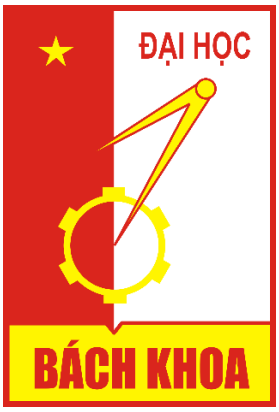
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI



ĐỒ ÁN TỐT NGHIỆP

Nhận diện phát âm từ khoá trên thiết bị

NGUYỄN THIỆN HỶ
hy.nt162111@sis.hust.edu.vn

Ngành Công nghệ thông tin và truyền thông
Chuyên ngành Khoa học máy tính

Giảng viên hướng dẫn: PGS. TS. Đỗ Phan Thuận

Chữ ký của GVHD

Bộ môn: Khoa học máy tính
Viện: Công nghệ thông tin và truyền thông

HÀ NỘI, 6/2021

ĐỀ TÀI TỐT NGHIỆP

1. **Thông tin sinh viên**

   Họ và tên: Nguyễn Thiện Hỷ
   Số điện thoại: 0364319191
   Email: hynt162111@sis.hust.edu.vn
   Lớp: CNTT2.02-K61
   Hệ đào tạo: Đại học chính quy
   Đồ án tốt nghiệp (ĐATN) thực hiện tại: Trường đại học Bách Khoa Hà Nội
   Thời gian thực hiện ĐATN: Từ ngày 1/2/2021 đến ngày 17/6/2021

2. **Nội dung ĐATN:**

   Tìm hiểu, xây dựng và triển khai giải pháp nhận diện phát âm từ khoá trên thiết bị dựa vào học máy, học sâu.

3. **Nhiệm vụ của ĐATN:**

   - Tìm hiểu các kỹ thuật xử lý tín hiệu âm thanh
   - Tìm hiểu phương pháp trích xuất đặc trưng của tín hiệu âm thanh
   - Tìm hiểu các mô hình SVM, TC-RESNET, MULTIHEAD-ATTENTION-RNN
   - Áp dụng các kỹ thuật xử lý, phương pháp trích xuất đặc trưng âm thanh vào các mô hình để huấn luyện và đánh giá
   - Triển khai mô hình lên thiết bị, đánh giá, so sánh kết quả thu được

4. **Lời cam đoan của sinh viên:**

Tôi - Nguyễn Thiện Hỷ - cam kết Đồ án tốt nghiệp (ĐATN) **"Nhận diện phát âm từ khoá trên thiết bị"** là công trình nghiên cứu triển khai của bản thân tôi, với sự giúp đỡ tận tình của PGS.TS. Đỗ Phan Thuận. Các số liệu, kết quả nêu trong ĐATN hoàn toàn trung thực, không sao chép toàn văn của bất kỳ công trình nào.

Hà Nội, ngày 17 tháng 6 năm 2021

Tác giả ĐATN

Nguyễn Thiện Hỷ

5. Xác nhận của giáo viên hướng dẫn về mức độ hoàn thành ĐATN và cho phép bảo vệ

Hà Nội, ngày 17 tháng 6 năm 2021

Giáo viên hướng dẫn

PGS.TS Đỗ Phan Thuận

Lời cảm ơn

Đầu tiên, em xin được gửi lời cảm ơn chân thành tới các thầy, các cô tại trường đại học Bách Khoa Hà Nội, đặc biệt là các thầy cô trong Viện công nghệ thông tin và truyền thông, đã có công rất lớn trong việc dạy bảo, chỉ dẫn tận tình trong suốt quá trình em học tập. Những kiến thức mà thầy cô mang lại là hành trang vững chắc, vô cùng quý báu cho em trên con đường sự nghiệp sau này.

Tiếp đó, em xin gửi lời cảm ơn chân thành đến PGS.TS Đỗ Phan Thuận, người đã đồng hành cùng em trong suốt quá trình em làm đồ án. Thầy là người dẫn đường, mở lối, cung cấp các tài liệu, trang thiết bị cần thiết, cũng như là nguồn động viên tinh thần rất lớn để em có thể hoàn thành đồ án này. Mình cũng xin cảm ơn tất cả những người bạn đã luôn đồng hành, sẵn sàng giúp đỡ, cùng nhau học tập trong suốt những năm tháng học đại học đầy gian truân, vất vả.

Cuối cùng con xin cảm ơn gia đình, cảm ơn ông bà, cảm ơn bố, cảm ơn mẹ, cảm ơn em gái đã luôn động viên về tinh thần và vật chất để con, anh có được thành quả như ngày hôm nay.

Trong quá trình làm đồ án này, dù em đã rất cố gắng, nhưng do hạn chế về thời gian thực hiện và thiếu kinh nghiệm thực tế nên không thể tránh khỏi những thiếu sót, em rất mong nhận được những đóng góp quý báu của thầy cô để đồ án của em ngày càng hoàn thiện hơn.

Em xin chân thành cảm ơn!

Tóm tắt nội dung đồ án

Nhận diện phát âm từ khoá trên các thiết bị tài nguyên thấp là một trong những bài toán quan trọng trong lĩnh vực xử lý tiếng nói. Việc điều khiển các thiết bị thông minh thông qua giọng nói mang nhiều giá trị thực tiễn cả về kinh tế, công nghệ, đặc biệt là trong thời buổi công nghiệp hoá, hiện đại hoá đất nước ngày nay.

Đồ án này giải quyết bài toán nhận diện phát âm từ khoá trên các thiết bị tài nguyên thấp sử dụng dữ liệu ghi âm được thu thập từ các bạn sinh viên trường đại học Bách Khoa Hà Nội. Đồ án sẽ đề xuất một số phương pháp xử lý tín hiệu âm thanh để làm giàu dữ liệu, tìm hiểu một số phương pháp trích xuất đặc trưng của tín hiệu âm thanh, xây dựng các mô hình nhận diện phát âm từ khoá dựa trên một số thuật toán học máy cơ bản như SVM, một số kiến trúc mạng nơ ron nhân tạo như TC-RESNET, MULTIHEAD-ATTENTION-RNN, cuối cùng là sẽ thử nghiệm về độ chính xác cũng như về thời gian dự đoán trên thiết bị JETSON NANO 2GB.

Nội dung đồ án bao gồm những phần sau:

- **Chương 1: Tổng quan đề tài**
  Giới thiệu bài toán, tầm quan trọng và tính thực tiễn của bài toán, giới thiệu một số nghiên cứu liên quan và đưa ra hướng tiếp cận của đồ án.
- **Chương 2: Cơ sở lý thuyết**
  Các lý thuyết cơ bản về tín hiệu âm thanh, biến đổi fourier, mô hình SVM, mạng nơ-ron tích chập, mạng nơ-ron hồi quy, cơ chế attention.
- **Chương 3: Phương pháp giải quyết bài toán**
  Trình bày chi tiết các bước thực hiện trong quá trình giải quyết bài toán.
- **Chương 4: Thực nghiệm và đánh giá**
  Quá trình thực nghiệm, đánh giá và so sánh các kết quả đạt được.
- **Chương 5: Kết luận và hướng phát triển**
  Tổng kết đóng góp của đồ án, khó khăn trong quá trình thực hiện và đưa ra hướng phát triển trong tương lai.

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC THUẬT NGỮ

English	Tiếng Việt
text	text
class	lớp
audio	audio
sampling	lấy mẫu
sample rate	tần số lấy mẫu
aliasing	mất mát thông tin
frame	khung tín hiệu
frame size	kích thước khung
frame overlap	khung chồng lấp
volume	âm lượng
loudless	độ lớn (về âm thanh)
amplitude	biên độ
pitch	cao độ
timbre	âm sắc
hop length	bước khung
frame rate	tỷ lệ khung
vocal folds	thanh quản
fourier transform	biến đổi fourier
magnitude	độ lớn (liên quan đến tần số)
phase	pha
hyperplane	siêu mặt phẳng
margin	lề
kernel	kernel
linear	tuyến tính
gamma	gamma
feature map	đặc trưng ánh xạ
padding	padding
stride	stride
vanishing gradient	mất mát đạo hàm
activation function	hàm kích hoạt
weight	trọng số mô hình
hadamard	hadamard
bias	bias
sigmoid	sigmoid
tanh	tanh
relu	rectified linear unit
softmax	softmax
vector	vector

encoder	encoder
decoder	decoder
attention	attention
loss	hàm mất mát
confusion matrix	ma trận nhầm lẫn
file	tệp (liên quan máy tính)
augment	làm giàu
train	tập huấn luyện
test	tập kiểm tra
input	đầu vào
overfit	quá khớp
webservice	webservice
server	máy chủ

DANH MỤC TỪ VIẾT TẮT

Từ viết tắt	Từ gốc
API	Application Programming Interface
GPU	Graphics processing unit
DFT	Discrete Fourier Transform
MFCC	Mel Frequency Cepstral Coefficients
SVM	Support Vector Machine
TC-RESNET	Temporal Convolution Residual Network
RNN	Recurrent neural network
LSTM	Long short term memory
CNN	Convolution neural network
HTTP	Hypertext Transfer Protocol
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
RBF	Radial basis function
RESNET	Residual Network
ĐATN	Đồ án tốt nghiệp
SV	Sinh viên
CNTT	Công nghệ thông tin

CHƯƠNG 1. TỔNG QUAN

1.1 Giới thiệu chung

Trong thời buổi phát triển của nền công nghệ thông tin hiện nay, việc người dân trên thế giới nói chung cũng như ở Việt Nam nói riêng sở hữu một chiếc điện thoại thông minh không còn gì là quá xa lạ. Cùng với đó là sự phát triển của vô số các thiết bị thông minh nhỏ gọn khác, phải kể đến như: Camera thông minh, máy hút bụi thông minh, loa thông minh, vv... Chưa kể đến việc con người ngày càng có xu hướng làm việc tay chân ít hơn, thay vào đó họ sử dụng các thiết bị thông minh, người máy được lập trình sẵn cho các công việc cụ thể. Các thiết bị, người máy đó có thể được điều khiển bằng tay, bằng nút bấm, hay thậm chí bằng cả giọng nói của con người. Lợi ích của việc này là hiệu suất công việc sẽ cao hơn, người máy hay thiết bị thông minh có thể giải quyết được các vấn đề mà con người không làm được. Để đạt được những lợi ích như vậy thì đòi hỏi phải có sự liên hệ mật thiết đến việc xây dựng các ứng dụng, mô hình, hệ thống trên thiết bị thông minh. Gần đây, với sự phát triển mạnh mẽ về phần cứng cũng như tài nguyên máy tính, việc xây dựng một hệ thống nhận diện khuôn mặt, nhận diện giọng nói trên một máy tính cá nhân cũng không còn quá là khó khăn. Vấn đề đặt ra là làm thế nào để có thể xây dựng được các hệ thống đó trên các thiết bị thông minh với cấu hình tài nguyên giới hạn ?

Chắc hẳn chúng ta đã nghe qua những cái tên rất nổi tiếng như: Google Assistant, Apple’s Siri, Cortana, vv... Chúng là một trong số hàng ngàn ứng dụng nhận diện giọng nói trên nhiều thiết bị thông minh được con người sử dụng hàng ngày. Ngày nay, việc điều khiển một thiết bị thông minh theo mệnh lệnh của con người thông qua lời nói, cử chỉ là một yêu cầu rất cần thiết và quan trọng. Để đáp ứng nhu cầu đó, trong những năm gần đây, các nhà phát triển đã nghiên cứu, xây dựng các mô hình nhận diện phát âm từ khoá điều khiển cơ bản cho các thiết bị điện tử.

Tầm quan trọng của bài toán nhận diện phát âm từ khoá trong các thiết bị thông minh:

- Tăng doanh thu kinh doanh: Việc cho phép người dùng có thể điều khiển các thiết bị thông minh của họ thay vì chỉ làm việc cứng nhắc với những câu lệnh được lập trình sẵn, sẽ kích thích người dùng có xu hướng mua và sử dụng các thiết bị này nhiều hơn. Từ đó mà doanh thu từ các thiết bị thông minh cũng tăng theo.
- Tiết kiệm thời gian tương tác giữa con người với con người, con người với thiết bị: Thay vì phải mở khoá màn hình, bấm nghe cuộc gọi đến thì chỉ cần sử dụng lời nói hay cử chỉ bàn tay, một cách nhanh chóng có thể nhận được cuộc gọi ở đầu dây bên kia. Thay vì gõ chữ hay bấm nút điều khiển trên thiết bị, chỉ cần có một hệ thống nhận diện giọng nói đơn giản, mọi vấn đề phức tạp, khó khăn đó sẽ được giải quyết một cách nhanh chóng, dễ dàng.

- Tiết kiệm tài nguyên tính toán, chi phí sản xuất: Thay vì phải sử dụng các thiết bị phần cứng chuyên dụng, đắt đỏ thì ta hoàn toàn có thể xây dựng được các mô hình nhận diện thông minh với chi phí tính toán nhỏ hơn, tốn ít tài nguyên hơn. Nhờ đó mà chi phí sản xuất phần cứng cho các thiết bị này cũng sẽ rẻ hơn rất nhiều, tiếp cận phổ cập đến người dùng được nhiều hơn.
- Đóng góp một phần không nhỏ cho hệ sinh thái thông minh toàn cầu: Hiện nay, ở rất nhiều nơi đã áp dụng các mô hình nhà thông minh, camera nhận diện khuôn mặt trong các trường đại học, internet vạn vật, vv... Đó là các hệ sinh thái thông minh. Việc một thiết bị thông minh có đầy đủ các chức năng phục vụ con người như: nhận diện khuôn mặt, nhận diện giọng nói, nhận diện cử chỉ bàn tay, vv... sẽ góp phần tạo một hệ sinh thái thông minh hiện đại, sáng tạo, góp phần thúc đẩy phát triển kinh tế trong nước cũng như trên toàn thế giới.

1.2 Các vấn đề đặt ra

Để có thể xây dựng một mô hình nhận diện phát âm từ khoá dựa trên giọng nói của con người, có một số thách thức đặt ra cho các nhà phát triển như sau:

- Vấn đề thu thập dữ liệu: Hiện nay, trên internet đã công bố rất nhiều các bộ dữ liệu phát âm từ khoá cho nhiều ngôn ngữ khác nhau như: Tiếng Anh, tiếng Pháp, tiếng Trung Quốc, vv... Tuy nhiên, nguồn dữ liệu phát âm từ khoá cho tiếng Việt vẫn đang còn rất hạn chế. Một thách thức cơ bản tiếp theo là sự khác biệt khá rõ ràng về âm sắc giữa các vùng miền, ví dụ như: người miền Bắc phát âm khác người miền Trung, có nhiều người nói bị ngọng âm “l”, có nhiều người phát âm khá là khó nghe, vv... dẫn đến khó khăn trong việc phân bố dữ liệu huấn luyện, thông thường thì người ta chỉ tập chung xây dựng mô hình cho một phân bố dữ liệu cụ thể như: dữ liệu giọng nam miền Bắc, dữ liệu giọng nam miền Nam, vv...
- Dữ liệu âm thanh phải tương đối sạch: Việc có một chiếc micro thu âm chuẩn, chống nhiễu đối với một người dùng bình thường thực sự là không cần thiết. Chính vì vậy mà các âm thanh thu thập được khó tránh khỏi tác động của các yếu tố bên ngoài như: tiếng ồn xe cộ, tiếng rè của micro, hay tiếng micro quá nhỏ, ... Việc xử lý các đoạn âm thanh quá nhiều nhiễu, nghe không rõ ràng vẫn đang còn là thách thức rất lớn đối với các nhà phát triển ứng dụng.
- Mô hình nhận diện phải đủ nhỏ để có thể đưa ứng dụng được lên thiết bị: Với sự tiến bộ của công nghệ lưu trữ bộ nhớ, việc sở hữu một chiếc thẻ nhớ với dung lượng vài chục gigabyte, một thanh ram vài gigabyte hay một chiếc GPU thông thường không phải là điều quá khó khăn. Sự bùng lên của trí tuệ nhân tạo, sự phát triển của các thư viện có sẵn, giúp cho các lập trình viên có thể dễ dàng xây dựng các mô hình nhận diện giọng nói chỉ với một vài đoạn mã. Nhưng thách thức đặt ra là làm sao để các mô hình đó chạy được trên các thiết bị tài nguyên thấp, các thiết bị chỉ được cung cấp phần cứng phục vụ cho một mục đích cụ thể ?. Để giải quyết vấn

đề này thì người ta đã nghĩ ra một số phương pháp khác nhau giúp tối ưu, cắt tỉa các mô hình nhận diện để có thể chạy được trên các thiết bị bị giới hạn tài nguyên mà vẫn đảm bảo yêu cầu về độ chính xác, trong khi đó tốc độ xử lý tăng lên rất nhiều.

1.3 Giới thiệu bài toán

Nhận diện phát âm từ khoá trên các thiết bị tài nguyên thấp, thiết bị thông minh là một bài toán quan trọng trong lĩnh vực xử lý tiếng nói. Mục tiêu chính của bài toán này là xây dựng một mô hình có thể nhận diện chính xác các âm thanh từ khoá được định nghĩa trước bởi con người, yêu cầu đặt ra là mô hình nhận diện có thể chạy được trên các thiết bị có tài nguyên thấp, đồng thời thời gian phản hồi cho một giao dịch dự đoán cần phải nhanh nhất có thể. Để có thể xây dựng được mô hình như thế, cần phải trải qua các kỹ thuật bóc tách, xử lý, trích xuất các đặc trưng quan trọng của âm thanh, sau cùng là nhận diện chúng bằng các thuật toán học máy, học sâu.

Bài toán được giải quyết trong đồ án:

Là bài toán nhận diện phát âm của các từ khoá điều khiển cơ bản như: Một, Hai, Ba, Bốn, Lên, Xuống, Trái, Phải, vv... Dữ liệu được sử dụng trong đồ án là dữ liệu âm thanh được thu thập từ các bạn sinh viên trường đại học Bách Khoa Hà Nội. Mục tiêu của mô hình nhận diện này là có thể chạy được trên thiết bị tài nguyên thấp, thời gian dự đoán nhanh, độ chính xác cao, cụ thể thiết bị được thử nghiệm cho mô hình trong bài toán này là **JETSON NANO 2GB** [1]**.**

1.4 Các nghiên cứu liên quan

Một số nghiên cứu thế giới:

- Nhóm tác giả Trung Quốc [2] đề xuất giải pháp sử dụng mạng Deep Neural Network (DNN) [3] cùng với kỹ thuật Connectionist Temporal Classifier (CTC) [4]. Dữ liệu được sử dụng cho mô hình này dựa trên bộ dữ liệu Mandarin LVCSR corpus [2] gồm khoảng 2.500 giờ audio được dịch ra text, ước tính chừng khoảng 1.9 triệu từ.

  Họ sử dụng phương pháp Mel frequency cepstral coefficients (MFCC) [5] để bóc tách đặc trưng cho các âm thanh từ khoá, và đã đạt được tỷ lệ báo động nhầm (false alarm rate [6]) khoảng 1.5%. Hơn nữa, họ còn đạt được hiệu suất đáng khả quan của mô hình khi tiến hành đưa lên thiết bị điện thoại thông minh với độ trễ cho một lần dự đoán là 0.3 giây.
- Nhóm tác giả Tara N. Sainath, Carolina Parada (New York, USA) [7] tập chung vào việc tối ưu số lượng tham số cũng như khối lượng tính toán của mô hình dựa trên mạng tích chập Convolutional Neural Network (CNN) [8]. Phương pháp của họ đề xuất đã đạt được kết quả với tỷ lệ false alarm rate tốt hơn khoảng 27 - 44% so với các mô hình DNN truyền thống.
- Một nghiên cứu khác đến từ 2 tác giả Raphael Tang, Jimmy Lin [9] đề xuất sử dụng cấu trúc mạng Deep residual networks (ResNets) [10] kết hợp với việc trích xuất đặc trưng MFCC cho bộ dữ liệu Google Speech Commands Dataset v1 [11] , đã đạt được tỷ lệ chính xác trên tập kiểm tra

khoảng 95.8%. Tuy nhiên thì nhóm tác giả vẫn chưa đưa ra được kết quả thử nghiệm về độ trễ thời gian dự đoán trên các thiết bị tài nguyên thấp.

- Cũng dựa trên bộ dữ liệu Google Speech Commands Dataset này, với việc áp dụng thêm cơ chế self-attention [12] nhóm tác giả tại đại học Lund [13] đã đạt được kết quả tốt nhất so với các mô hình khác cho đến thời điểm hiện tại, mô hình đạt độ chính xác khoảng 98.6% cho tập dữ liệu kiểm tra phiên bản 2 và 97.7% cho tập dữ liệu kiểm tra phiên bản 1. Mặc dù mô hình đạt được độ chính xác khá cao nhưng do cấu trúc khá phức tạp của mô hình đề xuất, có thể sẽ dẫn đến một vấn đề là độ trễ thời gian dự đoán trên các thiết bị tài nguyên thấp sẽ nhiều.

Một số nghiên cứu tại Việt Nam:

- Nhóm tác giả tại trường đại học FPT [14] xây dựng mô hình nhận diện phát âm các từ khoá điều khiển nhà thông minh sử dụng mạng hồi quy Long short-term memory (LSTM) [15] kết hợp với đặc trưng MFCC. Dữ liệu được thu thập lấy ý tưởng dựa trên các class của bộ dữ liệu Google Speech Commands Dataset với khoảng 3200 mẫu âm thanh cho mỗi class dữ liệu. Kết quả cao nhất mà mô hình của nhóm đạt được là 94.3% trên tập kiểm tra.
- Nhóm tác giả Nguyen Huu Binh, Nguyen Quoc Cuong, Tran Thi Anh Xuan [16] tập chung chủ yếu vào việc phân tích các yếu tố ảnh hưởng bên ngoài vào một hệ thống phân lớp phát âm các từ khoá, cụ thể họ tiến hành thực nghiệm mô hình nhận diện phát âm các từ khoá trên nhiều môi trường nhiễu khác nhau. Cùng với đó họ cũng so sánh hiệu năng mô hình dựa trên các khoảng cách khác nhau từ người nói đến thiết bị ghi âm được sử dụng. Kết quả cho thấy, mô hình học được trên bộ dữ liệu ở môi trường nhiễu có độ chính xác cao hơn so với mô hình học được trên bộ dữ liệu ở môi trường sạch (không có tiếng ồn).

1.5 Hướng tiếp cận của đồ án

Với sự thành công vượt bậc trong lĩnh vực trí tuệ nhân tạo của các ông lớn trên thế giới như: Google, Facebook, Amazon, vv... đặc biệt là trong lĩnh vực xử lý tiếng nói, câu hỏi đặt ra là làm sao để có thể xây dựng được một mô hình nhận diện tiếng nói nói chung cũng như mô hình nhận diện phát âm từ khoá nói riêng cho dữ liệu âm thanh Tiếng Việt.

Đồ án này sẽ trình bày các phương pháp trích xuất đặc trưng của âm thanh, các kỹ thuật xử lý tín hiệu âm thanh kỹ thuật số, thử nghiệm một số mô hình học máy, học sâu, xây dựng mô hình để nhận diện phát âm các từ khoá dựa trên bộ dữ liệu âm thanh được thu thập từ các bạn sinh viên trường đại học Bách Khoa Hà Nội. Đồ án cũng sẽ thử nghiệm các mô hình nhận diện phát âm từ khoá trên thiết bị **JETSON NANO 2GB**, so sánh về độ chính xác, thời gian tính toán giữa các mô hình.

Dưới sự thành công vượt bậc của mạng tích chập CNN trong nhiều lĩnh vực trí tuệ nhân tạo trong đó có lĩnh vực xử lý tiếng nói, đồ án này sẽ tập chung khai tác tối đa lợi thế của cấu trúc mạng này, mô hình được sử dụng trong đồ án là

TC-RESNET [17], một biến thể của cấu trúc mạng nổi tiếng RESNET [10] với số lượng tham số ít hơn, khối lượng tính toán ít hơn, độ chính xác cao hơn, rất thích hợp cho việc tích hợp vào các thiết bị tài nguyên thấp. Đồng thời cũng sẽ thử nghiệm thêm một số thuật toán học máy đơn giản như SVM [18], một số mô hình sử dụng cấu trúc mạng hồi quy như RNN [19], LSTM [15] để so sánh tính hiệu quả cũng như tính ứng dụng giữa các mô hình.

Tóm lại, các giai đoạn chính để xây dựng mô hình nhận diện phát âm từ khoá trên thiết bị bao gồm các bước như sau:

- Thu thập, tiền xử lý dữ liệu âm thanh từ sinh viên trường đại học Bách Khoa Hà Nội
- Sử dụng một số kỹ thuật xử lý âm thanh như: lọc nhiễu, nâng giảm cao độ, chuẩn hoá âm lượng, tua nhanh âm thanh nhằm mục đích làm giàu và đa dạng dữ liệu.
- Trích xuất các đặc trưng quan trọng của âm thanh sử dụng phương pháp MFCC
- Huấn luyện mô hình học máy, học sâu dựa trên các đặc trưng thu được
- Thực hiện kiểm tra độ chính xác, thời gian dự đoán của các mô hình học được trên thiết bị **JETSON NANO 2GB** và chứng minh tính ứng dụng của các mô hình.

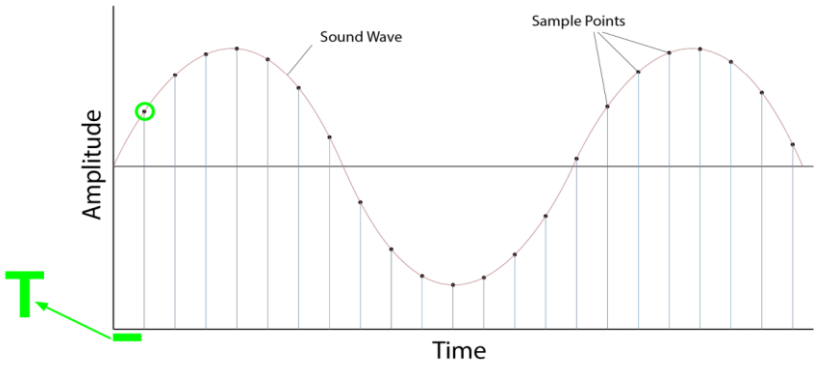
CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1 Giới thiệu chung về tín hiệu âm thanh

2.1.1 Tín hiệu âm thanh kỹ thuật số

Âm thanh mà chúng ta nghe thấy hàng ngày được đặc trưng bởi một dạng sóng, mà sóng có dạng là một hàm số liên tục cho nên âm thanh cũng có dạng là các tín hiệu liên tục. Trong khi đó, các máy tính hiện đại ngày nay của chúng ta, thường làm việc với các con số rời rạc, vì vậy cần một cơ chế nào đó để có thể chuyển từ tín hiệu liên tục về tín hiệu rời rạc, cũng như chuyển từ tín hiệu rời rạc về tín hiệu liên tục.

Để có thể chuyển từ tín hiệu âm thanh liên tục sang tín hiệu âm thanh rời rạc, người ta thường sử dụng phương pháp lấy mẫu (sampling). Ta cần lấy mẫu tại các điểm thời gian cách đều nhau với một tần số lấy mẫu xác định (sample rate), theo định lý lấy mẫu **Nyquist–Shannon** [20] thì với 1 tín hiệu có các tần số thành phần $\leq f_m$, để đảm bảo việc lấy mẫu không làm mất mát thông tin (aliasing) thì tần số lấy mẫu f_s phải đảm bảo $f_s \geq 2f_m$, trong khi đó tai người chỉ nghe được âm thanh trong khoảng từ f_m = 20Hz đến f_m = 20000Hz cho nên người ta thường lấy tần số lấy mẫu f_s = 44100Hz - tức là trong 1 giây lấy 44100 giá trị. Ngoài ra, ở một số trường hợp cụ thể, người ta cũng có thể dùng các giá trị khác như: 8000Hz, 16000Hz hoặc 48000Hz.



