộng

Chương 5

Các công trình nghiên cứu liên quan

Trong chương này, chúng tôi sẽ nêu một số hướng tiếp cận mà chúng tôi khảo sát được, giải thích lý do tại sao chúng tôi lại chọn hướng tiếp cận học sâu thay vì các hướng tiếp cận cổ điển. Cuối cùng dựa vào những hướng tiếp cận trên, chúng tôi đề xuất phương pháp của mình về việc cải tiến hàm mất mát và đề xuất mô hình để phù hợp hơn với những yêu cầu từ phía ứng dụng cho người dùng.

5.1 Cách tiếp cận cổ điển

Nhắc lại một chút về bài toán đang xét của chúng ta, bài toán thực tế mà ta cần phải giải là với dữ liệu đầu vào $y(t)$, mô hình sử dụng là f , $x(t)$ và $n(t)$ lần lượt là giọng nói sạch và nhiều lần trong giọng nói đầu vào mô hình

$$\hat{x}(t) = f(y(t)) \approx x(t). \quad (5.1)$$

Giá trị đầu ra của mô hình ta mong muốn sẽ có thể xấp xỉ giọng nói sạch $x(t)$, và vẫn có thể bảo toàn hết khả năng nội dung được truyền tải. Với cách tiếp cận cổ điển, mô hình là một bộ lọc $g(t)$, bộ lọc này sau đó được đem đi nhân tích chập cho tín hiệu đầu vào $x(t)$ để từ đó tạo thành tín hiệu xử lý $\hat{x}(t)$. Qua đó, ta có

$$\hat{x}(t) = f(y(t)) = (y * g)(t), \quad (5.2)$$

với $(y * g)(t)$ là tích chập của hai tín hiệu $y(t)$ và $g(t)$. Chúng tôi bắt đầu với việc xét biến đổi Fourier của tích chập hai tín hiệu $y(t)$ và $g(t)$ được định nghĩa như sau

$$\begin{aligned}\mathcal{F}\{(y * g)(t)\}(\omega) &= \int_{-\infty}^{+\infty} \left(\int_{-\infty}^{+\infty} y(\tau)g(t - \tau)d\tau \right) e^{-\omega t i} dt \\ &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} y(\tau)e^{-\omega \tau i} g(t - \tau)e^{-\omega(t-\tau)i} d\tau dt \\ &= \int_{-\infty}^{+\infty} y(\tau)e^{-\omega \tau i} d\tau \int_{-\infty}^{+\infty} g(t - \tau)e^{-\omega(t-\tau)i} d(t - \tau) \\ &= \mathcal{F}\{y(t)\}(\omega)\mathcal{F}\{g(t)\}(\omega),\end{aligned}\tag{5.3}$$

với $\mathcal{F}\{\cdot\}(\omega)$ là biến đổi Fourier của tín hiệu với bước sóng ω . Như vậy thông qua công thức (5.3), rõ ràng cách tiếp cận cổ điển đang cố gắng tìm một tín hiệu $g(t)$ sao cho biến đổi của nó khiến cho độ mờ mát của âm thanh sau khi nhân tích chập và âm thanh sạch nhất. Vậy trong cách tiếp cận cổ điển này, ta sẽ đi cực tiểu hóa hàm mờ mát dưới đây trên toàn bộ miền giá trị T như sau

$$L(\hat{x}, x) = \int_T (\hat{x}(t) - x(t))^2 dt = \int_T ((y * g)(t) - x(t))^2 dt.\tag{5.4}$$

Lấy đạo hàm của hàm trên, ta có

$$\begin{aligned}\frac{d}{d(g(t))} L(\hat{x}, x) &= \int_T \frac{d}{d(g(t))} ((y * g)(t) - x(t))^2 dt \\ &= 2 \int_T ((y * g)(t) - x(t)) \frac{d}{d(g(t))} \hat{x}(t) dt \\ &= 2 \int_T ((y * g)(t) - x(t)) \frac{d}{d(g(t))} \left(\int_{-\infty}^{+\infty} y(\tau)g(t - \tau)d\tau \right) dt \\ &= 2 \int_T ((y * g)(t) - x(t)) \left(\int_{-\infty}^{+\infty} y(\tau)d\tau \right) dt \\ &= 2C \int_T ((y * g)(t) - x(t)) dt.\end{aligned}$$

Bằng các phép biến đổi tích phân Leibniz, chúng tôi đã khai triển đạo hàm của hàm mờ mát được sử dụng trong các bộ lọc truyền thống, chúng tôi thu được đạo hàm sẽ tỉ lệ tuyến tính với sự sai khác giữa tín hiệu dự đoán và tín hiệu sạch, chúng tôi lại có dưới biến đổi Fourier, sự sai khác giữa $\hat{x}(t)$ và $x(t)$ được thể hiện

$$\mathcal{F}\{(y * g)(t) - x(t)\}(\omega) = \mathcal{F}\{y(t)\}(\omega)\mathcal{F}\{g(t)\}(\omega) - \mathcal{F}\{x(t)\}(\omega),$$

và với trường hợp tối ưu nhất khi đạo hàm của công thức (5.4) là 0, tương ứng với đó, giá trị biến đổi Fourier sai khác cũng là 0, chúng tôi suy ra được giá trị của biến đổi Fourier tương ứng với bộ lọc $g(t)$ là

$$\mathcal{F}\{g(t)\}(\omega) = \frac{\mathcal{F}\{x(t)\}(\omega)}{\mathcal{F}\{y(t)\}(\omega)}, \quad (5.5)$$

Vậy chúng tôi thu được biến đổi Fourier của bộ lọc $g(t)$ được tối ưu như trên.

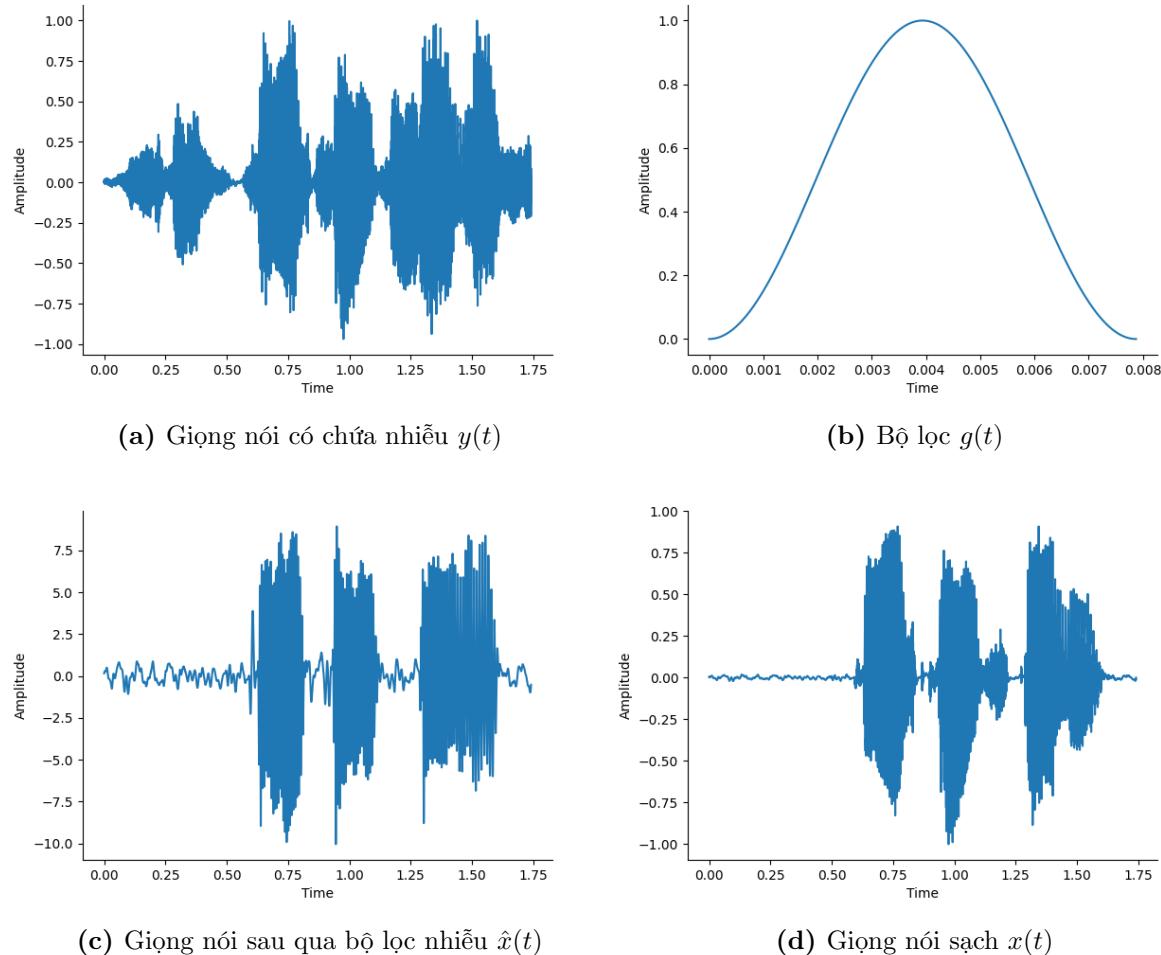
Dễ thấy được bộ lọc này là cố định cho tất cả các khoảng thời gian khác nhau, do đó, bộ lọc này sẽ hiệu quả với các âm nhiễu có spectrum cố định qua thời gian như nhiễu trắng và các loại nhiễu điện tử được gây ra bởi thiết bị. Nhưng hiệu quả của việc lọc này không được đảm bảo khi âm nhiễu có spectrum bị biến đổi theo thời gian. Đây cũng chính là hạn chế chính của cách tiếp cận cổ điển.

5.2 Tiếp cận theo bằng học sâu

Với mục tiêu khắc phục được hạn chế của cách tiếp cận truyền thống và tạo thành một bộ lọc có tính tự động cao với cả những loại nhiễu không được huấn luyện. Chúng tôi tiếp cận bài toán của mình theo hướng tiếp cận sử dụng học sâu thay cho bộ lọc $g(t)$ được nêu trong cách tiếp cận truyền thống.

Điều cần thiết nhất để có thể đưa ra một mô hình học máy tốt chính là dữ liệu và cách tiếp cận phù hợp. Việc xử lý dữ liệu và pha nhiễu sẽ được chúng tôi trình bày ở Mục 7.1, ở chương này, chúng tôi sẽ trình bày về các hướng tiếp cận của mô hình trong vấn đề lọc nhiễu trong âm thanh biểu diễn trên miền tần số thời gian (còn được gọi là time frequency domain). Trong miền tần số thời gian, âm thanh được biểu diễn dưới dạng spectrogram (hình ảnh biên độ của biến đổi Fourier thời gian ngắn). Spectrogram này sau đó được cho vào mô hình để dự đoán ra một “mask”, và nhân “mask” này với spectrogram ban đầu, sau khi thực hiện biến đổi Fourier thời gian ngắn nghịch, kết quả thu được đó chính là âm thanh xấp xỉ của giọng nói $\hat{x}(t)$ mà chúng tôi muốn mô hình lọc được.

Mô hình học sâu của chúng tôi tiếp cận theo hai hướng: “Mask” số thực và



Hình 5.1: Quá trình lọc âm nhiễu theo hướng tiếp cận cỗ điển

“Mask” số phức. Đối với “mask” số thực, mục tiêu của chúng tôi là cắt giảm những thành phần tần số nhiễu nhưng vẫn giữ lại các thành phần pha của chúng, nhưng điều này lại làm cho những phần nhiễu lẩn trong âm vẫn sẽ còn tồn tại và không bị loại bỏ hoàn toàn, tuy vậy, với cách tiếp cận học sâu, mô hình dự đoán ra một “mask” số phức từ đó có khả năng dự đoán cả thành phần biên độ lẩn góc của nhiễu tồn tại trong giọng nói từ đó có thể loại bỏ hoàn toàn được nhiễu ra khỏi âm. Dưới đây chúng tôi lần lượt sẽ trình bày về hai cách tiếp cận này và nêu các cách tiếp cận tiêu biểu mà chúng tôi tham khảo được.

5.2.1 Mask số thực trên miền tần số thời gian

Gọi $Y_k(\omega) = r_\omega e^{\phi_\omega i} \in \mathbb{C}$ là giá trị phức đại diện cho biến đổi Fourier thời gian ngắn tại khung thời gian thứ k với sóng có bước sóng ω của âm thanh đã pha nhiễu $y(t)$, $M_k(\omega) = r \in \mathbb{R}$ là giá trị “mask” ứng với giá trị của biến đổi Fourier thời gian ngắn. Giá trị biến đổi Fourier thời gian ngắn thu được sau khi áp “mask” này lên giá trị biến đổi Fourier thời gian ngắn của âm thanh sạch được tính như sau

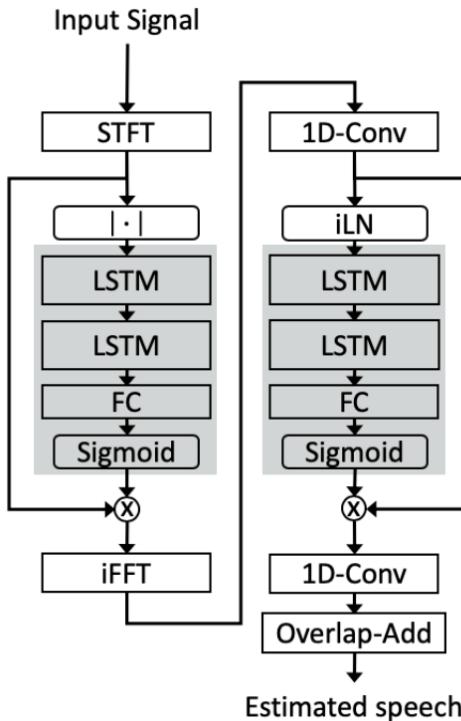
$$\hat{X}_k(\omega) = M_k(\omega)Y_k(\omega) = rr_\omega e^{\phi_\omega i}. \quad (5.6)$$

Với mask số thực, các hàm kích hoạt được sử dụng bên trong mạng hồi quy đều được giới hạn lại trong khoảng $(0, 1)$ hoặc $(-1, 1)$ khiến cho giá trị tối đa mà biên độ mới nhận được cũng sẽ chỉ nằm trong khoảng $(0, r_\omega]$. Đây chính là điều ta mong muốn, đối với âm thanh được mô hình xác định là giọng nói giá trị r này sẽ được đẩy lên rất gần với 1.0 từ đó biên độ sau khi lọc “mask” sẽ gần như bằng với giá trị biên độ gốc có trong âm thanh đầu vào r_ω . Tuy nhiên với biến đổi pha của giọng nói, “mask” số thực hoạt động không thực sự hiệu quả, bởi các giá trị pha đầu ra của mô hình (chỉ khi sử dụng tanh như hàm kích hoạt thì mới tồn tại sự điều chỉnh pha do phần âm $(-1, 0]$ của hàm tanh) chỉ là $k\pi$. Do vậy sự điều chỉnh về pha của giọng nói sẽ không có nhiều sự tác động, do vậy trong “mask” số thực, tác động của pha lên giọng nói sẽ được bỏ qua và giả định pha của giọng nói chính là pha trong giá trị đầu vào mô hình. Nhưng bù cho những khuyết điểm đó, các mô hình sử dụng “mask” số thực thường có cấu tạo đơn giản và hoạt động khá hiệu quả trong thực tế.

Dual signal Transformation LSTM. Mô hình đầu tiên mà chúng tôi muốn đề cập chính là Dual signal Transformation LSTM (DTLN). Kiến trúc của mô hình này được thể hiện như Hình 5.2.

Mô hình được chia làm hai phần chính: *phân lọc bằng cách chuyển tín hiệu sang miền tần số thời gian và mô hình tự học cách lọc trên miền biến đổi dữ liệu* (*chúng tôi gọi đây là miền thuộc tính - features domain*). Kết quả metrics của mô hình này trên bộ kiểm thử của chúng tôi được thể hiện trong Bảng 7.4. Hình 5.3 và Hình 5.4 là kết quả tại một số lớp trong mô hình DTLN.

Tuy với cách tiếp cận khá thú vị là tự học phép biến đổi dữ liệu ở mô hình này,



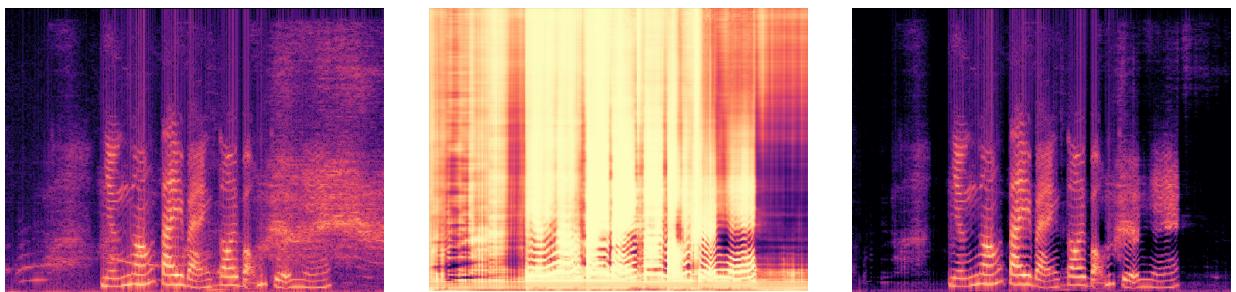
Hình 5.2: Kiến trúc mạng Dual signal Transformation LSTM (nguồn [3])

thì số lượng tham số cần tính toán để học hết phép biến đổi này chiếm khá nhiều chi phí tính toán. Nhưng kết quả lọc nhiễu của mô hình này khá tốt (Bảng 7.4) với điểm metrics khá cao trên tập kiểm thử của chúng tôi. Đồng thời với lượng tham số khoảng 967000 và tốc độ chạy thời gian thực trên một khung dữ liệu vào khoảng 2ms, chúng tôi cho rằng đây là một hướng tiếp cận phù hợp cho bài toán của mình. Do đó mà mô hình của chúng tôi thời điểm đầu chính là phiên bản thu gọn của mô hình DTLN này.

Instant Layer Normalization. Gọi F_k là một dãy tần số được trả về bởi biến đổi của biến đổi Fourier hay biến đổi do mô hình học được ở trên, tác giả [3] quy các vector này về một biến ngẫu nhiên và đặt ra phép chuẩn hóa trên biến này được định nghĩa như sau

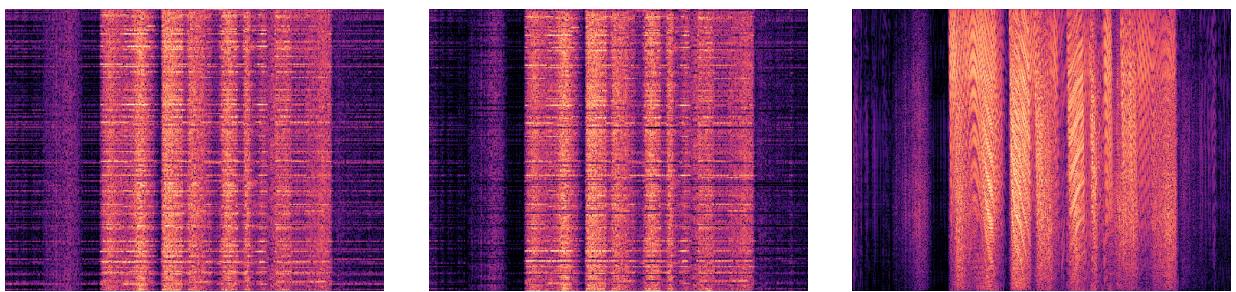
$$\tilde{F}_k = \gamma \frac{F_k - \bar{F}_k}{\sigma_{F_k}} + \beta, \quad (5.7)$$

với $\gamma, \beta \in \mathbb{R}$ và σ_{F_k} là độ lệch chuẩn được tính trên các kênh của F_k với trung bình thống kê \bar{F}_k . Giá trị $(F_k - \bar{F}_k)/\sigma_{F_k}$ là z-score được áp dụng vào đây nhằm mục đích chuẩn hóa phân phối của các giá trị trong F_k về phân phối có trung bình thống kê



(a) spectrogram của âm đầu vào (b) Mask được mô hình dự đoán (c) Kết quả của bước 1

Hình 5.3: Ở miền tần số thời gian, “mask” mà mô hình học được có khả năng xác định những phần nhiễu không liên quan tới giọng nói, ở bước này, chủ yếu mô hình học cách để loại bỏ các nhiễu mạnh làm cho giọng nói trong hơn bằng cách tổng hợp các sóng cosine lại với nhau thông qua biến đổi Fourier nghịch



(a) Đầu ra của lớp conv thứ nhất (b) Input của lớp conv thứ 2 (c) Đầu ra của lớp conv thứ 2

Hình 5.4: Ở bước này, mô hình học cách biểu diễn âm thanh đã được lọc qua ở phần trên sang một miền mới và tách các nhiễu ở đó, kì vọng với cách tiếp cận này, mô hình có thể loại bỏ hầu hết các nhiễu bên trong âm thanh mà làm mất mát ít thông tin nhất

là 0 và độ lệch chuẩn là 1.

Phần γ và β chính là dùng để thay đổi độ lệch chuẩn và trung bình thống kê của phân phối vừa nhận được, phân phối này được sử dụng chung cho toàn bộ các khung thời gian của chung một đoạn âm thanh nên mô hình sẽ cố gắng học ra được phân bố đem lại nhiều thông tin nhất có thể. Ngoài ra để tăng hiệu quả của phép chuẩn hóa này, tác giả cũng sử dụng hàm log lên biên độ được trả về bởi biến đổi Fourier và giá trị tự học của mô hình trước khi đem đi chuẩn hóa bằng công thức ở bên trên.

5.2.2 Mask số phức trên miền tần số thời gian

Xuất phát như một cải tiến của hướng tiếp cận theo “mask” số thực, “mask” số phức cho phép mô hình điều chỉnh cả về biên độ lẫn pha của âm thanh bên trong giọng nói cho phép mô hình loại bỏ hoàn toàn âm thanh nhiễu ra khỏi giọng nói. Điều này xuất phát từ định nghĩa của âm thanh, bất cứ một loại sóng âm nào cũng phải có ba thành phần cơ bản: *biên độ*, *pha* và *tốc độ góc*. Trong quá trình truyền tải, các nhiễu được tổng hợp lại với âm thanh

$$y(t) = x(t) + n(t).$$

Việc tổng hợp sóng âm như vậy, về bản chất cũng làm thay đổi cấu tạo của âm từ đó làm thay đổi luôn cả spectrogram của giọng nói ban đầu. Gọi $X_k(\omega)$ là spectrogram của giọng nói và $N_k(\omega)$ là spectrogram của âm nhiễu, do biến đổi Fourier có tính chất tuyến tính, nên spectrogram của âm tổng hợp $Y_k(\omega)$ sẽ là

$$Y_k(\omega) = X_k(\omega) + N_k(\omega).$$

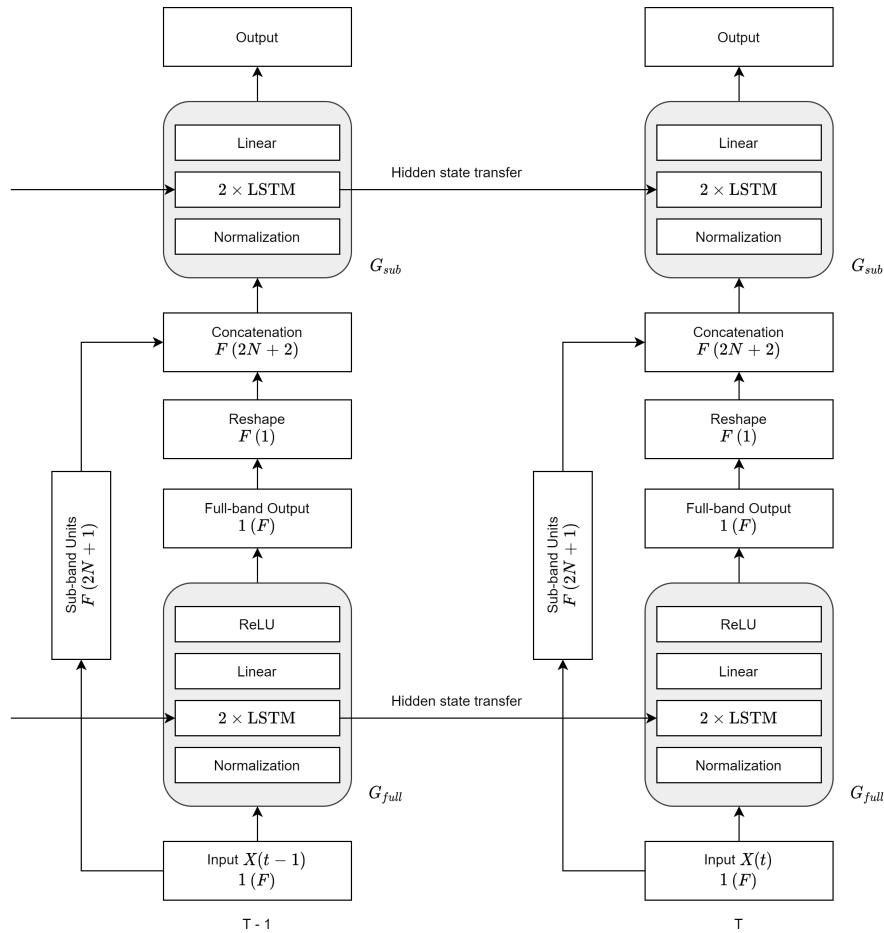
Thông qua việc tổng hợp này, biên độ và pha của $Y(\omega)$ đã bị biến đổi đi so với giọng nói ban đầu $X(\omega)$. Ở “mask” số thực, việc nhân thêm một giá trị $r \in \mathbb{R}$ cũng chỉ làm thay đổi giá trị biên độ của âm tổng hợp, cho dù pha có bị thay đổi một góc $k\pi$. Nhưng ở “mask” số phức, mô hình được tùy ý lựa chọn giá trị điều chỉnh cho cả biên độ lẫn tần số của âm, từ đó có thể cho chất lượng âm thanh tốt hơn so với “mask” số thực. Dưới đây là minh họa bằng công thức cho ý tưởng của cách tiếp cận dùng mask số phức lên spectrogram, với $M_k(\omega) = re^{\phi i}$ là “mask” số phức mà chúng tôi đề cập, $Y_k(\omega) = r_\omega e^{\phi_\omega i}$ là âm nhiễu đầu vào

$$\hat{X}_k(\omega) = M_k(\omega)Y_k(\omega) = rr_\omega e^{(\phi_\omega + \phi)i}.$$

Dễ thấy so với mô hình sử dụng “mask” số thực, mô hình sử dụng “mask” số phức cho phép biến đổi cả biên độ lẫn pha của âm nhiễu để lọc triệt để âm nhiễu còn tồn tại bên trong giọng nói. Đây cũng là “mask” được sử dụng các mô hình đạt được các thứ hạng cao trong các cuộc thi (FullSubNet đạt hạng 2 và DCCRN đạt hạng 3 trong cuộc thi Deep Noise Suppression [14]). Nhưng rõ ràng để có thể

học được sự biến đổi về cả biên độ lẫn pha, mô hình sẽ có cấu trúc tương đối phức tạp hơn so với mô hình sử dụng “mask” số thực, chi phí tính toán (được thể hiện thông qua số lượng tham số trong mô hình) cũng tăng lên đáng kể so với mô hình sử dụng “mask” số thực.

Mô hình sử dụng “mask” số phức mà chúng tôi tìm hiểu là mô hình **Fullband and Subband Fusion Model (FullSubNet)**. Phát triển dựa trên nền tảng của mô hình Sub-band [4], FullSubNet lọc nhiễu thông qua hai bước: *tổng hợp thông tin từ miền tần số thời gian sử dụng tất cả các dãy tần số* và *lọc nhiễu sử dụng các thông tin đã được tổng hợp*. Hình 5.5 miêu tả kiến trúc của mô hình này. Để hiểu rõ hơn cấu tạo cũng như chức năng của các bước, chúng tôi sẽ giới thiệu mô hình làm tiền đề cho FullSubNet là mô hình Subband.



Hình 5.5: Kiến trúc của mô hình Fullband and Subband Fusion Model (nguồn [1])

Subband Model. Với các cách tiếp cận phổ biến hiện giờ, hầu hết các mô hình đều sử dụng lớp LSTM với dữ liệu đầu vào là toàn bộ tần số tại một thời điểm,

nhưng cách với cách học như vậy lượng tham số đầu vào sẽ trở nên rất lớn, với động lực như vậy, tác giả đã sử dụng một phần cục bộ dữ liệu của tần số thay vì toàn bộ dữ liệu tần số trong một khung thời gian của miền tần số thời gian. Việc thay đổi từ sự phụ thuộc toàn cục sang cục bộ làm giảm lượng tham số của LSTM đầu vào trong mô hình ở Hình 5.6.

Để rõ ràng hơn, chúng tôi định nghĩa $X_k(\omega_i)$ là giá trị tại khung thời gian thứ k của sóng mang bước sóng ω_i lên giọng nói sạch, $Y_k(\omega_i)$ tương tự là giá trị tại khung thời gian k của sóng có bước sóng ω_i nhưng là của âm đĩa pha nhiễu, và $N_k(\omega_i)$ là dữ liệu của âm nhiễu. Do sự quan hệ tuyến tính trong biến đổi Fourier, chúng tôi dễ dàng có được biểu thức

$$Y_k(\omega_i) = X_k(\omega_i) + N_k(\omega_i),$$

và một mô hình f sử dụng toàn bộ dữ liệu tại khung thời gian k này, thì kết quả trả về sẽ là

$$\hat{X}_k(\omega_i) = f(|Y_k|) \times X_k(\omega_i),$$

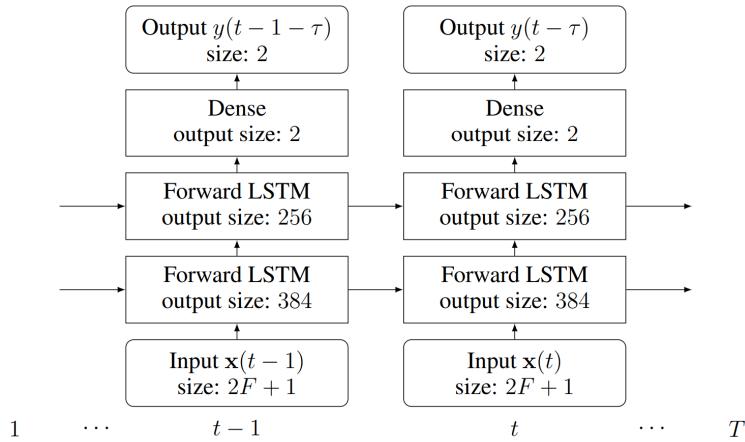
với $|Y_k|$ là vector chứa biên độ của tất cả các giá trị phức tại khung thời gian thứ k trong spectrogram. Nhưng vì tác giả trong [4] đề xuất sử dụng thay vì toàn bộ băng tần số như vậy rất khó có thể xác định quan hệ giữa các vùng tần số với nhau cũng như để xác định được nhiễu, họ sử dụng một dữ liệu thay thế

$$\hat{X}_k(\omega_i) = f(D_k(\omega_i, m)) \times X_k(\omega_i),$$

với

$D_k(\omega_i, m) = (|Y_k(\omega_{i-m})|, \dots, |Y_k(\omega_{i-1})|, |Y_k(\omega_i)|, |Y_k(\omega_{i+1})|, \dots, |Y_k(\omega_{i+m})|) \in \mathbb{R}^{2m+1}$ là một phần bao gồm m giá trị bước sóng kè cận với bước sóng trung tâm ω_i .

Rõ ràng với cách định nghĩa như vậy, mô hình có nhiều dữ liệu hơn để có thể học được các quan hệ giữa các vùng tần số với nhau. Đây chính là điểm cốt lõi của mô hình này, thay đổi dạng dữ liệu đầu vào của mô hình để giúp mô hình có nhiều thông tin hơn về âm thanh đầu vào.



Hình 5.6: Kiến trúc của mô hình Subband (nguồn [4])

Kết quả đầu ra của mô hình này cũng là một “mask” số phức với hai phần thực và ảo được dự đoán riêng biệt, nhân “mask” này với spectrogram ban đầu, thu lại được chính là một spectrogram sạch đã được lọc nhiễu.

