

**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**

**---------------------------------------**

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC NGÀNH KỸ THUẬT PHẦN MỀM

**XÂY DỰNG MÔ HÌNH CHUYỂN ĐỔI NGÔN NGỮ TỪ ÂM THANH THÀNH VĂN BẢN TIẾNG VIỆT VỚI CƠ CHẾ**

**ATTENTION TRONG TRANSFORMER**

#### CBHD: Ths. Vũ Thị Tuyết Mai Sinh viên: Nguyễn Lương Hùng Mã số sinh viên: 2019601064

Hà Nội – Năm 2023



**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**

**---------------------------------------**

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC NGÀNH KỸ THUẬT PHẦN MỀM

**XÂY DỰNG MÔ HÌNH CHUYỂN ĐỔI NGÔN NGỮ TỪ ÂM THANH THÀNH VĂN BẢN TIẾNG VIỆT VỚI CƠ CHẾ**

**ATTENTION TRONG TRANSFORMER**

#### CBHD: Ths. Vũ Thị Tuyết Mai Sinh viên: Nguyễn Lương Hùng Mã số sinh viên: 2019601064

Hà Nội – Năm 2023

# LỜI MỞ ĐẦU

Em xin được phép bày tỏ lòng biết ơn đến tất cả các cá nhân và tổ chức đã tạo điều kiện hỗ trợ, giúp đỡ em trong suốt quá trình học tập và nghiên cứu đề tài “*Xây dựng mô hình chuyển đổi ngôn ngữ từ âm thanh thành văn bản tiếng Việt với cơ chế Attention trong Transformer*”. Em cũng xin chân thành cảm ơn các thầy cô ở khoa công nghệ thông tin đã truyền đạt vốn kiến thức quý báu cho em trong suốt thời gian học tập tại trường. Nhờ có những lời hướng dẫn, dạy bảo của các thầy, các cô nên đề tài tốt nghiệp: “*Xây dựng mô hình chuyển đổi ngôn ngữ từ âm thanh thành văn bản tiếng Việt với cơ chế Attention trong Transformer*” của em mới có thể hoàn thiện. Một lần nữa, em xin chân thành cảm ơn Ths Vũ Thị Tuyết Mai – người đã trực tiếp giúp đỡ, quan tâm, hướng dẫn em hoàn thành tốt bài báo cáo này trong thời gian qua. Bài báo cáo đồ án của em thực hiện trong khoảng thời gian 9 tuần.

Bài báo cáo của em bao gồm 5 chương:

*Chương I: Tổng quan về đề tài*

*Chương II: Cơ sở lý thuyết thực hiện bài toán Speech to text. Chương III: Các mô-đun thực hiện bài toán Speech to text.*

*Chương IV: Các hướng tiếp cận của bài toán Speech to text Chương V: Thực nghiệm*

Bước đầu đi vào thực tế của em còn hạn chế và còn nhiều bỡ ngỡ nên không gặp nhiều thiếu sót, em rất mong nhận được những ý kiến đóng góp quý báu của quý thầy cô để kiến thức của em trong lĩnh vực này được hoàn thiện hơn đồng thời có điều kiện bổ sung, nâng cao kỹ năng nghề nghiệp của em.

Em xin chân thành cảm ơn!

Hà Nội, ngày 06 tháng 05 năm 2023 Sinh viên

**Nguyễn Lương Hùng**

# MỤC LỤC

[**LỜI MỞ ĐẦU** **3**](#_heading=h.gjdgxs)

[**MỤC LỤC** **4**](#_heading=h.30j0zll)

[**DANH MỤC CÁC CHỮ VIẾT TẮT** **6**](#_heading=h.1fob9te)

[**DANH MỤC BẢNG BIỂU** **7**](#_heading=h.3znysh7)

[**DANH MỤC HÌNH ẢNH** **8**](#_heading=h.2et92p0)

[**Chương I: Tổng quan về đề tài** **10**](#_heading=h.tyjcwt)

* 1. [Giới thiệu đề tài 10](#_heading=h.3dy6vkm)
  2. [Lý do chọn đề tài 10](#_heading=h.1t3h5sf)
  3. [Đối tượng nghiên cứu của đề tài 11](#_heading=h.4d34og8)
  4. [Ý nghĩa khoa học và thực tiễn của đề tài 11](#_heading=h.2s8eyo1)

[**Chương II: Cơ sở lý thuyết thực hiện bài toán Speech to text** **13**](#_heading=h.17dp8vu)

* 1. [Định nghĩa 13](#_heading=h.3rdcrjn)
  2. [Ứng dụng 13](#_heading=h.26in1rg)
  3. [Các dạng triển khai chính của bài toán Speech to text 15](#_heading=h.lnxbz9)
  4. [Các phương pháp tiếp cận 16](#_heading=h.35nkun2)

[**Chương III: Các mô-đun thực hiện bài toán Speech to text** **18**](#_heading=h.1ksv4uv)

* 1. [Cách hình thành mô hình bài toán Speech to text 18](#_heading=h.44sinio)
  2. [Tiền xử lý dữ liệu âm thanh (Pre-processing audio) 19](#_heading=h.2jxsxqh)
  3. [Mô hình âm thanh của bài toán Speech to text (Acoustic/Audio model) 21](#_heading=h.z337ya)
  4. [Mô hình ngôn ngữ (Language model) 22](#_heading=h.3j2qqm3)
  5. [Từ điển phát âm (pronunciation dictionary) 24](#_heading=h.1y810tw)
  6. [Giải mã (decoding) 24](#_heading=h.4i7ojhp)

[**Chương IV: Các hướng tiếp cận của bài toán Speech to text** **26**](#_heading=h.2xcytpi)

* 1. [Tín hiệu âm thanh giọng nói và đặc trưng âm thanh giọng nói (Signal and Feature of speech) 26](#_heading=h.1ci93xb)
     1. [Tín hiệu âm thanh giọng nói 26](#_heading=h.3whwml4)
     2. [Đặc trưng âm thanh giọng nói 27](#_heading=h.2bn6wsx)
  2. [Kiến trúc mô hình học sâu và các nhiệm vụ xử lý âm thanh giọng nói..](#_heading=h.qsh70q)

. 28

* + 1. [Mạng nơ-ron nhân tạo (Neural Networks) 28](#_heading=h.3as4poj)
    2. Mạng tích chập nơ-ron nhân tạo (Convolutional Neural Network