Hình 2.1: Mô tả việc lấy mẫu âm thanh (Tham số T biểu thị khoảng thời gian giữa 2 điểm lấy mẫu gần nhất [21])

Do là sự tổng hợp của vô số các dạng sóng có tần số và biên độ cụ thể cho nên bất kỳ tín hiệu âm thanh nào cũng đều có một số đặc trưng cơ bản [22]. Hầu hết tín hiệu âm thanh đều ổn định trong khoảng thời gian ngắn, thông thường là 20 đến 25ms tuỳ từng ngôn ngữ. Khi phân tích tín hiệu âm thanh, chúng ta thường sử dụng các phân tích ngắn hạn, tức là người ta thường phân khung (frame) thời gian cho tín hiệu âm thanh, mỗi khung là một đơn vị cơ bản để phân tích tín hiệu.

Trong mỗi khung, ta có thể quan sát 3 đặc tính âm học cơ bản nhất như:

- Âm lượng (Volume): đặc tính này đại diện cho độ lớn (Loudness) của tín hiệu âm thanh, tương quan với biên độ (Amplitude) của tín hiệu (biên độ của tín hiệu càng cao thì âm lượng của tín hiệu càng lớn)

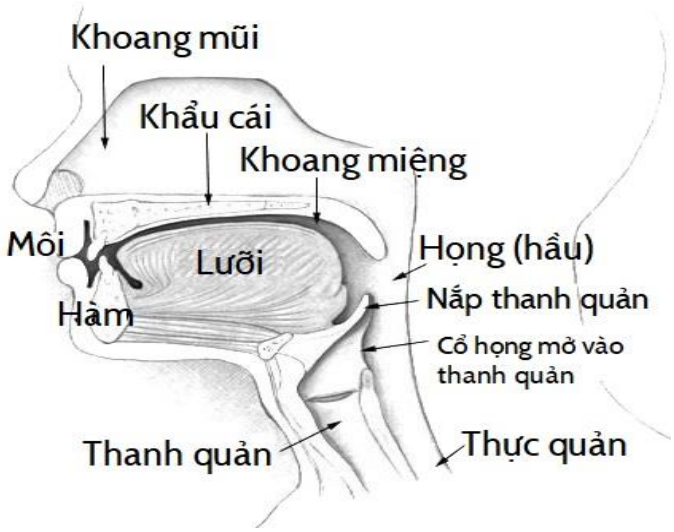
- Độ cao (Pitch): đặc tính này đại diện cho tỷ lệ rung của âm thanh, có thể biểu diễn bằng tần số cơ bản hoặc tương đương, nghịch đảo của chu kỳ cơ bản của tín hiệu tiếng nói.
- Âm sắc (Timbre): đặc tính này đại diện cho nội dung ngữ nghĩa (chẳng hạn một nguyên âm trong Tiếng Việt) của tín hiệu âm thanh, mà được đặc trưng bởi dạng sóng trong một chu kỳ cơ bản của tín hiệu tiếng nói.

Để tiện cho việc phát triển các thuật toán xử lý tín hiệu âm thanh, người ta định nghĩa ra một số các thuật ngữ thường được sử dụng trong phân tích khung tín hiệu:

- Kích thước khung (frame size): số điểm mẫu trong mỗi khung
- Chồng lấp khung (frame overlap): số điểm mẫu chồng lấp giữa các khung liên tiếp
- Bước khung (hop length): bằng với kích thước khung trừ đi khoảng chồng lấp
- Tỷ lệ khung (frame rate): số các khung trên một giây, bằng với tần số mẫu chia cho bước khung

2.1.2 Tiếng nói con người

- **Nguyên lý hình thành tiếng nói [23]**

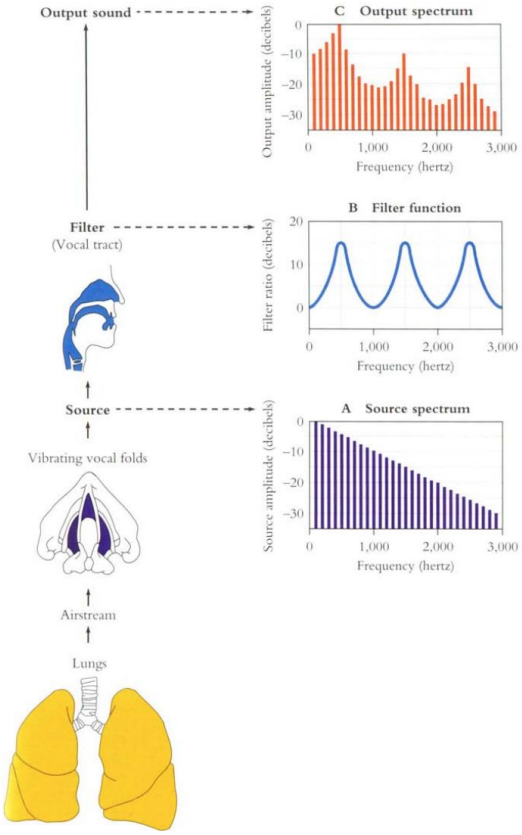


Hình 2.2: Các cơ quan phát ra tiếng nói của con người (nguồn: vi.wikipedia.org)

Tiếng nói của con người phát ra dựa trên hoạt động chủ yếu của một bộ phận, đó chính là thanh quản (Vocal folds). Khi ta nói một câu, phổi sẽ cung cấp một luồng không khí (hơi) để tạo ra một áp lực lên thanh quản. Áp lực này làm cho thanh quản được mở ra để luồng không khí có thể đi qua, khi áp lực giảm xuống thanh quản có cơ chế tự động đóng lại. Vì quá trình cung cấp không khí của phổi diễn ra không đồng đều cho nên việc đóng mở thanh quản như vậy sẽ khiến áp lực lại tăng lên và quá trình tái diễn. Dẫn đến sinh ra một khái niệm là chu kì đóng/mở thanh quản, tạo ra các tần số sóng âm với tần số cơ bản khoảng 125Hz đối với nam, 210Hz đối với nữ. Giờ thì ta đã hiểu vì sao giọng của các

bạn nữ thường có xu hướng cao hơn giọng của các bạn nam. Tần số này được gọi là fundamental frequency (tần số cơ bản) F0.

Chỉ có hoạt động của thanh quản thôi là chưa đủ. Để có thể hình thành lên tiếng nói còn cần đến các cơ quan quan trọng khác như: vòm họng, khoang miệng, lưỡi, răng, môi, mũi... Nếu để ý ở các thiết bị nhạc cụ như: guitar, sáo trúc, để các thiết bị này có thể hoạt động, chúng phải có một yếu tố rất quan trọng khác đó là một "bộ cộng hưởng". Các cơ quan đã nói ở trên cũng hoạt động theo cơ chế đó, khác ở chỗ là chúng có thể thay đổi hình dạng một cách linh hoạt. Bộ cộng hưởng thông minh của con người có tác dụng triệt tiêu một vài tần số hoặc khuếch đại 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 2.3: Quá trình tạo ra tiếng nói của con người [23]

Hình 2.3 mô tả rất chi tiết về cơ chế này. Source + Filter ⟶ Output sound . Tại spectrum của output sound, ta thấy có 3 đỉnh, các đỉnh này lần lượt gọi là các đỉnh F1, F2, F3 ... hay còn gọi là các formant. Giá trị, vị trí, sự thay đổi theo thời gian của các đỉnh này đặc trưng cho các âm vị (âm vị: trong nhiều loại ngôn ngữ, một kí tự/cụm kí tự trong các từ khác nhau có thể có nhiều cách phát âm khác nhau). Trong các phương pháp nhận diện giọng nói truyền thống, người ta thường cố gắng tách thông tin về các formant này ra khỏi F0 rồi mới sử dụng thông tin đó để nhận diện.

- **Cơ chế hoạt động của tai người [23]**

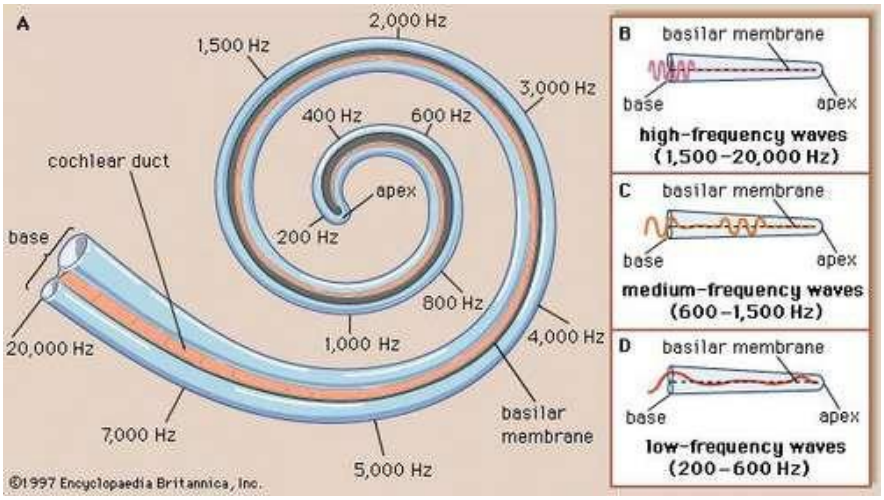
Trong nhận dạng tiếng nói, việc hiểu được cơ chế "nghe" của con người còn quan trọng hơn cách "nói". Như chúng ta đã biết, âm thanh, tiếng nói mà

chúng ta vẫn nghe hàng ngày là tổng hợp của rất nhiều sóng với các tần số, biên độ khác nhau. Các tần số này thường nằm trong khoảng từ 20Hz ⟶ 20000Hz. Tuy nhiên tai người (và các loài động vật) lại hoạt động một cách phi tuyến tính, tức là không phải độ cảm nhận âm thanh 10000Hz sẽ gấp 100 lần âm thanh 100Hz. Thường thì tai người rất nhạy cảm ở các âm thanh tần số thấp, kém nhạy cảm ở các âm thanh tần số cao. Quá trình tai của chúng ta hoạt động được miêu tả thông qua 3 bước như sau:

Bước 1: Các sóng âm thanh truyền tới tai, khiến cho các phân tử trong không khí va đập vào màng nhĩ, do có áp suất cho nên màng nhĩ lúc này sẽ rung lên.

Bước 2: Rung động của màng nhĩ sẽ truyền tới 3 xương nhỏ: malleus, incus, stapes, rồi sau đó truyền tới ốc tai (Ốc tai là một bộ phận dạng xoắn, rỗng như một con ốc)

Bước 3: Âm thanh được truyền dọc theo ốc tai, trong quá trình truyền, các sóng âm thanh sẽ tiếp xúc, va chạm với các tế bào lông cảm nhận âm thanh. Các tế bào này sẽ gửi tín hiệu đến não bộ khi có sóng truyền qua. Các tế bào ở phần đầu rắn hơn, rung động với tần số cao, các tế bào ở cuối mảnh hơn, rung động với tần số thấp. Do cấu tạo của ốc tai cùng với số lượng các tế bào lông mảnh chiếm phần lớn khiến cho việc cảm nhận của tai người (và động vật) trở nên phi tuyến tính, nhạy cảm ở tần số thấp, kém nhạy cảm hơn ở tần số cao.

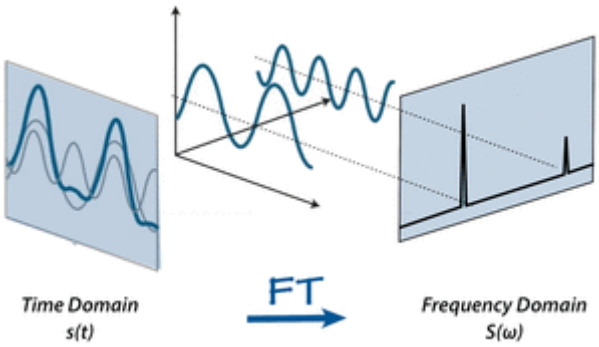


Hình 2.4: Cơ chế hoạt động của tai người [23]

2.2 Biến đổi fourier (Fourier Transform)

2.2.1 Ý tưởng chung

Âm thanh là một chuỗi tín hiệu theo thời gian, một đoạn âm thanh dài hai phút nhưng hàm lượng thông tin hữu ích có thể chỉ tập trung trong khoảng một phút giữa đoạn. Không những thế, âm thanh còn là một dạng hỗn tạp từ các sóng có tần số khác nhau, thắc mắc được đặt ra tại sao ta không tìm cách lưu trữ các đoạn âm thanh này ở dạng ngắn gọn hơn, giàu thông tin hơn bằng cách lưu chúng dưới dạng các tần số và biên độ cụ thể.

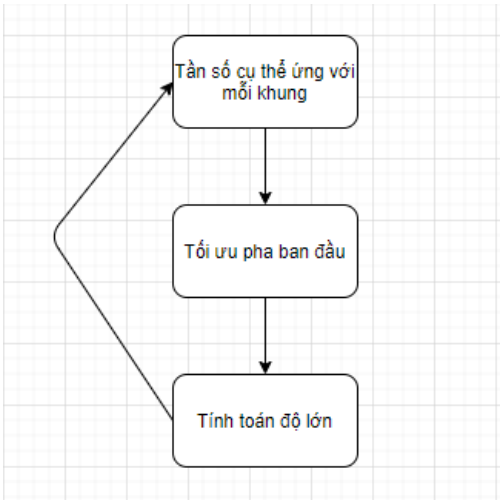


Hình 2.5: Biến đổi fourier cho 2 sóng sin tuần hoàn [23]

Trong hình 2.5, đoạn tín hiệu trong miền thời gian là sự kết hợp của 2 sóng tuần hoàn hình sin. 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 đó - biến đổi fourier [21]. Ta có một biểu diễn trực quan hơn, đơn giản hơn cho đoạn tín hiệu (một cách biểu diễn giàu thông tin hơn).

Xuất phát từ ý tưởng về sự tương đồng giữa tín hiệu âm thanh với các sóng hình sin ứng với những tần số khác nhau, sau khi thực hiện phân khung cho tín hiệu âm thanh, ta sẽ thực hiện biến đổi fourier trên mỗi khung này. Do âm thanh ổn định trên các khoảng thời gian ngắn hạn, cho nên mỗi khung sẽ có một tần số nhất định. Với mỗi tần số ta sẽ lưu 2 giá trị là độ lớn (magnitude) và pha ban đầu (phase). Giá trị độ lớn ở đây thể hiện mức độ tương đồng giữa tín hiệu với một sóng hình sin tại một tần số cụ thể nào đó. Nó phụ thuộc vào hàm số của pha ban đầu. Độ lớn cao chỉ ra rằng sự tương đồng này là lớn, và ngược lại.

Hình vẽ dưới đây mô tả các bước biến đổi fourier trên một khung thời gian ngắn hạn:



Hình 2.6: Các bước biến đổi fourier cơ bản cho một khung thời gian ngắn hạn

Như vậy, với Fourier Transfrom, ta đã chuyển đổi thông tin từ miền thời gian sang miền tần số. Ngược lại, ta có Inverse Fourier transform [21] (biến đổi Fourier ngược) để chuyển đổi thông tin từ miền tần số về miền thời gian. Fourier transform có ứng dụng rất lớn trong lĩnh vực xử lý tín hiệu (âm thanh, ảnh, thông tin).

2.2.2 Fourier cho miền giá trị liên tục

Công thức biến đổi fourier cho miền giá trị liên tục [21]:

$$\varphi_f = argmax_{\varphi \in [0,1)}(\int s(t) \cdot sin(2\pi \cdot (ft - \varphi))\, dt)$$

$$d_f = max_{\varphi \in [0,1)}(\int s(t) \cdot sin(2\pi \cdot (ft - \varphi))\, dt)$$

Trong đó:

- $s(t)$: Hàm số phụ thuộc vào thời gian của tín hiệu âm thanh. Giá trị của hàm tại mỗi thời điểm *t* xác định là giá trị biên độ của tín hiệu tại thời điểm đó.
- $\sin(2\pi \cdot (ft - \varphi))$: Hàm số sin để tính độ tương đồng với tín hiệu âm thanh
- φ: Pha ban đầu của hàm số *sin* cần tối ưu
- f: Tần số của tín hiệu âm thanh trong một khung thời gian nhất định

Công thức biểu diễn dưới dạng số phức:

$$e^{i\gamma} = cos(\gamma) + isin(\gamma)$$

$$\hat{s}(f) = \int s(t) \cdot e^{-i2\pi ft} dt = \int s(t) \cdot cos(-2\pi ft) dt + i \int s(t) \cdot sin(-2\pi ft) dt$$

Trong đó: $d_f = \sqrt{2}|\hat{s}(f)|$, $\varphi_f = -\frac{\gamma_f}{2\pi}$

2.2.3 Fourier cho miền giá trị rời rạc

Công thức biến đổi fourier cho miền giá trị rời rạc - Discrete Fourier Transform (DFT) [21]:

$$k = [0, M-1] = [0, N-1]$$

$$\hat{s}\left(\frac{k}{N}\right) = \sum_{n=0}^{N-1} s(n) \cdot e^{-i2\pi n \frac{k}{N}}$$

$$T = \frac{1}{sample\_rate}, Frequency(k) = \frac{k}{NT} = \frac{k\ x\ sample\_rate}{N}$$

Trong đó: $M = N$ là số điểm mẫu của tín hiệu âm thanh được tính bằng tổng thời gian của tín hiệu âm thanh nhân với tần số lấy mẫu:

$$N = time\_of\_audio\_signal * sample\_rate$$

2.3 Đặc trưng MFCC (Mel Frequency Cepstral Coefficents)

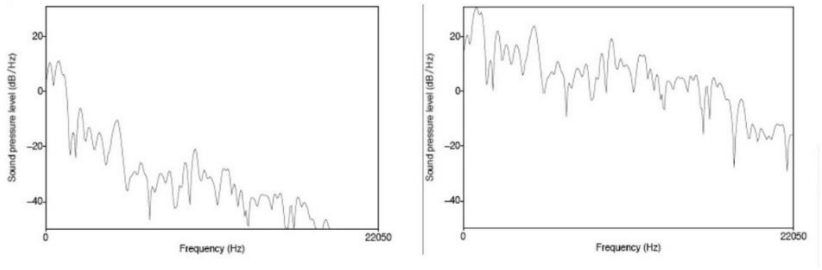
Mel Frequency Cepstral Coefficents – MFCC [5] là phương pháp trích xuất đặc trưng được sử dụng rộng rãi trong các bài toán nhận diện giọng nói, phát hiện âm thanh ngày nay. Được đề xuất bởi Davis và Mermelstein vào năm 1980, cho đến nay thì nó vẫn luôn là lựa chọn số một kể từ khi ra mắt. Các bước để trích xuất đặc trưng MFCC cho một tín hiệu âm thanh gồm có 5 giai đoạn chính: Pre-emphasis, Spectrogram, Mel filterbank, Cepstrum và MFCC.

2.3.1 Pre-emphasis

Nếu chúng ta để ý kỹ một chút, thì các ca sỹ khi hát ở tông chuẩn của họ thì âm thanh họ phát ra rất to và rõ, nhưng khi họ hát ở một tông cao hơn, tức là ở

một tần số cao hơn thì âm thanh phát ra có phần nhỏ hơn và bị ghìm hơn. Đây là chính là một đặc điểm quan trọng của tiếng nói con người: các âm ở tần số thấp có mức năng lượng cao, các âm ở tần số cao lại có mức năng lượng khá thấp. Trong khi đó, các tần số cao này vẫn mang nhiều thông tin về âm vị. Vì vậy người ta đã nghĩ ra một cơ chế để kích các tín hiệu ở tần số cao này lên.

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

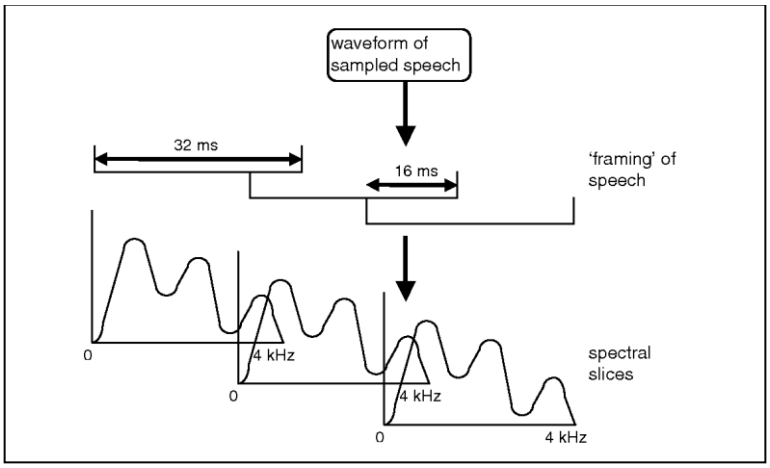


Hình 2.7: Minh hoạ quá trình Pre-empphasis [5]

2.3.2 Spectrogram

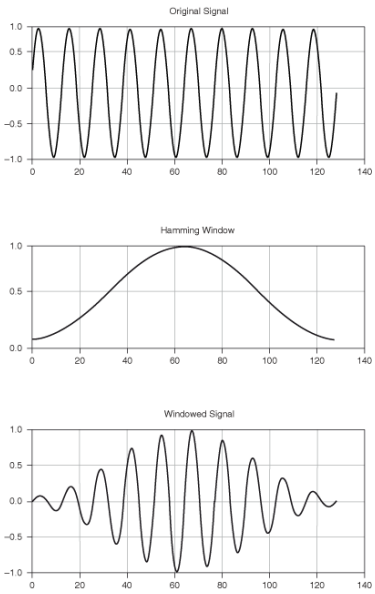
- **Windowing**

Như đã nói ở trên, âm thanh thường ổn định trong các khoảng thời gian ngắn hạn, tiến hành thực hiện cắt frame (phân khung) cho tín hiệu âm thanh rồi mới áp dụng biến đổi fourier trên từng frame này (DFT - Discrete Fourier Transform). Độ rộng mỗi frame khoảng 20 đến 25ms là vừa đủ để bao hết 1 phần âm thanh. Các frame được chồng lấp lên nhau khoảng 10 đến 15ms để có thể lưu lại sự thay đổi thông tin giữa 2 frame liền kề.



Hình 2.8: Minh hoạ quá trình cắt frame trên 1 đoạn tín hiệu âm thanh (nguồn: www.semanticscholar.org)

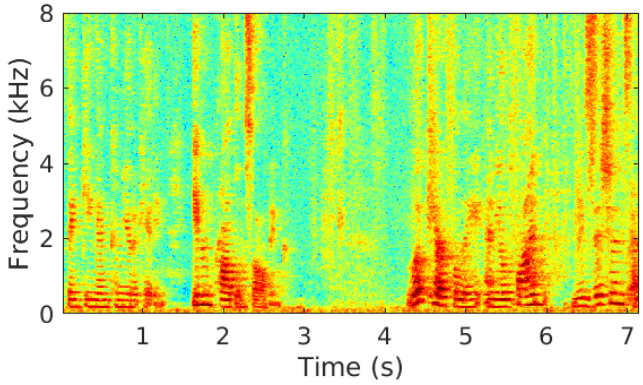
Ở chiều thời gian, ứng với mỗi thời điểm *t* xác định sẽ là giá trị biên độ của tín hiệu tại thời điểm đó. Khi đó, việc cắt frame sẽ gây ra một vấn đề lớn là giá trị ở điểm đầu và điểm cuối của mỗi frame sẽ bị quy đột ngột về giá trị khác phân phối với giá trị ở vùng trong của frame. Khi thực hiện biến đổi fourier trên frame này sẽ dẫn tới một hiện tượng là: các miền tần số cao sẽ chứa rất nhiều nhiễu. Bằng cách nhân chập frame với một số window phổ biến như: Hamming window, Hanning window, giá trị ở 2 biên của frame sẽ được giảm xuống từ từ.



Hình 2.9: Phương pháp Hamming window cho tín hiệu âm thanh (nguồn: www.physik.uzh.ch)

- **DFT**

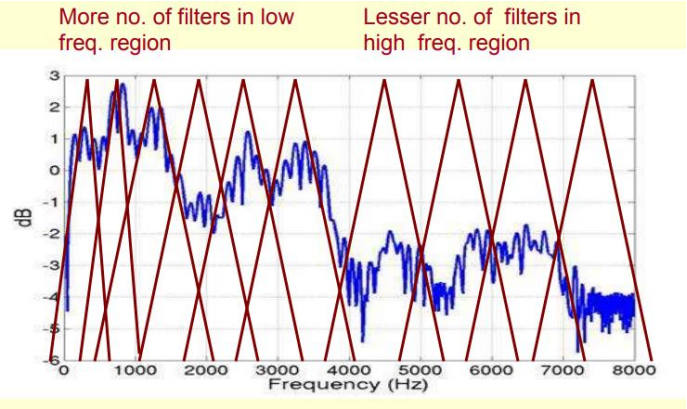
Sau khi thực hiện biến đổi fourier cho từng đoạn frame của tín hiệu, ứng với mỗi frame ta sẽ thu được 1 danh sách các giá trị độ lớn (magnitude) tương ứng với từng tần số từ $0 \rightarrow N$. (N ở đây là số điểm mẫu trong một khung tín hiệu âm thanh). Đây chính là một trong số các đặc trưng cơ bản được sử dụng trong các bài toán xử lý tiếng nói: Spectrogram. Hình 2.10 minh hoạ trục x là trục thời gian (tương ứng với thứ tự các frame), trục y thể hiện dải tần số từ $0 \rightarrow$ 8KHz, giá trị magnitude tại từng tần số được thể hiện bằng màu sắc. Màu đậm thể hiện mức năng lượng cao, màu nhạt thể hiện mức năng lượng thấp.



Hình 2.10: Đặc trưng spectrogram (nguồn: www.researchgate.net)

2.3.3 Mel filterbank (Mel spectrogram)

Do đặc điểm về sinh học của tai người nên chúng ta cũng có một cách cảm nhận âm thanh khá khác so với các thiết bị đo. Các âm thanh ở tần số thấp, chúng ta cảm nhận rất tốt nhưng lại kém nhạy cảm ở các tần số cao hơn. Vì vậy, cần một cơ chế để ánh xạ giữa tín hiệu âm thanh trong các thiết bị với tai nghe của con người.



Hình 2.11: Minh hoạ quá trình Mel filterbank (nguồn: viblo.asia)

Quá trình xây dựng đặc trưng Mel-spectrogram gồm 2 bước chính sau đây:

Bước 1: Bình phương các giá trị trong spectrogram thu được dạng phổ công suất (DFT power spectrum).

Bước 2: Trên mỗi một dải tần số ta sẽ áp dụng một trong số các bộ lọc thông dải Mel-Frequency filter [24]. Giá trị đầu ra của từng filter chính là năng lượng của dải tần mà filter đó bao phủ. Tổng hợp kết quả của các dải tần lại ta thu được đặc trưng Mel-spectrogram.