Fullband and Subband Fusion Model. Ở mô hình FullSubNet, phần biến đổi tần số, gom cụm các tần số lại được thể hiện ở Sub-band Units trong Hình 5.5, đó chính là phần chuyển đổi từ Y_k sang $D_k(\omega_i, m)$ như chúng tôi đã đề cập. Phần tổng hợp thông tin thông qua sự tổng hợp spectrogram G_{full} được sử dụng thông qua hai lớp LSTM để tạo thành lớp tần số mới tại lớp Reshape, sau đó được tổng hợp với kết quả của Sub-band Units để tạo thành đầu vào cho mô hình Subband G_{sub} . Nhưng nhược điểm lớn nhất theo chúng tôi thấy của mô hình này đó là số lượng tham số lớn (khoảng 5 triệu tham số) và không phù hợp cho việc chạy trong thời gian thực trên các máy tính có tài nguyên hạn chế, dù rằng chất lượng âm thanh đầu ra bởi mô hình này hơn hẳn các mô hình khác mà chúng tôi khảo sát (dựa vào các điểm metrics được công bố) nhưng sau khi kiểm thử mô hình được huấn luyện sẵn, chúng tôi lại thu được kết quả khá thất vọng (Bảng 7.4).

5.3 Đề xuất hàm mất mát và mô hình

Với mục tiêu dùng mô hình có kiến trúc đơn giản hơn, tiết kiệm chi phí tính toán và quan trọng nhất đối với một ứng dụng thời gian thực mà chất lượng âm thanh vẫn đảm bảo không bị suy giảm so với mô hình gốc ban đầu (through các điểm

metrics đạt được ở các Bảng 7.6, Bảng 7.9, Bảng 7.7 và Bảng 7.5), để đạt được các điều trên, chúng tôi đề xuất cải tiến hàm mất mát để sử dụng huấn luyện mô hình của mình.

5.3.1 Hàm mất mát đề xuất

Xuất phát từ thực tế các mô hình mà chúng tôi thực nghiệm, hiện tượng bị khuyết âm thanh, nhiễu vẫn còn tồn tại ở những vùng xung quanh âm chính mà không bị loại bỏ hoàn toàn (điều này được đánh giá thông qua điểm BAK của metric DNSMOS và được chúng tôi phân tích ở Hình 5.8). Chúng tôi đề xuất cải tiến giúp mô hình tăng cường khả năng lọc nhiễu, nhưng nhiễu có biên độ dù không lớn thì vẫn sẽ bị loại bỏ triệt để. Trước hết, ta sẽ tìm hiểu một số hàm mất mát thông dụng sẽ được sử dụng trong phần này nhằm kết hợp để tạo nên hàm mất mát đề xuất.

Signal Distortion Rate. Signal Distortion Rate (tạm dịch là độ gián đoạn dữ liệu, SDR) còn được gọi là độ gián đoạn dữ liệu và được định nghĩa như sau

$$\text{SDR}(x, \hat{x}) = 10 \log_{10} \left(\frac{\|x\|_2^2}{\|x - \hat{x}\|_2^2} \right). \quad (5.8)$$

Công thức này đo cường độ của nhiễu còn tồn tại trong giọng nói sau khi lọc ($x - \hat{x}$) so với giọng nói sạch x thông qua công thức tính SNR (5.9). Sự khác biệt giữa x và \hat{x} càng lớn thì phần bên trong hàm log có giá trị càng gần về 0, và do đó giá trị SDR thu được sẽ dần tiến tới $-\infty$. Trong trường hợp ngược lại, khi \hat{x} dần tiến tới x làm cho phần bên trong hàm log tiến dần tới $+\infty$ và như vậy kéo theo giá trị của SDR cũng sẽ lớn lên theo. Để thấy, công thức của SDR có thể được chuyển đổi sang công thức của SNR bởi hệ thức sau

$$\text{SNR}(x, y) = 10 \log_{10} \left(\frac{\|x\|_2^2}{\|y - x\|_2^2} \right) = 10 \log_{10} \left(\frac{\|x\|_2^2}{\|n\|_2^2} \right), \quad (5.9)$$

trong đó x là giọng nói sạch, y là giọng đã trộn nhiễu, n là âm nhiễu.

Nhưng bản thân SDR vẫn còn tồn tại một số vấn đề, một vấn đề nghiêm trọng nhất có khả năng ảnh hưởng tới chất lượng của mô hình khi sử dụng hàm này đó

là việc giọng nói bị nhân thêm cho một hằng số sau khi lọc nhiễu. Như đã đề cập, mục tiêu bài toán của chúng ta là đi ước lượng lại giọng nói ban đầu từ giọng nói đã bị trộn nhiễu. Vậy nên trường hợp giọng nói bị nhân thêm một hằng số sau khi lọc nhiễu (tức là $\hat{x} = \mu x$ với $\mu \in \mathbb{R}^+$) là hoàn toàn có thể xảy ra. Nhưng việc giọng nói bị nhân hay không giọng nói mới là thứ quan trọng, tức là giá trị biên độ của chúng là không quan trọng. Do đó giá trị mà ta kì vọng mà SDR trả về sẽ tiến đến $+\infty$ nhưng khi tính toán, ta lại có

$$\begin{aligned} \text{SDR}(x, \hat{x}) &= 10 \log_{10} \left(\frac{\|x\|_2^2}{\|x - \hat{x}\|_2^2} \right) \\ &= 10 \log_{10} \left(\frac{\|x\|_2^2}{\|x - \mu x\|_2^2} \right) \\ &= 10 \log_{10} \left(\frac{1}{\|1 - \mu\|_2^2} \right). \end{aligned} \quad (5.10)$$

Rõ ràng giá trị này phụ thuộc vào hằng số μ , nghĩa rằng hai giọng nói này dù là chung một giọng nói gốc nhưng chỉ cần nhân thêm một hằng số lại thì chúng không giống nhau và đây không phải điều mà chúng ta kì vọng. Hàm SDR đang đánh giá sai trường hợp này, và đó cũng chính là lý do mà ta tìm hiểu hàm măt măt tiếp theo, Scale Invariant SDR, để khắc phục tình trạng trên.

Scale Invariant SDR. Vấn đề lớn nhất của SDR đó chính là dù $\hat{x} = \mu x$ với $\mu \in \mathbb{R}^+$ đi chăng nữa thì SDR vẫn mang xác định hai âm trên là khác biệt, nhưng rõ ràng việc nhân thêm một hằng số vào giọng nói không ảnh hưởng nhiều tới giá trị thu được vì vấn đề chính cần quan tâm ở đây là nhiễu có được loại bỏ đi hay không chứ không phải là giọng nói được nhân thêm bao nhiêu lần. Để giải quyết việc này, Scale Invariant SDR [28], bổ sung thêm một thành phần khiến cho giá trị của măt măt không còn phụ thuộc vào biên độ của giọng sạch và giọng nói dự đoán nữa. Về mặt toán học, SI-SDR được định nghĩa như sau

$$\text{SI-SDR}(x, \hat{x}) = 10 \log_{10} \left(\frac{\|\alpha x\|_2^2}{\|\alpha x - \hat{x}\|_2^2} \right), \quad (5.11)$$

với $\alpha = \arg \min_{\alpha} \|\alpha x - \hat{x}\|_2^2$.

Bằng cách lấy đạo hàm và tính cực trị theo α , ta thu được

$$\frac{d}{d\alpha} \|\alpha x - \hat{x}\|_2^2 = 2(\alpha x - \hat{x})^T x = 0, \quad (5.12)$$

ta có thể giải ra được giá trị của α là

$$\alpha = \frac{\hat{x}^T x}{\|x\|_2^2}. \quad (5.13)$$

Để kiểm chứng tính hiệu quả của hàm SI-SDR, sinh viên thực hiện tiến hành phân tích α cho trường hợp giọng nói bị nhân lên sau khi lọc nhiều như đã nói ở trên để thu được

$$\begin{aligned} \alpha &= \frac{\hat{x}^T x}{\|x\|_2^2} \\ &= \frac{\|\hat{x}\|_2 \|x\|_2 \cos(\angle \hat{x}, x)}{\|x\|_2^2} \\ &= \frac{\|\hat{x}\|_2 \cos(\angle \hat{x}, x)}{\|x\|_2}. \end{aligned} \quad (5.14)$$

Khi nhân hệ số này với x , giá trị của x lúc này được chuẩn hóa sau khi chia cho $\|x\|_2$, nhân lên một lượng bằng với $\|\hat{x}\|_2 \times \cos(\angle \hat{x}, x)$. Và giá trị hằng số nhân này phụ thuộc vào $\|\hat{x}\|_2$ cho nên dù cho giá trị đó tăng lên chừng nào thì x cũng sẽ được tăng lên một lượng tương ứng.

Để rõ hơn làm thế nào mà SI-SDR khắc phục tình trạng $\hat{x} = \mu x$ ta đã thấy ở trên, ta sẽ phân tích cách mà SI-SDR đánh giá trường hợp này. Trước hết, ta có giá trị α như sau

$$\begin{aligned} \alpha &= \frac{\|\hat{x}\|_2 \cos(\angle \hat{x}, x)}{\|x\|_2} \\ &= \frac{\mu \|x\|_2 \times 1}{\|x\|_2} \\ &= \mu. \end{aligned} \quad (5.15)$$

Lúc này, do \hat{x} chỉ là x được nhân thêm một lượng μ dương nên chỉ số α mà ta tính được bằng đúng giá trị μ này. Thay tất cả vào công thức của SI-SDR, ta thu được giá trị hàm này như sau

$$\begin{aligned}
 \text{SI-SDR}(x, \hat{x}) &= 10 \log_{10} \left(\frac{\|\alpha x\|_2^2}{\|\alpha x - \hat{x}\|_2^2} \right) \\
 &= 10 \log_{10} \left(\frac{\|\mu x\|_2^2}{\|\mu x - \mu x\|_2^2} \right) \\
 &= 10 \log_{10} \left(\frac{\|\mu x\|_2^2}{\|0\|_2^2} \right) \\
 &= +\infty.
 \end{aligned} \tag{5.16}$$

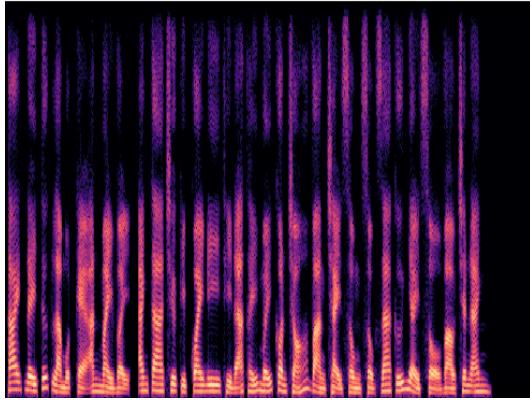
Lúc này SI-SDR đã đánh giá đúng bản chất nhiều bên trong hai giọng nói x và \hat{x} là rất nhỏ so với âm thanh sạch, đây cũng là một ví dụ cho thấy hàm SI-SDR có khả năng đánh giá linh hoạt hơn SDR trong trường hợp âm thanh ở những vùng biên độ khác nhau.

Đè xuất cải tiến hàm măt măt. Từ các quan sát về mặt dữ liệu, các tần số quan trọng mang thông tin giọng nói có biên độ trả về bởi biến đổi Fourier cao (phần màu vàng trong Hình 5.7a) và những phần kém quan trọng hơn không mang nhiều thông tin mà chỉ đơn thuần tạo nên âm điệu của giọng nói. Đối với nhiều, tuy sự biến đổi về mặt tần số cấu tạo theo thời gian diễn ra liên tục nhưng sự tương đồng về các tần số cấu tạo này với các tần số cấu tạo của giọng nói là khá cách biệt (bằng việc so sánh hai spectrogram ở Hình 5.7a và Hình 5.7b). Các mô hình thường được huấn luyện bằng các hàm măt măt phổ biến như SDR và SI-SDR cho kết quả nhiều nhiễu vẫn còn tồn đọng lại bên trong giọng nói sau khi lọc. Và đối với một số trường hợp nhiễu quá lớn sẽ gây ra tình trạng măt giọng nói. Những điều này có thể được thấy rõ thông qua điểm SIG của metric DNSMOS, metric STOI và PESQ.

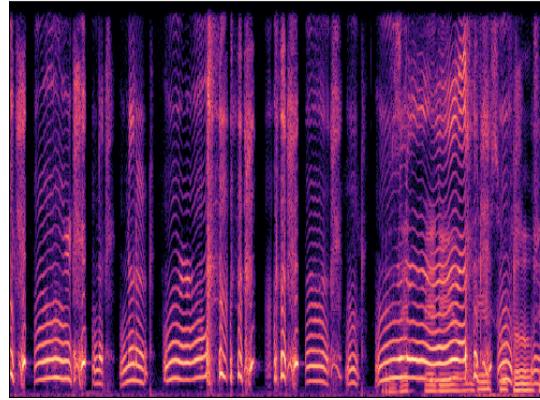
Để khắc phục các hạn chế này, chúng tôi đề xuất một hàm măt măt cải tiến được định nghĩa như sau

$$L(x, \hat{x}, n) = L_{main}(x, \hat{x})((1 + \alpha L_1(x, \hat{x})) - \beta |L_2(\hat{x}, n)|), \tag{5.17}$$

với L_{main} có thể là một trong các hàm măt măt thường dùng như SNR, SI-SDR; còn L_1 và L_2 lần lượt được định nghĩa bởi

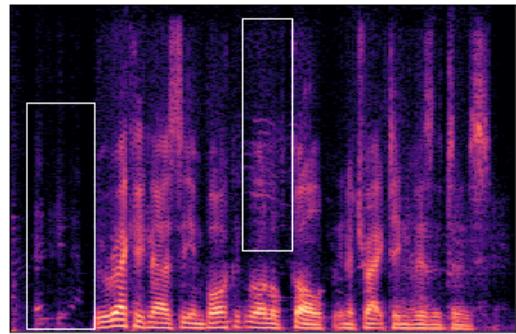
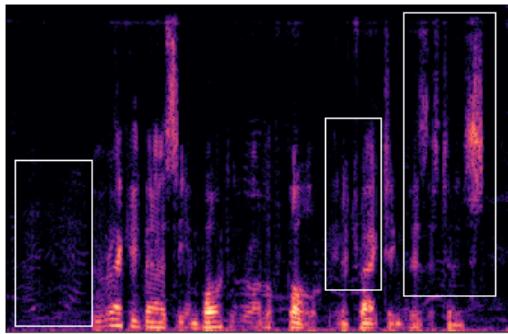


(a) spectrogram của giọng nói sạch



(b) spectrogram của âm nhiễu

Hình 5.7: Ý tưởng hình thành hàm tăng cường mất mát



Hình 5.8: Nhiễu vẫn còn tồn tại bên trong spectrogram âm sạch mà không bị lọc đi hoàn toàn (phần được khoanh trong hình chữ nhật màu trắng)

$$\begin{aligned} L_1(x, \hat{x}) &= \cos(\angle X, \hat{X}) \\ &= \frac{\text{FLAT}(|X|)^T \text{FLAT}(|\hat{X}|)}{\|\text{FLAT}(|X|)\|_2 \|\text{FLAT}(|\hat{X}|)\|_2}, \end{aligned} \quad (5.18)$$

$$\begin{aligned} L_2(\hat{x}, n) &= \cos(\angle \hat{X}, N) \\ &= \frac{\text{FLAT}(|\hat{X}|)^T \text{FLAT}(|N|)}{\|\text{FLAT}(|\hat{X}|)\|_2 \|\text{FLAT}(|N|)\|_2}, \end{aligned} \quad (5.19)$$

trong đó

- X, \hat{X}, N lần lượt là biến đổi Fourier thời gian ngắn của x, \hat{x} và n .
- Phép đuôi $\text{FLAT}(x)$ dùng để chuyển x từ miền $\mathbb{R}^{T \times F}$ sang thành miền \mathbb{R}^{TF} .
- Phép $|\cdot|$ trong công thức (5.18) và (5.19) là thể hiện cho phép lấy module cho từng số phức bên trong ma trận X, \hat{X} hay N .

Hai hàm tăng cường L_1 và L_2 được thực hiện dựa trên các ý tưởng về quan hệ giữa giọng nói dự đoán được với giọng nói sạch và nhiễu. Hàm L_1 chính là độ đo tương tự cosine giữa giọng nói dự đoán và giọng nói sạch, theo công thức (5.18). Các giá trị mà tại đó biến đổi Fourier cao (hay các thành phần thông tin mà chúng tôi muốn giữ lại) sẽ chiếm ưu thế trong phần tổng phía trên tử và ngược lại các thành phần nhỏ hơn sẽ không có nhiều đóng góp vào giá trị này.

Tương tự với L_1 , tuy nhiên với L_2 do đây là độ tương tự cosine giữa giọng nói dự đoán với nhiễu có trong giọng nói ghi nhận, nên giá trị này cần phải được tối thiểu hóa về 0. Các giá trị $\alpha, \beta \in \mathbb{R}^+$ là các giá trị thực nghiệm được chúng tôi thử nghiệm trên các lần huấn luyện mô hình, các giá trị này đạt hiệu quả ở $\alpha = 1.0$, $\beta = 2.0$ đối với các mô hình Post và $\alpha = 0.7$, $\beta = 2.0$ đối với các mô hình còn lại.

Tuy nhiên, để có thể tìm hiểu sâu hơn về khả năng tối ưu của hàm măt mát cải tiến này, chúng tôi xem xét đạo hàm của chúng theo \hat{x} . Ta không xét đạo hàm theo tham số của mô hình, vì tuy với các hàm măt mát khác nhau nhưng đạo hàm của chúng theo tham số θ luôn có dạng

$$\frac{d}{d\theta} L = \frac{d}{d\hat{x}} L \frac{d}{d\theta} \hat{x},$$

với \hat{x} được xem là đầu ra của mô hình. Do đó, với L_{main} là hàm SI-SDR được sử dụng để huấn luyện, chúng tôi xét đạo hàm của toàn bộ hàm măt mát theo \hat{x} thay vì bộ tham số θ . Ta có

$$\begin{aligned} \frac{d}{d\hat{x}} L_{main}(x, \hat{x}) &= \frac{d}{d\hat{x}} \left(10 \log_{10} \left(\frac{\|\gamma x\|_2^2}{\|\gamma x - \hat{x}\|_2^2} \right) \right) \\ &= \frac{20}{\ln(10)} \frac{\|\gamma x - \hat{x}\|_2^2}{\|\gamma x\|_2^2} \frac{\|\gamma x\|_2^2}{\|\gamma x - \hat{x}\|_2^4} (\gamma x - \hat{x}) \\ &= \frac{20}{\ln(10)} \frac{(\gamma x - \hat{x})}{\|\gamma x - \hat{x}\|_2^2}. \end{aligned}$$

Mặt khác

$$\begin{aligned} \left\| \frac{d}{d\hat{x}} L_{main}(x, \hat{x}) \right\|_2 &= \left\| \frac{20}{\ln(10)} \frac{(\gamma x - \hat{x})}{\|\gamma x - \hat{x}\|_2^2} \right\|_2 \\ &\leq C_m \frac{1}{\|\gamma x - \hat{x}\|_2}. \end{aligned} \tag{5.20}$$

Và hơn nữa

$$\begin{aligned}
 \frac{d}{d\hat{x}} \hat{X}(\omega, \tau) &= \frac{d}{d\hat{x}} \left(\int_{-\infty}^{+\infty} w(t - \tau) \hat{x}(t) e^{-\omega t i} dt \right) \\
 &= \int_{-\infty}^{+\infty} w(t - \tau) \frac{d}{d\hat{x}} (\hat{x}(t)) e^{-\omega t i} dt \\
 &= \int_{-\infty}^{+\infty} w(t - \tau) e^{-\omega t i} dt \\
 &= C,
 \end{aligned}$$

với C là một hằng số theo ω và τ , nên từ đó đạo hàm của toàn bộ spectrogram theo \hat{x} sẽ được biểu diễn như sau

$$\begin{aligned}
 \frac{d}{d\hat{x}} \hat{X} &= \int_{\omega} \int_{\tau} \frac{d}{d\hat{x}} \hat{X}(\omega, \tau) d\omega d\tau \\
 &= \int_{\omega} \int_{\tau} C d\omega d\tau \\
 &= C.
 \end{aligned} \tag{5.21}$$

Như vậy đạo hàm trên toàn bộ spectrogram theo \hat{x} sẽ cho ta một hằng số.

Gọi

$$L_e = ((1 + \alpha L_1) - \beta |L_2|), \tag{5.22}$$

ta có

$$\begin{aligned}
 \frac{d}{d\hat{x}} L_e &= \alpha \frac{d}{d\hat{x}} L_1 - \beta s(L_2) \frac{d}{d\hat{x}} L_2 \\
 &= \sum \left(\alpha \frac{|X|}{\|X\|_2} - \beta s(L_2) \frac{|N|}{\|N\|_2} \right) \frac{d}{d\hat{x}} \frac{|\hat{X}|}{\|\hat{X}\|_2},
 \end{aligned} \tag{5.23}$$

với

$$s(x) = \begin{cases} 1, & \text{nếu } x \geq 0, \\ -1, & \text{nếu } x < 0. \end{cases} \tag{5.24}$$

Ta khai triển tiếp tục đạo hàm của hàm tăng cường mắt mát

$$\begin{aligned} \frac{d}{d\hat{x}} \frac{|\hat{X}|}{\|\hat{X}\|_2} &= \left(\frac{1}{\|\hat{X}\|_2} - \frac{|\hat{X}|^2}{\|\hat{X}\|_2^3} \right) \frac{d}{d\hat{x}} |\hat{X}| \\ &= C \left(\frac{1}{\|\hat{X}\|_2} - \frac{|\hat{X}|^2}{\|\hat{X}\|_2^3} \right) \frac{\hat{X}}{|\hat{X}|}, \end{aligned}$$

và lấy chuẩn Euclid của đạo hàm

$$\begin{aligned} \left\| \frac{d}{d\hat{x}} L_e \right\|_2 &= \left\| \sum \left(\alpha \frac{|X|}{\|X\|_2} - \beta s(L_2) \frac{|N|}{\|N\|_2} \right) \frac{d}{d\hat{x}} \frac{|\hat{X}|}{\|\hat{X}\|_2} \right\|_2 \\ &\leq C_1 \left\| \frac{d}{d\hat{x}} \frac{|\hat{X}|}{\|\hat{X}\|_2} \right\|_2 \\ &= C_1 C_2 \left\| \left(\frac{1}{\|\hat{X}\|_2} - \frac{|\hat{X}|^2}{\|\hat{X}\|_2^3} \right) \frac{\hat{X}}{|\hat{X}|} \right\|_2 \\ &\leq C_1 C_2 C_3 \left\| \frac{1}{\|\hat{X}\|_2} - \frac{|\hat{X}|^2}{\|\hat{X}\|_2^3} \right\|_2 \\ &\leq C_1 C_2 C_3 \left(\left\| \frac{1}{\|\hat{X}\|_2} \right\|_2 + \left\| \frac{|\hat{X}|^2}{\|\hat{X}\|_2^3} \right\|_2 \right) \\ &\leq C_1 C_2 C_3 \left(\left\| \frac{1}{\|\hat{X}\|_2} \right\|_2 + \left\| \frac{\|\hat{X}\|_2^2}{\|\hat{X}\|_2^3} \right\|_2 \right) \\ &\leq C_e \frac{1}{\|\hat{X}\|_2}. \end{aligned} \tag{5.25}$$

Chúng tôi gộp hai đạo hàm của hàm măt măt chính L_{main} và hàm cải tiến do chúng tôi đề xuất L_e , từ đó thu được

$$\left\| \frac{d}{d\hat{x}} (L_{main} L_e) \right\|_2 \leq \|L_{main}\|_2 \left\| \frac{d}{d\hat{x}} L_e \right\|_2 + \|L_e\|_2 \left\| \frac{d}{d\hat{x}} L_{main} \right\|_2. \tag{5.26}$$

Thay các đạo hàm được tính toán ở trên thay vào công thức trên, ta thu được

$$\begin{aligned} \left\| \frac{d}{d\hat{x}} (L_{main} L_e) \right\|_2 &\leq \|L_{main}\|_2 \left\| \frac{d}{d\hat{x}} L_e \right\|_2 + \|L_e\|_2 \left\| \frac{d}{d\hat{x}} L_{main} \right\|_2 \\ &\leq C_e \log \left(\frac{1}{\|\alpha x - \hat{x}\|_2^2} \right) \frac{1}{\|\hat{X}\|_2} + C_m \|L_e\|_2 \frac{1}{\|\alpha x - \hat{x}\|_2}. \end{aligned}$$

Vì $\|L_e\|_2 \leq (1 + \alpha)$, do đó

$$\begin{aligned} \left\| \frac{d}{d\hat{x}}(L_{main}L_e) \right\|_2 &\leq C_e \log \left(\frac{1}{\|\alpha x - \hat{x}\|_2^2} \right) \frac{1}{\|\hat{X}\|_2} + C_m \|L_e\|_2 \frac{1}{\|\alpha x - \hat{x}\|_2} \\ &\leq C_e \log \left(\frac{1}{\|\alpha x - \hat{x}\|_2^2} \right) \frac{1}{\|\hat{X}\|_2} + C_m(1 + \alpha) \frac{1}{\|\alpha x - \hat{x}\|_2}. \end{aligned} \quad (5.27)$$

Ta lại có

$$\begin{aligned} \|\hat{X}\|_2^2 &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} |w(t - \tau)\hat{x}(t)e^{-\omega t i}|^2 dt d\tau d\omega \\ &= \int_{-\infty}^{+\infty} d\omega \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} |w(t - \tau)\hat{x}(t)|^2 dt d\tau \\ &\leq \int_{-\infty}^{+\infty} d\omega \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} |w(t - \tau)|^2 |\hat{x}(t)|^2 dt d\tau \\ &= C_\omega C_w \|\hat{x}\|_2^2, \end{aligned} \quad (5.28)$$

nếu lấy căn hai về của bất đẳng thức, ta thu được

$$\|\hat{X}\|_2 \leq C_\omega C_w \|\hat{x}\|_2. \quad (5.29)$$

Ta cũng thu được các kết quả tương tự kết quả tương tự với các cặp spectrogram và dữ liệu miền thời gian khác. Thay kết quả vừa tìm được vào (5.27), gọi N' là spectrogram của âm nhiễu còn lại sau khi lọc, ta có

$$\begin{aligned} \left\| \frac{d}{d\hat{x}}(L_{main}L_e) \right\|_2 &\leq C_e \log \left(\frac{1}{\|\gamma x - \hat{x}\|_2^2} \right) \frac{1}{\|\hat{X}\|_2} + C_m(1 + \alpha) \frac{1}{\|\gamma x - \hat{x}\|_2} \\ &\leq C_e \log \left(\frac{1}{\|N'\|_2^2} \right) \frac{1}{\|\gamma X + N'\|_2} + C_m(1 + \alpha) \frac{1}{\|N'\|_2}. \end{aligned} \quad (5.30)$$

Từ công thức (5.30), ta thấy rằng có hai trường hợp đạo hàm lấn át (một trong hai đạo hàm rất lớn so với đạo hàm còn lại). Trường hợp đầu tiên xảy ra khi giá trị chuẩn Euclid của N' rất lớn so với giá trị chuẩn Euclid của X tức là mô hình đang ở những epochs đầu trong quá trình huấn luyện, giá trị chuẩn Euclid của spectrogram nhiễu N' lúc này tiệm cận đến vô cùng. Do đó, đạo hàm tổng lúc này sẽ là

$$\begin{aligned}
 \left\| \frac{d}{d\hat{x}}(L_{main}L_e) \right\|_2 &\leq C_e \log \left(\frac{1}{\|N'\|_2^2} \right) \frac{1}{\|\gamma X + N'\|_2} + C_m(1+\alpha) \frac{1}{\|N'\|_2} \\
 &\leq C_e \log \left(\frac{1}{\|N'\|_2^2} \right) \frac{1}{\|\gamma X\|_2} + C_m(1+\alpha) \frac{1}{\|N'\|_2} \\
 &\approx C_e \log \left(\frac{1}{\|N'\|_2^2} \right) \frac{1}{\|\gamma X\|_2}.
 \end{aligned}$$

Khi đó, đạo hàm sẽ gần như chỉ phụ thuộc vào hàm mất mát tăng cường của ta mà lấn át đi giá trị của đạo hàm từ L_{main} .

Tuy nhiên, trong trường hợp ngược lại, tức giá trị chuẩn Euclid của N' lúc này rất nhỏ so với chuẩn Euclid X hay giá trị chuẩn Euclid của N' lúc này sẽ tiệm cận về 0 và mô hình đang ở những epochs sau của quá trình huấn luyện, lúc này ta lại có

$$\begin{aligned}
 \left\| \frac{d}{d\hat{x}}(L_{main}L_e) \right\|_2 &\leq C_e \log \left(\frac{1}{\|N'\|_2^2} \right) \frac{1}{\|\gamma X + N'\|_2} + C_m(1+\alpha) \frac{1}{\|N'\|_2} \\
 &\leq C_e \log \left(\frac{1}{\|N'\|_2^2} \right) \frac{1}{\|\gamma X\|_2} + C_m(1+\alpha) \frac{1}{\|N'\|_2} \\
 &\approx C_m(1+\alpha) \frac{1}{\|N'\|_2}.
 \end{aligned}$$

Tức là với những epochs sau, đạo hàm của hàm cải tiến không còn lấn át nữa mà ngược lại, nó bị hàm L_{main} lấn át đi. Điều này được chúng tôi quan sát bằng thực nghiệm, ở một vài epoch đầu, mô hình hội tụ với giá trị cosine L_1 rất nhanh về một giá trị (chúng tôi quan sát được giá trị này thường là 0.9) trước khi chậm lại và tăng dần dần lên giá trị tối đa (0.98) ở tất cả các epochs sau đó trong quá trình huấn luyện.

Vậy qua những phân tích vừa rồi, việc sử dụng hàm tăng cường mất mát của chúng tôi sẽ khiến quá trình huấn luyện được chia làm hai giai đoạn tương ứng với hai trường hợp L_e lấn át L_{main} và L_{main} lấn át L_e . Kết quả và quá trình chuẩn bị cho huấn luyện cũng như kiểm thử của luận văn này được trình bày trong Chương 7.

5.3.2 Mô hình đè xuất

Sau khi đã khảo sát một số mô hình trong cuộc thi Deep Noise Suppression [14] của Microsoft, tuy các mô hình sử dụng “mask” số phức lớn như DCCRN [2] và

FullSubNet [1], điểm metrics được công bố cao hơn hẳn các mô hình nhỏ sử dụng “mask” số thực như DTLN [3] nhưng chúng tôi tìm thấy được ở các mô hình này một điểm chung. Tất cả các mô hình mà chúng tôi tham khảo được đều sử dụng hai lớp LSTM (hoặc RNN) chồng lên nhau như một bộ lọc nhiễu chính.

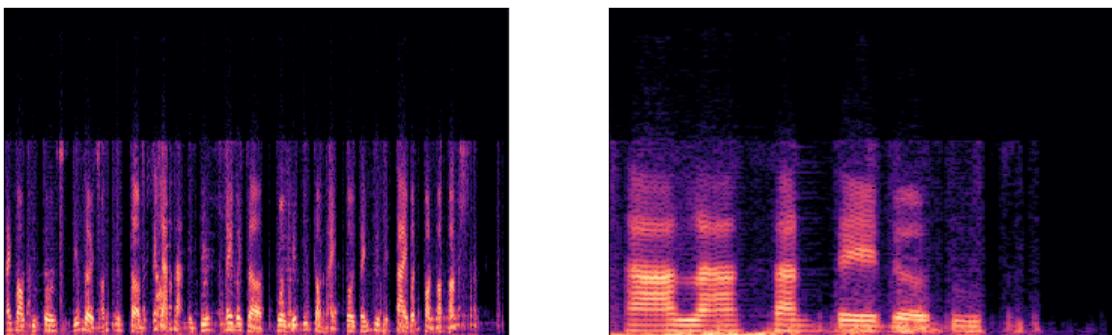
Ở DTLN, hai lớp LSTM chồng lên nhau được sử dụng trong cả hai phần của mô hình, tương tự cho FullSubNet, hai LSTM được đặt ở phần mạng Subband. Đối với DCCRN, với cách tiếp cận từ UNet, sau khi qua các lớp Encoder (một tập hợp các lớp tích chập 2 chiều) vẫn là hai lớp RNN được sử dụng để dự đoán ra “mask” số phức lọc nhiễu. Do đó, ý tưởng về hai lớp LSTM chồng lên nhau cũng chính là khởi nguồn cho mô hình đề xuất của chúng tôi.

Tiêu chí	Mục tiêu
Tham số	< 700000
Thời gian chạy	< 8ms

Bảng 5.1: Bảng các tiêu chí để thiết kế mạng học sâu

Phiên bản 1: Mô hình sơ khởi. Với các yêu cầu được nêu trong Bảng 5.1, điều rõ ràng là ngoài vấn đề về độ hiệu quả của mô hình chúng tôi cần phải đảm bảo cả vấn đề về số lượng tham số và độ trễ của mô hình, ngoài ra nguồn tài nguyên được sử dụng để huấn luyện mô hình cũng cần phải được cân nhắc. Như đã đề cập ở phần trước, đây là một mô hình sử dụng “mask” số thực, tuy kết quả của mô hình này so với các mô hình “mask” số phức khác không được tốt lắm nhưng kết quả của mô hình này lại tốt hơn hẳn so với các mô hình khác khi đem đi kiểm thử trên bộ dữ liệu của chúng tôi (Bảng 7.6). Hơn nữa lượng tham số nhỏ hơn khá nhiều so với hai mô hình “mask” số phức khác được chúng tôi tìm hiểu là FullSubNet và DCCRN. Do đó, DTLN là mô hình khởi đầu trong luận văn này.

