**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

# ĐỒ ÁN TỐT NGHIỆP

# Thử nghiệm nhận dạng tiếng nói tiếng Việt cho người khuyết tật giọng nói bằng phương pháp học sâu

**PHAN XUÂN PHÚC**
phuc.px156248@sis.hust.edu.vn

**Giảng viên hướng dẫn:** TS. Nguyễn Hồng Quang

Chữ ký của GVHD

**Bộ môn:** Kỹ thuật máy tính
**Viện:** Công nghệ thông tin và Truyền thông

**HÀ NỘI, 12/2019**

# PHIẾU GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

## 1. Thông tin về sinh viên

Họ và tên sinh viên: Phan Xuân Phúc

Điện thoại liên lạc: 0399150374 Email: *phanxuanphucnd@gmail.com*

Lớp: CN-CNTT 1 K60 Hệ đào tạo: Cử nhân công nghệ

Đồ án tốt nghiệp được thực hiện tại: Viện Công nghệ thông tin và Truyền thông - Trường đại học Bách Khoa Hà Nội

Thời gian làm ĐATN: từ 9/2019 đến 27/12/2019.

## 2. Mục đích nội dung của ĐATN

Xây dựng, thử nghiệm nhận dạng tiếng nói tiếng Việt cho người khuyết tật giọng nói bằng phương pháp học sâu.

## 3. Các nhiệm vụ cụ thể của ĐATN

- Tìm hiểu tổng quan xử lý tiếng nói;
- Tìm hiểu mạng nơ-ron nhân tạo, mạng nơ-ron tích chập và mạng hồi quy;
- Tìm hiểu, nghiên cứu các mô hình học sâu đã đạt hiệu quả tốt cho nhận dạng tiếng nói;
- Ứng dụng thử nghiệm vào bài toán nhận dạng tiếng nói tiếng Việt cho người khuyết tật giọng nói.

## 4. Lời cam đoan của sinh viên

Tôi - Phan Xuân Phúc – cam kết ĐATN là công trình nghiên cứu của bản thân tôi dưới sự hướng dẫn của TS. Nguyễn Hồng Quang.

Các kết quả nêu trong ĐATN là trung thực, không phải sao chép toàn văn của bất kỳ công trình nghiên cứu nào khác.

Hà Nội, ngày 27 tháng 12 năm 2019

Tác giả đồ án tốt nghiệp

Phan Xuân Phúc

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

Hà Nội, ngày 27 tháng 12 năm 2019

Giáo viên hướng dẫn

TS. Nguyễn Hồng Quang

## Lời cảm ơn

Lời cảm ơn chân thành em xin được gửi đến các thầy cô trường Đại học Bách Khoa Hà Nội và đặc biệt các thầy cô trong Viện Công nghệ thông tin và Truyền thông. Trong hơn bốn năm qua, em đã học được không những rất nhiều kiến thức mà còn trang bị cho em những hành trang vững chắc bước đi sau này. Qua những thời gian học tập, sống với Bách Khoa đã giúp em trở nên không ngại khó khăn thử thách, luôn sẵn sàng với tinh thần chiến đấu, học hỏi, kiên trì và không ngừng theo đuổi ước mơ, đam mê của bản thân.

Đặt biệt nhất, em xin gửi lời cảm ơn chân thành nhất tới thầy Nguyễn Hồng Quang, thầy đã giảng dạy trong quá trình học tập các học phần tại trường và hướng dẫn, giúp đỡ em hoàn thành ĐATN. Dẫu còn nhiều thiếu sót nhưng dưới sự giúp đỡ tận tình của thầy giúp em có động lực hoàn thành đề tài "Thử nghiệm xây dựng hệ nhận dạng tiếng nói tiếng Việt cho người khuyết tật giọng nói bằng phương pháp học sâu" một cách tốt nhất. Ngoài ra, em cũng xin cảm ơn đến công ty Ftech, đặc biệt các anh, các bạn trong team NLP đã giúp đỡ và tạo điều kiện để em hoàn thành tốt đồ án tốt nghiệp.

Cuối cùng, xin cảm ơn tới gia đình em đã hỗ trợ em trong việc thu thập dữ liệu và những người bạn luôn giúp đỡ nhau trong quá trình học tập tại trường. Em xin chân thành cảm ơn.

Sinh viên

Phan Xuân Phúc

## Tóm tắt nội dung đồ án

Đồ án này nhằm mục đích áp dụng các phương pháp học sâu thử nghiệm xây dựng hệ nhận dạng tiếng nói tiếng Việt hỗ trợ người khuyết tật giọng nói, giúp giải quyết các vấn đề khuyết điểm của người khuyết tật giọng nói ở Việt Nam.

Bố cục của đồ án gồm có 04 chương:

*Chương 1, Đặt vấn đề và hướng giải pháp*. Trong chương này sẽ trình bày vấn đề cần giải quyết, giới hạn phạm vi đề tài, hướng giải pháp cho vấn đề trong khuôn khổ đồ án.

*Chương 2, Một số kiến thức liên quan*. Trong chương này sẽ nói một số vấn đề xử lý dữ liệu âm thanh trong bài toán. Phần sau của chương trình bày các thuật toán sử dụng cho bài toán và một số vấn đề liên quan.

*Chương 3, Nhận dạng giọng nói của người khuyết tật giọng nói*. Nội dung của chương này sẽ trình bày cách thu thập dữ liệu, tiền xử lý dữ liệu đầu vào, vấn đề tăng cường dữ liệu huấn luyện và áp dụng mô hình vào bài toán. Phần sau đó của chương sẽ trình bày kết quả trên một số thử nghiệm cho bài toán.

*Chương 4, Kết luận và hướng phát triển*. Nội dung của chương sẽ trình bày về những điểm đã đạt được, những điểm còn chưa giải quyết được và hướng phát triển trong quá trình nghiên cứu tiếp theo.

Sinh viên thực hiện

# MỤC LỤC

# DANH MỤC HÌNH VẼ

# DANH MỤC BẢNG

# DANH MỤC TỪ VIẾT TẮT

| | |
|---|---|
| **API** | Application Programming Interface<br>Giao diện lập trình ứng dụng |
| **CER** | Character Error Rate<br>Tỷ lệ lỗi của ký tự |
| **CNN** | Convolutional Neural Network<br>Mạng rơ-ron tích chập |
| **CNTT** | Công nghệ thông tin |
| **CTC** | Connectionist Temporal Classification |
| **DNN** | Deep Neural Network<br>Mạng nơ-ron sâu |
| **DTW** | Dynamic Time Warping<br>Xoắn thời gian động |
| **DCT** | Discrete Cosine Transform<br>Biến đổi Cosin rời rạc |
| **DFT** | Discrete Fourier Transform<br>Biến đổi Fourier rời rạc |
| **ĐATN** | Đồ án tốt nghiệp |
| **HMM/GMM** | Hidden Markov Model/ Gaussian Mixed Model<br>Mô hình Makov ẩn / Mô hình Gaussian hỗn hợp |
| **FFT** | Fast Fourier Transform<br>Biến đổi Fourier nhanh |
| **GRU** | Gated Recurrent Unit |

| | |
|---|---|
| **MFCCs** | Mel Frequency Cepstral Coefficients |
| **RNN** | Recurrent Neural Network<br>Mạng hồi quy |
| **SER** | Sentence Error Rate<br>Tỷ lệ lỗi của câu |
| **WER** | Word Error Rate<br>Tỷ lệ lỗi của từ |

# DANH MỤC THUẬT NGỮ

| | |
|---|---|
| **Computer Vision** | Thị giác máy tính |
| **Deep Learning** | Học sâu |
| **Knowledege** | Tri thức |
| **File** | Tệp |
| **Nature Language Processing** | Xử lý ngôn ngữ tự nhiên |
| **Neural Network** | Mạng nơ-ron |
| **Network** | Mạng |
| **End-to-end** | Đầu cuối |
| **Vector** | Vec-tơ |
| **Smart phone** | Điện thoại thông minh |

# CHƯƠNG 1. ĐẶT VẤN ĐỀ VÀ HƯỚNG GIẢI PHÁP

## 1.1 Đặt vấn đề

Trong cuộc sống, tiếng nói là một ngôn ngữ giao tiếp cơ bản nhất của con người, có tầm quan trọng lớn trong đời sống hàng ngày và trong vấn đề phát triển kinh tế, xã hội. Vấn đề giao tiếp giữa con người với con người, hay giữa con người với máy cũng là một trong những đề tài nghiên cứu lớn của nhiều lĩnh vực và có những khó khăn nhất định nhưng mang lại nhiều ứng dụng thực tiễn.

Ở Việt Nam có tới 7% dân số (khoảng 6.2 triệu người) bị khuyết tật [1], trong đó có người khuyết tật giọng nói. Tồn tại những vấn đề cơ bản mà người khuyết tật giọng nói gặp khó khăn đó là trong giao tiếp trong cuộc sống chẳng hạn như nói chuyện với bạn bè, người thân; trao đổi nhóm trong công việc; hoặc thuyết trình trong một cuộc hội nghị, những công việc yêu cầu tiếng nói. Với những người khuyết tật giọng nói, họ gặp rất nhiều vấn đề hạn chế trong việc nói, phát âm ra lời nói nhưng người khác không hiểu, không nghe rõ và dẫn đến việc thiếu tự tin khi nói, giao tiếp với người khác, hay hạn chế trong việc tìm kiếm các công việc cho bản thân. Tôi cũng là một người trong số đó. Những điều đó làm cho bản thân chúng tôi cảm thấy bị kém may mắn, thường bị cô lập hơn với mọi người.

Trên thế giới cũng đã có rất nhiều công trình nghiên cứu áp dụng công nghệ vào việc hỗ trợ cho người khuyết tật và đạt được những thành tựu đáng kể. Tôi đã suy nghĩ và ước muốn về việc làm sao đó để khắc phục những khuyết điểm đó. Với mong muốn của bản thân, cùng với sự phát triển mạnh mẽ của khoa học kỹ thuật và công nghệ, tôi hướng tới việc áp dụng công nghệ vào việc xây dựng các hệ thống, công cụ cho người khuyết tật giọng nói. Bước đầu tiên trong kế hoạch đó, tôi nghiên cứu tìm hiểu trong đồ án này là nghiên cứu, thử nghiệm xây dựng một hệ nhận dạng tiếng nói tiếng Việt cho người khuyết tật giọng nói giúp nhận dạng được tiếng nói tiếng Việt của người khuyết tật giọng nói.

## 1.2 Phạm vi đề tài

Đồ án hướng tới việc xây dựng hệ nhận dạng tiếng nói tiếng Việt cho người khuyết tật về giọng nói. Cơ sở dữ liệu cho đồ án này gặp rất nhiều hạn chế trong việc thu thập dữ liệu để huấn luyện mô hình thử nghiệm. Do mang tính thử nghiệm và có nhiều hạn chế về thời gian, số lượng người được thu thập nên đồ án chỉ thực hiện trên bộ dữ liệu nhỏ (từ đơn, từ ghép) cho ngôn ngữ tiếng Việt.

## 1.3 Hướng giải pháp

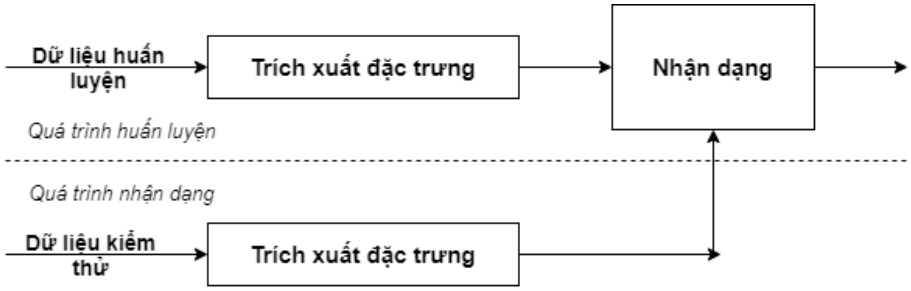
Nhận dạng tiếng nói cho tiếng Việt đã được rất nhiều nghiên cứu và được phát triển trong những năm qua bằng những phương pháp khác nhau. Đặng Đức Ngọc [2] đã giới thiệu sử dụng mạng rơ-ron và mô hình Markov ẩn cho nhận dạng tiếng nói tiếng Việt năm 2003. Năm 2008, Vũ Tất Thắng và cộng sự [3] đề xuất phương pháp nhận dạng thanh điệu sử dụng mạng nơ-ron percepton cho bài toán phức tạp đó là nhận dạng tiếng nói cho tiếng Việt với bộ từ vựng lớn. TS. Nguyễn Hồng Quang và cộng sự [4] đã tích hợp thông tin thanh điệu cho các âm vị sử dụng với bộ công cụ HTK vào năm 2010. Năm 2017, TS. Nguyễn Thị

Thanh và cộng sự [5] sử dụng bộ công cụ Kaldi cho nhận dạng tiếng nói tiếng Việt. Và rất nhiều công trình nghiên cứu mang lại giá trị cho bài toán nhận dạng tiếng nói tiếng Việt.

Với những ngôn ngữ khác trên thế giới, chẳng hạn như tiếng Anh, có rất nhiều hướng tiếp cận mang lại kết quả khả quan chẳng hạn như phương pháp tiếp cận dựa trên Knowledge [6], dựa trên Dynamic Time Warping [7], dựa trên mô hình HMM/GMM [8], và Neural Networks [9].

Trong những năm gần đây, với những thành công vượt trội của việc áp dụng phương pháp học sâu (Deep Learning) trong giải quyết các bài toán phức tạp trong thị giác máy tính (Computer Vision), xử lý ngôn ngữ tự nhiên (Natural Language Processing). Năm 2014 và 2015, hai mô hình end-to-end được giới thiệu cho nhận dạng tiếng nói, gọi là Deep Speech 1 [10] và Deep Speech 2 [11] được áp dụng trên ngôn ngữ tiếng Anh và Mandarin đã mang lại những hiệu quả vượt trội so với các phương pháp truyền thống khác, cũng giải quyết những thách thức trong hệ thống nhận dạng tiếng nói. Vì thế, việc áp dụng phương pháp học sâu là sự lựa chọn thử nghiệm tối ưu trong việc giải quyết bài toán về nhận dạng tiếng nói tiếng Việt cho người khuyết tật giọng nói.

Trong phạm vi đồ án, tôi thực hiện thử nghiệm áp dụng phương pháp học sâu, sử dụng một số mô hình như Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) dựa trên kiến trúc của mô hình Deep Speech 1 [10] và Deep Speech 2 [11] vào việc nhận dạng tiếng nói cho người khuyết tật giọng nói, so sánh hiệu quả của chúng trên một số thử nghiệm cơ bản được mô tả cụ thể ở phần 3.4. Kết quả thu được sẽ giúp đánh giá chất lượng của các kiến trúc mô hình qua độ đo Word Error Rate, Character Error Rate và Sentence Error Rate của bài toán.




*Hình 1.1 Tổng quan hệ trình huấn luyện và nhận dạng của hệ thống.*

Quá trình xây dựng hệ thống nhận dạng tiếng nói gồm 2 giai đoạn: huấn luyện và nhận dạng. Trong quá trình huấn luyện, các đặc trưng cho tín hiệu tiếng nói được trích xuất, bộ nhận dạng sẽ được huấn luyện để tối thiểu hàm chi phí. Trong quá trình nhận dạng, bộ nhận dạng đã được huấn luyện đưa ra bản phiên âm ứng với tín hiệu tiếng nói cần nhận dạng.

