ĐẠI HỌC BÁCH KHOA HÀ NỘI

ĐỒ ÁN TỐT NGHIỆP

NGHIÊN CỨU ỨNG DỤNG MẠNG NƠ-RON LSTM TRONG NHẬN DẠNG CÂU LỆNH ĐIỀU KHIỂN ĐIỀU HOÀ BẰNG TIẾNG NÓI

LÊ ĐÌNH KHÁNH
Khanh.LD191905@sis.hust.edu.vn

Ngành KT Điều khiển & Tự động hóa

Giảng viên hướng dẫn: TS. Trần Thị Anh Xuân

Chữ ký của GVHD

Khoa: Tự động hóa
Trường: Điện – Điện tử

HÀ NỘI, 2/2024

BỘ GIÁO DỤC & ĐÀO TẠO
ĐH BÁCH KHOA HÀ NỘI

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập – Tự do - Hạnh phúc

NHIỆM VỤ
ĐỒ ÁN TỐT NGHIỆP

Họ và tên sinh viên: Lê Đình Khánh

Khóa: 64 Trường: Điện- Điện tử Ngành: KT ĐK &TĐH

1. *Tên đề tài:*

Nghiên cứu ứng dụng mạng nơ-ron LSTM trong nhận dạng câu lệnh điều khiển điều hoà bằng tiếng nói.

2. *Nội dung đề tài:*

Đề tài "Nghiên cứu ứng dụng mạng nơ-ron LSTM trong nhận dạng câu lệnh điều khiển điều hoà bằng tiếng nói" với các nội dung như sau:

- Tìm hiểu về xử lý tiếng nói và đặc trưng MFCC.
- Tìm hiểu về Học Sâu và mạng nơ-ron LSTM.
- Xây dựng mô hình nhận dạng tiếng nói điều khiển điều hoà bằng tiếng nói.
- Thử nghiệm và đánh giá kết quả.

3. *Thời gian giao đề tài:*10/2023.
4. *Thời gian hoàn thành:* 01/2024.

Ngày ... tháng ... năm 2024

CÁN BỘ HƯỚNG DẪN

Lời cảm ơn

Đầu tiên, tôi xin bày tỏ lòng biết ơn sâu sắc với các thầy cô giáo ở Đại học Bách Khoa Hà Nội đã tạo điều kiện cho tôi được học tập và phát triển trong một môi trường chuyên nghiệp, năng động và đầy sáng tạo. Trong quá trình được rèn luyện tại đây, tôi đã học tập được nhiều kiến thức và kỹ năng, cả về chuyên môn lẫn các kỹ năng mềm áp dụng trong cuộc sống. Tôi xin gửi lời cảm ơn chân thành tới cô giáo TS. Trần Thị Anh Xuân - người đã đồng hành cùng tôi thực hiện đề tài đồ án tốt nghiệp trong suốt khoảng thời gian vừa qua. Cô đã cho tôi những góp ý, lời khuyên và hỗ trợ tôi rất nhiều trong quá trình thực hiện đề tài. Tôi cũng xin cảm ơn gia đình, bạn bè và các anh chị đã giúp đỡ tôi trong quá trình thu thập bộ dữ liệu phục vụ đề tài. Sau một thời gian nghiên cứu, tôi đã xây dựng được mô hình nhận dạng câu lệnh theo mục tiêu đề tài. Tuy nhiên, do giới hạn về thời gian và kiến thức, tôi có thể không tránh khỏi những thiếu sót trong đề tài của mình. Vì vậy, tôi rất mong nhận được những ý kiến đóng góp quý báu từ mọi người để tôi có thể hoàn thiện hơn đề tài của mình. Một lần nữa, tôi muốn bày tỏ lòng biết ơn sâu sắc đến tất cả những người đã giúp đỡ và hỗ trợ tôi trong suốt quá trình này.

Tóm tắt nội dung đồ án

Với nội dung đề tài của mình, tôi cần thực hiện: xây dựng được mô hình ứng dụng mạng nơ-ron LSTM để nhận dạng câu lệnh điều khiển điều hoà bằng tiếng nói. Cụ thể hơn, tôi sẽ tiến hành xây dựng mô hình để phân biệt 5 câu lệnh: bật điều hoà, tắt điều hoà, tăng 1 độ, giảm 1 độ, bật 26 độ với nhiễu là âm thanh thường có trong căn phòng. Từ yêu cầu này, tôi sẽ tiến hành thu thập cơ sở dữ liệu gồm các câu lệnh trên và nhiễu; sau đó xây dựng mô hình ứng dụng mạng nơ-ron LSTM để phân biệt các nhãn này; khi mô hình cho kết quả tốt thì sẽ tiến hành thử nghiệm thực tế. Quá trình thu thập dữ liệu sẽ sử dụng phần mềm Audacity với thiết bị thu âm là micrô của máy tính; quá trình huấn luyện và thử nghiệm sẽ tiến hành trên 2 dịch vụ máy tính chủ yếu là Google Colab và Jupyter Notebook với khung phát triển (framework) là TensorFlow. Sau quá trình nghiên cứu và thực hiện đề tài, tôi đã xây dựng được mô hình phù hợp với yêu cầu đề ra, tiến hành thử nghiệm trong môi trường thực của căn phòng khá tốt, có thể phát triển để sử dụng vào các ứng dụng IoT - mô hình vạn vật kết nối, nhà thông minh… Với đề tài này, trong tương lai, tôi định hướng sẽ phát triển để nhận dạng thêm nhiều câu lệnh hơn nữa, đa dạng hoá các câu lệnh để có thể điều khiển điều hoà, phát triển thành một sản phẩm hoàn chỉnh để có thể phục vụ con người. Qua quá trình nghiên cứu, tôi đã được học và hiểu hơn về cách hình thành và cấu tạo của ngôn ngữ, cách hoạt động của một mô hình mạng nơ-ron nhân tạo, cũng như cách để tạo ra một tập dữ liệu âm thanh tốt nhất.

Sinh viên thực hiện
Ký và ghi rõ họ tên

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI

1.1 Nhận dạng tiếng nói và ứng dụng

Trong thời đại số hóa ngày nay, việc nhận dạng giọng nói đã trở thành một trong những xu hướng quan trọng nhất, được thúc đẩy bởi sự phát triển của Học Sâu (Deep Learning), Internet of Things (IoT) và xu hướng điều khiển thiết bị không chạm. Nhận dạng giọng nói không chỉ mang lại tiện ích và trải nghiệm người dùng tốt hơn mà còn mở ra nhiều cơ hội mới cho các lĩnh vực như tài chính, y tế, giao thông vận tải và giáo dục.

Hình 1.1 Ứng dụng của nhận dạng giọng nói

Theo báo cáo từ tổ chức nghiên cứu thị trường và công nghệ Gartner, từ năm 2019 đến năm 2022, số lượng người dùng dịch vụ nhận dạng giọng nói đã tăng mạnh mẽ từ khoảng 50 triệu người lên đến hơn 200 triệu người trên toàn cầu.

Ở Việt Nam, cũng không ngoại lệ, xu hướng nhận dạng giọng nói cũng đang trở nên phổ biến hơn. Theo số liệu thống kê, số lượng người dùng điện thoại thông minh tại Việt Nam đã tăng mạnh trong những năm gần đây, năm 2023 rơi vào khoảng 63.8 triệu người dùng [1]. Với sự gia tăng này, các ứng dụng nhận dạng giọng nói đã thu hút sự chú ý đặc biệt. Dự kiến trong giai đoạn tiếp theo, sự phát triển của IoT và xu hướng điều khiển thiết bị không chạm cũng sẽ đóng một vai trò quan trọng trong việc thúc đẩy sự phát triển của công nghệ nhận dạng giọng nói ở Việt Nam.

1.2 Các thách thức của đề tài

Việc xây dựng mô hình nhận dạng giọng nói ở Việt Nam đang đối diện với một số hạn chế đáng chú ý, chủ yếu là từ nguồn cơ sở dữ liệu âm thanh, bao gồm:

- Thiếu cơ sở dữ liệu đa dạng: một trong những hạn chế chính là sự thiếu hụt cơ sở dữ liệu đủ lớn và đa dạng. Cơ sở dữ liệu giọng nói đa phương tiện với các biến thể về giọng, ngữ điệu và phong cách nói của người Việt chưa được phát triển đầy đủ. Điều này gây khó khăn cho việc huấn luyện mô hình nhận dạng giọng nói để có thể nhận dạng chính xác các biến thể giọng nói khác nhau.

- Sự đa dạng về vùng miền: ở Việt Nam, có sự đa dạng về giọng miền, từ giọng miền Bắc, miền Trung đến giọng miền Nam. Mỗi vùng miền lại có đặc điểm riêng về ngữ điệu, âm điệu và phát âm, gây ra sự biến động lớn trong dữ liệu giọng nói và làm tăng độ khó trong việc nhận dạng chính xác.
- Tác động từ nhiễu: môi trường sống và làm việc ở Việt Nam thường đầy ồn ào và nhiễu loạn, từ tiếng động giao thông đến tiếng động xã hội. Những yếu tố nhiễu này có thể ảnh hưởng đến chất lượng của tín hiệu âm thanh, làm giảm độ chính xác của quá trình nhận dạng giọng nói.

1.3 Mục tiêu và phạm vi của đề tài

Từ những xu thế cũng như thách thức trên, tôi đã lựa chọn đề tài "Nghiên cứu ứng dụng mạng nơ-ron LSTM trong nhận dạng câu lệnh điều khiển điều hoà bằng tiếng nói". Với mục tiêu xây dựng được một mô hình có thể nhận dạng được một số câu lệnh thông dụng để điều khiển điều hoà: bật điều hoà, tắt điều hoà, tăng 1 độ, giảm 1 độ, bật 26 độ. (Đây là những chức năng cơ bản và phổ biến của điều khiển điều hoà, ngưỡng 26 độ là ngưỡng thường được bật nhất).

Trong phạm vi nghiên cứu, tôi tiến hành triển khai trong môi trường tiếng nói sạch, rất ít nhiễu; nghiên cứu chạy thử nghiệm trên máy tính để đánh giá chất lượng của mô hình đề xuất.

1.4 Bố cục đồ án

Từ những thách thức, mục tiêu cũng như phạm vi của đề tài như trên, tôi đã nghiên cứu và xây dựng bố cục đồ án với các nội dung chính như sau:

- Cơ sở lý thuyết về xử lý tiếng nói: tìm hiểu về kiến thức nền tảng của xử lý tiếng nói và đặc trưng MFCC sử dụng trong mô hình.
- Cơ sở lý thuyết về học sâu: tìm hiểu về học sâu và một số loại mạng nơ-ron nhân tạo, lựa chọn loại mạng nơ-ron sử dụng trong đề tài.
- Xây dựng mô hình nhận dạng tiếng nói: đề xuất xây dựng cơ sở dữ liệu, tăng cường dữ liệu âm thanh và đưa ra mô hình nhận dạng từ kích hoạt, nhận dạng câu lệnh.
- Thử nghiệm và đánh giá kết quả: phân chia cơ sở dữ liệu và kết quả thử nghiệm.
- Kết luận và hướng phát triển: so sánh kết quả đã đạt được so với mục tiêu của đề tài đề ra, rút ra các khó khăn và nhược điểm, từ đó đề xuất hướng phát triển đề tài.

1.5 Kết luận chương

Như vậy, ở chương này chúng ta tìm hiểu được về ứng dụng cũng như thách thức của nhận dạng tiếng nói. Qua đó, nắm được mục tiêu và phạm vi đề tài "Nghiên cứu ứng dụng mạng nơ-ron LSTM trong nhận dạng câu lệnh điều khiển điều hoà bằng tiếng nói", xây dựng được bố cục của đồ án, tiếp theo, tôi sẽ giới thiệu về cơ sở lý thuyết về xử lý tiếng nói.

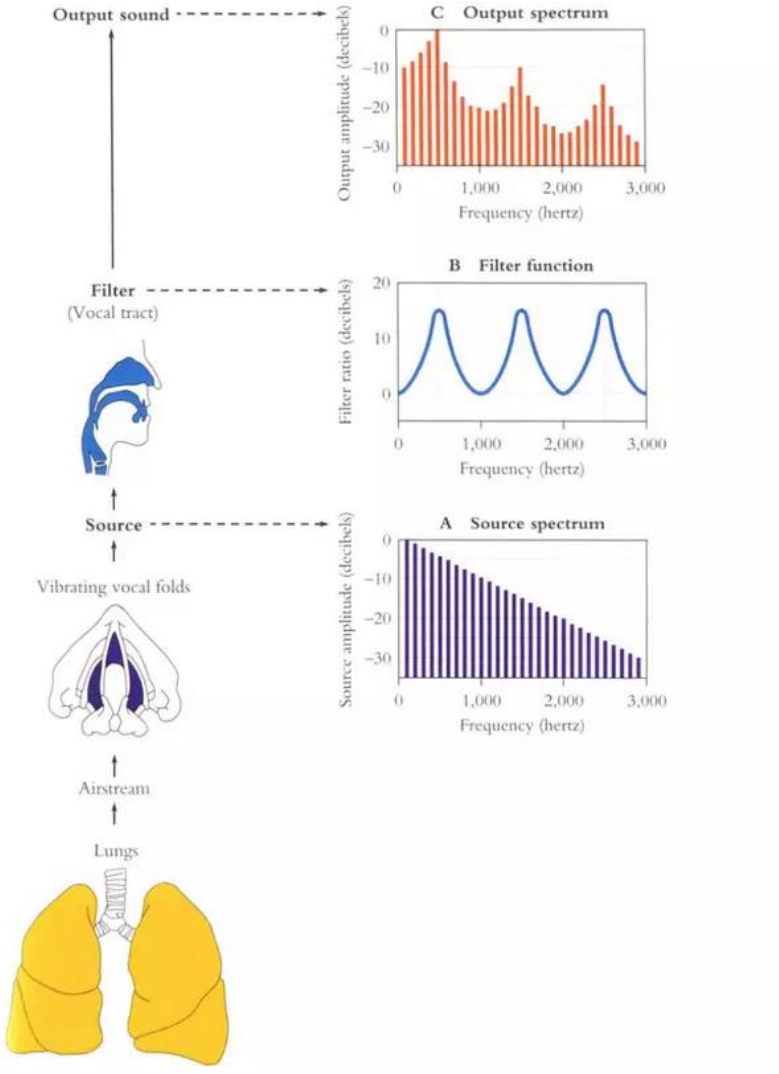
CHƯƠNG 2. CƠ SỞ LÝ THUYẾT VỀ XỬ LÝ TIẾNG NÓI

2.1 Kiến thức nền tảng về xử lý tiếng nói

2.1.1 Nguyên lý hình thành tiếng nói

Đầu tiên, một luồng hơi được đẩy từ phổi tạo áp lực lên thanh quản. Dưới áp lực đó, thanh quản mở ra để luồng không khí thoát qua, áp lực giảm xuống khiến thanh quản tự động đóng lại. Việc đóng lại như vậy lại khiến áp lực tăng lên và quá trình trên tái diễn. Các chu kì đóng mở thanh quản này liên tục tái diễn, tạo ra các tần số sóng âm với tần số cơ bản khoảng 125Hz đối với nam, 210Hz đối với nữ. Đó cũng là lý do giọng nữ thường có xu hướng cao hơn giọng nam. Tần số này gọi là tần số cơ bản F0 [2].

Như vậy thanh quản đã tạo ra các tần số sóng âm cơ bản. Tuy nhiên để hình thành nên tiếng nói cần các cơ quan khác như: vòm họng, khoang miệng, lưỡi, răng, môi, mũi… Các cơ quan này hoạt động như một bộ "cộng hưởng", giống hộp đàn ghi-ta, nhưng có khả năng thay đổi hình dạng linh hoạt. Bộ cộng hưởng này có tác dụng khuếch đại 1 vài tần số, triệt tiêu một vài tần số khác để tạo ra âm thanh. Khả năng thay đổi hình dạng linh hoạt của nó giúp tạo ra các âm thanh khác nhau để hình thành lên tiếng nói.



Hình 2.1 Cơ chế hình thành tiếng nói ở người

Hình 2.1 mô tả chi tiết hơn về cơ chế này: với đầu vào là luồng không khí từ phổi, qua thanh quản sẽ tạo ra các tần số sóng âm, sau đó các cơ quan có chức năng như bộ lọc tạo thành đầu ra là âm thanh tiếng nói.

Tại phổ của giọng nói ở đầu ra, ta thấy có 3 đỉnh, 3 đỉnh này thường được gọi là các "formant" F1, F2, F3. Trong nhận dạng tiếng nói truyền thống, các thông tin của những "formant" này thường được tách ra khỏi F0 rồi mới sử dụng để nhận dạng giọng nói.

2.1.2 Âm tố, âm vị, âm tiết

a) Âm tố

Khi phát âm các âm tiết "tan" và "lan" có sự khác nhau. Sự khác nhau ở đây rõ ràng là do "t" và "l" gây ra. Như vậy có thể phân tích âm tiết thành những yếu tố nhỏ hơn, "tan" do 3 âm "t", "a", "n" phối hợp thành, và "lan" do 3 âm "l", "a", "n" phối hợp thành. Người ta gọi các yếu tố vừa tách ra khỏi 2 âm tiết trên là âm tố. Âm tố được ghi vào giữa hai kí hiệu [], ví dụ: âm tố [a], [b], [c] ...

Âm tố là đơn vị ngữ âm nhỏ nhất trong lời nói. Một âm tố "a" ở ba người nói sẽ có ba cách phát âm khác nhau. Thậm chí, một người khi phát âm "a" ở ba thời điểm phát âm khác nhau, thì âm "a" khi phát ra cũng không hoàn toàn giống nhau. Đứng về mặt phát âm, tiếng nói có vô số âm tố khác nhau. Có 3 loại âm tố là nguyên âm, phụ âm, bán âm (bán nguyên âm hay bán phụ âm).

- Nguyên âm có đặc điểm là khi phát âm không bị luồng hơi cản lại, ví dụ âm a, u, i, e, o…
- Phụ âm có đặc điểm là khi phát âm thì luồng hơi bị cản lại, ví dụ âm p, b, t, m, n…
- Bán âm có đặc điểm giống nguyên âm về mặt cấu tạo, và giống phụ âm về mặt chức năng (nên còn được gọi là bán nguyên âm hay bán phụ âm), ví dụ /u/ (ngắn), /i/ (ngắn).

b) Âm vị

Như đã nói ở phần âm tố, cách phát âm một âm "a" của mỗi người và ngay ở một người, trong những thời điểm khác nhau, cũng không hoàn toàn như nhau. Và do đó, tiếng nói có vô số âm cụ thể của "a". Dựa vào những nét chung nhất, người ta quy nó về một đơn vị khu biệt, có chức năng phân biệt nghĩa, gọi là âm vị.

Âm vị trong tiếng Việt là đơn vị ngữ âm nhỏ nhất có chức năng khu biệt nghĩa. Nếu số lượng âm tố là vô số, thì số lượng âm vị là có hạn, khoảng vài chục đơn vị trong một ngôn ngữ. Để phân biệt với âm tố, âm vị được ghi ở giữa hai kí hiệu //, ví dụ: âm vị /a/, /u/, /o/.

c) Âm tiết (tiếng)

- Khái niệm

Lời nói của con người là một chuỗi âm thanh được phát ra kế tiếp nhau trong không gian và thời gian. Việc phân tích chuỗi âm thanh ấy người ta nhận ra được các đơn vị của ngữ âm. Với câu nói "Hà Nội mùa này vắng những cơn mưa", người nghe sẽ nghe được những khúc đoạn tự nhiên trong chuỗi lời nói đó như sau:

Hà / Nội / mùa / này / vắng / những / cơn / mưa

Những khúc đoạn âm thanh này không thể chia nhỏ hơn được nữa cho dù có cố tình phát âm thật chậm, thật tách biệt. Điều đó chứng tỏ rằng, đây là những khúc đoạn âm thanh tự nhiên nhỏ nhất khi phát âm, và được gọi là âm tiết. Với tiếng Anh và nhiều ngôn ngữ khác, 1 từ được ghép bởi nhiều âm tiết. Ví dụ từ "want" có 1 âm tiết, "wanna" có 2 âm tiếng, "computer" có 3 âm tiết .... Trong khi đó, trong tiếng Việt, một âm tiết bao giờ cũng được phát ra với một thanh điệu, và tách rời với âm tiết khác. Vì vậy, việc nhận ra âm tiết trong tiếng Việt là dễ dàng hơn nhiều so với các ngôn ngữ Ấn Âu. Trên chữ viết, mỗi âm tiết tiếng Việt được ghi thành một "chữ".

- Cấu tạo

Trong ngữ cảm của người Việt, âm tiết tuy được phát âm liền một hơi, nhưng không phải là một khối bất biến mà có cấu tạo lắp ghép. Khối lắp ghép ấy có thể tháo rời từng bộ phận của âm tiết này để hoán vị với bộ phận tương ứng của ở âm tiết khác. Ví dụ:

+ Tiền đâu - đầu tiên (đảo trật tự âm tiết và hoán vị thanh điệu "`").

+ Hiện đại - hại điện (hoán vị phần sau "iên" cho "ai").

+ Nhỉ đay - nhảy đi (thanh điệu giữ nguyên vị trí cùng với phần đầu "nh" và "đ").

Từ ví dụ trên có thể thấy âm tiết tiếng Việt có 3 bộ phận: thanh điệu, phần đầu và phần sau.

+ Thanh điệu là một yếu tố thể hiện độ cao và sự chuyển biến của độ cao trong mỗi âm tiết, tiếng Việt có các thanh điệu: huyền, hỏi, sắc, ngã, nặng và không dấu.

+ Phần đầu của âm tiết được xác định là âm đầu, vì ở vị trí này chỉ có một âm vị tham gia cấu tạo.

+ Phần sau của âm tiết được gọi là phần vần. Người Việt chưa biết chữ không cảm nhận được cấu tạo của phần vần. Vào lớp 1, trẻ em bắt đầu "đánh vần", tức là phân tích, tổng hợp các yếu tố tạo nên vần, rồi ghép với âm đầu để nhận ra âm tiết. Ví dụ:

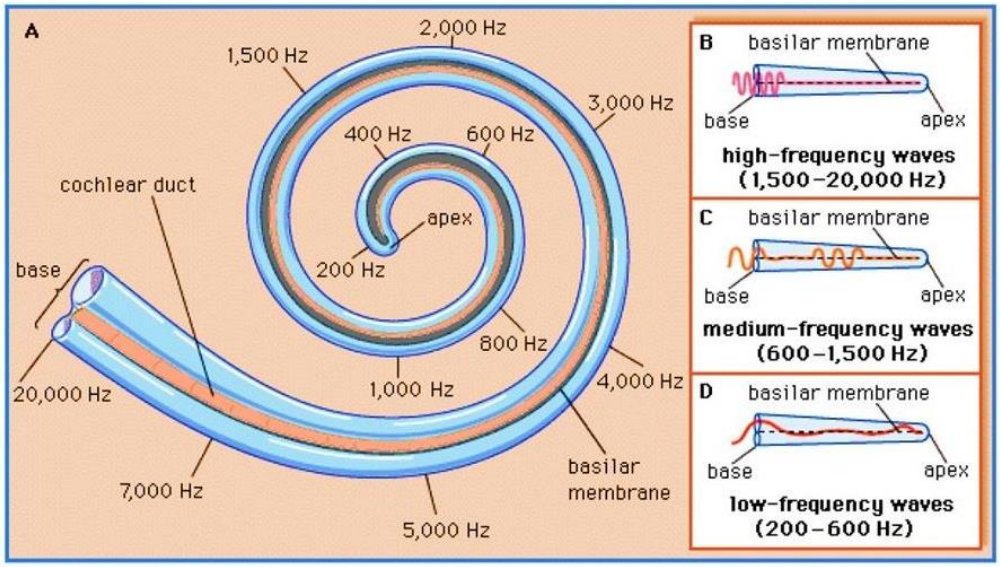
U + Â + N = UÂN, X + UÂN = XUÂN

Các âm đầu vần, giữa vần và cuối vần (u, â, n) được gọi là âm đệm, âm chính và âm cuối.

2.1.3 Cơ chế hoạt động của tai

Âm thanh, tiếng nói mà chúng ta vẫn nghe hằng ngày là sự pha trộn của rất nhiều sóng với các tần số khác nhau. Khoảng tần số này thường nằm từ 20Hz đến 20000Hz. Tuy nhiên tai người (và các loài động vật) hoạt động phi tuyến tính, tức không phải rằng độ cảm nhận âm thanh 20000Hz sẽ gấp 1000 lần âm thanh 20Hz. Thường thì tai người rất nhạy cảm ở âm thanh tần số thấp, kém nhạy cảm ở tần số cao.

Âm thanh được truyền đến tai và va chạm vào màng nhĩ, khiến cho màng nhĩ bắt đầu rung lên. Rung động này được truyền qua ba xương nhỏ trước khi đến ốc tai. Ốc tai, có hình dạng xoắn và rỗng như một con ốc, chứa các dịch nhầy giúp truyền âm thanh. Trên bề mặt của ốc tai, các tế bào lông phát hiện âm thanh và chuyển đổi chúng thành tín hiệu gửi đến não bộ. Các tế bào lông ở phần đầu của ốc tai thường cứng hơn và phản ứng với các tần số cao hơn. Ngược lại, các tế bào lông ở phần sâu bên trong ít cứng hơn và phản ứng tốt hơn với các tần số thấp. Sự kết hợp giữa cấu trúc phức tạp của ốc tai và phân bố không đồng đều của các tế bào lông tạo ra một cảm giác cảm nhận âm thanh không tuyến tính, nơi tai nhạy cảm với âm thanh ở tần số thấp và ít nhạy cảm hơn ở tần số cao.