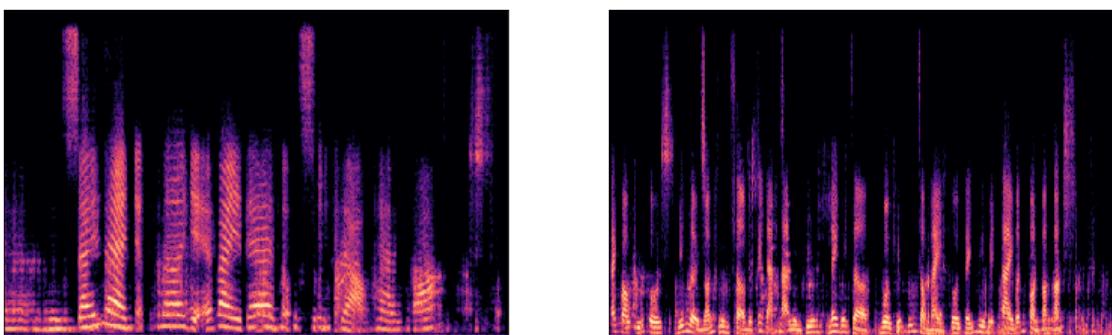
Bị ràng buộc bởi các yêu cầu về thời gian thực và cả nguồn tài nguyên huấn luyện, chúng tôi bắt đầu từ việc cắt giảm tham số của mô hình DTLN. Việc cắt giảm tham số được bắt nguồn từ bộ phận lọc nhiễu, 4 lớp LSTM 128 units, 2 lớp tích chập 1 chiều lần lượt có kích thước 128 và 512. Chúng tôi bắt đầu cắt giảm lượng tham số bên trong LSTM. Các lớp LSTM sau khi được cắt giảm đạt kết quả tốt nhất ở 64 units, 2 lớp tích chập cũng được chúng tôi tinh chỉnh xuống còn 64 và 256. Việc cắt giảm này cho kết quả lượng tham số mô hình được giảm xuống 3 lần (vào khoảng 363.000 tham số). Nhưng việc cắt giảm này cũng để lại một số



Hình 5.9: Một số kết quả được kết quả huấn luyện của mô hình DTLN thu gọn bị cắt một nửa tần số, nhiễu vẫn còn khá nhiều trong spectrogram sau khi đã lọc

hậu quả, vì lớp tích chập 1 chiều sau cùng chỉ có kích thước 256 nên khi qua giải thuật Overlap Add, các giá trị tần số bị cắt ngắn đi một nửa (tần số lấy mẫu của chúng tôi là 16000 mẫu/s nên giới hạn tần số Nyquist sẽ nằm ở 8000 Hz, sau khi cắt giảm nó chỉ còn là 4000 Hz).

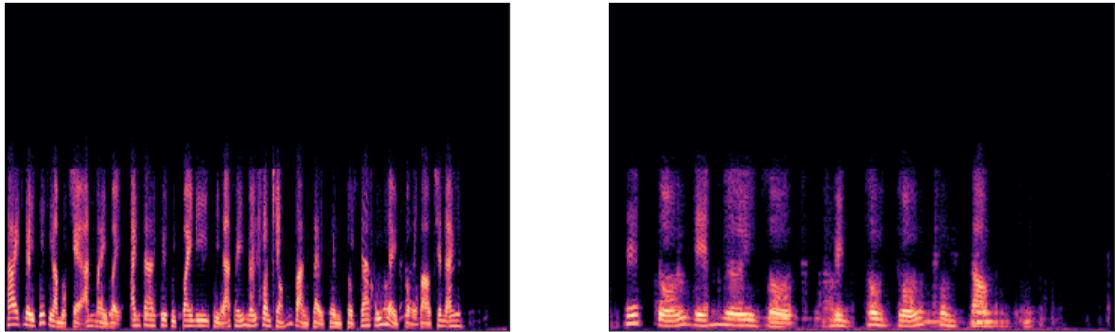
Để kiểm thử chất lượng mô hình này ngoài việc sử dụng metrics để kiểm thử, chúng tôi còn sử dụng con người để kiểm thử các kết quả mô hình¹. Nhưng các phản hồi về cho chúng tôi rằng chất lượng lọc nhiễu còn khá kém, tiếng rè của các âm sau khi lọc, và chính những hạn chế này của mô hình DTLN thu gọn đã giúp chúng tôi tìm kiếm ra được mô hình sơ khởi của mình.



Hình 5.10: Một số kết quả được kết quả huấn luyện của mô hình Post, kết quả được cải thiện đáng kể so với mô hình DTLN thu gọn

Với mục tiêu ban đầu là cải thiện chất lượng của đầu ra mô hình DTLN thu gọn, chúng tôi đè xuất mô hình **Post**. Bằng nhiều thử nghiệm, chúng tôi nhận thấy rằng kiến trúc LSTM chồng, cho phép truyền trạng thái từ LSTM trước sang

¹Kiểm thử tại <https://colab.research.google.com/drive/1Z8Gv2V71yNUhmHVm4Rf-X9fleOxbtZzm>



Hình 5.11: Một số kết quả được kết quả huấn luyện của mô hình kết hợp giữa DTLN thu gọn và Post

Lớp	Lượng tham số	Kích thước đầu ra	
1	Input	($N \times T \times 257$)	
2	InstantNorm	514	($N \times T \times 257$)
3	LSTM 64 Tanh	82432	($N \times T \times 64$), ($N \times 64$), ($N \times 64$)
4	Dense 128 Tanh	8320	($N \times T \times 128$)
5	LSTM 64 Tanh	49408	($N \times T \times 64$)
6	Dense 257 Tanh	16705	($N \times T \times 257$)
Tổng		157379	

Bảng 5.2: Cấu trúc của mô hình Post sơ khởi

cho LSTM sau và có một lớp Dense xen giữa hai lớp LSTM này đem lại kết quả khả quan nhất.

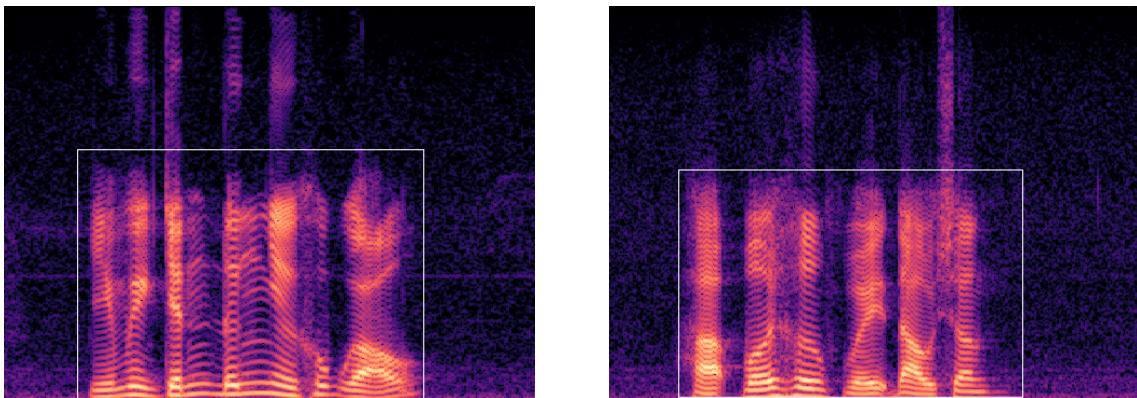
Kết quả là mô hình Post lúc này còn cho kết quả tương đương (trên bộ kiểm thử của tập Edinburgh [17]) với cả mô hình DTLN dù lượng tham số chỉ bằng 1/5 và cấu trúc mô hình cũng đơn giản hơn rất nhiều.

Bảng 5.2 mô tả cấu trúc mạng Post, lớp LSTM đầu (lớp thứ 2) ngoài việc trả về kết quả còn trả về thêm cả các trạng thái sau khi chạy. Các trạng thái này được chuyển tới lớp LSTM thứ 2 (lớp thứ 4) và được sử dụng để lọc lại một lần nữa trước khi qua lớp Dense và trả về “mask” số thực cho spectrogram ban đầu.

Phiên bản 2: Lookahead. Được khá nhiều bài báo khác sử dụng [1, 4], lookahead cho phép sử dụng nhiều khung thời gian để dự đoán cho thời gian hiện tại. Tuy việc này sẽ làm tăng lượng tham số đầu vào và tăng độ trễ (vì sử dụng các khung thời gian sau đó để dự đoán cho khung hiện tại) của mô hình lên, tuy nhiên cách làm này đem lại kết quả rất tốt (Bảng 7.6).

Theo như chúng tôi quan sát được từ dữ liệu, các tần số biến đổi âm thanh

trong spectrogram không thay đổi một cách đột ngột (từ tần số thấp lên một tần số nào đó cao hơn ngay lập tức hoặc ngược lại) mà chúng luôn có sự thay đổi một cách từ từ và đều đặn như được thể hiện trong Hình 5.12.

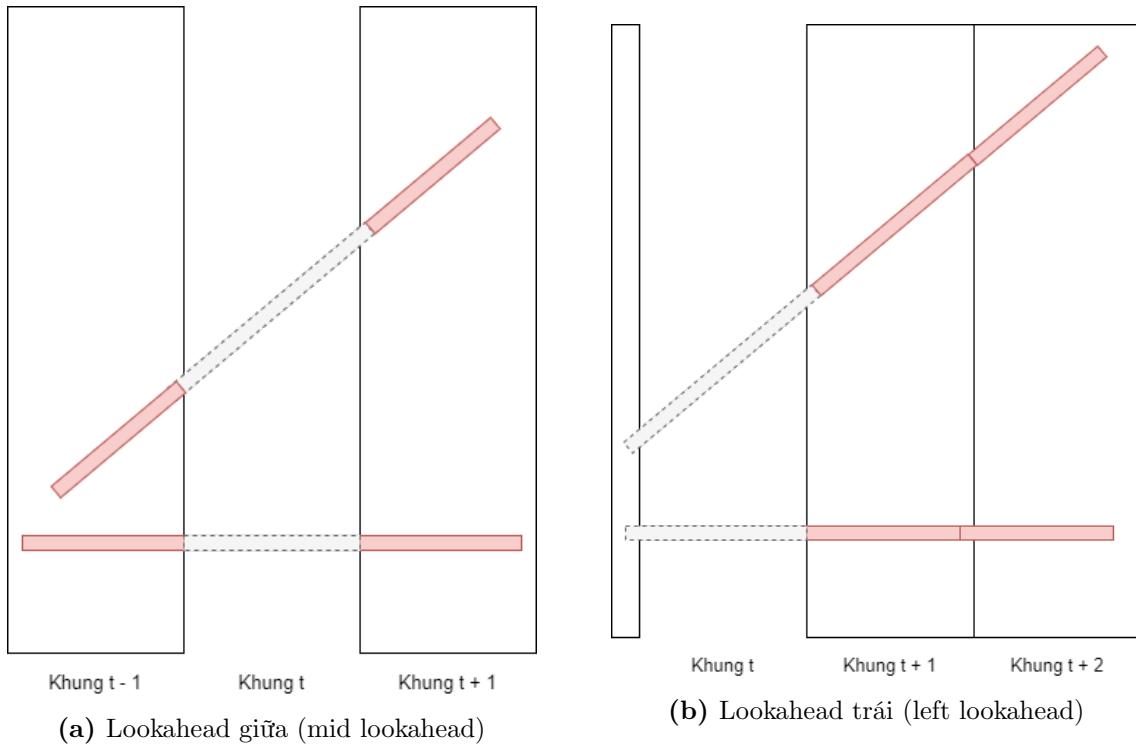


Hình 5.12: Tần số thay đổi dần dần theo thời gian, tuy biên độ có lúc mạnh lúc nhẹ nhưng tần số vẫn thay đổi đều đặn không bị đột ngột cho đến khi kết thúc

Lấy ý tưởng từ quan sát sự thay đổi này, lookahead tận dụng các khung phía sau (trong trường hợp chúng tôi sử dụng là cả phía trước khung hiện tại) để dự đoán ra kết quả cho khung hiện tại. Để minh họa cho ý tưởng này, chúng tôi diễn tả như trong Hình 5.13. Giả định rằng ta đang có hai dữ liệu tần số tại khung thứ $t - 1$ và khung thứ $t + 1$ (các phần được tô màu đỏ), rõ ràng theo những quan sát từ dữ liệu mà chúng tôi đã đề cập, khung thứ t được dự đoán (phần được tô xám) sao cho nó sẽ làm cho tần số cấu tạo biến đổi đều đặn và không bị gián đoạn đột ngột.

Việc sử dụng lookahead vào trong mô hình Post của chúng tôi sẽ làm thay đổi cấu trúc của mô hình lại như trong Bảng 5.3. Số lượng tham số trong mô hình của chúng tôi lúc này khoảng 288000 tham số, các tham số lúc này tập trung chủ yếu ở lớp LSTM thứ nhất (khoảng 214016 tham số) chiếm khoảng 74% lượng tham số trong mạng, điều này gây ra sự bất cân đối giữa các lớp. Cũng ở cải tiến này, chúng tôi bắt đầu phân hóa chức năng của các lớp trong mô hình. Cụ thể:

1. *Mã hóa dữ liệu:* Ở bước này, mô hình cố gắng học cách để mã hóa dữ liệu, như đã nêu trong Bảng 5.3 dữ liệu đầu vào được nối từ 3 vectors lấy từ spectrum của 3 khung kề cận nhau để làm dữ liệu đầu vào cho bộ phận lọc nhiễu chính.
2. *Bộ lọc nhiễu chính:* Bộ lọc nhiễu này thực chất chính là mô hình Post nguyên



Hình 5.13: Ý tưởng sử dụng lookahead để dự đoán kết quả hiện tại của chúng tôi

nguyên bản của chúng tôi được nêu trong Bảng 5.2, dữ liệu từ bộ phận mã hóa được đưa vào bộ phận này để dự đoán ra “mask” số thực cho spectrogram đầu vào.

Phiên bản 3: Cải tiến Lookahead. Một vấn đề đối với mô hình sử dụng lookahead đó chính là kích thước dữ liệu đầu vào sẽ tăng tuyến tính theo lượng khung thời gian nhìn được và lượng tham số cũng như độ trễ của mô hình cũng

Lớp	Lượng tham số	Kích thước đầu ra
1	Input	$(N \times T \times 257)$
2	InstantNorm	$(N \times T \times 257)$
3	ZeroPadding 0, 2	$(N \times (T + 2) \times 257)$
4	Framing 3, 1	$(N \times T \times 3 \times 257)$
5	Concat	$(N \times T \times 771)$
6	LSTM 64 Tanh	$(N \times T \times 64), (N \times 64), (N \times 64)$
7	Dense 128 Tanh	$(N \times T \times 128)$
8	LSTM 64 Tanh	$(N \times T \times 64)$
9	Dense 257 Tanh	$(N \times T \times 257)$
Tổng	288963	

Bảng 5.3: Cấu trúc của mô hình Post sử dụng lookahead

từ đó mà tăng theo.

Ở phần cải tiến thứ hai, chúng tôi đã cho thấy kiến trúc mô hình sử dụng lookahead, tuy chỉ là sử dụng ba khung thời gian (2 khung sau và 1 khung ở hiện tại). Nhưng tại lớp LSTM đầu vào, lượng tham số tăng lên rất nhiều, vì vậy, nếu chúng tôi muốn sử dụng lookahead hiệu quả hơn đòi hỏi chúng tôi cần một cơ chế để có thể vừa thu gọn lượng tham số vừa có thể đảm bảo các thông tin đầu vào mô hình. Với những yêu cầu này, chúng tôi đề xuất ra một cơ chế *nén dữ liệu* sử dụng mạng tích chập.

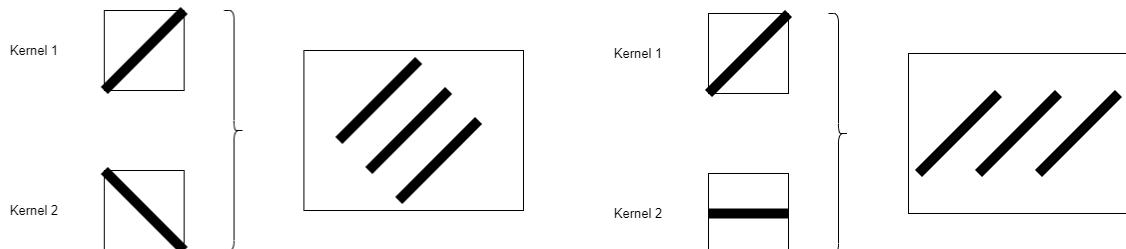
Như ở phần trước chúng tôi đã có giới thiệu, mô hình chúng tôi cơ bản sẽ được chia làm hai phần: *Nén dữ liệu* và *Bộ lọc nhiễu chính*. Trong phần này, các cải tiến của chúng tôi sẽ tập trung vào phần nén dữ liệu. Mục tiêu lần này của chúng tôi là lookahead trong năm khung thay vì chỉ ba như ở cải tiến hai và cắt giảm lượng tham số sẽ được sử dụng trong mô hình. Vì lookahead trong năm khung sẽ sẽ làm cho dữ liệu đầu vào tăng lên rất nhiều (kích thước của đầu vào mô hình lúc này sẽ là $(N \times T \times 1285)$) và làm cho kích thước của LSTM đầu vào lúc này cũng tăng lên theo (theo chúng tôi ước lượng, lớp LSTM đầu vào sẽ có khoảng 345600 tham số). Để làm điều này, trước hết chúng tôi chia đầu vào thành các khoảng năm khung thời gian trùng lặp nhau (1 khung phía trước và 3 khung phía sau khung hiện tại). Sau đó chúng tôi cho tập hợp dữ liệu này qua một số lớp tích chập sử dụng hàm kích hoạt tuyến tính và lần lượt đi qua các bước masking để nén dữ liệu và cuối cùng là trả về kết quả cho bộ lọc nhiễu. Chi tiết các lớp được chúng tôi liệt kê ở Bảng 5.4.

Từ dữ liệu đầu vào, để đảm bảo mô hình sẽ sử dụng một khung trước và ba khung sau để dự đoán khung thời gian hiện tại, chúng tôi mở rộng biên của đoạn âm thanh đầu vào bằng cách bù không nhờ đó mà dữ liệu đầu vào chúng tôi từ T sẽ được mở rộng lên thành $T + 4$ khung thời gian. Sau khi đã bù không và chắc chắn rằng mô hình sẽ sử dụng năm khung liên tục để dự đoán khung hiện tại, chúng tôi tiến hành cắt $T + 4$ khung thời gian này ra thành từng cụm năm khung liên tục nhau, sau mỗi lần cắt, chúng tôi dịch lên một khung và lại tiếp tục lấy năm khung đó làm thành một cụm và cứ như vậy, chúng tôi có được T cụm dữ liệu đầu vào có kích thước (5×257). Sau đó chúng tôi thực hiện hai lần tích chập với kernel (3×3) lên cụm dữ liệu này, sở dĩ chúng tôi sử dụng hai lần tích chập với kernel ($3 \times$

Lớp	Lượng tham số	Kích thước đầu ra
1	Input	$(N \times T \times 257)$
2	ZeroPadding 1, 3	$(N \times (T + 4) \times 257)$
3	Framing 5, 1	$(N \times T \times 5 \times 257)$
4	Expand Dims	$(N \times T \times 5 \times 257 \times 1)$
5	Conv2D 32 (3×3) Linear	320 $(N \times T \times 3 \times 255 \times 32)$
6	Conv2D 32 (3×3) Linear	9248 $(N \times T \times 1 \times 253 \times 32)$
7	Reduce Dims	x $(N \times T \times 253 \times 32)$
8	Framing 4, 4	$(N \times T \times 63 \times 4 \times 32)$
9	Reshape	$(N \times T \times 63 \times 32 \times 4)$
10	Masking	20 $(N \times T \times 63 \times 32 \times 4)$
11	Summing	$(N \times T \times 63 \times 32)$
12	Dense 4 Tanh	132 $(N \times T \times 63 \times 4)$
13	Concat	$(N \times T \times 252)$
Tổng		9720

Bảng 5.4: Cấu trúc của bộ nén dữ liệu ở cải tiến thứ 3

3) mà lại không sử dụng một lần tích chập kernel (5×5) là vì một phần là do giới hạn về phần cứng của chúng tôi không cho phép và một phần khác cũng là để tận dụng tính chất được miêu tả trong Hình 5.14 của tích chập. Việc sử dụng hai lớp kernel chồng lên nhau giúp chúng tôi có thể phát hiện nhiều khuôn mẫu (pattern) của cùng một loại kernel trên hình đầu vào.



Hình 5.14: Minh họa tính chất của tích chập khi sử dụng kernel 2 lên kết quả tích chập của kernel 1 lên hình đầu vào sẽ cho phép phát hiện ra nhiều khuôn mẫu tương tự nhau

Tận dụng tính chất này của tích chập và đặc tính của âm thanh là một số tần số thường có sự biến đổi đồng thời với nhau tạo nên các cạnh rõ nét thể hiện lên spectrogram, từ đó tổng hợp ra các đặc tính đặc trưng trong bốn khung thời gian liên tiếp nhau (giữa các cụm bốn khung này không có overlap) thông qua masking và summing như trong Bảng 5.4. Cuối cùng, dữ liệu thông qua một lớp biến đổi và trả về cho bộ lọc nhiễu chính.

Như vậy sau ba lần cải tiến, chúng tôi đã có được mô hình lọc nhiễu mới trong luận văn tốt nghiệp này. Tuy nhiên trong quá trình hiện thực vào ứng dụng việc

Lớp	Lượng tham số	Kích thước đầu ra
1 Input		(N × T × 257)
2 ZeroPadding 1, 3		(N × (T + 4) × 257)
3 Framing 5, 1		(N × T × 5 × 257)
4 Expand Dims		(N × T × 5 × 257 × 1)
5 Conv2D 32 (3 × 3) Linear	320	(N × T × 3 × 255 × 32)
6 Conv2D 32 (3 × 3) Linear	9248	(N × T × 1 × 253 × 32)
7 Reduce Dims		(N × T × 253 × 32)
8 Framing 4, 4		(N × T × 63 × 4 × 32)
9 Reshape		(N × T × 63 × 32 × 4)
10 Masking	20	(N × T × 63 × 32 × 4)
11 Summing		(N × T × 63 × 32)
12 Dense 4 Tanh	132	(N × T × 63 × 4)
13 Concat		(N × T × 252)
14 LSTM 64 Tanh	81152	(N × T × 64), (N × 64), (N × 64)
15 Dense 128 Tanh	8320	(N × T × 128)
16 LSTM 64 Tanh	49408	(N × T × 64)
17 Dense 257 Tanh	16705	(N × T × 257)
Tổng	165305	

Bảng 5.5: Cấu trúc của mô hình Post sử dụng bộ nén dữ liệu cải tiến

Lớp	Lượng tham số	Kích thước đầu ra
1 Input		(N × T × 257)
2 Bộ nén dữ liệu	9720	(N × T × 252)
3 LSTM 64 Tanh	81152	(N × T × 64), (N × 64), (N × 64)
4 Dense 128 Tanh	8320	(N × T × 128)
5 LSTM 64 Tanh	49408	(N × T × 64)
6 Dense 257 Tanh	16705	(N × T × 257)
Tổng	165305	

Bảng 5.6: Cấu trúc của mô hình lọc nhiễu tinh

truyền trạng thái ở lớp LSTM thứ nhất và thứ hai không đem lại nhiều hiệu quả, do các đầu vào dưới dạng một stream liên tục và các đầu vào này thường rất ngắn (khoảng 64ms cho mỗi lần chạy). Vì vậy để khắc phục tình trạng này, chúng tôi quyết định tách mô hình được cải tiến nêu ở Bảng 5.5 ra thành hai loại: *Mô hình lọc nhiễu tinh* và *Mô hình lọc nhiễu động*. Mô hình lọc nhiễu tinh vẫn giữ nguyên cấu trúc của mô hình Post ở Phiên bản 3, chỉ có mô hình lọc nhiễu động sẽ được loại bỏ các trạng thái liên kết từ LSTM thứ nhất sang LSTM thứ hai và giữ nguyên các lớp còn lại. Sự khác biệt giữa hai loại mô hình này được nêu trong Bảng 5.6 và Bảng 5.7.

Tuy các theo các cách tiếp cận của chúng tôi đè xuất đều sử dụng “mask” số thực, nhưng chúng tôi vẫn cần nhắc sử dụng “mask” số phức trong các thử nghiệm của mô hình. Các mô hình Post số phức có cấu tạo tương tự như các mô hình được

Lớp	Lượng tham số	Kích thước đầu ra
1	Input	(N × T × 257)
2	Bộ nén dữ liệu	(N × T × 252)
3	LSTM 64 Tanh	(N × T × 64)
4	Dense 128 Tanh	(N × T × 128)
5	LSTM 64 Tanh	(N × T × 64)
6	Dense 257 Tanh	(N × T × 257)
Tổng	165305	

Bảng 5.7: Cấu trúc của mô hình lọc nhiễu động

đề xuất ở trên nhưng các lớp tính toán được thay thành các lớp được hiện thực dưới cơ chế phức được đề xuất bởi [2]. Tuy nhiên, số lượng tham số và thời gian tính toán của mô hình bị tăng lên rất nhiều (các lớp LSTM phức được cấu tạo từ hai LSTM thực vậy nên lượng tham số sẽ bị tăng lên hai lần), với mô hình Post Phiên bản 2 được hiện thực “mask” phức, lượng tham số tăng lên xấp xỉ 700000 tham số cho ba lookahead. Do đó, chúng tôi quyết định không hiện thực mô hình Post Phiên bản 3 dùng “mask” số phức để làm mô hình chính cho ứng dụng.

Chương 6

Phân tích thiết kế hệ thống

Trong chương này, chúng tôi sẽ trình bày về thiết kế của ứng dụng sử dụng các mô hình lọc nhiễu của chúng tôi. Ứng dụng của chúng tôi được chia làm hai phần: *Lọc nhiễu tĩnh* và *Lọc nhiễu động*. Phần lọc nhiễu tĩnh có chức năng lọc các file âm thanh được người dùng đưa vào hoặc từ dữ liệu được ghi âm lại thông qua giao diện ứng dụng chính. Trong khi đó, phần lọc nhiễu động được thiết kế như một ứng dụng độc lập khởi ứng dụng chính, là một dịch vụ của hệ điều hành có chức năng lấy dữ liệu từ microphone thật rồi lọc nhiễu và sau đó đưa lại vào microphone ảo thông qua cơ chế quản lý driver của hệ điều hành.

6.1 Yêu cầu hệ thống

Dựa vào các đặc điểm về dữ liệu, chúng tôi chia dữ liệu đầu vào của hệ thống thành hai loại:

1. *Dữ liệu tĩnh*: Dạng dữ liệu này được lưu trữ vĩnh viễn trên đĩa cứng hoặc tạm thời trên RAM thông qua cơ chế ghi âm của ứng dụng. Dữ liệu tĩnh không biến đổi theo thời gian, nếu một đoạn âm thanh đã được ghi âm hoặc đã được ghi xuống đĩa cứng thì chúng sẽ luôn nằm ở đó cho đến khi bị xóa đi, hoặc ứng dụng bị đóng. Vậy nên dễ thấy, đối với dữ liệu này, chất lượng lọc nhiễu là yếu tố quan trọng nhất, thời gian chạy mô hình để lọc nhiễu có thể lớn và không có sự ràng buộc cho thời gian chạy này.

2. *Dữ liệu động*: Trái với dữ liệu tĩnh, dữ liệu động là một luồng âm thanh liên tục được chuyển từ microphone tới hệ thống lọc nhiễu và việc lọc nhiễu được thực hiện trong thời gian thực. Vậy nên ngoài việc đảm bảo rằng mô hình chúng tôi phải trả về giọng nói đã được lọc nhiễu có chất lượng tốt, ta còn phải đảm bảo rằng mô hình này cũng có khả năng hoạt động trong thời gian thực và không tiêu tốn quá nhiều tài nguyên của người dùng. Đây cũng chính là các ràng buộc của dữ liệu đặt ra cho mô hình để từ đó các điều kiện ràng buộc về lượng tham số cũng như thời gian chạy như trong Bảng 5.1 được đặt ra ở Chương 5.

Thông qua các nhin nhận về đặc điểm của từng loại dữ liệu, ta có một số ràng buộc đối với từng loại hệ thống được nêu trong Bảng 6.1.

Yêu cầu	Lọc nhiễu tĩnh	Lọc nhiễu động
Chất lượng	Tốt	Tốt
Độ trễ	Thấp	Có thể cao

Bảng 6.1: Mô tả yêu cầu về mặt dữ liệu của hệ thống

Từ các yếu tố được nêu ra ở trên, chúng tôi có thể lần lượt đặt ra các yêu cầu phi chức năng cho từng hệ thống từ đó đề xuất ra các thiết kế phù hợp. Chi tiết các yêu cầu phi chức năng được chúng tôi nêu ở Bảng 6.2 và Bảng 6.3.

Lọc nhiễu tĩnh và Lọc nhiễu động	
1	Lượng tiêu thụ CPU không vượt quá 25% (khoảng 1 core đổi với CPU 4 cores)
2	Lượng RAM tiêu thụ dưới 4GB
3	Âm thanh sau khi lọc không còn hoặc ít nhiễu so với ban đầu

Bảng 6.2: Yêu cầu phi chức năng chung đối với cả hai hệ thống

	Lọc nhiễu tĩnh	Lọc nhiễu động
4	Cho phép lọc nhiễu trên các file có thời gian dài tối khoảng 20 phút	Khung lọc nhiễu cố định 64ms
5	Độ trễ được phép cao	Độ trễ mô hình phải thấp (nhỏ hơn thời gian chuyển khung là 8ms)

Bảng 6.3: Yêu cầu phi chức năng riêng biệt cho từng hệ thống

Trong Bảng 6.2, chúng tôi đặt ra các yêu cầu chung cho cả hai hệ thống lọc nhiễu, từ các khảo sát của chúng tôi, chúng tôi nhận thấy các laptop thông thường có số lượng core trong CPU vào khoảng 4 cores (đối với dòng Intel i5) và lượng RAM tập trung nhiều khoảng 4GB, vậy nên chúng tôi chọn đây là hệ thống sẽ được nhắm đến để hiện thực trong luận văn này. Ngoài hai yếu tố về CPU và RAM,

chúng tôi đặt ra một yêu tố về chất lượng âm thanh sau khi lọc đối với cả hai hệ thống lọc nhiễu. Trong Bảng 6.3, chúng tôi đặt ra các yêu cầu riêng cho từng loại hệ thống thông qua các đặc điểm của từng loại dữ liệu. Từ các yêu cầu phi chức năng này, chúng tôi đặt ra các yêu cầu chức năng khác đối với hệ thống được nêu trong Bảng 6.4.

Lọc nhiễu tĩnh	Lọc nhiễu động
1 Hỗ trợ đọc âm thanh từ file	Hỗ trợ chế độ lọc và không lọc nhiễu
2 Hỗ trợ ghi âm từ microphone	
3 Hỗ trợ xuất âm thanh đã lọc	
4 Hỗ trợ hiển thị âm thanh	
5 Cho phép điều khiển hệ thống lọc nhiễu động	

Bảng 6.4: Yêu cầu chức năng đối với hệ thống

6.2 Hệ thống lọc nhiễu tĩnh

Trước hết chúng tôi sẽ nêu một số usecases của hệ thống lọc nhiễu tĩnh. Hệ thống lọc nhiễu tĩnh là một hệ thống được sử dụng thông qua giao diện của ứng dụng chính. Do vậy các usecases mà chúng tôi nêu lên ở đây sẽ ảnh hưởng trực tiếp tới việc thiết kế cũng như hiện thực ứng dụng chính này. Các usecases chính trong hệ thống lọc nhiễu tĩnh bao gồm Usecase 6.5, Usecase 6.6, Usecase 6.8 và Usecase 6.9.