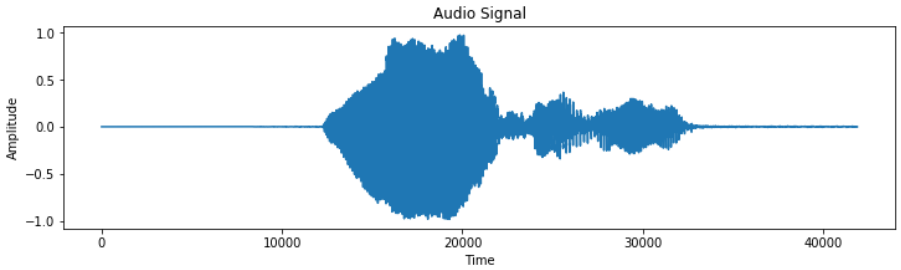
# CHƯƠNG 2. MỘT SỐ KIẾN THỨC LIÊN QUAN

## 2.1 Một số vấn đề xử lý dữ liệu âm thanh

### 2.1.1 Đọc dữ liệu từ file âm thanh

Chúng ta đều nghe nhạc/ âm thanh trên máy tính, điện thoại. Thông thường, chúng ở định dạng .mp3, .m4a, ... Nhưng các file định dạng đó không phải là âm thanh thực tế. Đó chỉ là một cách biểu diễn âm thanh trên máy tính, điện thoại và chúng ta không thể đọc trực tiếp từ chúng giống như đọc từ các file .txt, .csv, …

Âm thanh/ tiếng nói được thể hiện dưới dạng sóng, gồm 2 trục. Trục $x$ biểu diễn thời gian và trục $y$ biểu diễn biên độ. Tại mỗi thời điểm $t$, chúng ta có một giá trị biên độ xác định. Như ví dụ minh họa ở Hình 2.1 là một biểu diễn âm thanh của phát âm "mười sáu".




*Hình 2.1 Một sóng âm thanh của phát âm "mười sáu".*

Chúng ta sẽ phải sử dụng các file âm thanh dạng sóng có định dạng .wav để đọc chúng. Trong Python, thư viện Librosa [2] hỗ trợ cho phép chúng ta đọc các file .wav. Sau khi đọc từ file .wav chúng ta thu được một loạt con số như ví dụ dưới đây đọc một file âm thanh có độ dài là 1 giây và tỷ lệ lấy mẫu là 16000 Hz của phát âm *"mười sáu"*:

array([ 0.06896514, 0.1007529 , 0.08946026, ..., -0.09958471, -0.10146017, -0.11512566], dtype=float32)

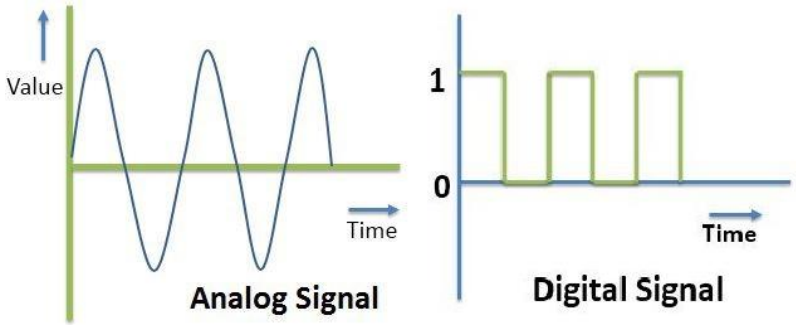
Các giá trị này biểu diễn cho trục $y$ của sóng, tức là biên độ. Còn trục $x$ chính là chiều dài của mảng. Trong trường hợp này sẽ là 16000.

### 2.1.2 Tỷ lệ lấy mẫu (Sampling rate)

Tỷ lệ lấy mẫu là số lượng mẫu được thu thập trong 1 giây. Nó ở dạng digital [12], là một chuỗi các giá trị rời rạc, và được thu thập trong khoảng cách thời gian bằng nhau. Như ví dụ trong phần 2.1.1 thì tỷ lệ lấy mẫu là 16000 Hz, tức là thu thập 16000 mẫu trên 1 giây. Khi tỷ lệ lấy mẫu càng cao thì độ mất mát thông tin càng nhỏ, nhưng đó là việc đánh đổi cần thiết trong việc lưu trữ dữ liệu âm thanh.

Thật vậy, ta xét một đoạn sóng âm thanh có độ dài là 1 giây. Nếu nó ở dạng analog [13], tức là nó có các giá trị biên độ cho mọi thời điểm, mỗi giá trị tương ứng với mỗi nano giây hoặc có thể là pico giây. Giả sử, đoạn sóng âm thanh 1 giây nói trên có mỗi giá trị cho mỗi pico giây, tức là sẽ có $1e + 12$ giá trị. Việc lưu trữ trong máy tính, đối với kiểu dữ liệu float mất 4 bytes để lưu trữ. Với đoạn

âm thanh trên phải mất $1e + 12 * 4$ bytes, tức là hơn 3.5 terabyes chỉ để lưu một file âm thanh có độ dài là 1 giây. Vì thể, chúng ta phải chuyển đổi các sóng âm thanh thành dạng digital. Hình 2.2 minh họa sóng ở hai dạng Analog và Digital.




*Hình 2.2 Minh họa sóng dưới dạng Analog và Digital. Nguồn [14]*

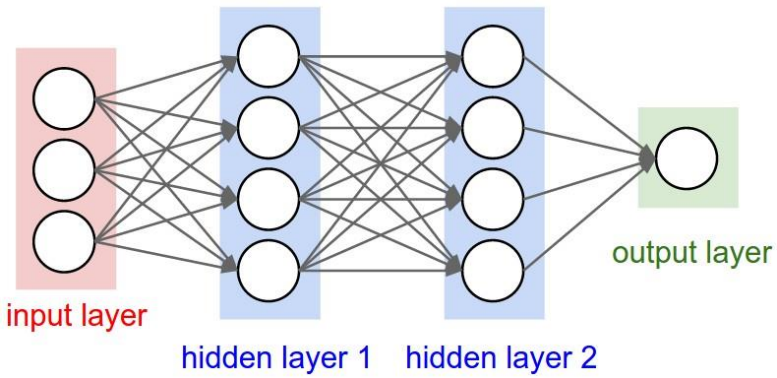
Thông thường, tỷ lệ lấy mẫu mặc định là 22050 bởi con người có thể nghe tần số từ 20 Hz – 20k Hz. Trong bài toán nhận dạng giọng nói, việc tỷ lệ lấy mẫu thông thường là 16000 Hz là đủ.

## 2.2 Mạng nơ-ron (Neural Network)

### 2.2.1 Giới thiệu tổng quan

Neural là tính từ của neuron (nơ-ron) chỉ bộ phận tế bào của hệ thần kinh có tính chất liên kết nhau. Chúng là thành phần quan trọng của bộ não giúp chúng ta có nhận thức, nhận biết. Network là mạng cấu trúc đồ thị. Vì thế neural network là hệ thống tính toán lấy cảm hứng từ sự hoạt động của các nơ-ron trong hệ thần kinh.

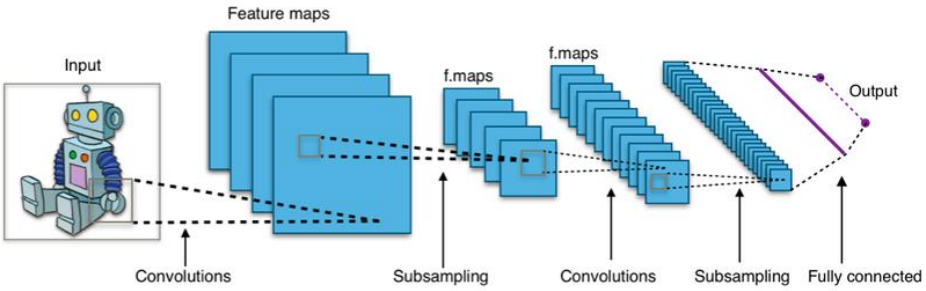
Kiến trúc chung của một mạng rơ-ron gồm nhiều nơ-ron, gọi là node, chúng được liên kết với nhau. Về cơ bản, mạng nơ-ron chia thành 3 lớp chính: lớp đầu tiên là lớp đầu vào (input layer), các lớp ở giữa gọi là lớp ẩn (hidden layer) và lớp cuối cùng được gọi là lớp đầu ra (output layer). Hình 2.3 minh họa một mạng nơ-ron cơ bản gồm 2 lớp ẩn.




*Hình 2.3 Ví dụ về mạng neural network, 2 hidden layer. Nguồn [15]*

Một mạng nơ-ron chỉ có 1 lớp đầu vào, 1 lớp đầu ra nhưng có thể có rất nhiều lớp ẩn. Trong mạng nơ-ron, mỗi node là một nơ-ron có hàm kích hoạt [29] có thể khác nhau. Tuy nhiên, trong thực tế người ta thường áp dụng chúng chung một hàm kích hoạt mục đích cho việc tối thiểu hóa độ phức tạp tính toán.

### 2.2.2 Mô hình mạng CNN (Convolutional Neural Network)




*Hình 2.4 Mô hình CNN. Nguồn [16]*

Về cơ bản mạng tích chập (Convolutional neural network) là mạng nơ-ron truyền thẳng, trong đó kiến trúc gồm nhiều thành phần được ghép nối với nhau theo câu cấu trúc: lớp tích chập (Convolutional), Pooling, ReLU và Fully Connected.

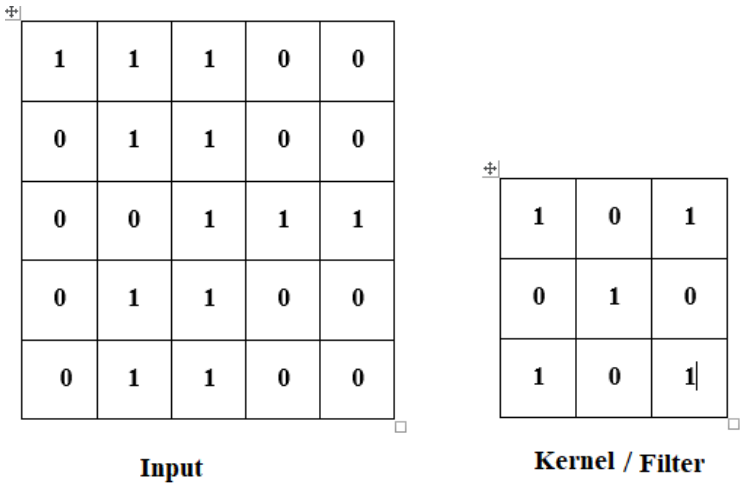
#### *2.2.2.1. Lớp tích chập (Convolutional Layer)*

Phép tích chập được thực hiện bằng phép toán giữa hai hàm đã có ($f$ và $g$). Công thức phép toán tích chập như sau:

$$k(x,y) \times f(x,y) = \sum_{u=-\frac{m}{2}}^{\frac{m}{2}} \sum_{u=-\frac{n}{2}}^{\frac{n}{2}} k(u,v) f(x-u, y-v)$$

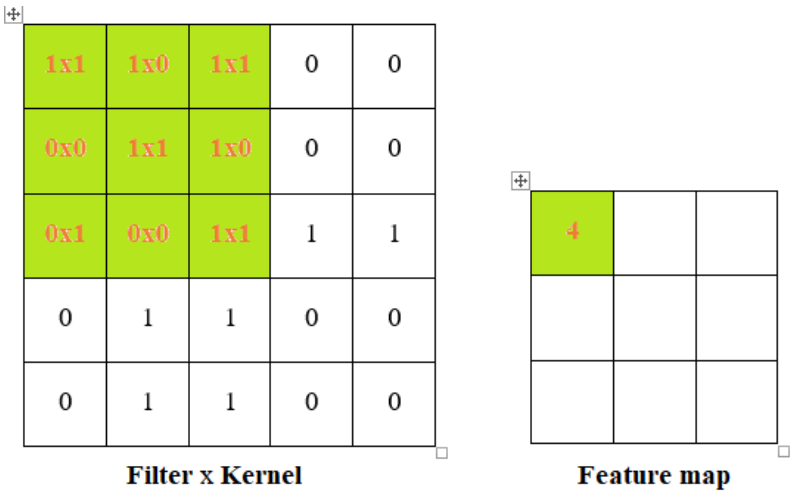
Trong CNN, lớp tích chập chính là hidden layer, nhưng lớp tích chập là một tập các *feature map* và mỗi *feature map* này là một bản scan (quét) (hay còn hiểu là một góc nhìn) của đầu vào ban đầu, được trích xuất ra thành các đặc trưng cụ thể. Việc thực hiện phụ thuộc vào filter (bộ lọc) và kernel (nhân). Nó thực hiện việc quét ma trận dữ liệu đầu vào theo trình tự từ trái qua phải, rồi sau đó từ trên xuống dưới và nhân tương ứng từng giá trị của ma trận đầu với ma trận kernel rồi cộng tổng lại. Sau đó, nó đưa qua hàm kích hoạt (activation function) (chẳng hạn như, sigmoid, relu, …), thu được một con số cụ thể. Sau quá trình thực hiện xong ta thu được một ma trận, nó chính là *feature map*.

Hình 2.5 minh họa một ma trận đầu vào kích thước $5 \times 5$ và một filter có kích thước $3 \times 3$.




*Hình 2.5 Ma trận đầu vào và kernel.*

Sau khi thực hiện việc quét kernel qua từng phần từ của input, đồng thời thực hiện các phép tính toán như nói trên ta thu được giá trị tại *feature map*. Hình 2.6, minh họa ma trận $Input \times Filter$ và giá trị tại $Feature\ map$.




*Hình 2.6 Ma trận Input × Kernel và Feature map.*

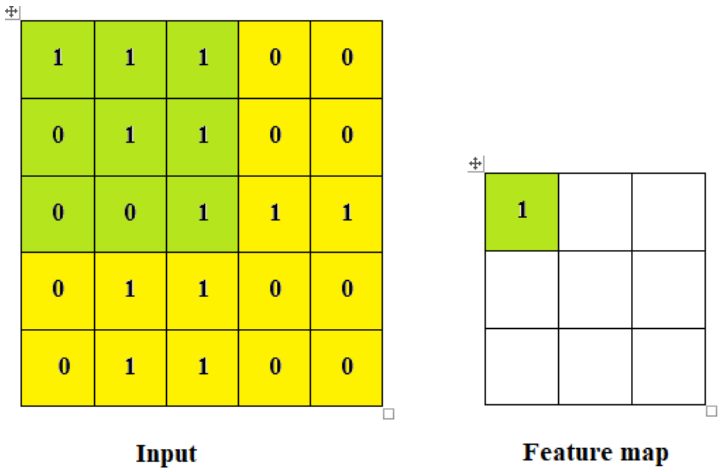
Bộ lọc (filter) hay còn gọi là bộ phát hiện đặc trưng được coi là nơ-ron của lớp tích chập. Nó có đầu vào là các trọng số của input và giá trị đầu ra như một nơ-ron. Nếu có một lớp tích chập thì đầu vào sẽ là các giá trị đầu vào ban đầu, nếu kiến trúc mạng có chiều sâu hơn thì lớp tích chập sẽ lấy đầu vào từ *feature map* trước đó.