- CNN) 29

* + 1. [Mạng nơ-ron hồi quy (Recurrent Neural Networks - RNNs) 30](#_heading=h.1pxezwc)
    2. [Bộ nhớ dài- ngắn hạn (Long-Short Term Memory - LSTM) 34](#_heading=h.49x2ik5)
    3. [Mô hình Sequence to Sequence (Seq2seq) 38](#_heading=h.2p2csry)
       1. [Cấu trúc, cơ chế hoạt động và hạn chế của mô hình Seq2seq](#_heading=h.147n2zr)

. 38

* + - 1. [Cơ chế Attention trong mô hình seq2seq 41](#_heading=h.3o7alnk)
    1. [Mô hình Transformer 43](#_heading=h.23ckvvd)
       1. Cơ chế Attention trong mô hình Transformer 43
       2. [Cấu trúc của mô hình Transformer 47](#_heading=h.32hioqz)
  1. [Hàm mất mát (Loss function) 54](#_heading=h.1hmsyys)
     1. [Định nghĩa 54](#_heading=h.41mghml)
     2. [Connectionist Temporal Classification (CTC loss) trong bài toán speech to text 56](#_heading=h.2grqrue)
  2. [Phương pháp đánh giá 58](#_heading=h.vx1227)
     1. [Word Error Rate (WER) 59](#_heading=h.3fwokq0)
     2. [Character Error Rate (CER) 59](#_heading=h.1v1yuxt)
     3. [Beam-search output 60](#_heading=h.4f1mdlm)
     4. [Greedy search 61](#_heading=h.2u6wntf)

[**Chương V: Thực nghiệm** **63**](#_heading=h.19c6y18)

* 1. [Môi trường Anaconda 63](#_heading=h.3tbugp1)
  2. [Ngôn ngữ Python 64](#_heading=h.28h4qwu)
  3. [Các thư viện chính sử dụng trong bài toán 66](#_heading=h.nmf14n)
     1. [Thư viện Pytorch 66](#_heading=h.37m2jsg)
     2. [Thư viện torch audio 68](#_heading=h.1mrcu09)
     3. [Thư viện transformer 69](#_heading=h.46r0co2)
     4. [Thư viện FastAPI 72](#_heading=h.2lwamvv)
  4. [Bộ dữ liệu huấn luyện 73](#_heading=h.111kx3o)
     1. [Bộ dữ liệu âm thanh LJSpeech 73](#_heading=h.3l18frh)
     2. [Bộ dữ liệu VIVOS 75](#_heading=h.206ipza)
  5. [Xây dựng hệ thống 77](#_heading=h.4k668n3)
     1. [Huấn luyện mô hình Transformer với hàm mất mát là CTC loss:..](#_heading=h.2zbgiuw)

. 78

* + 1. [Kết quả huấn luyện của mô hình 84](#_heading=h.1egqt2p)
    2. [Thử nghiệm mô hình và triển khai mô hình 85](#_heading=h.3ygebqi)
    3. [Triển khai mô hình với FastAPI 87](#_heading=h.2dlolyb)

[**KẾT LUẬN** **90**](#_heading=h.sqyw64)

1. [Kết quả đạt được 90](#_heading=h.3cqmetx)
2. [Hướng phát triển 90](#_heading=h.1rvwp1q)

[**TÀI LIỆU THAM KHẢO** **91**](#_heading=h.4bvk7pj)

# DANH MỤC CÁC CHỮ VIẾT TẮT

| Từ viết tắt | Giải thích |
| --- | --- |
| API | Application Programming Interface |
| AI | Artificial Intelligence |
| ML | Machine Learning |
| ASR | Automatic SPeech Recognition |
| WER | Word Error Rate |
| CER | Character Error Rate |
| CNN | Convolutional Neural Network |
| RNN | Recurrent Neural Network |
| LSTM | Long-Short Term Memory |
| RMI | Remote Method Invocation |
| FFN | Feed Forward Network |
| CTC loss | Connectionist Temporal Classification loss |
| FCN | Fully Connected Network |
| GRU | Gated Recurrent Unit |
| LM | Language Model |

# DANH MỤC BẢNG BIỂU

Bảng 1: Bảng kết quả WER khi có mô hình ngôn ngữ và khi không có mô hình ngôn ngữ 83