Tên usecase	Load file âm thanh
Người thực hiện	Người dùng
Mô tả	Người dùng load file âm thanh cần lọc vào hệ thống
Tiền điều kiện	Không đang ghi âm hay đang load đoạn âm thanh khác
Hậu điều kiện	File được load lên và hiển thị ra màn hình
Luồng chạy	<ol style="list-style-type: none"> 1. Ở giao diện chính, người dùng nhấn nút “Import” 2. Cửa sổ mở file hiện lên, người dùng chọn 1 file cần load 3. Task load file được tạo ra và thực thi 4. Hiển thị âm thanh được load ra màn hình

UseCase 6.5: UseCase load file âm thanh của hệ thống lọc nhiễu tĩnh

Được thiết kế theo cơ chế bất đồng bộ, hệ thống lọc nhiễu động được tích hợp vào ứng dụng chính của chúng tôi như một chức năng trong đó. Một trong các

Tên usecase	Ghi âm
Người thực hiện	Người dùng
Mô tả	Người dùng sử dụng hệ thống để ghi âm
Tiền điều kiện	Không đang ghi âm hay đang load đoạn âm thanh khác, luôn có ít nhất một microphone
Hậu điều kiện	Âm thanh được ghi và hiển thị ra màn hình
Luồng chạy	<ol style="list-style-type: none"> 1. Ở giao diện chính, người dùng nhấn nút “Recording” 2. Cửa sổ ghi âm hiện lên, người dùng tiến hành ghi âm 3. Hiển thị âm thanh được load ra màn hình

Usecase 6.6: Usecase ghi âm của hệ thống lọc nhiễu tĩnh

Tên usecase	Xuất file âm thanh
Người thực hiện	Người dùng
Mô tả	Người dùng xuất âm thanh đã được lọc ra file
Tiền điều kiện	Không có file nào khác đang được xuất
Hậu điều kiện	File được xuất ra bộ nhớ
Luồng chạy	<ol style="list-style-type: none"> 1. Ở giao diện chính, người dùng nhấn nút “Export” 2. Cửa sổ lưu file hiện lên, người dùng nhập tên file cần lưu 3. Task lưu file được tạo ra và thực thi 4. Hiển thị thông báo trạng thái của task khi hoàn thành

Usecase 6.7: Usecase xuất file âm thanh của hệ thống lọc nhiễu tĩnh

thành phần cơ bản nhất của một hệ thống bất đồng bộ là *Luồng (thread)* và *Hệ thống quản lý trạng thái*. Các luồng này được chúng tôi thiết kế đặc trưng cho từng loại tác vụ (task) và các tác vụ này đặc trưng cho từng loại nhiệm vụ tương ứng với các biến môi trường, các tiền điều kiện khác nhau tùy thuộc vào đối tượng được gắn với tác vụ đó. Một quy trình chạy của tác vụ được chúng tôi định nghĩa gồm ba bước:

1. *Acquire*: Trong bước này, tác vụ của chúng tôi sẽ khóa các tương tác của người dùng đối với ứng dụng lại đồng thời cũng thiết lập các môi trường cần thiết để tác vụ có thể được thực thi. Ở bước này, các biến môi trường được giả định là không thay đổi trong quá trình thực thi và sẽ chỉ được sao chép dữ liệu sang tác vụ khi quá trình chính bắt đầu thực hiện. Tác vụ sẽ được chuyển đổi trạng

Tên usecase	Lọc nhiễu
Người thực hiện	Người dùng
Mô tả	Người dùng lọc nhiễu âm thanh cần lọc
Tiền điều kiện	Không có file nào đang được lọc
Hậu điều kiện	Âm thanh được lọc và hiển thị ra màn hình
Luồng chạy	<ol style="list-style-type: none"> 1. Ở giao diện chính, người dùng nhấn nút “Lọc nhiễu” 2. Task lọc nhiễu được tạo ra và thực thi 3. Hiển thị âm thanh đã được lọc nhiễu ra màn hình

Usecase 6.8: Usecase lọc nhiễu của hệ thống lọc nhiễu tĩnh

Tên usecase	Điều khiển hệ thống lọc nhiễu động
Người thực hiện	Người dùng
Mô tả	Người dùng điều khiển hệ thống lọc nhiễu động lọc hay không lọc âm thanh
Tiền điều kiện	Hệ thống lọc nhiễu động đang hoạt động
Hậu điều kiện	Không
Luồng chạy	<ol style="list-style-type: none"> 1. Ở giao diện chính, người dùng nhấp vào “Microphone” 2. Ứng dụng gọi xuống hệ thống lọc nhiễu động và thay đổi biến điều khiển bộ lọc

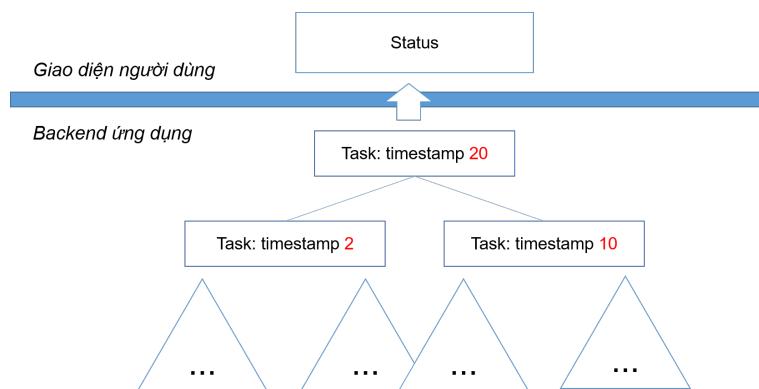
Usecase 6.9: Usecase điều khiển hệ thống lọc nhiễu động của hệ thống lọc nhiễu tĩnh

thái từ *READY* sang *ACQUIRE*.

2. *Functional:* Sau khi đã thực hiện bước *Acquire*, các thành phần tương tác với người dùng đều đã được khóa, các biến môi trường cũng đã được thiết lập cho việc truyền tải dữ liệu. Trạng thái lúc này của tác vụ được chuyển từ *ACQUIRE* sang *PROCESSING*. Tùy vào từng loại tác vụ, từng loại nhiệm vụ, bước này sẽ có thể được định nghĩa với các luồng thực thi khác nhau. Tất cả các kết quả, sao chép dữ liệu vào và ra khỏi tác vụ để vào biến môi trường đều được thực thi ở đây. Nếu một lỗi nào đó xuất hiện ở bước này, tác vụ sẽ lập tức chuyển đổi trạng thái sang *TERMINATE* và thực thi mở khóa các phần tương tác đã bị khóa ở bước *Acquire*.
3. *Release:* Sau khi thực hiện bước *Functional*, trạng thái tác vụ có thể là *PROCESSING* hoặc *TERMINATE* và bắt đầu tiến hành trả lại quyền tương

tác đối với ứng dụng cho người dùng. Lúc này trạng thái tác vụ được chuyển sang *EXIT* nếu trạng thái hiện tại đang là *PROCESSING*. Các mã trạng thái cũng được trả về cho bộ quản lý, các tương tác cũng được mở lại và ứng dụng bắt đầu tương tác lại với người dùng.

Để hệ thống này có thể hoạt động hiệu quả cũng như cho phép người dùng quản lý được các tác vụ đang được thực thi bên dưới ứng dụng, việc thiết kế một bộ quản lý trạng thái là một điều cần thiết. Đối với mỗi tác vụ, các mã thời gian (timestamp) được lấy theo thời gian mà các tác vụ này được hệ thống tạo ra. Bằng cách sử dụng các mã này kết hợp với việc báo cáo trạng thái trong từng bước của mỗi tác vụ, chúng tôi thiết kế ra được bộ quản lý trạng thái có cấu tạo như Hình 6.1.



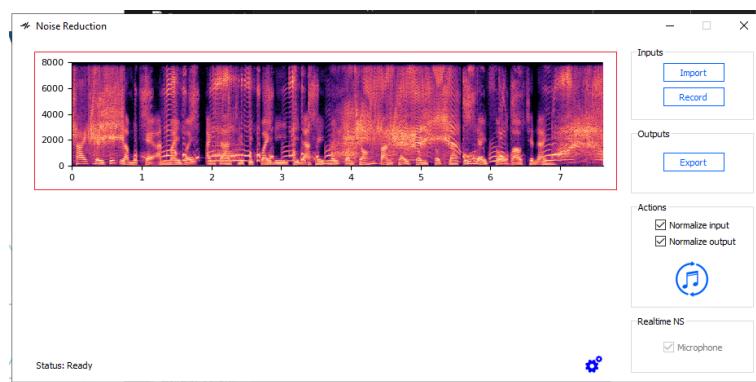
Hình 6.1: Kiến trúc của bộ quản lý trạng thái

Cấu tạo của bộ quản lý trạng thái này khá đơn giản, các tác vụ được sắp xếp sao cho việc lấy tác vụ có mã thời gian lớn nhất (bằng cách so sánh các giá trị của mã này như trong Hình 6.1). Trạng thái của tác vụ này sau đó sẽ được báo cáo về cho giao diện và hiển thị lại cho người dùng. Việc sắp xếp các tác vụ được chúng tôi sử dụng max heap với lý do đây là một cách thuận tiện để có thể lấy phần tử lớn nhất chỉ trong thời gian $O(1)$ và thuận tiện trong cách quản lý các tác vụ mới được thêm vào. Nhưng về bản chất ta có thể sử dụng nhiều các cấu trúc dữ liệu khác để có thể thỏa mãn được điều này như BTree, cây nhị phân, hoặc bằng cách sắp xếp mảng này theo thứ tự từ bé đến lớn.

Với các tác vụ, luồng và bộ quản lý trạng thái, bây giờ tất cả các chức năng của ứng dụng chúng tôi đều có thể được thiết kế như biến thể của các thành phần

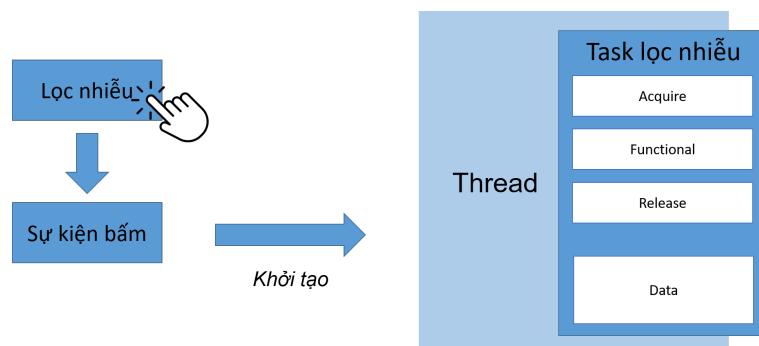
trên. Để thể hiện rõ điều này, ta sẽ lấy một ví dụ về tác vụ lọc nhiễu của ứng dụng chúng tôi. Tác vụ lọc nhiễu của chúng tôi có thể bắt đầu được thực hiện sau khi dữ liệu về file âm thanh cần được lọc nhiễu đã được tải lên ứng dụng hoặc một đoạn ghi âm đã được người dùng sử dụng ứng dụng ghi âm vào, tác vụ lọc nhiễu sẽ được bắt đầu như sau:

1. *Trạng thái ban đầu:* Ở trạng thái ban đầu, tác vụ lọc nhiễu đã được khởi tạo, đăng kí với bộ quản lý trạng thái và đang ở trạng thái ban đầu *INIT*, dữ liệu đã được tải lên và được hiển thị ở giao diện người dùng.



Hình 6.2: Trạng thái ban đầu của tác vụ lọc nhiễu (dữ liệu được tải lên thể hiện dưới dạng spectrogram trong khung màu đỏ)

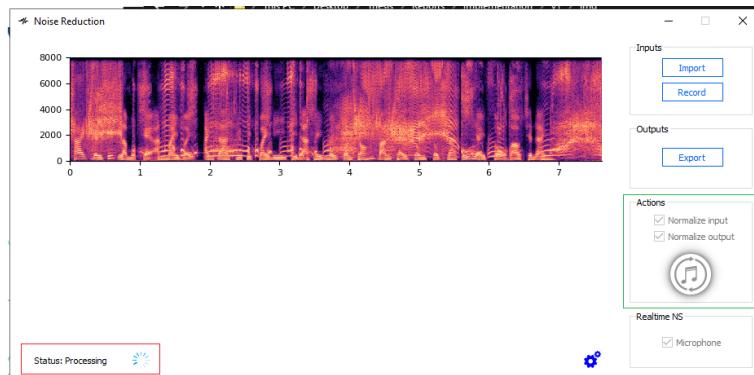
2. *Sự kiện lọc nhiễu được khởi tạo:* Sau khi đã khởi tạo tác vụ lọc nhiễu, một luồng ứng với tác vụ này được tạo ra (chúng tôi gọi luồng này là thread context) để bắt đầu việc thực thi cho tác vụ lọc nhiễu này, trạng thái của tác vụ lúc này được chuyển thành *READY*. Hình 6.3 thể hiện việc kích hoạt sự kiện người dùng bấm nút lọc nhiễu cho đến khi thread context được tạo ra và bắt đầu thực thi.



Hình 6.3: Chuỗi các hành động được tạo ra bởi sự kiện lọc nhiễu

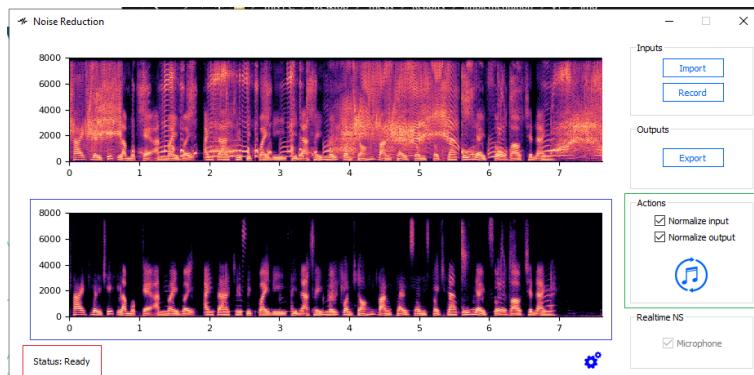
3. *Thực thi tác vụ:* Tác vụ thực thi theo thứ tự *Acquire*, *Functional* và *Release*.

Trạng thái tác vụ chuyển đổi tương ứng theo từng bước thực hiện tác vụ, hình ảnh tác vụ đang ở trạng thái *PROCESSING* được thể hiện như Hình 6.4.



Hình 6.4: Tác vụ đang trong quá trình xử lý, phần bị khóa (khung màu xanh) khóa các tương tác giữa ứng dụng với người dùng lại. Trạng thái (khung màu đỏ) hiển thị về cho người dùng thể hiện trạng thái của tác vụ mới nhất (tác vụ lọc nhiễu đang thực thi)

4. *Kết thúc tác vụ:* Sau khi thực thi xong, các phần bị khóa ở bước *Acquire* được cho phép tương tác tiếp tục với người dùng, kết quả lọc nhiễu được hiển thị, tác vụ lọc nhiễu lúc này được hủy đăng ký với bộ quản lý trạng thái và trạng thái được hiển thị lúc này là tác vụ có mã thời gian lớn nhất tiếp theo vẫn đang còn được thực thi.



Hình 6.5: Tác vụ hoàn thành, kết quả lọc (khung màu xanh dương) được hiển thị ra cho người dùng. Trạng thái (khung màu đỏ) được cập nhật và tác vụ hủy đăng ký với bộ quản lý. Các vùng bị khóa (khung màu xanh lá) được phép tiếp tục tương tác với người dùng

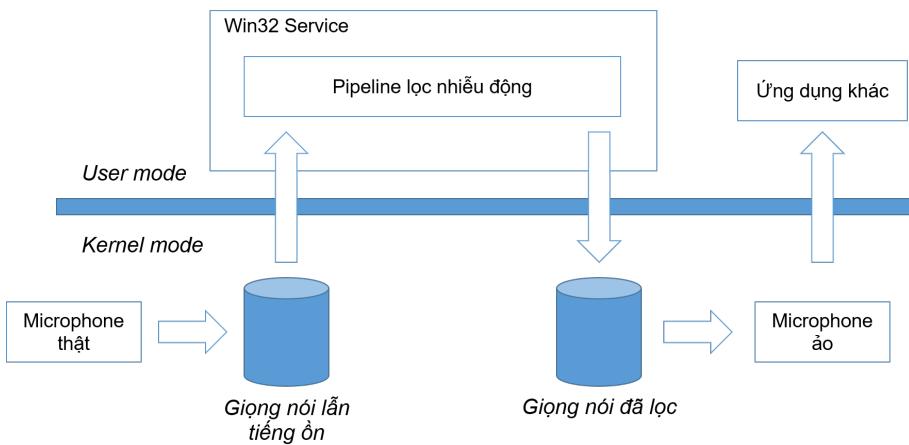
6.3 Hệ thống lọc nhiễu động

Hệ thống lọc nhiễu động chỉ được sử dụng vào một mục đích là di chuyển dữ liệu từ microphone thật (âm thanh có lỗn nhiễu) để lọc trong thời gian thực và trả lại về cho microphone ảo (âm thanh đã lọc). Do đó, usecase của hệ thống này cũng ít phức tạp hơn, chú trọng vào việc lọc nhiễu trực tiếp từ microphone thật để trả về cho microphone ảo và cũng vì vậy mà ít tương tác hơn với người dùng. Usecase chi tiết của hệ thống lọc nhiễu động bao gồm Usecase 6.10.

Tên usecase	Thay đổi bộ lọc
Người thực hiện	Người dùng (qua giao diện chính)
Mô tả	Thông qua giao diện chính, người dùng thay đổi việc có sử dụng bộ lọc nhiễu hay không
Tiền điều kiện	Hệ thống lọc nhiễu động đang hoạt động
Hậu điều kiện	Bộ lọc được thay đổi
Luồng chạy	<ol style="list-style-type: none"> 1. Ở giao diện chính, người dùng nhập vào “Microphone” 2. Ứng dụng gọi xuống hệ thống lọc nhiễu động và thay đổi biến điều khiển bộ lọc 3. Biến điều khiển thay đổi, bộ lọc trong worker của hệ thống lọc nhiễu động thay đổi

UseCase 6.10: UseCase chuyển đổi bộ lọc của hệ thống lọc nhiễu động

Khác với thiết kế của ứng dụng và bộ lọc nhiễu tĩnh, bộ lọc nhiễu động được thiết kế như một ứng dụng độc lập hoàn toàn với ứng dụng chính, sử dụng các cơ chế liên lạc thông qua hệ điều hành để liên lạc giữa hai ứng dụng này. Hình 6.6 thể hiện luồng dữ liệu chạy trong ứng dụng lọc nhiễu động. Dữ liệu về giọng nói được microphone thật của máy tính ghi nhận, với giả định rằng người dùng đang ở trong môi trường nhiều tiếng ồn nên giọng nói được ghi nhận ở đây sẽ là tập hợp của cả giọng nói lẫn với tiếng ồn cần phải loại bỏ. Sau đó các dữ liệu giọng nói này được chuyển từ kernel mode của hệ điều hành lên cho đường ống (pipeline) lọc nhiễu động của chúng tôi. Sau khi đã đi qua đường ống lọc nhiễu này, giọng nói ban đầu có lẫn tiếng ồn bây giờ đã trở thành giọng nói sạch và được đưa trở lại kernel mode cho hệ điều hành thông qua microphone ảo từ đó các ứng dụng khác (kể cả ứng dụng của chúng tôi) sẽ sử dụng microphone ảo này để giao tiếp thay cho microphone thật ban đầu.



Hình 6.6: Luồng di chuyển của dữ liệu trong ứng dụng lọc nhiễu động

Dễ thấy thông qua Hình 6.6, hệ thống lọc nhiễu động này được chia ra làm ba phần: *Dường ống lọc nhiễu động*, *Win32 Service* và *Microphone ảo*. Win32 Service là tên gọi cho một tập hợp các dịch vụ chạy ngầm trên nền tảng hệ điều hành Windows. Các dịch vụ này được phát triển từ khá sớm (từ thời của Windows NT) và khá tương tự như các dịch vụ được sử dụng trong các hệ điều hành khác như UNIX và OSX. Các dịch vụ này bản thân chúng vẫn là một ứng dụng hoàn chỉnh, tuy không có giao diện nhưng vẫn phải hỗ trợ các chức năng của một dịch vụ. Các dịch vụ này được tạo ra để phục vụ cho khá nhiều mục đích khác nhau như hỗ trợ các web server, chạy các cơ sở dữ liệu hay các phần mềm bảo mật liên quan tới người sử dụng. Nhưng điều quan trọng ở đây là các dịch vụ này có thể truy xuất vào các dữ liệu được hệ điều hành bảo vệ như hệ thống lọc nhiễu của chúng tôi đang thực hiện. Các dữ liệu khi đi từ user mode sang kernel mode thường yêu cầu người sử dụng phải cấp quyền administrator cho ứng dụng (điều này được thể hiện ở việc khi mở ứng dụng điều khiển bật tắt dịch vụ của chúng tôi người dùng vẫn phải cấp quyền administrator). Nhưng đối với dịch vụ, tài khoản được sử dụng để kích hoạt dịch vụ mặc định là LOCAL ACCOUNT, tài khoản này cho phép dịch vụ truy xuất vào vùng được hệ điều hành bảo vệ mà người dùng không cần thiết phải trực tiếp cấp quyền cho dịch vụ.

Các dịch vụ chịu chung một sự quản lý của hệ điều hành thông qua một bộ quản lý gọi là **Service Control Manager (SCM)**. Bộ quản lý này có thể được xem là một built-in web server của hệ điều hành cho phép các ứng dụng khác truy xuất trạng thái, điều khiển và gửi các mã lệnh tới cho dịch vụ. Trong hệ thống của

chúng tôi, các mã lệnh được chúng tôi sử dụng bao gồm:

1. *SERVICE_CONTROL_SWITCH_NOFILTER* (0xFF): Tắt đi đường ống lọc nhiễu để dữ liệu trực tiếp đi từ microphone thật sang microphone ảo.
2. *SERVICE_CONTROL_SWITCH_PIPELINE* (0xFE): Bật lại đường ống lọc nhiễu trong dịch vụ.

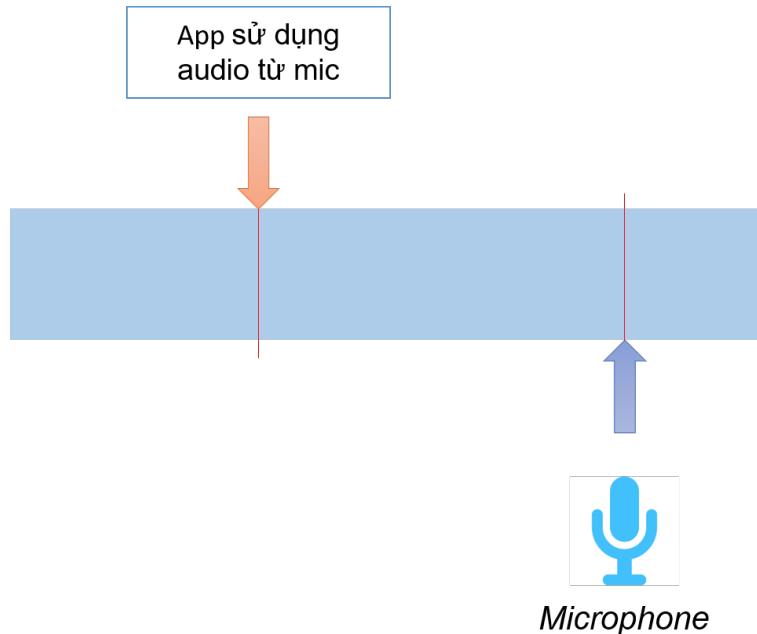
Bên dưới dịch vụ, khi nhận được các mã này, biến điều khiển sẽ được bật và tắt tương ứng với mã lệnh nhận được từ đó điều khiển việc chuyển dữ liệu qua đường ống lọc nhiễu. Các mã được dịch vụ nhận được từ SCM là một số nguyên dương bốn bytes (trong Windows, thường được gọi là DWORD), phạm vi và các mã lệnh cố định được định nghĩa như trong Bảng 6.11.

MACRO	DWORD
SERVICE_CONTROL_STOP	0x01
SERVICE_CONTROL_PAUSE	0x02
SERVICE_CONTROL_CONTINUE	0x03
SERVICE_CONTROL_INTERROGATE	0x04
SERVICE_CONTROL_PARAMCHANGE	0x06
SERVICE_CONTROL_NETBINDADD	0x07
SERVICE_CONTROL_NETBINDREMOVE	0x08
SERVICE_CONTROL_NETBINDEENABLE	0x09
SERVICE_CONTROL_NETBINDDISABLE	0x0A
Định nghĩa tự do	0x80 - 0xFF

Bảng 6.11: Các mã lệnh cố định và phạm vi định nghĩa bổ sung của mã điều khiển dịch vụ

Microphone ở khía cạnh phần mềm có thể được xem như một hàng đợi xoay vòng với dữ liệu cố định (kích thước này phụ thuộc vào tần số lấy mẫu và kích thước dữ liệu ứng dụng yêu cầu). Hình 6.7 thể hiện cho cấu tạo của một microphone trong hệ điều hành Windows. Dữ liệu được microphone thật ghi nhận và sau đó ghi xuống hàng đợi dưới dạng các số nguyên kiểu “long” bốn bytes. Dữ liệu được viết liên tục từ đầu hàng đợi cho tới khi write head này đạt tới cuối của hàng đợi, sau đó lại vòng ngược lên đầu và một vòng ghi như vậy lại tiếp tục. Mặt khác, khi ứng dụng đọc dữ liệu được ghi xuống hàng đợi này cũng thông qua read head và cũng tương tự như write head, khi ứng dụng đọc và read head đạt tới cuối của hàng đợi, read head sẽ tự động chuyển lên đầu hàng đợi và tiếp tục đọc tiếp các dữ liệu tiếp theo. Ở đây sự ràng buộc giữa write head và read head giống như đầu

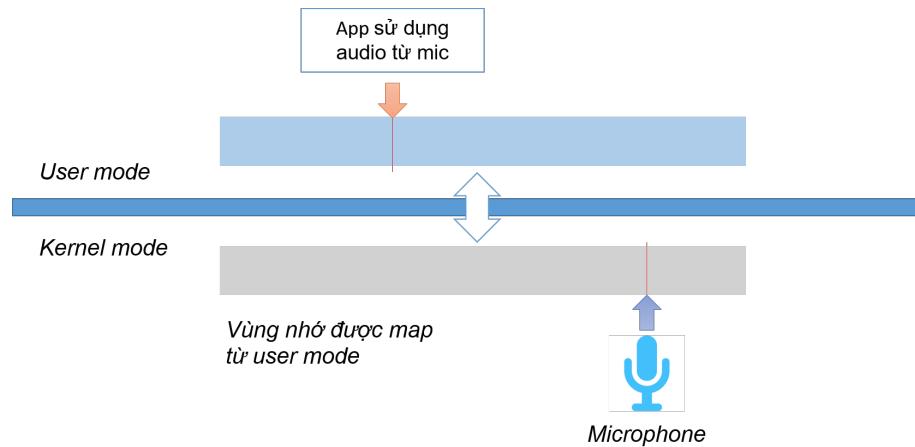
và cuối của một hàng đợi, read head sẽ phải luôn nằm trước write head, do vậy điều này sẽ dẫn tới độ trễ xuất hiện trong bản thân sự chênh lệch giữa read head và write head này.



Hình 6.7: Cấu tạo của microphone (phần mềm), microphone thật truyền dữ liệu vào hàng đợi (hình màu xanh dương) này thông qua write head (mũi tên màu xanh) và ứng dụng đọc dữ liệu được ghi vào thông qua read head (mũi tên cam)

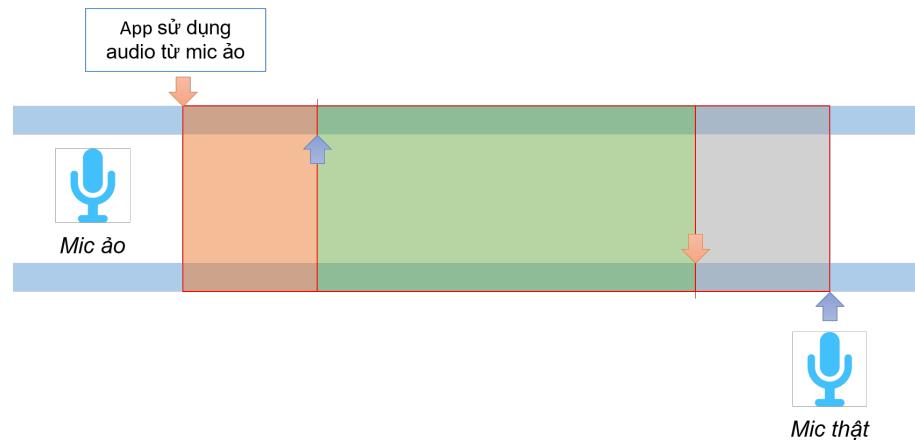
Nhưng việc đọc và ghi giữa microphone và ứng dụng không thực sự xảy ra trên cùng một vùng nhớ. Điều này bị ràng buộc bởi hệ điều hành, vùng nhớ được cấp phát bởi kernel mode sẽ không được phép truy xuất từ user mode và ngược lại. Sự ràng buộc này xảy ra khiến cho dữ liệu trước khi thực sự tới được vùng nhớ chung của microphone phải đi qua nhiều tầng lớp chuyển đổi và kiểm tra dữ liệu khiến cho việc ghi và đọc dữ liệu trở nên rất chậm. Với các công nghệ mới hơn, đặc biệt là với cơ chế truy xuất vùng nhớ trực tiếp (Direct Memory Access - DMA) về dữ liệu âm thanh [29], nút thắt ở việc truyền dữ liệu từ kernel mode sang user mode được giải quyết.

Việc truy xuất trực tiếp làm cho phép việc truyền dữ liệu xảy ra trực tiếp giữa người sử dụng máy tính và phần cứng giúp giảm thiểu độ trễ trong quá trình truyền tải dữ liệu âm thanh. Hình 6.8 thể hiện sự truyền tải dữ liệu từ microphone sang cho ứng dụng ở tầng kernel mode thông qua cơ chế memory mapping giữa



Hình 6.8: Cấu tạo của microphone (phần mềm), thông qua cơ chế truy xuất vùng nhớ trực tiếp (DMA) cho phép giảm thiểu thời gian truyền tải dữ liệu âm thanh với độ trễ thấp (nguồn [29])

hai mode của hệ điều hành. Vùng nhớ ban đầu được cấp phát tại user mode dưới dạng một dãy các đoạn nhớ (được khởi tạo trong quá trình kích hoạt thiết bị), các vùng nhớ này sau đó thông qua cơ chế mapping của hệ điều hành để chuyển đổi địa chỉ về vùng nhớ của kernel mode và truy xuất như một vùng nhớ liên tục được cấp phát bình thường.



Hình 6.9: Tổng độ trễ xảy ra trong hệ thống lọc nhiễu động, tổng độ trễ bao gồm độ trễ của microphone thật (vùng màu xám), độ trễ của đường ống lọc nhiễu (vùng màu xanh lá) và độ trễ của microphone ảo (vùng màu cam)

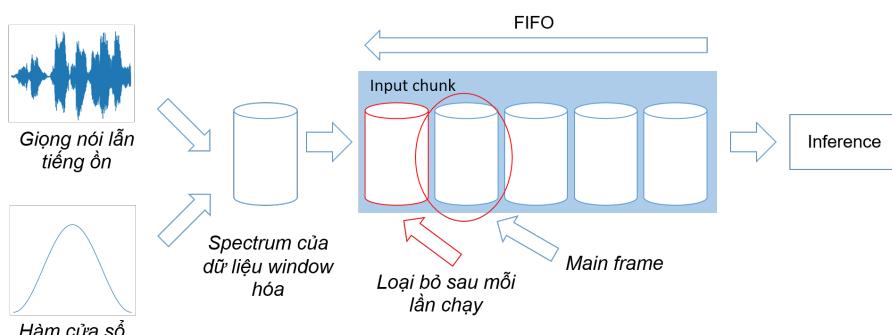
Với cách hoạt động như vậy, ta cần phải ước lượng độ trễ có thể giữa các thành phần trong hệ thống. Hình 6.9 thể hiện toàn bộ các độ trễ có thể xảy ra từ khi dữ liệu được microphone thật ghi vào hàng đợi của nó cho tới khi ứng dụng đọc

được dữ liệu sau khi đã được lọc. Tổng thời gian đi qua hệ thống lọc nhiễu động từ lúc dữ liệu được ghi nhận ở microphone thật cho tới khi ứng dụng đọc được dữ liệu đã được lọc nhiễu tương ứng bao gồm: *độ trễ của microphone thật*, *độ trễ của đường ống lọc nhiễu* và *độ trễ của microphone ảo*. Vì độ trễ của đường ống lọc nhiễu chiếm phần lớn độ trễ của toàn bộ hệ thống nên trong phần kết quả, chúng tôi sẽ chỉ đo đặc các độ trễ của đường ống lọc nhiễu và xem nó là xấp xỉ cho toàn bộ hệ thống lọc nhiễu động này.

Một phần quan trọng nhất trong hệ thống của chúng tôi chính là đường ống lọc nhiễu động. Đường ống của chúng tôi được chia làm ba phần:

1. *Input*: Dữ liệu theo miền thời gian được tiền xử lý và biến đổi về miền tần số thời gian thành đầu vào của mô hình.
2. *Inference*: Lưu trữ trạng thái trước, thiết lập và thực thi mô hình chính.
3. *Output*: Dữ liệu sau khi nhân với mask số thực được dự đoán bởi mô hình được biến đổi ngược về miền thời gian từ đó thực hiện tổng hợp lại và trả về kết quả.