Tổng quát hóa, khi ta nhân tích chập ma trận input kích thước $n \times n$ vào bộ lọc kích thước $f \times f$, ta thu được một ma trận kích thước:

$$(n - f + 1) \times (n - f + 1).$$

Trong lớp tích chập còn có hai tham số quan trọng khác đó là *Stride* và *Padding*.

*Stride* là khoảng cách giữa hai kernel khi thực hiện quét. Ví dụ, khi stride=1, kernel sẽ quét hai cạnh nhau (tức là ô số 1 rồi đến ô số 2). Còn khi stride=2, kernel sẽ quét ô số 1 rồi đến ô số 3, và bỏ qua ô số 2. Hình 2.7 minh họa ví dụ với stride=1.



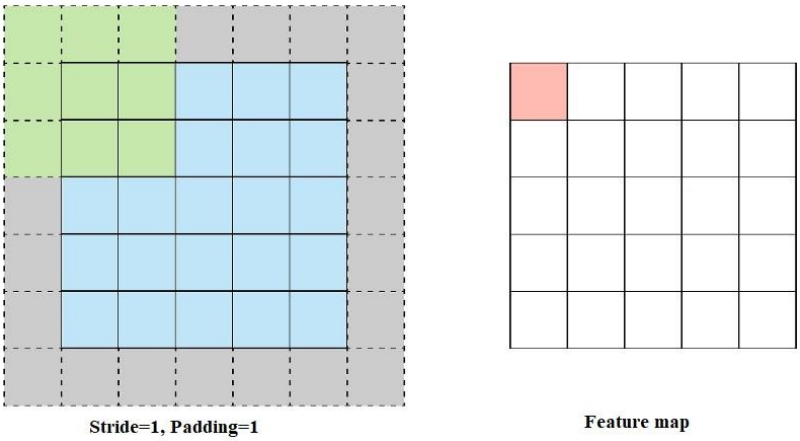

*Hình 2.7 Ví dụ với stride=1.*

Tổng quát hóa, kích thước ma trận đầu ra lúc này với stride=$s$ được tính bởi:

$$(\frac{n-f}{s}+1)\times(\frac{n-f}{s}+1)$$

Nếu $n-f$ không chia hết cho $s$, khi đó ta có thể thực hiện việc lấy chặn dưới.

Mỗi một lần sử dụng phép tích chập, kích thước feature map bị giảm xuống. Chúng ta có thể sử dụng *padding* để giữ nguyên kích cỡ feature map so với ban đầu. Khi điều chỉnh padding=$p$, tức là ta đã thêm $p$ ô bao bọc xung quanh các cạnh của ma trận input. Hình 2.8 minh họa ví dụ với padding=1.




*Hình 2.8 Ví dụ với stride=1 và padding=1.*

*Padding* còn giúp tránh việc mất mát thông tin tại các vùng gần cạnh của ma trận input, bởi vì vùng trung tâm của ma trận input được bao phủ bởi rất nhiều vùng kích thước bộ lọc, trong khi những vùng ở góc hoặc cạnh chỉ được bao phủ 1 hoặc 2 lần.

Tổng quát hóa, ma trận feature map khi sử dụng với padding=$p$, stride=$s$ được tính bởi:

$$(\frac{n-f+2p}{s}+1)\times(\frac{n-f+2p}{s}+1)$$

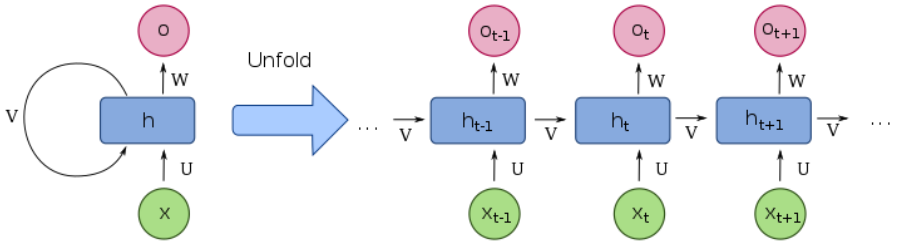
*2.2.2.2. Lớp Fully-Connected*

Lớp Fully-connected là lớp mà các nơ-ron liên kết đầy đủ với nhau. Lớp này cũng tương tự như một fully-connected trong mạng nơ-ron nhân tạo. Trong kiến trúc mô hình tôi sử dụng cho bài toán sử dụng 1 đến 2 lớp fully-connected với hàm kích hoạt ReLU hoặc Softmax với lớp đầu ra có số unit bằng với số ký tự đầu ra mong muốn.

### 2.2.3 Mô hình mạng RNN (Recurrent Neural Network)

Ý tưởng chính của mạng RNN là sử dụng chuỗi thông tin. Trong các mạng nơ-ron truyền thống tất cả các đầu vào và đầu ra độc lập với nhau. Tức là chúng không liên kết thành chuỗi với nhau. Với các mô hình này không phù hợp với nhiều bài toạn dạng tuần tự. Ví dụ như, nếu muốn đoán một từ tiếp theo xuất hiện trong một câu thì ta cần biết các từ phía trước nó xuất hiện như thế nào. RNN sinh ra để giải quyết vấn đề đó.

RNN xem dữ liệu đầu vào là một chuỗi (sequence) liên tục, nối tiếp nhau theo thứ tự thời gian. Ví dụ như một đoạn text (văn bản) là một chuỗi các từ vựng (words) hoặc một chuỗi các ký tự (characters). Tại thời điểm $t$, với dữ liệu đầu vào $x_t$ ta có kết quả output là $y_t$. Khác với mạng lan truyền thẳng, $y_t$ được sử dụng làm đầu vào cho thời điểm $(t+1)$. Điều này cho phép RNN có thể lưu trữ và truyền thông tin đến thời điểm tiếp theo. Hình 2.9 minh họa ví dụ về mô hình RNN.




*Hình 2.9 Mô hình mạng Recurrent Neural Network. Nguồn [27]*

Việc tính toán bên trong mạng RNN được thực thi như sau:

- $x_t$ là đầu vào tại thời điểm $t$.
- $h_t$ là trạng thái ẩn ở thời điểm $t$. Nó chính là bộ nhớ của mạng và được tính toán dựa trên các trạng thái ẩn trước và đầu vào tại bước đó: $h_t = f(Ux_t + Vh_{t-1})$. Hàm $f$ thường là một hàm phi tuyến chẳng hạng như *tanh, ReLU, …*
- $o_t$ là đầu ra tại thời điểm $t$. Công thức tính: $o_t = g(Wh_t)$.

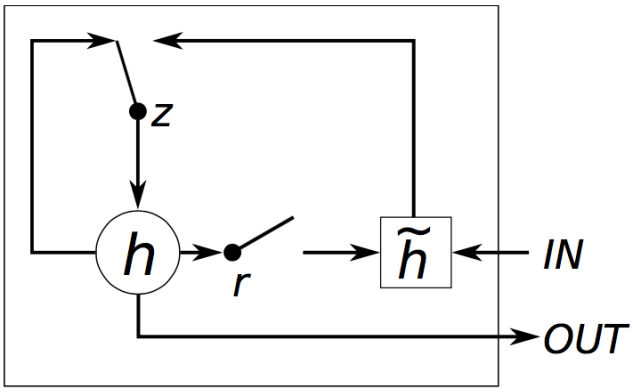
Về mặt lý thuyết, RNN có thể xử lý và lưu thông tin của chuỗi với độ dài bất kỳ. Nhưng trong thực tế, RNN chỉ hiệu quả khi chuỗi dữ liệu ngắn, có độ lớn không quá dài). Nguyên nhân của vấn đề này là do hiện tượng triệt tiêu gradient (vanishing gradient). Để khắc phục những vấn đề này người ta đã xây dựng một

số cấu trúc biến thể của mạng RNN là LSTM, GRU. Viêc áp dụng hai kiến trúc mô hình này giúp hạn chế vấn đề phụ thuộc xa của mạng RNN.

Một biến thể của mạng RNN có tên gọi là Gated Recurrent Unit (GRU) mà tôi sử dụng trong đồ án. Về cơ bản, mạng GRU có cấu trúc như RNN thuần nhưng khác nhau ở cách tính toán các trạng thái ẩn. Công thức tính toán bên trong mạng GRU như sau:

$$z = \sigma(x_t U^z + h_{t-1} V^z)$$
$$r = \sigma(x_t U^r + h_{t-1} V^r)$$
$$g = \tanh(x_t U^g + h_{t-1} \circ r)\, V^h)$$
$$h_t = (1 - z) \circ g + z \circ h_{t-1}$$

GRU chỉ có 2 cổng: cổng thiết lập lại $r$ và cổng cập nhập $z$. Cổng thiết lập lại sẽ quyết định cách kết hợp giữa đầu vào hiện tại với bộ nhớ trước, còn cổng cập nhập sẽ chỉ định có bao nhiêu thông tin về bộ nhớ trước nên giữa lại. Như vậy RNN thuần cũng là một dạng đặc biệt của GRU, với đầu ra của cổng thiết lập lại là 1 và cổng cập nhập là 0. Hình 2.10 mô tả kiến trúc mô hình mạng GRU.




*Hình 2.10 Cấu trúc mạng GRU. Nguồn [28]*

### 2.2.4 Một số vấn đề cần lưu ý

*2.2.4.1. Vấn đề triệt tiêu/ bùng nổ gradient (Vanishing /exploding gradient)*

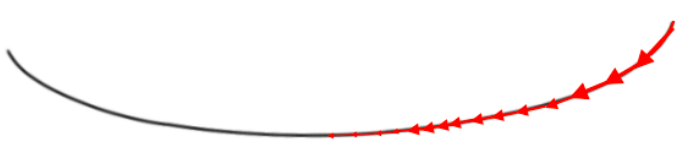
Giải thuật mang tính cốt lõi giúp việc huấn luyện các mô hình học sâu có thể dễ dàng thực thi tính toán và cập nhật trọng số là giải thuật lan truyền ngược (*Backpropagation*) [17]. Ý tưởng chung của thuật toán là sẽ đi từ output layer (lớp đầu ra) đến input layer (lớp đầu vào) và tính toán gradient của cost function (hàm chi phí) tương ứng cho từng parameter weights (tham số trọng số) của mạng. Sau đó, *Gradient Descent* (GD) [30] sẽ được sử dụng để cập nhật các tham số đó.

Quá trình trên sẽ được lặp lại cho tới khi các parameter của mạng hội tụ. Thông thường chúng ta sẽ có một hyperparameter định nghĩa cho số lượng vòng lặp để thực hiện quá trình trên. Hyperparameter đó thường được gọi là số Epoch (tức là

số lần mà training set được duyệt qua một lần và weights được cập nhật). Nếu số lượng vòng lặp quá nhỏ, DNN (Deep Neural Network) có thể sẽ không cho ra kết quả tốt, và ngược lại thì thời gian huấn luyện sẽ quá dài nếu số lượng vòng lặp quá lớn. Ở đây ta có một đánh đổi giữa độ chính xác và thời gian huấn luyện.

Tuy nhiên trên thực tế gradients thường sẽ có giá trị nhỏ dần khi đi xuống các layer thấp hơn. Kết quả là các cập nhật thực hiện bởi Gradient Descent không làm thay đổi nhiều weights của các layer đó, khiến chúng không thể hội tụ và DNN sẽ không thu được kết quả tốt. Hiện tượng này được gọi là *Vanishing Gradients*.

Trong hình 2.9 minh họa cost function có dạng đường cong dẹt, chúng ta sẽ cần khá nhiều lần cập nhật (Gradient Descent step) để tìm được điểm global minimum (điểm cực tiểu toàn cục).



*Hình 2.11 Minh họa quá trình cập nhật gradient.*

Trong nhiều trường hợp khác, gradients có thể có giá trị lớn hơn trong quá trình backpropagation, khiến một số layers có giá trị cập nhật cho weights quá lớn khiến chúng phân kỳ (phân rã), tất nhiên DNN cũng sẽ không có kết quả như mong muốn. Hiện tượng này được gọi là *Exploding Gradients*, và thường gặp khi sử dụng RNNs.

Trong quá trình huấn luyện DNN, chúng ta có thể gặp phải các vấn đề liên quan đến việc gradients không ổn định khiến cho tốc độ học của các layer khác nhau chênh lệch khá nhiều. Hai hiện tượng trên là một trong những nguyên nhân khiến mạng nơ-ron không nhận được sự quan tâm trong một thời gian khá dài. Tuy nhiên trong một nghiên cứu được thực hiện bởi Xavier Glorot và Yoshua Bengio năm 2010 [18]. Các tác giả đã đưa ra một số nguyên nhân dẫn đến hiện tượng trên. Trong đó việc lựa chọn activation function và kỹ thuật weight initialization là hai nguyên nhân chính.

Tóm lại, hai vấn đề vanishing/ exploding gradient khiến cho quá trình huấn luyện trở nên khó khăn và hiệu suất của mô hình cũng bị giảm. Để tránh được các vấn đề này, mô hình cần sử dụng các hàm kích hoạt có tính chất hạn chế nhược điểm của hai vấn đề trên trong quá trình backpropagation. Ví dụ như ReLU, Leak ReLU, … Bên cạnh đó việc áp dụng các kỹ thuật như Batch Normalization cũng là một cách hạn chế vấn đề trên.

#### *2.2.4.2. Tiêu chí đánh giá mô hình*

Có rất phương pháp tiếp cận cho bài toán nhận dạng tiếng nói tiếng Việt trong nhiều năm qua. Việc so sánh, đánh giá các mô hình dựa trên các tiêu chí sau:

- **Độ đo** *(Metrics)*

Độ đo là các con số để đánh giá khả năng của mô hình để biết tỷ lệ chính xác của mô hình. Trong nhận dạng tiếng nói người ta hay sử dụng độ đo là tỷ lệ lỗi của từ (WER). Ngoài ra, họ cũng sử dụng các độ đo tỷ lệ lỗi của chữ cái (CER) và tỷ lệ lỗi của câu (SER).

- **Tốc độ** *(Speed)*

Tốc độ là chi phí tính toán liên quan đến quá trình tạo ra, sử dụng mô hình.

- **Sức mạnh** *(Robustness)*

Sức mạnh là khả năng mà mô hình đưa ra các dự đoán đúng với các dữ liệu chứa nhiễu và dữ liệu bị mất mát.

- **Khả năng mở rộng** *(Scalability)*

Khả năng mở rộng là khả năng mà mô hình được xây dựng hiệu quả với các miền dữ liệu khác nhau.

- **Tính hiểu được** *(Interpretability)*

Tính hiểu được là mức độ chi tiết của thông tin được cung cấp bởi mô hình.

- **Tính đơn giản** *(Simplicity)*

Tính đơn giản là mức độ đơn giản, kích thước của mô hình.