# DANH MỤC HÌNH ẢNH

Hình 1.1: Mô tả hoạt động của đề tài 11

Hình 1.2: Ý nghĩa của đề tài 12

Hình 2.1: Ứng dụng Transkriptor 14

Hình 2.2: Nền tảng Amazon transcribe medical với Speech to text 15

Hình 2.3: Các phương pháp tiếp cận của bài toán Speech to text 16

Hình 3.1: Các bước tính toán của quá trình tiếp cận thống kê 19

Hình 3.2: Các bước thực hiện của quá trình tiếp cận thống kê 19

Hình 3.3: Quy trình (pipeline) xử lý âm thanh cơ bản 21

Hình 3.4: Mô tả quá trình thực hiện của mô hình âm thanh 22

Hình 3.5: Mô tả kết quả thực hiện của mô hình âm thanh 22

Hình 4.1: Tín hiệu âm thanh giọng nói 27

Hình 4.2: Time-domain và frequency-domain của âm thanh giọng nói 27

Hình 4.3: Hình ảnh biểu diễn mạng nơ-ron nhân tạo 28

Hình 4.4: Minh họa cấu trúc của mạng CNN 30

Hình 4.5: Cấu trúc cơ bản của RNN 31

Hình 4.6: Biểu diễn các công thức tính toán của một khối trong RNN 32

Hình 4.7: Các cấu trúc của một số dạng RNN 32

Hình 4.8: Cấu trúc của RNN với hàm kích hoạt tanh 35

Hình 4.9: Cấu trúc của LSTM với hàm kích hoạt sigmoid và tanh 35

Hình 4.10: Minh họa cấu trúc của một cell state 35

Hình 4.11: Minh họa cấu trúc của GRU 37

Hình 4.12: Hình ảnh cấu trúc của mô hình seq2seq 38

Hình 4.13: Cơ chế dựa vào weight global và weight local 41

Hình 4.14: Cấu trúc của mô hình seq2seq có sử dụng attention 42

Hình 4.15: Miêu tả hoạt động của self-attention 45

Hình 4.16: Mô tả nhiệm vụ và chức năng của các vector query, key, value 46

Hình 4.17: Mô tả một multi-head attention 46

Hình 4.18: Hình ảnh mô tả cấu trúc của mô hình Transformer 48

Hình 4.19: Hình ảnh mô tả sự liên kết giữa decoder và encoder 49

Hình 4.20: Vị trí của positional encoder trong mô hình Transformer 49

Hình 4.21: Hình ảnh vị trí của lớp masked multi-head attention 51

Hình 4.22: Hình ảnh biểu diễn kết quả của minh họa của CTC - loss 58

Hình 5.1: Logo Anaconda 63

Hình 5.2: Logo python 64

Hình 5.3: Logo Pytorch 66

Hình 5.4: Logo torch audio 68

Hình 5.5: Logo transformer 70

Hình 5.6: Logo FastAPI 72

Hình 5.7: Hình ảnh bộ dữ liệu LJSpeech 73

Hình 5.8: Chi tiết về bộ dữ liệu LJSpeech 74

Hình 5.9: Hình ảnh bộ dữ liệu VIVOS 75

Hình 5.10: Chi tiết về bộ dữ liệu VIVOS 77

Hình 5.11: Thông tin về bộ dữ liệu VIVOS 77

Hình 5.12: Hình ảnh minh họa luồng hoạt động của bài toán 78

Hình 5.13: Hình ảnh đồ thị trực quan âm thanh dưới dạng time-domain 79

Hình 5.14: Hình ảnh minh họa về sample rate 80

Hình 5.15: Spectrogram của file âm thanh 81

Hình 5.16: Kết quả huấn luyện mô hình 84

Hình 5.17: Giao diện Fast API 89

Hình 5.18: Output cuối cùng nhận được của API 89

## Chương I: Tổng quan về đề tài

### Giới thiệu đề tài.

Đề tài chuyển đổi ngôn ngữ âm thanh thành văn bản hay nói cách khác là máy dịch âm thanh là một chủ đề rất nổi bật và xu hướng phát triển của lĩnh vực khoa học máy tính và trí tuệ nhân tạo. Nó liên quan đến việc sử dụng các kỹ năng phân tích âm thanh cũng như xử lý ngôn ngữ tự nhiên vô cùng phức tạp. Những kết quả của nghiên cứu này hoàn toàn có thể được áp dụng trong các buổi họp hay các buổi hội thảo. Việc ghi lại nội dung buổi họp hay nói cách khác là một thư ký ảo ghi lại các nội dung trong cuộc họp.

Một số ứng dụng của đề tài này bao gồm: phân biệt ngôn ngữ, chuyển đổi âm thanh thành văn bản.

Đề tài này đòi hỏi sự kết hợp giữa các kỹ thuật như xử lý và phân tích âm thanh, học máy, và trí tuệ nhân tạo. Nó cũng đòi hỏi sự sáng tạo trong việc thiết kế hệ thống và các thuật toán để đạt được kết quả tốt nhất.

Thông qua đó việc tiếp cận đối với bài toán trên cũng có nhiều hướng khác nhau. Trong bài báo cáo này, chúng ta chỉ tập trung vào việc phân tích âm thanh và chuyển đổi thành văn bản, bài toán chuyển đổi âm thanh thành văn bản được gọi là bài toán Speech to Text

### Lý do chọn đề tài.

Mô hình chuyển đổi ngôn ngữ từ âm thanh thành văn bản tiếng Việt với cơ chế Attention trong Transformer tập trung vào các nội dung tìm hiểu chính như sau:

* + - Machine learning
    - Model language
    - Attention mechanism
    - Transformer model
    - Deploy a model with FastAPI

Việc ứng dụng chuyển đổi âm thanh thành văn bản là rất cần thiết, dù không thể hoàn toàn thay thế con người nhưng cũng giúp chúng ta phần nào giảm bớt được việc phải viết lại văn bản khi nghe âm thanh:

* + - Đóng vai trò như một thư ký trong cuộc họp, ghi lại thông tin, các ý kiến của từng người tham gia.
    - Tăng hiệu quả của buổi họp: Việc ghi lại các nội dung cuộc họp với các chi tiết tương ứng.
    - Giảm lượng công việc con người: Thay vì phải liên tục viết thông tin thì giờ thư ký chỉ cần sửa một số từ là hoàn thiện phần văn bản.

Không chỉ vậy, khi mà chúng ta dùng các video không có phụ đề, bạn hoàn toàn có thể dùng âm thanh để xuất ra văn bản sau đó dịch văn bản đó để hiểu được nội dung của video đó.

Hình 1.1: Mô tả hoạt động của đề tài.

Vì vậy, tính cấp thiết của đề tài “Xây dựng mô hình chuyển đổi ngôn ngữ từ âm thanh thành văn bản tiếng Việt với cơ chế Attention trong Transformer” là rất cao và đang thu hút sự quan tâm của nhiều nhà nghiên cứu và doanh nghiệp.

### Đối tượng nghiên cứu của đề tài.

Trong phạm vi đề tài và bài báo cáo, em tiến hành các nội dung sau

* + - Định nghĩa và các đặc trưng của bài toán Speech to text.
    - Các hướng tiếp cận chính cho bài toán Speech to text.
    - Kết quả thực nghiệm.

### Ý nghĩa khoa học và thực tiễn của đề tài.

Về mặt thực tiễn, bài toán Speech to text có ý nghĩa rất lớn trong việc phân tích âm thanh và chuyển đổi ngôn ngữ âm thanh thành dạng văn bản, giảm thiếu công việc của con người, hỗ trợ trực tiếp trong các cuộc họp hoặc các buổi thuyết trình, dịch các âm thanh của video để tiến hành dịch, mang lại trải nghiệm tốt cho người dùng.



Hình 1.2: Ý nghĩa của đề tài

Về mặt khoa học, đề tài đã tổng kết các thuật toán tiêu biểu trong việc phân tích âm thanh và chuyển đổi âm thanh thành văn bản, cả truyền thống lẫn hiện đại, thực hiện việc phân tích thuật toán và đánh giá khả năng của từng thuật toán. Bên cạnh đó, đề tài đã thử nghiệm áp dụng mô hình Transformer với cơ chế Attention vào bài toán cho kết quả khá tốt, thể hiện ra được tính ứng dụng lớn của bài toán và có thể phát triển xa hơn nữa.