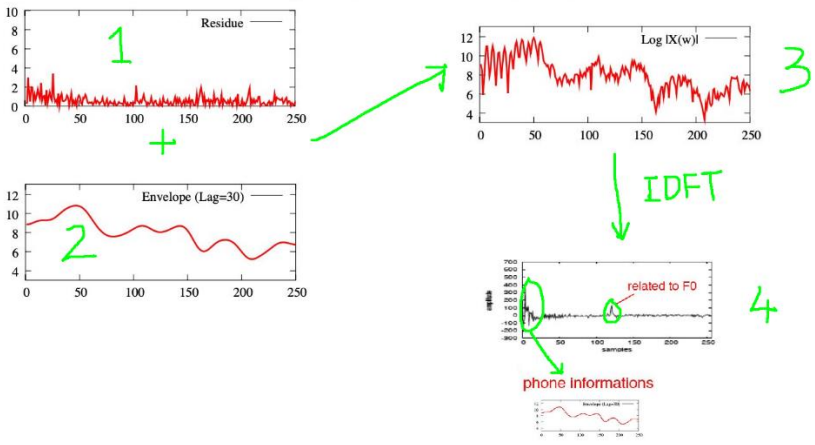
2.3.4 Cepstrum

- **Log**

Thực tế cho thấy 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 log trên Mel-spectrogram. Điều này còn giúp giảm thiểu các biến thể âm thanh không đáng kể trong nhận diện giọng nói.

- **Biến đổi fourier ngược (IDFT - Inverse DFT)**

Giọng nói của chúng ta có một tần số cơ bản F0. Tần số này ở nam giới khoảng 125Hz, ở nữ thì khoảng 210Hz. Đây là đặc trưng chính cho cao độ giọng nói của từng người. Thông tin về cao độ này không giúp ích trong nhận diện giọng nói, nên ta cần tìm cách để loại thông tin về F0 đi, giúp các mô hình nhận diện không bị phụ thuộc vào cao độ giọng của từng người.



Hình 2.12: Minh hoạ quá trình Cepstrum [5]

Hình 2.12 chính là phổ năng lượng Mel-spectrogram mà chúng ta thu được ở bước trước. Thông tin quan trọng chúng ta cần là phần 2, thông tin cần loại bỏ là phần 1. Để loại bỏ đi thông tin về F0, ta sử dụng phép biến đổi Fourier ngược (IDFT) về miền thời gian, thu được Cepstrum.

Khi đó, với Cepstrum thu được, phần thông tin liên quan tới F0 và phần thông tin liên quan tới các tần số khác nằm tách biệt nhau như hai phần khoanh tròn trong hình 4. Ta chỉ đơn giản lấy thông tin trong phần đầu của cepstrum.

2.3.5 Đặc trưng MFCC

Đặc trưng MFCC cho một frame sẽ bao gồm 3 phần chính:

- 12 feature đầu tiên được lấy từ cepstrum

- Feature thứ 13 là năng lượng của frame đó

- 13 feature tiếp theo là đạo hàm bậc 1 (theo thời gian) của 12 feature đầu tiên: Mô tả thông tin về sự thay đổi năng lượng từ frame thứ *t* đến frame thứ *t + 1*.

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

- 13 feature cuối cùng là đạo hàm bậc 2 (theo thời gian) của 12 feature đầu tiên

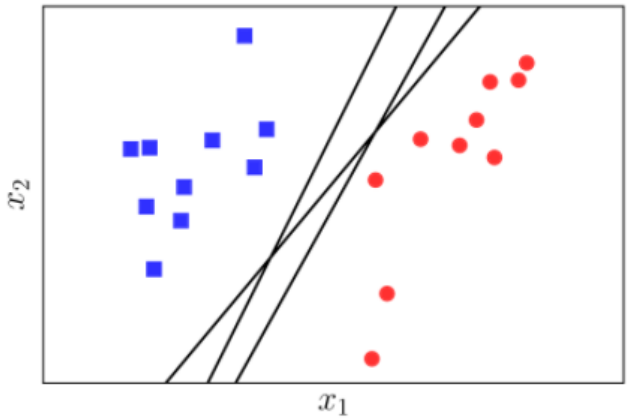
$$b(t) = \frac{d(t+1) - d(t-1)}{2}$$

Kết hợp 3 phần này lại ta sẽ có 39 feature. Đây là chính là đặc trưng MFCC cần tìm. Trong một số bài toán cụ thể, ta có thể lựa chọn số hệ số MFCC phù hợp mà không nhất thiết phải tuân theo các công thức trên.

2.4 Một số mô hình học máy, học sâu

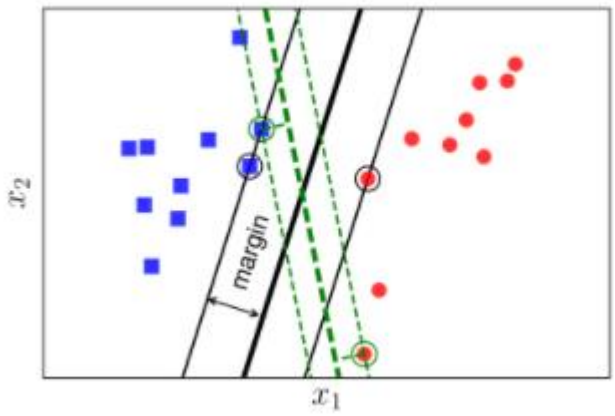
2.4.1 Mô hình SVM (Support Vector Machine)

SVM [18] bản chất là bài toán tìm một siêu mặt phẳng (hyperplane) để phân chia hai lớp trong bài toán phân loại nhị phân. Có vô số mặt phẳng như vậy (hình 2.13)



Hình 2.13: Các mặt phân cách 2 class trong bài toán phân loại nhị phân [18]

Thuật toán SVM mục tiêu tìm ra một đường phân chia sao cho khoảng cách từ điểm gần nhất trong mỗi lớp tới đường phân chia là bằng nhau. Khi khoảng cách này đã bằng nhau rồi thì còn cần thêm một yếu tố nữa là làm sao cho nó lớn nhất có thể. Đại lượng khoảng cách này được gọi chung là margin.



Hình 2.14: Margin của 2 lớp là bằng nhau và lớn nhất có thể [18]

Bài toán tối ưu trong SVM chính là bài toán tìm ma trận trọng số *w* và hệ số bias *b* sao cho margin này đạt giá trị lớn nhất:

$$(w, b) = argmax_{w,b}\{\frac{1}{||w||^2} min_n y_n(w^T x_n + b)\}$$

Một số tham số cần chú ý:

- C: Độ thiên vị về dữ liệu nhiễu, C càng cao thì càng thiên vị, C càng thấp thì càng không thiên vị
- Kernel: Thuật toán SVM thực hiện "chiếu" chiều của dữ liệu sang các hệ toạ độ khác (hệ toạ độ mà dữ liệu thuộc hai lớp với mong muốn có thể phân chia được) thông qua các kernel phổ biến: Linear, Rbf, ...
- Gamma: Tham số nhằm điều chỉnh mức độ quá khớp (overfit) của mô hình

- **Kỹ thuật One and Rest**

Đối với các bài toán phân loại nhiều lớp (class), ta coi một lớp làm gốc, các lớp còn lại được coi là lớp thứ hai. Giả sử có *n* lớp, thực hiện *n* bộ phân loại nhị phân. Bộ nào cho kết quả tốt nhất thì sẽ được dùng để cập nhật trọng số.

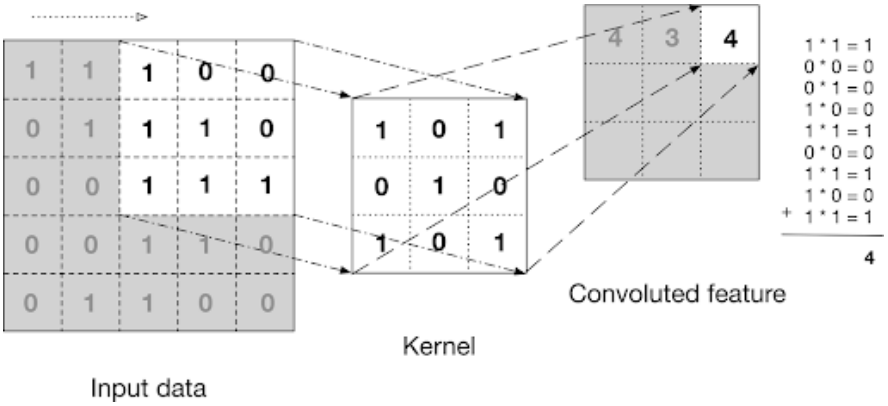
2.4.2 Mạng tích chập CNN (Convolutional Neural Network)

Mạng nơ-ron tích chập (CNN) [25] là các mạng được thiết kế để làm việc với những dữ liệu đầu vào có cấu trúc dạng lưới, ở đó những đặc trưng trong các vùng cục bộ của lưới được đáng chú ý hơn. Các trường hợp đặc biệt khác của dữ liệu cấu trúc lưới có thể kể đến như: văn bản, chuỗi thời gian, chuỗi tín hiệu, ...

- **Phép tích chập [25]**

Một phép tính tích chập là một phép toán nhân tích vô hướng giữa một tập trọng số cấu trúc lưới (filter hay còn được gọi là kernel) với các đầu vào có cấu trúc lưới tương tự được rút ra từ các không gian cục bộ khác nhau trong khối

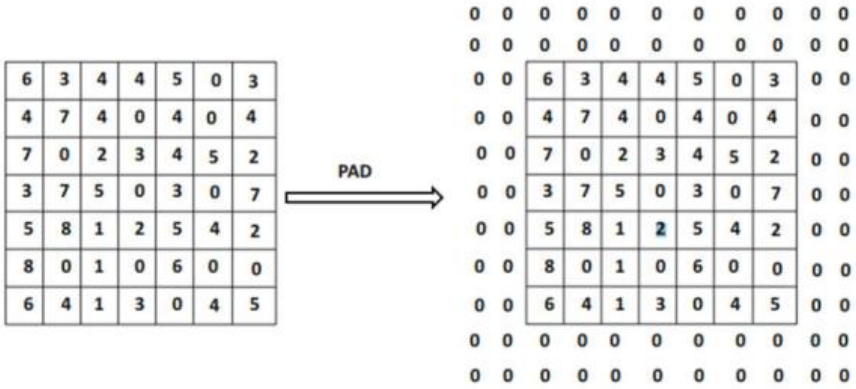
lượng đầu vào. Loại phép tính này hữu ích với những dữ liệu có mức độ phụ thuộc không gian hoặc cục bộ khá cao, điển hình như là dữ liệu về hình ảnh. Vì thế, CNN được định nghĩa như một mạng sử dụng phép tính tích chập trong ít nhất một lớp, mặc dù hầu hết các mạng CNN sử dụng phép tính này trong nhiều lớp.



Hình 2.15: Phương pháp tính tích chập [25]

- **Padding [25]**

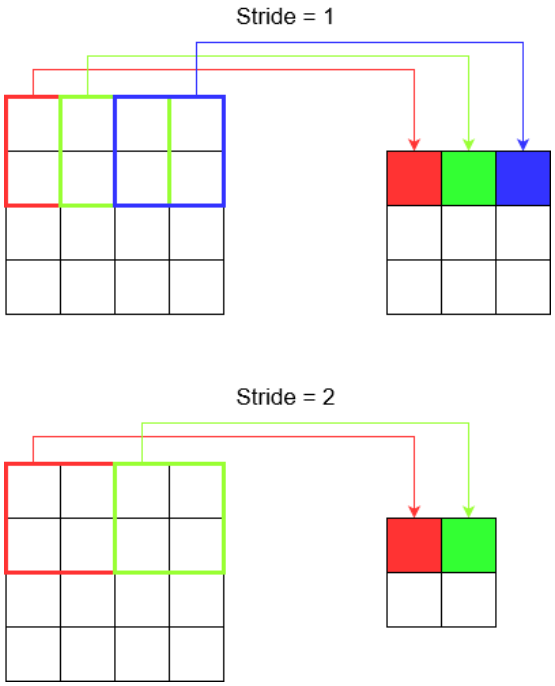
Quan sát thấy rằng các thao tác chập làm giảm kích thước của lớp thứ (*m + 1*) so với kích thước của lớp *m*. Loại giảm kích thước này có xu hướng làm mất mất một số thông tin dọc theo đường biên của hình ảnh (hoặc của feature map, trong trường hợp của các lớp ẩn). Để giải quyết vấn đề này, người ta sử dụng padding. Trong padding, ta sẽ thêm một lượng pixel có giá trị bằng nhau (thông thường là giá trị 0) xung quanh các đường biên của feature map.



Hình 2.16: Padding trong CNN [25]

- **Strides [25]**

Mức độ chi tiết của convolution được thể hiện thông qua khái niệm về các bước tiến (strides). Stride càng nhỏ thì mức độ chi tiết của CNN càng cao, stride càng lớn thì mức độ chi tiết của CNN càng thấp. Trong hầu hết các mô hình học sâu hiện nay, thông thường giá trị của stride sẽ được đặt là 1, 2 hoặc lớn hơn. Đối với các bài toán làm việc với những dữ liệu chuỗi, chẳng hạn như tín hiệu âm thanh, thường thì ta sẽ chọn stride = 1 hoặc stride = 2 để mô hình có thể nắm bắt được các mối liên hệ về đặc trưng giữa các frame liên tiếp nhau của tín hiệu âm thanh.

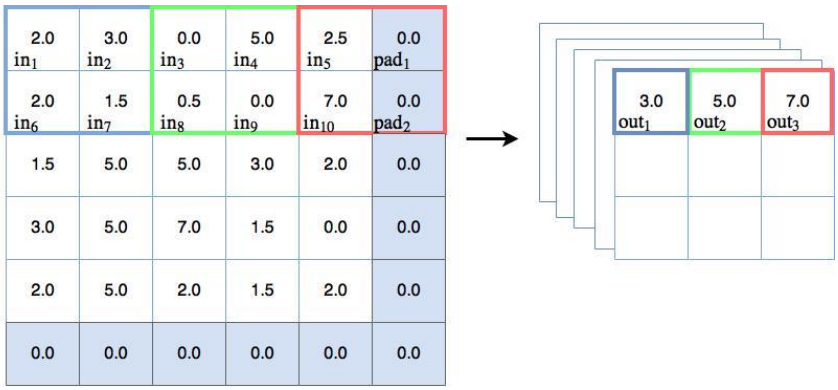


Hình 2.17: Stride trong CNN (nguồn: stackoverflow.com)

- **Pooling [26]**

Lớp pooling thường được sử dụng ngay sau lớp tích chập để đơn giản hóa thông tin đầu ra nhằm giảm bớt số lượng nơ-ron. Có 3 thủ tục pooling phổ biến là: Max Pooling, Average Pooling, Sum Pooling

- Max Pooling là thủ tục chọn giá trị lớn nhất trong vùng đầu vào của feature map
- Average Pooling là thủ tục tính giá trị trung bình trong vùng đầu vào của feature map
- Sum Pooling là thủ tục tính tổng của tất cả các giá trị trong vùng đầu vào của feature map



Hình 2.18: Pooling trong CNN (nguồn: adventuresinmachinelearning.com)

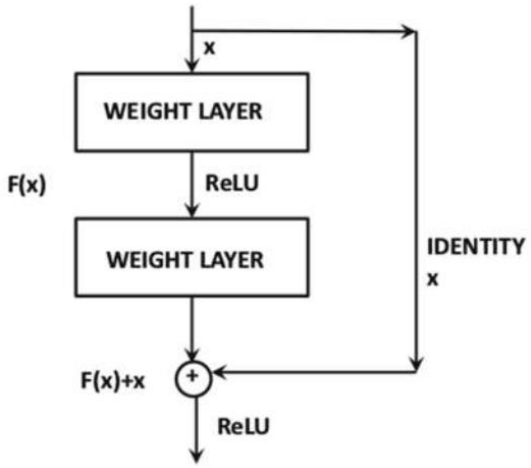
2.4.3 Tổng quan cấu trúc mạng RESNET

ResNet (Residual Network) [10] là một mạng CNN được giới thiệu đến công chúng vào năm 2015 bởi nhóm tác giả Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Sự ra đời của ResNet là một bước tiến vượt bậc trong lĩnh vực xử lý hình ảnh (Computer Vision). Nó đã giành được vị trí thứ nhất trong cuộc thi ILSVRC 2015 [10] với tỉ lệ lỗi top 5 chỉ 3.6%, nhận được sự quan

tâm của đông đảo các chuyên gia nghiên cứu về học máy, học sâu. Đã có rất nhiều paper nổi tiếng sử dụng các biến thể của kiến trúc mạng này. Một vài biến thể có thể kể đến của kiến trúc ResNet với số lớp khác nhau như: ResNet-18, ResNet-34, ResNet-50, ResNet-101, ResNet-152,... Các số đằng sau cụm từ ResNet nhằm mục đích chỉ kiến trúc của ResNet với một số lớp nhất định.

Trong quá trình thiết kế các mạng nơ ron học sâu nhiều lớp, một vấn đề thường xảy ra khi áp dụng thuật toán lan truyền ngược là hiện tượng mất mát đạo hàm (vanishing gradient) dẫn tới quá trình học tập không tốt. Giải pháp cơ bản cho vấn đề này là giảm bớt độ sâu của mạng. Tuy nhiên, trong rất nhiều bài toán xử lý với những dữ liệu có không gian đặc trưng cực kỳ lớn, ví dụ như ảnh của một con hổ có độ phân giải 4K, thì việc thiết kế một mạng nông sẽ không đem lại một hiệu suất khả thi. Mạng ResNet được ra đời nhằm giải quyết vấn đề đó.

Giải pháp mà ResNet đưa ra là sử dụng các kết nối "tắt" (skip connections) đồng nhất giữa các lớp nhằm mục đích cho phép việc sao chép giữa chúng, đồng thời đề xuất kiến trúc lặp lại đặc tính (so với kiến trúc kế thừa). Một khối như vậy được gọi là một Residual Block (khối dư), được mô tả trong hình dưới đây :



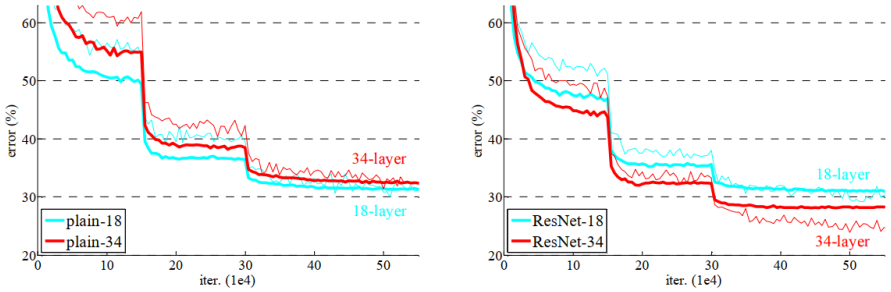
Hình 2.19: Kiến trúc cơ bản của một khối ResNet (nguồn: uet.vnu.edu.vn)

Hình 2.19 hiển thị cấu trúc chung của khối dư được sử dụng trong mạng RESNET. Điểm đặc biệt ở đây là có thêm một mũi tên cong xuất phát từ đầu và kết thúc tại cuối khối dư. Áp dụng thuật toán lan truyền xuôi (forward-propagation [27]) ta có:

$$out\_layer\_0 = weight\_1(x)$$
$$out\_layer\_1 = ReLU(out\_layer\_0)$$
$$F(x) = weight\_2(out\_layer\_1)$$
$$out\_layer\_3 = F(x) + x$$
$$out\_layer\_4 = ReLU(out\_layer\_3)$$

Một lợi ích nữa của các kết nối tắt này là chúng giúp cho kích cỡ không gian cũng như độ sâu của đầu vào không thay đổi từ lớp này cho đến lớp khác,

qua đó đã phần nào gỡ bỏ cản trở cho luồng gradient, giảm thiểu vấn đề mất mát đạo hàm.



Hình 2.20: Thực nghiệm tính hiệu quả của kiến trúc ResNet do nhóm tác giả thực hiện trên bộ dữ liệu ImageNet 2012 classification dataset [10]

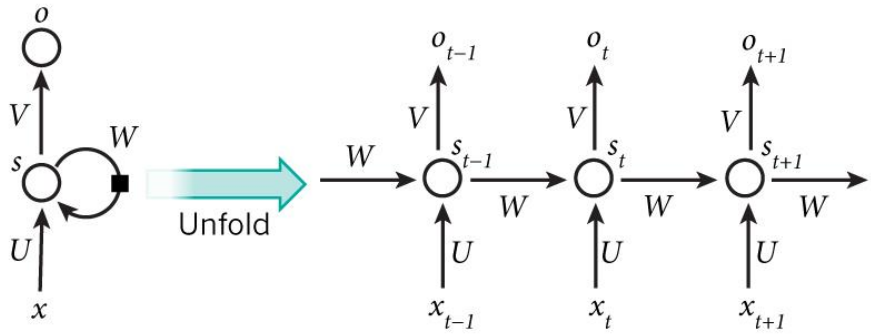
Sau ResNet, hàng loạt các biến thể khác của kiến trúc này được giới thiệu, có thể kể đến như: ResNeXt, DenseNet, Stochastic Depth, ... Thực nghiệm cho thấy những kiến trúc này có thể huấn luyện mạng nơ-ron với độ sâu hàng nghìn lớp mà vẫn mang lại kết quả chính xác cao. Đây chính là lý do đưa ResNet nhanh chóng trở thành kiến trúc mạng phổ biến nhất cho đến ngày nay trong lĩnh vực Computer Vision.

2.4.4 Mô hình mạng hồi quy

- **Mạng hồi quy RNN (Recurrent neural network) [19]**

Một Neural Network sẽ bao gồm 3 phần chính là Input layer, Hidden layer và Output layer, ta có thể thấy là đầu vào và đầu ra của mạng neuron này là độc lập với nhau. Như vậy mô hình này không phù hợp với những bài toán dạng chuỗi như mô tả, hoàn thành câu, ... vì những dự đoán tiếp theo như từ tiếp theo sẽ phụ thuộc vào vị trí của nó trong câu và những từ đằng trước nó.

Đây chính là lý do mà RNN ra đời với ý tưởng chính là sử dụng một bộ nhớ để lưu lại thông tin từ những bước tính toán xử lý trước để dựa vào đó nó có thể đưa ra dự đoán chính xác nhất cho bước dự đoán hiện tại.



Hình 2.21: Cấu trúc cơ bản của mạng RNN (nguồn: dominhhai.github.io)

Việc tính toán bên trong một nút mạng RNN sẽ được thực hiện như sau:

Bước 1: Với x_t là đầu vào tại bước t, s_t là trạng thái ẩn tại bước t (chính là bộ nhớ của mạng) sẽ được tính toán dựa trên cả các trạng thái ẩn phía trước và đầu vào tại bước đó:

$$s(t) = f(Ux_t + Ws_{t-1})$$

Bước 2: Tính toán đầu ra tại bước *t* theo công thức: $o_t = g(Vs_t)$

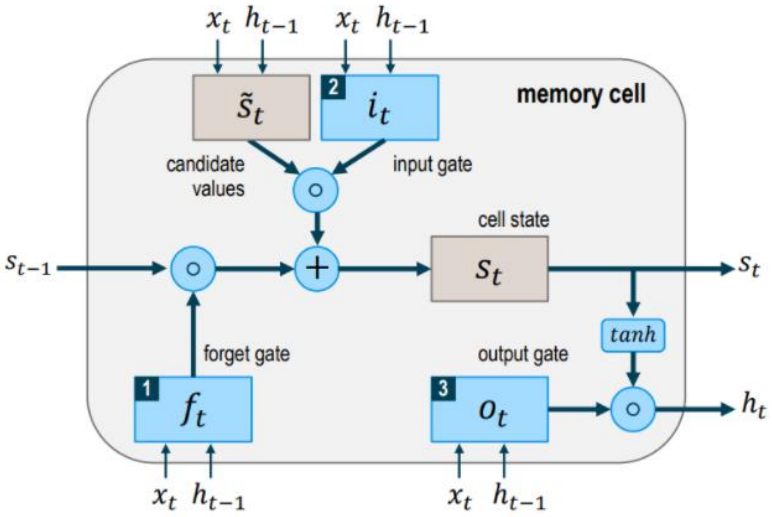
Trong đó: *W, U, V* là các ma trận trọng số, được học trong quá trình huấn luyện, *f* và *g* là các hàm kích hoạt (activation function).

Trên lý thuyết, RNN có khả năng ghi nhớ thông tin của tất cả vị trí phía trước vị trí hiện tại, nhưng khi áp dụng thuật toán lan truyền ngược (back-propagation [27]) thì RNN gặp phải hiện tượng mất mát đạo hàm (vanishing gradient), các state ở càng xa thì đạo hàm càng nhỏ (gần về 0), dẫn đến việc cập nhật trọng số cho mô hình sẽ không phụ thuộc nhiều vào các state này. Do đó, mô hình RNN chỉ có thể ghi nhớ tốt một vài trạng thái phía trước mà quên đi những đặc trưng quan trọng của các trạng thái ở xa.

- **Mạng hồi quy LSTM (Long short term memory) [15]**

LSTM là một phiên bản mở rộng của mạng RNN, được đề xuất vào năm 1997 bởi nhóm tác giả Sepp Hochreiter và Jürgen Schmidhuber. LSTM được thiết kế để giải quyết các bài toán về phụ thuộc xa (long-term dependencies), một vấn đề mà RNN không làm được do bị ảnh hưởng bởi vấn đề mất đạo hàm.

Kiến trúc bên trong một khối (tế bào) LSTM:



Hình 2.22: Cấu trúc bên trong một tế bào LSTM [15]

Quá trình xử lý thông tin trong một tế bào LSTM diễn ra như sau:

Bước 1: Xử lý các thông tin phía trước:

- Tế bào LSTM quyết định những thông tin nào là quan trọng từ nhân (cell internal state) tế bào LSTM phía trước đó, được định danh bởi s_{t-1}.
- Giá trị f_t của forget gate (cổng quên) tại bước thời gian *t* được tính dựa trên giá trị đầu ra từ tế bào LSTM ở bước trước đó h_{t-1}, giá trị đầu vào hiện tại của chuỗi đặc trưng x_t, bias b_f thêm vào của forget gate.

$$f_t = sigmoid(W_{f,x}x_t + W_{f,h}h_{t-1} + b_f)$$

Tế bào LSTM được coi là "hoàn toàn quên" các thông tin phía trước nếu $f_t = 0$

Tế bào LSTM được coi là "hoàn toàn ghi nhớ" các thông tin phía trước nếu $f_t = 1$

Bước 2: Xử lý thông tin trong tế bào hiện tại

Bước này sẽ thực hiện tính toán thông tin cho nhân tế bào LSTM hiện tại được định danh bởi s_t, gồm 3 phần chính:

- $\tilde{s}_t$ biểu diễn những thông tin tiềm năng (candidate value) được tính toán từ các LSTM trước và đầu vào hiện tại:

$$\tilde{s}_t = tanh(W_{\tilde{s},x} x_t + W_{\tilde{s},h} h_{t-1} + b_{\tilde{s}})$$

- i_t biểu diễn cho giá trị cổng vào (input gate) theo đó cũng được tính như sau:

$$i_t = tanh(W_{i,x} x_t + W_{i,h} h_{t-1} + b_i)$$

- s_t được tính dựa trên kết quả thu được từ các bước trước với phép nhân Hadamard theo từng phần tử (Hadamard product) được ký hiệu bằng o:

$$s_t = f_t \, o \, s_{t-1} + i_t \, o \, \tilde{s}_t$$

Bước 3: Tính toán giá trị đầu ra cho tế bào

- Giá trị đầu ra h_t của tế bào LSTM được tính toán dựa theo hai phương trình sau:

$$o_t = sigmoid(W_{o,x} x_t + W_{o,h} h_{t-1} + b_o)$$

$$h_t = o_t \, o \tanh(s_t)$$

Trong đó:

- x_t là vector đầu vào tại mỗi bước thời gian *t*.
- $W_{f,x}$, $W_{f,h}$, $W_{\tilde{s},x}$, $W_{\tilde{s},h}$, $W_{i,x}$, $W_{i,h}$, $W_{o,x}$, $W_{o,h}$ là các ma trận trọng số trong mỗi tế bào LSTM.
- b_f, $b_{\tilde{s}}$, b_i, b_o là các vector bias
- f_t, i_t, o_t lần lượt chứa các giá trị kích hoạt cho các cổng forget gate, input gate và output gate tương ứng.
- s_t, $\tilde{s}$ lần lượt là các vector đại diện cho cell internal state và candidate value
- h_t là giá trị đầu ra của tế bào LSTM.