# CHƯƠNG 3. NHẬN DẠNG TIẾNG NÓI TIẾNG VIỆT CHO NGƯỜI KHUYẾT TẬT GIỌNG NÓI

## 3.1 Dữ liệu cho bài toán

Tập dữ liệu huấn luyện bao gồm 1600 từ vựng thông dụng nhất trong tiếng việt được thu thập trên Wikipedia [23]. Ý tưởng của tôi là thu thập một bộ dữ liệu nhỏ cho việc thử nghiệm, sao cho chúng bao phủ toàn bộ âm vị và chữ cái bao gồm cả thanh điệu trong tiếng Việt. Bộ dữ liệu này sẽ là chủ yếu là từ vựng đơn có chứa 134 âm vị và chứa đủ 89 chữ cái bao gồm các chữ cái trong tiếng Việt tính cả thanh điệu. Trong tiếng Việt có 6 thanh điệu đó là: thanh ngang (-), thanh huyền ( ` ), thanh ngã ( ~ ), thanh hỏi ( ̉ ), thanh sắc ( ′ ) và thanh nặng ( . ).

Bảng chữ cái tiếng Việt có 29 chữ cái, theo thứ tự:

[a, ă, â, b, c, d, đ, e, ê, g, h, i, k, l, m, n, o, ô, ơ, p, q, r, s, t, u, ư, v, x, y]

Trong đó có 12 chữ cái có thanh điệu bao gồm:

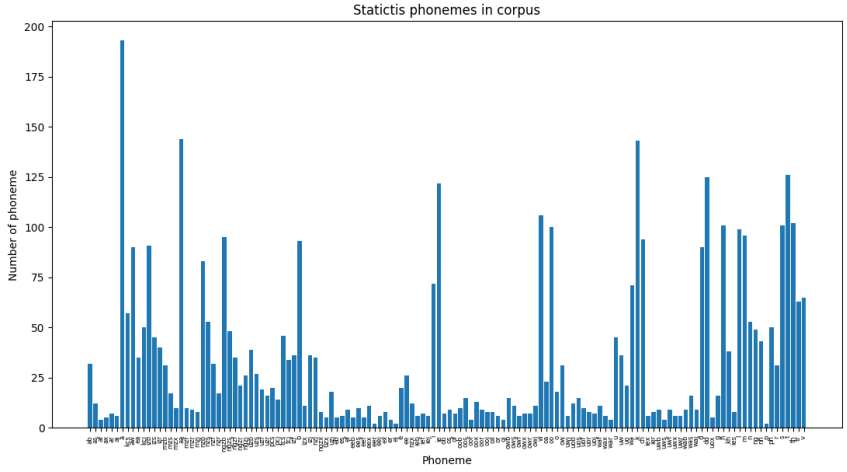
[a, ă, â, e, ê, i, o, ô, ơ, u, ư, y]

Bảng 3.1 minh họa phân tích âm vị của các biến thể thanh điệu khi phát âm “a” và “đa”.

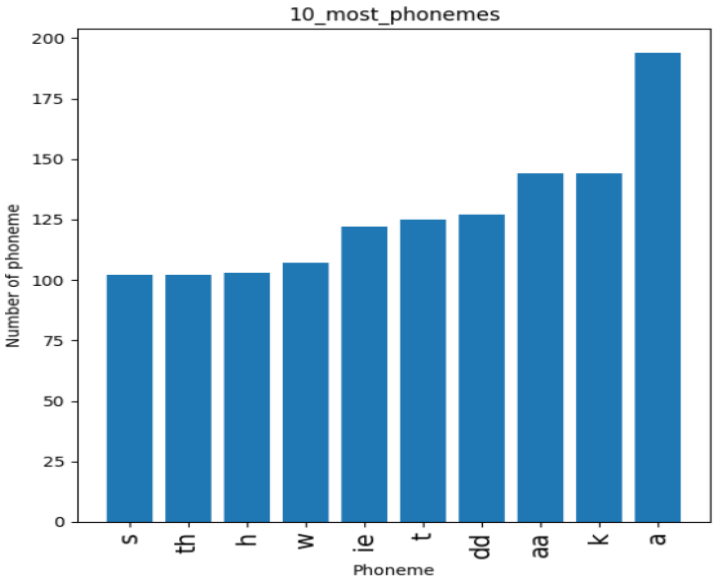
| Phát âm | Âm vị | Phát âm | Âm vị |
|---|---|---|---|
| a | ab | đa | dd ab |
| á | as | đá | dd as |
| à | af | đà | dd af |
| ã | ax | đã | dd ax |
| ả | ar | đả | dd ar |
| ạ | aj | dạ | dd aj |

*Bảng 3.1 Phân tích âm vị của các biến thể thanh điệu của phát âm “a” và “đa”.*

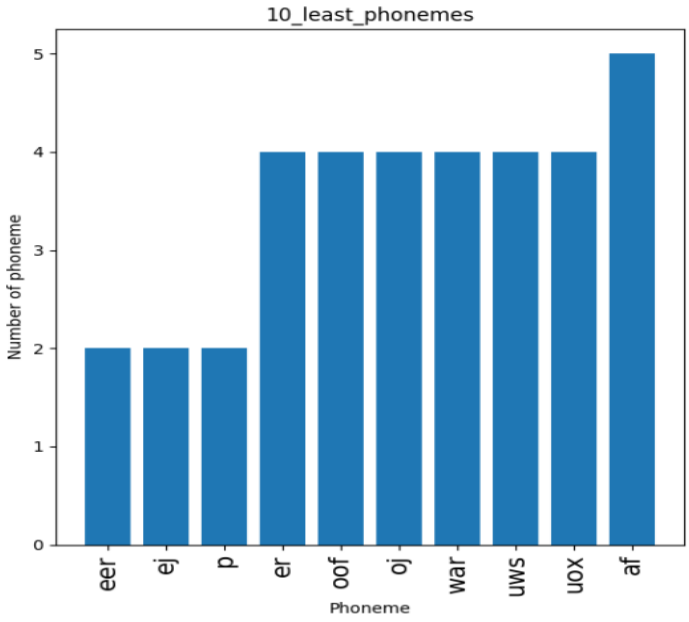
Cái nhìn tổng quan phân bố âm vị trong tiếng Việt có trong tập dữ liệu huấn luyện được mô tả trong hình 3.1. Hình 3.2 và hình 3.3 cho chúng ta thấy 10 âm vị xuất hiện nhiều nhất và ít nhất tương ứng trong tập dữ liệu.




*Hình 3.1 Tổng quan phân bố âm vị trong tiếng Việt có trong tập dữ liệu huấn luyện.*




*Hình 3.2 Mười âm vị xuất hiện nhiều nhất trong tập dữ liệu huấn luyện.*




*Hình 3.3 Mười âm vị xuất hiện ít nhất trong tập dữ liệu huấn luyện.*

Dữ liệu huấn luyện được ghi âm bởi 3 người nói gồm: bố tôi (DTM1), tôi (DTM2) và anh trai (DTM3). Dữ liệu được ghi âm tại 4 thời điểm khác nhau cho mỗi từ vựng, được ghi âm bởi trình ghi âm của điện thoại smart phone thu được dưới định dạng .mp3 hoặc .m4a.

Bộ dữ liệu đều có nhãn cho mỗi ghi âm. . Thời gian trung bình cho mỗi ghi âm là khoảng 1.8 giây. Bộ dữ liệu bao gồm 1546 từ vựng đơn, 48 từ vựng đôi và 6 từ vựng ba. Vì hạn chế trong việc thu thập dữ liệu nên tôi chỉ thử nghiệm trên tập dữ liệu với tập dữ liệu nhỏ chủ yếu là từ vựng đơn.

Bảng 3.2 mô tả thông số của bộ dữ liệu và phân chia tập dữ liệu huấn luyện vào kiểm thử.

| Người phát âm | Tập huấn luyện | Tập kiểm thử |
|---|---|---|
| DTM1 | 1600 * 3 | 1600*1 |
| DTM2 | 1600 * 3 | 1600*1 |
| DTM3 | 1600 * 3 | 1600*1 |
| Tổng | 14400 | 4800 |

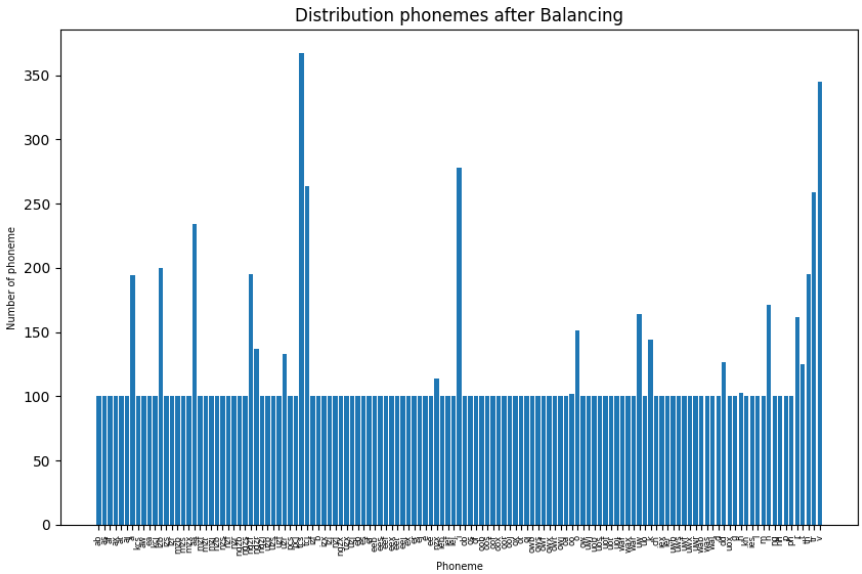
*Bảng 3.2 Mô tả dữ liệu cho tập huấn luyện và tập kiểm.*

## 3.2 Cân bằng dữ liệu

Theo quan sát phân bố tổng quan phân bố âm vị hình 3.1, trong bộ dữ liệu huấn luyện, ta thấy rằng dữ liệu được thu thập mất cân bằng giữa các âm vị. Âm vị là đơn vị phát âm nhỏ nhất của một từ, việc mất cân bằng như vậy là một trong những nguyên nhân dẫn đến làm giảm hiệu suất, tính tổng quát của mô hình. Để giải quyết vấn đề này, chúng ta cần phải cân bằng dữ liệu trước khi thực hiện việc huấn luyện mô hình.

Ý tưởng của tôi là tăng cường những âm vị có số lần xuất hiện ít hơn. Đầu tiên chúng ta xét ngưỡng xuất hiện cân bằng. Trong thử nghiệm tôi lấy ngưỡng cân bằng là 100 (xét với dữ liệu của 1 người). Với những âm vị có số lần xuất hiện ít hơn ngưỡng cân bằng, chúng ta phân chúng vào cụm xuất hiện ít (A), ngược lại với những âm vị có số lần xuất hiện lớn hơn ngưỡng cân bằng, chúng ta phân chúng vào cụm xuất hiện nhiều (B). Với những âm vị có trong cụm A, ta thực hiện việc tăng cường số lần xuất hiện của nó lên bằng cách: tìm kiếm từ (word) tương ứng chứa âm vị đó sao cho các âm vị được phân tích ra bởi từ (word) đó có nhiều nhất có thể trong cụm A và ít tồn tại nhất có thể trong cụm B. Từ đó, bộ dữ liệu huấn luyện được cân bằng hơn.

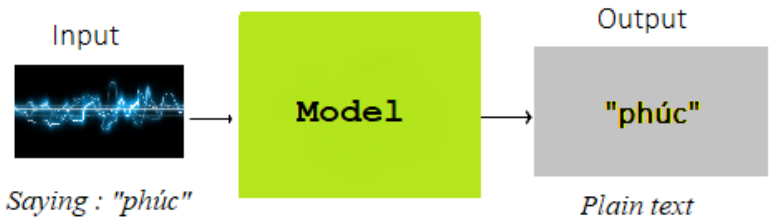
Hình 3.4 mô tả phân bố tổng quan các âm vị sau khi thực hiện việc cân bằng dữ liệu trong bộ huấn luyện.




*Hình 3.4 Phân bố âm vị trong tập huấn luyện sau khi được cân bằng.*

## 3.3 Thực hiện nhận dạng giọng nói

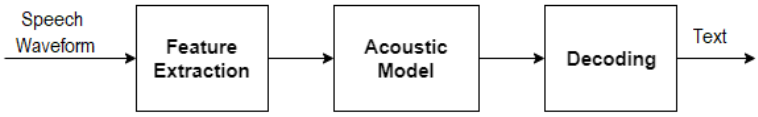
Nhận dạng tiếng nói là quá trình chuyển phát âm tiếng nói (speech) thành văn bản (text). Mục đích của mô hình là khi ta đưa một đoạn ghi âm/phát âm tiếng nói vào hệ thống thì mô hình sẽ đưa ra "bản dịch" của phát âm của đoạn tiếng nói đó. Hình 3.5 minh họa một tín hiệu tiếng nói *"phúc"* khi đi qua mô hình nhận dạng tiếng nói chúng ta thu được một văn bản đầu ra có text là *"phúc"*.




*Hình 3.5 Mô tả hệ nhận dạng tiếng nói.*

Đầu tiên, dữ liệu tiếng nói được trích xuất đặc trưng, ta thu được vector đặc trưng biểu diễn cho tín hiệu tiếng nói đó. Sau đó, các vector đặc trưng được đưa vào mô hình âm học (Acoustic Model). Mô hình âm học tôi triển khai thử nghiệm sử dụng là mô hình mạng rơ-ron sâu (Deep Neural Network – DNN). Đầu ra của mô hình DNN là một phân phối xác suất của bộ ký tự $C$ trên mỗi time-step $t$ (thời điểm). Cuối cùng, chúng ta thực hiện việc giải mã (decoding) đầu ra của mô hình DNN để lấy được phiên âm (transcript) của tín hiệu tiếng nói đó.

Hình 3.6 minh họa các bước trong quá trình nhận dạng tiếng nói.




*Hình 3.6 Các bước trong quá trình thực hiện nhận dạng tiếng nói.*

### 3.3.1 Tiền xử lý dữ liệu âm thanh / tiếng nói

Với dữ liệu được ghi âm bởi trình ghi âm trên điện thoại thu được ở định dạng .mp3 hoặc .m4a. Bước đầu tiên chúng ta phải chuyển đổi chúng sang định dạng .wav, mono-chanel và tỷ lệ lấy mẫu là 16k Hz.

Sử dụng mã lệnh thực hiện việc chuyển đổi (trên termial Linux):

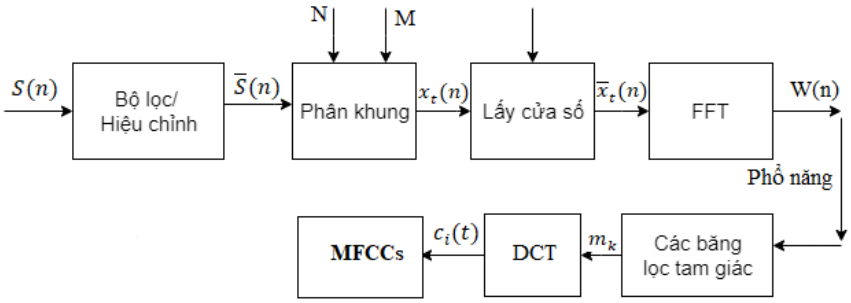
```
for f in *.m4a;
    do ffmpeg -i "$f" -acodec pcm_s16le -ac 1 -ar
16000 "${f/%m4a/wav}";
done
```

Tập dữ liệu thu được sau khi chuyển đổi trên chia thành các folder lưu trữ tập huấn luyện và tập kiểm thử theo các thử nghiệm như được mô tả trong phần 3.4

### 3.3.2 Trích xuất đặc trưng

Âm thanh/ tiếng nói của chúng ta phụ thuộc vào hình dạng của đường âm thanh (vocal tract) như: miệng, lưỡi, phổi, thanh quản, … Nếu xác định được chính xác hình dạng của chúng thì chúng ta có thể nhận ra được âm vị/ ký tự/ từ được phát âm. MFCCs (Mel Frequence Cepstral Coefficients) là phương pháp biểu diễn năng lượng phổ ngắn hạn của âm thanh/ tiếng nói, biểu thị cho hình dạng của đường âm thanh.