## Chương II: Cơ sở lý thuyết thực hiện bài toán Speech to text.

### Định nghĩa.

Speech to text, còn được gọi là speech recognition, là công nghệ cho phép máy tính hoặc thiết bị chuyển lời nói thành văn bản viết. Nó liên quan đến việc xử lý đầu vào âm thanh từ micrô hoặc các nguồn khác, nhận dạng các từ được nói và chuyển đổi chúng thành văn bản. Công nghệ này có nhiều ứng dụng, bao gồm các trợ lý kích hoạt bằng giọng nói như Siri và Alexa, phần mềm phiên âm cho các cuộc họp và phỏng vấn cũng như các công cụ trợ năng cho người khiếm thính. Công nghệ chuyển giọng nói thành văn bản đã được cải thiện đáng kể trong những năm gần đây và tiếp tục phát triển, mang lại kết quả ngày càng chính xác và đáng tin cậy.

Theo như Ilias Papastratis có viết trong trong bài viết “*Speech Recognition: a review of the different deep learning approaches”*: Nhận dạng giọng nói tự động (ASR) đề cập đến nhiệm vụ nhận dạng giọng nói của con người và dịch nó thành văn bản . Lĩnh vực nghiên cứu này đã đạt được rất nhiều sự tập trung trong những thập kỷ qua. Đây là một lĩnh vực nghiên cứu quan trọng về giao tiếp giữa người với máy. Các phương pháp ban đầu tập trung vào trích xuất tính năng thủ công và các kỹ thuật thông thường như Mô hình hỗn hợp Gaussian (GMM) , thuật toán Độ cong thời gian động (DTW) và Mô hình Markov ẩn (HMM) .

Gần đây, các mạng thần kinh như mạng thần kinh tái phát (RNN), mạng thần kinh tích chập (CNN) và trong những năm gần đây Transformers, đã được áp dụng trên ASR và đã đạt được hiệu suất tuyệt vời.

### Ứng dụng.

Hiện nay trên thế giới đã phát triển các hệ thống hỗ trợ chuyển đổi âm thanh thành văn bản. Tiêu biểu một số các phần mềm dưới đây

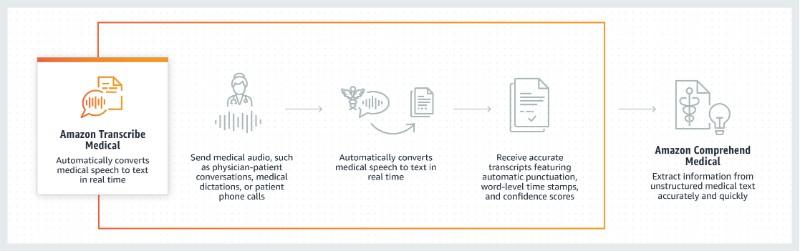
* + - Transkriptor: Chuyển đổi âm thanh thành văn bản: Tiện ích mở rộng Chrome được hỗ trợ bởi AI giúp chuyển đổi giọng nói thành tệp văn bản. Transkriptor cung cấp trải nghiệm phiên âm tự động tốt nhất cho các cuộc họp của bạn. Transkriptor tự động ghi chú cho bạn bằng hơn 100 ngôn ngữ và phương ngữ, bao gồm tiếng Anh, Pháp, Đức, Tây Ban Nha, Trung Quốc và Bồ Đào Nha. Tiện ích mở rộng Transkriptor Chrome rất dễ sử dụng. Nhấn ghi trước bất kỳ cuộc họp ảo nào và để

Transkriptor phát huy tác dụng của nó. Bạn có thể sử dụng Transkriptor cho bất kỳ cuộc họp nào trong Google Meet, Microsoft Teams và Cisco Webex. Transkriptor cũng có sẵn trên Web, iOS và Android.



Hình 2.1: Ứng dụng Transkriptor

* + - AMZS2T: Trình chuyển đổi âm thanh thành văn bản là một phần mềm chép lời tự động nhận dạng giọng nói và chép lời nói thành định dạng văn bản tương đương. Thông thường, con người sẽ nghe tệp âm thanh rồi gõ lại vào một tệp văn bản để sử dụng lại nội dung lời nói cho các phương tiện truyền thông khác nhau. Nhưng giờ đây, với trí tuệ nhân tạo, máy tính có thể dễ dàng chuyển đổi âm thanh thành văn bản trong thời gian ngắn và đưa nội dung vào sử dụng để phục vụ các mục đích khác nhau như tìm kiếm, tạo phụ đề và thu thập thông tin chuyên sâu. Trình chuyển đổi âm thanh thành văn bản là một phần mềm chép lời tự động nhận dạng giọng nói và chép lời nói thành định dạng văn bản tương đương. Thông thường, con người sẽ nghe tệp âm thanh rồi gõ lại vào một tệp văn bản để sử dụng lại nội dung lời nói cho các phương tiện truyền thông khác nhau. Nhưng giờ đây, với trí tuệ nhân tạo, máy tính có thể dễ dàng chuyển đổi âm thanh thành văn bản trong thời gian ngắn và đưa nội dung vào sử dụng để phục vụ các mục đích khác nhau như tìm kiếm, tạo phụ đề và thu thập thông tin chuyên sâu.



Hình 2.2: Nền tảng Amazon transcribe medical với Speech to text

Những sản phẩm và giải pháp này đang được phát triển và cải tiến liên tục để đáp ứng nhu cầu của thị trường bán lẻ ngày càng cạnh tranh và phức tạp hơn.