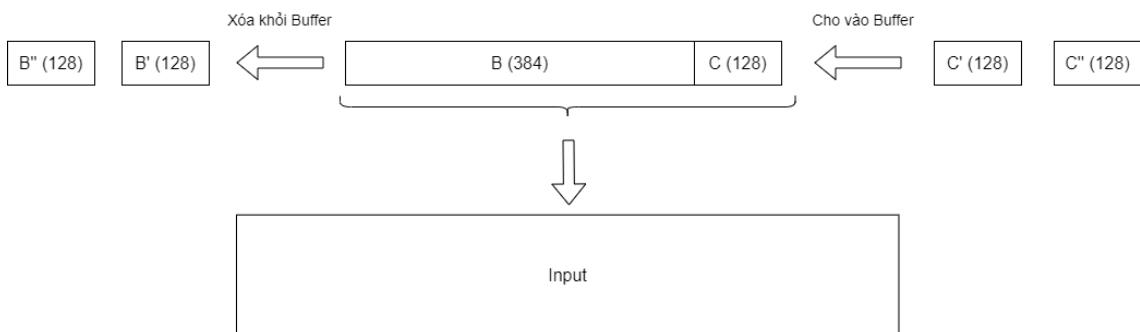
Dữ liệu từ microphone thật trả về cho mô hình của ta dưới dạng waveform trong miền thời gian. Để có thể thực thi được mô hình, ta cần dữ liệu đầu vào cần phải được biến đổi về spectrogram ở miền tần số thời gian. Để thực hiện điều này, ta thực hiện kiến trúc của Input như trong Hình 6.10.



Hình 6.10: Luồng di chuyển của dữ liệu trong phần Input

Mục tiêu của phần Input chính là giả lập lại biến đổi Fourier thời gian ngắn được đề cập ở Chương 2 trong công thức (2.1) với dữ liệu đầu vào $f(t)$ là một luồng dữ liệu đầu vào liên tục. Với kích thước cửa sổ cần xét là 512 mẫu (tương

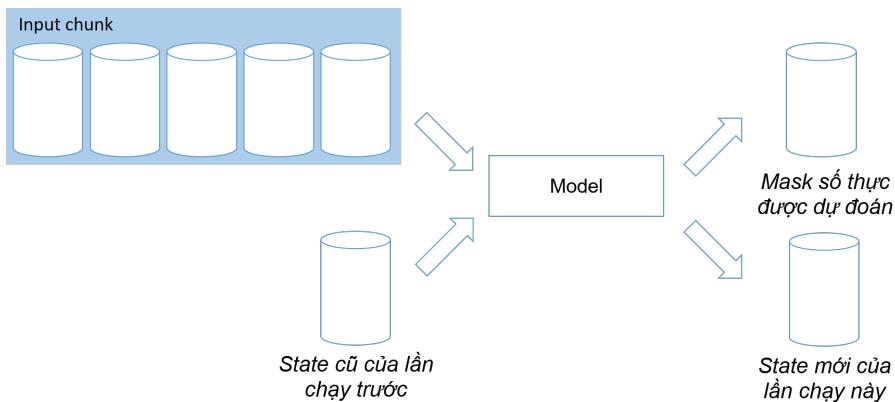
ứng 32ms với tần số lấy mẫu là 16000 mẫu/s) và khoảng cách giữa các cửa sổ này là 128 mẫu (tương ứng với 8ms). Ta thiết lập luồng dữ liệu dưới dạng các đoạn dữ liệu có kích thước đúng bằng khoảng cách giữa các cửa sổ (vì một cửa sổ có thể được chia thành 4 đoạn dữ liệu nhỏ hơn). Sau đó chúng tôi đặt 128 mẫu dữ liệu này vào chung hàng đợi FIFO có kích thước 512 mẫu và loại bỏ đi 128 mẫu ở đầu hàng. Mỗi lần lấy dữ liệu để đưa vào Input, ta sẽ đọc toàn bộ dữ liệu đang có mà không lấy chúng ra khỏi hàng đợi.



Hình 6.11: Hàng đợi đầu vào của Input

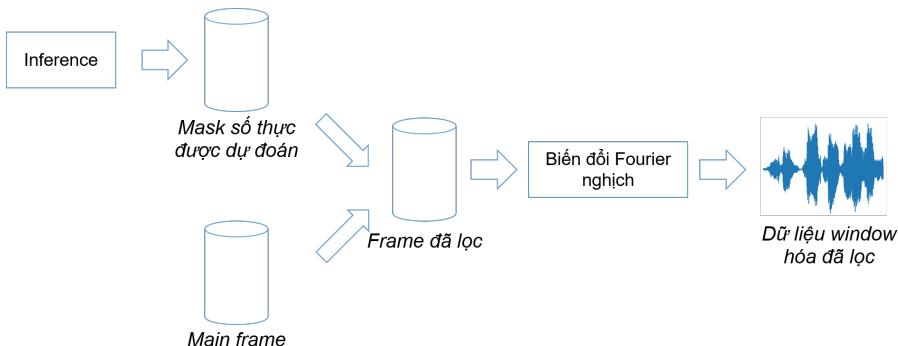
Cứ mỗi lần dữ liệu được đưa vào như vậy, ta thực hiện tính toán biến đổi Fourier trên dữ liệu được đưa vào sau khi nhân với một hàm cửa sổ. Hàm cửa sổ được sử dụng ở phần này là cửa sổ Hann, như đã trình bày trong Mục 2.1. Cửa sổ Hann rất hiệu quả trong việc tách các tần số gần nhau đối với dạng dữ liệu tổng hợp sóng của ta. Sau khi biến đổi Fourier rời rạc trên đoạn dữ liệu đã được cửa sổ (window) hóa này, ta thu được một dãy số phức đại diện cho pha và biên độ của sóng tại vị trí đó trong khoảng thời gian 32ms tương ứng. Cuối cùng để thỏa mãn được dữ liệu đầu vào cho mô hình lọc nhiễu, như đã đề cập ở Tiểu mục 5.3.2, mô hình cần sử dụng năm dãy tần số liên tục nhau để dự đoán ra “mask” số thực cho một dãy tần số nhất định mà chúng tôi gọi là “**Main frame**”. Vì vậy ở đây, sau khi đã thực hiện biến đổi Fourier rời rạc cho dữ liệu được cửa sổ hóa, chúng tôi chia các dãy tần số này lại vào một hàng đợi FIFO có kích thước năm dãy tần số (mỗi dãy tần số có kích thước 257 số phức). Và cứ như vậy, đầu vào của phần tiếp theo Inference sẽ là biên độ của năm dãy tần số được chứa trong hàng đợi này.

Sau khi nhận được đầu vào của mô hình từ Input, phần tiếp theo Inference, mô hình của ta sẽ được thực thi ở đây. Cũng như đã đề cập trong Chương 5, mô hình của ta có cấu tạo từ hai lớp LSTM chồng lên nhau để tạo thành bộ lọc nhiễu



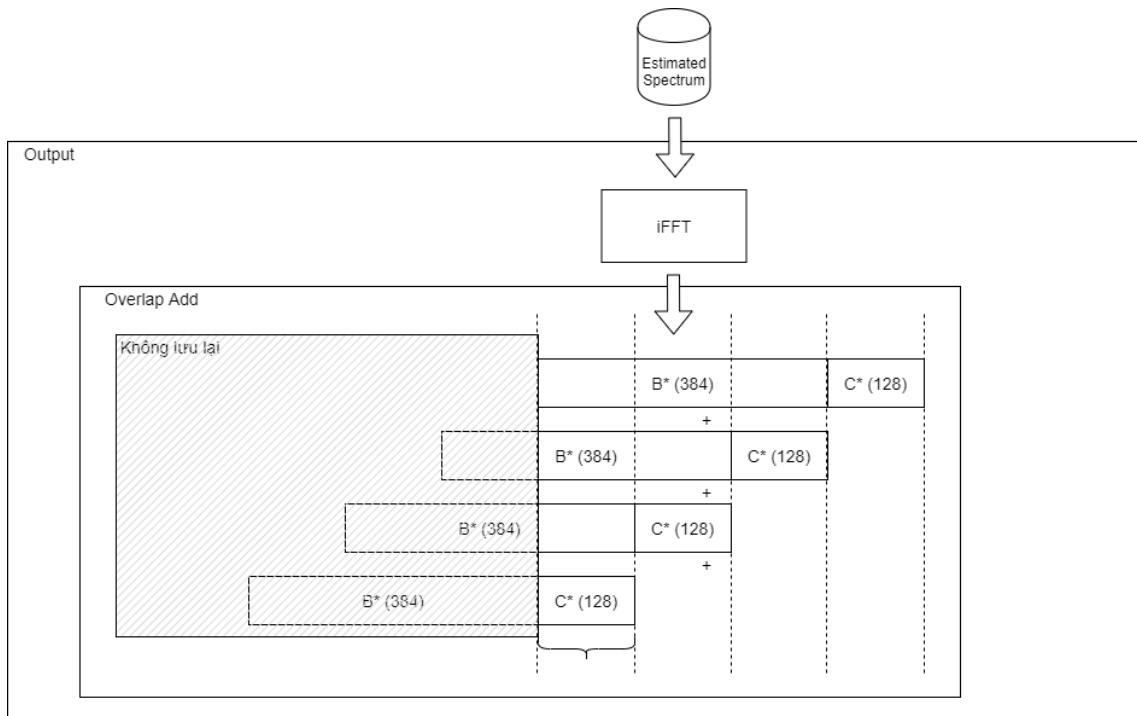
Hình 6.12: Luồng di chuyển của dữ liệu trong phần Inference

chính, vì vậy để lớp LSTM có thể hoạt động ổn định giống như khi kiểm thử mô hình lọc nhiễu trên các dữ liệu tĩnh, ta cần phải lưu trữ lại các trạng thái hoạt động của cả hai LSTM này. Sử dụng các trạng thái và dữ liệu đầu vào từ Input, chúng tôi tính toán được “mask” số thực để sử dụng cho Main frame, nhân chúng lại với nhau ta thu được dãy tần số đã được lọc nhiễu.



Hình 6.13: Luồng di chuyển của dữ liệu trong phần Output

Tới phần Output, ta đã có được dãy tần số đã được lọc nhiễu, và bây giờ ta cần là biến đổi dãy tần số này về miền thời gian và trả về cho người dùng. Để làm điều này, trước hết sẽ biến đổi Fourier nghịch để thu được dữ liệu đã lọc bị cửa sổ hóa. Nhờ vào đặc điểm của cửa sổ và biến đổi Fourier thời gian ngắn đã đề cập ở Tiểu mục 2.1.3, ta sử dụng phương pháp Overlap Add để khử đi việc dữ liệu bị cửa sổ hóa của đầu ra hiện tại. Hình 6.14 thể hiện cho cách chúng tôi thực hiện từ lúc nhận được dãy tần số đã được lọc và phương pháp Overlap Add lên dữ liệu bị cửa sổ hóa.



Hình 6.14: Sơ đồ hoạt động của phần Output

Do dữ liệu sau khi biến đổi luôn có kích thước là 512 mẫu. Dễ thấy, chỉ với 128 mẫu ở đầu của bước Overlap Add mới thỏa mãn được điều kiện của cửa sổ (tổng tất cả dữ liệu cửa sổ hóa mới trả về kết quả là giá trị bị nhân lên cho một hằng số) được nêu trong Mục 2.1. Kết quả trả về của chúng tôi, vì vậy cũng sẽ bị giới hạn bởi điều này và được cố định ở 128 mẫu đầu tiên trong Overlap Add.

Như vậy, tổng thể hệ thống lọc nhiễu động của ta sẽ bao gồm ba phần chính: *Vỏ bọc của ứng dụng (Win32 Service)*, *Bộ lọc chính (đường ống lọc nhiễu động)* và *Giao tiếp với ứng dụng (microphone ảo)*. Kết hợp các thành phần lại chúng tôi thu được một hệ thống lọc nhiễu hoàn chỉnh được nêu trong Hình 6.6. Hình 6.15 là tổng thể về lưu đồ hoạt động của đường ống lọc nhiễu động của chúng tôi.

Các công nghệ được sử dụng phần này bao gồm:

- 1) *Win32 Service*: Mẫu dịch vụ¹ viết bằng C++. Tuy đã có sẵn mẫu nhưng các tài liệu liên quan² chỉ hướng dẫn và chú thích cho C#. Đây là khó khăn khi tiếp cận nguồn tài liệu cho phần này của chúng tôi.

¹Template: <https://docs.microsoft.com/en-us/windows/win32/services/the-complete-service-sample>

²<https://docs.microsoft.com/en-us/dotnet/framework/windows-services/>

2) *Dường ống lọc nhiễu:*

- Eigen³ là thư viện dùng cho các phép toán trên vector với tốc độ chạy cao, chúng tôi sử dụng thư viện này để tính toán các kết quả trung gian trong đường ống.
- FFTW⁴ là thư viện tính toán biến đổi Fourier rời rạc và là một trong những thư viện sử dụng phổ biến trong công nghiệp.
- Miniaudio⁵ là thư viện dùng để chúng tôi đọc âm thanh từ microphone.
- Tensorflow Lite⁶ là thư viện dùng để chạy mô hình của chúng tôi ở ứng dụng.

3) *Microphone ảo:*

- WaveRT⁷ [29] là công nghệ⁸ đọc ghi dữ liệu dùng cho âm thanh mới nhất hiện tại của Microsoft, giúp độ trễ của chúng tôi về thời gian di chuyển giữa đường ống ra và vào microphone trở nên rất thấp.
- Driver Implementation [30, 31] là các khái niệm cơ bản về Driver cũng như bên trong hệ thống của Windows, từ đó hiểu được làm như thế nào để chuyển dữ liệu vào và ra khỏi kernel.

Tuy nhiên, đường ống lọc nhiễu này còn tồn tại một số hạn chế mà trong đó có thể kể đến là khả năng thích nghi với môi trường xung quanh của người dùng. Môi trường xung quanh người dùng liên tục thay đổi, một mô hình tốt là mô hình vừa đảm bảo được khả năng thời gian thực mà vẫn phải đảm bảo được chất lượng như lúc ban đầu. Vậy mô hình để xuất làm một ví dụ. Với thiết kế được ống lọc nhiễu động hiện tại, mô hình được huấn luyện là cố định trong tất cả các phiên hoạt động của ứng dụng. Rõ ràng để có thể thích nghi được với môi trường biến đổi liên tục, mô hình của ta sẽ phải liên tục được huấn luyện lại trên một bộ dữ liệu đã được cập nhật. Điều này rất kém hiệu quả. Và cũng là một trong những điểm hạn chế của thiết kế này.

³<https://eigen.tuxfamily.org/>

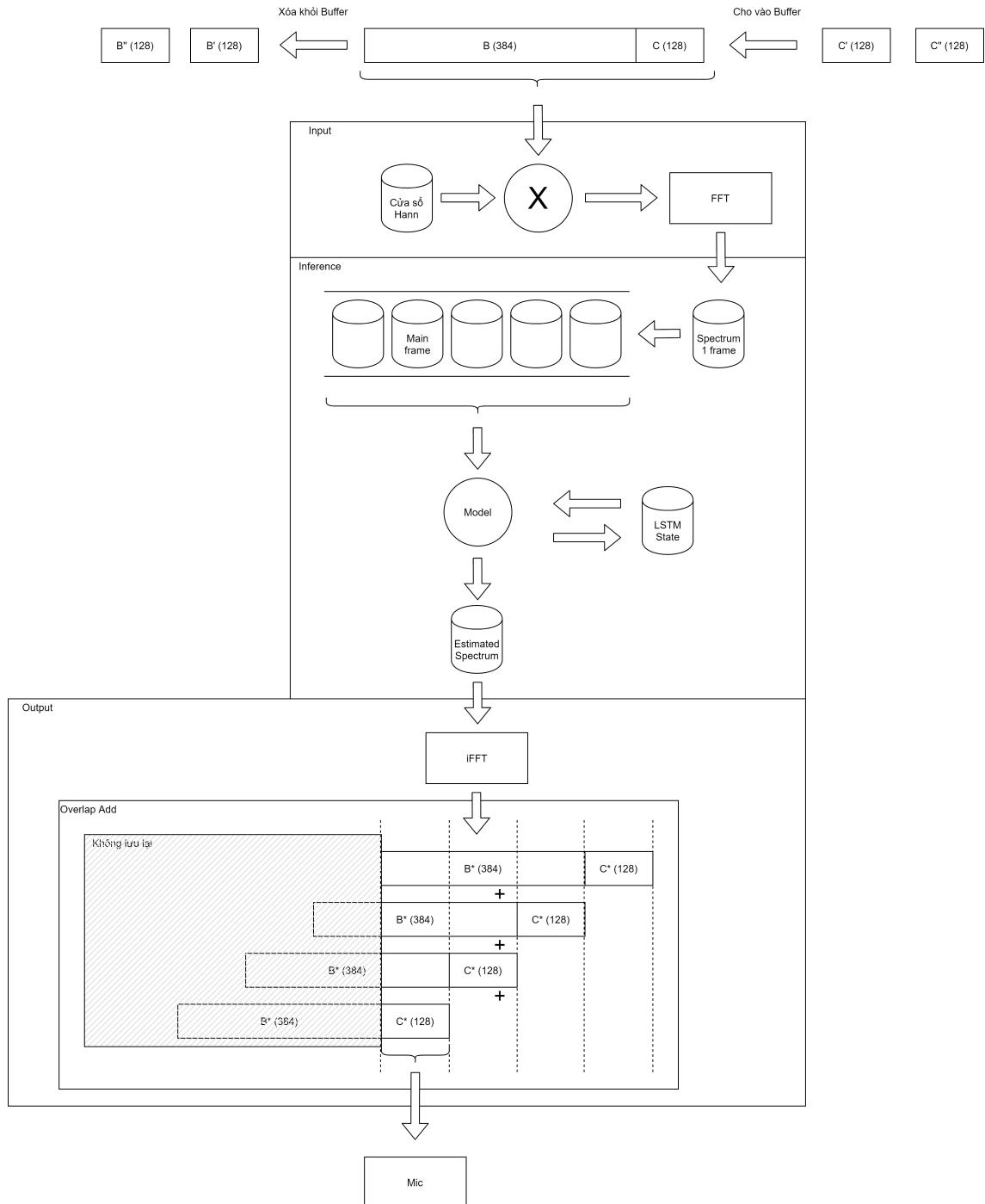
⁴<https://www.fftw.org/>

⁵<https://miniaud.io/>

⁶<https://www.tensorflow.org/lite>

⁷Doc: <https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/wavert-port-driver>

⁸Template: <https://github.com/microsoft/Windows-driver-samples/tree/master/audio/simpleaudiosample>



Hình 6.15: Lưu đồ hoạt động của đường ống lọc nhiễu động

Chương 7

Thực nghiệm và kết quả

Trong chương này, chúng tôi sẽ trình bày về quá trình xử lý dữ liệu, huấn luyện mô hình cũng như một số kết quả mà chúng tôi thu được với mô hình của mình. Các mô hình của chúng tôi sẽ được đem đi thử nghiệm so sánh với một số mô hình đã có và các mô hình được các ứng dụng công nghiệp mã nguồn đóng cũng như mã nguồn mở trên bộ dữ liệu tiếng Việt. Bộ dữ liệu mở rộng đa ngôn ngữ được chúng tôi thử nghiệm để kiểm thử khả năng tổng quát hóa trên nhiều loại ngôn ngữ khác nhau của mô hình.

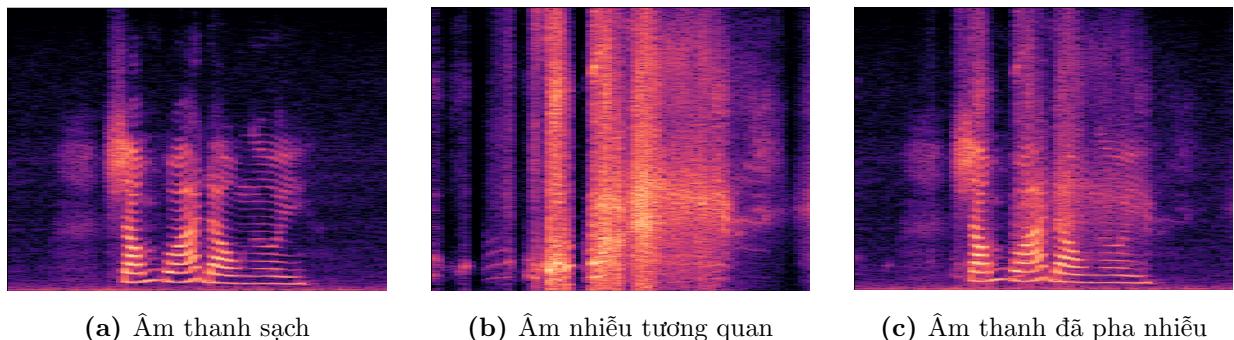
7.1 Chuẩn bị dữ liệu

Để chuẩn bị cho quá trình huấn luyện mô hình, chúng tôi cần phải bắt đầu từ khâu chuẩn bị dữ liệu. Dữ liệu được chúng tôi sử dụng ở trong quá trình huấn luyện là dữ liệu đã được tiền xử lý, quá trình tiền xử lý này sẽ trộn nhiễu với âm sạch theo một tỉ lệ cho trước, tỉ lệ mà chúng tôi đang nói đến là **Signal To Noise Rate (SNR)**. SNR được định nghĩa như sau

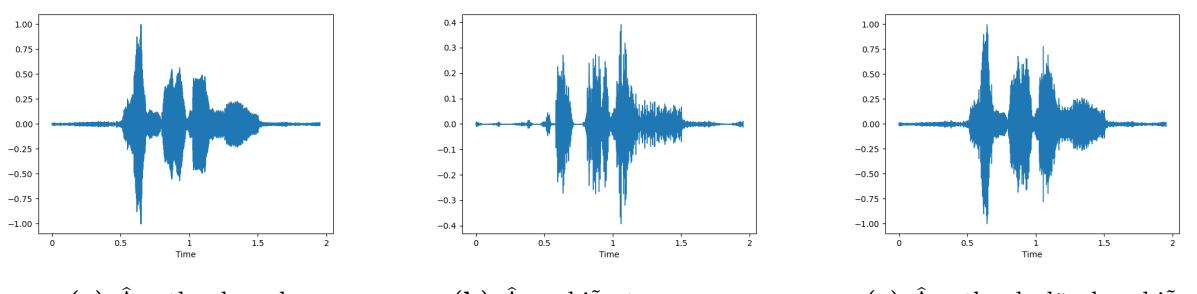
$$\text{SNR}(x, y) = 10 \log_{10} \left(\frac{\|x\|_2^2}{\|y\|_2^2} \right), \quad (7.1)$$

với $x, y \in \mathbb{R}^n$. Bằng công thức ở trên, chúng tôi đã tạo ra hai loại nhiễu trong dữ liệu của mình, đó là **nhiễu tương quan (correlated noise)** và **nhiễu không tương quan (uncorrelated noise)**. Điều khác biệt duy nhất ở hai loại nhiễu này

đó chính là sự biến đổi biên độ của nhiễu có tương quan với giọng nói hay không. Để làm rõ điều này ta lấy một ví dụ. Hình 7.1a và Hình 7.3a bên dưới chính là giọng nói sạch không có nhiễu, ta mong muốn pha một nhiễu vào trong âm thanh này sao cho nhiễu đó có tồn tại tại sự tương quan về mặt biên độ đối với giọng nói hay khi biên độ giọng nói tăng lên, nhiễu cũng sẽ tăng lên. Và ngược lại khi biên độ giọng nói giảm thì biên độ của nhiễu cũng sẽ bị giảm theo. Hai đại lượng này tỉ lệ với nhau nhưng vẫn không làm mất đi bản chất của nhiễu ban đầu. Kết quả của việc pha nhiễu này được thể hiện ở Hình 7.1 và Hình 7.2.

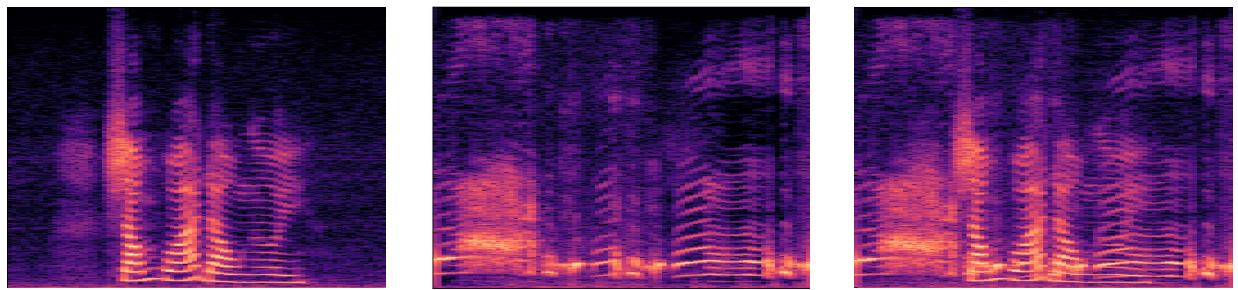


Hình 7.1: Kết quả spectrogram của việc trộn nhiễu tương quan



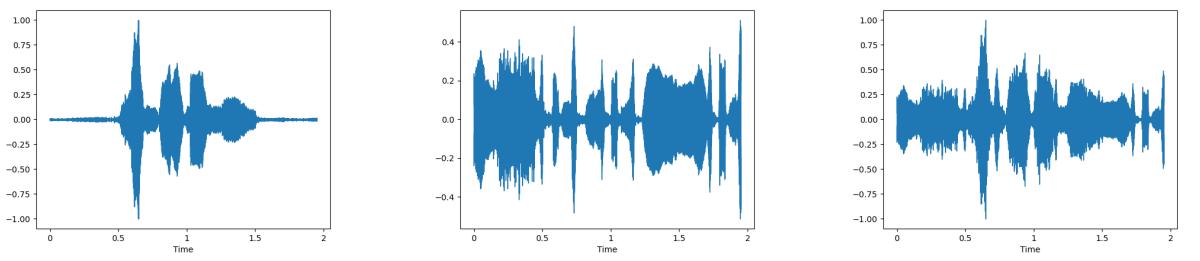
Hình 7.2: Kết quả waveform của việc trộn nhiễu tương quan

Trái với nhiễu tương quan, nhiễu không tương quan không có sự tương đồng như vậy. Nhiều ở trong trường hợp này chỉ được nhân thêm một hằng số sao cho sự giá trị SNR của nhiễu và âm sạch nằm ở một mức mong muốn. Hình 7.3 thể hiện spectrogram nhiễu trong trường hợp không có sự tương quan này. Về mặt toán học, cả hai loại nhiễu đều được tính toán dựa trên chung một chỉ số SNR cho trước và sử dụng công thức (7.1). Ở trong trường hợp nhiễu tương quan, công thức của SNR được thay đổi để phù hợp với sự biến đổi của giọng nói



(a) Âm thanh sạch (b) Âm nhiễu không tương quan (c) Âm thanh đã pha nhiễu

Hình 7.3: Kết quả spectrogram của việc trộn nhiễu không tương quan



(a) Âm thanh sạch (b) Âm nhiễu không tương quan (c) Âm thanh đã pha nhiễu

Hình 7.4: Kết quả waveform của việc trộn nhiễu không tương quan

$$n'_1(t) = n(t) \sqrt{\frac{x(t)^2}{10^{\alpha/10}}}. \quad (7.2)$$

Bằng cách chuẩn hóa lại giá trị nhiễu mong muốn được tính từ $\sqrt{x(t)^2/10^{\alpha/10}}$, chúng tôi nhân với giá trị nhiễu hiện tại $n(t)$, từ đó thu được sự biến đổi của nhiễu theo giọng nói như mong muốn. Tuy cách làm như thế này không đảm bảo được giá trị SNR được tính trong công thức ban đầu là đúng chính xác như SNR kì vọng α , vì trong trường hợp này, nếu ta chỉ đảm bảo được SNR trong công thức tại một thời điểm bất kỳ luôn là giá trị mong đợi α thì điều đó cũng không có nghĩa trong công thức ban đầu cũng như vậy. Nhưng trong nhiễu không tương quan, giá trị SNR tính theo công thức ban đầu hoàn toàn có thể được đảm bảo đúng với giá trị mong đợi. Nhieu không tương quan được định nghĩa như sau

$$n'_2(t) = \frac{n(t)}{\|n\|_2} \frac{\|x\|_2}{\sqrt{10^{\alpha/10}}}. \quad (7.3)$$

Việc SNR được cố định lại thay hiện chính xác trong công thức trên. Ở về $\|x\|_2/\sqrt{10^{\alpha/10}}$, thay vì sử dụng SNR trên từng thời điểm chúng tôi triển khai lại

đúng với chuẩn Euclid của $x(t)$. Từ đó tính ra được chuẩn Euclid của biên độ nhiễu $n(t)$ mới là $\|n'\|_2 = \|x\|_2 / \sqrt{10^{\alpha/10}}$.

Bằng cách nhân mỗi đại lượng tại $x(t)$ lên thêm một giá trị $\|n'\|_2 / \|n\|_2$, chúng tôi đang nhân giá trị chuẩn Euclid của $n(t)$ cho một hằng số để thỏa mãn giá trị tương ứng với SNR được tính toán từ $x(t)$. Từ đó SNR của nhiễu so với giọng nói sạch theo cách pha nhiễu này đảm bảo được SNR được tính ở công thức gốc dùng âm thanh sạch và nhiễu đã xử lý sẽ đúng với giá trị SNR mong đợi α .

Về dữ liệu, trong việc sử dụng để huấn luyện mô hình, dữ liệu sạch của chúng tôi gồm có hai loại, dữ liệu tiếng Anh và dữ liệu tiếng Việt. Về dữ liệu tiếng Anh, theo tìm hiểu của chúng tôi, các công trình nghiên cứu được công bố trong các cuộc thi giảm nhiễu âm thanh được công bố khá nhiều, do vậy chúng tôi đã sử dụng một phần dữ liệu trong các cuộc thi này về để sử dụng như nguồn âm thanh sạch của mình. Bảng 7.1 mô tả một số bộ dữ liệu tiếng Anh mà chúng tôi sử dụng.

Dataset	Kích thước	Mục tiêu	Số loại nhiễu	Giọng nói sạch
DNS [14]	1 TB	Giảm nhiễu	150	11350
Edinburgh Data [17]	5 GB	Giảm nhiễu, Text to Speech	2	28
Edinburgh Data [17]	10 GB	Giảm nhiễu, Text to Speech	2	56
CHIME [15]	115 GB	Speech to Text, tách giọng nói	0	32
DEMAND [16]	7.4 GB	Noise Dataset	18	0

Bảng 7.1: Các tập dữ liệu tiếng Anh được sử dụng trong luận văn

Bên cạnh đó, chúng tôi cũng đã thu thập được một số bộ dữ liệu tiếng Việt. Tuy mục đích của chúng không phải dùng để giảm nhiễu trong âm thanh mà là để dùng cho nhận diện giọng nói sang chữ viết nên chúng tôi phải tự lọc lại dữ liệu. Vì nhiễu đã được pha trước vào dữ liệu và thứ mà chúng tôi cần là một giọng nói sạch, cũng như lựa ra những bộ dữ liệu có chất lượng tốt để dùng cho việc huấn luyện của mình.

Bảng 7.2 dưới đây là mô tả về bộ dữ liệu tiếng Việt mà chúng tôi sẽ sử dụng. Để đảm bảo tính công bằng giữa các mô hình đã có và mô hình của mình, mô hình chúng tôi sẽ chỉ được huấn luyện trên bộ dữ liệu tiếng Anh. Bộ dữ liệu tiếng Việt sẽ được sử dụng như bộ kiểm thử chất lượng của mô hình.

Bộ dữ liệu mà chúng tôi dùng để huấn luyện mô hình sẽ bao gồm các âm sạch được lấy ngẫu nhiên 4 lần để lai tạo với các âm nhiễu khác nhau, tỉ lệ chia nhiễu

Dataset	Kích thước	Mục tiêu	Số file âm thanh
FPT Open Dataset	1.6 GB	Speech to Text	15049
VIVOS [18]	1.4 GB	Speech to Text	12426
VLSP 2019	23 GB	Speech to Text	160000
VIET-TTS	5 GB	Speech to Text, Text to Speech	22884

Bảng 7.2: Các tập dữ liệu tiếng Việt được sử dụng trong luận văn

tương đồng và nhiễu không tương đồng là 3:7, với SNR được lấy ngẫu nhiên trong khoảng $[-6, 20]$.

Tuy nhiên trong quá trình hiện thực mô hình, chúng tôi gặp phải một số vấn đề liên quan tới biên độ của giọng nói đầu vào từ microphone của người dùng. Với giả định khi huấn luyện mô hình rằng microphone của người dùng sẽ luôn nằm ở đúng vị trí của miệng lúc nói, do vậy các âm thanh chúng tôi dùng để huấn luyện luôn được chuẩn hóa về biên độ lớn nhất là 1.0, nhưng thực tế lại không phải vậy.