Hình 2.2 Cấu tạo ốc tai

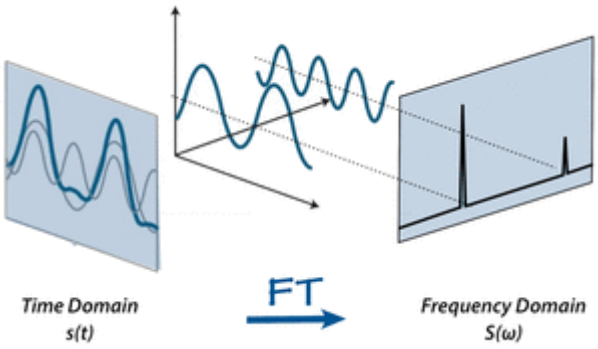
Cụ thể, Hình 2.2 cho thấy rằng:

- Các tần số âm thanh cao được xử lý ở gần đỉnh ốc tai, trong khi tần số âm thanh thấp được xử lý ở gần đáy ốc tai.
- Tần số âm thanh cao nhất mà ốc tai có thể xử lý giảm dần khi di chuyển từ đỉnh ốc tai xuống đáy ốc tai.
- Ví dụ, nếu con người nghe thấy một âm thanh có tần số 10.000 Hz, các tế bào lông ở phần đỉnh của ốc tai sẽ rung động. Các tế bào lông này sẽ gửi tín hiệu điện đến não, và não bộ sẽ giải thích các tín hiệu này là âm thanh có tần số 10.000 Hz.

Trong xử lý tiếng nói, cần một cơ chế để ánh xạ giữa tín hiệu âm thanh thu được bằng cảm biến và độ cảm nhận của tai người. Việc ánh xạ này được thực hiện bởi bộ lọc Mel, tôi sẽ trình bày chi tiết hơn về bô lọc Mel ở phần sau.

2.1.4 Biến đổi Fourier

Một mảng kiến thức không thể thiếu khi làm việc với tín hiệu âm thanh là xử lí tín hiệu số, trọng tâm là biến đổi Fourier. Biến đổi Fourier là một phép biến đổi toán học chuyển đổi một hàm số hoặc một tín hiệu từ miền thời gian sang miền tần số.



Hình 2.3 Biến đổi Fourier

Âm thanh là một chuỗi tín hiệu rất dài, nhưng hàm lượng thông tin trong đó không nhiều. Mà âm thanh lại được kết hợp từ các sóng có tần số khác nhau, nên các nhà nghiên cứu đã phân giải một đoạn âm thanh ngắn thành các sóng với tần số và biên độ cụ thể. Điều đó được minh hoạ khá dễ hiểu bằng Hình 2.3. Trong hình, một đoạn âm thanh trong miền thời gian được kết hợp từ 2 sóng tuần hoàn. Do 2 sóng này có tính chất tuần hoàn, thay vì phải lưu giá trị theo thời gian, ta chỉ cần lưu lại tần số, biên độ và pha giao động của các sóng này. Từ đó, có một cách biểu diễn mới giàu thông tin hơn cho đoạn âm thanh đó.

Như vậy, với biến đổi Fourier, ta đã chuyển đổi thông tin từ miền thời gian sang miền tần số.

+ Công thức Biến đổi Fourier cho hàm liên tục:

$$F(w) = \int_{-\infty}^{\infty} f(x)e^{-iwx}dx \qquad PT\ (2.1)$$

Trong đó:

F(w) là kết quả biến đổi tín hiệu từ miền thời gian sang miền tần số.

w là tần số góc, w = 2πf, đơn vị rad/s.

i là biến phức.

f(x) là tín hiệu liên tục trong miền thời gian.

+ Vì tín hiệu âm thanh thu được qua micrô là dãy số rời rạc nên với bài toán này ta sẽ sử dụng biến đổi Fourier cho hàm rời rạc (DFT). Dãy của N số phức x_0, x_1, … x_{N-1} được biến đổi thành chuỗi của N số phức X_0 , X_1, … X_{N-1} bởi công thức:

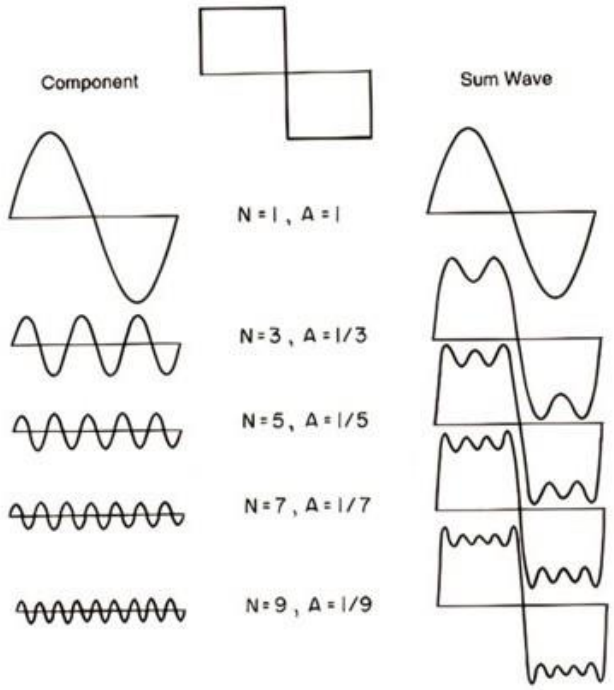
$$X_n = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N}kn} \qquad PT\ (2.2)$$

Với k = 0,1,2…N-1

+ Trong các thuật toán hiện nay, thay vì dùng thuật toán DFT gốc, người ta dùng biến đổi Fourier nhanh (FFT) - 1 thuật toán hiệu quả và nhanh để tăng tốc độ tính toán. FFT dựa trên phương pháp chia để trị và kỹ thuật đệ quy để giảm độ phức tạp tính toán, là một cách để đạt được cùng kết quả với DFT nhưng nhanh

hơn nhiều: tính DFT của N điểm trực tiếp theo định nghĩa cần N^2 phép tính, trong khi FFT tính ra cùng kết quả đó trong $N*\log_2(N)$ phép tính.

Biến đổi Fourier là biến đổi đối xứng, tức 1 thông tin được biến đổi Fourier từ miền thời gian sang miền tần số, ta có thể biến đổi Fourier ngược để khôi phục thông tin từ miền tần số lại về miền thời gian. Hình 2.4 là 1 minh hoạ 1 sóng vuông được phân giải thành các sóng sin. Có thể thấy với giá trị N càng cao, độ chính xác càng lớn.



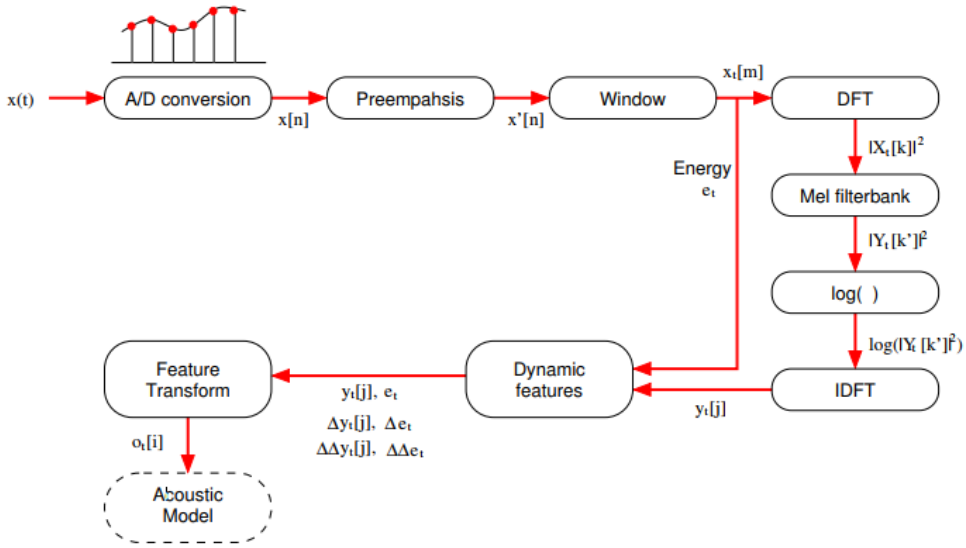
Hình 2.4 Một sóng vuông được phân giải thành các sóng sin

2.2 Đặc trưng MFCC

MFCC, viết tắt của "Mel-Frequency Cepstral Coefficients", là một phương pháp trong xử lý tín hiệu âm thanh, được sử dụng để biểu diễn và trích xuất các đặc trưng quan trọng từ tín hiệu âm thanh [3].

Quá trình tính toán Mel-frequency cepstral coefficients (MFCC) có thể được mô tả theo quy trình sau: tín hiệu âm thanh được chia thành các đoạn ngắn có độ dài cố định (25ms) và được xếp chồng lên nhau với khoảng cách (10ms). Mỗi phân đoạn âm thanh này sau đó được chuyển đổi và tính toán để tạo ra một tập hợp gồm 39 đặc trưng. Tính chất của mỗi bộ 39 đặc trưng này là sự độc lập cao, ít nhiễu, và kích thước nhỏ đủ để đảm bảo tính hiệu quả của quá trình tính toán. Đồng thời, nó cung cấp đủ thông tin cần thiết để bảo đảm chất lượng cho các thuật toán nhận dạng.

Hình 2.5 mô tả luồng xử lý từ âm thanh đầu vào để tạo ra 39 đặc trưng MFCC:



Hình 2.5 Luồng xử lý âm thanh từ đầu vào đến MFCC.

2.2.1 Chuyển đổi A/D và Khuếch đại tín hiệu (Pre-emphasis)

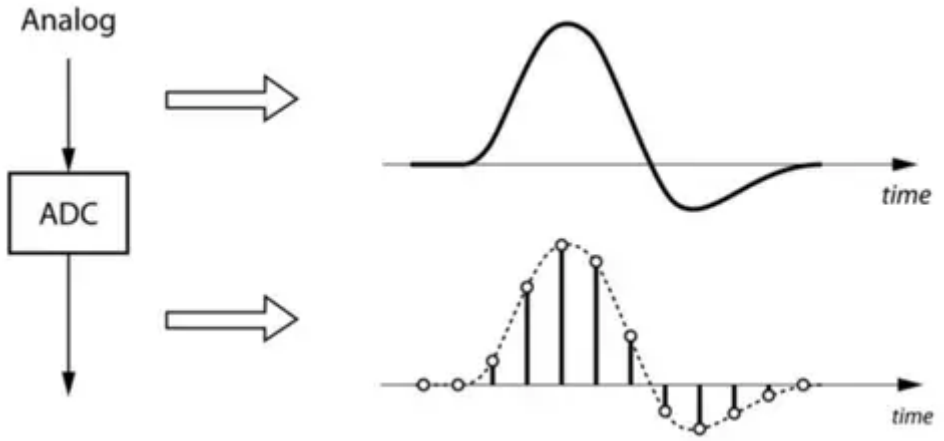
Chuyển đổi A/D

Âm thanh là dạng tín hiệu liên tục, trong khi đó máy tính làm việc với các con số rời rạc. Ta cần lấy mẫu tại các khoảng thời gian cách đều nhau với 1 tần số lấy mẫu xác định (sample rate) để chuyển từ dạng tín hiệu liên tục về dạng rời rạc.

Ví dụ: tần số lấy mẫu là 8000 có nghĩa là trong 1s lấy 8000 giá trị.

Tai người nghe được âm thanh trong khoảng tần số 20Hz đến 20000 Hz. Theo định lý lấy mẫu Nyquist–Shannon: với một tín hiệu có các tần số thành phần $\leq f_m$, để đảm bảo việc lấy mẫu không làm mất mát thông tin, tần số lấy mẫu f_s phải đảm bảo: $f_s \geq 2f_m$.

Vậy để đảm bảo việc lấy mẫu không làm mất mát thông tin, tần số lấy mẫu: $f_s = 44100$Hz. Tuy nhiên trong nhiều trường hợp, người ta chỉ cần lấy $f_s = 8000$ Hz hoặc $f_s = 16000$Hz.



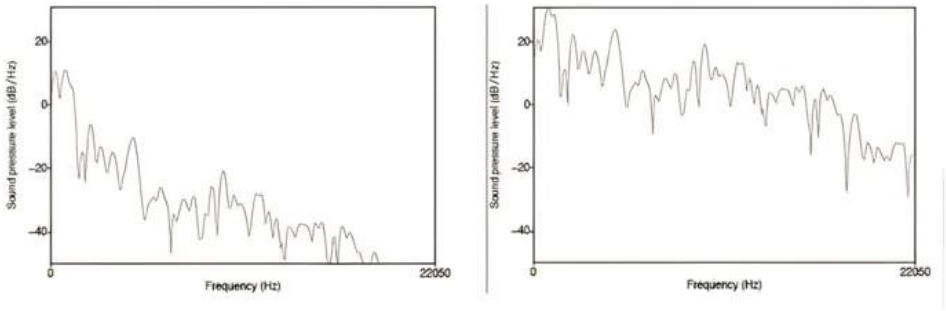
Hình 2.6 Biến đổi ADC

Hình 2.6 biển diễn âm thanh dưới dạng đồ thì gồm 2 trục: trục hoàng là trục thời gian, trục tung là trục biên độ. Trong đề tài, chúng ta sẽ sử dụng tần số lấy mẫu là $f_s = 16000$Hz.

Khuếch đại tín hiệu (Pre-emphasis)

Vì cấu trúc đặc biệt của thanh quản và các phần phát âm, tiếng nói của chúng ta có đặc điểm là âm thanh ở tần số thấp thường có năng lượng cao, trong khi âm thanh ở tần số cao thường có năng lượng thấp hơn. Trong khi đó, các tần số cao này vẫn chứa nhiều thông tin về âm vị. Vì vậy, chúng ta cần một bước khuếch đại tín hiệu để kích các tín hiệu ở tần số cao này lên.

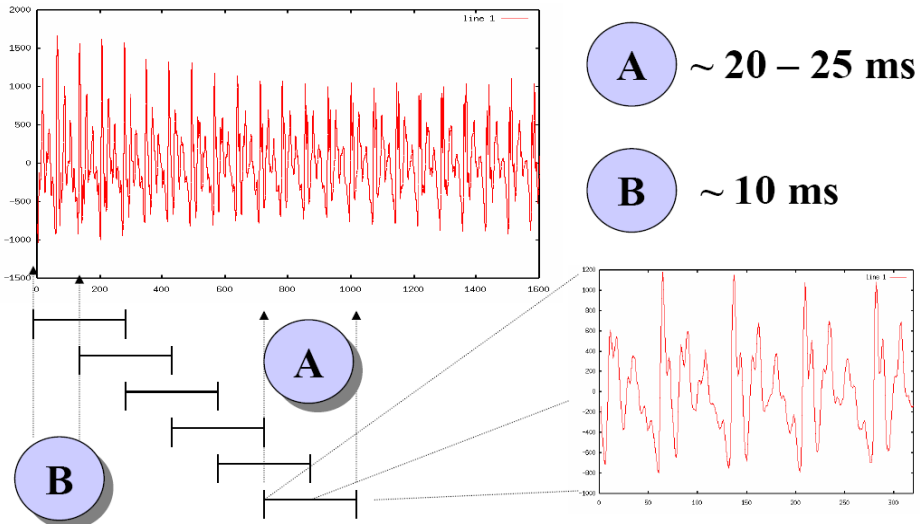
$$x'[t_d] = x[t_d] - \alpha x[t_d - 1] \qquad 0.95 < \alpha < 0.99$$



Hình 2.7 Khuếch đại tín hiệu

2.2.2 Lấy cửa sổ tín hiệu (Windowing)

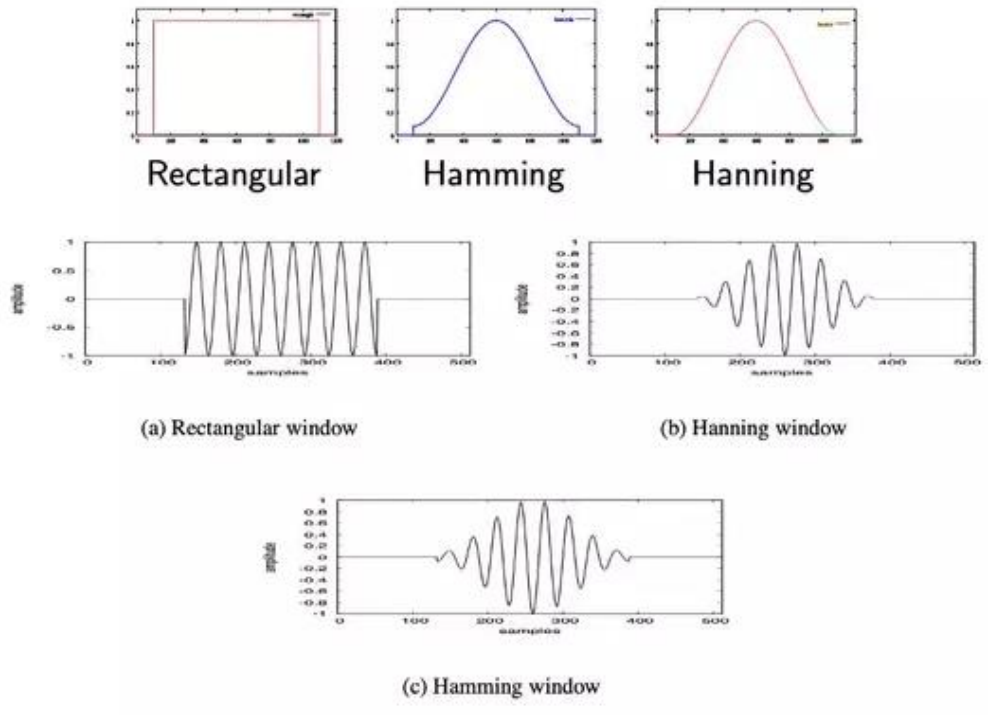
Thay vì biến đổi Fourier trên cả đoạn âm thanh dài, ta trượt 1 cửa sổ dọc theo tín hiệu để lấy ra các khung âm thanh rồi mới áp dụng DFT trên từng khung này. Tốc độ nói của con người trung bình khoảng 3-4 từ mỗi giây, mỗi từ khoảng 3-4 âm, mỗi âm chia thành 3-4 phần, như vậy 1 giây âm thanh được chia thành 36 - 40 phần, ta chọn độ rộng mỗi khung khoảng 20 - 25ms là vừa đủ rộng để bao 1 phần âm thanh. Các khung được xếp chồng lên nhau khoảng 10ms để có thể ghi lại sự thay đổi ngữ cảnh.



Hình 2.8 Cách lấy cửa sổ tín hiệu

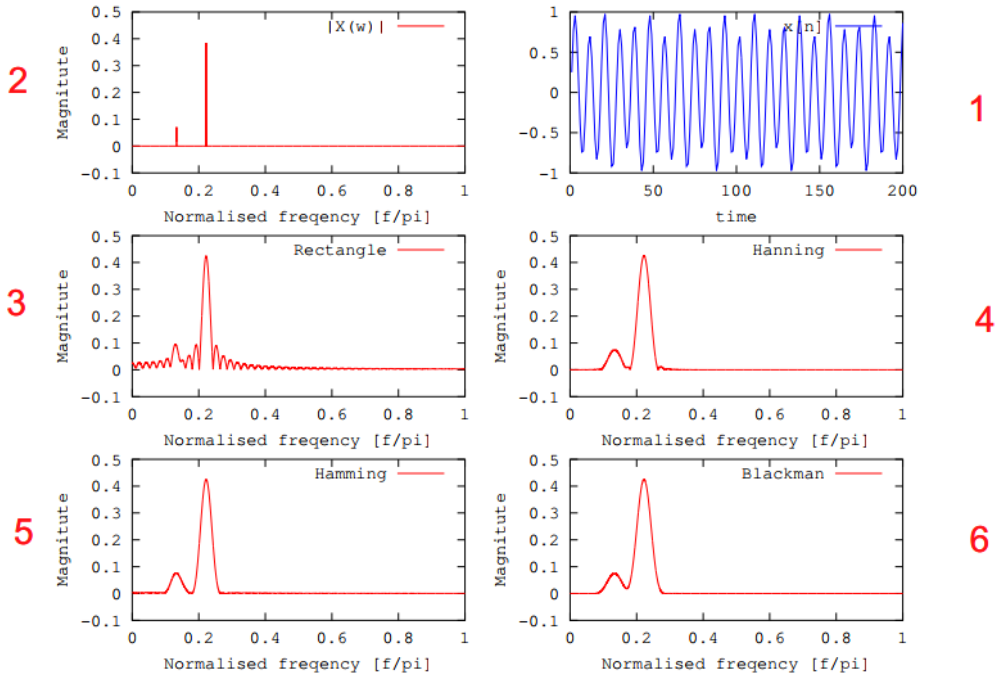
Tuy nhiên, việc cắt khung sẽ làm các giá trị ở 2 biên của khung bị giảm đột ngột (về giá trị 0), sẽ dẫn tới hiện tượng: khi DFT sang miền tần số sẽ có rất nhiều nhiễu ở tần số cao. Để khắc phục điều này, ta cần làm mượt bằng cách nhân chập

khung với một loại cửa sổ. Có 1 vài loại cửa sổ phổ biến như: Hamming, Hanning... có tác dụng làm giá trị biên khung giảm xuống từ từ từ.



Hình 2.9 Một số loại cửa sổ phổ biến

Hình 2.10 sẽ cho ta thấy rõ được tác dụng của các cửa số tín hiệu này. Trong các hình nhỏ, Hình 2.10.1 là một khung được cắt ra từ âm thanh gốc, âm thanh gốc là sự kết hợp của 2 sóng Hình 2.10.2. Nếu áp dụng cửa sổ Rectangle (tức là cắt trực tiếp), tín hiệu miền tần số tương ứng là Hình 2.10.3, ta có thể thấy tín hiệu này chứa rất nhiều nhiễu. Nếu áp dụng các cửa số tín hiệu như Hanning, Hamming, Blackman… tín hiệu miền tần số thu được khá mượt với sóng ở Hình 2.10.4,5,6.



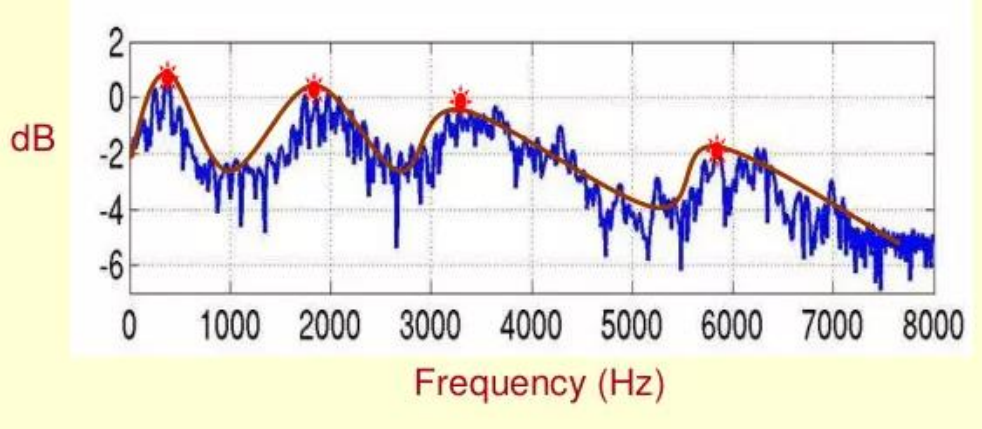
Hình 2.10 Tác dụng của một số loại cửa sổ

2.2.3 Biến đổi Fourier cho hàm rời rạc (DFT)

Trên từng khung, ta áp dụng DFT theo công thức:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}kn} \qquad PT\ (2.3)$$

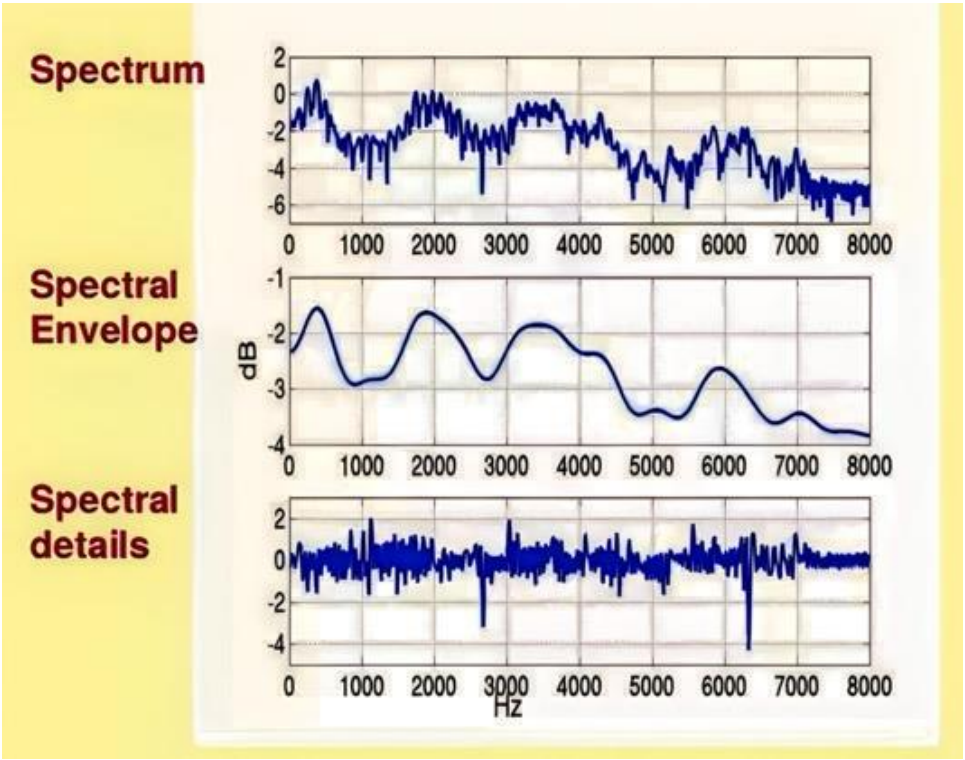
Kết quả của biến đổi này là âm phổ (spectrum) trên từng khung, được biểu diễn dưới dạng hai chiều: trục tung là cường độ (dB), trục hoành là tần số (Hz).