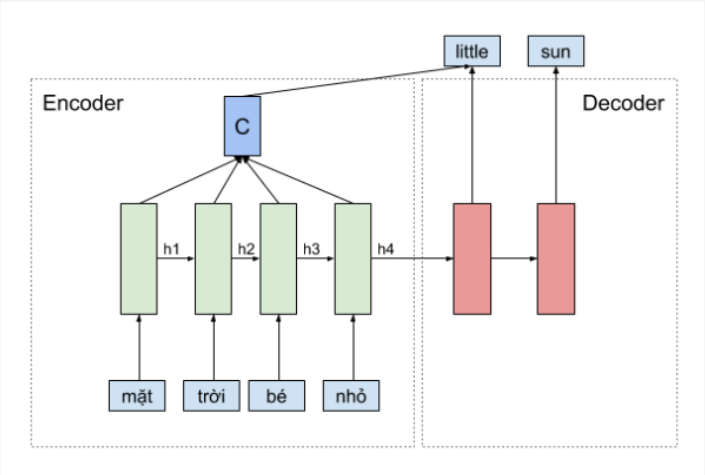
2.4.5 Cơ chế attention

Cơ chế attention [12] hiểu đơn giản là trung bình có trọng số của những "thứ" mà chúng ta nghĩ là nó cần thiết cho bài toán, điều đặc biệt là trọng số này do mô hình tự học được trong quá trình huấn luyện. Có khá nhiều loại cơ chế attention khác nhau, tuy nhiên tổng quan chúng ta có 2 loại:

- Hard Attention: Sử dụng học tăng cường để học vị trí cần phải chú ý
- Soft Attention: Học bộ trọng số bằng thuật toán lan truyền ngược

Lấy bài toán dịch máy làm ví dụ. Mô hình tổng quát cho bài toán này gồm 2 phần: Encoder và Decoder [28]. Khối Encoder làm nhiệm vụ biểu diễn thông

tin của một câu đầu vào thành biểu diễn của một vector duy nhất. Khối Decoder có nhiệm vụ phân giải những thông tin trong vector này để có thể dịch thành câu đích. Vấn đề nằm ở chỗ, những câu dài sẽ không được dịch chính xác vì thông tin không được lưu trữ đầy đủ trong một vector biểu diễn duy nhất.



Hình 2.23: Cấu trúc chung của bài toán dịch máy [12]

Cơ chế attention sinh ra để giải quyết vấn đề này, thay vì phải biểu diễn tất cả các từ "mặt", "trời", "bé", "nhỏ" bởi một vector biểu diễn duy nhất thì ta sẽ tập trung vào từng phần của câu. Ví dụ như: "mặt trời" và "bé nhỏ". Một lợi thế có thể thấy của cơ chế attention là nó cho phép mô hình có thể hiểu được những từ hay phần ảnh nào quyết định đến kết quả hiện tại.

Cơ chế attention cho bài toán dịch máy bao gồm 3 bước tính toán cơ bản như sau:

Bước 1: Tiến hành kết hợp từng vector đặc trưng đầu ra ở khối encoder (*h1, h2, h3, h4*) với vector đặc trưng đầu ra hiện tại của khối decoder (hình 2.24) thông qua một mô hình ***a*** nào đó (chẳng hạn như tầng fully connected).

$$e_{ij} = a(s_i, h_j)$$

Trong đó: a là mô hình học các hệ số attention , s_i là vector đặc trưng đầu ra hiện tại của khối decoder, e_{ij} là hệ số sau khi tổng hợp s_i với h_j

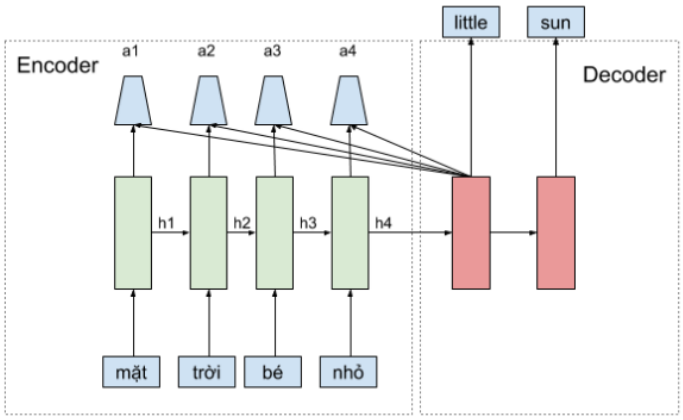
Bước 2: Chuẩn hoá các hệ số e này lại bằng cách sử dụng hàm softmax [29] ta thu được các hệ số có dạng:

$$\alpha_{ij} = sofmax(e_{ij})$$

Bước 3: Thực hiện tính toán vector đặc trưng bổ sung *c* chứa thông tin cho từ hiện tại ở khối decoder bằng cách tính trung bình có trọng số như sau:

$$c_i = sum_{j=1}^{T} \alpha_{ij} h_j$$

Trong đó: c_i là vector đặc trưng bổ sung cần tính toán cho từ hiện tại thứ *i* tại khối decoder, *T* là tổng số từ đầu vào trong khối encoder, α_{ij} là trọng số được học ở bước 2, được tính riêng cho từng từ tại khối decoder, còn h_j được định nghĩa là giá trị đầu ra hiện tại tại khối mạng thứ *j* tương ứng với từ đầu vào thứ *j*.



Hình 2.24: Minh hoạ cơ chế attention [12]

2.5 Phương pháp tính giá trị hàm mất mát

2.5.1 Hàm Mean Square Error (Bình phương lỗi trung bình)

Hàm mất mát MSE có công thức được định nghĩa như sau:

$$Loss(W; x_i; y_i) = \frac{1}{M}(\sum_{i=1}^{M}\left|\left|pred(x_i) - y_i\right|\right|^2)$$

Trong đó: $pred(x_i)$ và y_i lần lượt là giá trị nhãn dự đoán và giá trị nhãn thực sự của đầu vào x_i, M là số điểm dữ liệu của bài toán đang xét.

2.5.2 Hàm Cross entropy (Entropy chéo)

Công thức tính giá trị hàm mất mát giữa đầu ra dự đoán và đầu ra thực sự của một điểm dữ liệu x_i được tính như sau:

$$Loss(W; x_i; y_i) = \sum_{j=1}^{C} y_{ji} \log(p_{ji})$$

Trong đó: y_{ji} và p_{ji} lần lượt là giá trị của phần tử thứ j trong vector xác suất y_i và p_i, C là số lớp cần phân loại của bài toán, giá trị của p_i được tính thông qua ma trận trọng số W và đầu vào x_i.

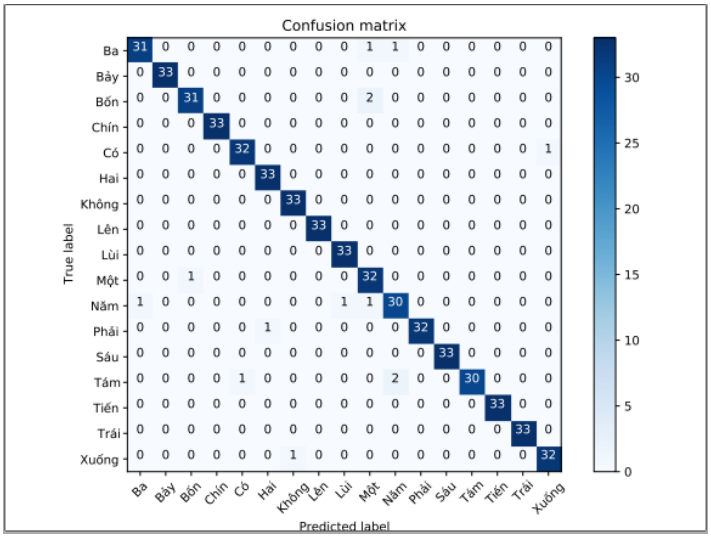
2.6 Phương pháp đánh giá mô hình phân loại

2.6.1 Accuracy (Độ chính xác)

$$\text{Độ chính xác} = \frac{\text{tổng số điểm được dự đoán đúng}}{\text{tổng số điểm trong tập dữ liệu đang xét}}$$

2.6.2 Confusion Matrix (Ma trận nhầm lẫn)

Cách tính sử dụng accuracy như ở trên chỉ cho chúng ta biết được bao nhiêu phần trăm lượng dữ liệu được phân loại đúng mà không chỉ ra được cụ thể mỗi class được phân loại như thế nào, lớp nào được phân loại đúng nhiều nhất, và dữ liệu thuộc lớp nào thường bị phân loại nhầm vào lớp khác. Để có thể đánh giá được các giá trị này, chúng ta sử dụng một ma trận được gọi là confusion matrix [6].



Hình 2.25: Minh hoạ ma trận nhầm lẫn

Với các bài toán có nhiều lớp dữ liệu, một mô hình tốt sẽ cho một confusion matrix có các phần tử trên đường chéo chính có giá trị lớn, các phần tử còn lại có giá trị nhỏ.

2.6.3 Precision và Recall

Với bài toán phân loại mà tập dữ liệu của các lớp là chênh lệch nhau rất nhiều, có một phép đo hiệu quả thường được sử dụng là Precision-Recall [6].

Precision (Độ chính xác): Được tính bằng tổng số mẫu dữ liệu thuộc lớp *c* được phân loại đúng chia cho tổng số mẫu dữ liệu được phân loại vào lớp *c*

$$precision = \frac{\text{tổng số mẫu thuộc lớp } c \text{ được phân loại đúng}}{\text{tổng số mẫu thuộc lớp } c \text{ được phân loại đúng} + \text{tổng số mẫu được phân loại là } c \text{ nhưng thực sự không thuộc lớp } c}$$

Recall (Độ triệu hồi): Được tính bằng tổng số mẫu dữ liệu thuộc lớp *c* được phân loại đúng chia cho tổng số mẫu dữ liệu thực sự thuộc vào lớp *c*

$$recall = \frac{\text{tổng số mẫu thuộc lớp } c \text{ được phân loại đúng}}{\text{tổng số mẫu thuộc lớp } c \text{ được phân loại đúng} + \text{tổng số mẫu được phân loại khác } c \text{ nhưng thực sự thuộc lớp } c}$$

Khi Precision = 1, mọi điểm tìm được đều thực sự thuộc lớp *c*, tức không có điểm khác lớp *c* nào lẫn vào kết quả. Tuy nhiên, Precision = 1 không đảm bảo mô hình là tốt, vì câu hỏi đặt ra là liệu mô hình đã tìm được tất cả các điểm thuộc lớp *c* hay chưa. Nếu một mô hình chỉ tìm được đúng một điểm thuộc lớp *c* mà nó chắc chắn nhất thì ta không thể gọi nó là một mô hình tốt.

Khi Recall = 1, mọi điểm thực sự thuộc lớp *c* đều được tìm thấy. Tuy nhiên, đại lượng này lại không đo liệu có bao nhiêu điểm không thuộc lớp *c* bị lẫn trong đó. Nếu mô hình phân loại mọi điểm là thuộc lớp *c* thì chắc chắn Recall = 1, tuy nhiên dễ nhận ra đây là một mô hình cực tồi.

Từ đó ta có thể thấy rằng một mô hình tốt là mô hình có cả Precision và Recall đều cao. Để kết hợp 2 tiêu chí này lại với nhau, người ta sử dụng thêm một đại lượng nữa là F1-SCORE [6]:

$$F_1 = \frac{2.Precision.Recall}{Precision + Recall} = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}}$$

Một mô hình được coi là tốt khi có F1 lớn, F1 có giá trị càng lớn khi Precision và Recall càng lớn.

Classes	Precision (%)	Recall (%)	F1-score (%)	Amount of Data
Class 1	0.96	1.00	0.98	65
Class 2	**1.00**	**1.00**	**1.00**	65
Class 3	**1.00**	**1.00**	**1.00**	65
Class 4	0.93	0.99	0.95	75
Class 5	1.00	0.93	0.97	15
Class 6	0.91	1.00	0.95	40
Class 7	0.99	0.97	0.98	70
Class 8	0.98	0.94	0.96	50
Class 9	0.98	0.98	0.98	60
Class 10	0.97	1.00	0.98	60
Class 11	**1.00**	**1.00**	**1.00**	70
Class 12	0.99	0.97	0.98	70
Class 13	1.00	0.99	0.99	70
Class 14	**1.00**	**1.00**	**1.00**	60
Class 15	0.98	1.00	0.99	60
Class 16	1.00	0.90	0.95	30
Class 17	0.98	0.89	0.93	45
Class 18	0.96	0.98	0.97	45
Class 19	0.93	1.00	0.96	25
Class 20	**1.00**	**1.00**	**1.00**	25
Class 21	0.98	1.00	0.99	65
Class 22	0.98	0.98	0.98	65
Class 23	0.98	0.91	0.94	65
Class 24	1.00	0.98	0.99	65

Hình 2.26: Phương pháp đánh giá Precision-Recall-F1 (nguồn: researchgate.net)

CHƯƠNG 3. PHƯƠNG PHÁP GIẢI QUYẾT BÀI TOÁN

Bài toán nhận diện phát âm từ khoá bản chất là bài toán phân loại các âm thanh từ khoá. Sở dĩ ta có thể sử dụng từ nhận diện ở đây là bởi vì sau một quá trình huấn luyện với nhiều đoạn âm thanh từ khoá của nhiều giọng đọc khác nhau, ta có thể nhận diện được âm thanh từ khoá của một người khác nói ra mà chưa hề hay biết đến giọng đọc của người đó.

Phương pháp chủ yếu được sử dụng để xây dựng mô hình nhận diện phát âm từ khoá là sử dụng các thuật toán học máy, học sâu. Ưu điểm của các thuật toán học máy cơ bản là phương pháp huấn luyện đơn giản và tốc độ xử lý (nhận diện) nhanh, tuy nhiên, thông thường các mô hình học máy đều vấp phải một nhược điểm đó là: độ chính xác thấp, các không gian đặc trưng biểu diễn cho mô hình đầu vào không bao quát được hết ý nghĩa của dữ liệu. Trong những năm gần đây, dưới sự phát triển vượt bậc của ngành trí tuệ nhân tạo, các thuật toán học sâu đã mang lại những kết quả vô cùng khả quan cho các bài toán nhận diện văn bản, nhận diện giọng nói. Tuy nhiên, các thuật toán học sâu cũng gặp phải một số vấn đề nổi trội như: Kiến trúc mạng phức tạp, thời gian huấn luyện và dự đoán chậm.

Để có thể xây dựng một mô hình nhận diện phát âm từ khoá trên các thiết bị tài nguyên thấp, cần phải giải quyết 3 vấn đề sau đây:

Thứ nhất, nguồn dữ liệu phải đủ lớn: Mỗi người có một âm sắc, giọng đọc khác nhau, cho nên yêu cầu có một nguồn dữ liệu âm thanh đủ lớn để mô hình có thể được tổng quát hoá và chính xác hơn.

Thứ hai, dữ liệu cho quá trình huấn luyện phải tương đối sạch: Việc xử lý các audio kém chất lượng, nhiều nhiễu đến nay vẫn còn là thách thức rất lớn đối với các nhà phát triển.

Thứ ba, mô hình huấn luyện phải đủ nhỏ, thời gian phản hồi nhanh: Để có thể chạy được trên các thiết bị tài nguyên thấp, cấu trúc của các mô hình không được quá phức tạp, không gian lưu trữ cho từng mô hình cũng phải nhỏ nhất có thể. Việc chạy một mô hình quá lớn trên các thiết bị cấu hình thấp tốn rất nhiều thời gian, hơn nữa sẽ không có tính ứng dụng thực tiễn.

Sau đây, chúng tôi xin trình bày các bước chính để giải quyết bài toán trên:

Bước 1: Thu thập, tiền xử lý dữ liệu: trong phạm vi của đồ án, nguồn dữ liệu âm thanh được thu thập chủ yếu từ các bạn sinh viên trường đại học Bách Khoa Hà Nội.

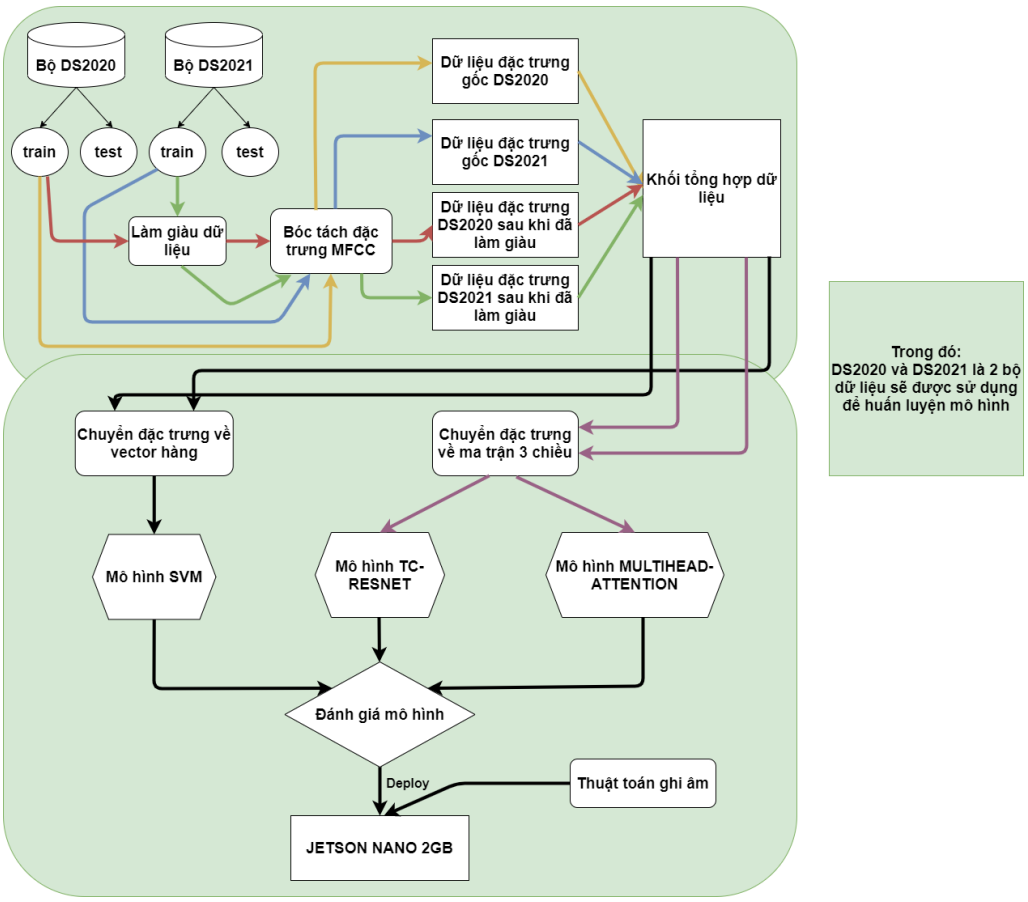
Bước 2: Làm giàu dữ liệu, bóc tách đặc trưng cho đầu vào của mô hình

Bước 3: Xây dựng mô hình phân loại dựa trên các mô hình SVM, TC-RESNET, MULTIHEAD-ATTENTION-RNN

Bước 4: Xây dựng thuật toán ghi âm audio theo luồng thời gian thực

Bước 5: Đưa mô hình lên thiết bị JETSON NANO 2GB

Dưới đây là tổng hợp quá trình xây dựng một mô hình nhận diện phát âm từ khoá:



Hình 3.1: Quá trình xây dựng mô hình nhận diện phát âm từ khoá

Chi tiết các bước giải quyết bài toán được trình bày cụ thể trong các phần dưới đây.

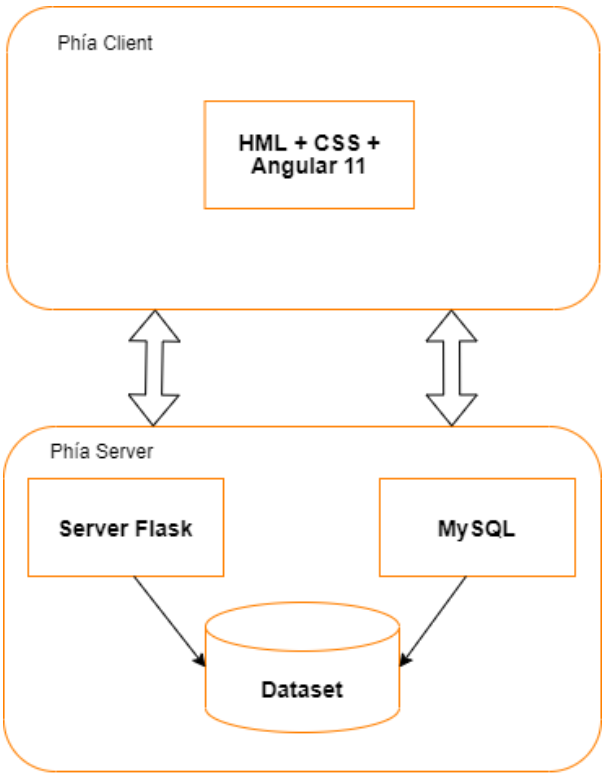
3.1 Thu thập, tiền xử lý dữ liệu

3.1.1 Phương pháp thu thập

Dữ liệu được thu thập thông qua một website thu âm được thiết kế bởi bạn Trần Minh Hiếu - sinh viên trường đại học Bách Khoa Hà Nội trong môn PROJECT 2. Kiến trúc website bao gồm 2 phần chính: Phần giao diện được viết bằng ngôn ngữ javascript sử dụng thư viện Angular 11, phần server backend được viết bằng ngôn ngữ python sử dụng thư viện Flask. Phía giao diện client kết nối với phía server thông qua các API HTTP, server sử dụng hệ quản trị cơ sở dữ liệu MySQL để lưu trữ thông tin các bản ghi âm, thông tin xác thực bản ghi âm cho người dùng.

Thời gian tối đa cho một lần ghi âm từ khoá là 2 giây, sau thời gian đó, hệ thống sẽ chuyển sang giao diện ghi âm cho một từ khoá khác. Mỗi file âm thanh ghi âm được có độ dài trung bình là 2 giây, tổng cộng sẽ có 17 nhãn cần phải thu âm: Một, Hai, Ba, Bốn, Năm, Sáu, Bảy, Tám, Chín, Không, Có, Tiến, Lùi, Lên, Xuống, Trái, Phải. Các nhãn từ khoá được lấy ý tưởng dựa trên các câu lệnh điều khiển cơ bản cho một thiết bị robot thông minh. Nguồn dữ liệu được thu thập chủ yếu từ các bạn sinh viên Bách Khoa Hà Nội dưới sự giúp đỡ của PGS.TS. Đỗ Phan Thuận và một số bạn bè, cộng đồng mạng trên facebook.

Hình vẽ dưới đây mô tả tổng quan kiến trúc hệ thống thu âm từ khoá:



Hình 3.2: Kiến trúc tổng thể của website thu âm

Trong quá trình thực nghiệm, chúng tôi xây dựng và đánh giá mô hình dựa trên 2 bộ dữ liệu sau:

Bộ DS2020: Bộ dữ liệu audio từ khoá đã được thu thập từ năm 2020 được cung cấp bởi PGS.TS. Đỗ Phan Thuận

Bộ DS2021: Bộ dữ liệu audio được thu thập tính đến hết tháng 4 năm 2021 + Bộ DS2020

Do phạm vi thu âm còn nhỏ, cũng như điều kiện thời gian không cho phép, cho nên số lượng file audio trong cả 2 bộ dữ liệu còn khá ít (khoảng 5000 file). Chính vì lẽ đó, để mô hình có thể học tập tốt hơn, thì cần thêm một số phương pháp làm giàu dữ liệu cũng như các kỹ thuật trích xuất đặc trưng hợp lý (sẽ nói rõ hơn ở phần sau).

3.1.2 Tiền xử lý dữ liệu

Trong quá trình ghi âm các audio âm thanh, có một vài yếu tố chủ quan cũng như khách quan mà người dùng gặp phải như: microphone không nhận, nơi thu âm quá nhiều tiếng ồn, hay đơn giản là chưa kịp ghi âm, dẫn tới một vấn đề là: có một số audio không có tiếng, một số audio thì có quá nhiều nhiễu. Bước tiền xử lý đóng góp một vai trò rất quan trọng cho quá trình huấn luyện sau này, một mô hình tốt phụ thuộc rất nhiều vào chất lượng của dữ liệu huấn luyện.

Để có thể giải quyết vấn đề này, chúng tôi tiến hành nghe lại tất cả các audio thu âm, trong quá trình nghe, chúng tôi sẽ loại bỏ đi các audio không có ý nghĩa theo các tiêu chí như sau:

- Có nhiều nhiễu và không thể nghe thấy âm thanh của từ khoá
- Không nghe thấy âm thanh gì trong file audio

- Có thể nghe thấy âm thanh tiếng người nhưng không rõ, không thể hiểu được ý nghĩa của từ khoá
- Một số audio khi nghe thì rất khó phân biệt giữa các nhãn của từ khoá cũng sẽ được loại bỏ (ví dụ từ "Bảy" nhưng khi phát âm thì lại đọc thành "Ba" + "y" trong đó âm "y" rất khó nghe)

Để có thể đánh giá mô hình phân loại trên nhiều miền dữ liệu khác nhau, trong quá trình lọc audio nhiễu, chúng tôi tiến hành đánh dấu lại những audio nào là giọng nữ, những audio nào là giọng nam, đánh dấu lại kiểu giọng của audio như: thuộc miền Bắc, miền Trung hay miền Nam.

3.2 Làm giàu dữ liệu, trích xuất đặc trưng cho đầu vào của mô hình

3.2.1 Các đặc trưng chung cơ bản của dữ liệu

Trong quá trình nghe lại các audio từ khoá, chúng tôi nhận thấy có một số đặc trưng chung cơ bản trong giọng đọc của các bạn sinh viên như sau:

Thứ nhất: Đa phần thì các audio có giọng đọc to rõ, không bị ngắt quãng, ngắt nhịp.

Thứ hai: Một số audio giọng đọc to rõ, nhưng có một số từ khoá chưa kịp đọc hết, mới chỉ đọc được một nửa từ, nhưng vẫn có thể hiểu được ý nghĩa từ khoá. Ví dụ như từ: "Tám", âm "Tờ" và âm "á" được đọc rất rõ nhưng âm "mờ" bị lướt đi rất nhanh.