Hình 7.5: Luôn tồn tại khoảng cách từ miệng người nói tới microphone

Hình 7.5 thể hiện cho việc này, với sự xuất hiện của khoảng cách từ miệng người nói tới microphone như vậy, biên độ của âm thanh lúc này sẽ không thể được đảm bảo như giả định của chúng tôi cũng như khi người dùng thay đổi vị trí của microphone, điều này cũng gây ra sự biến đổi trong biên độ. Nhận thấy điều này, bằng các thực nghiệm, chúng tôi xác định đối với microphone của chúng tôi khoảng hoạt động của biên độ thường nằm trong khoảng $[0.2, 0.6]$ và phải để rất gần (gần như là trước miệng) để biên độ được microphone ghi nhận chạm tới mức 1.0.

Vậy nên để có thể giúp cho mô hình của chúng tôi có thể hoạt động trong các môi trường như vậy, chúng tôi giả lập lại tính huống trên bằng cách thay đổi một

cách ngẫu nhiên trong một giá trị $\delta \in [0.1, 1.0]$. Và giọng nói chúng tôi dùng để huấn luyện mô hình dùng để sẽ được nhân với hệ số δ này.

Phát hiện này của chúng tôi khá trễ, vào giai đoạn mà chúng tôi đang hiện thực ứng dụng để áp dụng mô hình nên các mô hình phiên bản trước Phiên bản 3, sẽ không được kiểm thử trên các âm chưa được chuẩn hóa về biên độ có giá trị là 1.0. Chỉ có phiên bản cuối và các mô hình cạnh tranh sẽ được kiểm thử trên tất cả biên độ.

Sau khi đã chuẩn bị hoàn tất dữ liệu, chúng tôi bắt đầu khâu chuẩn bị dữ liệu kiểm thử. Khác với dữ liệu huấn luyện, dữ liệu kiểm thử chỉ được chúng tôi sử dụng trên tiếng Việt được lấy từ bộ kiểm thử của tập dữ liệu VIVOS [18] thay vì sử dụng tiếng Anh và có thể có nhiều loại nhiễu không tương đồng cùng tồn tại trong cùng một đoạn âm thanh để giả lập lại môi trường xung quanh người dùng. Bộ dữ liệu kiểm thử được chúng tôi chia thành năm mức biên độ A là $\{0.2, 0.4, 0.6, 0.8, 1.0\}$, và tương ứng cho mỗi mức biên độ sẽ có sáu mức SNR là $\{-5, 0, 5, 10, 15, 20\}$. Mỗi cặp giá trị A và SNR này sẽ được tạo một bộ dữ liệu nhỏ có 400 mẫu.

Như vậy với bộ dữ liệu kiểm thử trên, chúng tôi sẽ tiến hành thử nghiệm mô hình trên tất cả 12000 mẫu dữ liệu có số lượng nhiễu được lấy ngẫu nhiên từ một đến bốn loại (có thể trùng) với các mức biên độ và SNR tương ứng. Chúng tôi chạy thử các metrics của mình trên bộ kiểm thử này và thu được kết quả như Bảng 7.3.

Bộ kiểm thử	STOI	NBPESQ	WBPESQ	SIG	BAK
SNR = -5	0.64 (0.11)	1.36 (0.27)	1.10 (0.12)	3.66 (0.34)	0.98 (0.71)
SNR = 0	0.75 (0.09)	1.54 (0.36)	1.16 (0.19)	3.89 (0.27)	1.21 (0.73)
SNR = 5	0.84 (0.08)	1.78 (0.41)	1.29 (0.28)	4.15 (0.20)	1.63 (0.76)
SNR = 10	0.90 (0.07)	2.11 (0.46)	1.54 (0.37)	4.32 (0.15)	2.20 (0.69)
SNR = 15	0.94 (0.05)	2.57 (0.50)	1.97 (0.48)	4.39 (0.15)	2.75 (0.60)
SNR = 20	0.96 (0.04)	3.08 (0.48)	2.51 (0.51)	4.38 (0.19)	3.25 (0.46)

Bảng 7.3: Metric trung bình trên từng bộ kiểm thử tiếng Việt ứng với các SNR khác nhau

Để có thể kiểm thử kĩ hơn nữa khả năng lọc nhiễu của mô hình, chúng tôi tiến hành tạo thêm một bộ kiểm thử khác. Trong bộ trong bộ này, chúng tôi sử dụng giọng nói được lấy từ sách nói ở thư viện mở LibriVox¹ của sáu thứ tiếng khác nhau bao gồm: *tiếng Nhật, tiếng Pháp, tiếng Nga, tiếng Đức, tiếng Bồ Đào Nha* và *tiếng Trung Quốc*. Vì đã xác định việc thay đổi biên độ của âm đầu vào không làm

¹<https://librivox.org/>

ảnh hưởng nhiều tới chất lượng lọc nhiễu của mô hình (through qua các số liệu được trình bày trong phần kết quả), chúng tôi thực hiện lai tạo giọng nói trên cùng với sáu mức SNR. Các âm nhiễu có thể xuất hiện cùng lúc bên trong giọng nói nhưng với biên độ được lấy ngẫu nhiên theo phân phối đều trong khoảng [0.1, 1.0]. Chúng tôi cũng tiến hành chạy kiểm thử các kết quả metrics trên bộ kiểm thử đa ngôn ngữ này và thu được kết quả như trong Bảng 7.8.

7.2 Một số kết quả của mô hình lọc nhiễu

Bằng các phương pháp lai tạo, sử dụng các mô hình và hàm mất mát đã được đề xuất ở trên, chúng tôi đã huấn luyện ra được các mô hình với kết quả khá khả quan. Mô hình được chúng tôi sử dụng chính ở trong ứng dụng của mình là mô hình được đề xuất ở Phiên bản 3 với bộ nén dữ liệu sử dụng tích chập. Đó cũng chính là mô hình mà chúng tôi sẽ đem đi so sánh với mô hình đối thủ (mô hình có điểm metrics cao nhất trong các mô hình được chúng tôi tham khảo). Các metrics được chúng tôi sử dụng bao gồm STOI, PESQ và DNSMOS (chúng tôi chỉ xét SIG và BAK, OVR sẽ không được xét vì không mang nhiều thông tin về chất lượng mà chúng tôi muốn đánh giá).

Đầu tiên để có thể tiến hành so sánh chất lượng của mô hình chúng tôi với một số mô hình đã có được chúng tôi lấy từ các bài báo bao gồm FullSubNet [1], DCCRN [2] và DTLN [3]. Trong số ba mô hình này, chúng tôi sẽ chọn lấy mô hình có điểm metrics cao nhất từ đó đem chúng đi so sánh với mô hình của mình. Điểm metrics của bộ kiểm thử và các mô hình đối thủ được chúng tôi trình bày trong Bảng 7.4. Như các bảng metrics cũng đã cho thấy mô hình DTLN đạt các chỉ số metrics cao nhất so với hai mô hình còn lại do đó các chỉ số của mô hình DTLN này sẽ được đem đi so sánh với mô hình của chúng tôi.

Sau các phiên bản cải tiến khác nhau, mô hình Post của chúng tôi cũng đã đạt được một số kết quả đáng ghi nhận. Thông qua các số liệu có trong Bảng 7.6, chúng tôi có thể đánh giá được mô hình của chúng tôi tuy với lượng tham số chỉ bằng $1/9$ lượng tham số của các mô hình đã có (khoảng gần 1 triệu tham số) và lượng dữ liệu dùng để huấn luyện mô hình chỉ bằng $1/5$ so với của họ (mô hình của chúng tôi được huấn luyện trên khoảng 100 giờ dữ liệu đã lai tạo, trong khi mô

Bộ kiểm thử	Mô hình	STOI	NBPESQ	WBPESQ	SIG	BAK
SNR = -5	Baseline	0.64 (0.11)	1.36 (0.27)	1.10 (0.12)	3.66 (0.34)	0.98 (0.71)
	FullSubNet	0.54 (0.09)	1.22 (0.12)	1.06 (0.04)	2.85 (0.34)	1.87 (0.72)
	DCCRN	0.46 (0.09)	1.19 (0.15)	1.07 (0.06)	2.43 (0.33)	2.18 (0.58)
	DTLN	0.70 (0.12)	1.64 (0.36)	1.29 (0.21)	3.35 (0.37)	2.96 (0.43)
SNR = 0	Baseline	0.75 (0.09)	1.54 (0.36)	1.16 (0.19)	3.89 (0.27)	1.21 (0.73)
	FullSubNet	0.60 (0.07)	1.28 (0.13)	1.08 (0.05)	2.78 (0.36)	2.29 (0.57)
	DCCRN	0.55 (0.09)	1.31 (0.21)	1.11 (0.09)	2.66 (0.45)	2.37 (0.56)
	DTLN	0.82 (0.09)	2.07 (0.40)	1.55 (0.29)	3.59 (0.32)	3.29 (0.38)
SNR = 5	Baseline	0.84 (0.08)	1.78 (0.41)	1.29 (0.28)	4.15 (0.20)	1.63 (0.76)
	FullSubNet	0.64 (0.06)	1.37 (0.14)	1.11 (0.07)	2.70 (0.36)	2.65 (0.46)
	DCCRN	0.64 (0.08)	1.46 (0.25)	1.17 (0.13)	2.97 (0.53)	2.71 (0.50)
	DTLN	0.89 (0.07)	2.51 (0.39)	1.85 (0.34)	3.75 (0.32)	3.61 (0.33)
SNR = 10	Baseline	0.90 (0.07)	2.11 (0.46)	1.54 (0.37)	4.32 (0.15)	2.20 (0.69)
	FullSubNet	0.66 (0.06)	1.47 (0.16)	1.16 (0.08)	2.67 (0.37)	2.99 (0.39)
	DCCRN	0.69 (0.08)	1.64 (0.30)	1.25 (0.16)	3.20 (0.54)	3.13 (0.46)
	DTLN	0.93 (0.07)	2.87 (0.39)	2.15 (0.40)	3.90 (0.32)	3.85 (0.29)
SNR = 15	Baseline	0.94 (0.05)	2.57 (0.50)	1.97 (0.48)	4.39 (0.15)	2.75 (0.60)
	FullSubNet	0.68 (0.07)	1.55 (0.19)	1.20 (0.11)	2.69 (0.39)	3.25 (0.36)
	DCCRN	0.73 (0.09)	1.86 (0.39)	1.36 (0.23)	3.34 (0.55)	3.44 (0.44)
	DTLN	0.95 (0.05)	3.26 (0.36)	2.52 (0.43)	3.99 (0.32)	4.01 (0.25)
SNR = 20	Baseline	0.96 (0.04)	3.08 (0.48)	2.51 (0.51)	4.38 (0.19)	3.25 (0.46)
	FullSubNet	0.69 (0.07)	1.61 (0.23)	1.25 (0.15)	2.73 (0.44)	3.45 (0.35)
	DCCRN	0.75 (0.09)	2.07 (0.47)	1.47 (0.30)	3.38 (0.55)	3.65 (0.42)
	DTLN	0.96 (0.05)	3.56 (0.35)	2.86 (0.47)	4.07 (0.32)	4.13 (0.24)

Bảng 7.4: So sánh các mô hình thông qua các metrics trên bộ kiểm thử tiếng Việt

hình đã có được huấn luyện trên bộ dữ liệu trong cuộc thi DNS gồm khoảng 500 giờ dữ liệu sạch) nhưng mô hình chúng tôi vẫn đạt được mức độ đánh giá sự hiểu (STOI) tương đương với đối thủ (chênh lệch của STOI giữa mô hình chúng tôi và của đối thủ ít hơn 2%). Dù chất lượng âm thanh được đánh giá qua PESQ (cả wideband PESQ và narrowband PESQ) vẫn chưa thể vượt qua nhưng mức độ lọc nhiễu (BAK) từ âm thanh chúng tôi vẫn đạt được mức cao hơn hẳn mô hình đối thủ và hạn chế được sự mất mát của chất lượng giọng nói (SIG).

Để có thể so sánh một cách toàn diện mô hình được chúng tôi đề xuất với nhiều loại ngôn ngữ ở các ngưỡng SNR khác nhau, chúng tôi đã thực hiện kiểm thử với bộ dữ liệu đa ngôn ngữ được chúng tôi chuẩn bị ở phần trước. Ngoài ra cũng để có một cái nhìn khách quan hơn về chất lượng của mô hình chúng tôi với các sản phẩm mã nguồn mở và cả các sản phẩm thương mại, thông qua quá trình khảo

Trường hợp	Số mẫu	Độ trễ	Tứ phân vị 25%	Tứ phân vị 50%	Tứ phân vị 75%
(1.1)	34179	1.76 (0.76) ms	1.54 ms	1.62 ms	1.75 ms
(1.2)	35787	1.74 (0.48) ms	1.56 ms	1.65 ms	1.78 ms
(1.3)	42330	1.66 (0.37) ms	1.53 ms	1.61 ms	1.72 ms
(2)	31578	1.66 (0.44) ms	1.51 ms	1.60 ms	1.69 ms
(3)	36076	1.62 (0.44) ms	1.48 ms	1.56 ms	1.66 ms

Bảng 7.5: Độ trễ của pipeline lọc nhiễu động trong các trường hợp kiểm thử

sát chúng tôi chọn ra một số ứng dụng lọc nhiễu để so sánh với kết quả của mình bao gồm Sox² và CrystalSound³. Kết quả của việc kiểm thử mô hình của chúng tôi trên bộ dữ liệu đa ngôn ngữ và so sánh trên bộ dữ liệu tiếng Việt với Sox và CrystalSound được chúng tôi trình bày ở Bảng 7.9 và Bảng 7.7.

Ngoài vấn đề về chất lượng, độ trễ của mô hình động và pipeline chạy lọc nhiễu cũng rất quan trọng do vậy chúng tôi đã tiến hành đo đạc một số kết quả độ trễ của toàn bộ pipeline trong các trường hợp: *Người dùng đang nói chuyện thông qua Google Meet* (1), *Người dùng đang ghi âm bằng ứng dụng của chúng tôi* (2) và *Người dùng đang sử dụng ứng dụng nào cả* (3). Trong đó với trường hợp *Người dùng đang nói chuyện thông qua Google Meet* chúng tôi lại chia thêm thành các trường hợp con như *Người dùng chia sẻ màn hình (đang mở cửa sổ chia sẻ)* (1.1), *Người dùng chia sẻ màn hình (đang ẩn cửa sổ chia sẻ)* (1.2) và *Người dùng chỉ đang trò chuyện* (1.3). Với từng trường hợp như vậy chúng tôi thực nghiệm chạy thử trên máy ảo có CPU 4 cores, RAM 4GB và chạy trên hệ điều hành Windows 10, chúng tôi thu được độ trễ của pipeline được thể hiện như trong Bảng 7.5.

²<http://sox.sourceforge.net/sox.html>

³<https://crystalsound.ai/>

Chương 7. Thực nghiệm và kết quả

Bộ kiểm thử	Mô hình	STOI	NBPESQ	WBPESQ	SIG	BAK
SNR = -5	Baseline	0.64 (0.11)	1.36 (0.27)	1.10 (0.12)	3.66 (0.34)	0.98 (0.71)
	DTLN	0.70 (0.12)	1.64 (0.36)	1.29 (0.21)	3.35 (0.37)	2.96 (0.43)
	DTLN reduced	0.52 (0.16)	1.22 (0.21)	1.14 (0.15)	2.54 (0.41)	3.15 (0.28)
	DTLN Post	0.49 (0.18)	1.17 (0.16)	1.11 (0.09)	2.38 (0.41)	3.39 (0.28)
	Post v1	0.59 (0.13)	1.25 (0.22)	1.13 (0.12)	2.52 (0.46)	2.97 (0.38)
	Post v2	0.64 (0.13)	1.38 (0.26)	1.19 (0.16)	2.83 (0.43)	3.09 (0.42)
	Post v3 static	0.68 (0.12)	1.45 (0.29)	1.22 (0.17)	2.99 (0.42)	3.42 (0.38)
	Post v3 dynamic	0.68 (0.12)	1.41 (0.27)	1.19 (0.15)	3.02 (0.43)	3.52 (0.38)
SNR = 0	Baseline	0.75 (0.09)	1.54 (0.36)	1.16 (0.19)	3.89 (0.27)	1.21 (0.73)
	DTLN	0.82 (0.09)	2.07 (0.40)	1.55 (0.29)	3.59 (0.32)	3.29 (0.38)
	DTLN reduced	0.69 (0.15)	1.49 (0.33)	1.30 (0.23)	3.06 (0.42)	3.34 (0.33)
	DTLN Post	0.68 (0.16)	1.39 (0.30)	1.23 (0.18)	2.88 (0.44)	3.54 (0.33)
	Post v1	0.77 (0.10)	1.60 (0.38)	1.32 (0.24)	3.03 (0.42)	3.23 (0.42)
	Post v2	0.79 (0.10)	1.74 (0.37)	1.40 (0.27)	3.27 (0.38)	3.39 (0.43)
	Post v3 static	0.81 (0.08)	1.77 (0.37)	1.41 (0.27)	3.35 (0.36)	3.77 (0.33)
	Post v3 dynamic	0.81 (0.08)	1.74 (0.35)	1.37 (0.24)	3.42 (0.36)	3.83 (0.33)
SNR = 5	Baseline	0.84 (0.08)	1.78 (0.41)	1.29 (0.28)	4.15 (0.20)	1.63 (0.76)
	DTLN	0.89 (0.07)	2.51 (0.39)	1.85 (0.34)	3.75 (0.32)	3.61 (0.33)
	DTLN reduced	0.80 (0.13)	1.83 (0.44)	1.52 (0.34)	3.39 (0.37)	3.58 (0.33)
	DTLN Post	0.80 (0.13)	1.69 (0.39)	1.40 (0.26)	3.24 (0.40)	3.72 (0.32)
	Post v1	0.86 (0.07)	1.95 (0.35)	1.52 (0.26)	3.32 (0.36)	3.46 (0.37)
	Post v2	0.87 (0.07)	2.08 (0.36)	1.61 (0.29)	3.52 (0.37)	3.69 (0.34)
	Post v3 static	0.89 (0.07)	2.14 (0.38)	1.65 (0.32)	3.65 (0.34)	4.01 (0.26)
	Post v3 dynamic	0.89 (0.07)	2.10 (0.37)	1.59 (0.29)	3.70 (0.35)	4.06 (0.27)
SNR = 10	Baseline	0.90 (0.07)	2.11 (0.46)	1.54 (0.37)	4.32 (0.15)	2.20 (0.69)
	DTLN	0.93 (0.07)	2.87 (0.39)	2.15 (0.40)	3.90 (0.32)	3.85 (0.29)
	DTLN reduced	0.87 (0.11)	2.20 (0.51)	1.79 (0.42)	3.59 (0.34)	3.79 (0.32)
	DTLN Post	0.86 (0.11)	2.02 (0.46)	1.62 (0.34)	3.45 (0.36)	3.91 (0.30)
	Post v1	0.91 (0.07)	2.33 (0.42)	1.80 (0.34)	3.52 (0.36)	3.71 (0.34)
	Post v2	0.92 (0.06)	2.44 (0.40)	1.90 (0.35)	3.74 (0.36)	3.92 (0.32)
	Post v3 static	0.92 (0.06)	2.49 (0.42)	1.90 (0.39)	3.89 (0.32)	4.16 (0.24)
	Post v3 dynamic	0.92 (0.06)	2.45 (0.40)	1.83 (0.36)	3.91 (0.33)	4.20 (0.24)
SNR = 15	Baseline	0.94 (0.05)	2.57 (0.50)	1.97 (0.48)	4.39 (0.15)	2.75 (0.60)
	DTLN	0.95 (0.05)	3.26 (0.36)	2.52 (0.43)	3.99 (0.32)	4.01 (0.25)
	DTLN reduced	0.89 (0.10)	2.48 (0.56)	1.99 (0.51)	3.67 (0.31)	3.93 (0.26)
	DTLN Post	0.89 (0.10)	2.28 (0.52)	1.79 (0.43)	3.53 (0.33)	4.00 (0.25)
	Post v1	0.93 (0.06)	2.67 (0.46)	2.05 (0.41)	3.66 (0.36)	3.87 (0.31)
	Post v2	0.93 (0.06)	2.76 (0.46)	2.14 (0.43)	3.88 (0.36)	4.08 (0.27)
	Post v3 static	0.95 (0.05)	2.90 (0.44)	2.23 (0.45)	4.04 (0.30)	4.26 (0.22)
	Post v3 dynamic	0.95 (0.05)	2.83 (0.41)	2.13 (0.41)	4.04 (0.32)	4.29 (0.21)
SNR = 20	Baseline	0.96 (0.04)	3.08 (0.48)	2.51 (0.51)	4.38 (0.19)	3.25 (0.46)
	DTLN	0.96 (0.05)	3.56 (0.35)	2.86 (0.47)	4.07 (0.32)	4.13 (0.24)
	DTLN reduced	0.91 (0.09)	2.73 (0.60)	2.18 (0.57)	3.76 (0.31)	4.02 (0.26)
	DTLN Post	0.91 (0.09)	2.51 (0.57)	1.97 (0.49)	3.63 (0.32)	4.07 (0.26)
	Post v1	0.94 (0.05)	2.96 (0.49)	2.31 (0.47)	3.76 (0.35)	4.00 (0.31)
	Post v2	0.95 (0.05)	3.07 (0.47)	2.44 (0.47)	3.99 (0.35)	4.19 (0.25)
	Post v3 static	0.96 (0.05)	3.22 (0.47)	2.49 (0.51)	4.14 (0.30)	4.33 (0.21)
	Post v3 dynamic	0.96 (0.05)	3.12 (0.44)	2.38 (0.48)	4.14 (0.30)	4.35 (0.20)

Bảng 7.6: So sánh các metrics giữa các mô hình trên bộ kiểm thử tiếng Việt

Bộ kiểm thử	Mô hình	STOI	NBPESQ	WBPESQ	SIG	BAK
SNR = -5	Baseline	0.64 (0.11)	1.36 (0.27)	1.10 (0.12)	3.66 (0.34)	0.98 (0.71)
	Sox	0.48 (0.08)	1.16 (0.12)	1.06 (0.06)	3.04 (0.31)	2.90 (0.56)
	CrystalSound	0.44 (0.07)	1.78 (0.35)	1.40 (0.25)	3.49 (0.35)	3.03 (0.49)
	-	1.82 (0.36)	1.44 (0.25)	3.38 (0.34)	3.00 (0.44)	
	Post v3 static	0.68 (0.12)	1.45 (0.29)	1.22 (0.17)	2.99 (0.42)	3.42 (0.38)
SNR = 0	Post v3 dynamic	0.68 (0.12)	1.41 (0.27)	1.19 (0.15)	3.02 (0.43)	3.52 (0.38)
	Baseline	0.75 (0.09)	1.54 (0.36)	1.16 (0.19)	3.89 (0.27)	1.21 (0.73)
	Sox	0.58 (0.08)	1.23 (0.13)	1.09 (0.06)	3.11 (0.27)	3.25 (0.47)
	CrystalSound	0.48 (0.06)	2.18 (0.41)	1.70 (0.33)	3.68 (0.30)	3.37 (0.42)
	-	2.23 (0.44)	1.75 (0.35)	3.55 (0.30)	3.32 (0.40)	
SNR = 5	Post v3 static	0.81 (0.08)	1.77 (0.37)	1.41 (0.27)	3.35 (0.36)	3.77 (0.33)
	Post v3 dynamic	0.81 (0.08)	1.74 (0.35)	1.37 (0.24)	3.42 (0.36)	3.83 (0.33)
	Baseline	0.84 (0.08)	1.78 (0.41)	1.29 (0.28)	4.15 (0.20)	1.63 (0.76)
	Sox	0.63 (0.07)	1.32 (0.17)	1.13 (0.09)	3.15 (0.25)	3.58 (0.34)
	CrystalSound	0.50 (0.06)	2.51 (0.37)	1.97 (0.34)	3.82 (0.31)	3.62 (0.36)
SNR = 10	-	2.55 (0.41)	2.02 (0.38)	3.72 (0.31)	3.57 (0.35)	
	Post v3 static	0.89 (0.07)	2.14 (0.38)	1.65 (0.32)	3.65 (0.34)	4.01 (0.26)
	Post v3 dynamic	0.89 (0.07)	2.10 (0.37)	1.59 (0.29)	3.70 (0.35)	4.06 (0.27)
	Baseline	0.90 (0.07)	2.11 (0.46)	1.54 (0.37)	4.32 (0.15)	2.20 (0.69)
	Sox	0.65 (0.08)	1.36 (0.19)	1.16 (0.10)	3.13 (0.27)	3.78 (0.32)
SNR = 15	CrystalSound	0.51 (0.06)	2.81 (0.40)	2.23 (0.38)	3.93 (0.30)	3.81 (0.34)
	-	2.87 (0.43)	2.31 (0.42)	3.87 (0.30)	3.80 (0.36)	
	Post v3 static	0.92 (0.06)	2.49 (0.42)	1.90 (0.39)	3.89 (0.32)	4.16 (0.24)
	Post v3 dynamic	0.92 (0.06)	2.45 (0.40)	1.83 (0.36)	3.91 (0.33)	4.20 (0.24)
	Baseline	0.94 (0.05)	2.57 (0.50)	1.97 (0.48)	4.39 (0.15)	2.75 (0.60)
SNR = 20	Sox	0.65 (0.08)	1.39 (0.21)	1.20 (0.13)	3.11 (0.29)	3.85 (0.31)
	CrystalSound	0.51 (0.05)	3.07 (0.48)	2.46 (0.42)	4.00 (0.28)	3.89 (0.30)
	-	3.18 (0.45)	2.59 (0.46)	3.95 (0.29)	3.90 (0.30)	
	Post v3 static	0.95 (0.05)	2.90 (0.44)	2.23 (0.45)	4.04 (0.30)	4.26 (0.22)
	Post v3 dynamic	0.95 (0.05)	2.83 (0.41)	2.13 (0.41)	4.04 (0.32)	4.29 (0.21)

Bảng 7.7: So sánh các metrics giữa mô hình đề xuất và các ứng dụng Sox phiên bản 14.4.2, CrystalSound phiên bản 0.15.2 (dòng trên) và phiên bản 1.4.0.0 (dòng dưới) trên bộ kiểm thử tiếng Việt

Bộ kiểm thử	Ngôn ngữ	STOI	NBPESQ	WBPESQ	SIG	BAK
SNR = -5	Japanese	0.53 (0.09)	1.39 (0.41)	1.19 (0.37)	3.51 (0.31)	1.01 (0.67)
	French	0.58 (0.12)	1.32 (0.27)	1.10 (0.17)	3.62 (0.33)	1.15 (0.71)
	Russian	0.60 (0.11)	1.29 (0.27)	1.08 (0.14)	3.60 (0.30)	1.21 (0.72)
	Germany	0.63 (0.10)	1.36 (0.23)	1.11 (0.19)	3.63 (0.32)	1.11 (0.67)
	Portuguese	0.57 (0.10)	1.40 (0.22)	1.09 (0.10)	3.62 (0.28)	1.15 (0.63)
	Chinese	0.52 (0.11)	1.38 (0.33)	1.10 (0.21)	3.64 (0.31)	1.07 (0.66)
SNR = 0	Japanese	0.63 (0.09)	1.38 (0.32)	1.13 (0.19)	3.63 (0.30)	1.09 (0.60)
	French	0.68 (0.12)	1.40 (0.29)	1.10 (0.12)	3.81 (0.28)	1.34 (0.71)
	Russian	0.69 (0.10)	1.34 (0.24)	1.08 (0.08)	3.78 (0.27)	1.33 (0.67)
	Germany	0.74 (0.09)	1.46 (0.27)	1.12 (0.14)	3.83 (0.29)	1.30 (0.68)
	Portuguese	0.68 (0.10)	1.49 (0.36)	1.13 (0.22)	3.80 (0.25)	1.36 (0.71)
	Chinese	0.62 (0.12)	1.51 (0.36)	1.11 (0.16)	3.82 (0.26)	1.28 (0.68)
SNR = 5	Japanese	0.73 (0.08)	1.45 (0.32)	1.15 (0.17)	3.80 (0.27)	1.33 (0.67)
	French	0.78 (0.12)	1.56 (0.36)	1.18 (0.23)	4.05 (0.24)	1.70 (0.73)
	Russian	0.77 (0.10)	1.53 (0.36)	1.15 (0.21)	3.96 (0.30)	1.79 (0.77)
	Germany	0.83 (0.07)	1.65 (0.35)	1.22 (0.24)	4.07 (0.23)	1.69 (0.78)
	Portuguese	0.77 (0.08)	1.70 (0.36)	1.22 (0.22)	4.02 (0.23)	1.68 (0.70)
	Chinese	0.70 (0.12)	1.73 (0.40)	1.21 (0.20)	4.03 (0.22)	1.57 (0.68)
SNR = 10	Japanese	0.83 (0.07)	1.66 (0.42)	1.27 (0.31)	4.00 (0.21)	1.66 (0.69)
	French	0.86 (0.10)	1.89 (0.47)	1.37 (0.30)	4.23 (0.20)	2.08 (0.72)
	Russian	0.85 (0.09)	1.76 (0.45)	1.29 (0.29)	4.12 (0.31)	2.18 (0.75)
	Germany	0.90 (0.05)	1.96 (0.38)	1.40 (0.27)	4.23 (0.19)	2.12 (0.71)
	Portuguese	0.85 (0.08)	2.01 (0.46)	1.43 (0.34)	4.17 (0.21)	2.14 (0.70)
	Chinese	0.78 (0.13)	2.12 (0.53)	1.47 (0.41)	4.17 (0.19)	1.97 (0.79)
SNR = 15	Japanese	0.90 (0.05)	1.96 (0.48)	1.48 (0.41)	4.18 (0.15)	2.12 (0.69)
	French	0.91 (0.08)	2.27 (0.51)	1.69 (0.41)	4.33 (0.18)	2.59 (0.65)
	Russian	0.90 (0.07)	2.02 (0.51)	1.50 (0.39)	4.17 (0.36)	2.57 (0.65)
	Germany	0.94 (0.04)	2.37 (0.45)	1.77 (0.42)	4.33 (0.17)	2.68 (0.65)
	Portuguese	0.90 (0.07)	2.45 (0.55)	1.82 (0.49)	4.24 (0.21)	2.57 (0.65)
	Chinese	0.83 (0.13)	2.53 (0.51)	1.84 (0.42)	4.23 (0.17)	2.41 (0.69)
SNR = 20	Japanese	0.95 (0.03)	2.36 (0.55)	1.85 (0.55)	4.26 (0.14)	2.68 (0.57)
	French	0.95 (0.05)	2.68 (0.51)	2.13 (0.48)	4.35 (0.19)	3.09 (0.54)
	Russian	0.94 (0.07)	2.55 (0.59)	2.01 (0.58)	4.26 (0.31)	3.13 (0.50)
	Germany	0.97 (0.03)	2.76 (0.45)	2.21 (0.46)	4.35 (0.19)	3.11 (0.57)
	Portuguese	0.94 (0.05)	2.81 (0.54)	2.25 (0.53)	4.26 (0.20)	3.02 (0.52)
	Chinese	0.89 (0.10)	3.00 (0.49)	2.38 (0.47)	4.25 (0.15)	2.84 (0.62)