Hình 2.11 Kết quả phép biến đổi DFT

Ở Hình 2.11,các điểm màu đỏ là các "formants", là nơi có các tần số áp đảo, mang đặc tính của âm thanh. Đường màu đỏ gọi là Spectral Envelopes. Mục tiêu chính của ta là tìm được đường màu đỏ này.

Gọi âm phổ là X[k] có hai thành phần là Spectral Envelopes H[k] và Spectral Details E[k]



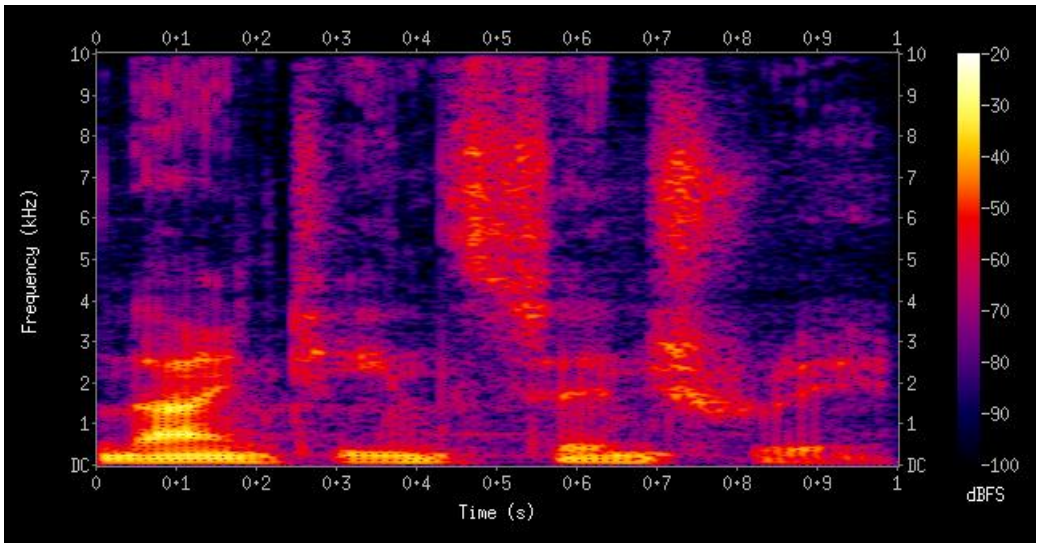
Hình 2.12 Phân tích âm phổ

Để tách được H[k], ta cần phải lấy logarit của Spectrum và lấy phần ở tần số thấp:

$$X[k] = H[k] * E[k] \qquad PT\ (2.4)$$

$$\log(X[k]) = \log(H[k]) + \log(H[k]) \qquad PT\ (2.5)$$

Mỗi khung dữ liệu cung cấp một danh sách các cường độ (magnitude) tương ứng với các tần số từ 0 đến N. Sau khi áp dụng qua tất cả các khung, chúng ta thu được một biểu đồ phổ tần số (spectrogram) như được minh họa trong Hình 2.13. Trục x biểu diễn thời gian (theo thứ tự của các khung), trong khi trục y đại diện cho dải tần số từ 0 đến 10000 Hz. Cường độ tại mỗi tần số được biểu thị bằng màu sắc. Thông qua quan sát của biểu đồ phổ tần số này, chúng ta nhận thấy rằng tần số thấp thường có cường độ cao, trong khi tần số cao thường có cường độ thấp.



Hình 2.13 Phổ tần số

Trên đây là cách tạo ra biểu đồ phổ tần số. Tuy nhiên, trong nhiều bài toán, đặc biệt là trong việc nhận dạng câu lệnh, biểu đồ phổ tần số không phải luôn là sự lựa chọn lý tưởng. Vì vậy, cần thêm một số bước tính toán để thu được Mel-frequency cepstral coefficients (MFCC), một phương pháp phổ biến và hiệu quả hơn so với biểu đồ phổ tần số.

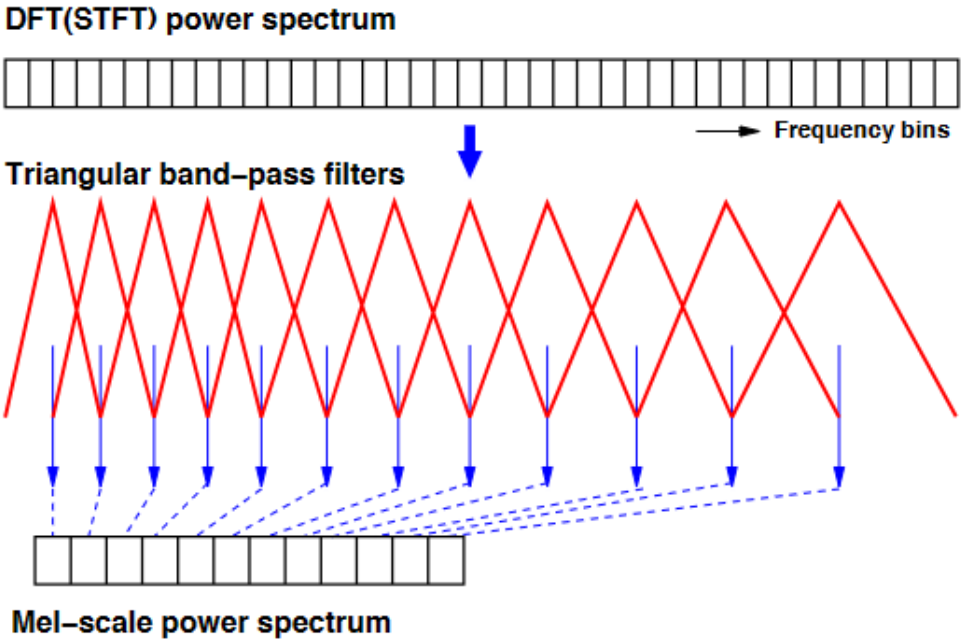
2.2.4 Bộ lọc Mel

Như đã mô tả ở Cơ chế hoạt động của tai, cách cảm nhận của tai người là phi tuyến tính, không giống các thiết bị đo. Tai người cảm nhận tốt ở các tần số thấp, kém nhạy cảm với các tần số cao, nên ta cần 1 cơ chế ánh xạ tương tự như vậy – thang Mel.

Dưới tần số 1000Hz, thang Mel gần như tuyến tính với thang tần số tuyến tính. Trên điểm tham chiếu 1000Hz, người nghe có xu hướng cảm nhận được mức tăng cao độ giống nhau với các khoảng tần số ngày càng dài hơn. Do đó mối quan hệ giữa thang đo Mel và thang tần số tuyến tính là phi tuyến tính và xấp xỉ logarit trên 1000Hz. Phương trình sau đây mô tả mối quan hệ toán học giữa thang đo Mel và thang tần số tuyến tính:

$$f_{Mel} = 1127.01\ln\left(\frac{f}{700} + 1\right) \qquad PT\ (2.6)$$

Trong đó f_{Mel} là tần số Mel tính bằng mels và f là tần số tuyến tính tính bằng Hz.



Hình 2.14 Cơ chế ánh xạ

Ban đầu, ta bình phương các giá trị trong phổ tần số thu được ở bước DFT để tạo ra Phổ công suất (Power Spectrum). Sau đó, ta áp dụng 1 tập các bộ lọc thông dải Mel trên từng khoảng tần số (mỗi bộ lọc áp dụng trên 1 dải tần xác định). Giá trị đầu ra của từng bộ lọc là năng lượng dải tần số mà bộ lọc đó bao phủ được. Ta thu được Phổ công suất Mel (Mel-scale Power Spectrum). Từ Hình 2.14 ta cũng thấy, các bộ lọc dùng cho dải tần thấp thường hẹp hơn các bộ lọc dùng cho dải tần cao.

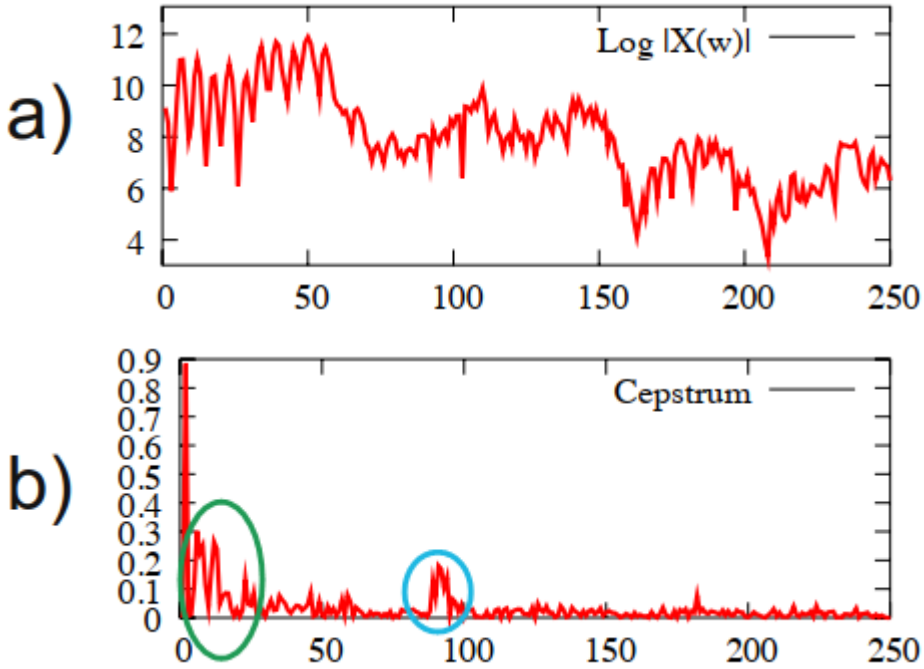
2.2.5 Log và Biến đổi Fourier ngược (IDFT)

Log

Bộ lọc Mel trả về phổ công suất của âm thanh, hay còn gọi là phổ năng lượng. Thực tế rằng con người kém nhạy cảm trong sự thay đổi năng lượng ở các tần số cao, nhạy cảm hơn ở tần số thấp. Vì vậy ta sẽ tính logarit trên phổ công suất Mel. Điều này còn giúp giảm các biến thể âm thanh không đáng kể để nhận dạng giọng nói.

Biến đổi Fourier ngược (IDFT)

Như đã mô tả ở chương 2, giọng nói của chúng ta có tần số F0 - tần số cơ bản và các "formant" F1, F2, F3 ... Tần số F0 ở nam giới khoảng 125 Hz, ở nữ là 210 Hz, đặc trưng cho cao độ giọng nói ở từng người. Thông tin về cao độ này không giúp ích trong nhận dạng giọng nói, nên ta cần tìm cách để loại thông tin về F0 đi, giúp các mô hình nhận dạng không bị phụ thuộc vào cao độ giọng từng người.



Hình 2.15 Biến đổi Fourier ngược (IDFT)

Với đầu vào là phổ logarit ở Hình 2.15a, ta sử dụng IDFT để tạo ra Cepstrum ở Hình 2.15b. Cepstrum bây giờ sẽ giống như tín hiệu giọng nói, biểu diễn dưới dạng hai chiều. Nhưng giá trị sẽ khác : trục tung là độ lớn (không có đơn vị) và là tần số quefrency (ms).

Khi đó, với Cepstrum thu được, phần thông tin liên quan tới F0 và phần thông tin liên quan tới F1, F2, F3 ... nằm tách biệt nhau như 2 phần khoanh tròn trong Hình 2.15b. Ta chỉ đơn giản lấy thông tin trong đoạn đầu của Cepstrum (phần được khoanh tròn màu xanh lá).

Để tính MFCC, ta chỉ cần lấy 12 giá trị đầu tiên.

2.2.6 MFCC

Như vậy, mỗi khung ta đã trích xuất ra được 12 đặc trưng Cepstral làm 12 đặc trưng đầu tiên của MFCC. Đặc trưng thứ thứ 13 là năng lượng của khung đó đó, tính theo công thức:

$$Energy = \sum_{n=1}^{N} {x_n}^2 \qquad PT\ (2.7)$$

Với N là số lượng mẫu trong khung và x_n là biên độ tại mẫu thứ n.

Thực tế, có 2 cách để chọn ra 13 đặc trưng đầu tiên của MFCC: lấy trực tiếp 13 đặc trưng Cepstral hoặc 12 đặc trưng Cepstral và năng lượng của khung đó. Ở đề tài này, tôi sẽ lấy 13 đặc trưng Cepstral.

Trong xử lý tiếng nói, thông tin về bối cảnh và sự thay đổi rất quan trọng. Ví dụ tại những điểm mở đầu hoặc kết thúc ở nhiều phụ âm, sự thay đổi này rất rõ rệt, có thể nhận dạng các âm vị dựa vào sự thay đổi này. 13 hệ số tiếp theo chính là

đạo hàm bậc 1 (theo thời gian) của 13 đặc trưng đầu tiên. Nó chứa thông tin về sự thay đổi từ khung thứ t-1 đến khung t+1. Công thức:

$$d(t) = \frac{c(t+1) - c(t-1)}{2} \qquad PT\ (2.8)$$

Tương tự như vậy, 13 giá trị cuối của MFCC là sự thay đổi d(t) theo thời gian - đạo hàm của d(t), đồng thời là đạo hàm bậc 2 của c(t). Công thức:

$$b(t) = \frac{d(t+1) - d(t-1)}{2} \qquad PT\ (2.9)$$

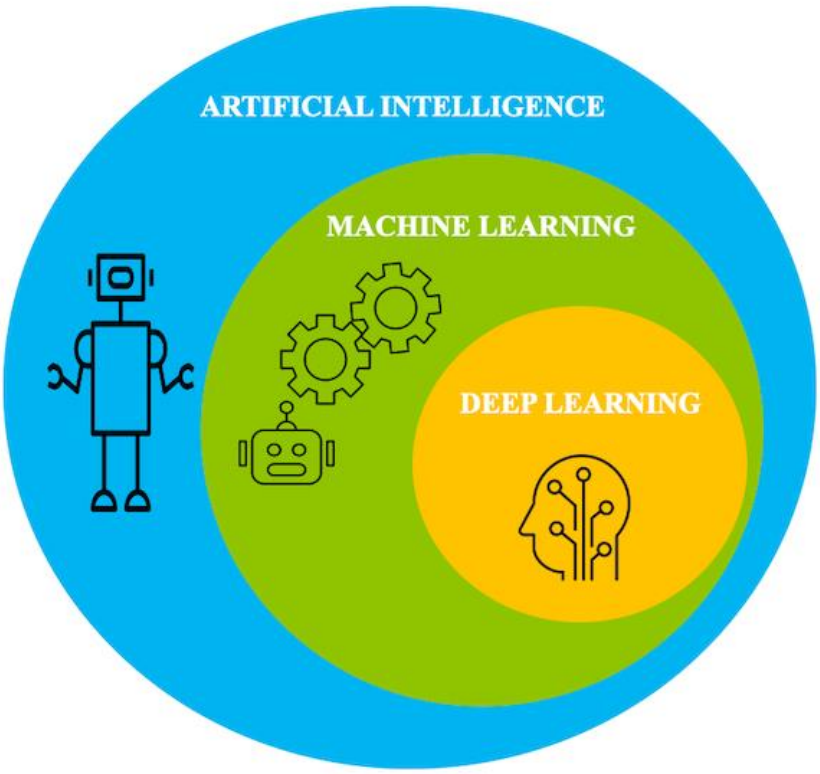
Vậy, từ 13 đặc trưng MFCC ban đầu, ta đạo hàm 2 lần và thu được 39 đặc trưng. Đây chính là 39 đặc trưng MFCC mà ta sẽ sử dụng trong mô hình.

2.3 Kết luận chương

Như vậy, trong chương này, chúng ta đã tìm hiểu được cơ bản về xử lý tiếng nói: nguyên lý hình thành tiếng nói và các thành phần ngữ âm của tiếng nói; cách trích xuất đặc trưng MFCC. Các khung 39 đặc trưng MFCC này sẽ được đưa vào mô hình học sâu để tiến hành dự đoán nhận dạng câu lệnh. Chương tiếp theo, tôi sẽ giới thiệu cơ sở lý thuyết về học sâu.

CHƯƠNG 3. CƠ SỞ LÝ THUYẾT VỀ HỌC SÂU

3.1 Khái niệm



Hình 3.1 Học sâu

Học sâu (Deep Learning) là một phương thức trong lĩnh vực trí tuệ nhân tạo (AI), được sử dụng để huấn luyện máy tính xử lý dữ liệu theo cách được lấy cảm hứng từ bộ não con người. Mô hình học sâu có thể nhận diện nhiều hình mẫu phức tạp trong hình ảnh, văn bản, âm thanh và các dữ liệu khác để tạo ra thông tin chuyên sâu và dự đoán chính xác. Học sâu thúc đẩy nhiều ứng dụng AI được sử dụng trong các sản phẩm hàng ngày, chẳng hạn như:

+ Trợ lý kỹ thuật số.

+ Điều khiển các thiết bị từ xa kích hoạt bằng giọng nói.

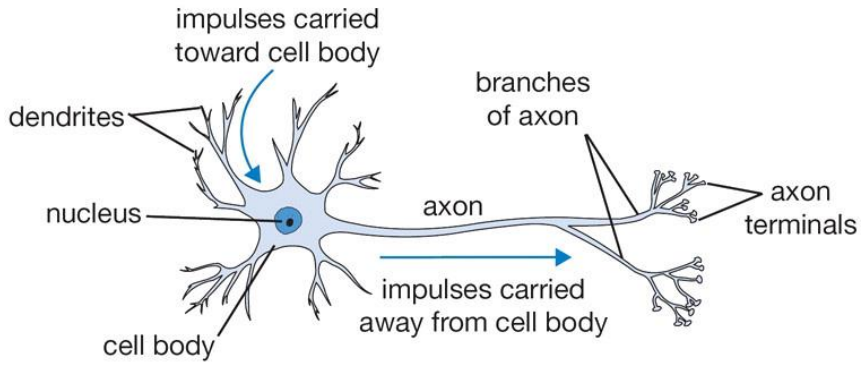
+ Phát hiện gian lận.

+ Nhận dạng khuôn mặt tự động.

…

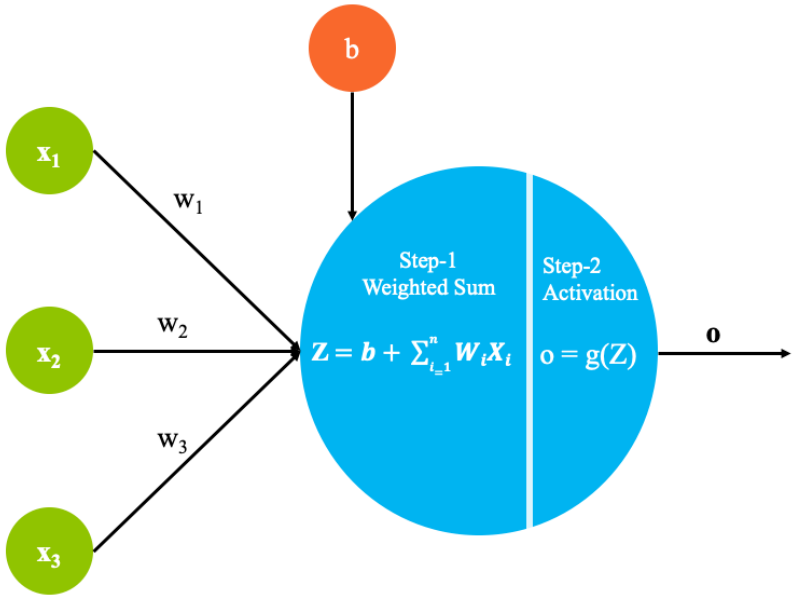
3.2 Học sâu và bộ não con người

Trong nỗ lực tạo ra các hệ thống học tương tự như cách con người học, kiến trúc cơ bản cho việc học sâu được lấy cảm hứng từ cấu trúc của bộ não con người. Vì lý do này, khá nhiều thuật ngữ cơ bản trong học sâu có thể được ánh xạ trở lại thần kinh học. Tương tự như cách các tế bào thần kinh (nơ-ron) hình thành các khối xây dựng cơ bản của não, kiến trúc học sâu chứa một đơn vị tính toán cho phép mô hình hóa các hàm phi tuyến được gọi là “perceptron” [4].



Hình 3.2 Cấu trúc nơ-ron

Cấu trúc "perceptron" trong học sâu mô phỏng một nơ-ron trong não của chúng ta, cần có nhiều giá trị đầu vào, ở phần nhân tế bào chọn thông tin nào đi qua và chuyển thông tin đó đến nơ-ron tiếp theo.



Hình 3.3 Cấu trúc perceptron trong học sâu

Các bước tính toán trong cấu trúc perceptron:

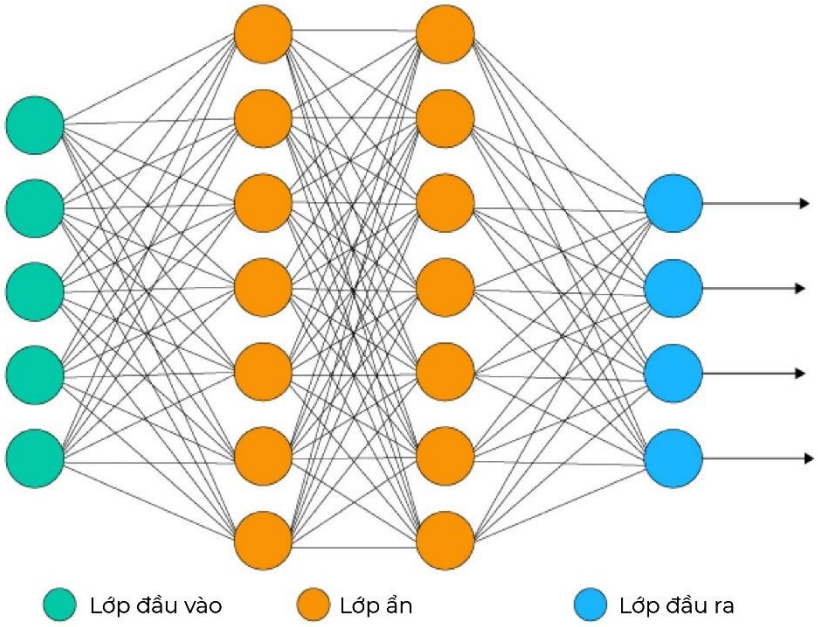
- Bước 1: Tính tổng trọng số
  + Các đầu vào từ x_1 đến x_n, cũng có thể được ký hiệu bằng véc-tơ X. X_i đại diện cho mục nhập thứ i từ tập dữ liệu. Mỗi mục nhập từ tập dữ liệu chứa n biến phụ thuộc.
  + Các trọng số từ w_1 đến w_n, có thể được ký hiệu là ma trận W.
  + Một số hạng thiên vị b, là một hằng số.
- Bước 2: Chức năng kích hoạt
  + Đầu ra của bước 1 bây giờ được chuyển qua một chức năng kích hoạt. Hàm kích hoạt g là một hàm toán học cho phép biến đổi kết quả đầu ra sang định dạng phi tuyến tính mong muốn trước khi nó được gửi đến lớp tiếp theo. Nó ánh xạ kết quả tổng kết đến một phạm vi mong muốn. Điều này giúp xác định xem liệu nơ-ron có cần được kích hoạt hay không.

+ Ví dụ, một hàm sigmoid ánh xạ các giá trị đến phạm vi [0,1], rất hữu ích đối với các ứng dụng dự đoán các xác suất. Tôi sẽ trình bày về các hàm kích hoạt kỹ hơn ở phần sau.

Sự kỳ diệu của học sâu bắt đầu với "perceptron" khiêm tốn. Tương tự như cách nơ-ron trong não người truyền các xung điện trong hệ thống thần kinh của chúng ta, tế bào cảm thụ nhận một danh sách các tín hiệu đầu vào và chuyển chúng thành tín hiệu đầu ra.

"Perceptron" nhằm mục đích hiểu được biểu diễn dữ liệu bằng cách xếp chồng nhiều lớp lại với nhau, trong đó mỗi lớp chịu trách nhiệm hiểu một số phần của đầu vào. Một lớp có thể được coi là một tập hợp các đơn vị tính toán học cách phát hiện sự xuất hiện lặp lại của các giá trị.

Mỗi lớp "perceptron" chịu trách nhiệm giải thích một mẫu cụ thể trong dữ liệu. Một mạng lưới các "perceptron" này bắt chước cách các nơ-ron trong não tạo thành một mạng lưới, vì vậy kiến trúc được gọi là mạng nơ-ron (hay mạng nơ-ron nhân tạo). Ở dạng cơ bản nhất, mạng nơ-ron nhân tạo chứa ba lớp: lớp đầu vào, lớp ẩn và lớp đầu ra.