Trong đồ án này, tôi sử dụng phương pháp trích xuất đặc trưng MFCCs thử nghiệm trong bài toán này bởi tính hiệu quả của nó thông qua việc phân tích Cepstral theo thang đo Mel. Phương pháp này được xây dựng dựa trên sự cảm nhận của tai người đối với dải tần số khác nhau. Với các tần số thấp (<1000 Hz), độ cảm nhận của tai người là tuyến tính. Đối với các tần số cao, độ biến thiên tuân theo hàm logarit. Các băng lọc tuyến tính ở tần số thấp và biến thiên theo hàm logarit ở tần số cao được sử dụng để trích xuất đặc trưng âm học quan trọng cho tiếng nói. Hình 3.7 mô tả sơ đồ tính toán các hệ số MFCCs.




*Hình 3.7 Sơ đồ tính toán các hệ số MFCCs.*

#### *3.3.2.1. Làm rõ tín hiệu (Pre-emphasis)*

Một tín hiệu tiếng nói $s(n)$ được đưa qua bộ lọc số bậc thấp để phổ đồng đều hơn, giảm ảnh hưởng gây ra cho các xử lý tín hiệu sau này. Thường bộ lọc này cố định bậc một, có dạng như sau:

$$H(z) = 1 - a.z^{-1}, \quad 0.9 \leq a \leq 1.0$$

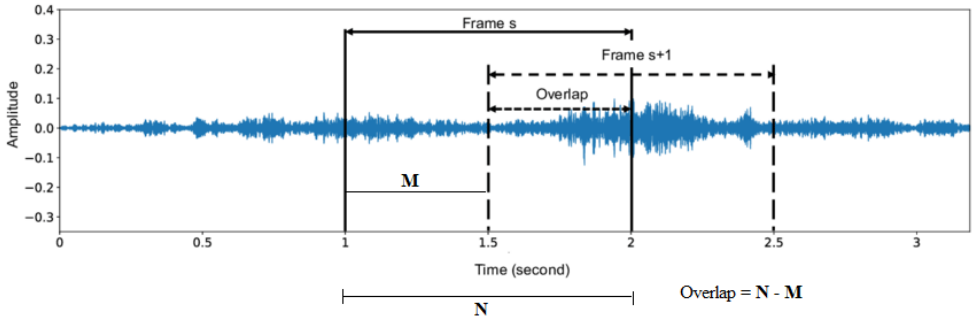
Quan hệ giữa tín hiệu ra với tín hiệu vào:

$$\overline{s}(n) = s(n) - a.s(n-1)$$

Thông thường, giá trị của $a$ là 0.97.

#### *3.3.2.2. Phân khung (Framing Blocking)*

Tín hiệu sau khi hiệu chỉnh $\overline{s}(n)$ được phân thành các khung (frame), mỗi khung có $N$ mẫu; hai khung liên tiếp nhau lệch nhau $M$ mẫu. Khung đầu tiên chứa $N$ mẫu, khung thứ hai chồng lên khung thứ nhất $N - M$ mẫu. Tương tự như thế cho đến khi các mẫu tiếng nói được phân khung. Hình 3.5 mô tả ví dụ phân khung một đoạn tín hiệu.




*Hình 3.8 Ví dụ phân khung của một đoạn tín hiệu. Nguồn [22]*

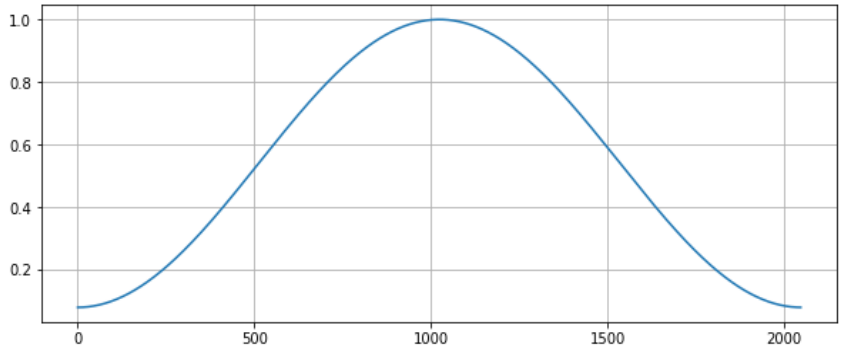
*3.3.2.3. Lấy cửa sổ (Windowing)*

Sau khi thực hiện việc phân khung, bước tiếp theo là lấy cửa sổ cho mỗi khung để làm rõ tín hiệu, đồng thời giảm tính gián đoạn tín hiệu ở đầu và cuối cho mỗi khung trong quá trình trích xuất đặc trưng.

Công thức hàm cửa sổ:

$$w(n) = \begin{cases} \alpha + (1-\alpha).\cos\left(\frac{2\pi n}{N}\right) & |n| \le \frac{N}{2} \\ 0 & |n| > \frac{N}{2} \end{cases}$$

Ở đây, chúng tôi áp dụng cửa sổ Hamming, tức là khi đó $\alpha = 0.54$. Hình 3.5 dưới đây minh họa của sổ Hamming.

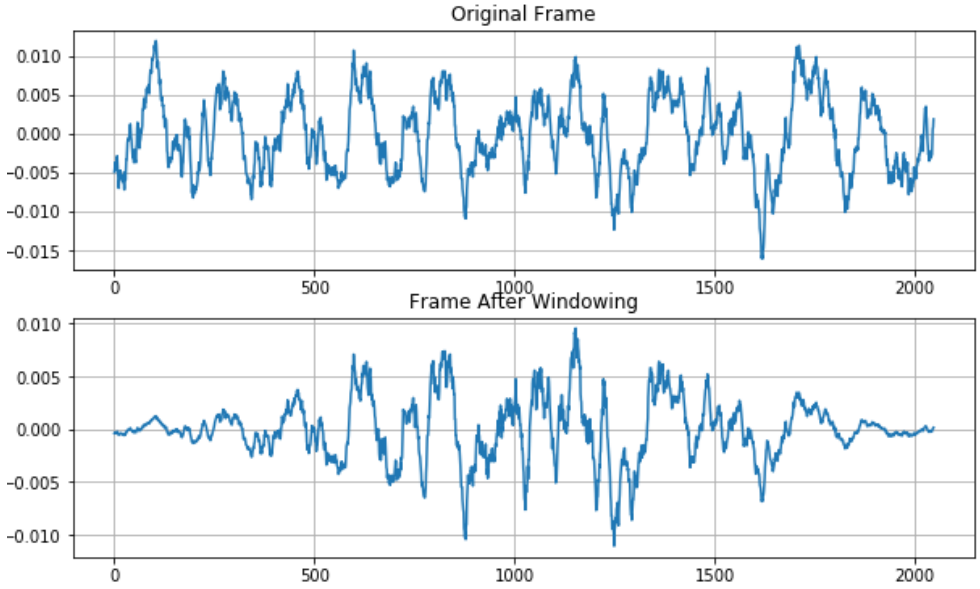


*Hình 3.9 Cửa sổ Hamming.*

Kết quả khi lấy cửa số của khung $x(n)$ theo công thức:

$$\overline{x}_t(n) = x_t(n).w(n), \quad 0 \le n \le N-1$$

Hình 3.6 là một khung của đoạn tín hiệu phát âm *"mười sáu"* mô tả tín hiệu trước và sau khi áp dụng hàm cửa sổ Hamming.




*Hình 3.10 Mô tả một frame trước và sau khi áp dụng cửa sổ Hamming trong tín hiệu phát âm "mười sáu".*

### *3.3.2.4. Biến đổi FFT (Fast Fourier Transform)*

FFT thực hiện việc chuyển đổi mỗi khung với $N$ mẫu từ miền thời gian sang miền tần số. FFT là thuật toán tình DFT nhanh. DFT được xác định như sau:

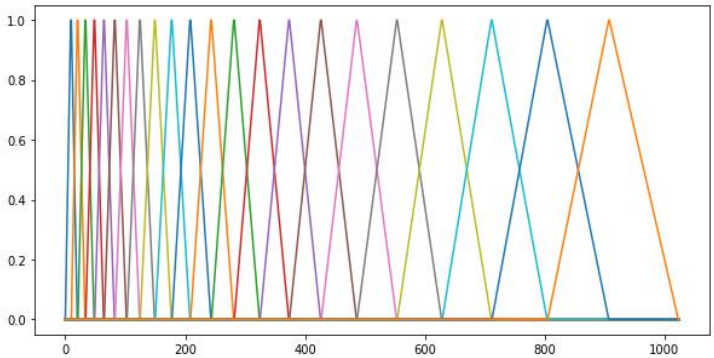
$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-\frac{j2\pi kn}{N}}, \quad k = 0, 1, 2, \dots, N-1$$

Biến đổi sang thang đo Mel trên miền tần số

Công thức gần đúng biểu diễn quan hệ giữa tần số ở thang đo Mel và thang đo tuyến tính như sau:

$$mel(f) = 2595 * \log_{10}\left(1 + \frac{f}{7000}\right)$$

Hình 3.6 mô tả băng lọc tần số Mel với tần số từ 0 đến Fs/2 (Fs=16000 là tần số lấy mẫu), số lượng bộ lọc Mel $K$ là 22, kích thước FFT là 2048 và tỷ lệ mấy mẫu 16000 mà đã được áp dụng vào giải quyết bài toán.




*Hình 3.11 Băng lọc tần số Mel với số lượng bộ lọc là 22, kích thước FFT là 2048, sampling rate là 16kHz.*

Sau khi cho qua $K$ băng lọc tần số Mel, ta thu được một dãy các hệ số $m_k$ $(k = 1,2, \dots, K)$.

*3.3.2.5. Biến đổi Cosin rời rạc (DCT)*

Trong bước này sẽ chuyển logarit của các giá trị $m_k$về miền thời gian bằng phép biến đổi Cosin rời rạc. Sau quá trình biến đổi, đầu ra chúng ta thu được các hệ số MFCCs.

$$c_i = \sqrt{\frac{2}{K}} \sum_{k=1}^{K} \ln(m_k) \cos\left(\frac{\pi i}{K}(k - 0.5)\right), \quad i = 1,2, \dots, K$$

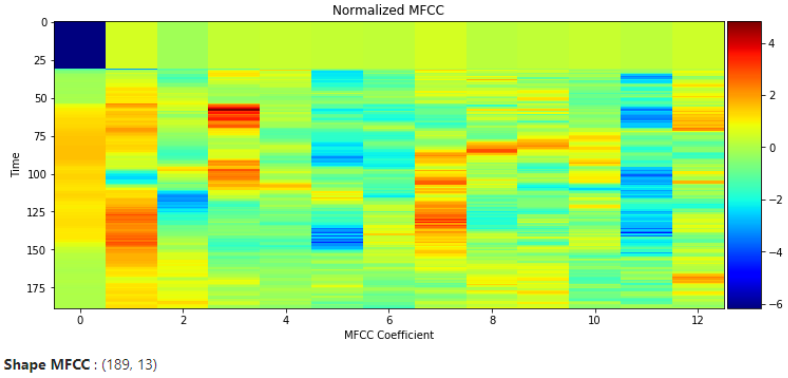
Trong đó, $K$ là số lượng băng lọc,

$m_k$ là giá trị năng lượng của băng lọc thứ $k$,

$i$ là bậc của hệ số.

Tôi lấy 12 hệ số MFCC đầu trong đoạn [1, 12] và thêm 1 hệ số năng lượng của khung sau khi đã được chuẩn hóa làm tham số đặc trưng cho tín hiệu tiếng nói.

Như vậy, đặc trưng thu được sau quá trình trên là một vector có 13 chiều biểu thị đặc trưng cho mỗi khung tín hiệu tiếng nói. Hình 3.7 là một ví dụ biểu diễn phát âm *"mười sáu"* theo hai trục $x$ biểu thị hệ số MFCC và trục $y$ biểu thị theo thời gian.




*Hình 3.12 Một biểu diễn MFCCs-13 chiều của phát âm "mười sáu".*

### 3.3.3 Chuẩn hóa dữ liệu

Trong đồ án này, tôi sử dụng phương pháp Standardization, đưa tất cả các giá trị theo công thức:

$$x' = \frac{x - \mu}{\sigma + \varepsilon}$$

Trong đó, $x$ là giá trị ban đầu, $x'$ là giá trị sau khi chuẩn hóa, $\mu$ và $\sigma$ tương ứng là trung bình và độ lệch chuẩn và $\varepsilon$ là hằng số có giá trị rất nhỏ.

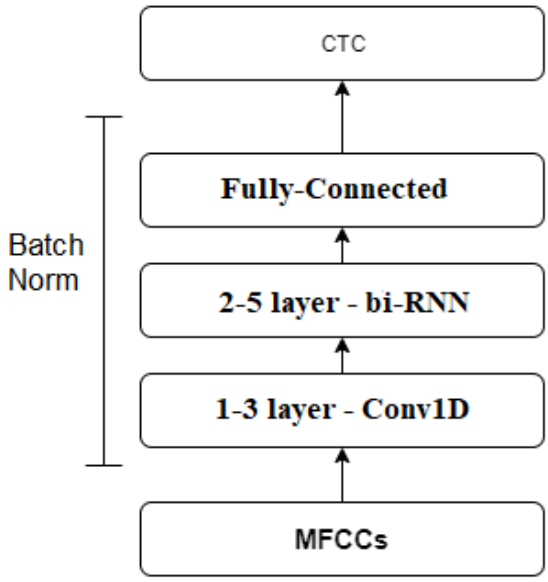
Trong thử nghiệm của tôi, $\mu$ và $\sigma$ được tính dựa trên 100 mẫu dữ liệu trong tập dữ liệu huấn luyện, $\varepsilon$ được lấy có giá trị là $1e - 14$.

### 3.3.4 Xây dựng kiến trúc mô hình

Sau khi thực hiện trích xuất đặc trưng cho tín hiệu cho tiếng nói (cụ thể, sử dụng phương pháp MFCCs) thì các đặc trưng này sẽ được đưa vào mô hình để học, mô hình này gọi là mô hình âm học. Mô hình âm học học mối quan hệ giữa tín hiệu tiếng nói và âm vị hoặc các đơn vị ngôn ngữ khác tạo nên văn bản lời nói.