Thứ ba: Có một sự khác biệt khá rõ ràng giữa cách phát âm của người miền Bắc với người miền Nam và người miền Trung. Ví dụ như từ "Bảy", người miền Bắc có xu hướng đọc là "Bẩy", nhưng người miền Nam lại có xu hướng đọc đúng âm tiết là "Bảy", người miền Trung thì đọc khá giống so với người miền Bắc.

Thứ tư: Có một số lượng nhỏ file audio có hiện tượng các âm bị kéo dài ra, chẳng hạn như từ "Tám", âm "a" bị nói kéo dài ra thêm một khoảng thời gian khiến cho cách phát âm gần giống như: nói âm "tá" xong rồi đến âm "ám".

Thứ năm: Một phát hiện tinh tế hơn là các âm đầu của một số từ như: Hai, Có, Không, Phải, gặp phải một số trường hợp là âm gió, tức là khi phát âm thanh quản sẽ không rung, dẫn đến khi phân tích đặc trưng âm thanh của các từ khoá này sẽ rất dễ bị nhầm lẫn với nhau.

Thứ sáu: Xét về mặt cảm nhận to nhỏ thì có một số file audio nói khá bé so với âm lượng trung bình của các file audio khác.

Thứ bảy: Trong những file audio chứa các âm thanh nhiễu từ môi trường bên ngoài (vẫn nghe thấy âm thanh từ khoá chính), có một vài âm thanh khác liên quan đến các âm của từ khoá cần nhận diện, đây là một trong những yếu tố lớn nhất ảnh hưởng xấu đến chất lượng dự đoán của mô hình nhận diện.

3.2.2 Phương pháp trích xuất đặc trưng

Như đã đề cập ở chương 2, phương pháp trích chọn đặc trưng MFCC có hiệu quả rõ rệt trong các bài toán xử lý, nhận diện tiếng nói. Nhận thấy được hiệu

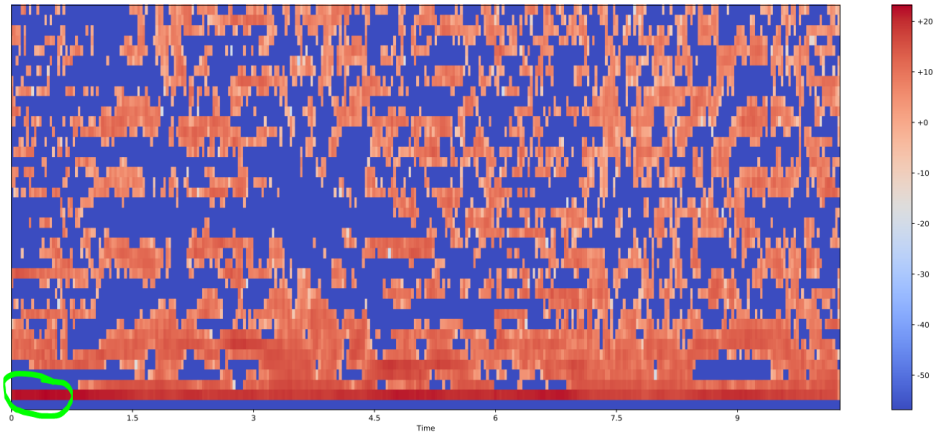
quả đó, việc lựa chọn MFCC làm phương pháp trích xuất đặc trưng cho đầu vào của mô hình nhận diện phát âm từ khoá là hoàn toàn khả thi.

Sau khi trải qua 5 bước cơ bản để trích xuất đặc trưng: Pre-emphasis, Spectrogram, Mel filterbank, Cepstrum và MFCC (đã nói ở chương 2), không gian đặc trưng MFCC được mô tả bởi một ma trận 2 chiều với số chiều được định nghĩa như sau:

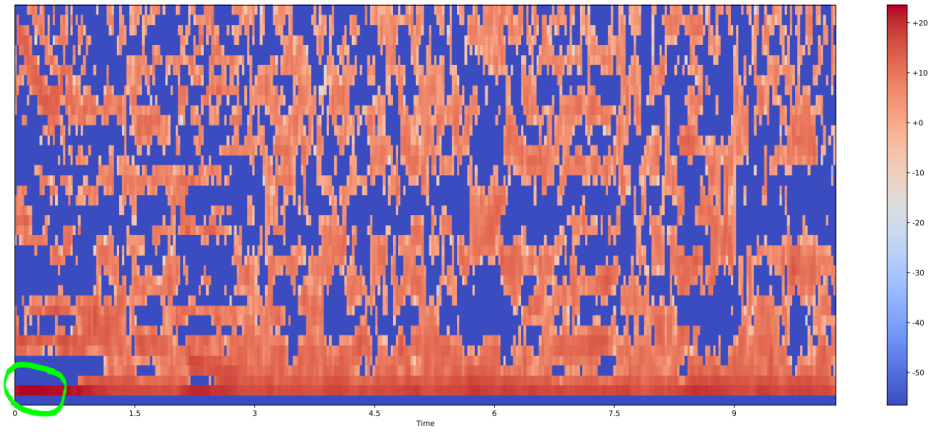
Chiều thứ nhất: tổng số hệ số MFCC mà ta muốn lấy (liên quan đến các tần số), thường được biểu diễn bởi tham số ***n_mfcc***

Chiều thứ hai: tổng số frame đang có trong tín hiệu audio đang xét

Dưới đây là minh hoạ kết quả trích xuất đặc trưng MFCC cho 2 file audio từ khoá "Ba" và từ khoá "Có" do cùng một người nói:



Hình 3.3: Đặc trưng MFCC cho từ khoá "ba"



Hình 3.4: Đặc trưng MFCC cho từ khoá "có"

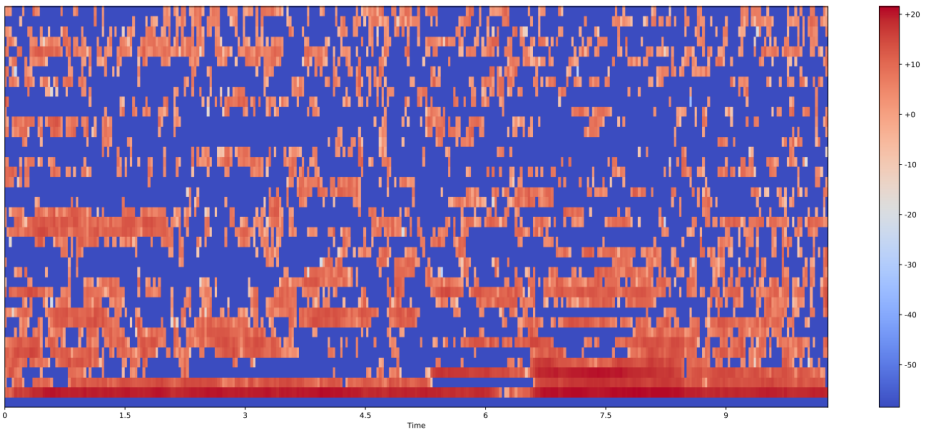
Trục x (Time) thể hiện thời gian tương ứng với các frame của audio từ khoá.

Trục y thể hiện cho các hệ số đặc trưng MFCC được trích xuất.

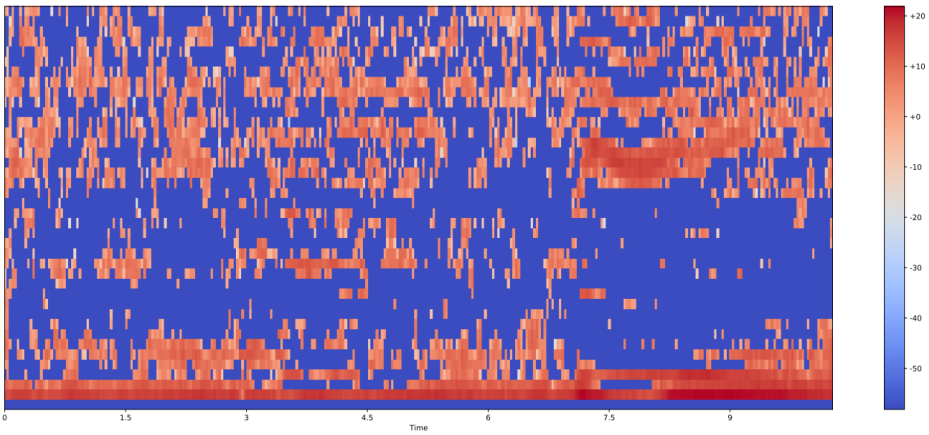
Thứ tự của các hệ số đặc trưng MFCC sẽ tuân theo thứ tự các tần số đặc trưng của audio từ khoá. Nhìn vào hình vẽ, ta có thể thấy rằng, có một số thời điểm tại các tần số xác định, thì chúng đều có mức năng lượng cao (phần khoanh tròn màu xanh lá). Âm "a" không mang dấu, xu hướng có mức năng lượng nhỏ

hơn âm "ó" mang dấu (dấu sắc). Do đó mà phân bố năng lượng của từ khoá "Có" (các điểm mang màu cam) có vẻ dày đặc và đậm hơn so với từ khoá "Ba".

Hai hình vẽ dưới đây sẽ mô tả thêm về sự khác nhau giữa đặc trưng MFCC của một giọng nam và một giọng nữ:



Hình 3.5: Đặc trưng MFCC cho từ khoá "lên" của một giọng nam



Hình 3.6: Đặc trưng MFCC cho từ khoá "lên" của một giọng nữ

Ta có thể thấy rằng, phân bổ mức năng lượng của giọng nữ tập chung ở các tần số cao, trong khi đó ở giọng nam thì lại tập chung ở các tần số thấp. Điều đó khẳng định đúng tính chất mà chúng ta đã nói ở chương 2: giọng nữ thường có xu hướng cao hơn giọng nam.

Do đặc trưng MFCC có dạng là một mảng 2 chiều (là một trong số các dạng của dữ liệu cấu trúc lưới), ta hoàn toàn có lý do để áp dụng nó trong các bài toán nhận tiếng nói thông qua sử dụng mô hình mạng tích chập CNN, một trong số đó là bài toán nhận diện phát âm từ khoá.

3.2.3 Phương pháp làm giàu dữ liệu

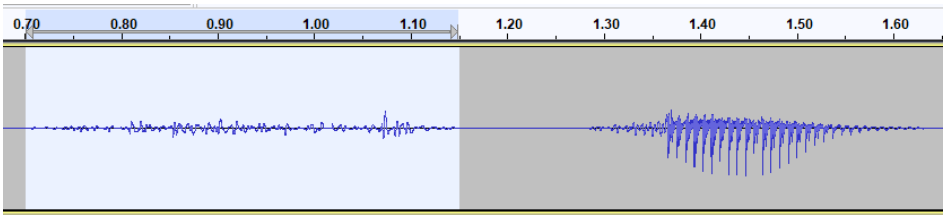
Qua các phân tích về dữ liệu cũng như đặc trưng ở trên, chúng tôi đề xuất một số phương pháp làm giàu dữ liệu như sau:

Phương pháp 1: Loại bỏ nhiễu

Trong các đoạn audio từ khoá, sẽ có những đoạn âm thanh không mong muốn khác như: tiếng xe cộ, tiếng microphone rè, tiếng đám đông nói chuyện,

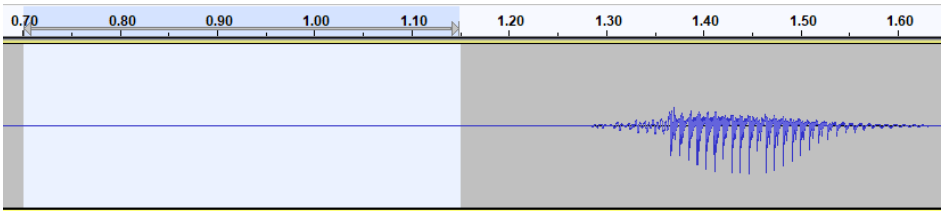
tiếng quạt máy, ... Các loại nhiễu này phân bổ theo 2 trường hợp chính: một là lẫn trực tiếp vào âm thanh của từ khoá gốc, hai là ở các đoạn ngoài rìa. Phương pháp để xử lý trường hợp thứ nhất là ta có thể dùng các bộ lọc nhiễu âm thanh, ví dụ như: denoiser của facebook. Tuy nhiên, phương pháp này vấp phải một số vấn đề như: thời gian xử lý chậm, có thể làm thay đổi đi đặc tính tự nhiên về âm sắc của giọng nói (chẳng hạn như giọng nghe sẽ bị đục hơn). Để xử lý trường hợp thứ hai, ta tiến hành đánh dấu và "làm im lặng" các đoạn âm thanh nhiễu bằng việc sử dụng phần mềm có sẵn bên thứ ba đó là **Audacity.**

Hình vẽ dưới đây mô tả cơ chế đánh dấu các đoạn âm thanh nhiễu:



Hình 3.7: Một đoạn âm thanh nhiễu (nền trắng xanh)

Kết quả sau khi "làm im lặng" đoạn âm thanh nhiễu này:



Hình 3.8: Đoạn âm thanh sau khi đã lọc bỏ nhiễu

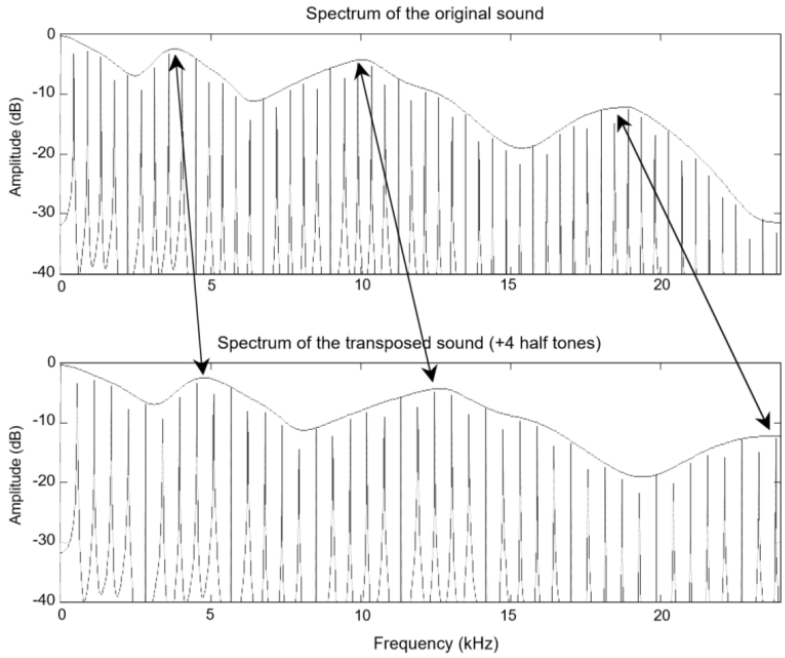
Như vậy, ứng với mỗi 1 file audio (chứa nhiễu) sẽ có 2 file augment audio: một file có nhiễu và một file được lọc bỏ nhiễu. Do hạn chế về mặt thời gian, hiện tại thì chúng tôi mới chỉ hoàn thành việc loại bỏ nhiễu cho bộ dữ liệu DS2020.

Phương pháp 2: Nâng, giảm cao độ của audio

Một vấn đề đặt ra trong các bộ dữ liệu đó là tỷ lệ giữa số lượng audio giọng nam với số lượng audio giọng nữ chênh nhau khá nhiều. Khoảng 86% cho nam và 14% cho nữ. Hơn nữa, một đặc tính quan trọng mà ta đã nêu ở chương 2 là: giọng nữ thường cao hơn giọng nam, mà yếu tố để làm lên cái được gọi là "cao" hơn đó chính là cao độ (hay tần số) của âm thanh. Do giọng nữ bản chất về cao độ đã cao rồi cho nên phương pháp này chúng tôi chỉ áp dụng với các audio giọng nam.

Cơ chế của phương pháp này là sẽ nâng hoặc giảm các tần số hiện có trong audio hiện tại theo các mức định sẵn. Có hai tham số cơ bản cần chú ý khi thực hiện nâng hoặc giảm cao độ của âm thanh đó là: số bước nâng (***n_step***) và số bước (số cung được xét) trong một quãng nhạc (***bin per octave***). Thông thường một quãng nhạc sẽ bao gồm 12 nốt nhạc cơ bản: đồ, đồ thăng, rê, rê thăng, ... , khi đó mỗi một bước nâng hoặc giảm cao độ sẽ tương ứng là một nửa cung (khoảng cách giữa hai nốt nhạc liền kề). Người ta lấy nốt "la" trong quãng nhạc thứ 4 tính từ 0 (được ký hiệu là "A4") làm gốc, có tần số là 440Hz.

Hình vẽ dưới đây mô tả ví dụ về sự thay đổi của tín hiệu âm thanh trước và sau khi nâng cao độ:



Hình 3.9: Sự thay đổi tín hiệu âm thanh trước và sau khi nâng cao độ [30]

Với mỗi một file audio từ khoá giọng nam, chúng tôi sẽ tiến hành nâng và giảm cao độ theo các thông số cụ thể (sẽ nói ở phần thực nghiệm). Phương pháp này mong muốn mô hình có thể nhận diện giọng nói của các bạn nữ sẽ được tốt hơn trong hoàn cảnh dữ liệu đang bị thiếu hụt về giọng nữ.

Phương pháp 3: Tua nhanh thời gian cho audio

Tốc độ đọc của mỗi người là khác nhau, tuy nhiên có một số nhỏ đọc khá nhanh, cho nên muốn mô hình có thể nhận diện tốt các trường hợp này, phương pháp được đề xuất là tua nhanh thời gian cho audio. Phương pháp này sẽ làm cho các âm trong từ khoá được nói nhanh hơn, cơ sở của nó bao gồm các bước chính sau đây:

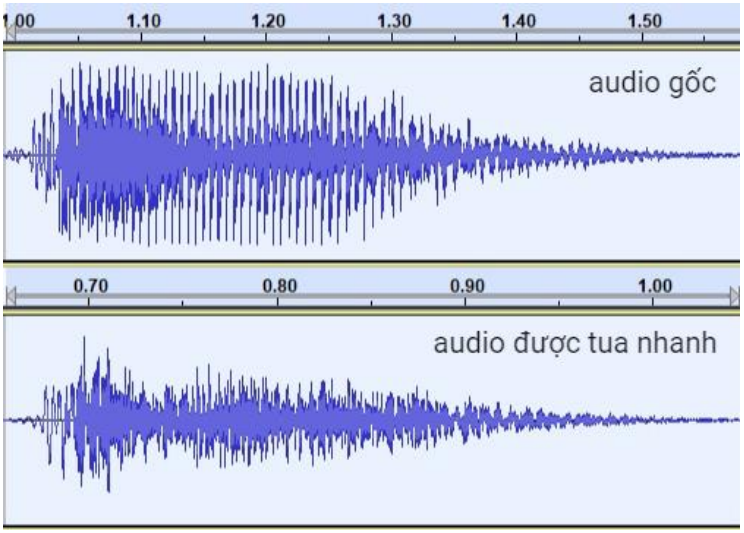
Bước 1: Sử dụng phép biến đổi fourier đưa miền tín hiệu theo thời gian về miền tần số, ta thu được một ma trận *f*. Ma trận này có chiều thứ nhất biểu thị cho số frame trong audio, chiều thứ hai biểu thị cho số tần số đại diện cho tín hiệu ban đầu.

Bước 2: Tiến hành thay đổi bước thời gian mới (tính bằng số frame) thông qua một hệ số factor: *factor < 1* thì bước thời gian rộng hơn, *factor > 1* thì bước thời gian hẹp hơn. Ta thu được số frame mới tương ứng với bước thời gian thay đổi.

Bước 3: Thực hiện thay đổi số điểm mẫu trong một frame, giảm (nếu muốn tua nhanh) hoặc tăng (nếu muốn dãn âm thanh) một cơ số lần.

Bước 4: Tính toán lại các pha ban đầu cho từng frame, sau đó xây dựng một ma trận fourier mới *f'* có chiều thứ nhất là số frame mới, chiều thứ hai là số tần số đại diện cho tín hiệu ban đầu.

Bước 5: Thực hiện biến đổi fourier ngược cho ma trận *f'*, ta thu được tín hiệu audio đã được tua nhanh về thời gian.



Hình 3.10: Sự khác nhau về tín hiệu giữa audio gốc với audio được tua nhanh về thời gian

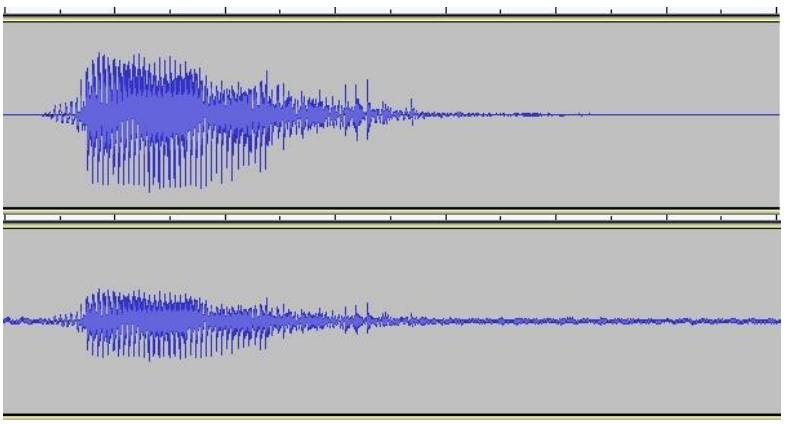
Phương pháp 4: Trộn các loại âm thanh nhiễu vào audio gốc

Trong quá trình ghi âm, khó tránh khỏi các yếu tố nhiễu bên ngoài, vì vậy cần có một cơ chế nào đó để mô phỏng lại các yếu tố này trên các dữ liệu âm thanh từ khoá. Bộ dữ liệu audio nhiễu được sử dụng là bộ dữ liệu của google. Trong bộ dữ liệu này họ cung cấp một số file audio nhiễu theo các khía cạnh khác nhau như: nhiễu trắng (white noise), nhiễu hồng (pink noise), tiếng câu cá, tiếng xe đạp đi, tiếng mèo kêu, ... Tiến hành trộn các audio nhiễu với các audio gốc theo cơ chế như sau:

Đầu tiên, chúng tôi tiến hành tạo một mẫu tín hiệu nhiễu được bắt đầu tại một thời điểm bất kỳ trong audio gốc. Tiếp theo, chúng tôi thực hiện tính tổng biên độ giữa tín hiệu gốc với tín hiệu nhiễu theo một hệ số tỷ lệ γ (tuân theo phân phối chuẩn) theo công thức như sau:

$$tin\_hieu\_audio\_moi = tin\_hieu\_nhieu * \gamma + (1 - \gamma) * tin\_hieu\_audio\_ban\_dau$$

Hình vẽ dưới đây mô tả sự thay đổi của một tín hiệu âm thanh trước và sau khi trộn thêm âm thanh nhiễu:



Hình 3.11: Sự khác nhau giữa tín hiệu âm thanh trước và sau khi trộn nhiễu

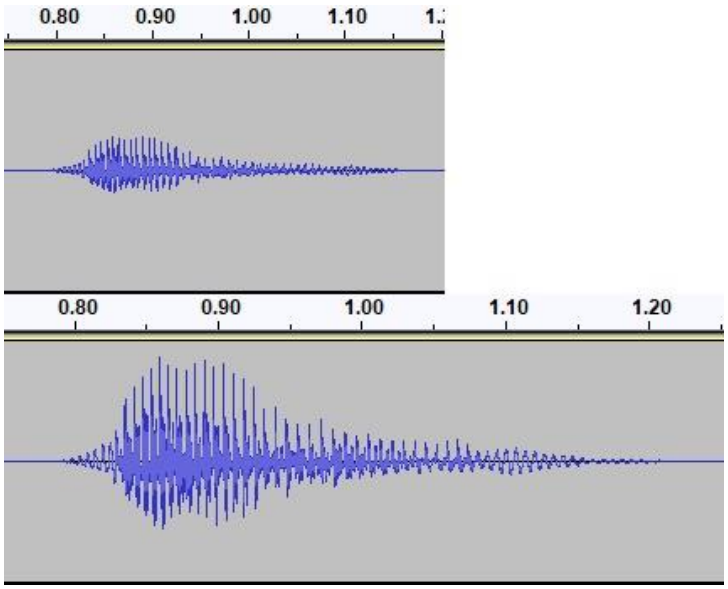
Phương pháp 5: Chuẩn hoá âm lượng cho dữ liệu audio

Có rất nhiều nguyên nhân chủ quan cũng như khách quan khiến cho âm lượng của audio bị nhỏ. Để giải quyết vấn đề này, có hai hướng được đề xuất như sau:

Một là chúng tôi sẽ tiến hành chuẩn hoá âm lượng cho tất cả các audio về một mức nhất định mà tai người có thể nghe được.

Hai là chúng tôi sẽ giảm và tăng âm lượng cho từng file với mục đích mô hình sau này có thể học tập được những audio nói nhỏ mà không cần thêm một bước chuẩn hoá âm lượng.

Hạn chế của phương pháp thứ 2 là đối với những audio đã nói to, khi ta nâng âm lượng (biên độ) lên thì xác suất xảy ra khả năng vượt qua ngưỡng vỡ tiếng là rất cao (ngưỡng vỡ tiếng là ngưỡng mà khi biên độ audio vượt quá thì âm sắc của audio sẽ bị thay đổi - nghe sẽ bị rè hơn).



Hình 3.12: Minh hoạ việc tăng âm lượng cho audio

3.3 Xây dựng mô hình phân loại

Trải qua các bước thu thập, làm giàu dữ liệu, bóc tách đặc trưng, ta cần thêm một bước nữa là lựa chọn một mô hình học máy, học sâu phù hợp. Để xây dựng mô hình nhận diện phát âm từ khoá cũng như để tiện so sánh về tính hiệu quả giữa các mô hình với nhau, chúng tôi đề xuất sử dụng 3 mô hình sau:

Mô hình SVM: Đại diện cho các mô hình học máy thống kê cơ bản.

Mô hình TC-RESNET: Đại diện cho các mô hình chỉ sử dụng mạng tích chập CNN.

Mô hình MULTIHEAD-ATTENTION-RNN: Một mô hình đại diện cho các mô hình mạng nơ ron hồi quy có kết hợp thêm cơ chế attention.

3.3.1 Phương pháp chia tập train, test

Do phân bố dữ liệu hiện có chủ yếu thuộc giọng nam miền Bắc, chúng tôi sẽ thực hiện chia tập train, test cho 2 bộ dữ liệu DS2020 và DS2021 theo các cách thức như sau:

Bộ DS2020 sẽ thực hiện chỉ lấy 145 file test (tỷ lệ train : test = 95 : 5) trong toàn bộ tập dữ liệu. Phân bố về dữ liệu vùng miền trong tập 145 file này sẽ tương đối đều nhau (tức là ứng với mỗi nhãn, sẽ có cả giọng nam, giọng nữ, giọng miền Bắc, miền Nam, miền Trung).

Bộ DS2021 sẽ thực hiện chia tập train, test theo 2 tỷ lệ là 80:20 và 90:10, trong đó phân phối dữ liệu của tập test được chọn theo hai phương án: Chỉ bao gồm audio giọng nam miền Bắc hoặc chỉ bao gồm audio giọng nam, nữ miền Bắc.

Với cùng một tập test chỉ bao gồm giọng nam miền Bắc đã chia cho bộ dữ liệu DS2021, tiến hành chỉ lấy một nửa dữ liệu trong tập train để huấn luyện cho mô hình (Mục đích của việc này nhằm chứng minh sự cần thiết của dữ liệu cho mô hình nhận diện phát âm từ khoá).