Hình 3.4 Mạng nơ-ron nhân tạo

3.3 Quá trình huấn luyện mạng nơ-ron nhân tạo

Các tính toán được thảo luận trong các phần trước xảy ra cho tất cả các nơ-ron trong mạng nơ-ron bao gồm cả lớp đầu ra và một lần truyền như vậy được gọi là quá trình truyền chuyển tiếp.

Sau khi hoàn thành một lần chuyển tiếp, lớp đầu ra phải so sánh kết quả của nó với các nhãn thực tế và điều chỉnh trọng số dựa trên sự khác biệt giữa giá trị thực tế và các giá trị dự đoán. Sự truyền ngược qua mạng nơ-ron được gọi là quá trình lan truyền ngược. Những điều cơ bản của quá trình này có thể được trình bày như sau:

- Mạng hoạt động để giảm thiểu một hàm chi phí, ví dụ: lỗi phát sinh trên tất cả các điểm trong mẫu dữ liệu. Tại lớp đầu ra, mạng phải tính toán tổng sai số (chênh lệch giữa giá trị thực tế và giá trị dự đoán) cho tất cả các điểm dữ liệu và lấy đạo hàm của nó đối với trọng số ở lớp đó. Đạo hàm của hàm lỗi đối với trọng số được gọi là "gradient" của lớp đó.
- Trọng số cho lớp đó sau đó được cập nhật dựa trên "gradient". Bản cập nhật này có thể là bản thân "gradient" hoặc một yếu tố của nó. Yếu tố này được gọi là tỷ lệ học tập và nó kiểm soát mức độ lớn của các bước mô hình thực hiện để thay đổi trọng lượng.
- Quá trình sau đó được lặp lại cho một lớp trước đó và tiếp tục cho đến khi đến được lớp đầu tiên.
- Trong quá trình này, các giá trị của "gradient" từ các lớp trước đó có thể được sử dụng lại, giúp tính toán "gradient" hiệu quả.
- Kết quả của một lần truyền chuyển tiếp và lan truyền ngược là sự thay đổi trọng số của các lớp mạng và đưa hệ thống tiến gần hơn đến việc mô hình hóa tập dữ liệu được cung cấp cho nó. Bởi vì quá trình này sử dụng "gradient" để giảm thiểu lỗi tổng thể, quá trình hội tụ các tham số của mạng nơ-ron đến mức tối ưu được gọi là "gradient descent".

3.4 Một số khái niệm tính toán toán học

3.4.1 Hàm kích hoạt

Hàm kích hoạt đóng một vài trò quan trọng trong việc quyết định chất lượng của mô hình, có 2 lý do chính cho việc sử dụng hàm kích hoạt:

- Tạo sự phi tuyến cho mô hình: Thực tế thì dữ liệu sẽ có phân bố phức tạp và sử dụng một hàm tuyến tính là không đủ mạnh để có thể biểu diễn, việc sử dụng hàm phi tuyến giúp cho mô hình học tốt hơn những "đặc trưng đặc biệt" của bài toán.
- Giữ các giá trị đầu ra trong khoảng nhất định: Nếu không sử dụng hàm kích hoạt và với một mô hình hàng triệu tham số thì kết quả của phép nhân tuyến tính sẽ có thể là một giá trị rất lớn (dương vô cùng) hoặc rất bé (âm vô cùng). Từ đó có thể gây ra những vấn đề về mặt tính toán (nan) và mạng rất khó để có thể hội tụ. Việc sử dụng hàm kích hoạt có thể giới hạn đầu ra ở một khoảng giá trị nào đó, ví dụ như hàm sigmoid, softmax giới hạn giá trị đầu ra trong khoảng (0, 1) cho dù kết quả của phép nhân tuyến tính là bao nhiêu đi chăng nữa.

Với đầu vào là các giá trị đặc trưng MFCC và đầu ra xác suất của các câu lệnh. Đối với đầu vào là một véc-tơ x thì đầu ra của nó sẽ là:

$$y = \omega^T x + b \qquad PT\ (3.1)$$

Trong đó:

+ x là vector có kích thước n.

+ w là vector tham số có kích thước n.

+ y là giá trị đầu ra.

+ b là số thực.

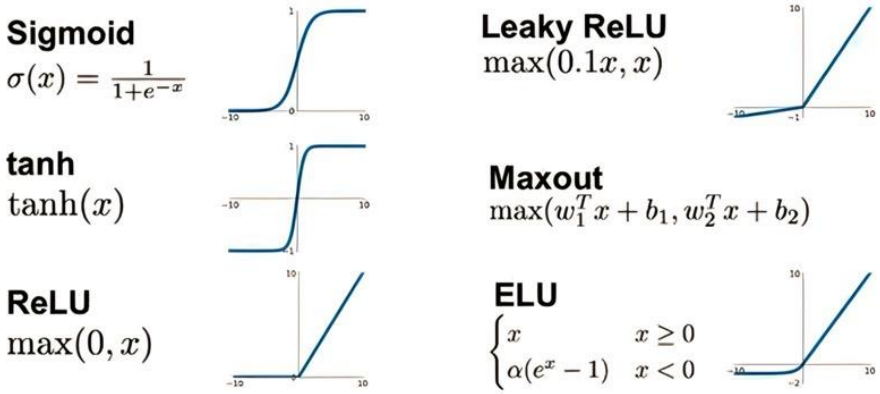
Vì đầu ra y ta cần là xác suất của bài toán phân loại, nên nó nằm trong khoảng từ 0 đến 1, nhưng với phương trình trên nó có thể cho ra bất kỳ số thực nào. Vì vậy, hàm kích hoạt softmax để đưa ra xác xuất đầu ra:

$$\hat{y} = s(\omega^T x + b) \qquad PT\ (3.2)$$

Với bất kì giá trị nào, hàm sofmax sẽ đưa ra giá trị từ 0 đến 1, tổng các phần tử đúng bằng 1. Công thức hàm sigmoid với đầu ra gồm N phần tử như sau:

$$s_i(z) = \frac{e^{z_i}}{\sum_0^{N-1} e^{z_i}} \qquad PT\ (3.3)$$

Một số hàm kích hoạt phổ biến khác [5]:



Hình 3.5 Một số hàm kích hoạt phổ biến.

3.4.2 Hàm chi phí

Để huấn luyện các tham số w và b, chúng ta cần một hàm chi phí. Ta muốn tìm các tham số w và b sao cho ít nhất trên tập huấn luyện, các đầu ra ta có $\hat{y}$ gần với các giá trị thực tế y nhất.

Đối với bài toán phân loại có hai hàm mất mát phổ được định nghĩa như sau:

- Phân loại nhị phân (Binary Crossentropy Loss):

$$L(\hat{y}, y) = -(ylog(\hat{y}) + (1-y)\log(1-\hat{y})) \qquad PT\ (3.4)$$

- Phân loại nhiều lớp (Categorical Crossentropy Loss):

$$L(\hat{y}, y) = -\sum_{i=0}^{N-1} y_i * \log(\hat{y}_i) \qquad PT\ (3.5)$$

Hàm mất mát trên chỉ định nghĩa cho một ví dụ đào tạo duy nhất. Mặt khác, chúng ta cần một hàm chi phí dành cho toàn bộ tập huấn luyện. Với m là số mẫu trong tập huấn huyện, hàm chi phí được định nghĩa:

$$J(\omega, b) = \frac{1}{m}\sum_{i=1}^{m} L\left(\hat{y}^{(i)}, y^{(i)}\right) \qquad PT\ (3.6)$$

Chúng ta muốn giá trị hàm chi phí càng nhỏ càng tốt, vì vậy chúng ta muốn tìm được w, b để tối ưu hóa hàm trên.

3.4.3 Gradient Descent

Đây là một kĩ thuật để tìm được các giá trị w,b làm cho hàm chí phí đạt giá trị cực tiểu. Hàm chi phí có bản chất lồi (tức là chỉ có một cực tiểu toàn cầu). Ta có các bước tìm "gradient descent":

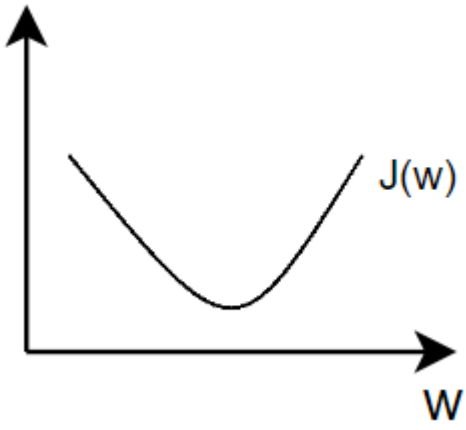
- Khởi tạo trọng số: Bắt đầu với một bộ trọng số ngẫu nhiên hoặc được khởi tạo một cách xác định.
- Dự đoán: Sử dụng các trọng số hiện tại để thực hiện dự đoán trên tập dữ liệu đào tạo.
- Tính toán đạo hàm (Gradient): Tính toán đạo hàm của hàm mất mát theo từng trọng số. Đạo hàm này đại diện cho hướng và độ lớn của sự thay đổi của hàm mất mát.
- Cập nhật trọng số: Sử dụng đạo hàm tính được, cập nhật trọng số của mô hình theo hướng ngược với "gradient" để giảm giá trị của hàm mất mát. Bước cập nhật được điều chỉnh thông qua một tỷ lệ học tập (learning rate).
- Lặp lại: Lặp lại quá trình từ bước 2 cho đến khi đạt được điều kiện dừng, ví dụ như đạt đến số lần lặp cố định hoặc khi giá trị của hàm mất mát không thay đổi đáng kể.

Phương trình cập nhật cho "gradient desent" :

$$\omega := \omega - \alpha \frac{dJ(\omega)}{d\omega} \qquad \textit{PT (3.7)}$$

Ở đây, α là tỷ lệ học tập - một hằng số dương nhỏ, được đặt để kiểm soát kích thước của bước cập nhật.

Nếu chúng ta ở phía bên phải của biểu đồ hiển thị ở dưới, độ dốc sẽ dương. Sử dụng phương trình cập nhật, chúng ta sẽ di chuyển sang trái (tức là hướng xuống) cho đến khi đạt cực tiểu toàn cầu. Trong khi đó, nếu chúng ta ở phía bên trái, độ dốc sẽ âm và do đó chúng ta sẽ tiến một bước về phía bên phải (hướng xuống) cho đến khi đạt được cực tiểu toàn cầu.



Hình 3.6 Đồ thị hàm chi phí theo w

Các phương trình cập nhật cho các tham số w,b là:

$$\omega := \omega - \alpha \frac{dJ(\omega, b)}{d\omega} \qquad PT\ (3.8)$$

$$b := b - \alpha \frac{dJ(\omega, b)}{db} \qquad PT\ (3.9)$$

3.5 Các loại mạng nơ-ron nhân tạo

Mạng nơ-ron nhân tạo có thể được phân chia thành nhiều loại khác nhau, sử dụng cho những mục đích khác nhau. Một số đại diện mạng nơ-ron phổ biến, thường gặp có thể kể đến:

- Mạng Perceptron

Đây là mạng nơ-ron lâu đời nhất, do Frank Rosenblatt tạo ra vào năm 1958. Đây có một lớp nơ-ron duy nhất và là dạng mạng nơ-ron đơn giản nhất.

- Mạng nơ-ron truyền thẳng

Mạng nơ-ron truyền thẳng (Feedforward), hay còn gọi là mạng Perceptron nhiều lớp (MLP - Multi-Layer Perceptrons), là mạng nơ-ron phổ biến nhất. Chúng bao gồm một lớp đầu vào, một hoặc nhiều lớp ẩn và một lớp đầu ra.

Mạng nơ-ron truyền thẳng xử lý dữ liệu một chiều, từ lớp đầu vào đến lớp đầu ra. Mỗi nút trong cùng một lớp được kết nối với tất cả các nút trong lớp tiếp theo. Mạng truyền thẳng sử dụng một quy trình phản hồi để cải thiện dự đoán theo thời gian.

- Mạng nơ-ron tích chập

Hoạt động tương tự như mạng nơ-ron truyền thẳng, tuy nhiên, mạng nơ-ron tích chập (CNN - Convolutional Neural Networks) thường được sử dụng để nhận dạng hình ảnh, nhận dạng mẫu.

Những lớp ẩn trong mạng nơ-ron tích chập thực hiện chức năng toán học cụ thể, như tóm tắt hoặc sàng lọc, được gọi là tích chập. Loại mạng này khai thác nguyên tắc từ đại số tuyến tính, đặc biệt là phép nhân ma trận, để xác định các mẫu trong một hình ảnh.

- Mạng nơ-ron hồi quy

Mạng nơ-ron hồi quy (RNN - Recurrent Neural Networks) ra đời với ý tưởng sử dụng một bộ nhớ nhằm lưu lại thông tin từ những bước tính toán xử lý trước. Sau đó, dựa vào nó để đưa ra dự đoán và kết quả chính xác nhất.

Các thuật toán trong mạng này chủ yếu được sử dụng trong phân tích dữ liệu chuỗi thời gian để đưa ra dự đoán về kết quả trong tương lai, chẳng hạn như dự đoán tiếng nói, dự đoán thị trường chứng khoán hoặc dự báo doanh số bán hàng.

3.6 Mạng nơ-ron sử dụng

Đối với đề tài này, tôi sẽ sử dụng hai mô hình: mô hình nhận dạng từ kích hoạt và mô hình nhận dạng câu lệnh. Đặc điểm của hai mô hình được mô tả dưới bảng sau:

Bảng 3.1 Bảng so sánh hai mô hình

Yêu cầu	Nhận dạng từ kích hoạt	Nhận dạng câu lệnh
Khả năng biểu diễn	Ít phức tạp Đầu ra chỉ phân loại 2 lớp	Phức tạp Đầu ra phân loại nhiều lớp
Kích thước mô hình	Nhỏ, yêu cầu ít tài nguyên	Lớn, yêu cầu nhiều tài nguyên
Thời gian chạy	Liên tục	Không liên tục
Mạng nơ-ron đề xuất	CNN	LSTM

3.6.1 Mạng nơ-ron tích chập (CNN)

CNN là một loại mạng nơ-ron sử dụng rộng rãi trong lĩnh vực thị giác máy tính và xử lý ảnh [6]. CNN được thiết kế để tự động học và nhận dạng các đặc trưng từ dữ liệu đầu vào. Đặc điểm chính của CNN là khả năng học được các đặc trưng cục bộ, bằng cách sử dụng các bộ lọc để trích xuất thông tin từ các phần nhỏ của hình ảnh. Sau đó, thông tin này được tổng hợp và chuyển tiếp qua các lớp mạng để đưa ra dự đoán hoặc trích xuất thông tin.

Trong bài toán nhận dạng từ kích hoạt, CNN được sử dụng để nhận dạng từ kích hoạt từ đặc trưng tín hiệu âm thanh đầu vào: nhận dạng xem dữ liệu đầu vào có phải là của từ kích hoạt hay không. Lý do chính cho việc sử dụng CNN trong bài toán này là:

- Khả năng trích xuất đặc trưng từ dữ liệu âm thanh: CNN có khả năng học và nhận biết các đặc trưng quan trọng từ tín hiệu âm thanh như biên độ, tần số và thời gian. Điều này giúp cho việc nhận dạng từ kích hoạt trở nên hiệu quả.
- Xử lý dữ liệu không gian và thời gian: Tín hiệu âm thanh được coi như dữ liệu hai chiều trong không gian và CNN có khả năng xử lý dữ liệu hai chiều thông qua việc sử dụng các lớp Convolution và Pooling.
- Phân loại cục bộ: Các từ kích hoạt thường có những đặc điểm cục bộ trong tín hiệu âm thanh, ví dụ như các biên độ đột ngột hoặc các bước sóng cụ thể. CNN có khả năng học và nhận biết các đặc điểm này thông qua việc sử dụng các bộ lọc nhỏ và lớp tích chập.
- Hiệu suất và tính linh hoạt: CNN có thể được tinh chỉnh và điều chỉnh linh hoạt để phù hợp với các bài toán cụ thể, và có hiệu suất tốt khi áp dụng vào các bài toán nhận dạng âm thanh nhưng không quá phức tạp như bài toán nhận dạng từ kích hoạt .

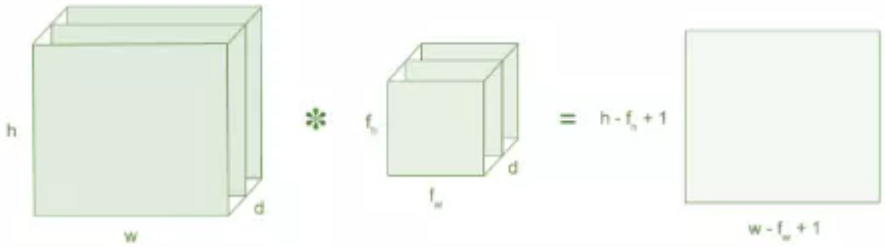
Với đặc trưng cho xử lý ảnh, CNN thường được thiết kế với đầu vào giống đầu vào của máy tính: máy tính coi hình ảnh đầu vào là 1 mảng pixel và nó phụ thuộc vào độ phân giải của hình ảnh. Dựa trên độ phân giải hình ảnh, máy tính sẽ thấy đầu vào có kích thước: H x W x D (H: Chiều cao, W: Chiều rộng, D: Độ dày). Ví dụ: Hình ảnh là mảng ma trận RGB 6x6x3 (3 ở đây là giá trị RGB).

Áp dụng với đầu vào là các đặc trưng MFCC của âm thanh, mỗi pixel sẽ giống với 1 đặc trưng MFCC mà chúng ta trích xuất được. Sau đây, tôi sẽ giới thiệu một số thành phần quan trọng của CNN:

a) Lớp tích chập (Convolution Layer)

Tích chập là lớp đầu tiên để trích xuất các tính năng từ đầu vào. Tích chập duy trì mối quan hệ giữa các đặc trưng MFCC bằng cách tìm hiểu các tính năng của các ô vuông nhỏ ở dữ liệu đầu vào. Đây là một 1 phép toán có 2 đầu vào: ma trận đặc trưng và bộ lọc (hoặc nhân).

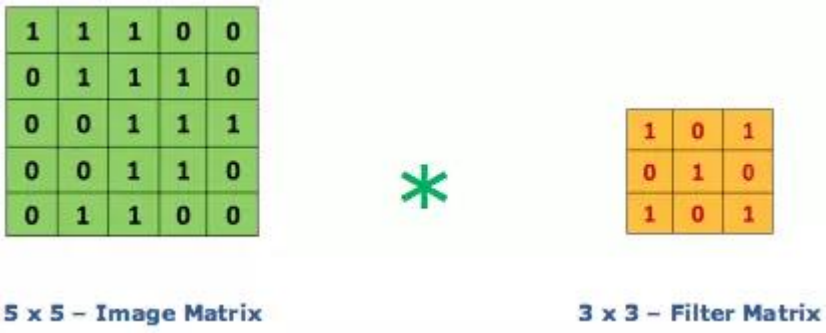
Hình 3.7 mô tả cách hoạt động của lớp tích chập: với đầu vào là ma trận có kích thước (h x w x d), qua một bộ lọc có kích thước (f_h x f_w x d) sẽ cho đầu ra có kích thước (h – f_h +1) x (w – f_w + 1) x 1.



Hình 3.7 Hoạt động của lớp tích chập

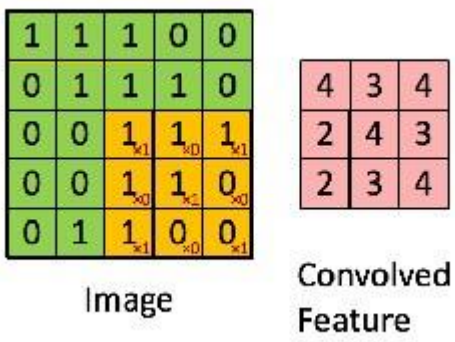
Với mỗi bộ lọc khác nhau ta sẽ học được những đặc trưng khác nhau của dữ liệu đầu vào. Nên trong thực tế, với mỗi lớp tích chập, ta thường dùng nhiều bộ lọc để học được nhiều thuộc tính của ma trận đầu vào. Vì mỗi bộ lọc cho đầu ra là 1 ma trận có kích thước (h – fh +1) x (w – fw + 1) nên k bộ lọc sẽ cho ra k ma trận đầu ra. Ta kết hợp k ma trận đầu ra này sẽ này được một ma trận 3 chiều có kích thước (h – f_h +1) x (w – f_w + 1) x k.

Xem xét 1 ma trận 5 x 5 có giá trị các ô là 0 và 1. Ma trận bộ lọc 3 x 3 như Hình 3.8



Hình 3.8 Ví dụ về cách hoạt động của lớp tích chập

Kết quả của lớp tích chập của ma trận đầu vào 5 x 5 nhân với ma trận bộ lọc (3x3) gọi là 'Convolved feature' như Hình 3.9.

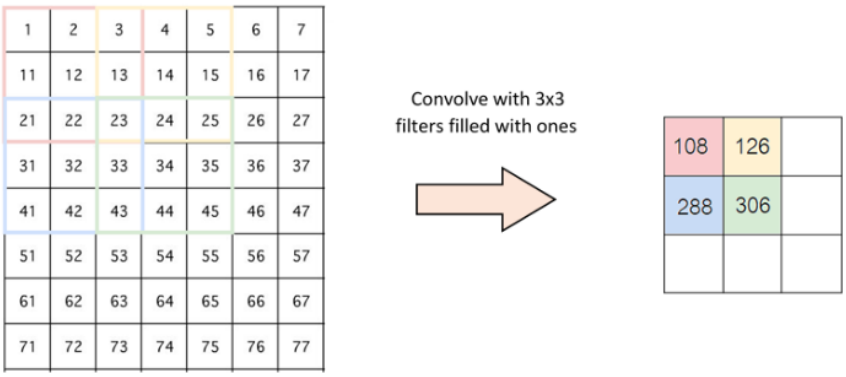


Hình 3.9 kết quả của ví dụ về cách hoạt động của lớp tích chập

Sự kết hợp của 1 ma trận đầu vào với các bộ lọc khác nhau có thể thực hiện làm nổi bật các đặc trưng khác nhau của dữ liệu. Về giá trị của ma trận, thường được khởi tạo ngẫu nhiên trong quá trình huấn luyện mạng nơ-ron và được cập nhật theo cách hiệu quả nhất để mạng có thể học được các đặc trưng từ dữ liệu. Những giá trị này thường bắt đầu từ các giá trị ngẫu nhiên và được cập nhật trong quá trình lan truyền ngược trong quá trình huấn luyện mạng.

b) Bước nhảy (Stride)

Bước nhảy là số phần tử thay đổi trên ma trận đầu vào. Khi bước nhảy là 1 thì ta di chuyển bộ lọc 1 phần tử. Khi bước nhảy là 2 thì ta di chuyển bộ lọc đi 2 phần tử và tiếp tục như vậy. Hình 3.10 là lớp tích chập hoạt động với bước nhảy là 2, bộ lọc là ma trận (3x3), các phần tử đều có giá trị 1.



Hình 3.10 Ví dụ về cách hoạt động của bước nhảy

c) Đệm (Padding)

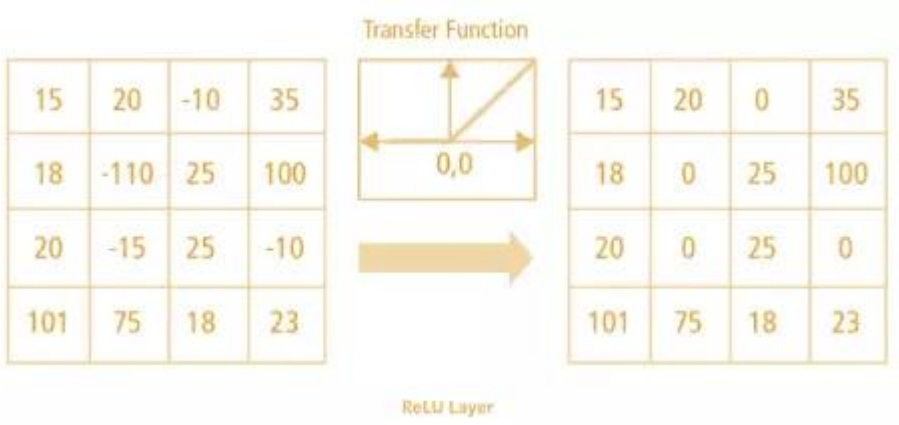
Đôi khi bộ lọc không phù hợp với kích thước ma trận đầu vào. Ta có hai lựa chọn:

- Chèn thêm các số 0 vào 4 đường biên của ma trận (đệm).
- Cắt bớt ma trận đầu vào tại những điểm không phù hợp với bộ lọc.

d) Hàm phi tuyến (ReLu)

ReLU viết tắt của Rectified Linear Unit, là 1 hàm phi tuyến. Với đầu ra là: $f(x) = \max(0, x)$.

Như đã trình bày ở Hàm kích hoạt, hàm ReLu giúp mô hình học được các đặc trưng phi tuyến, ngoài ra nó còn phù hợp với đặc điểm của dữ liệu: dữ liệu trong thế giới mà chúng ta tìm hiểu là các giá trị tuyến tính không âm.