Bảng 7.8: Metric trên từng bộ kiểm thử đa ngôn ngữ

Bộ kiểm thử	Ngôn ngữ	Mô hình	STOI	NBPESQ	WPBESQ	SIG	BAK
Vietnamese		Baseline	0.64 (0.11)	1.36 (0.27)	1.10 (0.12)	3.66 (0.34)	0.98 (0.71)
		Post v3 static	0.68 (0.12)	1.45 (0.29)	1.22 (0.17)	2.99 (0.42)	3.42 (0.38)
		Post v3 dynamic	0.68 (0.12)	1.41 (0.27)	1.19 (0.15)	3.02 (0.43)	3.52 (0.38)
Japanese		Baseline	0.53 (0.09)	1.39 (0.41)	1.19 (0.37)	3.51 (0.31)	1.01 (0.67)
		Post v3 static	0.54 (0.10)	1.31 (0.20)	1.12 (0.09)	2.87 (0.40)	3.35 (0.46)
		Post v3 dynamic	0.54 (0.10)	1.33 (0.20)	1.14 (0.10)	2.83 (0.34)	3.62 (0.33)
French		Baseline	0.58 (0.12)	1.32 (0.27)	1.10 (0.17)	3.62 (0.33)	1.15 (0.71)
		Post v3 static	0.61 (0.13)	1.37 (0.22)	1.13 (0.09)	2.99 (0.44)	3.48 (0.39)
		Post v3 dynamic	0.62 (0.13)	1.37 (0.22)	1.14 (0.10)	2.93 (0.39)	3.61 (0.33)
SNR = -5	Russian	Baseline	0.60 (0.11)	1.29 (0.27)	1.08 (0.14)	3.60 (0.30)	1.21 (0.72)
		Post v3 static	0.60 (0.12)	1.35 (0.23)	1.11 (0.12)	2.91 (0.46)	3.46 (0.36)
		Post v3 dynamic	0.60 (0.13)	1.34 (0.23)	1.12 (0.11)	2.84 (0.44)	3.57 (0.35)
Germany		Baseline	0.63 (0.10)	1.36 (0.23)	1.11 (0.19)	3.63 (0.32)	1.11 (0.67)
		Post v3 static	0.67 (0.11)	1.36 (0.23)	1.13 (0.12)	3.05 (0.42)	3.51 (0.39)
		Post v3 dynamic	0.67 (0.11)	1.35 (0.22)	1.13 (0.11)	3.00 (0.37)	3.67 (0.32)
Portuguese		Baseline	0.57 (0.10)	1.40 (0.22)	1.09 (0.10)	3.62 (0.28)	1.15 (0.63)
		Post v3 static	0.62 (0.10)	1.39 (0.20)	1.12 (0.08)	3.02 (0.41)	3.51 (0.39)
		Post v3 dynamic	0.62 (0.11)	1.36 (0.19)	1.12 (0.08)	2.96 (0.38)	3.68 (0.31)
Chinese		Baseline	0.52 (0.11)	1.38 (0.33)	1.10 (0.21)	3.64 (0.31)	1.07 (0.66)
		Post v3 static	0.53 (0.12)	1.40 (0.28)	1.14 (0.13)	2.97 (0.48)	3.37 (0.40)
		Post v3 dynamic	0.54 (0.12)	1.39 (0.26)	1.14 (0.12)	2.89 (0.41)	3.47 (0.36)

(a) So sánh các metrics của mô hình đề xuất với các ngôn ngữ khác nhau có SNR = -5

Bảng 7.9: Metrics mô hình đề xuất với các loại ngôn ngữ

Bộ kiểm thử		Ngôn ngữ	Mô hình	STOI	NBPESQ	WPBESQ	SIG	BAK
Vietnamese			Baseline	0.53 (0.09)	1.39 (0.41)	1.19 (0.37)	3.51 (0.31)	1.01 (0.67)
			Post v3 static	0.81 (0.08)	1.77 (0.37)	1.41 (0.27)	3.35 (0.36)	3.77 (0.33)
			Post v3 dynamic	0.81 (0.08)	1.74 (0.35)	1.37 (0.24)	3.42 (0.36)	3.83 (0.33)
Japanese			Baseline	0.63 (0.09)	1.38 (0.32)	1.13 (0.19)	3.63 (0.30)	1.09 (0.60)
			Post v3 static	0.66 (0.09)	1.50 (0.25)	1.20 (0.12)	3.12 (0.34)	3.65 (0.44)
			Post v3 dynamic	0.67 (0.09)	1.51 (0.26)	1.22 (0.13)	3.07 (0.30)	3.86 (0.28)
French			Baseline	0.68 (0.12)	1.40 (0.29)	1.10 (0.12)	3.81 (0.28)	1.34 (0.71)
			Post v3 static	0.72 (0.12)	1.59 (0.28)	1.22 (0.15)	3.29 (0.38)	3.76 (0.38)
			Post v3 dynamic	0.73 (0.13)	1.59 (0.28)	1.23 (0.16)	3.24 (0.39)	3.85 (0.34)
SNR = 0			Baseline	0.69 (0.10)	1.34 (0.24)	1.08 (0.08)	3.78 (0.27)	1.33 (0.67)
			Post v3 static	0.72 (0.11)	1.55 (0.26)	1.18 (0.13)	3.22 (0.40)	3.75 (0.34)
			Post v3 dynamic	0.72 (0.12)	1.54 (0.28)	1.19 (0.15)	3.18 (0.39)	3.81 (0.30)
Germany			Baseline	0.74 (0.09)	1.46 (0.27)	1.12 (0.14)	3.83 (0.29)	1.30 (0.68)
			Post v3 static	0.79 (0.09)	1.56 (0.27)	1.21 (0.14)	3.37 (0.33)	3.88 (0.35)
			Post v3 dynamic	0.79 (0.09)	1.53 (0.28)	1.20 (0.15)	3.34 (0.33)	3.98 (0.27)
Portuguese			Baseline	0.68 (0.10)	1.49 (0.36)	1.13 (0.22)	3.80 (0.25)	1.36 (0.71)
			Post v3 static	0.74 (0.08)	1.61 (0.27)	1.21 (0.14)	3.33 (0.36)	3.84 (0.31)
			Post v3 dynamic	0.74 (0.08)	1.59 (0.26)	1.20 (0.13)	3.30 (0.36)	3.93 (0.28)
Chinese			Baseline	0.62 (0.12)	1.51 (0.36)	1.11 (0.16)	3.82 (0.26)	1.28 (0.68)
			Post v3 static	0.65 (0.12)	1.61 (0.27)	1.21 (0.11)	3.30 (0.36)	3.60 (0.42)
			Post v3 dynamic	0.66 (0.12)	1.60 (0.27)	1.22 (0.11)	3.23 (0.33)	3.69 (0.37)

(b) So sánh các metrics của mô hình đề xuất với các ngôn ngữ khác nhau có SNR = 0

Bộ kiểm thử		Ngôn ngữ	Mô hình	STOI	NBPESQ	WBPEPSQ	SIG	BAK
Vietnamese	Baseline		0.84 (0.08)	1.78 (0.41)	1.29 (0.28)	4.15 (0.20)	1.63 (0.76)	
	Post v3 static		0.89 (0.07)	2.14 (0.38)	1.65 (0.32)	3.65 (0.34)	4.01 (0.26)	
	Post v3 dynamic		0.89 (0.07)	2.10 (0.37)	1.59 (0.29)	3.70 (0.35)	4.06 (0.27)	
Japanese	Baseline		0.73 (0.08)	1.45 (0.32)	1.15 (0.17)	3.80 (0.27)	1.33 (0.67)	
	Post v3 static		0.77 (0.07)	1.75 (0.29)	1.33 (0.17)	3.37 (0.27)	4.03 (0.31)	
	Post v3 dynamic		0.77 (0.07)	1.74 (0.28)	1.33 (0.17)	3.33 (0.29)	4.12 (0.22)	
French	Baseline		0.78 (0.12)	1.56 (0.36)	1.18 (0.23)	4.05 (0.24)	1.70 (0.73)	
	Post v3 static		0.81 (0.11)	1.89 (0.32)	1.37 (0.19)	3.63 (0.34)	4.07 (0.34)	
	Post v3 dynamic		0.81 (0.12)	1.86 (0.33)	1.36 (0.20)	3.60 (0.35)	4.11 (0.31)	
SNR = 5	Baseline		0.77 (0.10)	1.53 (0.36)	1.15 (0.21)	3.96 (0.30)	1.79 (0.77)	
	Post v3 static		0.78 (0.12)	1.80 (0.34)	1.29 (0.20)	3.50 (0.45)	3.99 (0.33)	
	Post v3 dynamic		0.78 (0.13)	1.76 (0.34)	1.28 (0.19)	3.45 (0.47)	4.00 (0.32)	
German	Baseline		0.83 (0.07)	1.65 (0.35)	1.22 (0.24)	4.07 (0.23)	1.69 (0.78)	
	Post v3 static		0.86 (0.06)	1.88 (0.32)	1.36 (0.20)	3.71 (0.30)	4.17 (0.30)	
	Post v3 dynamic		0.86 (0.07)	1.84 (0.34)	1.35 (0.21)	3.69 (0.30)	4.23 (0.21)	
Portuguese	Baseline		0.77 (0.08)	1.70 (0.36)	1.22 (0.22)	4.02 (0.23)	1.68 (0.70)	
	Post v3 static		0.82 (0.07)	1.88 (0.29)	1.33 (0.17)	3.65 (0.32)	4.12 (0.29)	
	Post v3 dynamic		0.82 (0.07)	1.85 (0.30)	1.31 (0.17)	3.63 (0.32)	4.17 (0.24)	
Chinese	Baseline		0.70 (0.12)	1.73 (0.40)	1.21 (0.20)	4.03 (0.22)	1.57 (0.68)	
	Post v3 static		0.74 (0.12)	1.95 (0.33)	1.38 (0.20)	3.58 (0.31)	3.84 (0.37)	
	Post v3 dynamic		0.73 (0.12)	1.91 (0.33)	1.36 (0.19)	3.50 (0.30)	3.88 (0.34)	

(c) So sánh các metrics của mô hình đề xuất với các ngôn ngữ khác nhau có SNR = 5

Bộ kiểm thử		Ngôn ngữ	Mô hình	STOI	NBPESQ	WPBESQ	SIG	BAK
Vietnamese			Baseline	0.90 (0.07)	2.11 (0.46)	1.54 (0.37)	4.32 (0.15)	2.20 (0.69)
			Post v3 static	0.92 (0.06)	2.49 (0.42)	1.90 (0.39)	3.89 (0.32)	4.16 (0.24)
			Post v3 dynamic	0.92 (0.06)	2.45 (0.40)	1.83 (0.36)	3.91 (0.33)	4.20 (0.24)
Japanese			Baseline	0.83 (0.07)	1.66 (0.42)	1.27 (0.31)	4.00 (0.21)	1.66 (0.69)
			Post v3 static	0.85 (0.05)	2.05 (0.34)	1.51 (0.22)	3.63 (0.25)	4.26 (0.23)
			Post v3 dynamic	0.86 (0.06)	2.06 (0.38)	1.52 (0.27)	3.60 (0.26)	4.29 (0.19)
French			Baseline	0.86 (0.10)	1.89 (0.47)	1.37 (0.30)	4.23 (0.20)	2.08 (0.72)
			Post v3 static	0.88 (0.10)	2.28 (0.38)	1.59 (0.28)	3.89 (0.32)	4.27 (0.30)
			Post v3 dynamic	0.87 (0.10)	2.25 (0.40)	1.58 (0.29)	3.88 (0.34)	4.29 (0.28)
SNR = 10			Baseline	0.85 (0.09)	1.76 (0.45)	1.29 (0.29)	4.12 (0.31)	2.18 (0.75)
			Post v3 static	0.85 (0.11)	2.14 (0.42)	1.50 (0.29)	3.74 (0.47)	4.17 (0.31)
			Post v3 dynamic	0.85 (0.11)	2.10 (0.42)	1.49 (0.29)	3.70 (0.49)	4.17 (0.33)
Germany			Baseline	0.90 (0.05)	1.96 (0.38)	1.40 (0.27)	4.23 (0.19)	2.12 (0.71)
			Post v3 static	0.91 (0.05)	2.24 (0.33)	1.56 (0.23)	3.90 (0.26)	4.32 (0.21)
			Post v3 dynamic	0.91 (0.05)	2.20 (0.36)	1.54 (0.25)	3.89 (0.27)	4.33 (0.19)
Portuguese			Baseline	0.85 (0.08)	2.01 (0.46)	1.43 (0.34)	4.17 (0.21)	2.14 (0.70)
			Post v3 static	0.88 (0.07)	2.23 (0.34)	1.53 (0.26)	3.88 (0.28)	4.30 (0.22)
			Post v3 dynamic	0.87 (0.07)	2.18 (0.33)	1.49 (0.24)	3.85 (0.29)	4.32 (0.19)
Chinese			Baseline	0.78 (0.13)	2.12 (0.53)	1.47 (0.41)	4.17 (0.19)	1.97 (0.79)
			Post v3 static	0.79 (0.13)	2.36 (0.35)	1.59 (0.25)	3.82 (0.27)	4.04 (0.35)
			Post v3 dynamic	0.78 (0.13)	2.26 (0.37)	1.54 (0.25)	3.75 (0.28)	4.07 (0.32)

(d) So sánh các metrics của mô hình đề xuất với các ngôn ngữ khác nhau có SNR = 10

Bộ kiểm thử		Ngôn ngữ	Mô hình	STOI	NBPESQ	WBPESQ	SIG	BAK
Vietnamese			Baseline	0.94 (0.05)	2.57 (0.50)	1.97 (0.48)	4.39 (0.15)	2.75 (0.60)
			Post v3 static	0.95 (0.05)	2.90 (0.44)	2.23 (0.45)	4.04 (0.30)	4.26 (0.22)
			Post v3 dynamic	0.95 (0.05)	2.83 (0.41)	2.13 (0.41)	4.04 (0.32)	4.29 (0.21)
Japanese			Baseline	0.90 (0.05)	1.96 (0.48)	1.48 (0.41)	4.18 (0.15)	2.12 (0.69)
			Post v3 static	0.91 (0.04)	2.41 (0.33)	1.75 (0.26)	3.84 (0.23)	4.42 (0.14)
			Post v3 dynamic	0.92 (0.04)	2.47 (0.36)	1.80 (0.31)	3.84 (0.24)	4.42 (0.14)
French			Baseline	0.91 (0.08)	2.27 (0.51)	1.69 (0.41)	4.33 (0.18)	2.59 (0.65)
			Post v3 static	0.92 (0.07)	2.64 (0.35)	1.84 (0.32)	4.08 (0.26)	4.42 (0.21)
			Post v3 dynamic	0.91 (0.08)	2.59 (0.37)	1.81 (0.32)	4.07 (0.27)	4.42 (0.19)
SNR = 15			Baseline	0.90 (0.07)	2.02 (0.51)	1.50 (0.39)	4.17 (0.36)	2.57 (0.65)
			Post v3 static	0.88 (0.11)	2.41 (0.48)	1.69 (0.37)	3.85 (0.51)	4.26 (0.34)
			Post v3 dynamic	0.88 (0.11)	2.34 (0.47)	1.65 (0.34)	3.82 (0.53)	4.26 (0.33)
Germany			Baseline	0.94 (0.04)	2.37 (0.45)	1.77 (0.42)	4.33 (0.17)	2.68 (0.65)
			Post v3 static	0.94 (0.04)	2.65 (0.36)	1.86 (0.32)	4.08 (0.24)	4.44 (0.17)
			Post v3 dynamic	0.94 (0.04)	2.64 (0.36)	1.85 (0.32)	4.08 (0.24)	4.43 (0.16)
Portuguese			Baseline	0.90 (0.07)	2.45 (0.55)	1.82 (0.49)	4.24 (0.21)	2.57 (0.65)
			Post v3 static	0.91 (0.07)	2.56 (0.34)	1.76 (0.29)	4.04 (0.28)	4.41 (0.19)
			Post v3 dynamic	0.91 (0.07)	2.52 (0.33)	1.72 (0.28)	4.02 (0.29)	4.41 (0.17)
Chinese			Baseline	0.83 (0.13)	2.53 (0.51)	1.84 (0.42)	4.23 (0.17)	2.41 (0.69)
			Post v3 static	0.83 (0.13)	2.74 (0.35)	1.86 (0.32)	3.98 (0.24)	4.20 (0.31)
			Post v3 dynamic	0.82 (0.12)	2.63 (0.37)	1.76 (0.31)	3.93 (0.25)	4.20 (0.28)

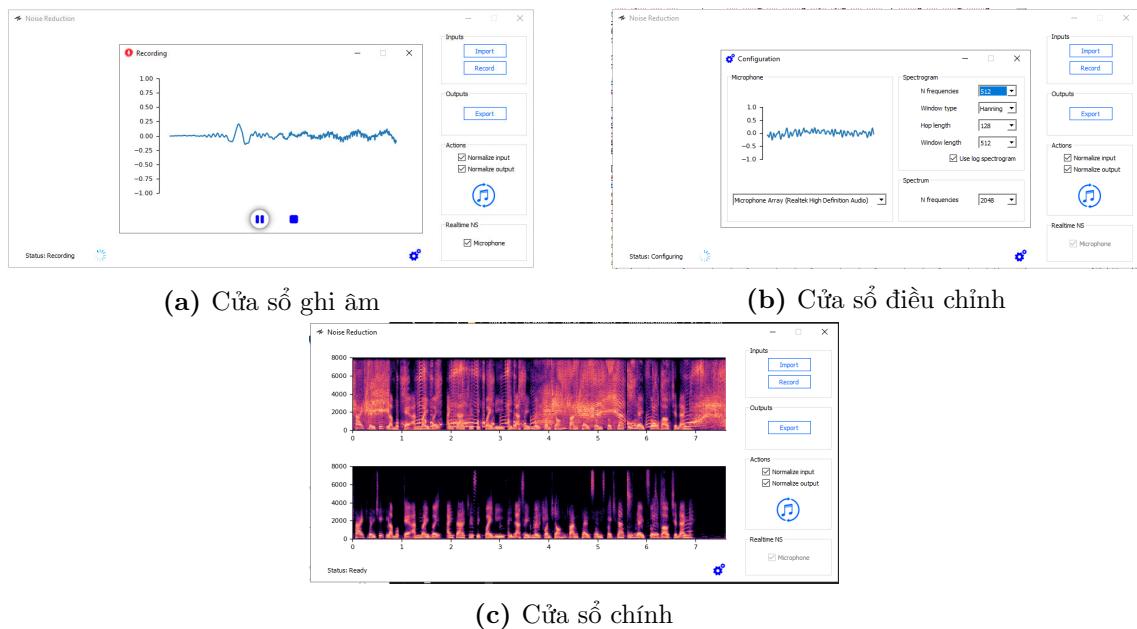
(e) So sánh các metrics của mô hình đề xuất với các ngôn ngữ khác nhau có SNR = 15

Bộ kiểm thử		Ngôn ngữ	Mô hình	STOI	NBPESQ	WBPESQ	SIG	BAK
Vietnamese			Baseline	0.96 (0.04)	3.08 (0.48)	2.51 (0.51)	4.38 (0.19)	3.25 (0.46)
			Post v3 static	0.96 (0.05)	3.22 (0.47)	2.49 (0.51)	4.14 (0.30)	4.33 (0.21)
			Post v3 dynamic	0.96 (0.05)	3.12 (0.44)	2.38 (0.48)	4.14 (0.30)	4.35 (0.20)
Japanese			Baseline	0.95 (0.03)	2.36 (0.55)	1.85 (0.55)	4.26 (0.14)	2.68 (0.57)
			Post v3 static	0.95 (0.02)	2.79 (0.31)	2.03 (0.31)	3.97 (0.21)	4.50 (0.14)
			Post v3 dynamic	0.96 (0.02)	2.92 (0.34)	2.16 (0.37)	4.00 (0.21)	4.50 (0.12)
French			Baseline	0.95 (0.05)	2.68 (0.51)	2.13 (0.48)	4.35 (0.19)	3.09 (0.54)
			Post v3 static	0.94 (0.06)	2.95 (0.35)	2.11 (0.37)	4.19 (0.24)	4.49 (0.21)
			Post v3 dynamic	0.94 (0.06)	2.92 (0.36)	2.06 (0.36)	4.19 (0.26)	4.48 (0.20)
SNR = 20			Baseline	0.94 (0.07)	2.55 (0.59)	2.01 (0.58)	4.26 (0.31)	3.13 (0.50)
			Post v3 static	0.91 (0.09)	2.75 (0.49)	1.95 (0.48)	4.04 (0.46)	4.37 (0.30)
			Post v3 dynamic	0.91 (0.09)	2.66 (0.47)	1.86 (0.44)	4.02 (0.45)	4.36 (0.29)
Germany			Baseline	0.97 (0.03)	2.76 (0.45)	2.21 (0.46)	4.35 (0.19)	3.11 (0.57)
			Post v3 static	0.96 (0.03)	2.98 (0.33)	2.14 (0.37)	4.17 (0.22)	4.50 (0.14)
			Post v3 dynamic	0.96 (0.03)	2.99 (0.33)	2.16 (0.37)	4.17 (0.22)	4.48 (0.15)
Portuguese			Baseline	0.94 (0.05)	2.81 (0.54)	2.25 (0.53)	4.26 (0.20)	3.02 (0.52)
			Post v3 static	0.93 (0.06)	2.85 (0.34)	2.00 (0.34)	4.12 (0.24)	4.44 (0.18)
			Post v3 dynamic	0.93 (0.06)	2.82 (0.36)	1.96 (0.35)	4.12 (0.26)	4.45 (0.17)
Chinese			Baseline	0.89 (0.10)	3.00 (0.49)	2.38 (0.47)	4.25 (0.15)	2.84 (0.62)
			Post v3 static	0.86 (0.10)	3.05 (0.35)	2.09 (0.37)	4.09 (0.22)	4.28 (0.28)
			Post v3 dynamic	0.86 (0.10)	2.91 (0.38)	1.95 (0.35)	4.04 (0.24)	4.28 (0.26)

(f) So sánh các metrics của mô hình đề xuất với các ngôn ngữ khác nhau có SNR = 20

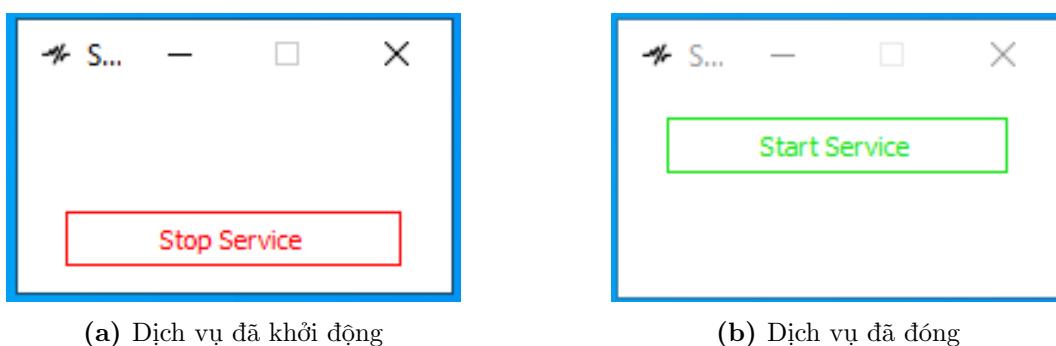
7.3 Một số kết quả của ứng dụng lọc nhiễu

Ứng dụng lọc nhiễu của chúng tôi được chia làm hai phần: *Lọc nhiễu tĩnh* và *Lọc nhiễu động*. Phần lọc nhiễu tĩnh được chúng tôi tích hợp vào ứng dụng, giao diện ứng dụng được chúng tôi thiết kế như trong Hình 7.6. Trong Hình 7.6 thể hiện cửa sổ chính, cửa sổ điều chỉnh và cửa sổ ghi âm của ứng dụng chúng tôi.



Hình 7.6: Một số hình ảnh của ứng dụng lọc nhiễu tĩnh

Phần lọc nhiễu động được chúng tôi thiết kế như một dịch vụ của hệ điều hành và cho phép điều khiển thông qua một giao diện độc lập, các hình ảnh của ứng dụng này khi kích hoạt và đóng dịch vụ lọc nhiễu được cài đặt trên hệ điều hành được thể hiện như trong Hình 7.7.



Hình 7.7: Một số hình ảnh của ứng dụng lọc nhiễu động

Một trong những khó khăn lớn nhất của chúng tôi trong quá trình hoàn thành ứng dụng đó là việc kiểm thử ứng dụng lọc nhiễu động. Môi trường chạy ở dưới nền tảng của hệ điều hành được gọi là **môi trường không lỗi** (exception-free environment). Trong môi trường loại này, các lỗi phát sinh ra dù có nghiêm trọng cũng không thể bị “bắt”(catch) bởi bất cứ cơ chế nào. Do đó, để tránh phát sinh tình trạng lỗi nghiêm trọng do lỗi của code ở kernel, chúng tôi cần kiểm tra rất kĩ các dòng này, kể cả trong quá trình viết code cũng như khi ra chạy thử. Đồng thời để kiểm tra sự ổn định của nó, chúng tôi cũng tiến hành kiểm thử trên từng đoạn năm phút hội thoại thực (dùng để đo Bảng 7.5). Vì vậy nên ở ứng dụng động sẽ không kiểm thử thực tế theo testcase mà sẽ được đem vào thử nghiệm trong thực tế.

Ở ứng dụng tĩnh, ta sẽ sử dụng các trường hợp kiểm thử trên phần mềm sau cùng. Các trường hợp này được chia như sau

- 1) *Bộ quản lý trạng thái*: Tạo liên tục một số lượng luồng (functional thread) bắt kí (ít hơn 5 luồng), mỗi lần tạo cách nhau 0.1ms. Mỗi luồng chờ trong 5s và in ra thứ tự được tạo của mình. Nếu thứ tự thỏa mãn từ bé đến lớn thì bộ quản lý hoạt động đúng.
- 2) *Luồng*: Luồng kiểm thử được tạo ra, in ra timestamp mở đầu sau đó chờ trong 1s và sau đó lại in ra timestamp của thời điểm hiện tại. Nếu timestamp mở đầu bé hơn timestamp lúc sau thì luồng hoạt động đúng.

Đây chính là hai thành phần cốt lõi của ứng dụng, các tác vụ khác hầu hết chỉ sử dụng thư viện để chạy mô hình, đọc dữ liệu từ microphone hoặc gán dữ liệu vào một biến ở môi trường. Một số tác vụ khác lại phải sử dụng bằng kiểm thực trực quan hóa như tác vụ hiển thị ra màn hình. Hầu hết việc kiểm thử của chúng tôi được thực hiện bằng thủ công và đây cũng chính là một trong những khó khăn lớn nhất trong luận văn này.

Chương 8

Tổng kết

8.1 Kết quả đạt được

Trong luận văn này, ngoài việc hiện thực được ứng dụng sử dụng mô hình lọc nhiễu bằng học sâu, chúng tôi cũng đã đề xuất một hàm tăng cường măt măt, phân tích các đặc điểm của hàm tăng cường măt măt và phân tích một số đặc điểm về đạo hàm của chúng (Mục 5.3), đồng thời chúng tôi cũng sử dụng chính hàm măt măt đã được tăng cường để huấn luyện mô hình được chúng tôi đề xuất và đạt được các kết quả khá khả quan (Bảng 7.6).

Các bộ dữ liệu được chúng tôi tái tạo và giả lập cho các môi trường phức tạp lẫn nhiều loại tiếng ồn và cho nhiều loại ngôn ngữ khác nhau. Điều này cho thấy hiệu quả mà hàm tăng cường măt măt mà chúng tôi đề xuất khá lớn. Với hạn chế về mặt dữ liệu sạch lẫn số lượng nhiễu và cả hạ tầng sử dụng để huấn luyện mô hình, mô hình chúng tôi với một số metrics vẫn vượt qua cả những mô hình đạt các thứ hạng cao trong các cuộc thi lớn (Bảng 7.6). Không chỉ vậy, việc ứng dụng mô hình vào lọc nhiễu thời gian thực cũng được chúng tôi chú trọng và kết quả là chúng tôi cũng đã hiện thực một ứng dụng có khả năng lọc nhiễu trên cả file lẫn lọc trong thời gian thực (Mục 7.3). Một số yêu cầu được chúng tôi đặt ra vượt cả kì vọng ban đầu (Bảng 7.7 và Bảng 7.5).

8.2 Hạn chế và hướng phát triển

Tuy vậy, các hướng tiếp cận của chúng tôi còn một số điểm hạn chế. Với loại nhiễu được giới hạn lại và không quá nhiều (Chương 1), chúng tôi dễ dàng thấy được nhiều điều bất cập khi ứng dụng mô hình vào thực tế. Tuy có kết quả khá ổn định trên tập đa ngôn ngữ, nhưng kết quả vẫn có thể được cải thiện (Bảng 7.9). Hiện tại chỉ huấn luyện mô hình với tiếng Anh vẫn đang là hạn chế của chúng tôi. Do chỉ tập trung vào các âm tiết mang thông tin quan trọng bên trong giọng nói, chúng tôi bỏ qua các yếu tố về chất giọng, âm điệu, ngữ nghĩa của câu từ. Điều này đôi lúc làm cho mô hình bị nhầm lẫn, giữ lại các âm nhiễu tương tự giọng nói mà đáng lẽ phải được loại bỏ đi. Không những vậy, việc loại bỏ các âm cầu tạo nên ngữ điệu cũng làm cho giọng nói của người nói trong môi trường nhiễu mạnh bị vang và trầm xuống rất nhiều. Mục tiêu kế tiếp của chúng tôi chính là khắc phục những điểm trên, cải tiến mô hình và mở rộng dữ liệu huấn luyện của mình.

Không những vậy, việc “continual learning” cho mô hình cũng cần được chúng tôi chú trọng. Như đã có trình bày ở cuối Mục 6.3, mô hình không chỉ cần thỏa mãn tính thời gian thực mà còn phải liên tục linh hoạt thích nghi với môi trường xung quanh người dùng. Do đó, ngoài mục tiêu khắc phục hạn chế và cải thiện mô hình, chúng tôi đặt ra mục tiêu làm cho mô hình này có khả năng thích nghi đối với môi trường xung quanh người dùng.

Tài liệu tham khảo

- [1] Hao, X., Su, X., Horaud, R., & Li, X. (2021, June). FullSubNet: A Full-Band and Sub-Band Fusion Model for Real-Time Single-Channel Speech Enhancement. In ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 6633-6637). IEEE.
- [2] Hu, Y., Liu, Y., Lv, S., Xing, M., Zhang, S., Fu, Y., Wu, J., Zhang, B., Xie, L. (2020) DCCRN: Deep Complex Convolution Recurrent Network for Phase-Aware Speech Enhancement. Proc. Interspeech 2020, 2472-2476, doi: 10.21437/Interspeech.2020-2537
- [3] Westhausen, N.L., Meyer, B.T. (2020) Dual-Signal Transformation LSTM Network for Real-Time Noise Suppression. Proc. Interspeech 2020, 2477-2481, doi: 10.21437/Interspeech.2020-2631.
- [4] Li, X., & Horaud, R. (2020, October). Online Monaural Speech Enhancement Using Delayed Subband LSTM. In Interspeech 2020 (pp. 2462-2466).
- [5] Smith, Julius O. (2003). Mathematics of the Discrete Fourier Transform (DFT) with Audio Applications, Second Edition, W3K Publishing, <http://books.w3k.org/>, ISBN 978-0-9745607-4-8.
- [6] Cooley, J. W., & Tukey, J. W. (1965). An algorithm for the machine calculation of complex Fourier series. Mathematics of computation, 19(90), 297-301.
- [7] Gentleman, W. M., & Sande, G. (1966, November). Fast Fourier transforms: for fun and profit. In Proceedings of the November 7-10, 1966, fall joint computer conference (pp. 563-578).
- [8] Brigham, E. O., & Morrow, R. E. (1967). The fast Fourier transform. IEEE spectrum, 4(12), 63-70.

- [9] Cooley, J. W., Lewis, P. A., & Welch, P. D. (1967). Historical notes on the fast Fourier transform. *Proceedings of the IEEE*, 55(10), 1675-1677.
- [10] Cochran, W. T., Cooley, J. W., Favin, D. L., Helms, H. D., Kaenel, R. A., Lang, W. W., ... & Welch, P. D. (1967). What is the fast Fourier transform?. *Proceedings of the IEEE*, 55(10), 1664-1674.
- [11] Bergland, G. D. (1968). Numerical analysis: A fast Fourier transform algorithm for real-valued series. *Communications of the ACM*, 11(10), 703-710.
- [12] Rader, C., & Brenner, N. (1976). A new principle for fast Fourier transformation. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 24(3), 264-266.
- [13] Nussbaumer, H. J. (1981). The fast Fourier transform. In *Fast Fourier Transform and Convolution Algorithms* (pp. 80-111). Springer, Berlin, Heidelberg.
- [14] Reddy, C. K., Gopal, V., Cutler, R., Beyrami, E., Cheng, R., Dubey, H., ... & Gehrke, J. (2020). The INTERSPEECH 2020 Deep Noise Suppression Challenge: Datasets, Subjective Testing Framework, and Challenge Results.
- [15] Barker, J., Watanabe, S., Vincent, E., & Trmal, J. (2018). The Fifth'CHiME'Speech Separation and Recognition Challenge: Dataset, Task and Baselines. *Proc. Interspeech 2018*, 1561-1565.
- [16] Thiemann, J., Ito, N., & Vincent, E. (2013, June). The Diverse Environments Multi-channel Acoustic Noise Database (DEMAND): A database of multichannel environmental noise recordings. In *Proceedings of Meetings on Acoustics ICA2013* (Vol. 19, No. 1, p. 035081). Acoustical Society of America.
- [17] Valentini-Botinhao, C. (2017). Noisy speech database for training speech enhancement algorithms and tts models.
- [18] Luong, H. T., & Vu, H. Q. (2016, December). A non-expert Kaldi 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)* (pp. 51-55).