Tóm lại, dựa vào phương pháp chia tập train, test như trên, chúng tôi sẽ có tổng cộng 7 bộ dữ liệu được định nghĩa cụ thể như sau:

Bộ DS1: Chia train, test theo tỷ lệ 95:5 trên bộ dữ liệu DS2020, trong đó, 145 file test đủ phân bố về vùng miền, giới tính

Bộ DS2: Chia train, test theo tỷ lệ 90:10 trên bộ dữ liệu DS2021, tập test chỉ bao gồm giọng nam miền Bắc

Bộ DS3: Chia train, test theo tỷ lệ 80:20 trên bộ dữ liệu DS2021, tập test chỉ bao gồm giọng nam miền Bắc

Bộ DS4: Chia train, test theo tỷ lệ 90:10 trên bộ dữ liệu DS2021, tập test bao gồm giọng nam, nữ miền Bắc

Bộ DS5: Chia train, test theo tỷ lệ 80:20 trên bộ dữ liệu DS2021, tập test bao gồm giọng nam, nữ miền Bắc

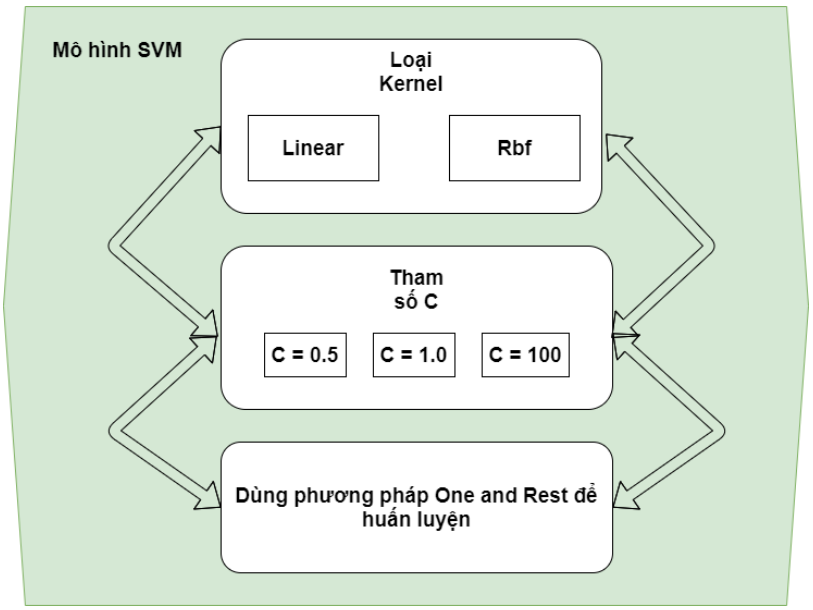
Bộ DS6: Chia train, test theo tỷ lệ 90:10 trên bộ dữ liệu DS2021, tập test chỉ bao gồm giọng nam miền Bắc, nhưng khi huấn luyện chỉ dùng một nửa dữ liệu trong tập train

Bộ DS7: Chia train, test theo tỷ lệ 80:20 trên bộ dữ liệu DS2021, tập test chỉ bao gồm giọng nam miền Bắc, nhưng khi huấn luyện chỉ dùng một nửa dữ liệu trong tập train

3.3.2 Mô hình SVM

Ý tưởng cơ bản đầu tiên để giải quyết bài toán này là sử dụng một mô hình học máy thống kê. Có rất nhiều thuật toán nổi tiếng như: Logistic Regression, SVM, Random Forest, ... Trong phạm vi của đồ án, chúng tôi sẽ sử dụng thuật toán SVM [18].

SVM thường hoạt động với những dữ liệu mà đặc trưng của nó được biểu diễn dưới dạng một vector hàng, trong đó mỗi phần tử trong vector này thể hiện một đặc tính nào đó của dữ liệu. Để có thể sử dụng đặc trưng MFCC cho mô hình này, ta cần một bước chuyển dữ liệu từ dạng ma trận 2 chiều về một vector hàng. Các thông số cũng như phương pháp huấn luyện thường được lựa chọn khi xây dựng mô hình SVM được mô tả chi tiết bởi sơ đồ dưới đây:



Hình 3.13: Phương pháp xây dựng mô hình SVM

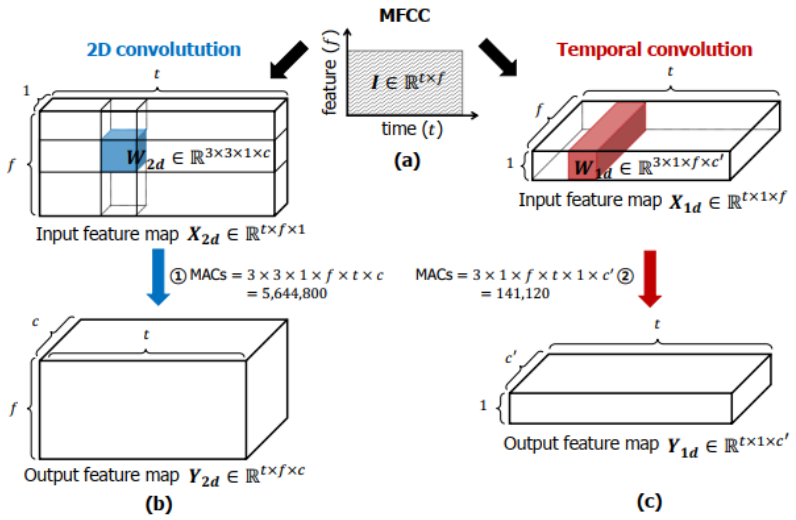
Ưu điểm của phương pháp này là thời gian dự đoán nhanh, mô hình dễ huấn luyện, tuy nhiên nhược điểm chính là độ chính xác thấp, thời gian huấn luyện khá lâu, không gian đặc trưng không thể hiện được mối liên hệ giữa các thành phần với nhau.

3.3.3 Mô hình TC-RESNET

Hạn chế của những dữ liệu có cấu trúc dạng lưới là khối lượng tính toán lớn, dẫn đến một vấn đề là khi đưa mô hình lên các thiết bị tài nguyên thấp thì không đáp ứng được nhu cầu về thời gian tính toán cho một giao dịch dự đoán.

Các mô hình mạng tích chập CNN thường làm việc với dữ liệu không gian 3 chiều. Một bức ảnh khi được đưa vào CNN để xử lý thường có 3 chiều cơ bản: chiều dài, chiều rộng và chiều sâu. Như đã phân tích ở phần 3.2, một đặc trưng MFCC là một mảng 2 chiều, để nó có thể hoạt động trên CNN, đơn giản ta thêm một chiều thứ 3 cho nó và mặc định là bằng 1. Tuy nhiên, các nghiên cứu gần đây cho thấy, với cách biểu diễn đặc trưng đầu vào như vậy, yêu cầu một khối lượng tính toán khá lớn và không khả thi khi chạy trên các thiết bị tài nguyên thấp. Nhóm tác giả Hàn Quốc Seungwoo Choi , Seokjun Seo , Beomjun Shin , Hyeongmin Byun [17] đã đề xuất một phương pháp biến đổi chiều của đặc trưng đầu vào hiệu quả hơn (sử dụng kiến trúc CNN-1D thay cho kiến trúc CNN-2D), sau đó là kết hợp với kiến trúc mạng của RESNET (đã giới thiệu ở chương 2) tạo ra một mô hình vừa có độ chính xác cao, vừa có thời gian xử lý nhanh cho một giao dịch dự đoán. Mô hình này được nhóm tác giả đặt tên là TC-RESNET (Temporal Convolution Resnet).

Hình 3.13 mô tả sự khác nhau về khối lượng tính toán giữa 2 mô hình CNN-2D và CNN-1D:



Hình 3.14: Sự khác nhau về khối lượng tính toán giữa 2 mô hình CNN-2D và CNN-1D [17]

Như ta đã biết, CNN hoạt động với các bộ filter (hay còn gọi là kernel) để thực hiện phép tích chập cho từng phần cục bộ của dữ liệu dạng lưới. Phần lớn kích thước của các bộ filter này là nhỏ nên tại một thời điểm, khó có thể trích xuất đầy đủ các đặc tính quan trọng cả về tần số thấp cũng như tần số cao trong đặc trưng MFCC (Ở hình 3.14b, khối lục phương màu xanh chỉ bao phủ một phần giới hạn các đặc trưng về tần số của MFCC). Chúng ta có thể giải quyết vấn đề này bằng cách tăng kích thước stride, áp dụng pooling nhiều lớp, sử dụng cơ chế attention để lưu các thông tin quan trọng, tuy nhiên, điều này gặp phải một vấn đề là kiến trúc mạng sẽ rất phức tạp, khối lượng tính toán lớn, khiến cho mô hình khó có thể triển khai trên nhiều thiết bị, đặc biệt là trên thiết bị tài nguyên thấp.

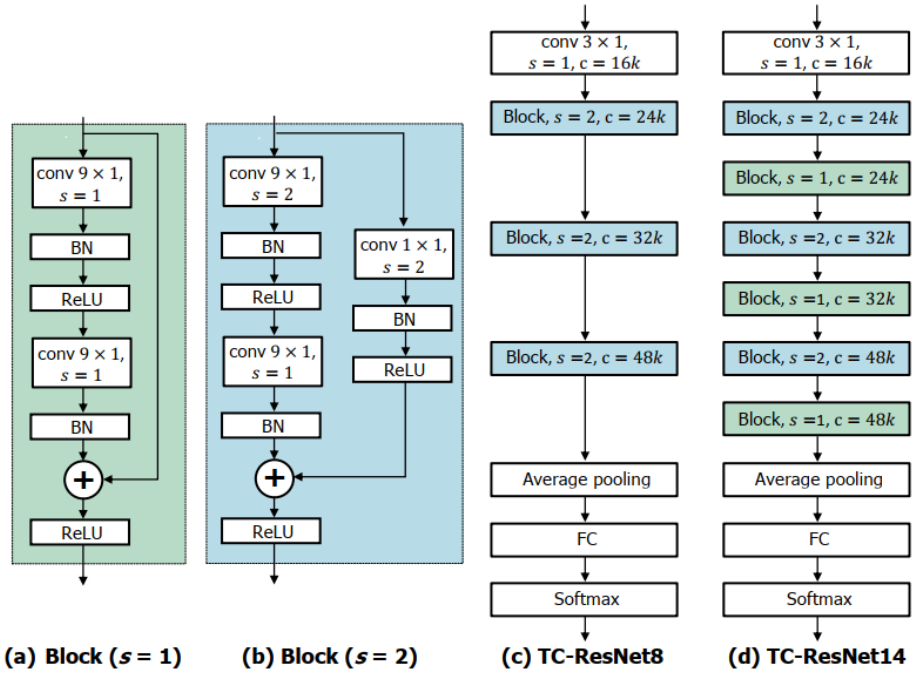
Xuất phát từ ý tưởng làm sao tại một thời điểm, mô hình có thể bao phủ trọn tất cả các đặc trưng về tần số, nhóm tác giả đề xuất chuyển chiều của đặc trưng đầu vào theo cơ chế: coi đặc trưng trên mỗi frame của MFCC như là một đặc trưng theo chuỗi thời gian. Nói cách khác, thay vì sử dụng đặc trưng đầu vào có dạng $X_{2d} \in R^{t\,x\,f\,x\,1}$, ta sử dụng đầu vào có dạng $X_{1d} \in R^{t\,x\,1\,x\,f}$(hình 3.14c).

Sau đây là một số lợi ích của phương pháp được đề xuất:

- Trong phương pháp đề xuất, tất cả các đặc trưng về tần số ở mức thấp cũng như ở mức cao đều được tính toán trong các lớp layer tiếp theo của mô hình CNN (hình 3.14c, khối tính toán màu đỏ bao gồm toàn bộ dải tần số của đặc trưng MFCC). Do đó mà mô hình có thể tận dụng triệt để các thông tin quan trọng tại các lớp phía trước. Điều này cho phép mô hình có thể đạt được hiệu suất tốt hơn ngay cả khi sử dụng một số lượng nhỏ các lớp.
- Giả sử với phương pháp biểu diễn đặc trưng đầu vào cũ, ta sử dụng *c* bộ lọc *kernel 3 x 3 x 1*, với phương pháp biểu diễn đặc trưng đề xuất, ta sử dụng *c'* bộ lọc *kernel 3 x 1 x f*, trong đó $c' = \frac{3\,x\,c}{f}$. Kết quả cho

thấy, phương pháp đề xuất có khối lượng tính toán nhỏ hơn so với phương pháp cũ (*141.120 phép tính* so với hơn *5 triệu phép tính* trong một bước truyền xuôi, ở đây giả sử các thông số của mô hình được cho trước với $t = 98, f = 40, c = 160, c' = 12$). Hơn nữa, giá trị đầu ra $Y_{1d} \in R^{t\ x\ 1\ x\ c'}$ của phương pháp mới có số chiều cũng nhỏ hơn so với $Y_{2d} \in R^{t\ x\ f\ x\ c}$ - giá trị đầu ra của phương pháp cũ.

Kiến trúc các lớp của mô hình TC-RESNET được mô tả chi tiết bằng hình vẽ dưới đây:



Hình 3.15: Kiến trúc các lớp của mô hình TC-RESNET [17]

Mô hình TC-RESNET kế thừa lại cơ chế hoạt động của cấu trúc mạng RESNET (đã giới thiệu ở chương 2). Mô hình sử dụng các *kernel m x 1* (*m = 3* cho lớp đầu tiên và *m = 9* cho các lớp còn lại). Lớp tích chập đầu tiên sử dụng *stride = 1, 16 x k (16k)* lớp *kernel 3 x 1* (tham số *k* ở đây dùng để điều chỉnh số lượng *kernel* trong một lớp tích chập). Cấu trúc các khối mạng riêng (Residual Block) sử dụng cơ chế kết nối tắt được mô tả chi tiết trong hình 3.15a và 3.15b. Hình 3.15c và 3.15d mô tả 2 cấu trúc mạng TC-RESNET8 và TC-RESNET14. TC-RESNET8 sử dụng 3 block sau lớp tích chập đầu tiên, trong khi đó thì TC-RESNET14 sử dụng tới 6 block.

Số lượng tham số cần học tập của mô hình TC-RESNET8 rơi vào khoảng *191 nghìn*, của mô hình TC-RESNET14 vào khoảng *350 nghìn* - ít hơn khá nhiều so với các mô hình học sâu phổ biến hiện tại, dẫn đến một hệ quả là thời gian dự đoán cho một input đầu vào sẽ nhanh hơn, hiệu quả hơn.

Thêm một lợi thế của TC-RESNET đó là nó tận dụng được khả năng tính toán song song của GPU do kiến trúc mô hình chỉ sử dụng các lớp tích chập CNN, rất thích hợp với những thiết bị sử dụng GPU có dung lượng bộ nhớ thấp như JETSON NANO 2GB.

Trong phạm vi của đồ án, chúng tôi sẽ tập chung tối ưu kết quả bài toán cũng như đưa ứng dụng mô hình lên thiết bị dựa trên kiến trúc TC-RESNET này.

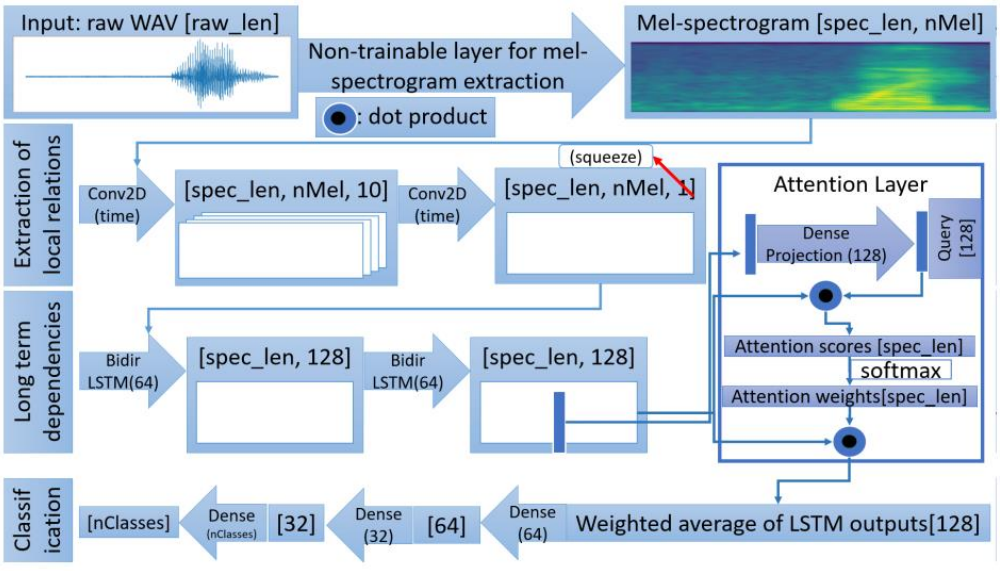
3.3.4 Mô hình MULTIHEAD-ATTENTION-RNN

Để so sánh tính hiệu quả về thời gian dự đoán giữa các mô hình với nhau, chúng tôi sẽ trình bày thêm một giải pháp nữa cho bài toán nhận diện phát âm từ khoá là sử dụng mô hình kết hợp giữa mô hình hồi quy với cơ chế attention.

Như đã giới thiệu ở chương 2, mô hình mạng hồi quy làm việc hiệu quả với các dữ liệu dạng chuỗi, mà audio là dữ liệu dạng chuỗi theo thời gian. Hơn nữa, các đặc trưng hữu ích của những audio này là không nhiều, tức là những đặc trưng quan trọng có thể phân bố ở bất kỳ đâu trong miền thời gian, lúc này ta nghĩ đến cơ chế attention. Đặc điểm nổi bật của cơ chế này là "trung bình có trọng số những thứ mà mô hình cho là cần thiết". Chính vì vậy, nhóm tác giả của google [31] đã đề xuất một mô hình kết hợp giữa các mạng hồi quy với cơ chế attention, mô hình đứng vị trí thứ hai trong dự án nhận diện phát âm từ khoá cho bộ dữ liệu "Google command dataset" với độ chính xác trên tập test khoảng 98%.

Thay vì sử dụng đặc trưng MFCC, mô hình này chỉ sử dụng mel-scale spectrogram làm đặc trưng đầu vào cho mô hình. Một tập hợp các lớp tích chập CNN-2D được tính toán dựa trên đặc trưng này nhằm mục đích tính toán các mối quan hệ địa phương giữa các dải tần số gần nhau. Các đầu ra ngay sau đó được đưa vào hai khối LSTM 2 chiều (Bidirectional long short term memory) để có thể thu thập được thông tin về cả 2 chiều của tín hiệu âm thanh. Tại thời điểm này, một trong số các vector đầu ra của khối LSTM cuối cùng (thông thường sẽ là vector ở giữa) sẽ được đưa vào lớp attention, sử dụng một vector query để đánh giá xem phần nào của audio là quan trọng nhất. Việc này sẽ được thực hiện lại *n* lần với các vector đầu ra khác nhau của khối LSTM (*n* là số cho trước, đây chính là lý do có chữ Multihead ở đầu tên của mô hình). Cuối cùng, các vector đầu ra sẽ được đưa qua 3 lớp kết nối đầy đủ để đưa ra kết quả đầu ra của mô hình.

Chi tiết về kiến trúc của mô hình có thể xem ở hình dưới đây:



Hình 3.16: Kiến trúc mô hình Multihead-attention-rnn với số attention=1 [31]

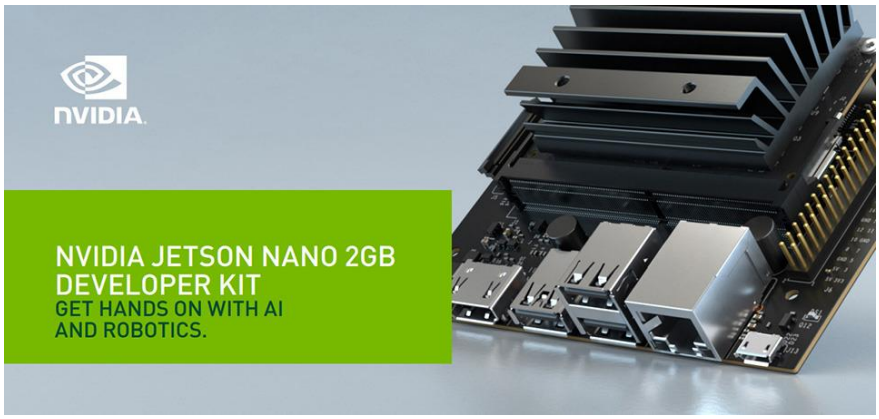
Ta có thể dễ nhận thấy rằng, mặc dù nhóm tác giả đã cho thấy được chức năng cũng như tính hiệu quả tại từng lớp trích xuất đặc trưng của mô hình, nhưng kiến trúc của nó thực sự còn quá phức tạp, khối lượng tính toán lớn, có thể mất nhiều thời gian dự đoán khi chạy trên các thiết bị tài nguyên thấp.

3.4 Phương pháp đưa mô hình lên thiết bị

3.4.1 Giới thiệu về JETSON NANO 2GB

JETSON NANO 2GB [1] là một thiết bị thông minh, một chiếc máy tính phiên bản thu gọn có CPU, RAM và GPU riêng biệt. Con số 2GB ở đây ám chỉ thiết bị được trang bị 2GB RAM. JETSON NANO 2GB được sử dụng rất nhiều trong các hệ thống điều khiển thông minh như ở camera trường học, robot nhà máy, xí nghiệp, ở đó yêu cầu một khối lượng lớn các công việc tự động, đòi hỏi các thao tác xử lý phải nhanh, độ chính xác cao. Hệ điều hành thường được sử dụng trên thiết bị này là Ubuntu. Điểm mạnh của JETSON NANO 2GB là nó có sử dụng GPU, rất thích hợp cho việc tính toán các hệ thống xử lý ảnh, nhận diện tiếng nói. Ngoài ra, cũng có một số thiết bị thông minh khác được sử dụng phổ biến trên thế giới như: Raspberry pi 3, Raspberry pi 4, vv...

Đồ án này có nhiệm vụ đưa mô hình nhận diện phát âm từ khoá lên thiết bị JETSON NANO 2GB với một độ chính xác cao, thời gian phản hồi cho một giao dịch dự đoán phải nhanh nhất có thể, tài nguyên hệ thống chiếm dụng phải là ít nhất. Có hai phương pháp để có thể chạy một mô hình nhận diện phát âm từ khoá trên JETSON NANO 2GB là đưa trực tiếp hoặc đưa thông qua một mô hình rút gọn.



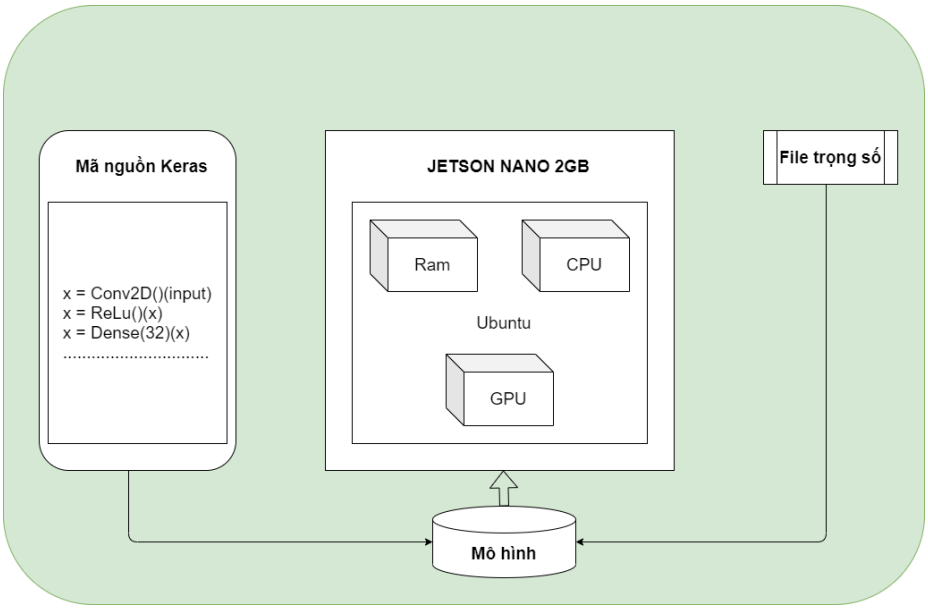
Hình 3.17: Thiết bị JETSON NANO 2GB [1]

3.4.2 Đưa trực tiếp

Mỗi một mô hình học máy cũng như học sâu sau khi huấn luyện xong thường có cấu trúc gồm hai phần chính: Phần đầu tiên là mã nguồn mô tả kiến trúc các lớp của mô hình, phần thứ hai là một file chứa các thông số của các ma trận trọng số của mô hình (hay còn gọi là weight).

Để mô hình có thể chạy được trên thiết bị, phương pháp đầu tiên là ta sẽ đưa trực tiếp cả 2 phần kể trên lên hệ thống. Kiến trúc các lớp của mô hình sẽ được định nghĩa bằng thư viện keras của tensorflow, file chứa các trọng số của

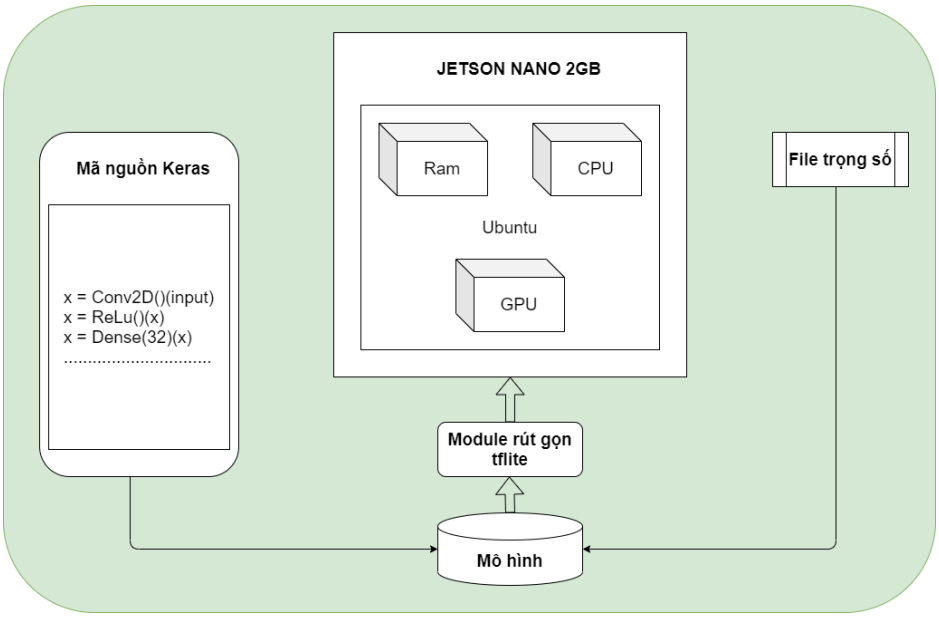
mô hình được lưu dưới dạng h5. Một giao dịch dự đoán sẽ được thực hiện thông qua kiến trúc mô hình, weight, và đầu vào hiện tại đang xét.