Hình 3.11 Hoạt động của hàm phi tuyến ReLu

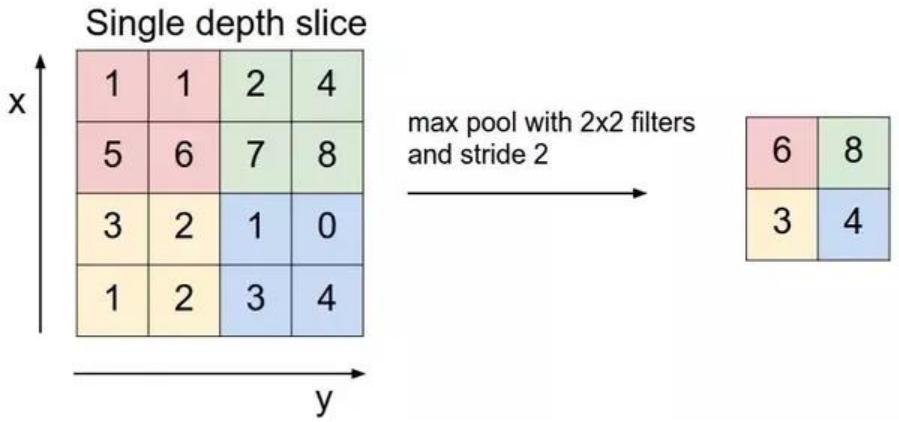
Có 1 số hàm phi tuyến khác như tanh, sigmoid cũng có thể được sử dụng thay cho ReLU. Nhưng hầu hết người ta thường dùng ReLU vì nó có hiệu suất tốt.

e) Lớp gộp (Pooling Layer)

Lớp gộp sẽ giảm bớt số lượng tham số khi đầu vào quá lớn. Không gian gộp còn được gọi là lấy mẫu con hoặc lấy mẫu xuống làm giảm kích thước của mỗi vùng nhưng vẫn giữ lại thông tin quan trọng. Lớp gộp có thể có nhiều loại khác nhau:

- Max Pooling: giá trị lớn nhất trong vùng được chọn làm giá trị đại diện cho vùng đó.
- Average Pooling: giá trị trung bình trong vùng được chọn làm giá trị đại diện cho vùng đó.
- Sum Pooling: tổng của các giá trị trong mỗi vùng được chọn làm giá trị đại diện cho vùng đó.

Hình 3.12 miêu tả về cách hoạt động của Max Pooling với bộ lọc (2x2) và bước nhảy là 2:

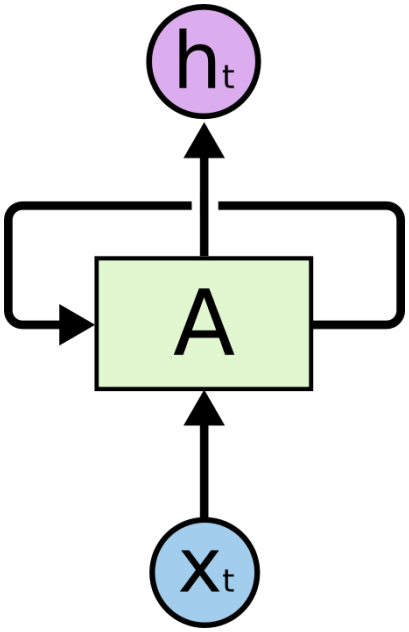


Hình 3.12 Cách hoạt động của Max Pooling

3.6.2 Mạng trí nhớ ngắn hạn định hướng dài hạn (LSTM)

a) Tìm hiểu chung về mạng nơ-ron hồi quy (RNN)

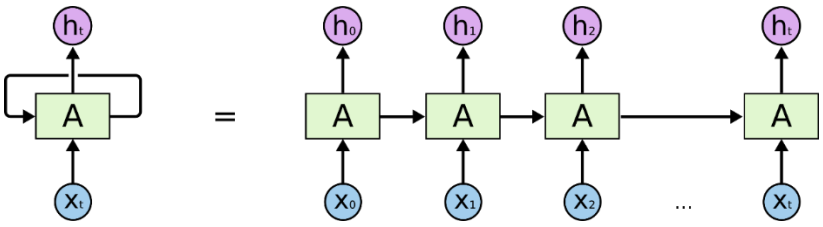
Trong lý thuyết về ngôn ngữ, ngữ nghĩa của một câu được tạo thành từ mối liên kết của những từ trong câu theo một cấu trúc ngữ pháp. Nếu xét từng từ một đứng riêng lẻ ta không thể hiểu được nội dụng của toàn bộ câu, nhưng dựa trên những từ xung quanh ta có thể hiểu được trọn vẹn một câu nói. Như vậy cần phải có một kiến trúc đặc biệt hơn cho các mạng nơ-ron biểu diễn ngôn ngữ nhằm mục đích liên kết các từ liền trước với các từ ở hiện tại để tạo ra mối liên hệ xâu chuỗi. Mạng nơ-ron hồi quy (RNN) đã được thiết kế đặc biệt để giải quyết yêu cầu này [7]:



Hình 3.13 Mạng nơ-ron truy hồi với vòng lặp

Hình 3.13 biểu diễn kiến trúc của một mạng nơ-ron hồi quy. Trong kiến trúc này mạng nơ-ron sử dụng một đầu vào là một véc-tơ x_t và trả ra đầu ra là một giá trị ẩn h_t. Đầu vào được đấu với một thân mạng nơ-ron (A) có tính chất hồi quy và thân này được đấu tới đầu ra h_t.

Vòng lặp ở thân mạng nơ-ron là điểm mấu chốt trong nguyên lý hoạt động của mạng nơ-ron hồi quy. Đây là chuỗi sao chép nhiều lần của cùng một kiến trúc nhằm cho phép các thành phần có thể kết nối liền mạch với nhau theo mô hình chuỗi. Đầu ra của vòng lặp trước chính là đầu vào của vòng lặp sau. Nếu trải phẳng thân mạng nơ-ron ta sẽ thu được một mô hình dạng:



Hình 3.14 Cấu trúc trải phẳng của mạng nơ-ron truy hồi

Kiến trúc mạng nơ-ron hồi quy này tỏ ra khá thành công trong các tác vụ của học sâu như: Nhận diện giọng nói, các mô hình ngôn ngữ, mô hình dịch, chú thích hình ảnh …

Hạn chế của mạng nơ-ron hồi quy

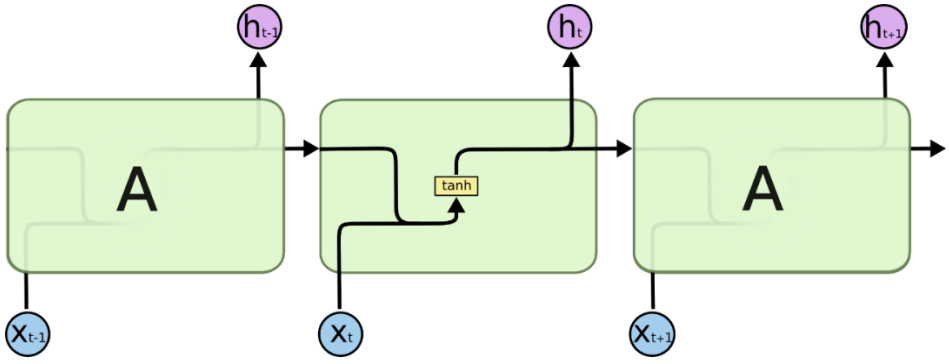
Một trong những điểm đặc biệt của RNN đó là nó có khả năng kết nối các thông tin liền trước với nhiệm vụ hiện tại, chẳng hạn như trong câu văn: 'Học sinh đang tới trường học'. Dường như trong một ngữ cảnh ngắn hạn, từ trường học có thể được dự báo ngay tức thì mà không cần thêm các thông tin từ những câu văn khác gần đó. Tuy nhiên có những tình huống đòi hỏi phải có nhiều thông tin hơn chẳng hạn như: 'hôm qua Bống đi học nhưng không mang áo mưa. Trên đường đi học trời mưa. Cặp sách của Bống bị ướt'. Chúng ta cần phải học để tìm ra từ ướt ở một ngữ cảnh dài hơn so với chỉ 1 câu. Tức là cần phải biết các sự kiện trước đó như trời mưa, không mang áo mưa để suy ra sự kiện bị ướt. Những sự liên kết ngữ nghĩa dài như vậy được gọi là phụ thuộc dài hạn (Long-term Dependencies). Về mặt lý thuyết mạng RNN có thể giải quyết được những sự phụ thuộc trong dài hạn. Tuy nhiên trên thực tế RNN lại cho thấy khả năng học trong dài hạn kém hơn. Một trong những nguyên nhân chính được giải thích đó là sự triệt tiêu đạo hàm của hàm chi phí sẽ diễn ra khi trải quả chuỗi dài các tính toán truy hồi. Do đó, một phiên bản mới của mạng RNN là Mạng trí nhớ ngắn hạn định hướng dài hạn (LSTM) ra đời nhằm khắc phục hiện tượng này nhờ một cơ chế đặc biệt.

Đó cũng là lý do tôi quyết định lựa chọn Mạng trí nhớ ngắn hạn định hướng dài hạn (LSTM) trong bài toán nhận dạng câu lệnh này.

b) Tìm hiểu chung về Mạng trí nhớ ngắn hạn định hướng dài hạn (LSTM - Long short-term memory)

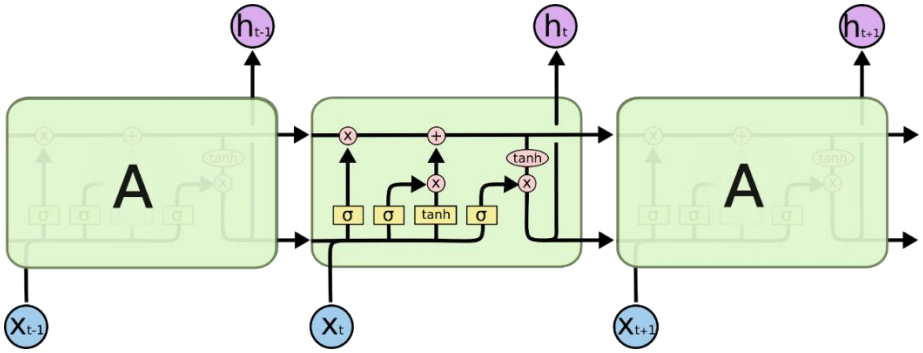
Mạng trí nhớ ngắn hạn định hướng dài hạn còn được viết tắt là LSTM làm một kiến trúc đặc biệt của RNN có khả năng học được sự phục thuộc trong dài hạn được giới thiệu bởi Hochreiter & Schmidhuber (1997). Kiến trúc này đã được phổ biến và sử dụng rộng rãi cho tới ngày nay. LSTM đã tỏ ra khắc phục được rất nhiều những hạn chế của RNN trước đây về triệt tiêu đạo hàm. Tuy nhiên cấu trúc của mạng này có phần phức tạp hơn mặc dù vẫn dữ được tư tưởng chính của RNN là sự sao chép các kiến trúc theo dạng chuỗi.

Một mạng RNN tiêu chuẩn sẽ có kiến trúc rất đơn giản chẳng hạn như đối với kiến trúc gồm một tầng ẩn là hàm tanh như Hình 3.15:

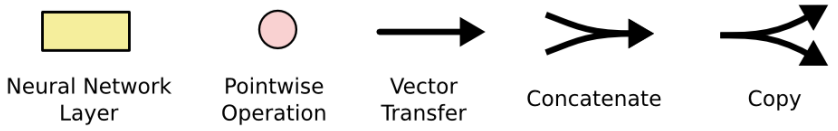


Hình 3.15 Sự lặp lại kiến trúc trong mạng RNN

LSTM cũng có một chuỗi dạng như thế nhưng phần kiến trúc lặp lại có cấu trúc khác biệt hơn. Thay vì chỉ có một tầng đơn, chúng có tới 4 tầng ẩn (3 sigmoid và 1 tanh) tương tác với nhau theo một cấu trúc đặc biệt.



Hình 3.16 Sự lặp lại kiến trúc trong mạng LSTM



Hình 3.17 Diễn giải các kí hiệu trong đồ thị mạng nơ-ron

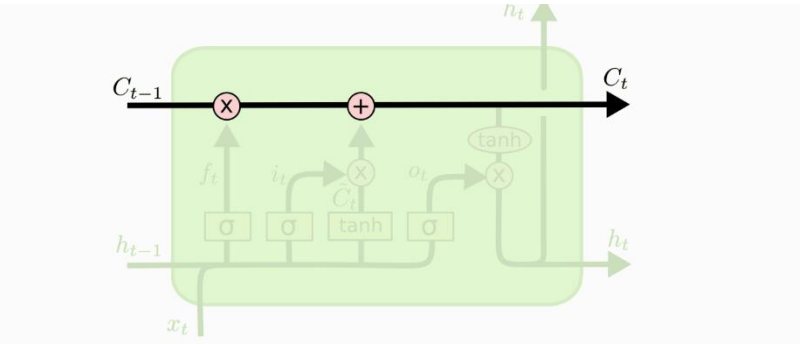
Một số khái niệm quan trọng trong mô hình:

- Tensor là một khái niệm cơ bản và quan trọng, đặc biệt là trong việc biểu diễn và xử lý dữ liệu đa chiều. Tensor là một cấu trúc dữ liệu đa chiều (n-dimensional array) có thể lưu trữ dữ liệu số học, như số vô hướng (0-D tensors), véc-tơ (1-D tensors), ma trận (2-D tensors), hoặc các cấu trúc đa chiều khác.
- Neural Network Layer: Đây là các lớp mạng nơ-ron thông thường, được sử dụng để thực hiện các phép tính tuyến tính và phi tuyến tính trên dữ liệu đầu vào. Trong LSTM, các Neural Network Layer thường được sử dụng để biến đổi và kết hợp dữ liệu từ các thời điểm trước đó hoặc từ các phần của dữ liệu đầu vào.
- Pointwise Operation: Đây là một phép tính được thực hiện độc lập trên từng cặp phần tử của hai tensor cùng kích thước. Trong LSTM, các Pointwise Operation thường được sử dụng để thực hiện các phép tính như phép cộng, phép nhân vô hướng hoặc các phép tính phi tuyến tính như hàm kích hoạt trên các đầu vào.
- Vector Transfer: Đây là quá trình chuyển đổi dữ liệu từ một không gian véc-tơ này sang một không gian véc-tơ khác thông qua một phép biến đổi tuyến tính hoặc phi tuyến tính. Trong LSTM, Vector Transfer thường được sử dụng để biến đổi các véc-tơ đầu vào hoặc kết quả từ các neural network layer thành các véc-tơ mới có kích thước hoặc đặc tính khác.
- Concatenate: Đây là phép nối các véc-tơ hoặc tensor lại với nhau theo một trục cụ thể. Trong LSTM, phép nối được sử dụng để kết hợp thông tin từ nhiều nguồn khác nhau, chẳng hạn như thông tin từ các thời điểm trước đó hoặc từ các phần khác nhau của dữ liệu đầu vào.

- Copy: Trong LSTM, việc sao chép thông tin từ các bước thời gian trước đó được thực hiện thông qua các cơ chế đặc biệt trong kiến trúc LSTM, chẳng hạn như cơ chế cổng quên (forget gate) và cơ chế cập nhật (update gate), giúp mạng giữ lại hoặc loại bỏ thông tin từ quá khứ tùy thuộc vào nhu cầu của nhiệm vụ cụ thể.

c) Cách hoạt động của LSTM

Ý tưởng chính của LSTM là thành phần ô trạng thái (cell state) được thể hiện qua đường chạy ngang qua đỉnh đồ thị như trong Hình 3.18:

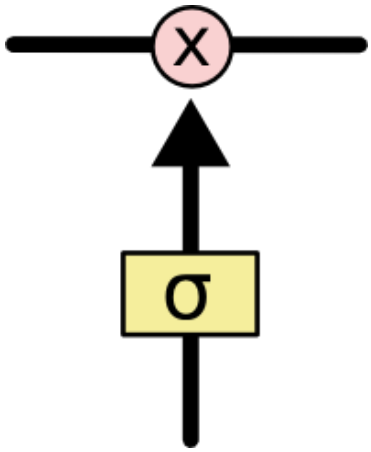


Hình 3.18 Đường đi của ô trạng thái trong mạng LSTM

Ô trạng thái là một dạng băng chuyền chạy thẳng xuyên suốt toàn bộ chuỗi với chỉ một vài tương tác tuyến tính nhỏ giúp cho thông tin có thể truyền dọc theo đồ thị mạng nơ ron ổn định.

LSTM có khả năng xóa và thêm thông tin vào ô trạng thái và điều chỉnh các luồng thông tin này thông qua các cấu trúc gọi là cổng.

Cổng là cơ chế đặc biệt để điều chỉnh luồng thông tin đi qua. Chúng được tổng hợp bởi một tầng ẩn của hàm kích hoạt sigmoid và với một toán tử nhân như trong Hình 3.19:



Hình 3.19 Một cổng của hàm sigmoid trong LSTM

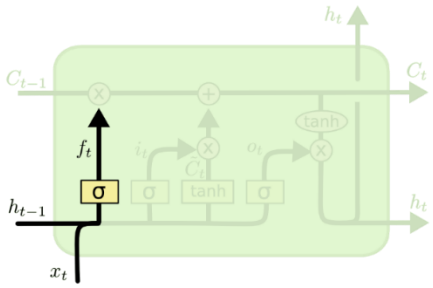
Hàm sigmoid sẽ cho đầu ra là một giá trị xác xuất nằm trong khoảng từ 0 đến 1, thể hiện rằng có bao nhiêu phần thông tin sẽ đi qua cổng. Giá trị bằng 0 ngụ ý rằng không cho phép thông tin nào đi qua, giá trị bằng 1 sẽ cho toàn bộ thông tin đi qua.

Một mạng LSTM sẽ có 3 cổng có kiến trúc dạng này để bảo vệ và kiểm soát các ô trạng thái.

d) Thứ tự các bước của LSTM

Bước đầu tiên trong LSTM sẽ quyết định xem thông tin nào sẽ được phép đi qua ô trạng thái. Nó được kiểm soát bởi hàm sigmoid trong một tầng gọi là tầng quên (forget gate layer). Đầu tiên nó nhận đầu vào là 2 giá trị: trạng thái ẩn h_{t-1} và dữ liệu đầu vào x_t, trả về một giá trị nằm trong khoảng 0 và 1 cho mỗi giá trị của ô trạng thái C_{t-1}. Nếu giá trị bằng 1 thể hiện 'giữ toàn bộ thông tin' và bằng 0 thể hiện 'bỏ qua toàn bộ chúng'.

Trở lại ví dụ về ngôn ngữ, chúng ta đang cố gắng dựa vào mối quan hệ giữa đặc trưng MFCC của các khung liên tiếp. Trong những bài toán như vậy, ô trạng thái có thể bao gồm tính chất của khung hiện tại, để cho khung tiếp theo được sử dụng chính xác. Một ví dụ dễ hiểu đối với ngôn ngữ: chúng ta đang mô tả về 'chủ ngữ' là một người bạn là con trai thì các đại từ nhân xưng ở tiếp theo phải là anh, chú thay vì cô, chị ấy. Tuy nhiên chủ ngữ không phải khi nào cũng cố định. Khi có một 'chủ ngữ' mới, chúng ta muốn quên đi loại của 'chủ ngữ' cũ. Do đó tầng quên cho phép cập nhật thông tin mới và lưu giữ giá trị của nó khi có thay đổi theo thời gian.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Hình 3.20 Tầng cổng quên

Với hàm kích hoạt sigmoid, W_f là các tham số và b là các hệ số điều chỉnh, hàm f_t có công thức như sau:

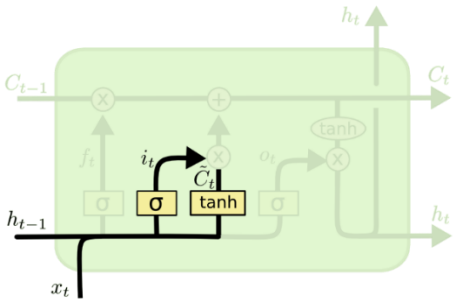
$$f_t = \; s(W_f \cdot [h_{t-1}, x_t] + \; b_f) \qquad \textit{PT (3.10)}$$

Bước tiếp theo sẽ quyết định loại thông tin nào sẽ được lưu trữ trong ô trạng thái. Bước này bao gồm 2 phần:

- Phần đầu tiên là một tầng ẩn của hàm sigmoid được gọi là tầng cổng vào quyết định giá trị bao nhiêu sẽ được cập nhật.
- Sau đó, tầng ẩn hàm tanh sẽ tạo ra một véc-tơ của một giá trị trạng thái mới $\tilde{C}_t$ mà có thể được thêm vào trạng thái.

Tiếp theo kết hợp kết quả của 2 tầng này để tạo thành một cập nhật cho trạng thái.

Trong ví dụ của mô hình ngôn ngữ, chúng ta muốn thêm loại của một chủ ngữ mới vào ô trạng thái để thay thế phần trạng thái cũ muốn quên đi.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Hình 3.21 Cập nhật giá trị cho ô trạng thái bằng cách kết hợp 2 kết quả từ tầng cổng vào và tầng ẩn hàm tanh

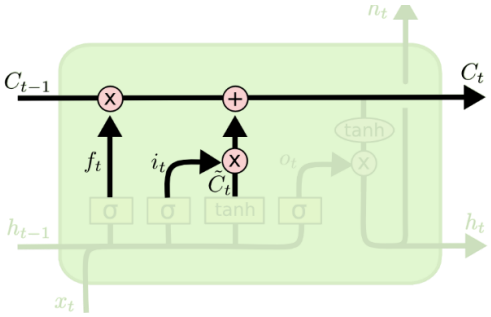
Với các hàm i_t và $\tilde{C}_t$ có công thức như sau:

$$i_t = \; s(W_i \cdot [h_{t-1}, x_t] + \; b_i) \qquad \textit{PT (3.11)}$$

$$\tilde{C}_t = \; tanh(W_C \cdot [h_{t-1}, x_t] + \; b_C) \qquad \textit{PT (3.12)}$$

Đây là thời điểm để cập nhật một ô trạng thái cũ C_{t-1} sang một trạng thái mới C_t. Những bước trước đó đã quyết định làm cái gì, tại bước này chỉ cần thực hiện nó.

Chúng ta nhân trạng thái cũ với f_t tương ứng với việc quên những thứ quyết định được phép quên sớm. Phần tử đề cử $i_t * \tilde{C}_t$ là một giá trị mới được tính toán tương ứng với bao nhiêu được cập nhật vào mỗi giá trị trạng thái.



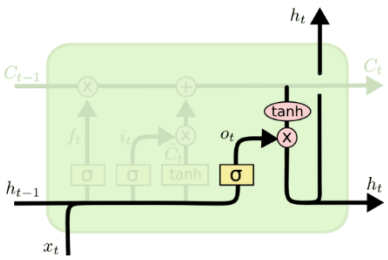
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Hình 3.22 Ô trạng thái mới

Với hàm C_t có công thức như sau:

$$C_t = \; f_t * C_{t-1} + i_t * \tilde{C}_t \qquad \textit{PT (3.13)}$$

Cuối cùng cần quyết định xem đầu ra sẽ trả về bao nhiêu. Kết quả ở đầu ra sẽ dựa trên ô trạng thái, nhưng sẽ là một phiên bản được lọc. Đầu tiên, chúng ta chạy qua một tầng sigmoid nơi quyết định phần nào của ô trạng thái sẽ ở đầu ra. Sau đó, ô trạng thái được đưa qua hàm tanh (để chuyển giá trị về khoảng -1 và 1) và nhân nó với đầu ra của một cổng sigmoid, do đó chỉ trả ra phần mà chúng ta quyết định.



$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Hình 3.23 Điều chỉnh thông tin ở đầu ra thông qua hàm tanh

Với các hàm 0_t và h_t có công thức như sau:

$$o_t = s(W_0 \cdot [h_{t-1}, x_t] + b_0) \quad \textit{PT (3.14)}$$

$$\tilde{C_t} = o_t * tanh(C_t) \quad \textit{PT (3.15)}$$

3.7 Kết luận chương

Như vây, qua chương này, chúng ta đã tìm hiểu được cơ sở lý thuyết về học sâu và hai mạng nơ-ron sử dụng trong đề tài cũng như lý do lựa chọn chúng: CNN phù hợp với bài toán nhận dạng từ kích hoạt yêu cầu kích thước mô hình nhỏ, ít phức tạp, ít tài nguyên, cần chạy liên tục; LSTM phù hợp với bài toán nhận dạng câu lệnh yêu cầu kích thức mô hình lớn, phức tạp, nhiều tài nguyên và không cần chạy liên tục. Ở chương tiếp theo, tôi sẽ đi vào xây dựng mô hình cho hai bài toán nhận dạng trên.

CHƯƠNG 4. XÂY DỰNG MÔ HÌNH NHẬN DẠNG TIẾNG NÓI

4.1 Đề xuất xây dựng cơ sở dữ liệu

Cơ sở dữ liệu đóng vai trò quan trọng trong học sâu bởi vì dữ liệu là yếu tố chính để huấn luyện và đánh giá các mô hình. Tôi đã tiến hành thu tập dữ liệu với các thành phần như sau:

- Từ kích hoạt: Wiki ơi.
- Câu lệnh: Bật điều hoà, tắt điều hoà, tăng 1 độ, giảm 1 độ, bật 26 độ. (Đây là những chức năng cơ bản và phổ biến của điều khiển điều hoà, ngưỡng 26 độ là ngưỡng thường được bật nhất).
- Nhiễu: để phân biệt với từ kích hoạt và câu lệnh, được thu từ môi trường thực tế ở phòng, công ty và trường.

Tập dữ liệu trên được thu 60 người (40 nam, 20 nữ), trong độ tuổi từ 20-30, tất cả đều là người Hà Tĩnh hoặc Nghệ An. Mỗi câu lệnh sẽ được mỗi người thu 5 lần bằng phần mềm Audacity với:

- Micrô: micrô trực tiếp của máy tính, gồm khoảng 30 máy tính khác nhau.
- Độ dài: câu kích hoạt:1s, câu lệnh: 1.5s.
- Chế độ âm thanh: mono (1 kênh).
- Tần số lấy mẫu: 16000Hz.
- Độ sâu bit: 32-bit float.