- [19] Streijl, R. C., Winkler, S., & Hands, D. S. (2016). Mean opinion score (MOS) revisited: methods and applications, limitations and alternatives. *Multimedia Systems*, 22(2), 213-227.
- [20] Taal, C. H., Hendriks, R. C., Heusdens, R., & Jensen, J. (2010, March). A short-time objective intelligibility measure for time-frequency weighted noisy speech. In 2010 IEEE international conference on acoustics, speech and signal processing (pp. 4214-4217). IEEE.
- [21] Rix, A. W., Beerends, J. G., Hollier, M. P., & Hekstra, A. P. (2001, May). Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs. In 2001 IEEE international conference on acoustics, speech, and signal processing. Proceedings (Cat. No. 01CH37221) (Vol. 2, pp. 749-752). IEEE.
- [22] Loizou, P. C. (2007). Chapter 11 “Objective Quality and Intelligibility Measures”. In *Speech Enhancement: Theory and Practice*, Second Edition. CRC press.
- [23] Reddy, C. K., Gopal, V., & Cutler, R. (2021, June). Dnsmos: A non-intrusive perceptual objective speech quality metric to evaluate noise suppressors. In ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 6493-6497). IEEE.
- [24] Reddy, C. K., Gopal, V., & Cutler, R. (2022, May). DNSMOS P. 835: A non-intrusive perceptual objective speech quality metric to evaluate noise suppressors. In ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 886-890). IEEE.
- [25] ITU-T Recommendation P.808, Subjective evaluation of speech quality with a crowdsourcing approach. International Telecommunication Union, Geneva, 2021.
- [26] ITU-T Recommendation P.835, Subjective test methodology for evaluating speech communication systems that include noise suppression algorithm. International Telecommunication Union, Geneva, 2003.
- [27] Loizou, P. C. (2011). Speech quality assessment. In *Multimedia analysis, processing and communications* (pp. 623-654). Springer, Berlin, Heidelberg.

- [28] Le Roux, J., Wisdom, S., Erdogan, H., & Hershey, J. R. (2019, May). SDR—half-baked or well done?. In ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 626-630). IEEE.
- [29] Berreth, F. (2010). Low latency real-time audio streaming. U.S. Patent No. 7,706,901. Washington, DC: U.S. Patent and Trademark Office.
- [30] Orwick, P., & Smith, G. (2007). Developing Drivers with the Windows Driver Foundation. Microsoft Press.
- [31] Andrea Allievi, Alex Ionescu, David A. Solomon, Kate Chase, Mark E. Russinovich. (2021). Windows Internals 7th Edition. Microsoft Press.
- [32] Smyth, T. (2020). Music 175: Psychoacoustics Spring 2020. Music Journal, 3(2), 45-52.
- [33] ITU-T Recommendation P.862.1, Mapping function for transforming P.862 raw result scores to MOS-LQO. International Telecommunication Union, Geneva, 2003.
- [34] ITU-T Recommendation P.862.2, Wideband extension to Recommendation P.862 for the assessment of wideband telephone networks and speech codecs. International Telecommunication Union, Geneva, 2007.
- [35] American National Standard Institute. (2004). ANSI S1.11: American National Standard Specification for Octave-Band and Fractional-Octave-Band Analog and Digital Filters. Acoustical Society of America (ASA).
- [36] Zwicker, E., & Fastl, H. (2013). Psychoacoustics: Facts and models (Vol. 22). Springer Science & Business Media.
- [37] Stevens, S. S. (1936). A scale for the measurement of a psychological magnitude: loudness. Psychological Review, 43(5), 405.
- [38] Stevens, S. S. (1955). The measurement of loudness. The Journal of the Acoustical Society of America, 27(5), 815-829.
- [39] Stevens, S. S. (1956). Calculation of the loudness of complex noise. The Journal of the Acoustical Society of America, 28(5), 807-832.

- [40] Marks, L. E., & Stevens, J. C. (1968). The form of the psychophysical function near threshold. *Perception & Psychophysics*, 4(5), 315-318.
- [41] Lochner, J. P. A., & Burger, J. F. (1961). Form of the loudness function in the presence of masking noise. *The Journal of the Acoustical Society of America*, 33(12), 1705-1707.
- [42] Zwicker, E., & Scharf, B. (1965). A model of loudness summation. *Psychological review*, 72(1), 3.
- [43] ISO-532-1. (2017). Acoustics - Method for calculating loudness - Zwicker's method. International Standard.
- [44] Allen, J. (1977). Short term spectral analysis, synthesis, and modification by discrete Fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 25(3), 235-238.
- [45] Allen, J. B., & Rabiner, L. R. (1977). A unified approach to short-time Fourier analysis and synthesis. *Proceedings of the IEEE*, 65(11), 1558-1564.
- [46] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8), 1735-1780.
- [47] Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. *Neural computation*, 12(10), 2451-2471.
- [48] Powell, M. J. D. (1981). Chapter 13 “Approximation to periodic functions”. In *Approximation theory and methods*. Cambridge university press.
- [49] Low, E. & Winther, R. (2001). Fourier Analysis. Notes for Mat 120B, University of Oslo.
- [50] National Instruments. Instrument Fundamentals: Understanding FFTs and Windowing. In *Instrument Fundamentals Series*, National Instruments, <https://www.ni.com/>.

Tài liệu tham khảo

Phụ lục A

Biến đổi Fourier

A.1 Phép biến đổi Fourier

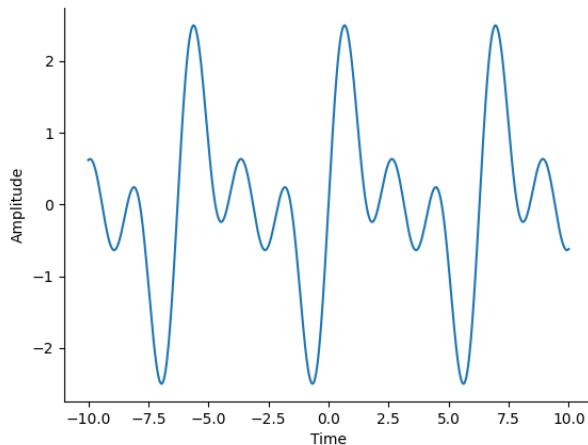
Trong thực tế, quá trình truyền âm trong không khí từ nguồn âm tới bộ phận nhận được (trong luận văn này, bộ phận thu nhận âm thanh từ nguồn phát là microphone của máy tính) luôn tiềm ẩn những sự gián đoạn làm ảnh hưởng tới thông tin mà âm thanh ban đầu truyền đi. Tới khi máy tính nhận được các thông tin từ microphone thì âm thanh nhận được đó chỉ là một tập hợp của âm thanh gốc và rất nhiều các âm thanh khác lẫn vào đó. Dễ thấy việc loại bỏ trực tiếp từ các tín hiệu thu được từ microphone là vẫn đề rất khó vì một số âm thanh có sự tương đồng với âm thanh cần làm sạch. Vì thế, ta trình bày một công cụ có khả năng phân tích các tín hiệu bị trộn lẫn này lại với nhau, đó là **Phép biến đổi Fourier**. Phép biến đổi Fourier có hai dạng: *Phép biến đổi Fourier liên tục* và *Phép biến đổi Fourier rời rạc*. Trước hết chúng tôi sẽ đề cập tới phép biến đổi Fourier liên tục.

A.1.1 Định nghĩa và ví dụ

Với một tín hiệu liên tục $f(t)$ theo thời gian t (ở đây ta xét hàm theo thời gian vì nó liên quan mật thiết tới vấn đề mà chúng tôi cần giải quyết), **biến đổi Fourier** với tốc độ góc ω (là tốc độ sóng dao động trong một chu kì) của $f(t)$ được định nghĩa như sau

$$F(\omega) = \int_{-\infty}^{+\infty} f(t)e^{-\omega t i} dt. \quad (\text{A.1})$$

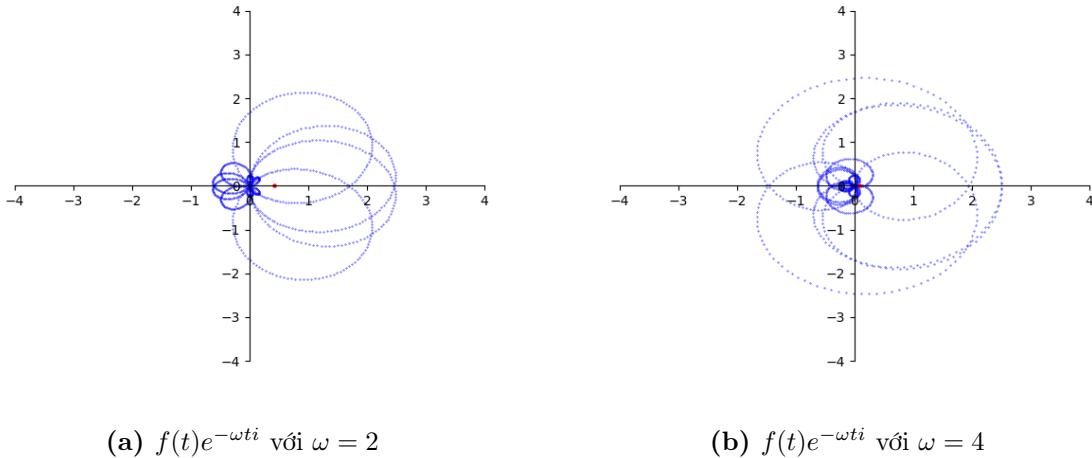
Trước hết ta sẽ thể hiện một cách trực quan công thức biến đổi Fourier khi áp dụng lên dữ liệu trước khi đi sâu vào bản chất của nó. Công thức trên có thể được chia làm hai phần $f(t)e^{-\omega t i}$ và phần tích phân trên miền thời gian trong khoảng $(-\infty, +\infty)$. Công thức của Fourier $e^{\phi i} = \cos(\phi) + i \sin(\phi)$ chính là biểu diễn cho một đường tròn có bán kính là 1, vậy nếu ta lấy $f(t)$ nhân với lại biểu thức này, việc đó ứng với việc uốn $f(t)$ từ hệ trục Cartesian Oft sang hệ trục phức hai chiều $O\Re S$. Để hiểu rõ hơn công thức này, chúng tôi lấy một ví dụ về việc xử lý Fourier trên dữ liệu cho trước. Cho một hàm theo biến thời gian $f(t) = \sin(t) + \sin(2t) + \sin(3t)$, Hình A.1 là đồ thị của hàm này trong hệ tọa độ Oft .



Hình A.1: Đồ thị hàm $f(t) = \sin(t) + \sin(2t) + \sin(3t)$

Bây giờ, ta sẽ thực hiện “cuốn” (wrap) hàm $f(t)$ bằng công thức $f(t)e^{-\omega t i}$ với $\omega = 4$. Chúng tôi thu được một đồ thị mới có hình dạng như Hình A.2. Sau khi đã thực hiện $f(t)e^{-\omega t i}$ chúng tôi lấy tích phân trên toàn bộ miền của t . Việc lấy tích phân này, dưới góc nhìn vật lý chính là đang đi tính khối tâm của đối tượng đang được xét. Cuối cùng, biến đổi Fourier tại ω trả về cho chúng tôi khối tâm hình học của $f(t)e^{-\omega t i}$.

Sinh viên thực hiện thử cho nhiều giá trị khác của ω và nhận thấy, ở một số vị trí, trọng tâm của vật trả về bởi công thức của Fourier thay đổi rất nhiều tùy vào giá trị ω mà ta sử dụng, Hình A.3 thể hiện giá trị biên độ (module) của giá trị



Hình A.2: Đồ thị hàm $f(t)$ sau khi nhân với $e^{-\omega t i}$, điểm đỏ là khói tâm của vật thể này, các giá trị trả về lúc này đều là số phức vì vậy không gian của chúng tôi sử dụng không còn là Oft nữa mà là $O\Re\Im$

phức được trả về của hàm dưới dấu tích phân ứng với các giá trị $\omega \in [-6, 6]$, biểu diễn của âm thanh dưới dạng này còn được gọi là **spectrum**.

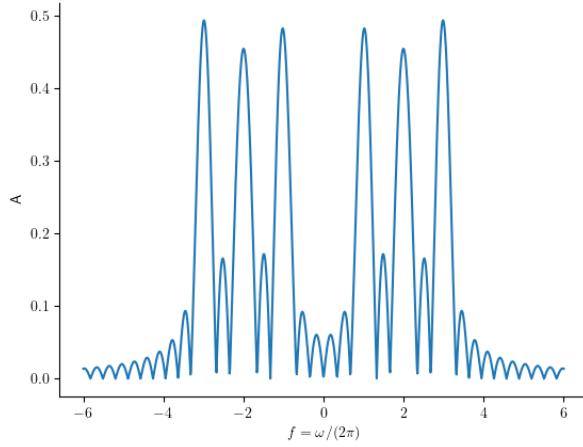
Có thể dễ dàng quan sát được một điều đặc biệt ở biến đổi Fourier, ở một số ω giá trị biên độ này rất lớn, và tương tự như vậy ở một số ω khác, giá trị biên độ trả về lại khá nhỏ. Các ω có biến đổi Fourier tại đó có giá trị biên độ lớn tương ứng với ω của các sóng cấu tạo nên $f(t)$ và ω của các sóng không có trong $f(t)$, giá trị biên độ thu được bởi biến đổi Fourier sẽ nhỏ hơn rất nhiều. Ta sẽ giải thích về hiện tượng đặc biệt này trong phần tiếp theo, đây cũng là điểm chính yếu khiến cho biến đổi Fourier trở nên rất quan trọng trong xử lý tín hiệu có dạng sóng.

A.1.2 Bản chất của biến đổi Fourier

Trước khi tiến hành phân tích bản chất của biến đổi Fourier, chúng tôi thu gọn phạm vi phân tích về lại một trường hợp đặc biệt của biến đổi Fourier là chuỗi Fourier. Chuỗi Fourier này được định nghĩa như sau

$$f(t) = \sum_{k=-\omega}^{\omega} c_k e^{kti}, \quad (\text{A.2})$$

với c_k là hệ số tương ứng với sóng thứ k , ω lúc này được giới hạn lại về miền số



Hình A.3: Đồ thị biến độ của trọng tâm (spectrum) trả về theo ω

nguyên dương \mathbb{Z}^+ . Bản chất của chuỗi Fourier này xuất phát từ định nghĩa của đa thức lượng giác (trigonometric polynomials). Đa thức lượng giác được định nghĩa trong [48, 49] có dạng như sau

$$f(t) = \frac{1}{2}a_0 + \sum_{k=1}^{\omega} (a_k \cos(kt) + b_k \sin(kt)), \quad (\text{A.3})$$

với các a_k, b_k là các hệ số được tính toán để xấp xỉ lại $f(t)$. Theo [48], các hàm được xấp xỉ bằng đa thức lượng giác cần phải thỏa mãn tính tuần hoàn sau từng chu kì T

$$f(t) = f(t + T). \quad (\text{A.4})$$

Với tính chất này, chọn $T = 2\pi$ ta kết luận được rằng [48, Theorem 13.1] với mọi $\epsilon \in \mathbb{R}$ sao cho $\epsilon > 0$, luôn tồn tại một đa thức lượng giác $p(t)$ sao cho

$$\|f(t) - p(t)\|_{\infty} \leq \epsilon. \quad (\text{A.5})$$

Kết luận này cũng dẫn tới sự hội tụ của đa thức lượng giác khi giá trị ϵ tiến về 0, hàm xấp xỉ $p(t)$ hoàn toàn có thể được chuyển sang dạng của chuỗi Fourier được định nghĩa như công thức (A.2) như sau

$$\begin{aligned}
 p(t) &= \frac{1}{2}a_0 + \sum_{k=1}^{\omega} (a_k \cos(kt) + b_k \sin(kt)) \\
 &= \frac{1}{2}a_0 + \sum_{k=1}^{\omega} \left(a_k \frac{e^{kti} + e^{-kti}}{2} + b_k \frac{e^{kti} - e^{-kti}}{2i} \right) \\
 &= \frac{1}{2}(a_0 + b_0) + \sum_{k=1}^{\omega} \frac{a_k + b_k}{2} e^{kti} + \sum_{k=1}^{\omega} \frac{a_k - b_k}{2} e^{-kti},
 \end{aligned}$$

với $c_0 = (a_0 + b_0)/2 = a_0/2$, $c_k = (a_k + b_k)/2$ và $c_{-k} = (a_k - b_k)/2$, từ đó thay vào công thức trên, ta thu được

$$\begin{aligned}
 p(t) &= \frac{1}{2}(a_0 + b_0) + \sum_{k=1}^{\omega} \frac{a_k + b_k}{2} e^{kti} + \sum_{k=1}^{\omega} \frac{a_k - b_k}{2} e^{-kti} \\
 &= \sum_{k=-\omega}^{\omega} c_k e^{kti}.
 \end{aligned} \tag{A.6}$$

Ta thu lại được công thức (A.6) từ định nghĩa của đa thức lượng giác tương tự như công thức chuỗi Fourier được định nghĩa ở (A.2). Tức là ta đã chứng minh cho khả năng xấp xỉ hàm của chuỗi Fourier [48, Theorem 13.1].

Cố gắng xấp xỉ lại $f(t)$ từ công thức đa thức lượng giác (A.3), sự xấp xỉ này đang cố gắng tối thiểu hóa lại hàm khoảng cách giữa hai hàm được xây dựng

$$d(f, p) = \int_{-\infty}^{+\infty} (f(t) - q(t))^2 dt, \tag{A.7}$$

với giả định rằng chu kì T của cả hai hàm này đều là 2π , người ta [48] đã thu hẹp cận của tích phân khoảng cách này về lại cận $[-\pi, \pi]$, do đó tích phân trên được đánh giá trong đoạn $[-\pi, \pi]$

$$d(f, p) = \int_{-\pi}^{\pi} (f(t) - q(t))^2 dt. \tag{A.8}$$

Dưới các điều kiện về tính trực giao giữa các hàm cosine và sine với nhau

$$\forall j, k \in \mathbb{Z}^+, j \neq k : \int_{-\pi}^{\pi} \sin(jt) \sin(kt) dx = 0, \tag{A.9}$$

$$\forall j, k \in \mathbb{Z}^+, j = k : \int_{-\pi}^{\pi} \sin(jt) \sin(kt) dx = 2\pi, \tag{A.10}$$

$$\forall j, k \in \mathbb{Z}^+, j \neq k : \int_{-\pi}^{\pi} \sin(jt) \sin(kt) dx = 0, \quad (\text{A.11})$$

$$\forall j, k \in \mathbb{Z}^+, j = k : \int_{-\pi}^{\pi} \sin(jt) \sin(jt) dx = 2\pi, \quad (\text{A.12})$$

$$\forall j, k \in \mathbb{Z}^+ : \int_{-\pi}^{\pi} \cos(jt) \sin(kt) dx = 0, \quad (\text{A.13})$$

ta có thể chứng minh [48, Theorem 13.2] để đa thức lượng giác ở công thức (A.3) đạt cực tiểu hàm khoảng cách giữa hai hàm (A.7) so với $f(t)$ khi và chỉ khi các hệ số a_k và b_k được tính như sau

$$a_j = \frac{1}{\pi} \int_{-\pi}^{\pi} f(t) \cos(jt) dt, \quad (\text{A.14})$$

$$b_j = \frac{1}{\pi} \int_{-\pi}^{\pi} f(t) \sin(jt) dt, \quad (\text{A.15})$$

với $j \in \mathbb{Z}^+$. Rõ ràng với các giá trị a_k và b_k được định nghĩa như trên, ta đang tạo ra một hệ trục tọa độ mới với các giá trị a_k và b_k tương ứng với các phép chiếu của $f(t)$ lần lượt lên các trục tọa độ được cấu tạo bởi $\cos(jt)$ và $\sin(jt)$. Nhờ vào điều này, ở các giá trị của ω cấu tạo nên sóng $f(t)$ sẽ đạt cực đại tương ứng với các giá trị $\omega = j$ với giá trị đúng bằng hình chiếu của chúng lên $\cos(jt)$ tương ứng.

Ví dụ. Để thấy rõ được tác dụng của phép chiếu này trên $f(t)$, ta xét một trường hợp cụ thể khi $f(t) = r \cos(jt + \phi)$, khi đó thực hiện tính toán các hệ số a_k và b_k , ta thu được

$$\begin{aligned} f'(t) &= \frac{1}{2} a_0 + \sum_{k=1}^{\omega} (a_k \cos(kt) + b_k \sin(kt)) \\ &= a_j \cos(jt) + b_j \sin(jt), \end{aligned} \quad (\text{A.16})$$

với a_j và b_j được tính như sau

$$\begin{aligned}
 a_j &= \frac{1}{\pi} \int_{-\pi}^{\pi} f(t) \cos(jt) dt \\
 &= \frac{1}{\pi} \int_{-\pi}^{\pi} r \cos(jt + \phi) \cos(jt) dt \\
 &= \frac{r}{\pi} \int_{-\pi}^{\pi} \cos(jt) \cos(jt) \cos(\phi) + \sin(jt) \cos(jt) \sin(\phi) dt \\
 &= r \cos(\phi),
 \end{aligned} \tag{A.17}$$

$$\begin{aligned}
 b_j &= \frac{1}{\pi} \int_{-\pi}^{\pi} f(t) \sin(jt) dt \\
 &= \frac{1}{\pi} \int_{-\pi}^{\pi} r \cos(jt + \phi) \sin(jt) dt \\
 &= \frac{r}{\pi} \int_{-\pi}^{\pi} \cos(jt) \sin(jt) \cos(\phi) + \sin(jt) \sin(jt) \sin(\phi) dt \\
 &= r \sin(\phi).
 \end{aligned} \tag{A.18}$$

Từ đó, (A.16) trở thành

$$\begin{aligned}
 f'(t) &= a_j \cos(jt) + b_j \sin(jt) \\
 &= r \cos(\phi) \cos(jt) + r \sin(\phi) \sin(jt) \\
 &= r \cos(jt + \phi) \\
 &= f(t).
 \end{aligned} \tag{A.19}$$

Công thức $f'(t)$ thu lại được đúng bằng với giá trị của $f(t)$ ban đầu và giá trị được ta ghi nhận được ở Hình A.3 bởi công thức Fourier chính là chuẩn Euclid của vector do $f(t)$ chiếu vào hệ trục này. Ta khai triển tiếp tục

$$a_j = r \cos(\phi) = r \left(\frac{e^{\phi i} + e^{-\phi i}}{2} \right), \tag{A.20}$$

$$b_j = r \sin(\phi) = r \left(\frac{e^{\phi i} - e^{-\phi i}}{2i} \right), \tag{A.21}$$

và thay các giá trị a_j và b_j mới tìm được này vào công thức c_j và c_{-j} để thu được

$$\begin{aligned}
 c_j &= a_j + b_j \\
 &= r \left(\frac{e^{\phi i} + e^{-\phi i}}{2} - \frac{e^{\phi i} - e^{-\phi i}}{2} \right) \\
 &= re^{-\phi i},
 \end{aligned} \tag{A.22}$$

$$\begin{aligned}
 c_{-j} &= a_j + b_j \\
 &= r \left(\frac{e^{\phi i} + e^{-\phi i}}{2} + \frac{e^{\phi i} - e^{-\phi i}}{2} \right) \\
 &= re^{\phi i}.
 \end{aligned} \tag{A.23}$$

Thông qua hai công thức của c_j và c_{-j} này, ta nhận thấy biên độ của hai tần số đối xứng nhau j và $-j$ sẽ có giá trị ngang nhau nhưng pha của j sẽ bị ngược với pha của tần số ban đầu trong $f(t)$.

A.1.3 Phép biến đổi Fourier ngược

Biến đổi Fourier ngoài việc cung cấp khả năng quan sát các thành phần sóng cấu tạo nên âm thanh ở miền tần số, còn cho phép chỉnh sửa và chuyển đổi ngược từ miền tần số về lại miền thời gian ban đầu. Như phần trước chúng tôi đã giới thiệu về biến đổi Fourier (thuận) liên tục, trong phần này ta sẽ tìm hiểu về phép biến đổi Fourier nghịch liên tục chuyển đổi dữ liệu từ miền tần số về miền thời gian.

Phép biến đổi Fourier ngược của $F(\omega)$ được định nghĩa như sau

$$f(t) = \int_{-\infty}^{\infty} F(\omega) e^{\omega t i} d\omega. \tag{A.24}$$

Công thức trên của biến đổi Fourier nghịch đơn thuần chỉ là sự tổng hợp của rất nhiều sóng với các bước sóng khác nhau. Vì trong luận văn này, chúng tôi đang xét tới âm thanh là một tập hợp của rất nhiều sóng khác nhau, do đó chúng tôi đặt giả thiết về $f(t)$ trong luận văn này để chứng minh các tính chất của $F(\omega)$ như sau

$$f(t) = \sum_j r_j \cos(\omega_j t + \phi_j). \tag{A.25}$$

Đây là một tổng hữu hạn các sóng có bước sóng ω_j , biên độ r_j và pha ban đầu ϕ_j . Chúng tôi chuyển đổi công thức trên sang dạng của công thức biến đổi Fourier nghịch

$$\begin{aligned} f(t) &= \sum_j r_j \cos(\omega_j t + \phi_j) \\ &= \Re \left(\sum_j r_j e^{(\omega_j t + \phi_j)i} \right) \\ &= \Re \left(\sum_j r_j e^{\phi_j i} e^{\omega_j t i} \right). \end{aligned} \quad (\text{A.26})$$

Bằng cách tách ra như vậy, với mỗi sóng thứ j , chúng tôi thu được một biểu thức $r_j e^{\phi_j i}$ đại diện cho pha ban đầu và biên độ của sóng. Sự tương đồng giữa hai công thức biến đổi Fourier nghịch và công thức (A.26) xuất hiện khi chúng tôi thay vì giả định $f(t)$ là tổng bao gồm một số các sóng rời rạc thì $f(t)$ lúc này sẽ được biểu diễn bằng tích phân của tất cả các bước sóng có trong $(-\infty, +\infty)$. Vậy nên, ta có

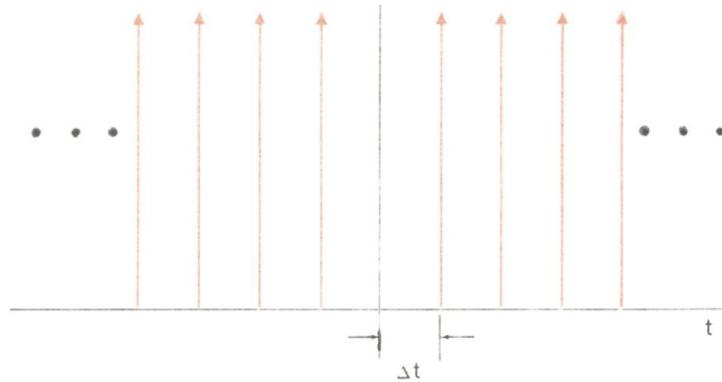
$$\begin{aligned} f(t) &= \Re \left(\int_{-\infty}^{\infty} r_j e^{\phi_j i} e^{\omega t i} d\omega \right) \\ &= \Re \left(\int_{-\infty}^{\infty} F(\omega) e^{\omega t i} d\omega \right), \end{aligned} \quad (\text{A.27})$$

và với việc chứng minh $F(\omega_j) = r_j e^{\phi_j i}$ thì ta thu lại được công thức biến đổi Fourier nghịch như đã nêu ở ban đầu. Bằng cách chuyển đổi qua lại giữa hai miền, tần số và thời gian, các bước sóng tổng hợp có thể được phân tách thành rất nhiều sóng đơn có bước sóng khác nhau cũng như từ một số sóng đơn cho trước. Sự phân tích này rất hữu ích trong việc loại bỏ các bước sóng nhiễu ra khỏi âm thanh gốc ban đầu.

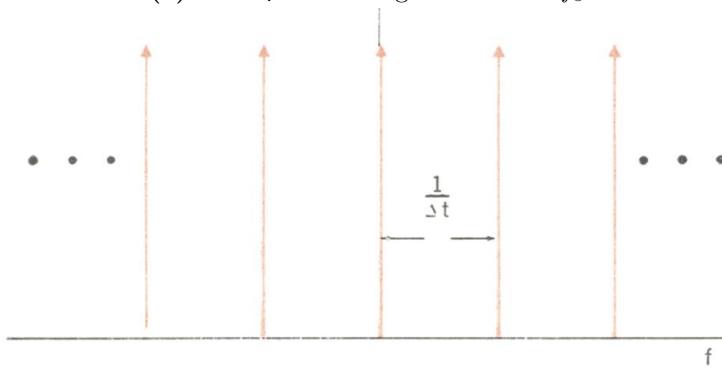
A.2 Biến đổi Fourier rời rạc

Trong phần này, ta sẽ bàn về cách chuyển đổi tích phân Fourier về dạng Fourier rời rạc được đề cập trong Chương 2. Xuất phát từ định nghĩa của biến đổi Fourier liên tục

$$X(\omega) = \int_{-\infty}^{+\infty} x(t)e^{-\omega t i} dt.$$



(a) Đồ thị theo thời gian của hàm f_s



(b) Đồ thị theo tần số của hàm F_s

Hình A.4: Đồ thị theo thời gian của f_s và đồ thị theo tần số của F_s (nguồn [8])

Brigham [8] rời rạc hóa $f(t)$ bằng cách nhân giữa $f(t)$ với $f_s(t)$ có đồ thị theo thời gian như Hình A.4a. Biến đổi Fourier của tích của hai hàm theo thời gian có thể được quy đổi lại thành tích chập của hai biến đổi Fourier giữa chúng, điều này có thể được chứng minh như sau

$$\begin{aligned}
 \mathcal{F}\{f(t)f_s(t)\}(\omega) &= \int_{-\infty}^{+\infty} f(t)f_s(t)e^{-\omega t i} dt \\
 &= \int_{-\infty}^{+\infty} \left(\int_{-\infty}^{+\infty} F(\omega') e^{\omega' t i} d\omega' \right) f_s(t) e^{-\omega t i} dt \\
 &= \int_{-\infty}^{+\infty} \left(\int_{-\infty}^{+\infty} f_s(t) e^{-(\omega - \omega') t i} dt \right) F(\omega') d\omega' \\
 &= \int_{-\infty}^{+\infty} F_s(\omega - \omega') F(\omega') d\omega' \\
 &= (F_s * F)(\omega).
 \end{aligned} \tag{A.28}$$

Qua công thức trên, người ta [8] đã chứng minh được rằng spectrum được tạo ra bởi tích chập giữa F và F_s sẽ có tính tuần hoàn với chu kỳ là $1/\Delta t$ với Δt là khoảng thời gian lấy mẫu. Từ đó, ta suy ra được công thức rời rạc của biến đổi Fourier liên tục như sau

$$X(f) = \sum_{k=0}^{N-1} x_k e^{-2\pi f k \Delta t i} \Delta t. \tag{A.29}$$

Ta lại tiếp tục đổi biến số của công thức $X(f)$ một lần nữa, lần này biến f được biến đổi dựa vào định lý lấy mẫu của Nyquist (Nyquist-Shannon sampling theorem) [5]. Theo định lý Nyquist, tần số được truyền đi sẽ không bao giờ vượt quá được một nửa tần số truyền đi. Do đó, tần số sóng tồn tại bên trong $x(t)$ sau khi đã chịu ảnh hưởng bởi quá trình rời rạc hóa sẽ không bao giờ vượt quá được một nửa tần số lấy mẫu của sóng này, vậy nên $-f_0/2 \leq f \leq f_0/2$ với f_0 là tần số truyền tải hay tần số lấy mẫu theo định lý của Nyquist. Ta bắt đầu tiến hành rời rạc hóa f trong khoảng $[-f_0/2, f_0/2]$, bằng cách chia khoảng nhỏ này thành N đoạn có kích thước Δf , N có thể xem là số lượng mẫu được lấy từ $x(t)$, do vậy ta có $f = j\Delta f$ với $j \in \mathbb{Z}^+$. Ta lại có sự liên hệ giữa Δt , f_0 và Δf như sau

$$\begin{cases} f_0 = \frac{1}{\Delta t}, \\ \Delta t = \frac{T}{N}, \\ \Delta f = \frac{f_0}{N} = \frac{1}{T}. \end{cases} \tag{A.30}$$

Thay các công thức (A.30) vào công thức (A.29) chúng tôi thu được

$$\begin{aligned}
 X(f) &= \sum_{k=0}^{N-1} x_k e^{-2\pi f k \Delta t i} \Delta t \\
 &= \sum_{k=0}^{N-1} x_k e^{-2\pi j k \Delta f \Delta t i} \Delta t \\
 &= \sum_{k=0}^{N-1} x_k e^{-2\pi j k \frac{1}{T} \frac{T}{N} i} \frac{T}{N} \\
 &= \sum_{k=0}^{N-1} x_k e^{-2\pi \frac{j k}{N} i} \frac{T}{N}.
 \end{aligned} \tag{A.31}$$

Và vì T/N là hằng số với mọi j, k nên ta có thể bỏ nó ra khỏi công thức $X(f)$ (lúc này đã chuyển thành $X(j)$) để tạo thành công thức chính của biến đổi Fourier rồi rắc

$$X(j) = \sum_{k=0}^{N-1} x_k e^{-2\pi \frac{j k}{N} i} = \sum_{k=0}^{N-1} x_k W^{jk}. \tag{A.32}$$