Hình 3.18: Phương pháp đưa trực tiếp mô hình lên JETSON NANO 2GB

3.4.3 Đưa thông qua dạng mô hình rút gọn

Bản chất của phương pháp này là sẽ cắt tỉa một vài lớp nhỏ trong mô hình nhằm mục đích giảm thiểu khối lượng tính toán. Một cơ chế nữa cũng làm tăng đáng kể hiệu năng xử lý của CPU, GPU đó là phương pháp chuyển kiểu dữ liệu hiện thời về dạng nhỏ hơn (ví dụ như thay vì thực hiện tính toán trên kiểu dữ liệu float32 thì ta tính toán trên kiểu float16). Cơ chế này cũng giúp tối ưu không gian lưu trữ dữ liệu của hệ thống. Để hiện thoá vấn đề này, thì google đã cung cấp một công cụ có tên là **tflite_converter** [32] với mục đích chuyển đổi một mô hình học sâu phức tạp, sau một số bước cắt tỉa cũng như tối ưu mô hình, tạo ra một mô hình mới nhỏ gọn hơn, thời gian xử lý nhanh hơn, rất hiệu quả khi chạy trên các thiết bị tài nguyên thấp, cụ thể trong bài toán này là JETSON NANO 2GB.



Hình 3.19: Đưa mô hình lên thiết bị bằng module tflite

Tuy nhiên, một hạn chế của phương pháp này là độ chính xác của mô hình rút gọn sẽ bị giảm đi một chút (không đáng kể so với mô hình gốc ban đầu - khoảng 1 đến 2%), lý do là ta đã thực hiện cắt tỉa một số lớp của mô hình gốc, dẫn đến trong quá trình lan truyền xuôi, một số feature map không còn thể hiện đầy đủ các đặc tính để phân biệt giữa các class giống như mô hình gốc ban đầu nữa.

3.5 Xây dựng thuật toán ghi âm audio theo luồng thời gian thực

Như chúng ta đã biết, mô hình nhận diện phát âm từ khoá hoạt động với đầu vào là đặc trưng MFCC của tín hiệu âm thanh, đầu ra là định nghĩa từ khoá tương ứng với tín hiệu âm thanh đó. Đặc trưng này có thể được trích xuất từ một trong 2 cách sau đây:

- Các file audio đã được thu âm sẵn, được lưu dưới một định dạng nào đó (mp3, wav, flac, ...)
- Dữ liệu audio được tổng hợp theo luồng thời gian thực thông qua microphone của người dùng.

Từ 2 cách trích xuất đặc trưng như trên, ta sẽ có 2 cách để mô hình có thể nhận diện phát âm từ khoá theo luồng thời gian thực thông qua quá trình ghi âm như sau:

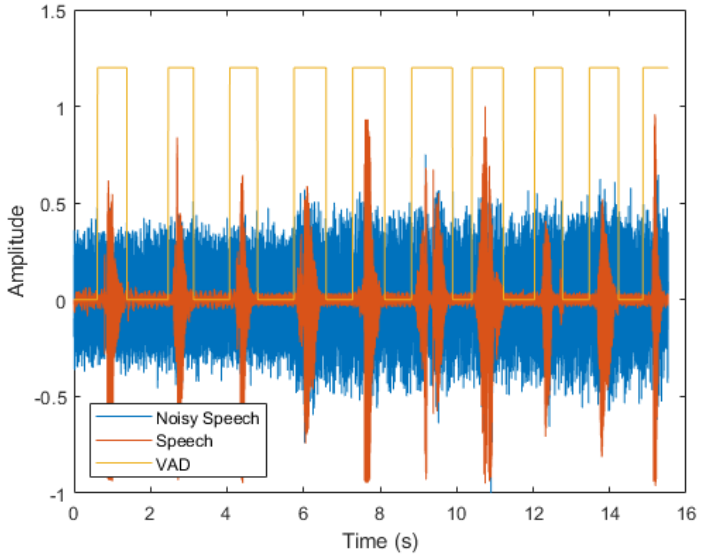
Cách 1: Đặt giới hạn thời gian cho một lần thu âm (mỗi file audio huấn luyện có độ dài trung bình là 2 giây nên ta sẽ đặt là 2 giây). Dữ liệu sau khi thu âm sẽ được trích xuất đặc trưng và đưa vào mô hình để dự đoán. Phương pháp này có một nhược điểm rất lớn đó là thời gian nhận diện phát âm từ khoá từ lúc thu âm đến lúc dự đoán ra kết quả luôn phải lớn hơn 2 giây, hơn nữa, ta sẽ không xác định được thời điểm khi nào người thu âm sẽ nói, do đó hệ thống có thể sẽ không hoạt động tốt nếu như người dùng chưa nói kịp hoặc mới chỉ nói được một phần âm thanh từ khoá.

Cách 2: Ta sẽ tìm ra một cơ chế để nhận diện thời điểm bắt đầu mà người thu âm sẽ nói, đồng thời ta cũng sẽ thống kê ra thời gian trung bình để một người bình thường có thể nói xong một từ khoá rồi mới đặt một giới hạn thời gian thu âm hợp lý, qua đó thì ta sẽ tiết kiệm được thời gian, hơn là chờ đợi microphone tổng hợp dữ liệu âm thanh khi tuân theo một độ dài audio fix cứng (2 giây).

3.5.1 Thuật toán phát hiện giọng nói

Giải pháp để tìm ra thời điểm bắt đầu giọng nói của người thu âm được giải quyết bằng một bài toán khác đó là: bài toán nhận diện giọng nói (Voice Activity Detection - VAD). Nhiệm vụ của bài toán này là, với một chuỗi tín hiệu âm thanh đầu vào, ta cần phân biệt xem nó có phải là tiếng người hay là một loại âm thanh nhiễu nào đó. Có rất nhiều phương pháp cũng như thuật toán để giải quyết bài toán này, một trong số đó là thuật toán **webrtcvad** nổi tiếng của google. Tư tưởng chính của thuật toán này là sử dụng một phân phối chuẩn để dự đoán xác suất xem một chuỗi tín hiệu âm thanh đầu vào có phải là tiếng nói hay là không. Thông thường thì thuật toán sẽ xử lý lần lượt với từng frame trong chuỗi tín hiệu âm thanh gốc.

Quá trình thu âm là quá trình mà microphone nhận các sóng âm thanh liên tục từ người thu âm, chuyển các sóng đó về miền tín hiệu rời rạc, các miền tín hiệu này được tổng hợp liên tục theo luồng thời gian thực, cho nên ta hoàn toàn có thể áp dụng thuật toán webrtcvad để dự đoán cho từng phần âm thanh của miền tín hiệu trong quá trình ghi âm. Từ đó thì ta có thể biết được đâu chính là thời điểm mà người thu âm bắt đầu phát ra tiếng nói, một yếu tố quan trọng trong quá trình nhận diện phát âm từ khoá.



Hình 3.20: Minh hoạ bài toán nhận diện giọng nói [33]

3.5.2 Thuật toán tổng hợp giọng nói theo luồng thời gian thực

Việc biết được thời điểm bắt đầu giọng nói thôi là chưa đủ, biết khi nào là kết thúc âm thanh của từ, biết khi nào là nên kết thúc luồng ghi âm cũng là một yếu tố rất quan trọng quyết định đến thời gian nhận diện phát âm từ khoá. Sau đây là mô tả chi tiết về giải thuật tổng hợp giọng nói theo luồng thời gian thực:

Đầu vào: các frame của tín hiệu âm thanh đã được thu âm đến thời điểm hiện tại

Đầu ra: đoạn âm thanh được nhận diện là tiếng người

Ta định nghĩa:

- sample_rate: tần số lấy mẫu âm thanh
- frame_duration_ms: thời gian tính theo giây trong một audio frame
- frame_size = phần nguyên của $\frac{\text{sample\_rate x frame\_duration\_ms}}{1000}$: kích thước (tính theo số điểm lấy mẫu) trong một frame.
- num_frame_in_speech_indentify: số frame sẽ được xem xét để tổng hợp giọng nói
- buffer_data[]: mảng lưu lại toàn bộ các frame trong quá trình thu âm
- buffer_frame_speech_indentify[num_frame_in_speech_indentify]: mảng lưu lại các audio frame dùng để tổng hợp giọng nói.

Cơ sở cốt lõi của giải thuật:

Thuật toán sử dụng một mảng buffer_frame_speech_indentify có độ dài cố định để lưu các frame audio**,** và sẽ ghi âm cho đến khi nhận được tỷ lệ số frame được cho là tiếng nói lớn hơn 80 hoặc 90% tổng số frame trong mảng này, cùng lúc đó thời điểm mà người thu âm bắt đầu nói sẽ được lưu lại và quá trình ghi âm kết thúc. Dữ liệu được đưa vào mô hình nhận diện sẽ là dữ liệu bắt đầu từ thời điểm ghi âm cho đến khi kết thúc.

Mã giả cho giải thuật:

```
got_a_sentence = False;
stop_record = False;
while not stop_record: # vòng lặp thu âm
    khởi tạo mảng buffer_frame_speech_indentify theo giá trị 0;
    speech_indentify_index = 0;
    khởi tạo mảng buffer_data;
    index = 0; biến đếm số frame được thu âm cho đến hiện tại
    start_point = 0; thời điểm bắt đầu tiếng nói
    while not got_a_sentence: # vòng lặp tổng hợp tiếng nói
        frame = read_frame(); đọc một frame từ microphone
        buffer_data.append(frame); thêm frame vào mảng buffer_data
        time_use = time(); thời gian ghi âm cho đến thời điểm hiện tại
        index += frame_size;
        active = webrtcvad(frame); sử dụng webrtcvad nhận diện tiếng nói
        active = 1 nếu như frame này là tiếng nói;
        active = 0 nếu như frame này là âm thanh nhiễu;
        if active = 1:
            buffer_frame_speech_indentify[speech_indentify_index]  = 1;
        else:
            buffer_frame_speech_indentify[speech_indentify_index]  = 0;
        speech_indentify_index += 1;
        speech_indentify_index %= num_frame_in_speech_indentify;
        if num_voiced > 0.8 x num_frame_in_array_speech_indentify:
            start_point = index – frame_size x 13;
            got_a_sentence = True;
    audio_data = buffer_data[start_point:];
    stop_record = True;
```

Ta có thể dễ nhận thấy rằng, độ phức tạp của thuật toán này là $O(n^2)$, tuy nhiên do số lượng frame cho một từ khoá rất nhỏ (khoảng 13 frame), cho nên thuật toán hoàn toàn có thể hoạt động một cách “real time” (theo thời gian thực) trên các thiết bị tài nguyên thấp.

CHƯƠNG 4. THỰC NGHIỆM VÀ ĐÁNH GIÁ

4.1 Dữ liệu

Dưới đây là bảng phân bố dữ liệu vùng miền cho 2 tập dữ liệu DS2020 và DS2021.

Bộ dữ liệu	Tổng số file audio	Số audio miền Bắc	Số audio miền Trung	Số audio miền Nam	Số audio giọng nữ	Số audio giọng nam
DS2020	2968	2802	142	24	422	2546
DS2021	5434	5214	165	55	615	4819

Nhìn vào bảng phân bố dữ liệu bên trên, ta thấy rằng dữ liệu audio tập chung phủ cho giọng nam miền Bắc, số lượng audio ở miền Trung và miền Nam chiếm tỷ lệ khá thấp khoảng 5.6% cho bộ dữ liệu DS2020 và 4.51% cho bộ dữ liệu DS2021. Tỷ lệ giọng nam cũng chiếm ưu thế so với giọng nữ, khoảng 86% cho nam và 14% cho nữ. Do việc thu âm chỉ ở một phạm vi nhất định cho nên đồ án này sẽ tập chung tối ưu, cũng như thử nghiệm mô hình nhận diện phát âm từ khoá cho dữ liệu audio giọng nam miền Bắc.

Phân bố về số file audio cho từng nhãn trong từng tập dữ liệu được mô tả bởi bảng sau:

Bộ DS2020		Bộ DS2021	
Tên nhãn	Số lượng file audio	Tên nhãn	Số lượng file audio
Ba	178	Ba	326
Bảy	166	Bảy	315
Bốn	175	Bốn	323
Chín	191	Chín	337
Có	196	Có	349
Hai	178	Hai	328
Không	184	Không	325
Lên	191	Lên	342
Lùi	168	Lùi	317
Một	168	Một	311
Năm	172	Năm	323
Phải	172	Phải	318
Sáu	125	Sáu	212
Tám	175	Tám	326
Tiến	182	Tiến	330
Trái	173	Trái	329
Xuống	174	Xuống	323

Như đã trình bày trong phần 3.3.1, chương 3, bộ dữ liệu DS2020, DS2021 sẽ được chia ra thành các bộ dữ liệu DS1, DS2, DS3, DS4, DS5, DS6, DS7 có phân bố như trong các bảng sau:

Bộ DS2020 lấy tập test theo đủ phân phối nam, nữ, vùng miền:

Tập dữ liệu DS1	**Tổng số file audio**	**Số audio miền Bắc**	**Số audio miền Trung**	**Số audio miền Nam**	**Số audio giọng nữ**	**Số audio giọng nam**
Train – 95%	2823	2682	119	22	378	2445
Test – 5%	145	120	23	2	44	101

Bộ DS2021 chỉ lấy tập test giọng nam miền Bắc:

Tập dữ liệu DS2	**Tổng số file audio**	**Số audio miền Bắc**	**Số audio miền Trung**	**Số audio miền Nam**	**Số audio giọng nữ**	**Số audio giọng nam**
Train – 90%	4873	4653	165	55	615	4258
Test – 10%	561	561	0	0	0	561

Tập dữ liệu DS3	**Tổng số file audio**	**Số audio miền Bắc**	**Số audio miền Trung**	**Số audio miền Nam**	**Số audio giọng nữ**	**Số audio giọng nam**
Train – 80%	4312	4092	165	55	615	3697
Test – 20%	1122	1122	0	0	0	1122

Bộ DS2021 chỉ lấy tập test giọng nam, nữ miền Bắc:

Tập dữ liệu DS4	**Tổng số file audio**	**Số audio miền Bắc**	**Số audio miền Trung**	**Số audio miền Nam**	**Số audio giọng nữ**	**Số audio giọng nam**
Train – 90%	4873	4653	165	55	343	4530
Test – 10%	561	561	0	0	272	289

Tập dữ liệu DS5	**Tổng số file audio**	**Số audio miền Bắc**	**Số audio miền Trung**	**Số audio miền Nam**	**Số audio giọng nữ**	**Số audio giọng nam**
Train – 80%	4312	4092	165	55	343	3969
Test – 20%	1122	1122	0	0	272	850

Bộ DS2021 chỉ lấy một nửa dữ liệu trong tập train để huấn luyện, tập test là giọng nam miền Bắc:

Tập dữ liệu DS6	Tổng số file audio	Số audio miền Bắc	Số audio miền Trung	Số audio miền Nam	Số audio giọng nữ	Số audio giọng nam
Train – 45%	2410	2305	81	24	294	2116
Test – 10%	561	561	0	0	0	561

Tập dữ liệu DS7	Tổng số file audio	Số audio miền Bắc	Số audio miền Trung	Số audio miền Nam	Số audio giọng nữ	Số audio giọng nam
Train – 40%	2130	2023	76	31	294	1836
Test – 20%	1122	1122	0	0	0	1122

4.2 Môi trường cài đặt

4.2.1 Môi trường huấn luyện mô hình

Tất cả các mô hình của bài toán được xây dựng trên máy tính cá nhân có cấu hình như sau: CPU i5 8300H 2.3GHz, RAM 8G DDR4, GPU NVIDIA GTX 1050 4G GDDR5

Hệ điều hành: Windows 10

Ngôn ngữ lập trình: Python 3.6.9

Các thư viện chính được sử dụng:

- Tensorflow, keras: Dùng cho việc thiết kế kiến trúc và huấn luyện mô hình
- Librosa: Dùng cho việc xử lý, bóc tách đặc trưng của âm thanh
- Pyaudio: Dùng cho việc ghi âm thanh người dùng từ microphone
- Webrtcvad: Dùng cho việc phát hiện giọng nói
- Ngoài ra còn một số thư viện khác phục vụ cho các công việc liên quan đến quản lý file, cấu hình, ...

4.2.2 Môi trường thử nghiệm nhận diện phát âm từ khoá

Mô hình nhận diện phát âm từ khoá sẽ được thử nghiệm trên thiết bị JETSON NANO 2GB có cấu hình như sau:

- CPU: Quad-core ARM A57 @ 1.43 GHz
- RAM: 2GB 64-bit LPDDR4 25.6 GB/s
- GPU: 128-core NVIDIA Maxwell 2GB VRAM

Hệ điều hành: Ubuntu 18.04

Ngôn ngữ lập trình, các thư viện sử dụng đều giống với môi trường huấn luyện đã nói ở trên.

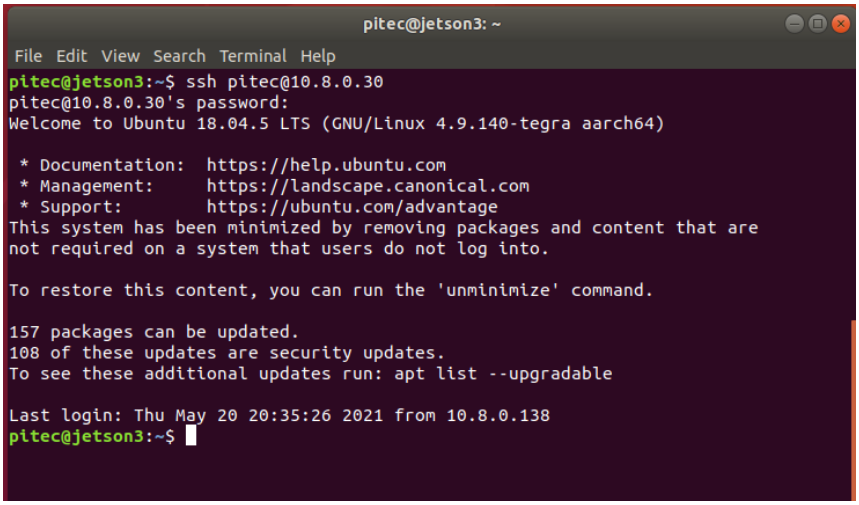
4.3 Cách cài đặt triển khai, đưa mô hình lên thiết bị

4.3.1 Cài đặt các thư viện cần thiết

- **Cài đặt tensorflow bản 2.3.1**
  python3 –m pip install tensorflow==2.3.1
- **Cài đặt numba bản 0.48**
  python3 –m pip install numba==0.48
- **Cài đặt librosa bản 0.7.0**
  python3 –m pip install librosa==0.7.0
- **Cài đặt pyaudio**
  sudo apt-get install portaudio19-dev python-pyaudio python3-pyaudio
  python3 –m pip install PyAudio
- **Cài đặt webrtcvad**
  python3 –m pip install webrtcvad

4.3.2 Đưa mô hình lên thiết bị JETSON NANO 2GB

Để có thể điều khiển thiết bị JETSON NANO 2GB từ xa, chúng tôi sẽ kết nối thông qua giao thức ssh. Ví dụ để kết nối đến một thiết bị JETSON NANO 2GB có hostname là: pitec@10.8.0.30, ta thực hiện câu lệnh: ssh pitec@10.8.0.30, sau đó nhập mật khẩu là có thể truy cập được vào giao diện dòng lệnh điều khiển của thiết bị.



Hình 4.1: Kết nối đến thiết bị JETSON NANO 2GB

Chúng tôi sẽ sử dụng git là công cụ để có thể đưa mã nguồn mô hình nhận diện phát âm từ khoá lên trên thiết bị: sudo apt-get install git

4.4 Các giai đoạn và kết quả thực nghiệm

4.4.1 Xây dựng các mô hình phân loại trên máy tính cá nhân

Giai đoạn này sẽ thực hiện huấn luyện các mô hình nhận diện phát âm từ khoá trên các bộ dữ liệu DS1, DS2, DS3, DS4, DS5, DS6, DS7 đã được định nghĩa ở phần 4.1.

Các tham số được dùng cho mô hình SVM:

- Tham số kernel: rbf
- Tham số C: 1.0
- Tham số max_iter (số vòng lặp tối đa): 1000

Các siêu tham số, hàm mất mát được dùng cho mô hình TC-RESNET, MULTIHEAD-ATTENTION-RNN:

- Kích thước batch: batch_size = 32
- Tốc độ học: learning_rate = 1e-5
- Số epochs: epochs = 30
- Hàm mất mát: sparse_categorical_crossentropy

Các tham số dùng cho việc trích xuất đặc trưng MFCC:

- Kích thước frame: n_fft = 400
- Tần số lấy mẫu: sr = 16000
- Số hệ số MFCC cần lấy: n_mfcc = 40
- Bước khung: hop_length = 100

Các tham số dùng cho việc làm giàu dữ liệu:

- Số bước nâng cao độ: pitch_change = 2
- Số cung được xem xét: bins_per_octave = 12
- Hệ số γ dùng cho việc trộn âm thanh nhiễu: Lấy theo phân phối chuẩn trong khoảng (0, 0.1)
- Âm lượng tối đa cho audio: max_dBFS = -16
- Âm lượng tối thiểu cho audio: min_dBFS = -50

Mô hình SVM:

Kết quả độ chính xác, điểm F1 trên tập test sau quá trình huấn luyện mô hình SVM với hai bộ dữ liệu DS1, DS2, được mô tả chi tiết trong bảng sau:

SVM		
Bộ dữ liệu	**Độ chính xác**	**Điểm F1**
DS1	31.72	30.28
DS2	49.55	48.84

Ta có thể thấy rằng, thuật toán SVM dường như hoạt động không hiệu quả dưới không gian đặc trưng MFCC, hơn thế nữa, thời gian huấn luyện cho mô hình rất lâu (khoảng 9 tiếng đồng hồ), độ chính xác cho tập dữ liệu DS1 chỉ đạt khoảng 32%, trong khi đó xét trên tập DS2 (có tập test chỉ bao gồm các audio giọng nam miền Bắc), độ chính xác cũng chỉ đạt được khoảng 50%. Vì lý do đó, mà chúng tôi sẽ không tiến hành thử nghiệm mô hình SVM cho các bộ dữ liệu còn lại, để tiết kiệm chi phí thời gian cho các mô hình có độ chính xác cao hơn.

Dưới đây là bảng phân tích Precision, Recall cho từng class của mô hình:

Class	**Precision**	**Recall**	**F1-Score**
Ba	0.38	0.30	0.34
Bảy	0.37	0.48	0.42
Bốn	0.54	0.64	0.58
Chín	0.60	0.64	0.62
Có	0.62	0.64	0.63
Hai	0.37	0.42	0.39

Không	0.52	0.36	0.43
Lên	0.70	0.64	0.67
Lùi	0.55	0.73	0.62
Một	0.47	0.64	0.54
Năm	0.46	0.39	0.43
Phải	0.46	0.33	0.39
Sáu	0.80	0.24	0.37
Tám	0.32	0.30	0.31
Tiến	0.50	0.64	0.56
Trái	0.49	0.52	0.50
Xuống	0.50	0.52	0.51

Mô hình TC-RESNET:

Kết quả độ chính xác, điểm F1 trên tập test sau quá trình huấn luyện mô hình TC-RESNET với bảy bộ dữ liệu từ DS1 đến DS7, được mô tả chi tiết trong hai bảng sau:

- **Mô hình không sử dụng phương pháp làm giàu dữ liệu:**

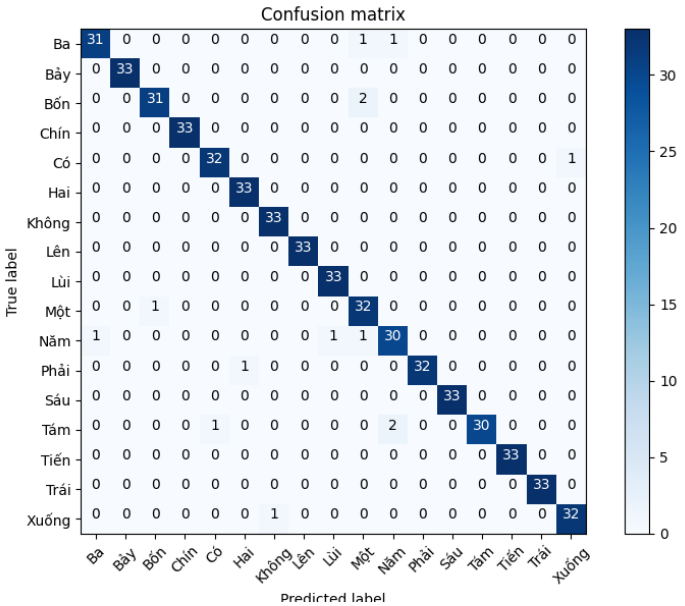
TC-RESNET8			**TC-RESNET14**		
Bộ dữ liệu	**Độ chính xác**	**Điểm F1**	**Bộ dữ liệu**	**Độ chính xác**	**Điểm F1**
DS1	93.79	93.79	DS1	95.17	95.22
DS2	95.90	95.88	DS2	96.61	96.62
DS3	95.28	95.31	DS3	96.08	96.09
DS4	93.94	93.92	DS4	95.37	95.39
DS5	93.49	93.49	DS5	94.21	94.22
DS6	93.76	93.78	DS6	93.76	93.78
DS7	93.40	93.39	DS7	93.67	93.69

- **Mô hình sử dụng phương pháp làm giàu dữ liệu:**

TC-RESNET8			**TC-RESNET14**		
Bộ dữ liệu	**Độ chính xác**	**Điểm F1**	**Bộ dữ liệu**	**Độ chính xác**	**Điểm F1**
DS1	95.17	95.16	DS1	96.55	96.59
DS2	96.26	96.25	DS2	97.50	97.50
DS3	96.08	96.09	DS3	96.17	96.18
DS4	94.47	94.47	DS4	96.43	96.42
DS5	93.85	93.87	DS5	95.54	95.57
DS6	94.30	94.30	DS6	94.47	94.48
DS7	93.58	93.56	DS7	94.12	94.11

Qua hai bảng kết quả ở trên, ta thấy được hiệu quả rõ rệt của phương pháp làm giàu dữ liệu (đã giới thiệu ở chương 3). Mô hình đạt kết quả tốt nhất ở tập dữ liệu DS2 (dữ liệu được chia theo tỷ lệ 90:10, trong đó tập test chỉ chứa giọng nam miền Bắc) với độ chính xác khoảng 97.50%. Với việc sử dụng phương pháp nâng, giảm cao độ, mô hình phân loại cho bộ dữ liệu DS4 (dữ liệu được chia theo tỷ lệ 90:10, trong đó tập test chứa 50% là giọng nam miền Bắc, 50% là giọng nữ miền Bắc) cho một kết quả rất khả quan với độ chính xác khoảng 96.43%. Ta cũng thấy được sự khác nhau về độ chính xác trên tập test khi sử dụng số lượng các mẫu huấn luyện khác nhau, cụ thể là trên cùng một tập test thì mô hình cho bộ dữ liệu DS2 có độ chính xác cao hơn mô hình cho bộ dữ liệu DS6 (Hơn kém nhau khoảng 3%).