Kiến trúc mô hình âm học của tôi sử dụng để cho đồ án dựa trên sự thành công của kiến trúc mô hình Deep Speech [10] và Deep Speech 2 [11] cho ngôn ngữ tiếng Anh và Mandarin. Trong Deep Speech, họ sử dụng 2-3 lớp Conv2D kết hợp với 3-7 lớp bidirectional Recurrent (mạng hồi quy 2 chiều). Tuy nhiên, với dữ liệu tương đối nhỏ, kiến trúc mô hình tôi sử dụng bao gồm 1-2 lớp Conv1D kết hợp với 2-3 lớp bidirectional RNN/GRU, 1-2 lớp Fully Connected và sử dụng Batch Normalization cho mỗi lớp. Cách hoạt động của lớp Convolutional và lớp Recurrent đã được nêu ở phần giới thiệu về mô hình CNN và RNN. Ở mỗi lớp Convolution, tôi sử dụng hàm kích hoạt *ReLU*, còn ở các lớp Recurrent tôi sử dụng hàm kích hoạt *Tanh*. Lớp Fully-Connected cuối cùng được đưa vào hàm Softmax. Đầu ra tương ứng của nó là một phân phối xác xuất trên các ký tự. Tôi sử dụng bao gồm 89 chữ cái trong tiếng Việt (tính cả thanh điệu), 1 ký tự khoảng trắng. Để tổng quát hóa với các trường hợp ngoại lệ khi tồn tại một phát âm từ trong tiếng Anh chẳng hạn, tôi sử dụng thêm 4 ký tự chữ cái trong tiếng Anh không có trong tiếng Việt bao gồm {*f, j, w, z*} và 1 ký tự *blank* xử lý cho hai chữ cái lặp lại trong một từ. Tổng thế đầu ra của lớp Softmax có 95 units.

Trong đồ án này, tôi thử nghiệm một số mô hình khác nhau dựa trên sự thay đổi về số lớp Convolution hoặc lớp Recurrent. Thông số chi tiết kiến trúc của mô hình tôi trình bày phần 3.4. Hình 3.13 mô tả kiến trúc các lớp mô hình áp dụng cho nhận dạng tiếng nói tiếng Việt cho người khuyết tật giọng nói.




*Hình 3.13 Kiến trúc chung mô hình cho thử nghiệm nhận dạng giọng nói tiếng Việt cho người khuyết tật giọng nói.*

**Hàm ReLU:**

$$f(x) = \max(0, x)$$

Hàm ReLU được sử dụng nhằm mục đích giảm thiểu độ phức tạp của tính làm cho tốc độ hội tụ và tính toán nhanh hơn.

**Hàm Tanh:**

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Hàm Tanh nhận giá trị trong khoảng (-1, 1).

**Hàm Softmax:**

$$a_i = \frac{\exp(z_i)}{\sum_j^C \exp(z_j)}$$

Trong đó, $C$ là số classes đầu ra. ($C = 95$).

Hàm Softmax được dùng cho mô hình bởi bì đầu ra của $C$ classes tương ứng sẽ được phân loại có tổng bằng 1. Đầu ra của nó sẽ là một phân phối xác suất cho mỗi class. Từ đó cho phép chúng ta dễ dàng decoding đầu ra tương ứng.

**Batch Normalization:**

Batch Normalization là một phương pháp tốt khi huận luyện mô hình học sâu. Mục tiêu của phương pháp là chuẩn hóa đặc trưng về trạng thái zero mean (kỳ vọng bằng 0) và unit-variance (phương sai đơn vi, thông thường nó là 1). Công thức như sau:

$$\hat{x}^{(k)} = \frac{x^{(k)} - E[x^{(k)}]}{\sqrt{Var[x^{(k)}]}}$$

$$y^{(k)} = \gamma^{(k)}\hat{x}^{(k)} + \beta^{(k)}$$

Trong đó, $\gamma$ và $\beta$ là các tham số cần học.

Việc sử dụng Batch Normalization có những ưu điểm:

- Cho phép chúng ta thiết lập tốc độ học lớn.
- Giảm sự phụ thuộc vào việc khởi tạo  khởi tạo.
- Có vai trò như một kỹ thuật regularization (chính quy hóa) [25], trách việc mô hình overfiting (quá khớp) [26].

**Hàm mục tiêu CTC (Connectionist Temporal Classification):**

Quá trình huấn luyện mô hình sử dụng hàm mục tiêu CTC. CTC là hàm mục tiêu cho phép mạng Recurrent được huấn luyện cho các mục đích phiên âm chuỗi mà không yêu cầu bất kỳ việc căn chỉnh trước đó giữa chuỗi đầu vào (input) và chuỗi nhãn (transcript).

Output layer bao gồm một unit duy nhất cho mỗi ký tự. Giả sử chuỗi input $x$ có độ dài $T$, vector output $y_t$ được chuẩn hóa với hàm Softmax là xác suất của ký tự ứng với chỉ số $k$ ở time-step (thời điểm) $t$:

$$\Pr(k,t|x) = \frac{\exp(y_t^k)}{\sum_{k'} \exp\left(y_t^{k'}\right)}$$

Trong đó, $y_t^k$ là phần tử thứ $k$ của $y_t$.

Một "*căn chỉnh*" (alignment) $a$ là chuỗi đầu ra có thể với độ dài $T$. Xác suất $\Pr(a|x)$ của $a$ là tích của các xác suất của ký tự tương ứng trên từng time-step.

$$\Pr(a|x) = \prod_{t=1}^{T} \Pr(a_t,t|x)$$

Đối với một chuỗi nhãn cụ thể, có nhiều alignment khả thi. Ví dụ như các alignment (a, a, b), (-, a, b), (a, b, b) và ( a, b, -) đều tương ứng với nhãn (a, b). Khi có cùng một nhãn ký tự xuất hiện trên các time-step liên tiếp, các phần lặp lại này được loại bỏ. Biểu thị $\mathcal{B}$ là toán tử loại bỏ các nhãn lặp lại (và các ký tự "blank" nếu xét) từ các alignment. Tổng xác suất của một chuỗi nhãn $y$ bằng tổng xác suất của các alignment tương ứng với nó như sau:

$$\Pr(y|x) = \sum_{a \in \mathcal{B}^{-1}(y)} \Pr(a|x)$$

Với dữ liệu chưa được phân đoạn, chúng ta không biết được các nhãn ký tự cho từng time-step cụ thể như thế nào, vì thế chúng ta xét trên tất cả các vị trí mà nó có thể xảy ra khả thi, tương ứng với chuỗi nhãn $y$.

Từ đó, với một chuỗi nhãn mục tiêu $y^*$, công thức của hàm mục tiêu CTC:

$$CTC(x) = -\log(\Pr(y^*|x)) \text{ [20]}$$

Khi chúng ta xây dựng hàm mục tiêu, hay còn gọi là hàm mất mát (loss function), việc tối ưu hàm mất mát trong bài toán này được thực hiện bởi thuật toán Stochastic Gradient Descent (SGD) [31] với learning rate (tốc độ học) được thiết lập sẵn phù hợp với bài toán theo thử nghiệm. Ngoài ra, tôi áp dụng các kỹ thuật Momentum và Nesterov [21] vào việc cập nhật trọng số của mô hình bằng phương pháp backpropagation [17]. Mô hình sẽ tối ưu để cho cả dữ liệu huấn luyện và dữ liệu kiểm thử đều có hàm mất mát là nhỏ nhất. Sau một số lượng vòng lặp nhất định mô hình sẽ được dừng lại dựa trên quan sát sự biến đổi của giá trị hàm mất mát của tập traning và tập valid. Mô hình sau khi được huấn luyện sẽ được lưu vào để lúc kiểm thử chúng ta chỉ cần thực hiện việc gọi lại mô hình mà không cần thiết huấn luyện lại từ đầu.

### 3.3.5 Giải mã đầu ra (Decoding)

Đầu ra của mô hình âm học là một phân phối xác suất của bộ ký tự $C$ trên mỗi time-step. Nó chính là một CTC network vì có tính chất Temporal Classifer (tức là, phân loại theo thời gian). Đầu ra của nó là một lớp Softmax có số unit bằng $C$. Trong đồ án, tôi xét tập $C$ gồm 95 ký tự, mỗi ký tự tương ứng với một

class. Kết quả của các unit này là xác suất phân loại các class tương ứng tại các thơi điểm nhất định. Kết hợp các unit lại sẽ hợp thành các cách "*căn chỉnh*" (alignment) $y$ cho một chuỗi đầu vào $x$. Xác suất của một $y$ bằng tổng xác suất của tất cả các alignments có thể có của nó.

Với ma trận xác suất đầu ra, chúng ta thực hiện việc giải mã (decoding) để lấy văn bản text tương ứng của phát âm.

Có hai thuật toán tôi tiếp cận cho việc decode đầu ra này là *Greedy Search* (hay còn gọi là *max decoding*) và *Beam Search*.

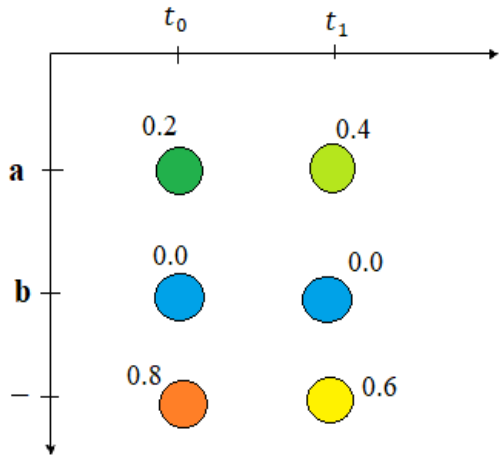
#### *3.3.5.1. Sử dụng thuật toán Greedy Search*

Thuật toán này Greedy Search là phương pháp tiếp cận đơn giản nhất để decode ma trận đầu ra theo các bước sau:

(1) Nối các ký tự khả thi nhất (tức là có xác suất lớn nhất) trên từng time-step, gọi là Best-Candidate.

(2) Sau đó, loại bỏ những ký tự lặp lại. Và ta thu được text đầu ra sau khi decoding tưng ứng với text đầu ra dự đoán.

Ví dụ cho đầu ra của một phát âm "*a*", với bộ chữ cái gồm {*a, b*} và ký tự '_' tương ứng với khoảng trắng, được minh họa như hình 3.15.




*Hình 3.14 Minh họa ma trận đầu ra của CTC với phát âm "a" với bộ chữ cái gồm {a, b}.*

Với giả thuyết đầu ra như vậy ta truy xuất được 9 path bao gồm *{ a a, a b, a _, b a, b b, b _, _ a, _ b, _ _}*. Một path chính là một khả năng có thể trước khi rút gọn. Theo các bước của thuật toán decoding Greedy Search, ta có:

(1): '_ _ ' ( Dấu '_' tương ứng với kí tự khoảng cách. )

(2): ''

Đầu ra sẽ là "" (tức là rỗng) có xác suất là tích xác suất của các ký tự trên từng time-step, tức là *0.8*0.6* bằng *0.48*.

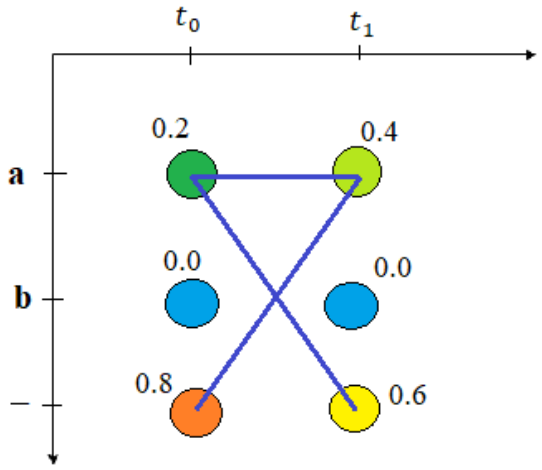
Mã giải thuật toán Greedy Search được mô tả trong hình 3.15.

```
Data:   Ma trận đầu ra của DNN  mat
Result: Text được giải mã

candidate = {}                                          (1)
score = 1                                               (2)
for t = 1 -> n_timestep do                              (3)
    c = charBestScore(mat(t))                           (4)
    if c != candidate[-1]                               (5)
        candidate ∪ c                                   (6)
    score = mat(t, c) * score                           (7)
end                                                     (8)

return candidate, score                                 (9)
```

*Hình 3.15 Thuật toán Greedy Search.*

Greedy search lấy các ký tự có score (điểm số) lớn nhất trên mỗi tim-step. Candidate là một đầu ra khả thi nhất. Ban đầu candidate được khởi tạo rỗng (1) và score tương ứng bằng 1 (2). Ở mỗi time-step lấy ký tự có score lớn nhất (4). Nếu ký tự được lấy khác với ký tự phía trước nó, tức là ký tự cuối cùng đã có trong candidate thì ta thêm ký tự đó vào candidate (6) và score tương ứng được tính lại bằng cách nhân với xác suất đầu ra của ký tự đó ở time-step tương ứng (7). Cuối cùng, ta thu được đầu ra cho thuật toán Greedy Search và score tương ứng (9).

Việc sử dụng thuật toán Greedy Search mang lại tốc độ rất nhanh. Nếu có $C$ ký tự và $T$ time-steps, thuật toán chạy với độ phức tạp là $O(T * C)$.




*Hình 3.16 Tất cả các path tương ứng với đầu ra là text "a".*

Vấn đề với sử dụng thuật toán Greedy Search chỉ đơn thuần lấy chữ cái với xác suất cao nhất ở mỗi time-step. Trong thực tế, chúng ta có thể truy vết được nhiều path thông qua CTC đều cho cùng nhãn, dựa trên quy tắt rút gọn rồi sau đó chọn nhãn với xác xuất lớn nhất. Như ví dụ minh họa trên, các path {*a a, a _, _ a*} đều cho chúng ta đầu ra "*a*" với xác suất bằng tổng xác suất tất cả các path tương ứng của nó. Hình 3.11 mô tả tất cả các path tương ứng với đầu ra "*a*". Tức là xác suất cho đầu ra "*a*" là 0.2*0.4 + 0.2*0.6 + 0.8*0.4= 0.52 > 0.48. Vì thế đầu ra "a" khả thi hơn "".

Ta có thể thấy rằng, việc áp dụng Greedy Search còn có hạn chế trong một số trường hợp nhất định.

### *3.3.5.2. Sử dụng thuật toán Beam Search*

Thuật toán Beam Search được xây dựng nhằm mục đích giải quyết vấn đề hạn chế của thuật toán Greedy Search. Hình 3.17 mô tả thuật toán Beam Search.

```
Data:   Ma trận đầu ra của DNN  mat, BW
Result: Text được giải mã

beams = {∅}                                              (1)
scores( ∅ , 0) = 1                                       (2)
for t = 1 .. do                                          (3)
    bestBeams = bestBeams(beams, BW)                     (4)
    beams = {}                                           (5)
    for b in bestBeams do                                (6)
        beams = beams U b                                (7)
        scores(b, t) = calcScore(mat, b, t)              (8)
        for c in alphabet do                             (9)
            b' = b + c                                   (10)
            scores(b', t) = calcScore(mat, b', t)        (11)
            beams = beams U b'                           (12)
        end                                              (13)
    end                                                  (14)
end                                                      (15)

return beastBeams(beams, 1)                              (16)
```

*Hình 3.17 Thuật toán Beam Search.*

Beam Search lặp đi lặp lại việc tạo ra các *beams* và *scores*. Một beam tức là một đầu ra có thể và score là điểm số tương ứng của nó. Danh sách các beam ban đầu được khởi tạo rỗng (1) và score tương ứng (2). Sau đó, thuật toán lặp lại trên tất cả các time-step của ma trận đầu ra (3-15). *Beam-With* ($BW$) là số lượng beam được giữ lại ở từng time-step. Tại mỗi time-step, chỉ có BW có score tốt nhất từ các time-step trước được giữ lại (4). Đối với mỗi beam này, score ở time-step hiện tại được tính như (8). Sau đó, mỗi beam được mở rộng bởi tất cả các ký tự có thể (10). Ở đây của chúng ta là các ký tự trong bảng chữ cái tiếng Việt tính cả thanh điệu của nó và ký tự khoảng cách. Và sau đó, score được tính (11). Sau time-step cuối cùng, beam tốt nhất được trả về (16).