4.2 Tăng cường dữ liệu âm thanh

Trong Học sâu, tăng cường dữ liệu là một kỹ thuật quan trọng để cải thiện hiệu suất của mô hình bằng cách tạo ra thêm dữ liệu huấn luyện từ dữ liệu ban đầu thông qua các biến đổi hoặc biến đổi ngẫu nhiên. Tăng cường dữ liệu có nhiều vai trò như:

- Chống overfitting: overfitting là hiện tượng mô hình học quá mức từ dữ liệu huấn luyện và không thể tổng quát hóa tốt cho dữ liệu mới. Tăng cường dữ liệu có thể giảm thiểu hiện tượng này bằng cách giúp mô hình học được biểu diễn tổng quát hóa hơn với các dữ liệu mới.
- Tăng cường độ chính xác: Bằng cách tăng cường dữ liệu, mô hình được huấn luyện trên một lượng lớn hơn các trường hợp có thể xảy ra trong thực tế, giúp nó học được các đặc điểm chung hơn và tránh "overfitting".
- Tăng tính đa dạng của dữ liệu: Bằng cách áp dụng các biến đổi và biến đổi ngẫu nhiên, tăng cường dữ liệu có thể tạo ra các biến thể của dữ liệu gốc, tăng tính đa dạng của dữ liệu huấn luyện và giúp mô hình học được các biểu diễn phong phú hơn về các đối tượng trong dữ liệu.
- Tiết kiệm chi phí thu thập dữ liệu: Thay vì thu thập thêm dữ liệu mới, tăng cường dữ liệu cho phép sử dụng lại dữ liệu hiện có một cách hiệu quả và tiết kiệm chi phí.
- Cải thiện hiệu suất trên dữ liệu nhỏ: Trong các tình huống mà lượng dữ liệu huấn luyện có hạn, tăng cường dữ liệu có thể giúp cải thiện hiệu suất của mô hình bằng cách tạo ra thêm dữ liệu huấn luyện từ dữ liệu có sẵn.

Nói chung, tăng cường dữ liệu đóng vai trò quan trọng trong việc cải thiện hiệu suất và khả năng tổng quát hóa của mô hình học sâu bằng cách tạo ra thêm dữ liệu đa dạng và phong phú từ dữ liệu huấn luyện ban đầu. Đối với đề tài của mình, tôi sử dụng thư viện 'audiomentations' [8] – một thư viện phổ biến để tăng cường dữ liệu trong xử lý âm thanh. Sau đây là 11 hiệu ứng tôi sử dụng trong đề tài của mình:

- NoiseInjection: Thêm nhiễu vào âm thanh, giúp mô hình học được khả năng chịu đựng với nhiễu và tăng cường tổng quát hóa.
- PitchShifting: Thay đổi tần số của âm thanh một cách ngẫu nhiên, giúp mô hình học được các biến thể của âm thanh và tổng quát hóa tốt hơn.
- TimeStretching: Mở rộng hoặc thu ngắn thời gian của âm thanh một cách ngẫu nhiên, giúp tăng tính đa dạng của dữ liệu.
- Padding: Thêm dữ liệu giả vào âm thanh, đảm bảo kích thước của âm thanh đầu vào đủ lớn để đạt được kích thước mong muốn cho quy trình xử lý.
- BandPassFilter: Áp dụng bộ lọc qua băng để lọc ra các tần số trong một phạm vi nhất định, tăng cường tính đa dạng của dữ liệu âm thanh.
- Gain: Thay đổi độ lớn của âm thanh, giúp mô hình học được ổn định hơn với các cường độ âm thanh khác nhau.
- TimeMask: Loại bỏ một phần của âm thanh trong thời gian ngẫu nhiên, giúp mô hình học được ổn định hơn với sự thiếu dữ liệu.
- HighPassFilter: Áp dụng bộ lọc qua băng để lọc ra các tần số cao hơn một ngưỡng nhất định.
- ClippingDistortion: Tạo ra hiệu ứng méo âm, giúp mô hình học được ổn định hơn với các biến thể của âm thanh.
- AirAbsorption: Mô phỏng hiệu ứng hấp thụ không khí, giúp mô hình học được ổn định hơn với các điều kiện môi trường khác nhau.
- Trim: Loại bỏ các phần âm thanh có độ lớn nhỏ hơn một ngưỡng nhất định, giúp làm sạch dữ liệu và tăng cường khả năng tổng quát hóa.

Như vậy, sau bước tăng cưỡng dữ liệu với 11 hiệu ứng, tôi đã tạo ra tập dữ liệu mới có số lượng mẫu gấp 12 lần tập dữ liệu ban đầu.

```python
def get_augs(aug):
    augmentations = []
    if aug == 'NoiseInjection':
        augmentations.append(AddGaussianNoise(min_amplitude=0.001, max_amplitude=0.0014, p=1.0))
    elif aug == 'PitchShifting':
        augmentations.append(PitchShift(min_semitones=-2.5, max_semitones=2.5, p=1.0))
    elif aug == 'TimeStretching':
        augmentations.append(TimeStretch(min_rate=1.04, max_rate=1.05, p=1.0))
    elif aug == 'Padding':
        augmentations.append(Padding(min_fraction=0.04,max_fraction=0.05, p=1.0))
    elif aug == 'BandPassFilter':
        augmentations.append(BandPassFilter(min_center_freq=2200, max_center_freq=2400, p=1.0))
    elif aug == 'Gain':
      augmentations.append(Gain(min_gain_in_db=-10, max_gain_in_db=10,p=1.0))
    elif aug == 'TimeMask':
      augmentations.append(TimeMask(min_band_part=0.14, max_band_part=0.16,p=1.0))
    elif aug == 'HighPassFilter':
      augmentations.append(HighPassFilter(min_cutoff_freq=1600, max_cutoff_freq=1800, p=1))
    elif aug == 'ClippingDistortion':
      augmentations.append(ClippingDistortion(min_percentile_threshold =0, max_percentile_threshold=8, p=1))
    elif aug == 'AirAbsorption':
      augmentations.append(AirAbsorption(min_distance=10.0,max_distance=40.0,p=1.0,))
    elif aug == "Trim":
      augmentations.append(Trim(top_db=30.0,p=1.0))
    return augmentations
```

Hình 4.1 Các hiệu ứng tăng cường dữ liệu âm thanh

4.3 Trích xuất đặc trưng MFCC

Do trong xử lý âm thanh, ngữ cảnh rất quan trọng do đó, mỗi cửa sổ tín hiệu cần được chia thành 3 trạng thái: nối với âm trước, âm của chính nó, nối với âm sau. Mỗi giây, trung bình một người nói khoảng 3 từ, mỗi từ có 4 âm, mỗi âm chia ra 3 trạng thái nên cần phân biệt 36 trạng thái trong 1s. Vậy độ dài mỗi cửa sổ tín hiệu bằng 25ms là hợp lý. Các cửa sổ cũng cần xếp chồng với nhau một khoảng 10ms để lưu trữ thông tin.

Để trích xuất đặc trưng MFCC tôi sử dụng thư viện librosa [9]:

```python
def features_extractor(file_name):
    audio,sample_rate  = librosa.load(file_name, sr=16000, mono=True)
    frame_length = int(0.025 * sample_rate)
    hop_length = int(0.015 * sample_rate)
    n_fft = 2 ** int(np.ceil(np.log2(frame_length)))
    mfcc = librosa.feature.mfcc(y=audio, sr=sample_rate, n_fft=n_fft, hop_length=hop_length, n_mfcc=13)
    desired_size = 100
    mfccs = np.zeros((13,desired_size))
    mfccs[:, :min(desired_size, mfcc.shape[1])] = mfcc[:, :min(desired_size, mfcc.shape[1])]
    delta_mfccs = librosa.feature.delta(mfccs)
    delta2_mfccs = librosa.feature.delta(mfccs, order=2)
    mfccs_features = np.concatenate((mfccs, delta_mfccs, delta2_mfccs))
    mfccs_features = np.transpose(mfccs_features)

    return mfccs_features
```

Hình 4.2 Trích xuất đặc trưng MFCC qua thư viện Librosa

Với:

- file_name: Tên của tệp âm thanh cần trích xuất đặc trưng MFCC.
- librosa.load(file_name, sr=16000, mono=True): Hàm "load" từ thư viện librosa được sử dụng để tải tệp âm thanh từ "file_name". Tham số "sr=16000" chỉ định tỷ lệ mẫu là 16000 Hz, tức là số mẫu âm thanh được thu trong một giây. Tham số "mono=True" chỉ định rằng âm thanh sẽ được tải dưới dạng một kênh.
- frame_length: Độ dài của khung (frame) trong quá trình trích xuất đặc trưng MFCC, tính bằng số mẫu. Như đã giải thích ở trên, ta sẽ lấy độ dài khung như sau: frame_length = 25ms.

- hop_length: Khoảng cách giữa các khung liên tiếp, cũng được tính bằng số mẫu. Thông thường, khoảng cách này nhỏ hơn độ dài của khung để có sự chồng lấn giữa các khung, giúp giảm thiểu mất mát thông tin. Ta có thể tính thời gian xếp chồng: t = frame_length - hop_length = 10 ms.
- n_fft: Số mẫu được sử dụng cho mỗi phần tần số trong biến đổi Fourier (FFT). Trong trường hợp này, n_fft được thiết lập sao cho nó lớn hơn hoặc bằng frame_length và là một số mũ của 2 gần với frame_length. ở đây ta có: n_fft = 512.
- n_mfcc: Số lượng hệ số MFCC cần trích xuất – 13 hệ số MFCC.
- librosa.feature.mfcc(y=audio,sr=sample_rate,n_fft=n_fft,hop_length=hop_length,n_mfcc=13): Hàm MFCC từ thư viện Librosa được sử dụng để trích xuất các hệ số MFCC từ tệp âm thanh audio với các thông số như sr, n_fft, hop_length, và n_mfcc đã tính toán ở trên.
- desired_size: Kích thước mong muốn cho đặc trưng MFCCs sau khi đã trích xuất. Với đầu vào là tệp âm thanh dài t (s). Ta có công thức tính số khung:

$$desired\_size = \left[\frac{t*1000-frame\_length}{hop\_length}\right]+1 \qquad PT\ (4.1)$$

  Áp dụng công thức trên, với mô hình nhận dạng từ kích hoạt, đầu vào là tệp âm thanh dài 1s sẽ có "desired_size = 66"; mô hình nhận dạng câu lệnh có đầu vào là tệp âm thanh dài 1.5s sẽ có "desired_size = 100".
- np.zeros((13,desired_size)): Tạo một mảng numpy có kích thước (13, desired_size) với tất cả các phần tử được thiết lập thành 0.0. Đây sẽ là ma trận cho các hệ số MFCCs.
- mfccs[:, :min(desired_size, mfcc.shape[1])] = mfcc[:, :min(desired_size, mfcc.shape[1])] được sử dụng để cắt hoặc mở rộng ma trận MFCC sao cho nó có kích thước mong muốn desired_size. Nếu mfcc.shape[1] (số lượng cột trong ma trận MFCC ban đầu) lớn hơn desired_size, thì chỉ một phần của ma trận MFCC ban đầu sẽ được sử dụng. Ngược lại, nếu mfcc.shape[1] nhỏ hơn desired_size, thì ma trận MFCC sẽ được đẩy vào một phần của mfccs với kích thước mới được mở rộng lên desired_size. Kết quả sẽ tạo ra một ma trận có kích thước (13, desired_size).
- librosa.feature.delta(mfccs): Hàm delta từ librosa.feature được sử dụng để tính toán độ biến đổi của các hệ số MFCC theo thời gian.
- librosa.feature.delta(mfccs, order=2): Hàm delta được gọi lần nữa với tham số order=2, để tính toán độ biến đổi bậc hai của các hệ số MFCC theo thời gian.
- np.concatenate((mfccs, delta_mfccs, delta2_mfccs)): Hàm concatenate từ thư viện numpy được sử dụng để nối các ma trận MFCC, delta MFCC, và delta bậc hai MFCC thành một ma trận duy nhất – ma trận có kích thước (39, desired_size).
- np.transpose(mfccs_features): chuyển đổi ma trận mfccs_features để có được ma trận kết quả cuối cùng, trong đó mỗi hàng đại diện cho một khung và mỗi cột đại diện cho một đặc trưng.

- return mfccs_features: Trả về ma trận đặc trưng MFCCs đã được chuẩn bị - ma trận có kích thước (desired_size,39).

4.4 Mô hình đề xuất

4.4.1 Mô hình nhận dạng từ kích hoạt

Đầu vào của lớp Conv2D là một tensor 4D có kích thước: (số lượng mẫu, chiều cao, chiều rộng, kênh). Trong đó:

- Số lượng mẫu (batch_size): Số lượng mẫu dữ liệu trong mỗi lần đưa vào mạng để huấn luyện hoặc dự đoán.
- Chiều cao (height): Chiều cao của tensor đầu vào.
- Chiều rộng (width): Chiều rộng của tensor đầu vào.
- Kênh (channels): Số lượng kênh tensor đầu vào.

Với đầu vào là tensor MFCC có kích thước (66,39,1) tương ứng với (chiều cao, chiều rộng, kênh), mô hình nhận dạng từ kích hoạt của tôi được đề xuất như sau:

```
def create_cnn_model(input_shape):
    model = models.Sequential()

    model.add(layers.Conv2D(16, (3, 3), activation='relu', input_shape=input_shape))
    model.add(layers.MaxPooling2D((2, 2)))

    model.add(layers.Conv2D(64, (3, 3), activation='relu'))
    model.add(layers.MaxPooling2D((2, 2)))

    model.add(layers.Flatten())
    model.add(layers.Dense(16, activation='relu'))
    model.add(layers.Dense(2, activation='sigmoid'))


    return model
```

Hình 4.3 Mô hình nhận dạng từ kích hoạt

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 64, 37, 16)        160

 max_pooling2d (MaxPooling2  (None, 32, 18, 16)        0
 D)

 conv2d_1 (Conv2D)           (None, 30, 16, 64)        9280

 max_pooling2d_1 (MaxPoolin  (None, 15, 8, 64)         0
 g2D)

 flatten (Flatten)           (None, 7680)              0

 dense (Dense)               (None, 16)                122896

 dense_1 (Dense)             (None, 2)                 34

=================================================================
Total params: 132370 (517.07 KB)
Trainable params: 132370 (517.07 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Hình 4.4 Thông số cụ thể của từng lớp trong mô hình nhận dạng từ kích hoạt

Khi không đề cập cụ thể, mặc định bước nhảy trong lớp MaxPooling2D thường được thiết lập với giá trị mặc định là bằng kích thước của cửa sổ (pool size), nghĩa là bước nhảy sẽ bằng kích thước của cửa sổ sau mỗi lần gộp. Còn đệm mặc định sẽ là không. Sau đây tôi sẽ giải thích cụ thể vai trò cũng như cách hoạt động của từng lớp trong Bảng 4.1:

Bảng 4.1 Bảng thông số cụ thể của từng lớp trong mô hình nhận dạng từ kích hoạt

Lớp	Vai trò	Cách hoạt động
Conv2D	Thực hiện việc chập trên đầu vào bằng 16 bộ lọc 3x3; tạo đầu ra có kích thước (64x37x16).	Mỗi bộ lọc chập qua đầu vào, thực hiện tích chập và áp dụng hàm kích hoạt ReLU để tạo ra các feature map.
MaxPooling2D	Thực hiện phép gộp trên từng feature map để giảm kích thước; tạo đầu ra có kích thước (32x18x16).	Lấy giá trị lớn nhất từ mỗi cửa sổ 2x2 trượt qua trên feature map từ lớp trước để giảm kích thước của chúng.
Conv2D_1	Tiếp tục trích xuất các đặc trưng phức tạp từ dữ liệu. Tạo đầu ra có kích thước (30x16x64).	Mỗi bộ lọc chập qua feature map từ lớp trước để tạo ra các feature map mới với độ sâu tăng lên.
MaxPooling2D_1	Tiếp tục thực hiện phép gộp để giảm kích thước. Tạo đầu ra có kích thước (15x8x64).	Lấy giá trị lớn nhất từ mỗi cửa sổ 2x2 trên feature map từ lớp trước để giảm kích thước của chúng.
Flatten	Chuyển đổi ma trận 2D thành vector 1D. Tạo đầu ra có kích thước 7680.	Làm phẳng ma trận thành một vector bằng cách xếp các hàng của ma trận vào nhau.
Dense	Kết nối đầy đủ với 16 đơn vị và sử dụng hàm kích hoạt ReLU.	Tính tổng có trọng số của các đầu vào và áp dụng hàm kích hoạt ReLU để tạo ra đầu ra.
Dense_1	Kết nối đầy đủ cuối cùng với 2 đơn vị và sử dụng sigmoid.	Tính tổng có trọng số của các đầu vào và áp dụng hàm kích hoạt sigmoid để tính toán xác suất trên hai lớp "từ kích hoạt" và "không phải từ kích hoạt", xác suất này nằm trong khoảng (0,1).

Mô hình trên có tổng tham số là 132370, khá phù hợp với yêu cầu kích thước nhỏ, cần ít tài nguyên để chạy liên tục của mô hình nhận dạng từ kích hoạt.

4.4.2 Mô hình nhận dạng câu lệnh

Đối với mô hình nhận dạng câu lệnh, tôi có nghiên cứu một số bài báo như: "MFCC based hybrid fingerprinting method for audio classification through

LSTM" của K. Banuroopaa, D. Shanmuga Priyaa [10]; "CSVC-Net: Code-Switched Voice Command Classification using Deep CNN-LSTM Network" của Arowa Yasmeen, Fariha Ishrat Rahman, Sabbir Ahmed [11]: Md. Hasanul Kabir; "Voice Conversion Based Augmentation and a Hybrid CNN-LSTM Model for Improving Speaker-Independent Keyword Recognition on Limited Datasets" của Yeshanew Ale Wubet; Kuang-Yow Lian [12]… và tham khảo các mô hình hướng dẫn mẫu của Keras.

Trong đó, mô hình CNN-LSTM có tổng tham số lớn (hơn 5 triệu tham số) nên trong phạm vi đề tài, tôi sẽ thực hiện trên ba mô hình: mô hình LSTM tham khảo hai tác giả K. Banuroopaa, D. Shanmuga Priyaa; mô hình CSVC-Net của các tác giả của Arowa Yasmeen, Fariha Ishrat Rahman, Sabbir Ahmed, Md. Hasanul Kabir và mô hình do tôi đề xuất.

Đầu vào của lớp LSTM là một tensor 3D có kích thước: (số lượng mẫu, bước thời gian, đặc trưng). Trong đó:

- Số lượng mẫu (batch size): Số lượng mẫu dữ liệu trong mỗi lần đưa vào mạng để huấn luyện hoặc dự đoán.
- Bước thời gian (timesteps): Đây là số lượng bước thời gian trong chuỗi dữ liệu. Đối với đề tài này, bước thời gian chính là số khung trích xuất đặc trưng MFCC (100).
- Đặc trưng (features): Đây là số lượng tính năng hoặc biến đặc trưng trong mỗi bước thời gian. Trong đề tài này chính là 39 đặc trưng MFCC.

Với đầu vào là tensor MFCC có kích thước (100,39) ứng với (bước thời gian, đặc trưng), Hình 4.5 mô tả các thành phần của mô hình do tôi đề xuất:

```
def get_sequence_model():
    frame_features_input = Input(data["x_train"].shape[1:])
    x = LSTM(units=256, return_sequences=True)(frame_features_input)
    x = keras.layers.LSTM(units=128)(x)
    x = keras.layers.Dropout(0.2)(x)
    x = keras.layers.Dense(64, activation="relu")(x)
    x = keras.layers.Dropout(0.2)(x)
    output = keras.layers.Dense(data["y_train"].shape[1], activation="softmax")(x)
    rnn_model = keras.Model(frame_features_input, output)
    return rnn_model
model = get_sequence_model()
```

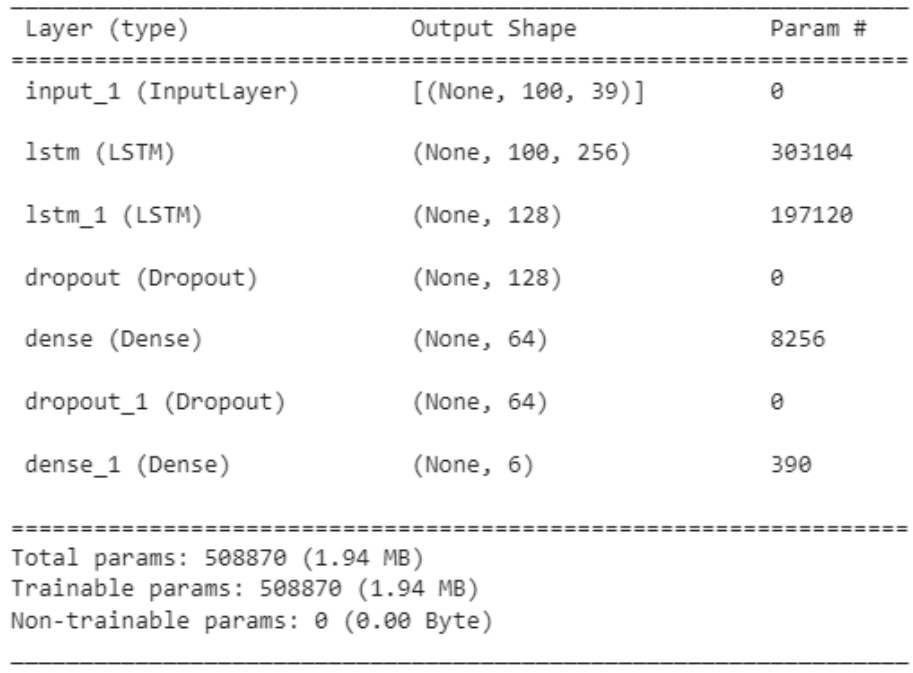
Hình 4.5 Mô hình nhận dạng câu lệnh

Một số khái niệm quan trọng trong mô hình nhận dạng câu lệnh:

- Units: xác định số lượng đơn vị LSTM (cũng gọi là nơ-ron) trong mỗi lớp LSTM. Mỗi đơn vị LSTM là một mạng nơ-ron đơn giản với các kết nối trọng số và các phép toán để kiểm soát thông tin. Số lượng units ảnh hưởng đến sức mạnh và khả năng học của mô hình LSTM. Càng nhiều đơn vị, mô hình có thể học được các mẫu phức tạp hơn, nhưng cũng cần nhiều tài nguyên tính toán hơn.
- Return_sequences: xác định liệu mạng sẽ trả về kết quả tại mỗi bước thời gian hoặc chỉ ở bước thời gian cuối cùng.

+ Khi return_sequences=True, LSTM sẽ trả về kết quả tại mỗi bước thời gian. Điều này hữu ích khi muốn đầu ra từ LSTM cũng là một chuỗi.
Trong lớp LSTM đầu tiên, tôi sẽ lấy kết quả tại mỗi bước thời gian (100 bước) để có thể lưu trữ được nhiều thông tin hơn và truyền chúng tại mỗi bước thời gian tới lớp LSTM tiếp theo để xử lý.
+ Khi return_sequences=False (mặc định), LSTM sẽ chỉ trả về kết quả ở bước thời gian cuối cùng, phù hợp cho các tác vụ như phân loại hoặc dự đoán một giá trị duy nhất từ một chuỗi đầu vào.
Sau khi có đầu vào là chuỗi có kích thước (100,256) từ lớp LSTM đầu tiên, lớp LSTM thứ 2 sẽ chỉ lấy kết quả ở bước thời gian cuối cùng, để giảm kích thước đầu ra và tăng tốc độ huấn luyện, nhưng vẫn phù hợp với mục tiêu là bài toán nhận dạng câu lệnh.

- Dropout: giúp giảm "overfitting" bằng cách tắt một phần các đơn vị nơ-ron ngẫu nhiên trong quá trình huấn luyện.
Trong mô hình, tôi sử dụng hai lớp Dropout có hệ số dropout là 0.2, tức là mỗi lớp sẽ tắt 20% các đơn vị nơ-ron ngẫu nhiên khi đang huấn luyện.
- Lớp Dense: được sử dụng để biểu diễn thông tin từ lớp LSTM vào dạng phù hợp để dự đoán đầu ra.
+ Hàm kích hoạt "relu" được sử dụng trong lớp Dense đầu tiên để học các biểu diễn phi tuyến tính.
+ Hàm kích hoạt "softmax" được sử dụng trong lớp Dense cuối cùng để tính xác suất của các lớp đầu ra, phù hợp với một bài toán nhận dạng.



Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 100, 39)]	0
lstm (LSTM)	(None, 100, 256)	303104
lstm_1 (LSTM)	(None, 128)	197120
dropout (Dropout)	(None, 128)	0
dense (Dense)	(None, 64)	8256
dropout_1 (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 6)	390

Total params: 508870 (1.94 MB)
Trainable params: 508870 (1.94 MB)
Non-trainable params: 0 (0.00 Byte)

Hình 4.6 Thông số cụ thể của từng lớp trong mô hình nhận dạng câu lệnh

Trong mô hình này, đầu ra sẽ là xác suất của 6 nhãn: 5 câu lệnh và 1 nhãn nhiễu môi trường với tổng số tham số là: 508870.

4.5 Kết luận chương

Như vậy, trong chương này, tôi đã xây dựng và tăng cường được bộ cơ sở dữ liệu âm thanh; đưa ra được 2 mô hình đề xuất để nhận dạng từ kích hoạt và câu lệnh phù hợp với yêu cầu đã đề xuất ở Bảng 3.1. Chương tiếp theo tôi sẽ đi vào thử nghiệm và đánh giá các mô hình trên và so sánh với mô hình ở các bài báo đã tìm hiểu.