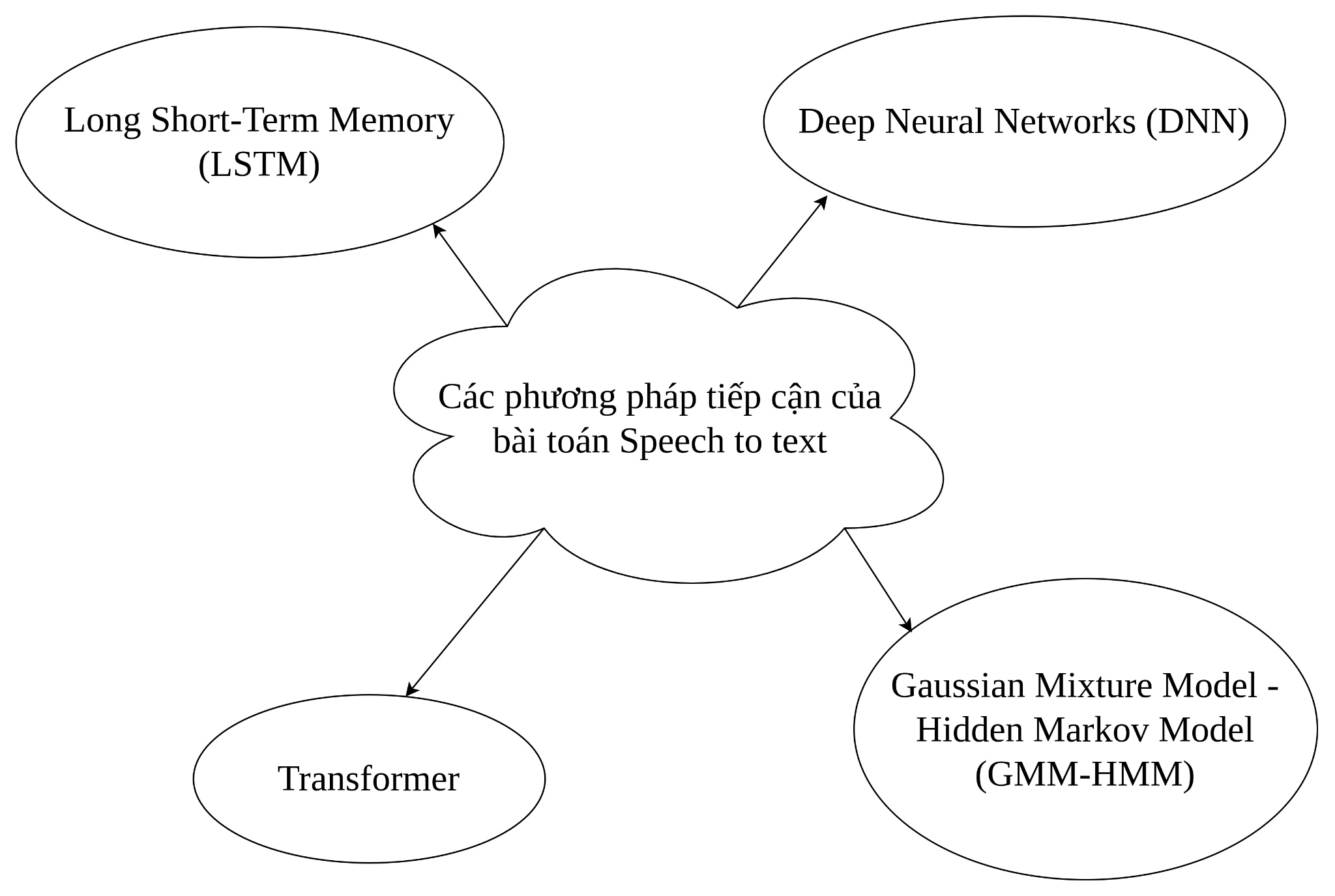
### Các dạng triển khai chính của bài toán Speech to text.

Các mô hình và hướng tiếp cận của bài toán bài toán Speech to text, bài toán Speech to text có thể được chia ra nhiều dạng khác nhau:

* + - Real-time STT: Đây là dạng STT mà hệ thống phải xử lý tín hiệu âm thanh ngay khi nó được ghi âm. Ví dụ như các ứng dụng ghi âm cuộc gọi điện thoại, trò chuyện trực tuyến, hoặc các ứng dụng chuyển đổi âm thanh sang văn bản ngay lập tức như các trợ lý ảo hoặc các hệ thống ghi chú.
    - Batch STT: Đây là dạng STT mà dữ liệu âm thanh được ghi âm sẵn trong một khoảng thời gian và sau đó được xử lý bằng các mô hình STT. Ví dụ như các ứng dụng chuyển đổi các file âm thanh như podcast, video, hoặc các tài liệu âm thanh sang văn bản.

Cả hai dạng STT đều có những ứng dụng và thách thức riêng. Với real-time STT, hệ thống phải xử lý tín hiệu âm thanh ngay lập tức và đưa ra kết quả trong thời gian ngắn, trong khi với batch STT, hệ thống có thể có nhiều thời gian để xử lý dữ liệu âm thanh và đưa ra kết quả chính xác hơn. Tuy nhiên, batch STT cũng đòi hỏi phải xử lý một lượng dữ liệu lớn hơn, và có thể mất nhiều thời gian để hoàn thành quá trình xử lý.

### Các phương pháp tiếp cận.



Hình 2.3: Các phương pháp tiếp cận của bài toán Speech to text

Có nhiều phương pháp tiếp cận khác nhau để giải quyết bài toán Speech-to-Text (STT), tùy thuộc vào các yêu cầu và điều kiện cụ thể của ứng dụng. Sau đây là một số phương pháp tiếp cận thường được sử dụng:

* + - Gaussian Mixture Model - Hidden Markov Model (GMM-HMM): Phương pháp này sử dụng mô hình Gaussian để mô hình hóa âm thanh và mô hình HMM để mô hình ngữ cảnh từ. Đây là phương pháp cổ điển nhất trong lĩnh vực Speech to text và vẫn được sử dụng rộng rãi.
    - Deep Neural Networks (DNN): Phương pháp này sử dụng mạng nơ-ron sâu để mô hình hóa âm thanh và ngữ cảnh từ. Các mô hình DNN được huấn luyện trên các tập dữ liệu lớn và có khả năng xử lý các đặc trưng phức tạp của âm thanh.
    - Long Short-Term Memory (LSTM): Phương pháp này sử dụng mạng nơ-ron LSTM để mô hình hóa âm thanh và ngữ cảnh từ. LSTM có khả năng xử lý các chuỗi dài và đưa ra dự đoán dựa trên các thông tin liên quan trong quá khứ.
    - Transformer: Phương pháp này sử dụng kiến trúc Transformer để mô hình hóa âm thanh và ngữ cảnh từ. Transformer có khả năng xử lý các chuỗi dài và đưa ra dự đoán dựa trên sự tương tác giữa các từ trong câu.

Kết hợp các phương pháp: Nhiều phương pháp tiếp cận có thể được kết hợp để cải thiện độ chính xác và hiệu suất của hệ thống Speech to text.

Các phương pháp này có ưu điểm và hạn chế riêng, và việc lựa chọn phương pháp phù hợp sẽ phụ thuộc vào nhiều yếu tố, bao gồm độ chính xác yêu cầu, khả năng tính toán, và độ phức tạp của ứng dụng.