Để tính toán *score* cho mỗi beam. Giả sử tổng xác suất của tất cả các path tương ứng với beam $b$ ở time-step $t$ bằng $Ptot(b,t)$. Có 2 trường hợp sảy ra: mở rộng bằng lặp lại ký tự cuối và mở rộng bằng một ký tự khác. Khi chúng ta thu gọn các path mở rộng, thu được hoặc là beam không thay đổi ("a" → "$a$") (copy) hoặc chúng ta thu được beam được mở rộng (a"→"$ab$") .

*Copy beam*: Giả sử chung ta thêm một ký tự giống với ký tự cuối trong beam. Khi đó, công thức tính score như sau:

$$Ptot(b,t) += Ptot(b,t-1).mat(b[-1],t)$$

Trong đó -1 là chỉ số của ký tự cuối trong beam.

*Extend beam*: Giả sử chúng ta thêm một ký tự $\boldsymbol{c}$ khác với ký tự cuối. Khi đó, công thức tính như sau:

$$Ptot(b+c) = Ptot(b, t-1).mat(c, t)$$

### 3.3.6 Phương pháp đánh giá

Trong đồ án này, tôi sử dụng độ đo tỷ lệ lỗi của từ (WER) để đánh giá hiệu suất của mô hình nhận dạng tiếng nói. Ngoài ra, tôi cũng sử dụng thêm một độ đo cũng khá phổ biến khác là tỉ lệ lỗi ký tự (CER) và tỉ lệ lỗi câu (SER) để đánh giá một cách tổng quan hơn hiệu quả của mô hình được xây dựng áp dụng cho nhận dạng tiếng nói của người khuyết tật. Công thức của các độ đo được tính toán như sau:

- **Word Error Rate:**

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

Trong đó, $S$ là số lượng từ thay thế,

$D$ là số lượng từ xóa đi,

$I$ là số lượng từ thêm vào,

$C$ là số lượng từ chính xác,

$N$ là tổng số lượng từ trong transcript ($N = S + D + C$).

- **Sentence Error Rate:**

$$SER = \frac{F}{N}$$

Trong đó, $F$ là số lượng câu sai,

$N$ là tổng số lượng câu.

- **Character Error Rate:**

$$CER = \frac{s + d + i}{n}$$

Trong đó, $s$ là số lượng ký tự thay thế,

$d$ là số lượng ký tự xóa đi,

$i$ là số lượng ký tự thêm vào,

$n$ là tổng số lượng ký tự trong transcript.

## 3.4 Kết quả thử nghiệm

Các thử nghiệm của tôi đã được thực hiện trên bộ dữ liệu như đã mô tả ở phần 3.1. Bộ dữ liệu bao gồm 1600 từ vựng được thu âm ở 4 thời điểm nói khác nhau của 3 người nói khuyết tật khác nhau. Tôi thực hiện việc chia dữ liệu thành hai phần:

- Tập dữ liệu training (huấn luyện):

Tập dữ liệu để huấn luyện mô hình gồm 1600 từ vựng được ghi âm ở 3 thời điểm của 3 người nói. Tổng cộng có 14400 phát âm.

- Tập dữ liệu valid (kiểm thử):

Tập dữ liệu để kiểm thử mô hình gồm 1600 từ vựng được ghi âm ở thời điểm còn lại của 3 người nói. Tổng cộng có 4800 phát âm.

Phương pháp trích xuất đặc trưng MFCCs tôi áp dụng cho toàn bộ thử nghiệm với windowing=20ms, stride=10ms. Tức là, mỗi frame có độ dài là 20ms và khoảng cách giữa các frame là 10ms. Số chiều đặc trưng MFCCs là 13 chiều.

Ban đầu, tôi sử dụng mạng RNN thuần hai chiều (bi-simple RNN) và sau đó, tôi thử nghiệm với việc sử dụng mạng GRU hai chiều (bi-GRU). Chi tiết cụ thể của hai mô hình như sau

- *Model 1* (2 Conv1D + 3 bi-simple RNN):

Kiến trúc *model 1* sử dụng hai lớp Conv1D và ba lớp bidirectional-simple RNN (RNN thuần hai chiều). Kiến trúc tốt nhất cho mạng RNN thuần được mô tả chi tiết trong hình 3.18. Các lớp Conv1D sử dụng bộ tham số filter=512, kernel=5, stride=1, hàm kích hoạt ReLU. Với các lớp bidirectional-simple RNN với units=1024, dropout=0.4, merge_mode=`sum`, hàm kích hoạt Tanh. Lớp cuối cùng là một lớp Fully-connected với output=95 và sử dụng hàm kích hoạt Softmax.

| Layer (type) | Output Shape | Params # |
|---|---|---|
| input (InputLayer) | (None, None, 13) | 0 |
| bn_1 (BatchNormalization) | (None, None, 13) | 52 |
| conv1D_1 (Conv1D) | (None, None, 512) | 33792 |
| conv1D_2(Conv1D) | (None, None, 512) | 1311232 |
| bn_2 (BatchNormalization) | (None, None, 512) | 2048 |
| bidir_rnn_1 (Bidirectional RNN) | (None, None, 1024) | 3147776 |
| bidir_rnn_2 (Bidirectional RNN) | (None, None, 1024) | 4196352 |
| bidir_rnn_3 (Bidirectional RNN) | (None, None, 1024) | 4196352 |
| bn_3 (BatchNormalization) | (None, None, 1024) | 4096 |
| time_distributed (TimeDistribited) | (None, None, 95) | 97375 |

Total params: 12,989,075
Trainable params: 12,985,977
Non-trainable params: 3,098

*Hình 3.18 Kiến trúc mô hình thử nghiệm 1 (Model 1) bao gồm 2 layer Conv1D và 3 layer bi-simple RNN.*

- *Model 2* (1 Conv1D + 2 bi-GRU):

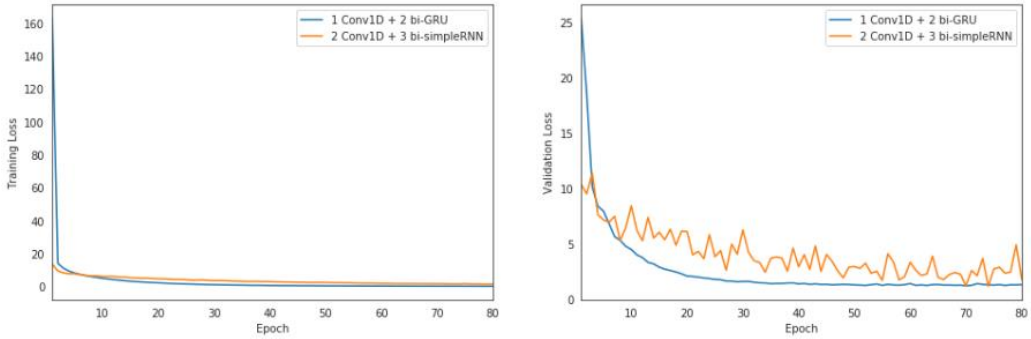
*Model 2* sử dụng một lớp Conv1D với filters=512, kernel=5, stride=1, hàm kích hoạt ReLU và hai lớp bidirectional-GRU với units=1024, dropout=0.5, merge_mode=`sum`, hàm kích hoạt được sử dụng là hàm Tanh. Giữa mỗi lớp sử dụng Batch Normalization. Lớp cuối cùng cũng là một Fully-connected có chiều output là 95 và sử dụng hàm kích hoạt Softmax. Chi tiết tham số, cấu trúc mô hình được mô tả trong hình 3.19.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input (InputLayer) | (None, None, 13) | 0 |
| conv1D_1 (Conv1D) | (None, None, 512) | 33792 |
| bn_conv_1d (BatchNormalization) | (None, None, 512) | 2048 |
| bidirectional_GRU_1 (bi-GRU) | (None, None, 1024) | 9443328 |
| bn_gru_1 (BatchNormalization) | (None, None, 1024) | 4096 |
| bidirectional_GRU_2 (bi-GRU) | (None, None, 1024) | 12589056 |
| bn_gru_2 (BatchNormalization) | (None, None, 1024) | 4096 |
| time_distributed_1 (TimeDistributed) | (None, None, 95) | 97375 |
| softmax (Activation) | (None, None, 95) | 0 |

**Total params: 22,173,791**
**Trainable params: 22,168,671**
**Non-trainable params: 5,120**

*Hình 3.19 Kiến trúc mô hình thử nghiệm 2 (Model 2) bao gồm 1 layer Conv1D và 2 layer bi-GRU.*

Hình 3.20 mô tả kết quả sự thay đổi của hàm mất mát của hai mô hình nói trên sau 80 epochs.




*Hình 3.20 Kết quả sự thay đổi hàm mất mát của hai mô hình Model1 và Model2 sau 80 epochs. Đường màu cam biểu thị cho Model 1 và đường màu xanh biểu thị cho Model 2.*

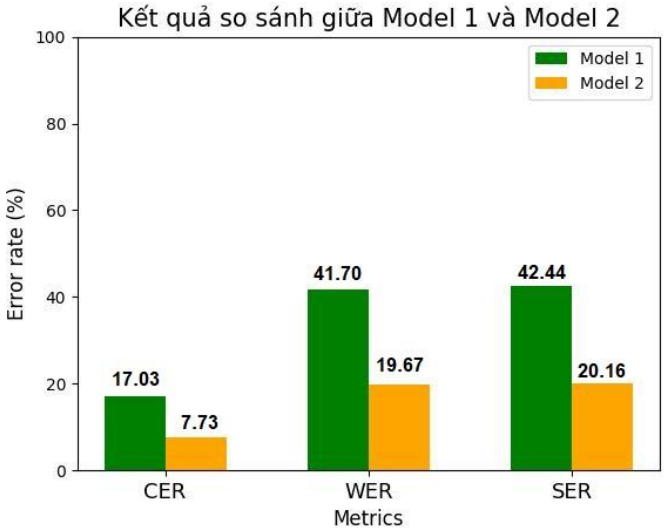
Việc decoding đầu ra, tôi thử nghiệm sử dụng cả hai thuật toán Greedy Search và Beam Search (BW=1000). Về cơ bản, trong thử nghiệm với bộ dữ liệu này, hai thuật toán đều cho kết quả tương tự nhau.

Kết quả của hai mô hình được kiểm thử trên tập kiểm thử như sau:

| Metrics / Models | CER | WER | SER |
|---|---|---|---|
| Model 1 (2 Conv1D + 3 bi-RNN) | 17.03 | 41.70 | 42.44 |
| Model 2 (1 Conv1D + 2 bi-GRU) | **7.73** | **19.67** | **20.16** |

*Bảng 3.3 Kết quả so sánh của hai mô hình thử nghiệm sử dụng mạng bi-simpleRNN và bi-GRU dựa trên ba độ đo CER, WER và SER. (%)*

Hình 3.21 minh họa kết quả so sánh của hai mô hình thử nghiệm *Model 1* và *Model 2* với ba độ đo CER, WER và SER (%).

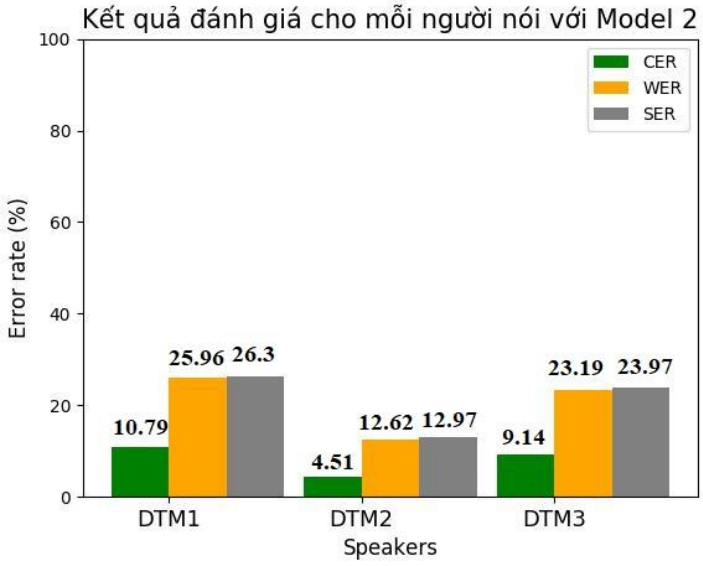



*Hình 3.21 Minh họa kết quả so sánh giữa Model 1 và Model 2 với ba độ đo CER, WER và SER (%).*

Từ hình 3.21 và kết quả bảng 3.3 cho ta thấy được kết quả vượt trội của mạng GRU 2 chiều và đạt được với kết quả các độ đo CER=7.73%, WER=19.67% và CER=20.16%.

Về cơ bản, hiệu quả của mạng GRU so với mạng RNN thuần đã được chứng minh trong nhiều thử nghiệm khác nhau. Trong thử nghiệm của đồ án này, kết quả trên cũng cho thấy hiệu quả rõ rệt của mạng GRU nhờ khả năng ghi nhớ phụ thuộc xa của nó.

Hình 3.22 mô tả kết quả đánh giá của mô hình *Model 2* cho kết quả nhận dạng tiếng nói của mỗi người.




*Hình 3.22 Kết quả đánh giá của Model 2 của mỗi người dựa trên ba độ đo CER, WER và SER (%).*

Ta có thể thấy rằng, mô hình học tốt nhất với dữ liệu tiếng nói của DTM2 (tức là tiếng nói của tôi) với tỷ lệ lỗi thấp hơn nhiều so với DTM1 (bố) và DTM3 (anh

trai). Về mặt định tính thì trong thực tế đánh giá của người nghe, giọng tôi cũng nghe rõ hơn của bố và anh trai.

Tham số tối ưu tôi sử dụng cho việc huấn luyện và đánh giá mô hình của *Model 2* như sau:

- Phương pháp tối ưu (Optimizer): SGD;
- Tốc độ học (Learning rate): 0.02;
- Weight decay: $1e-6$;
- Momentum: 0.9;
- Nesterove: True;
- Clip-norm: 5;
- Batch size: 32;
- Dropout: 0.5.