CHƯƠNG 5. THỬ NGHIỆM VÀ ĐÁNH GIÁ KẾT QUẢ

Với đề tài này, tôi sử dụng 2 dịch vụ máy tính chủ yếu là Google Colab và Jupyter Notebook với khung phát triển là TensorFlow. Google Colab để thực hiện huấn luyện và kiểm tra mô hình, Jupyter Notebook để chạy chương trình nhận dạng tiếng nói. Trong đó, các thông số, tập dữ liệu áp dụng giống nhau với tất cả các quá trình để so sánh kết quả thử nghiệm khách quan nhất.

5.1 Phân chia cơ sở dữ liệu

Với dữ liệu gốc khoảng 300 mẫu/một câu lệnh, sau bước tăng cường dữ liệu âm thanh, tôi tạo ra tập dữ liệu mới có số lượng mẫu gấp 12 lần tập dữ liệu ban đầu: khoảng 3600 mẫu/một câu lệnh. Trong mô hình, tôi có thêm một nhãn nhiễu để tiến hành phân biệt giữa câu lệnh và môi trường, nhãn này có số lượng mẫu gấp khoảng 2.5 lần số lượng mẫu của một câu lệnh : khoảng 9000 mẫu.

Bộ dữ liệu này sẽ được chia ra là 3 tập dữ liệu: Huấn luyện / Đánh giá / Kiểm tra (Train/Validation/Test) với tỷ lệ tương ứng: 72/18/10. Vai trò của các tập dữ liệu như sau:

- Tập dữ liệu Huấn luyện (Train):là tập dữ liệu được sử dụng để huấn luyện mô hình. Tập này bao gồm các mẫu dữ liệu cùng với nhãn tương ứng. Mô hình sẽ học từ dữ liệu này để hiểu mối quan hệ giữa đầu vào và đầu ra và điều chỉnh các tham số để tối ưu hóa hiệu suất.
- Tập dữ liệu Đánh giá (Validation): là tập dữ liệu được sử dụng trong quá trình huấn luyện để đánh giá hiệu suất của mô hình trong quá trình tinh chỉnh siêu tham số và chọn mô hình tốt nhất. Nó giúp đánh giá sự tổng quát hóa của mô hình mà không làm ô nhiễm dữ liệu kiểm tra, và từ đó giúp tránh việc mô hình bị quá mức (overfitting) hoặc thiếu mức (underfitting) trên dữ liệu mới.
- Tập dữ liệu Kiểm tra (Test): là tập dữ liệu độc lập được sử dụng để đánh giá hiệu suất của mô hình sau khi đã được huấn luyện. Mô hình không được phép "nhìn thấy" dữ liệu trong tập này trong quá trình huấn luyện, và việc đánh giá sẽ cho biết khả năng tổng quát hóa của mô hình trên dữ liệu mới.

Dựa trên các vai trò trên, tôi đã xây dựng tập kiểm tra bao gồm các câu lệnh được thu từ giọng nói của 6 người, không có trong tập huấn luyện và đánh giá.

	Unnamed: 0	filepath	label
24723	24723	/content/gdrive/MyDrive/DATN/Augument/150s_28_...	Train_True_AIS_150_29
24724	24724	/content/gdrive/MyDrive/DATN/Augument/150s_28_...	Train_True_AIS_150_29
24725	24725	/content/gdrive/MyDrive/DATN/Augument/150s_28_...	Train_True_AIS_150_29
24726	24726	/content/gdrive/MyDrive/DATN/Augument/150s_28_...	Train_True_AIS_150_29
24727	24727	/content/gdrive/MyDrive/DATN/Augument/150s_28_...	Train_True_AIS_150_29

Hình 5.1 Tổng số lượng mẫu của tập dữ liệu huấn luyện và đánh giá của mô hình nhận dạng câu lệnh

	feature	class
2813	[[-983.1458129882812, 38.142677307128906, -11....	Test_True_AIS_150_29
2814	[[-818.052001953125, 81.73710632324219, -19.44...	Test_True_AIS_150_29
2815	[[-581.4705810546875, 91.21708679199219, -77.1...	Test_True_AIS_150_29
2816	[[-827.3572998046875, 30.106035232543945, -7.2...	Test_True_AIS_150_29
2817	[[-525.3870849609375, -19.670202255249023, -15...	Test_True_AIS_150_29

Hình 5.2 Tổng số lượng mẫu của tập dữ liệu kiểm tra của mô hình nhận dạng câu lệnh

5.2 Một số thông số huấn luyện

Để tăng chất lượng mô hình, ta cần lựa chọn các thông số huấn luyện phù hợp, sau đây tôi sẽ giải thích một số thông số sử dụng trong đề tài:

- loss: Đây là hàm mất mát được sử dụng trong quá trình huấn luyện mô hình. Đối với mô hình nhận dạng từ kích hoạt, tôi sử dụng hàm mất mát nhị phân 'binary_crossentropy' cho mô hình của mình, mỗi mẫu dữ liệu chỉ thuộc vào một trong hai lớp. Với y là nhãn thực tế của mẫu dữ liệu (1 hoặc 0), p là xác suất dự đoán mà mô hình dự đoán cho lớp, công thức tính hàm mất mát nhị phân:

$$\text{loss}(y, p) = -(y \cdot \log(p) + (1 - y) \cdot \log(1 - p)) \qquad PT\ (5.1)$$

  Còn với mô hình nhận dạng câu lệnh, tôi sử dụng hàm mất mát 'categorical_crossentropy' – một hàm mất mát thích hợp cho bài toán phân loại nhiều lớp. y là véc-tơ one-hot encoding của nhãn thực tế (được biểu diễn dưới dạng một véc-tơ có độ dài bằng số nhãn, với một phần tử là 1 và các phần tử còn lại là 0), và p là véc-tơ xác suất dự đoán mà mô hình dự đoán cho mỗi nhãn, hàm categorical_crossentropy được tính như sau:

$$\text{loss}(y, p) = -\sum_{i=1}^{N} y_i \log(p_i) \qquad PT\ (5.2)$$

  Trong đó: y_i là phần tử thứ i của vector nhãn thực tế y; p_i là phần tử thứ i của vector xác suất dự đoán p.
- metrics: Đây là các tiêu chí được sử dụng để đánh giá hiệu suất của mô hình trong quá trình huấn luyện. Trong trường hợp này, chỉ có tiêu chí là accuracy, tức là tỷ lệ các dự đoán chính xác trên tập dữ liệu.
- optimizer: Đây là thuật toán tối ưu hóa được sử dụng để cập nhật trọng số của mạng nơ-ron trong quá trình huấn luyện. Trong trường hợp này, tôi sử dụng 'adam' – một trong những thuật toán tối ưu hóa phổ biến và hiệu quả, kết hợp giữa kỹ thuật học tốc độ biến đổi (momentum) và kỹ thuật học tốc độ học tự điều chỉnh (adaptive learning rate).
- callbacks: Đây là danh sách các hàm gọi lại được sử dụng trong quá trình huấn luyện. Trong trường hợp này, chỉ có callback checkpointer.
- epochs: chu kỳ huấn luyện - số lần lặp lại toàn bộ dữ liệu huấn luyện trong quá trình huấn luyện.

- checkpointer: Đây là một "callback" được sử dụng để lưu trữ mô hình với hiệu suất tốt nhất trên tập dữ liệu đánh giá. Mô hình sẽ được lưu lại dưới dạng một tệp h5 khi hiệu suất trên tập đánh giá cải thiện sau mỗi epoch.
- batch_size: số lượng mẫu dữ liệu được sử dụng để cập nhật trọng số của mạng trong mỗi lần thực hiện thuật toán lan truyền ngược. Đối với mỗi lần cập nhật trọng số, một batch_size dữ liệu được sử dụng.

```
model.compile( loss='binary_crossentropy', metrics=[ 'accuracy'],optimizer='adam')
checkpointer = ModelCheckpoint(filepath='/content/gdrive/MyDrive/DATN/1_Wakeup_Model/WuW_CNN_Sigmoid.h5',
                               monitor='val_accuracy', verbose=1, save_best_only=True)
callbacks = [checkpointer]
batch_size = 10
epochs = 50
# train the model using the training set and validating using validation set
history = model.fit(data["x_train"], data["y_train"], validation_data=(data["x_val"], data["y_val"]),
          batch_size=batch_size, epochs=epochs, callbacks=callbacks)
```

Hình 5.3 Thông số huấn luyện mô hình nhận dạng từ kích hoạt

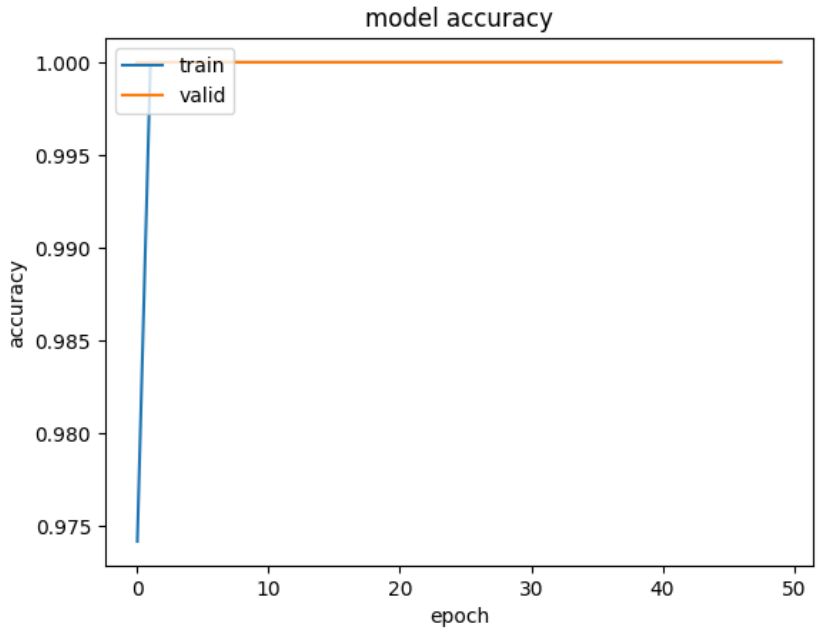
```
model.compile( loss='categorical_crossentropy', metrics=[ 'accuracy'],optimizer='adam')
checkpointer = ModelCheckpoint(filepath='/content/gdrive/MyDrive/DATN/Augument/Augu_500_Noise_Au.h5',
                               monitor='val_accuracy', verbose=1, save_best_only=True)
callbacks = [checkpointer]
batch_size = 20
epochs = 100

history = model.fit(data["x_train"], data["y_train"], validation_data=(data["x_val"], data["y_val"]),
          batch_size=batch_size, epochs=epochs, callbacks=callbacks)
```

Hình 5.4 Thông số huấn luyện mô hình nhận dạng câu lệnh

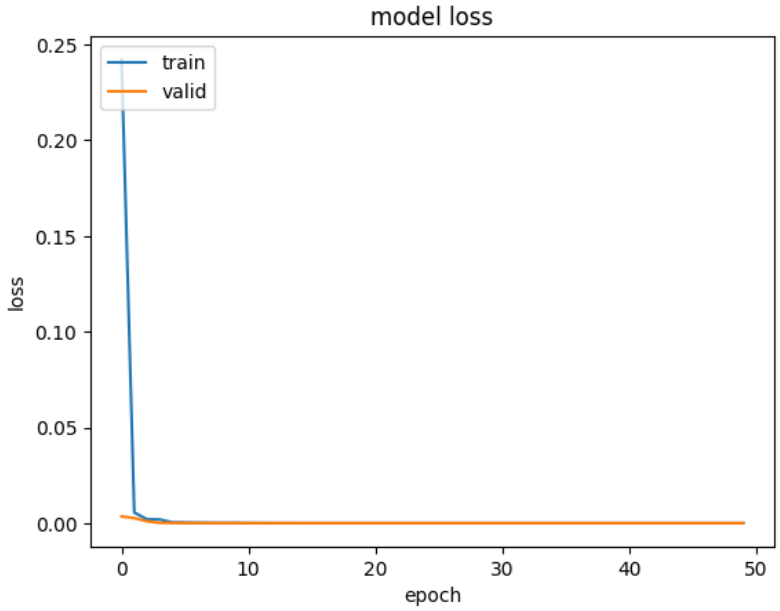
5.3 Kết quả mô hình nhận dạng từ kích hoạt

5.3.1 Không có tăng cường dữ liệu

Theo từng chu kỳ huấn luyện, độ chính xác (accuracy), hàm mất mát (loss) của mô hình trong quá trình huấn luyện trên tập dữ liệu huấn luyện và đánh giá không được tăng cường, thể hiện trong Hình 5.5 và Hình 5.6:



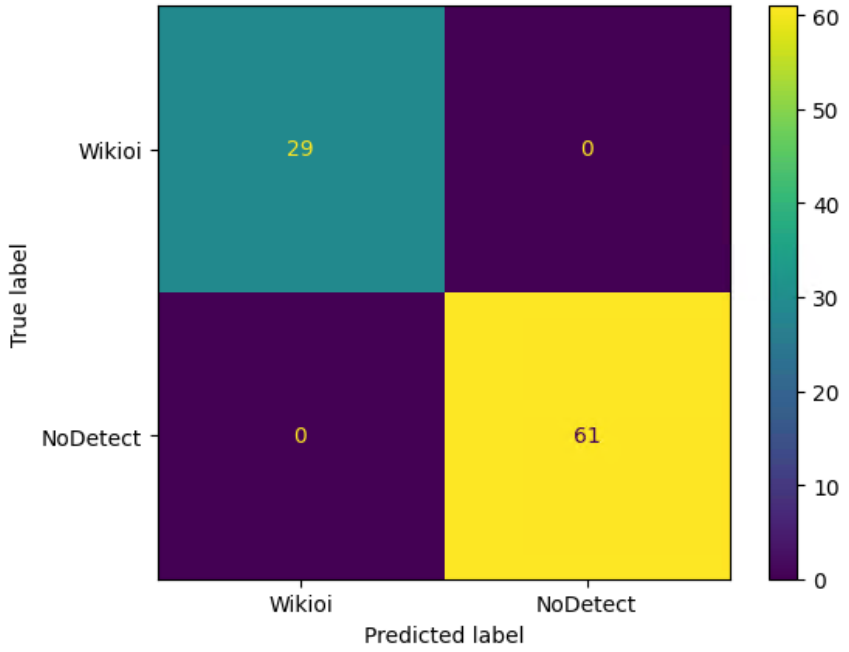
Hình 5.5 Độ chính xác của mô hình nhận dạng từ kích hoạt khi không có tăng cường dữ liệu



Hình 5.6 Hàm mất mát của mô hình nhận dạng từ kích hoạt khi không có tăng cường dữ liệu

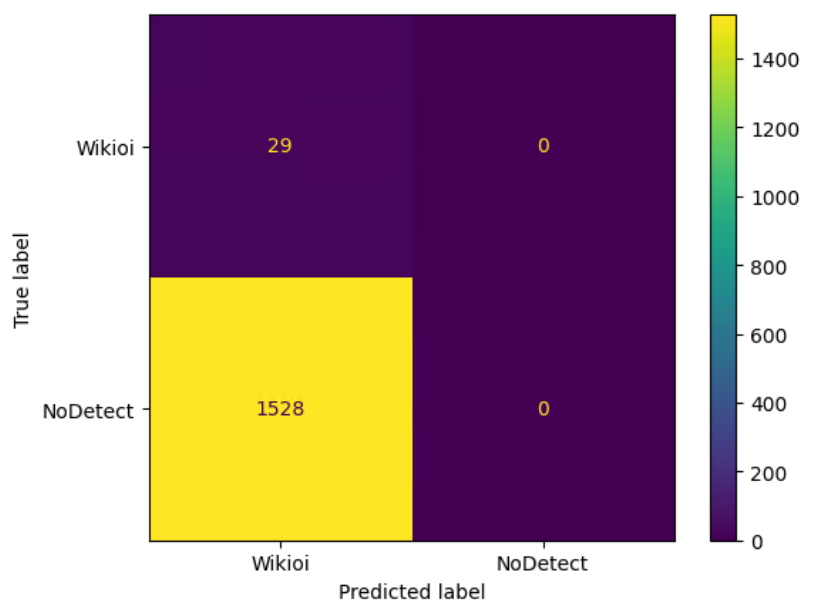
Kết quả đánh giá với mô hình có hiệu suất tốt nhất trên 214 mẫu của tập đánh giá: hàm mất mát: 0.0035; độ chính xác: 100.00%.

Kết quả đánh giá với mô hình có hiệu suất tốt nhất trên 90 mẫu của tập kiểm tra ít nhiễu (khoảng lặng, ít tiếng ồn): hàm mất mát: 0.0057; độ chính xác: 100.00%.



Hình 5.7 Kết quả dự đoán trên tập kiểm tra ít nhiễu của mô hình kích hoạt khi không có tăng cường dữ liệu

Kết quả đánh giá với mô hình có hiệu suất tốt nhất trên 1557 mẫu của tập kiểm tra nhiều nhiễu (tiếng nói, tiếng ồn): hàm mất mát: 6.9017; độ chính xác: 1.86%.

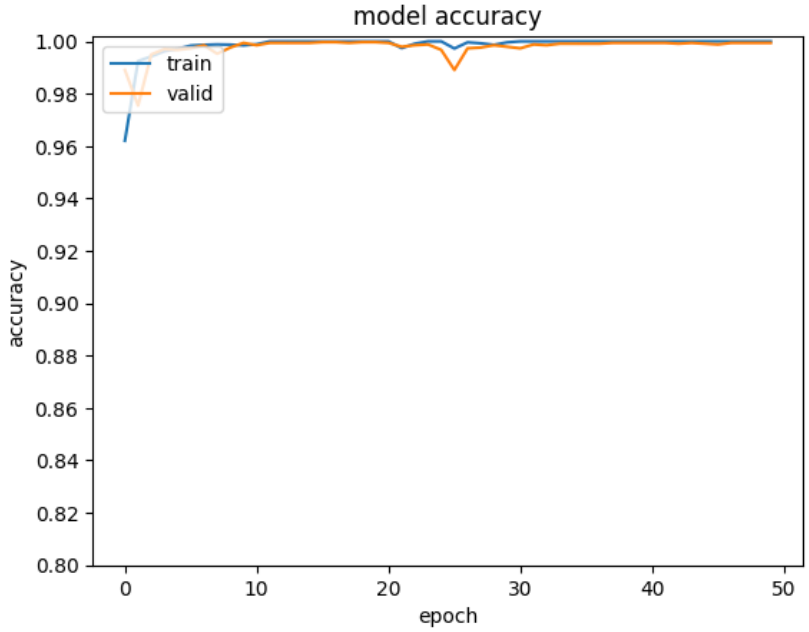


Hình 5.8 Kết quả dự đoán trên tập kiểm tra nhiều nhiễu của mô hình kích hoạt khi không có tăng cường dữ liệu

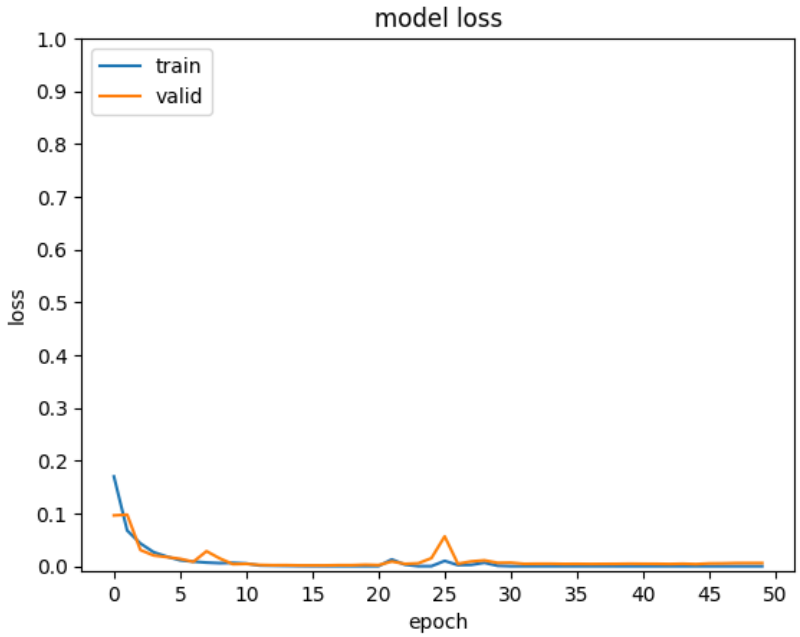
Qua kết quả ở Hình 5.8, ta thấy được với tập dữ liệu không tăng cường, mô hình nhận dạng không tốt với tập dữ liệu nhiều nhiễu (tiếng nói, tiếng ồn). Vấn đề này sẽ được khắc phục với tập dữ liệu được tăng cường.

5.3.2 Có tăng cường dữ liệu

Theo từng chu kỳ huấn luyện, độ chính xác (accuracy), hàm mất mát (loss) của mô hình trong quá trình huấn luyện trên tập dữ liệu huấn luyện và đánh giá được tăng cường, thể hiện trong Hình 5.9 và Hình 5.10:



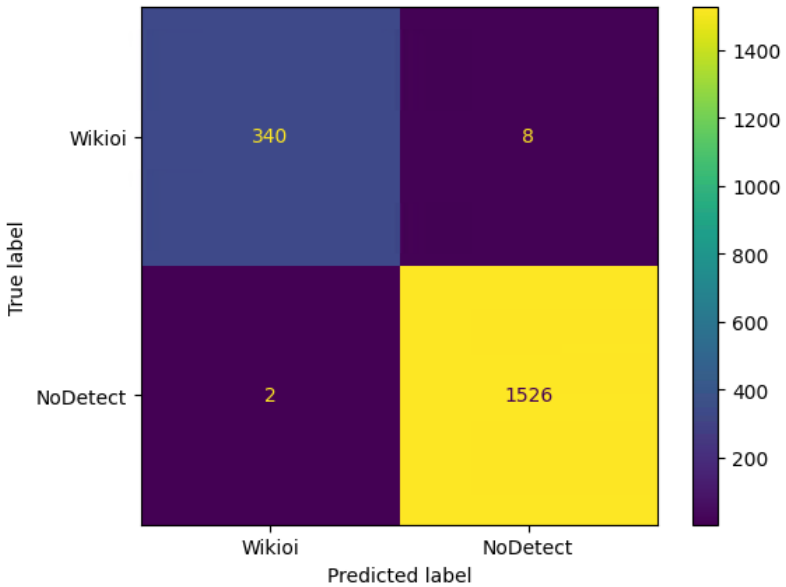
Hình 5.9 Độ chính xác của mô hình nhận dạng từ kích hoạt khi có tăng cường dữ liệu



Hình 5.10 Hàm mất mát của mô hình nhận dạng từ kích hoạt khi có tăng cường dữ liệu

Kết quả đánh giá với mô hình có hiệu suất tốt nhất trên 3387 mẫu của tập đánh giá: hàm mất mát: 0.0014; độ chính xác: 99.97%

Kết quả đánh giá với mô hình có hiệu suất tốt nhất trên 1876 mẫu của tập kiểm tra nhiều nhiễu (tiếng nói, tiếng ồn): hàm mất mát: 0.0257; độ chính xác: 99.47%.



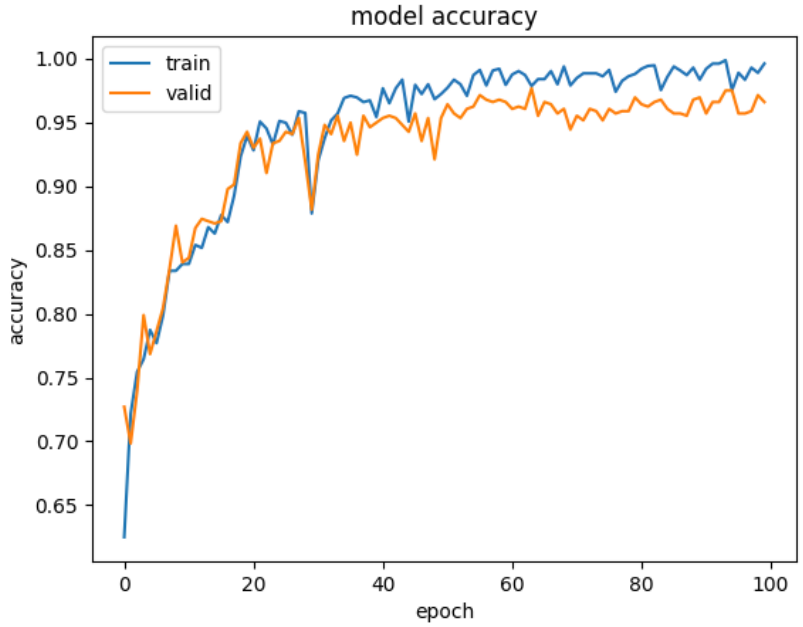
Hình 5.11 Kết quả dự đoán trên tập kiểm tra nhiều nhiễu của mô hình kích hoạt khi có tăng cường dữ liệu

Như ta thấy, cùng với một tập kiểm tra nhiễu, đối với tập dữ liệu được tăng cường, mô hình nhận dạng từ kích hoạt cho chất lượng tốt hơn, có thể hoạt động được trong môi trường nhiều nhiễu âm thanh như giọng nói, tiếng ồn – môi trường thực của ngôi nhà.