## Chương III: Các mô-đun thực hiện bài toán Speech to text.

Ở chương II, chúng ta đã có góc nhìn tổng quan về bài toán Speech to text, khái niệm, ứng dụng, các dạng, tính đặc trưng và phương pháp tiếp cận. Bài toán Speech to text thông thường sẽ bao gồm các module chính, đó là:

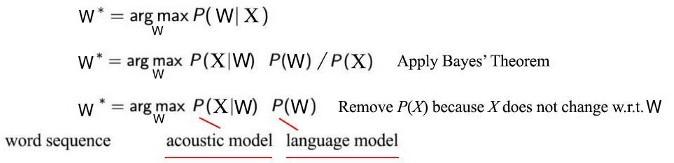
* + - Pre-processing module: Module này có nhiệm vụ tiền xử lý dữ liệu âm thanh trước khi đưa vào các mô hình Speech to text. Module này bao gồm các bước như lọc nhiễu, cân bằng âm lượng, cắt bỏ khoảng trắng, và tách các đoạn tiếng nói riêng lẻ.
    - Acoustic modeling module: Module này sử dụng các mô hình học máy để mô hình hóa các đặc trưng âm thanh, như amplitude, frequency, và spectral features, và xác định các đơn vị âm vị (phonemes) trong tín hiệu âm thanh. Các phương pháp phổ biến trong module này bao gồm Gaussian mixture models và deep neural networks.
    - Language modeling module: Module này sử dụng các mô hình ngôn ngữ để mô hình hóa cấu trúc ngữ pháp và từ vựng của ngôn ngữ đang được nói. Module này có nhiệm vụ giúp đưa ra các dự đoán chính xác cho các từ tiếp theo trong câu dựa trên ngữ cảnh từ trong câu và đối chọi với các từ đã biết.
    - Decoding module: Module này sử dụng các thuật toán để lựa chọn các giải thích tốt nhất cho tín hiệu âm thanh đã được mã hóa thành văn bản. Module này cũng đảm nhiệm nhiệm vụ tạo ra kết quả đầu ra cuối cùng của hệ thống STT.
    - Post-processing module: Module này có nhiệm vụ xử lý kết quả đầu ra của hệ thống STT để cải thiện độ chính xác và độ tin cậy của kết quả. Các phương pháp post-processing bao gồm xử lý trùng lặp, sửa lỗi chính tả và xử lý các từ không chính xác.

Trong chương này, chúng ta sẽ tìm hiểu về các module trên.

### Cách hình thành mô hình bài toán Speech to text.

Mục tiêu chính của cách tiếp cận này là xây dựng một mô hình thống kê để dự đoán các chuỗi văn bản (W) dựa trên các vector đặc trưng (X)

Có thể giải thích mô hình thống kê như sau: Tìm kiếm tất cả các chuỗi từ có thể có (hạn chế độ dài tối đa) và tìm một chuỗi từ phù hợp nhất với đặc điểm âm thanh đầu vào



Trong đó:

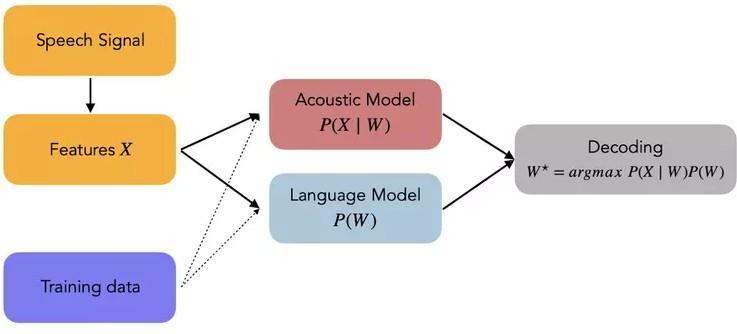
Hình 3.1: Các bước tính toán của quá trình tiếp cận thống kê.

* 𝑎𝑟𝑔𝑚𝑎𝑥 là không gian tìm kiếm, một chức năng của bộ từ vựng.

𝑤

* 𝑃(𝑋|𝑊): là mô hình âm thanh.
* 𝑃(𝑊): là mô hình ngôn ngữ

PỏCác bước thực hiện với mô hình được miêu tả qua các bước sau:



Hình 3.2: Các bước thực hiện của quá trình tiếp cận thống kê.

### Tiền xử lý dữ liệu âm thanh (Pre-processing audio)

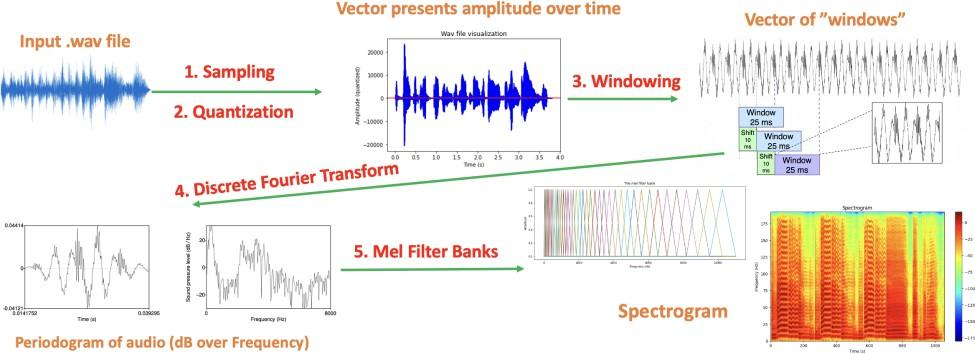
Pre-processing module trong bài toán Speech-to-Text (STT) là một bước quan trọng để chuẩn bị dữ liệu âm thanh trước khi đưa vào các mô hình STT. Module này bao gồm các bước sau:

* + - Loại bỏ nhiễu: Dữ liệu âm thanh thường bị ảnh hưởng bởi nhiễu từ các nguồn bên ngoài như tiếng ồn, tiếng động, tiếng gió, v.v. Các kỹ thuật lọc như Butterworth hay Wiener filter có thể được sử dụng để giảm thiểu nhiễu.
    - Cân bằng âm lượng: Dữ liệu âm thanh thường có độ lớn khác nhau và không đồng đều, do đó cân bằng âm lượng giúp đưa tất cả các phần của tín hiệu âm thanh về mức độ tương đương.
    - Tách khoảng trắng: Nếu tín hiệu âm thanh chứa các khoảng trắng không cần thiết giữa các từ, chúng cần được loại bỏ để cải thiện độ chính xác của STT.
    - Tách các đoạn tiếng nói riêng lẻ: Module này sẽ giúp phát hiện và tách các đoạn tiếng nói riêng lẻ trong dữ liệu đầu vào, giúp cho việc phân tích và mã hóa dữ liệu trở nên dễ dàng hơn.
    - Chuyển đổi tần số: Một số mô hình STT cần dữ liệu đầu vào được chuyển đổi sang tần số để đưa vào xử lý. Các kỹ thuật chuyển đổi tần số như Discrete Fourier Transform (DFT) và Fast Fourier Transform (FFT) có thể được sử dụng để thực hiện việc này.
    - Tách kênh âm thanh: Nếu tín hiệu âm thanh có nhiều kênh (ví dụ như âm thanh stereo), tách kênh âm thanh giúp phân tích và mã hóa chúng độc lập với nhau.