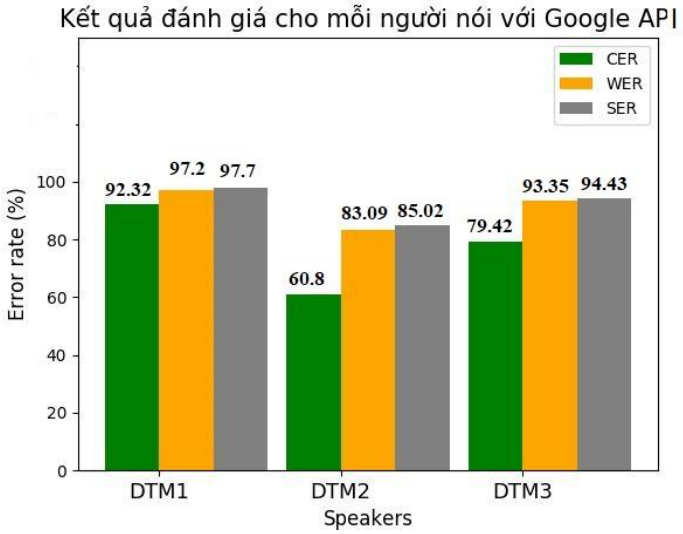
Để trực quan hơn việc đánh giá hiệu quả của mô hình, tôi sử dụng Google API [32] để đánh giá nhận dạng tiếng nói trên cùng dữ liệu của tập thử nghiệm cho hai mô hình đã thử nghiệm nói trên và so sánh nó với kết quả mô hình tối ưu nhất trog hai mô hình tôi đã xây dựng.

Bảng 3.4 mô tả kết quả đánh giá khi sử dụng Google API để nhận dạng tiếng nói cho người khuyết tật giọng nói.

| Metrics / Models | CER | WER | SER |
|---|---|---|---|
| Google API [32] | 75.34 | 90.21 | 91.51 |

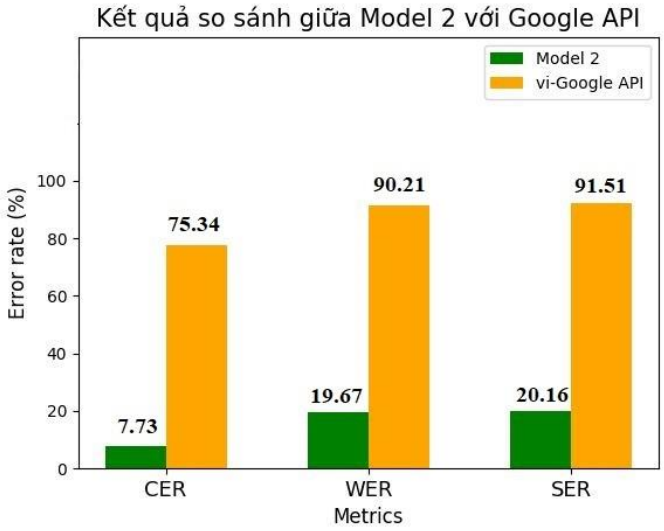
*Bảng 3.4 Kết quả đánh giá của Google API cho nhận dạng tiếng nói của người khuyết tật giọng nói trên dữ liệu tập thử nghiệm (%).*

Hình 3.23 mô tả kết quả đánh giá sử dụng Google API nhận dạng tiếng nói của mỗi người.




*Hình 3.23 Kết quả đánh giá của Google API của mỗi người dựa trên độ đo CER, WER và SER (%).*

Hình 3.23 minh họa kết quả so sánh các độ đo giữa mô hình Model 2 và sử dụng Google API.




*Hình 3.24 Kết quả so sánh giữa mô hình Model 2 và sử dụng Google API dựa trên CER, WER và SER (%).*

Dựa vào kết so sánh giữa mô hình *Model 2* và Google API cho nhận dạng tiếng nói của người khuyết tật giọng nói trong hình 3.24 ta thấy rằng, mô hình *Model 2* vượt trội hoàn toàn so với kết quả của Google API. Tuy nhiên, cũng có thể lý giải cho việc đó vì mô hình *Model 2* được huấn luyện với dữ liệu tiếng nói của người khuyết tật giọng nói nên hiệu quả mang lại của nó sẽ tốt hơn. Từ đó cũng minh chứng cho chúng ta thấy rằng mô hình *Model 2* đã hoạt động tốt cho vấn đề nhận dạng tiếng nói của người khuyết tật giọng nói.

## 3.5 Đánh giá tổng quan mô hình cho việc áp dụng vào hệ thống nhận dạng tiếng nói cho người khuyết tật giọng nói

Với kết quả thử nghiệm nhận dạng dựa trên các độ đo đã được đánh giá trong phần 3.4, ta thấy rằng mô hình với kiến trúc được xây dựng gồm các một Conv1D kết hợp với hai lớp bidirectional-GRU đã mang lại hiệu suất tốt trên dữ liệu với người khuyết tật giọng nói trên bộ dữ liệu thử nghiệm.

Kết quả đánh giá với mô hình tốt nhất cho thử nghiệm *(Model 2)* trên bộ dữ liệu thu thập vẫn còn một số trường hợp nhận dạng sai. Cụ thể các trường hợp dẫn đến quá trình nhận dạng sai ký tự như sau:

- Các chữ cái mà phát âm giống nhau ví dụ như: *"ia"* với *"ya"; "iê"* với *"yê"; "âu"* với *"ô", "au"* với *"o"*.
- Một số chữ cái mà người khuyết tật khó phát âm phân biệt như: *"tr"* với *"t"; "x"* với *"s"; "gi"* với *"d"; "nh"* với *"n"*.
- Một số chữ cái mà khi người khuyết tật phát âm nghe bị lệch như: *"kh"* với *"c"; "l"* với *"n"; "t"* với *"c"; "ng"* với *"n"*.

- Trong một số trường hợp bị phát ra thiếu âm chẳng hạn như *"q"* (tức là, *"quý"* thành *"úy")*; "tr" thành "t" hoặc "" (tức là *"trong"* thành *"tong"* hoặc *"ong"* hay *"dấu"* thành *"ấu"*…
- Và một số trường hợp mô hình vẫn còn nhận dạng sai thanh điệu của các nguyên âm.

Việc nhận dạng sai dù chỉ một ký tự dẫn đến kết quả tỷ lệ lỗi của từ cao hơn nhiều so với tỷ lệ lỗi của ký tự. Tỷ lệ lỗi của từ và tỷ lệ lỗi của câu tương đồng nhau vì bộ dữ liệu chủ yếu là từ đơn. Song, bên cạnh đó vẫn có một số từ ghép không nhận dạng đúng hoàn chỉnh.

Với các mô hình học sâu, dữ liệu đóng vai trò quan trọng, ảnh hưởng lớn đến hiệu suất của mô hình. Kiến trúc mô hình đã thử nghiệm trên một lượng dữ liệu nhỏ, nhưng cũng đã cho ta thấy được tính khả quan của mô hình áp dụng phương pháp học sâu. Với một hệ thống nhận dạng tiếng nói lớn hơn, chúng ta cũng cần thời gian thử nghiệm và đánh giá.

Nhìn chung, phương pháp áp dụng các mô hình học sâu nói chung cho bài toán nhận dạng tiếng nói tiếng Việt cho người khuyết tật giọng nói là khả thi.

# CHƯƠNG 4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## 4.1 Kết luận

Đồ án đã trình bày thử nghiệm phương pháp sử dụng mô hình phương pháp học sâu vào bài toán nhận dạng tiếng nói tiếng Việt cho người khuyết tật về giọng nói. Cụ thể nội dung đã đạt được như sau:

- Bước đầu thực hiện được định hướng của bản thân. Nghiên cứu, xây dựng hệ thống hữu ích hỗ trợ cho người khuyết tật nói chung và người khuyết tật giọng nói nói riêng.
- Thu thập dữ liệu thử nghiệm; nghiên cứu, tìm hiểu các phương pháp tiền xử lý dữ liệu, trích xuất đặc trưng cho dữ liệu tiếng nói.
- Tìm hiểu, nghiên cứu một số kiến trúc mô hình học sâu, cài đặt thử nghiệm một số mô hình phù hợp với bộ dữ liệu và so sánh, đánh giá các mô hình thử nghiệm.

**Những khó khăn khi thực hiện đồ án:**

- Do hạn chế về mặt thời gian và miền áp dụng của đề tài đặt ra, dữ liệu của bài toán thu thập còn rất ít với sự đa dạng về từ vựng, người nói và độ dài của mỗi ghi âm. Viêc thu thập dữ liệu thực tế từ người thân trong gia đình mất khá nhiều thời gian để hoàn thành và tiền xử lý với những dữ liệu thu thập được.
- Với việc tìm hiểu sang một lĩnh vực xử lý tiếng nói. Và thử nghiệm cho bài toán nhận dạng tiếng nói cho người khuyết tật, mất nhiều thời gian để tìm hiểu kiến thức nền tảng.
- Việc kiểm soát cũng như hiểu sâu về mạng rơ-ron còn tương đối khó khăn. Đồng thời việc huấn luyện, đánh giá các mô hình tốn rất nhiều thời gian, và tài nguyên.

## 4.2 Hướng phát triển trong tương lai

Qua đánh giá kết quả của các thử nghiệm, chúng ta thấy rõ được hạn chế của dữ liệu huấn luyện và phương pháp học sâu. Trong tương lai, việc cải thiện kết quả cần được thực hiện bao gồm:

- Tăng cường việc thu thập dữ liệu và cải thiện độ rộng và sâu của tập dữ liệu huấn luyện.
- Tiến hành cải thiện việc tiền xử lý dữ liệu, trích xuất đặc trưng bằng các phương pháp hiệu quả hơn.
- Cải kiến kiến trúc mô hình âm học để nó học hiệu quả hơn với dữ liệu.
- Cải tiến công đoạn decoding đầu ra, chẳng hạn như tích hợp các mô hình ngôn ngữ hiệu quả, nhằm cải thiện chất lượng mô hình nhận dạng.
- Thử nghiệm với mô hình nhận dạng tiếng nói với dữ liệu đa dạng thực tế hơn, nhận dạng với chuỗi phát âm dài, …

- Nghiên cứu các phương pháp để cải thiện chất lượng mô hình nhận dạng tiếng nói đa dạng hóa người khuyết tật giọng nói. Tăng cường dữ liệu bằng các phương pháp chẳng hạn như mô hình sinh dữ liệu, … để tăng cường dữ liệu, đồng thời giảm hạn về dữ liệu huấn luyện cần thu thập.
- Nghiên cứu hướng tiếp cận chuyển đổi tiếng nói của người khuyết tật sang tiếng nói của người nói chuẩn.
- Biến đổi các nghiên cứu thành sản phẩm ứng dụng thực tiễn, áp dụng cho những người khuyết tật.

Bên cạnh đó, các hệ thống nhận dạng tiếng nói đang gặp nhiều thách thức. Việc giải quyết những thách thức đó cũng là một hướng đi tương lai cho việc cải tiến bài toán này.

# TÀI LIỆU THAM KHẢO


[1] "Cổng thông tin bộ lao động thương bin và xã hội," 01 11 2019. [Online]. Available: http://www.molisa.gov.vn/Pages/tintuc/chitiet.aspx?tintucID=29543. [Accessed 10 12 2019].

[2] Đ. N. Đức, "Mạng nơ-ron và mô hình Markov ẩn trong nhận dạng tiếng nói," *Luận văn Tiến sĩ, Đại học Quốc Gia Hà Nội*, 2003.

[3] V. T. Thang, "Vietnamese tone recognition based on multi-layer perceptron network.," in *Conference of Oriental Chapter of the International Coordinating Committee on Speech Database and Speech I/O System*, Kyoto, 2008.

[4] N. H. Quang, "Automatic Speech Recognition for Vietnamese using HTK system," *IEEE-RIVF 2010, Ha Noi*, 2010.

[5] N. T. Thanh, "Nhận dạng tiếng việt nói sử dụng bộ công cụ Kaldi," 2017.

[6] Samouelian, "Knowledge based approach to speech recognition," 1994.

[7] C. Kare, "Speech recognition by Dynamic Time Warping," 2013.

[8] S. Boruah, "A study on HMM based speech recognition," 2016.

[9] A. B. Nassif, "Speech Recognition Using Deep Neural Networks: A Systematic Review," 2010.

[10] H. A., "Deep speech: Scaling up end-to-end speech recognition," 2014.

[11] A. D., "Deep speech 2: End-to-end speech recognition in english and mandarin," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning*, 2015.

[12] "Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Analog_signal. [Accessed 07 12 2019].

[13] "Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Digital_signal. [Accessed 07 12 2019].

[14] "TechDifferences," [Online]. Available: https://techdifferences.com/difference-between-analog-and-digital-signal.html. [Accessed 09 12 2019].

[15] "CS231n Course," Stanford, [Online]. Available: http://cs231n.github.io/neural-networks-1/. [Accessed 09 12 2019].

[16] N. L, "CNN - Convolution neural networks - Dress recognition," [Online]. Available: https://narengowda.github.io/cnn-convolution-neural-networks-dress-recognition/. [Accessed 10 12 2019].

[17] Y. L. Cun, "A Theoretical Framework for Back-Propagation," 1998.

[18] X. Glorot, "Understanding the difficulty of training deep feedforward neural networks," 2010.

[19] G. A., "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks.," *ICML*, 2006.

[20] G. A., "Towards end-to-end speech recognition with recurrent neural networks," *ICML*, 2014.

[21] A. Botev, "Nesterov's Accelerated Gradient and Momentum as approximations to Regularised Update Descent," 2016.

[22] S. Abdoli, "End-to-end environmental sound classification using a 1d convolutional neural network," 2019.

[23] "Wikipedia," [Online]. Available: https://www.ezglot.com/most-frequently-used-words.php?l=vie&s=wp-freq. [Accessed 10 10 2019].

[24] "Wikipedia," [Online]. Available: https://en.wiktionary.org/wiki/Category:Vietnamese_compound_words. [Accessed 10 10 2019].

[25] "Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Regularization_(mathematics). [Accessed 11 12 2019].

[26] Đ. M. Hải, "Hai's Blog," [Online]. Available: https://dominhhai.github.io/vi/2017/12/ml-overfitting/. [Accessed 17 12 2019].

[27] "Wikimedia Commons," [Online]. Available: https://commons.wikimedia.org/wiki/File:Recurrent_neural_network_unfold.svg. [Accessed 05 12 2019].

[28] A. Adate, "Evaluation of Gated Recurrent Neural Networks on Deep Sentence Classification," 2017.

[29] "Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Activation_function. [Accessed 04 12 2019].

[30] "Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Gradient_descent. [Accessed 04 12 2019].

[31] S. Ruder, "An overview of gradient descent optimization algorithms," 2017.

[32] "Google Cloud," [Online]. Available: https://cloud.google.com/speech-to-text/docs/reference/rest/?hl=vi. [Accessed 25 12 2019].

# PHỤ LỤC

Thông tin cấu hình máy tính sử dụng cho quá trình thực hiện đồ án tốt nghiệp:

- **Máy tính cá nhân:**

Operating System: Ubuntu 18.04.
System Model: Lenovo Thinkpad X250.
Processer: Intel ® Core ™ i5-5300U CPU @ 2.3GHz.
Memory: 8192MB RAM.

- **Máy tính server:**

Processor: Intel ® Core ™ i7-8700K CPU @ 3.7GHz.
Memory: 16384MB RAM.
GPU: NVIDIA Geforce GTX 1080 Ti.