Bảng ma trận nhầm lẫn giữa các class của mô hình:



Hình 4.2: Bảng ma trận nhầm lẫn của mô hình TC-RESNET với bộ dữ liệu DS2

Bảng phân tích Precision, Recall cho từng class của mô hình:

Class	Precision	Recall	F1-Score
Ba	0.97	0.94	0.95
Bảy	1.00	1.00	1.00
Bốn	0.97	0.94	0.95
Chín	1.00	1.00	1.00
Có	0.97	0.97	0.97
Hai	0.97	1.00	0.99
Không	0.97	1.00	0.99
Lên	1.00	1.00	1.00
Lùi	0.97	1.00	0.99
Một	0.89	0.97	0.93
Năm	0.91	0.91	0.91
Phải	1.00	0.97	0.98

Sáu	1.00	1.00	1.00
Tám	1.00	0.91	0.95
Tiến	1.00	1.00	1.00
Trái	1.00	1.00	1.00
Xuống	0.97	0.97	0.97

Mô hình MULTIHEAD-ATTENTION-RNN:

Kết quả độ chính xác, điểm F1 trên tập test sau quá trình huấn luyện mô hình MULTIHEAD-ATTENTION-RNN với bảy bộ dữ liệu từ DS1 đến DS7 được mô tả chi tiết trong hai bảng sau:

- **Mô hình không sử dụng phương pháp làm giàu dữ liệu:**

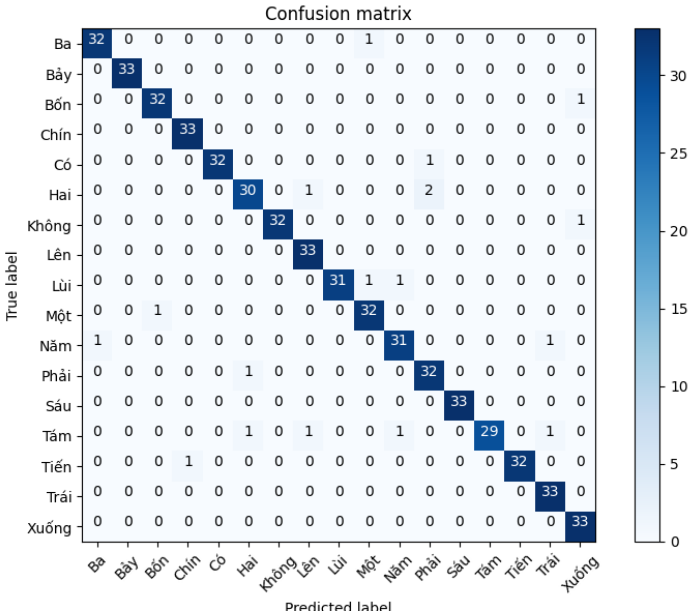
MULTIHEAD-ATTENTION-RNN		
Bộ dữ liệu	**Độ chính xác**	**Điểm F1**
DS1	86.90	86.87
DS2	95.72	95.70
DS3	94.12	94.09
DS4	90.02	89.99
DS5	89.13	89.11
DS6	92.34	92.36
DS7	91.44	91.44

- **Mô hình sử dụng phương pháp làm giàu dữ liệu:**

MULTIHEAD-ATTENTION-RNN		
Bộ dữ liệu	**Độ chính xác**	**Điểm F1**
DS1	93.79	93.86
DS2	96.79	96.78
DS3	96.00	96.00
DS4	94.47	94.45
DS5	93.85	93.86
DS6	94.65	94.64
DS7	93.94	93.95

Mô hình MULTIHEAD-ATTENTION-RNN lại một lần nữa chứng minh sự hiệu quả của phương pháp làm giàu dữ liệu. Mô hình đạt kết quả tốt nhất trên tập dữ liệu DS2 với độ chính xác khoảng 96.79%. Độ chính xác trên tập DS6, DS7 (sử dụng 50% dữ liệu trong tập train để huấn luyện) đều nhỏ 93%. Ta có thể thấy được tầm quan trọng của số lượng dữ liệu cần huấn luyện cho mô hình nhận diện phát âm từ khoá, một lượng dữ liệu đủ lớn, sạch sẽ làm cho mô hình của chúng ta càng tổng quát và càng chính xác hơn.

Bảng ma trận nhầm lẫn giữa các class của mô hình:



Hình 4.3: Bảng ma trận nhầm lẫn của mô hình MULTIHEAD-ATTENTION-RNN với bộ dữ liệu DS2

Bảng phân tích Precision, Recall cho từng class của mô hình:

Class	Precision	Recall	F1-Score
Ba	0.97	0.97	0.97
Bảy	1.00	1.00	1.00
Bốn	0.97	0.97	0.97
Chín	0.97	1.00	0.99
Có	1.00	0.97	0.98
Hai	0.94	0.91	0.92
Không	1.00	0.97	0.98
Lên	0.94	1.00	0.97
Lùi	1.00	0.94	0.97
Một	0.94	0.97	0.96
Năm	0.94	0.94	0.94
Phải	0.91	0.97	0.94
Sáu	1.00	1.00	1.00
Tám	1.00	0.88	0.94
Tiến	1.00	0.97	0.98
Trái	0.94	1.00	0.97
Xuống	0.94	1.00	0.97

4.4.2 Thực nghiệm nhận diện phát âm từ khoá trên máy tính cá nhân

Thời gian nhận diện phát âm cho một từ khoá sẽ bao gồm 3 thành phần cơ bản như sau:

- Thời gian tổng hợp, tiền xử lý dữ liệu từ microphone kể từ lúc bắt đầu nói
- Thời gian trích xuất đặc trưng âm thanh
- Thời gian dự đoán của mô hình

Hai bảng dưới đây sẽ so sánh chi tiết về thời gian nhận diện một từ khoá của 2 mô hình TC-RESNET và MULTIHEAD-ATTENTION-RNN trên máy tính cá nhân:

- **Mô hình dưới dạng keras**

TC-RESNET8		**TC-RESNET14**		**MULTIHEAD-ATTENTION-RNN**	
Phần thời gian	**Thời gian (ms)**	**Phần thời gian**	**Thời gian (ms)**	**Phần thời gian**	**Thời gian (ms)**
Tiền xử lý	5	Tiền xử lý	5	Tiền xử lý	5
Tách đặc trưng	3	Tách đặc trưng	3	Tách đặc trưng	5
Dự đoán	25	Dự đoán	29	Dự đoán	40
Tổng	33	Tổng	37	Tổng	50

- **Mô hình dưới dạng tflite**

TC-RESNET8		**TC-RESNET14**		**MULTIHEAD-ATTENTION-RNN**	
Phần thời gian	**Thời gian (ms)**	**Phần thời gian**	**Thời gian (ms)**	**Phần thời gian**	**Thời gian (ms)**
Tiền xử lý	5	Tiền xử lý	5	Tiền xử lý	5
Tách đặc trưng	3	Tách đặc trưng	3	Tách đặc trưng	5
Dự đoán	2	Dự đoán	2	Dự đoán	20
Tổng	10	Tổng	10	Tổng	30

Thông qua các kết quả so sánh ở trên, chúng tôi rút ra một số nhận xét như sau:

- Mô hình dưới dạng tflite cho thời gian dự đoán nhanh hơn khoảng 10 lần so với mô hình gốc keras
- Mô hình MULTIHEAD-ATTENTION-RNN có thời gian dự đoán chậm hơn khoảng 2 lần so với mô hình TC-RESNET. Điều này có thể giải thích do cơ chế tính toán song song của GPU không thực sự hiệu quả trên các cấu trúc mạng hồi quy (RNN, LSTM), trong khi đó lại rất hiệu quả cho những kiến trúc mạng có cấu trúc lưới, điển hình là CNN.
- Nhìn chung, thời gian dự đoán phát âm cho một từ khoá của các mô hình đều dưới 1 giây, cho nên ta có thể kết luận được mô hình nhận diện phát âm từ khoá hoạt động tốt trên các thiết bị máy tính cá nhân.

4.4.3 Thực nghiệm nhận diện phát âm từ khoá trên thiết bị JETSON NANO 2GB

- **Mô hình dưới dạng keras**

TC-RESNET8		TC-RESNET14		MULTIHEAD-ATTENTION-RNN	
Phần thời gian	**Thời gian (ms)**	**Phần thời gian**	**Thời gian (ms)**	**Phần thời gian**	**Thời gian (ms)**
Tiền xử lý	8	Tiền xử lý	8	Tiền xử lý	8
Tách đặc trưng	24 - 60	Tách đặc trưng	25 - 60	Tách đặc trưng	25 - 60
Dự đoán	221	Dự đoán	235	Dự đoán	313
Tổng	253 - 289	Tổng	268 - 304	Tổng	346 - 382

- **Mô hình dưới dạng tflite**

TC-RESNET8		TC-RESNET14		MULTIHEAD-ATTENTION-RNN	
Phần thời gian	**Thời gian (ms)**	**Phần thời gian**	**Thời gian (ms)**	**Phần thời gian**	**Thời gian (ms)**
Tiền xử lý	8	Tiền xử lý	8	Tiền xử lý	8
Tách đặc trưng	19 - 60	Tách đặc trưng	20 - 60	Tách đặc trưng	23 - 60
Dự đoán	4	Dự đoán	7	Dự đoán	278
Tổng	31 - 72	Tổng	35 - 76	Tổng	309 - 350

Ta nhận thấy rằng, để mô hình nhận diện phát âm từ khoá có thể nhận diện một cách "real time" (tuân theo thời gian thực) thì các mô hình dạng keras không thể đáp ứng như cầu đó, do vậy ta bắt buộc phải sử dụng chúng dưới dạng các mô hình tflite. Thời gian tổng cộng để nhận diện một từ khoá bởi mô hình TC-RESNET14 trong khoảng từ 0.04 đến 0.1 giây, bởi mô hình MULTIHEAD-ATTENTION-RNN trong khoảng 0.3 đến 0.7 giây. Hai mô hình này hoàn toàn có thể đáp ứng các nhu cầu về độ chính xác cũng như thời gian phản hồi cho một giao dịch dự đoán phát âm từ khoá.

Tóm lại, các mô hình nhận diện phát âm từ khoá được đề xuất trong đồ án hoạt động hiệu quả trên thiết bị JETSON NANO 2GB, sử dụng các kiến trúc lớp mạng không quá phức tạp, cồng kềnh mà vẫn cho được một độ chính xác và thời gian phản hồi mong muốn.

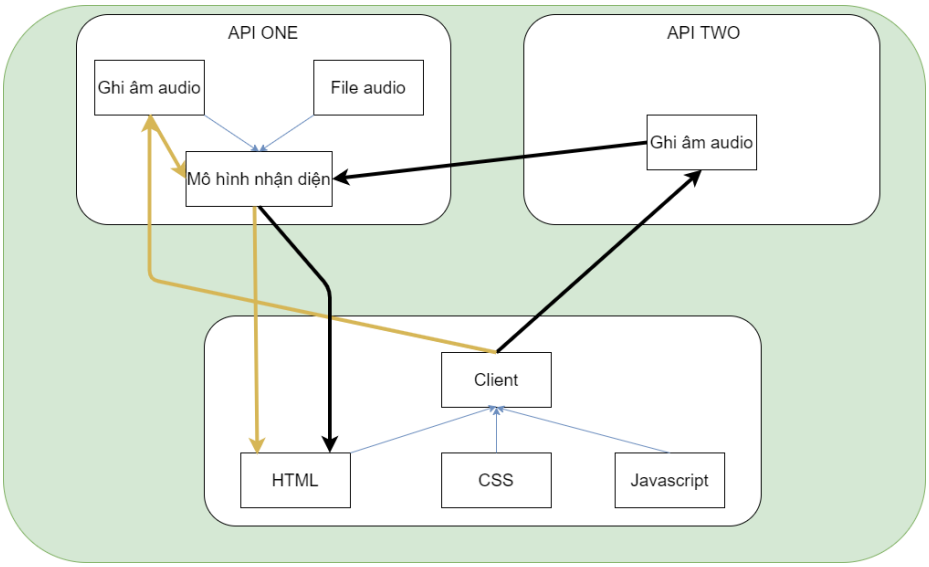
4.5 Cách thức demo mô hình, kết quả nhận diện phát âm từ khoá của một số người dùng cụ thể

4.5.1 Cách thức demo

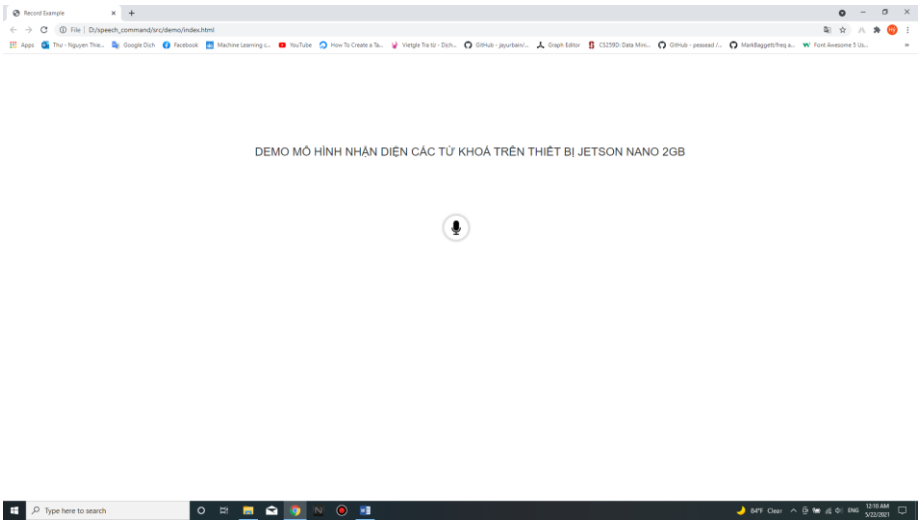
Về phía server, chúng tôi tiến hành xây dựng một webservice - API sử dụng thư viện flask của python cho mô hình nhận diện phát âm từ khoá. API này sẽ có 2 chức năng chính đó là ghi âm, tổng hợp dữ liệu audio từ microphone của thiết bị JETSON NANO 2GB, dự đoán nhãn đầu ra cho một file audio hoặc dữ liệu audio được tổng hợp từ microphone. Chúng tôi cũng sẽ xây dựng một webservice - API trên máy tính cá nhân với chức năng ghi âm audio từ microphone , api này có mục đích cho phép người dùng có thể sử dụng trực tiếp micro thu âm từ chính thiết bị của mình để demo nhận diện phát âm từ khoá.

Về phía client, chúng tôi sẽ xây dựng một template HTML, cùng với các câu lệnh javascript cơ bản để gửi và nhận dữ liệu thông qua 2 API đã được thiết kết như ở bên trên.

Chi tiết về cấu trúc hệ thống demo mô hình nhận diện phát âm từ khoá được mô tả bởi hình vẽ dưới đây:



Hình 4.4: Cấu trúc hệ thống demo nhận diện phát âm từ khoá trên thiết bị



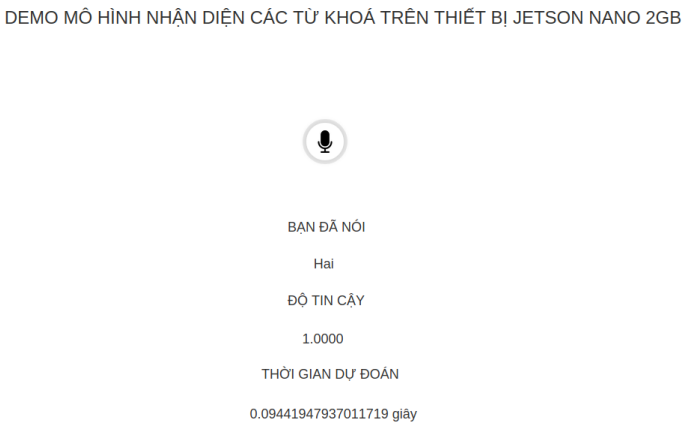
Hình 4.5: Giao diện web demo nhận diện phát âm từ khoá

4.5.2 Kết quả nhận diện phát âm từ khoá của một số người dùng cụ thể

Dưới đây là kết quả nhận diện phát âm từ khoá do một số đối tượng người dùng thực hiện:

- **Một số kết quả dự đoán đúng**

Hình 4.6: Kết quả dự đoán từ khoá "Bảy" của giọng nữ cao tuổi miền Bắc



Hình 4.7: Kết quả dự đoán từ khoá "Hai" của giọng nữ thiếu niên miền Bắc

Hình 4.8: Kết quả dự đoán từ khoá "Không" của giọng nam sinh viên miền Bắc

- **Một số kết quả dự đoán sai**

Hình 4.9: Kết quả dự đoán từ khoá bởi tiếng xịt xoạt

Hình 4.10: Kết quả dự đoán từ khoá bởi tiếng ồn ào xe cộ

Hình 4.11: Kết quả dự đoán từ khoá bởi tiếng vỗ tay

- **Nhận xét chung**

Mô hình nhận diện phát âm từ khoá phần nào đó đã dự đoán chính xác âm thanh các từ khoá được nói ra với một độ tin cậy rất cao (phần lớn đều lớn hơn 90%). Các âm thanh khác không thuộc phạm vi được huấn luyện hoặc là các âm thanh nhiễu được dự đoán ra nhãn từ khoá nhưng với một độ tin cậy rất thấp (xem hình 4.10 và 4.11). Tuy nhiên thì có một số đoạn âm thanh nhiễu lại bắt đầu

bằng các âm có xuất hiện trong các từ khoá cho nên mô hình rất dễ bị dự đoán nhầm (hình 4.12). Thời gian phản hồi cho một giao dịch dự đoán trong khoảng từ 0.05 đến 0.1 giây (nguyên nhân có thể do sự khác nhau về thời gian bóc tách đặc trưng MFCC cho các âm thanh từ khoá khác nhau).

CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Đóng góp của đồ án

- Thực hiện bóc tách, sử dụng các kỹ thuật xử lý tín hiệu âm thanh để làm giàu dữ liệu được thu âm từ các bạn sinh viên trường đại học Bách Khoa Hà Nội
- Thực hiện trích xuất những đặc trưng quan trọng của tín hiệu âm thanh
- Xây dựng mô hình học máy SVM thực hiện phân loại các audio từ khoá
- Xây dựng mô hình TC-RESNET thực hiện phân loại các audio từ khoá
- Xây dựng mô hình MULTIHEAD-ATTENTION-RNN thực hiện phân loại các audio từ khoá
- Xây dựng thuật toán ghi âm audio theo thời gian thực cho mô hình nhận diện phát âm từ khoá
- Chuyển đổi các mô hình huấn luyện được sang dạng rút gọn rồi đưa lên thử nghiệm trên thiết bị JETSON NANO 2GB
- Đánh giá kết quả, hiệu năng của các mô hình dựa trên các cách chia tập dữ liệu khác nhau
- Đánh giá hiệu năng, tính ứng dụng thực tiễn của các mô hình nhận diện phát âm từ khoá trên thiết bị JETSON NANO 2GB

5.2 Khó khăn trong quá trình thực hiện

- Hạn chế về việc thu thập dữ liệu huấn luyện. Website thu âm hiện tại mới chỉ tiếp cận được đến một phần nhỏ số lượng sinh viên trong trường đại học Bách Khoa Hà Nội, số lượng các audio thuộc các giọng miền Nam và miền Trung đang còn rất hạn chế và gặp rất nhiều khó khăn trong việc thu thập
- Dữ liệu thu thập được còn có khá nhiều âm thanh nhiễu, ảnh hưởng tới chất lượng của mô hình
- Hạn chế về cấu hình máy tính, thời gian huấn luyện cho các mô hình còn khá lâu
- Chưa có kinh nghiệm trong việc xử lý các tín hiệu âm thanh, các tham số trong các mô hình học sâu

5.3 Hướng phát triển trong tương lai

- Tiếp tục tìm hiểu, nghiên cứu, phát triển, tối ưu mô hình hiện tại
- Thu thập thêm dữ liệu từ nhiều nguồn khác nhau, đặc biệt là các audio giọng miền trong (miền Nam, miền Trung)
- Cải tiến mô hình, thử nghiệm các kiến trúc đơn giản hơn để giảm thời gian dự đoán cho một từ khoá mà vẫn đảm bảo được yêu cầu về độ chính xác
- Tìm hiểu thêm các phương pháp xử lý, trích xuất đặc trưng tín hiệu âm thanh khác nhằm nâng cao độ chính xác của mô hình nhận diện
- Tìm hiểu thêm các phương pháp khử nhiễu, cũng như phát hiện tiếng nói một cách chính xác hơn nhằm tiết kiệm thời gian tổng hợp dữ liệu âm thanh trong quá trình ghi âm
- Thử nghiệm mô hình nhận diện phát âm từ khoá trên nhiều loại thiết bị thông minh tài nguyên thấp khác nhau

5.4 Kết luận

Như vậy, em đã trình bày xong đồ án tốt nghiệp "Nhận diện phát âm từ khoá trên thiết bị". Với những kết quả đạt được của mô hình đề xuất, vẫn còn rất nhiều vấn đề cần phải giải quyết. Tuy nhiên, do hạn chế về mặt thời gian cũng như thiếu kinh nghiệm thực tế trong các dự án học máy, học sâu, đồ án sẽ khó tránh khỏi những thiếu xót trong quá trình thực hiện. Vì thế, em rất mong nhận được nhiều sự góp ý chân thành từ các quý thầy cô để em có thể hoàn thiện đồ án của mình hơn nữa.

TÀI LIỆU THAM KHẢO

[1] NEW JETSON NANO 2GB DEVELOPER KIT, detailed at https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano/education-projects/.

[2] Z. Wang, X. Li, J. Zhou, Small-footprint Keyword Spotting Using Deep Neural Network and Connectionist Temporal Classifier, cite as arXiv:1709.03665v1, China, 2017.

[3] Anastasia Kyrykovych, Listlink, Deep Neural Networks, Available at https://www.kdnuggets.com/2020/02/deep-neural-networks.html, 2020.

[4] Nguyễn Lê Huy [Deep learning], Giới thiệu về Connectionist Temporal Classification (CTC) (Phần 1).

[5] Nguyễn Trung Thành [Deep learning] Feature Extraction - MFCC cho xử lý tiếng nói Available at https://viblo.asia/p/feature-extraction-mfcc-cho-xu-ly-tieng-noi-4dbZN2xmZYM.

[6] Vũ Hữu Tiệp, Machine Learning cơ bản - Bài 33: Các phương pháp đánh giá một hệ thống phân lớp, Available at https://machinelearningcoban.com/2017/08/31/evaluation.

[7] T. N. Sainath and C. Parada, "Convolutional neural networks for small-footprint keyword spotting," in Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH), New York, 2015.

[8] Pham Van Chung, [Deep Learning] Tìm hiểu về mạng tích chập (CNN), Available at https://viblo.asia/p/deep-learning-tim-hieu-ve-mang-tich-chap-cnn-maGK73bOKj2.

[9] R. Tang and J. Lin, “Deep residual learning for small-footprint keyword spotting,” in Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2018.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Deep Residual Learning for Image Recognition, cite as arXiv:1512.03385, 2015.

[11] Google AI Blog, Launching the Speech Commands Dataset, Available at https://ai.googleblog.com/2017/08/launching-speech-commands-dataset.html, 2017.

[12] Quoc Pham, Tìm Hiểu và Áp Dụng Cơ Chế Attention - Understanding Attention Mechanism, Available at https://pbcquoc.github.io/attention/.

[13] Axel Berg, Mark O'Connor, Miguel Tairum Cruz, Keyword Transformer: A Self-Attention Model for Keyword Spotting, cite as arXiv:2104.00769v2.

[14] Phan Duy Hung, Truong Minh Giang, Le Hoang Nam, Phan Minh Duong,

Vietnamese Speech Command Recognition using Recurrent Neural Networks, International Journal of Advanced Computer Science and Applications(IJACSA), 2019.

[15] Nguyen Truong Long, Giải thích chi tiết về mạng Long Short-Term Memory (LSTM), Available at https://nguyentruonglong.net/giai-thich-chi-tiet-ve-mang-long-short-term-memory-lstm.html, 2018.

[16] Nguyen Huu Binh, Nguyen Quoc Cuong, Tran Thi Anh Xuan, AN EVALUATION OF SOME FACTORS AFFECTING ACCURACY OF THE VIETNAMESE KEYWORD SPOTTING SYSTEM, Available at https://online.jmst.info/index.php/jmst/article/view/116, 2020.

[17] Seungwoo Choi, Seokjun Seo, Beomjun Shin, Hyeongmin Byun, Martin Kersner, Beomsu Kim, Dongyoung Kim, Sungjoo Ha, Temporal Convolution for Real-time Keyword Spotting on Mobile Devices, cite as arXiv:1904.03814v2, 2019.

[18] Vũ Hữu Tiệp, Machine Learning cơ bản - Bài 19: Support Vector Machine, Available at https://machinelearningcoban.com/2017/04/09/smv/.

[19] Do Duong, Recurrent Neural Network (Phần 1): Tổng quan và ứng dụng, Available at https://viblo.asia/p/recurrent-neural-networkphan-1-tong-quan-va-ung-dung-jvElaB4m5kw.

[20] Robert Keim, The Nyquist–Shannon Theorem: Understanding Sampled Systems, Available at https://www.allaboutcircuits.com/technical-articles/nyquist-shannon-theorem-understanding-sampled-systems/, 2020.

[21] Valerio Velardo - The Sound of AI, Demystifying the Fourier Transform: The Intuition, Complex Numbers for Audio Signal Processing, Defining the Fourier Transform with Complex Numbers, Discrete Fourier Transform Explained Easily, Available at https://www.youtube.com/playlist?list=PL-wATfeyA, 2020.

[22] Audio-Technica, Các đặc trưng của âm thanh, Available at https://audiotechnicashop.vn/cac-dac-trung-cua-am-thanh/, 2016.

[23] Nguyễn Trung Thành [Deep Learning], Kiến thức nền tảng xử lý tiếng nói, Available at https://viblo.asia/p/kien-thuc-nen-tang-xu-ly-tieng-noi-speech-processing-jvElaAL6lkw, 2020.

[24] Nguyễn Lê Huy, Sơ lược về Mel Frequency Cepstral Coefficients (MFCCs), Available at https://viblo.asia/p/so-luoc-ve-mel-frequency-cepstral-coefficients-mfccs-1VgZv1m2KAw.

[25] Đại học quốc gia Hà Nội, trường đại học công nghệ, Báo cáo tiểu luận, Môn Khai phá dữ liệu, Available at http://uet.vnu.edu.vn/~thuyhq/PPNCKH/.

[26] TopDev.vn, Thuật toán CNN – Convolutional Neural Network, Available at https://topdev.vn/blog/thuat-toan-cnn-convolutional-neural-network/.

[27] Dive into deep learning, Chapter 4, session 4.7: Forward Propagation, Backward Propagation, and Computational Graphs, Available at https://d2l.ai/chapter_multilayer-perceptrons/backprop.html.

[28] Trí tuệ nhân tạo, Blog chia sẻ về Trí tuệ nhân tạo, Seq2Seq – Hiểu về mô hình Encoder-Decoder, Available at https://trituenhantao.io/kien-thuc/hieu-ve-mo-hinh-encoder-decoder-seq2seq/, 2019.

[29] Vũ Hữu Tiệp, Machine Learning cơ bản - Bài 13: Softmax Regression, Available at https://machinelearningcoban.com/2017/02/17/softmax/.

[30] DEGREE PROJECT IN ELECTRICAL ENGINEERING, 30 CREDITS, Pitch-shifting algorithm design and applications in music, 2019.

[31] Oleg Rybakov, Natasha Kononenko, Niranjan Subrahmanya, Mirko Visontai, Stella Laurenzo, Streaming keyword spotting on mobile devices, cite as arXiv:2005.06720v2, 2020.

[32] Google, TensorFlow Lite tutorial, Available at https://www.tensorflow.org/lite/guide.

[33] Voice Activity Detection in Noise Using Deep Learning, Available at https://www.mathworks.com/help/audio/ug/voice-activity-detection-in-noise-using-deep-learning.html.