Các bước tiền xử lý này giúp cho dữ liệu âm thanh được chuẩn bị tốt hơn trước khi đưa vào các mô hình STT, từ đó cải thiện độ chính xác và hiệu suất của hệ thống STT.

Để giải quyết bất kì bài toán học máy liên quan âm thanh, ta cần tìm hiểu về một số kĩ thuật xử lý âm thanh để có thể trích xuất đặc trưng một các phù hợp. Vì vậy trong phần đầu tiên, ta sẽ tìm hiểu qua một số bước xử lý cơ bản nhất và một số dạng đặc trưng âm thanh mà ta có thể trích xuất được từ một tệp âm thanh bất kì. Về mặt tổng quan, có rất nhiều phương pháp để xử lý tín

hiệu âm thanh, với mỗi cách thức sẽ có một ý nghĩa, ứng dụng, độ hiệu quả khác nhau cho từng bài toán.



Hình 3.3: Quy trình (pipeline) xử lý âm thanh cơ bản

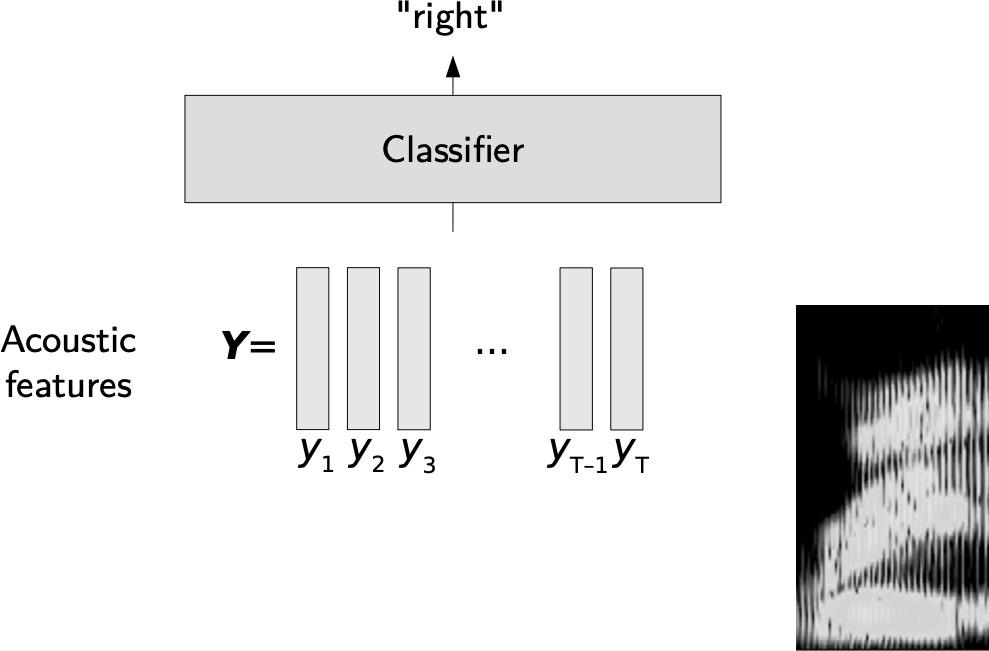
### Mô hình âm thanh của bài toán Speech to text (Acoustic/Audio model).

Mô hình âm thanh là một mô hình phức tạp, có nhiệm vụ mô hình hóa mối quan hệ giữa tín hiệu âm thanh và các đơn vị ngữ âm trong ngôn ngữ. Có nhiều thách thức trong việc khởi tạo mô hình liên quan tới tín hiệu âm thanh như các dạng sóng âm thanh của lời nói của con người. Bao gồm các ngữ điệu, khiến cho sóng âm trở nên đặc biệt. Không chỉ riêng âm thanh của con người mà còn có cả những tiếng ồn xung quanh gọi lại nhiễu (noise).

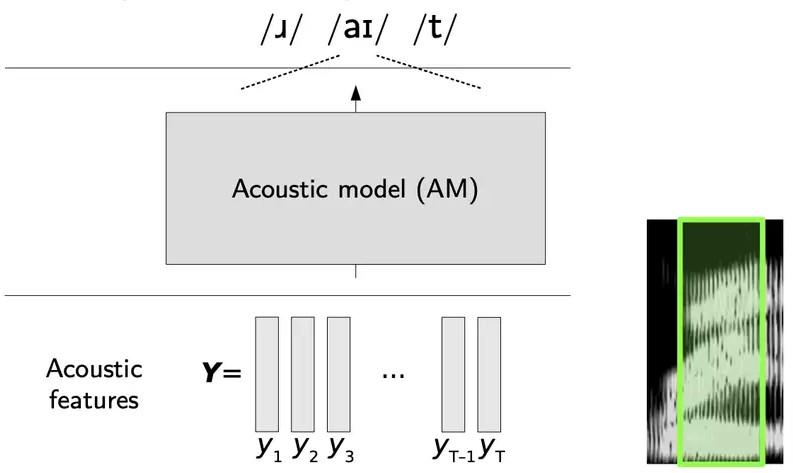
Mỗi môi trường đều có âm thanh và điểm đặc trưng riêng biệt. Ví dụ âm thanh trong phòng học chắc chắn sẽ khác nhiều so với âm thanh của ca sĩ trong phòng thu âm có chống ồn. Do đó, mô hình âm thanh là một vấn đề xử lý tín hiệu cũng như mô hình hóa vô cũng phức tạp và gặp nhiều khó khăn. Trong đó tín hiệu giọng nói thô cần được lọc ra khỏi các âm thanh nhiễu và không cần thiết.

Dạng sóng âm của mỗi từ hoặc âm vị của mỗi người là khác nhau. Mỗi người đều có cách nói cũng như âm điệu khác nhau. Thậm chí một câu với cùng một người thì tùy từng tâm trạng (tức giận, buồn bã, vui vẻ, phấn khích) sẽ có giọng điệu và âm lượng khác nhau.