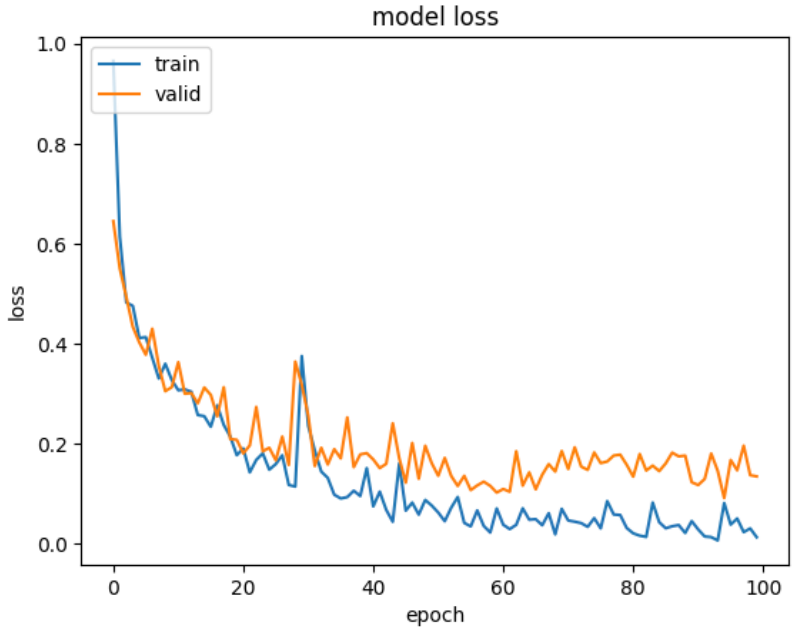
5.4 Kết quả mô hình nhận dạng câu lệnh

5.4.1 Không có tăng cường dữ liệu

Theo từng chu kỳ huấn luyện, độ chính xác (accuracy), hàm mất mát (loss) của mô hình trong quá trình huấn luyện trên tập dữ liệu huấn luyện và đánh giá không được tăng cường, thể hiện trong Hình 5.12 và Hình 5.13:



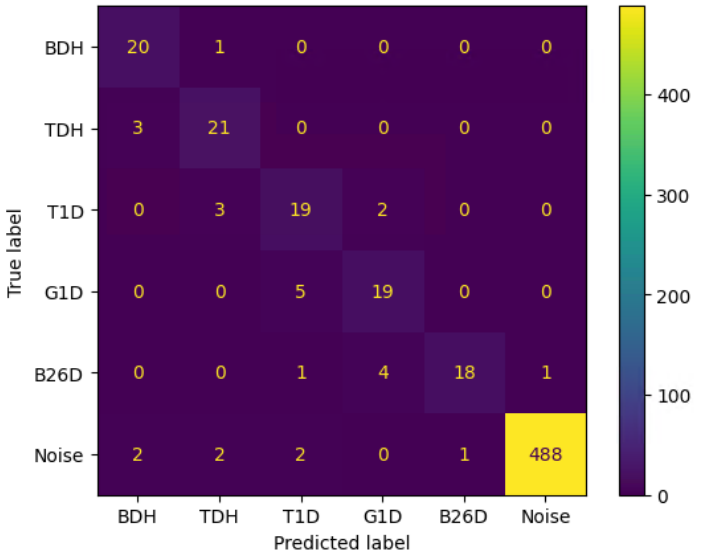
Hình 5.12 Độ chính xác của mô hình nhận dạng câu lệnh khi không có tăng cường dữ liệu



Hình 5.13 Hàm mất mát của mô hình nhận dạng câu lệnh khi không có tăng cường dữ liệu

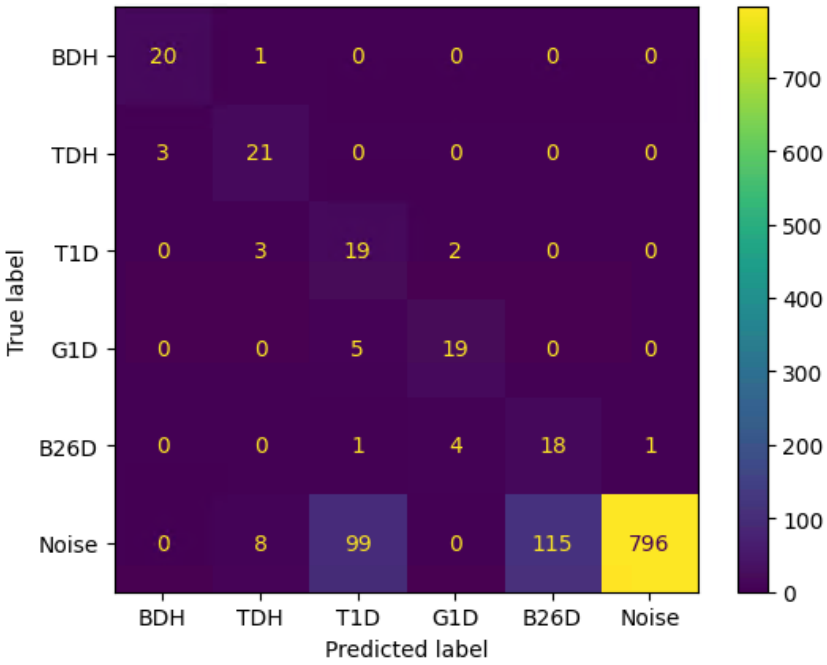
Kết quả đánh giá với mô hình có hiệu suất tốt nhất trên 557 mẫu của tập đánh giá: hàm mất mát: 0.1164; độ chính xác: 97.67%

Kết quả đánh giá với mô hình có hiệu suất tốt nhất trên 612 mẫu của tập kiểm tra ít nhiễu (khoảng lặng, ít tiếng ồn): hàm mất mát: 0.2750; độ chính xác: 95.59%.



Hình 5.14 Kết quả dự đoán trên tập kiểm tra ít nhiễu của mô hình nhận dạng câu lệnh khi không có tăng cường dữ liệu

Kết quả đánh giá với mô hình có hiệu suất tốt nhất trên 1135 mẫu của tập kiểm tra nhiều nhiễu (tiếng nói, tiếng ồn): hàm mất mát: 1.1897; độ chính xác: 78.68%.

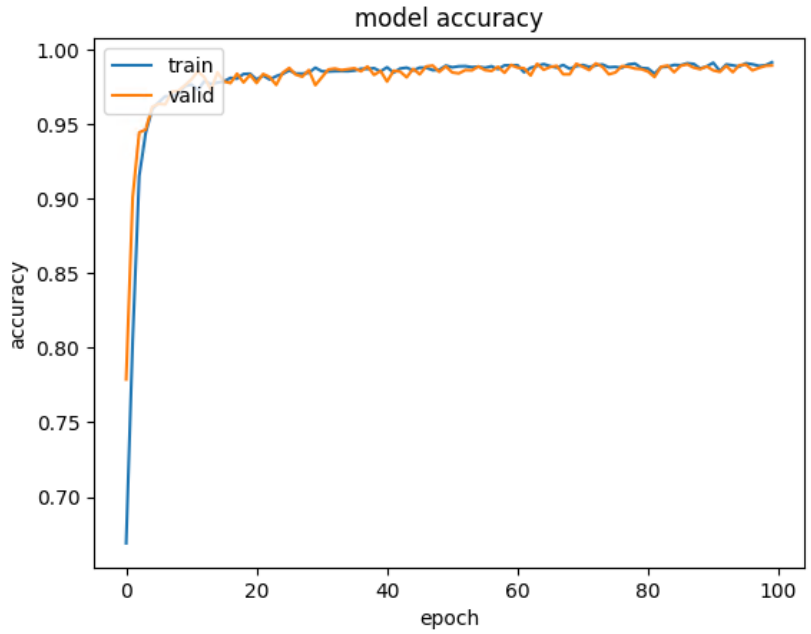


Hình 5.15 Kết quả dự đoán trên tập kiểm tra nhiều nhiễu của mô hình nhận dạng câu lệnh khi không có tăng cường dữ liệu

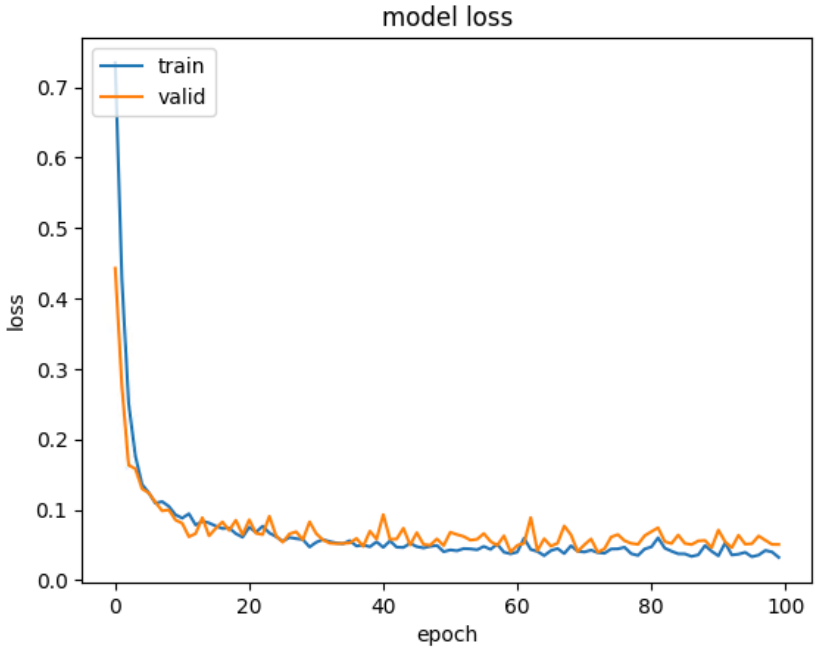
Qua kết quả trên, ta thấy được với tập dữ liệu không tăng cường, mô hình nhận dạng không tốt với tập dữ liệu nhiều nhiễu (tiếng nói, tiếng ồn). Vấn đề này sẽ được khắc phục với tập dữ liệu được tăng cường.

5.4.2 Có tăng cường dữ liệu

Theo từng chu kỳ huấn luyện, độ chính xác (accuracy), hàm mất mát (loss) của mô hình trong quá trình huấn luyện trên tập dữ liệu huấn luyện và đánh giá được tăng cường, thể hiện trong Hình 5.16 và Hình 5.17:



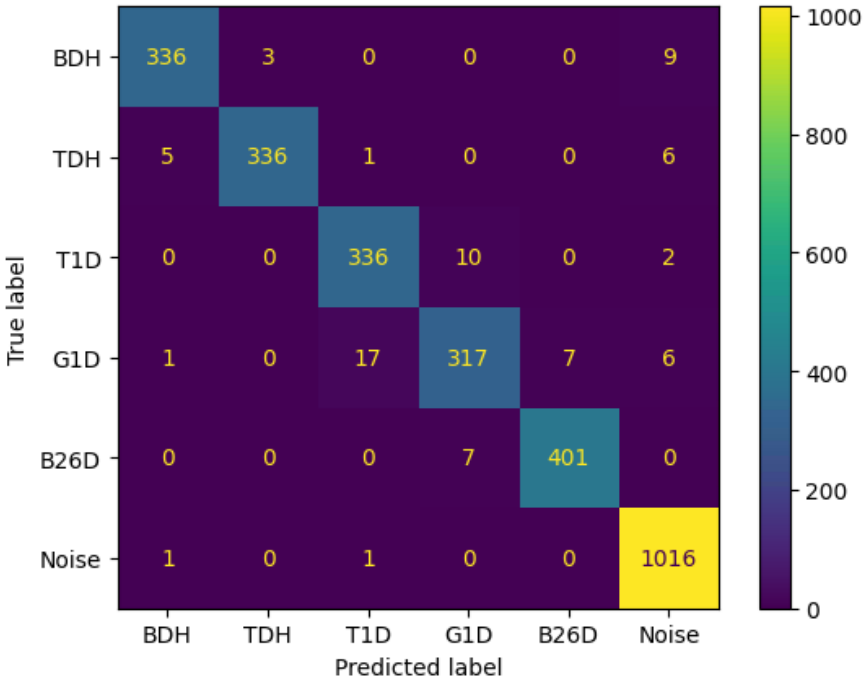
Hình 5.16 Độ chính xác của mô hình nhận dạng câu lệnh khi có tăng cường dữ liệu



Hình 5.17 Hàm mất mát của mô hình nhận dạng câu lệnh khi có tăng cường dữ liệu

Kết quả đánh giá với mô hình có hiệu suất tốt nhất trên 4946 mẫu của tập đánh giá: hàm mất mát: 0.0398; độ chính xác: 99.05%.

Kết quả đánh giá với mô hình có hiệu suất tốt nhất trên 2818 mẫu của tập kiểm tra nhiều nhiễu (tiếng nói, tiếng ồn): hàm mất mát: 0.1608; độ chính xác: 97.30%.



Hình 5.18 Kết quả dự đoán trên tập kiểm tra nhiều nhiễu của nhận dạng câu lệnh khi có tăng cường dữ liệu

Như ta thấy, với một tập kiểm tra nhiễu lớn và phức tạp hơn, đối với tập dữ liệu được tăng cường, mô hình nhận dạng câu lệnh cho chất lượng tốt hơn, có thể hoạt động được trong môi trường nhiều nhiễu âm thanh như giọng nói, tiếng ồn – môi trường thực của ngôi nhà.

5.4.3 So sánh kết quả mô hình đề xuất với các mô hình

Bảng 5.1 Bảng so sánh các mô hình trên tập dữ liệu không tăng cường

Mô hình	Tổng tham số	Tập đánh giá		Tập kiểm tra	
		Độ chính xác (%)	Hàm mất mát	Độ chính xác (%)	Hàm mất mát
LSTM	689414	98.92	0.0628	71.63	2.1477
CSVC-Net	1157006	100.00	0.0046	82.47	0.7796
Mô hình đề xuất	508870	97.67	0.1164	78.68	1.1897

Bảng 5.2 Bảng so sánh các mô hình trên tập dữ liệu tăng cường

Mô hình	Tổng tham số	Tập đánh giá		Tập kiểm tra	
		Độ chính xác (%)	Hàm mất mát	Độ chính xác (%)	Hàm mất mát
LSTM	689414	98.79	0.0646	94.85	0.2574
CSVC-Net	1157006	99.11	0.0517	96.10	0.2591
Mô hình đề xuất	508870	99.05	0.0398	97.30	0.1608

Bảng 5.1 và Bảng 5.2 so sánh các mô hình với cùng tập huấn luyện, tập đánh giá, tập kiểm tra nhiều nhiễu và các thông số của quá trình huấn luyện. Qua quá trình huấn luyện và kiểm tra, từ kết quả thu được, ta có thể thấy:

- Tổng tham số của các mô hình: mô hình đề xuất có tổng tham số ít nhất (508870).
- Kết quả trên tập kiểm tra:
  + Tập dữ liệu không tăng cường, tập kiểm tra ít nhiễu: mô hình CSVC-Net cho kết quả tốt nhất (độ chính xác: 82.47%, hàm mất mát: 0.7796); tiếp theo là mô hình đề xuất (độ chính xác: 78.68%, hàm mất mát: 1.1897).
  + Tập dữ liệu tăng cường, tập kiểm tra nhiều nhiễu: mô hình đề xuất cho kết quả tốt nhất (độ chính xác: 97.30%, hàm mất mát: 0.1608) tiếp theo là mô hình CSVC-Net (độ chính xác: 96.10%, hàm mất mát: 0.2591).

Từ những thông số trên, ta có thể thấy được mô hình CSVC-Net và mô hình đề xuất cho kết quả tốt hơn mô hình LSTM. Với tập dữ liệu tăng cường, tập kiểm tra nhiều nhiễu, mô hình đề xuất cho kết quả tốt nhất.

5.5 Thử nghiệm hệ thống

5.5.1 Mục tiêu thử nghiệm

Sau khi huấn luyện và thu được các kết quả như trên, tôi sẽ tiến hành thử nghiệm hệ thống để kiểm tra độ chính xác của mô hình trong điều kiện thực tế của căn phòng.

5.5.2 Triển khai thử nghiệm

Với 2 mô hình đề xuất, tôi sẽ xây dựng chương trình thử nghiệm như sau:

- Mô hình nhận dạng từ kích hoạt: chạy liên tục, ghi âm và cắt tệp âm thanh dài 1s, đưa vào mô hình, nếu nhận dạng là từ kích hoạt thì sẽ chạy mô hình nhận dạng câu lệnh, tạm dừng cắt tệp âm thanh còn không thì sẽ tiếp tục nhận dạng.
- Mô hình nhận dạng câu lệnh: sau khi được kích hoạt, sẽ cắt tệp âm thanh dài 1.5s, đưa vào mô hình, nếu nhận dạng là câu lệnh thì sẽ phản hồi còn không thì sẽ tiếp tục nhận dạng. Nếu quá 15s (10 lần cắt) mà không nhận dạng được câu lệnh nào (toàn nhiễu), thì sẽ quay lại chạy mô hình nhận dạng từ kích hoạt.

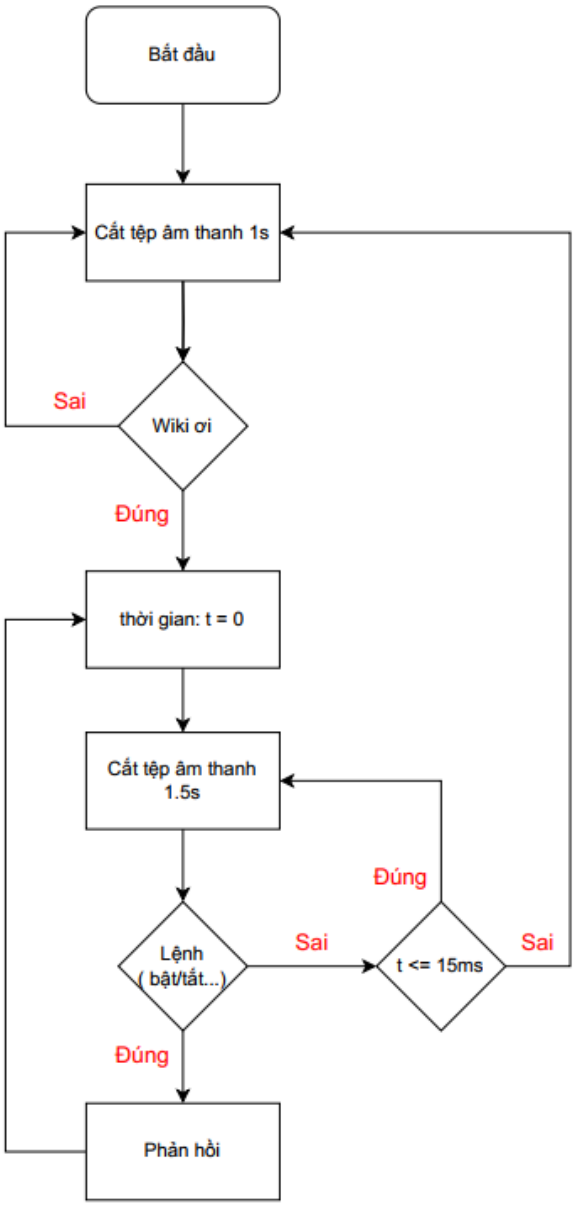
Lưu đồ thuật toán ở Hình 5.19 minh hoạ quá trình hoạt động của hệ thống.

5.5.3 Kết quả thử nghiệm

Sau quá trình thử nghiệm với các giọng nam nữ khác nhau, tôi rút ra được một số kết luận như sau:

- Mô hình nhận dạng từ kích hoạt hoạt động khá tốt, khá nhạy trong việc nhận dạng ra được từ kích hoạt trong môi trường nhiễu lớn, trong một số ít trường hợp vẫn nhận dạng sai hoặc không nhận dạng được.
- Mô hình nhận dạng câu lệnh hoạt động tốt, nhạy với nhận dạng ra câu lệnh, nhưng đối với những từ có phát âm gần giống nhau thì vẫn có lúc nhận dạng sai: như “bật nốt độ” với “bật 1 độ”.

Cả hai mô hình đều hoạt động khá tốt, đáp ứng được cơ bản các yêu cầu đề ra ở mục 5.5.1, nhưng vẫn có trường hợp không nhận dạng được hoặc nhận dạng sai đối với các từ có phát âm gần giống nhau, đây là một trong những yếu tố cần cải thiện của mô hình. Tôi đã ghi lại quá trình thử nghiệm đường dẫn này [13].



Hình 5.19 Lưu đồ thử nghiệm

5.6 Kết luận chương

Như vậy, trong chương này, ta đã làm rõ được cách phân chia cơ sở dữ liệu cũng như hiểu được một số thông số huấn luyện của mô hình. Với quá trình huấn luyện, đánh giá và kiểm tra, trên tập dữ liệu tăng cường, tập kiểm tra nhiều nhiễu, mô hình đề xuất cho kết quả tốt nhất, tổng tham số chỉ bằng một nửa mô hình CSVC-Net, qua đó thể hiện sự phù hợp với môi trường thực tế, có tính ứng dụng cao hơn các mô hình còn lại. Đối với quá trình thử nghiệm thực tế, mô hình đề xuất cũng cho kết quả khá tốt, đáp ứng được với điều kiện môi trường của căn phòng trong gia đình.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Mô hình đề xuất đã đáp ứng được cơ bản mục tiêu đã nêu ở mục 1.3: nghiên cứu ứng dụng mạng nơ-ron LSTM trong nhận dạng câu lệnh điều khiển điều hoà bằng tiếng nói. Quá trình thử nghiệm cho thấy mô hình có thể hoạt động được trong môi trường thực tế của căn phòng. Tuy nhiên, với môi trường nhiễu và giọng nói đa dạng, vẫn có trường hợp mô hình nhận dạng sai hoặc chưa nhận dạng được. Đây là một trong những yếu tố cần cải thiện của mô hình.

Trong quá trình thực hiện đề tài, tôi rút ra được một số vấn đề còn tồn tại như sau:

- Thiết bị thu âm thanh chưa có chất lượng cao, nên tập cơ sở dữ liệu chưa đồng bộ.
- Tạo cơ sở dữ liệu:
  + Từ kích hoạt và câu lệnh: tự thu, số lượng mẫu còn ít.
  + Tập nhiễu: chưa tạo được tập dữ liệu nhiễu đủ lớn và đa dạng để cải thiện mô hình.
- Quá trình huấn luyện mô hình trên dịch vụ Google Colab bị giới hạn về dung lượng và thời gian.

…

Sau một thời gian nghiên cứu đề tài, tôi đề xuất một số hướng phát triển trong tương lai như sau:

- Đã thu dữ liệu của các câu lệnh để tăng tính ứng dụng của mô hình: tăng/giảm 2 độ, bật 20-32 độ, bật chế độ tự động/sưởi ấm/làm mát/làm khô.
- Ghi âm thêm câu tạm biệt để thoát khỏi mô hình nhận dạng câu lệnh.
- Nghiên cứu ứng dụng thêm các loại mạng nơ-ron để nhận dạng được chính xác hơn trong mô hình.

Trên đây là báo cáo đồ án tốt nghiệp của tôi, vì thời gian và kiến thức còn nhiều hạn chế nên tôi mong được sự góp ý và hướng dẫn thêm của mọi người, hy vọng tôi có thể cộng tác cùng những người có đam mê về lĩnh vực nhận dạng giọng nói để phát triển mô hình dựa trên những gì hiện có, giúp mô hình có tính ứng dụng cao hơn nữa.

Một lần nữa, tôi xin chân thành cảm ơn mọi người.

TÀI LIỆU THAM KHẢO

[1] Vietnambiz, "Vietnambiz," 2023. [Online]. Available: https://vietnambiz.vn/luong-nguoi-dung-smartphone-tai-viet-nam-uoc-dung-thu-hai-dong-nam-a-vao-nam-2026-thuoc-top-nhieu-nhat-the-gioi-202359104330221.htm.

[2] T. T. Nguyễn, "Viblo," [Online]. Available: https://viblo.asia/p/kien-thuc-nen-tang-xu-ly-tieng-noi-speech-processing-jvElaAL6lkw.

[3] H. S. a. S. Renals, *Speech Signal Analysis*, 2015.

[4] Weekly Study Corp, "Weekly Study," [Online]. Available: https://www.weeklystudy.asia/2021/10/gioi-thieu-ve-hoc-sau-deep-learning-su-lien-quan-giua-dl-va-ml.html.

[5] V. Anh, "Aicurious," [Online]. Available: https://aicurious.io/blog/2019-09-23-cac-ham-kich-hoat-activation-function-trong-neural-networks.

[6] C. P. Van, "Viblo," [Online]. Available: https://viblo.asia/p/deep-learning-tim-hieu-ve-mang-tich-chap-cnn-maGK73bOKj2.

[7] Colah, "Colah," [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[8] I. Jordal, "Github," [Online]. Available: https://github.com/iver56/audiomentations.

[9] Librosa, "Librosa," [Online]. Available: https://librosa.org/doc/main/generated/librosa.feature.mfcc.html.

[10] D. S. P. K. Banuroopaa, "MFCC based hybrid fingerprinting method for audio classification through LSTM," *The International Journal of Nonlinear Analysis and Applications (IJNAA)*, 2022.

[11] F. I. R. S. A. M. H. K. Arowa Yasmeen, "CSVC-Net: Code-Switched Voice Command Classification using Deep CNN-LSTM Network," *Conference Paper*, 2021.

[12] Yeshanew, "Voice Conversion Based Augmentation and a Hybrid CNN-LSTM Model for Improving Speaker-Independent Keyword Recognition on Limited Datasets," *IEEE*, 2022.

[13] L. D. Khanh. [Online]. Available: https://www.youtube.com/shorts/FjfQR7-k39Y.