Một mô hình âm thanh tốt là phải có đủ độ chính xác, độ nhạy để phát hiện những âm thanh của các âm vị khác nhau. Hình 3.3 thể hiện rõ nhiệm vụ của mô hình âm thanh trong bài toán Speech to text



Hình 3.4: Mô tả quá trình thực hiện của mô hình âm thanh



Hình 3.5: Mô tả kết quả thực hiện của mô hình âm thanh

### Mô hình ngôn ngữ (Language model).

Mô hình âm thanh (acoustic model) được miêu tả như một chức năng để lắng nghe âm thanh, còn mô hình ngôn ngữ được miêu tả như một bộ não của con người, nhằm đọc hiểu và phân tích âm thanh đó. Language model có thể được huấn luyện trên nhiều loại dữ liệu khác nhau, bao gồm các bài báo, tài liệu kỹ thuật, chuyện trò, tin tức, v.v. Hệ thống nhận dạng giọng nói tự động

hiện nay thường sử dụng các mô hình ngôn ngữ phức tạp, sử dụng các kỹ thuật Deep Learning và Neural Networks để đạt được độ chính xác cao trong việc dự đoán các từ tiếp theo. Do đó, mô hình ngôn ngữ cần được cập nhật thường xuyên. Ví dụ khi ta nghe tiếng Việt, đôi khi chúng ta không nghe rõ từ (do mô hình âm thanh chưa đủ tốt) nhưng ta vẫn đoán được gần như chính xác từ ta không nghe được, đó là nhờ có mô hình ngôn ngữ. Mô hình ngôn ngữ dựa vào những thông tin trước đó về ngữ pháp đã giúp ta dự đoán được. Hoặc ở ví dụ, mô hình âm thanh khó có thể phân biệt 2 câu dưới đây do cách phát âm quá giống nhau.

VD: “Anh ấy thích uống nước lạnh” và “Anh ấy thích uốn nước lạnh”. Nhìn qua ta có thể chắc chắn là câu thứ nhất là đúng. Đó là do mô hình ngôn ngữ sẽ giúp ta lựa chọn câu đúng nhất. Mô hình ngôn ngữ đơn giản và phổ biến hiện nay là mô hình n-gram. Trong mô hình này, xác suất của từ thứ n được xác định dựa trên (n-1) từ đứng trước nó P(Wn|Wn-1...W1). N thường bằng 3 hoặc 4 cho các hệ thống nhận dạng tiếng nói hiện nay. Mô hình ngôn ngữ n-gram tuy đơn giản nhưng cũng có nhược điểm là không mô tả được sự phụ thuộc dài do giới hạn của n. Hiện nay có một hướng nghiên cứu là sử dụng mạng nơ ron hồi quy (RNN) để mô tả mô hình ngôn ngữ.

Tổng quan lại, language model là một thành phần quan trọng trong hệ thống nhận dạng giọng nói tự động, giúp cải thiện độ chính xác của kết quả nhận dạng. Mô hình ngôn ngữ được huấn luyện trên một tập dữ liệu lớn các văn bản đã biết trước, và sử dụng các kỹ thuật Deep Learning để đạt được độ chính xác cao trong việc dự đoán các từ tiếp theo trong câu.

Để huấn luyện một mô hình ngôn ngữ, cần có một tập dữ liệu lớn các văn bản, gọi là corpus, và một kiến trúc mô hình phù hợp để huấn luyện. Một số kiến trúc mô hình phổ biến được sử dụng để huấn luyện mô hình ngôn ngữ là LSTM (Long Short-Term Memory), GRU (Gated Recurrent Unit) và Transformer.

Mô hình ngôn ngữ là một thành phần quan trọng của trong hệ thống nhận dạng giọng nói, giúp cải thiện độ chính xác của kết quả nhận dạng âm thanh. Tùy thuộc vào yêu cầu của hệ thống để lựa chọn mô hình phù hợp như đã nêu ở trên.

### Từ điển phát âm (pronunciation dictionary)

Mô hình âm thanh thường dùng để mô hình hóa các âm vị (phoneme). Trong khi đó, mô hình ngôn ngữ lại thường sử dụng từ (word) để mô hình hóa. Do vậy cần có cầu nối giữa 2 mô hình này đó là *từ điển phát âm.* Từ điển phát âm mô tả cách phát âm của một từ bằng các biểu diễn từ đó duwois dạng một chuỗi các âm vị.

Từ điển phát âm là một cơ sở dữ liệu chứa danh sách các từ và bản phiên âm tương ứng của chúng. Một từ điển phát âm được sử dụng để giúp hệ thống nhận ra chính xác và phiên âm các từ nghe được, nó có thể sử dụng từ điển phát âm để tra cứu phiên âm của từ và sử dụng thông tin đó để chuyển các từ thành văn bản. Hệ thống cũng có thể sử dụng từ điển để giúp phân biệt giữa các từ nghe có vẻ giống nhau nhưng có ý nghĩa khác nhau. Trong trường hợp của người Việt Nam, một từ điển phát âm sẽ bao gồm các bản phiên âm của các từ Việt Nam, có tính đến các giai điệu khác nhau và âm thanh độc đáo của ngôn ngữ. Điều này có thể giúp cải thiện tính chính xác của các bản phiên âm từ văn bản cho người Việt Nam, vì hệ thống có thể sử dụng từ điển để nhận ra và phiên âm chính xác các từ có thể khó phân biệt dựa trên âm thanh.

### Giải mã (decoding).

Mỗi tín hiệu âm thanh đầu vào X, mô hình âm thanh sẽ đưa ra các xác suất (ta có thể gọi là acoustic score). Mô hình ngôn ngữ cũng đưa ra xác suất của mình (language score). Bộ giải mã (decoder) sẽ duyệt tất cả những chuỗi text (giả thuyết) có thể đúng.

Giải mã là quá trình mà một hệ thống lời nói sang văn bản dịch đầu vào âm thanh mà nó nhận được thành văn bản. Quá trình giải mã bao gồm một số bước, bao gồm mô hình âm thanh, mô hình ngôn ngữ và ứng dụng các thuật toán thống kê. Trong giai đoạn mô hình hóa âm, hệ thống phân tích đầu vào âm thanh để xác định các đơn vị ngữ âm tạo nên từng từ được nói. Điều này liên quan đến việc phù hợp với dạng sóng âm thanh với một tập hợp các mô hình âm thanh được xác định trước, đại diện cho các âm thanh và âm vị khác nhau của ngôn ngữ. Khi các đơn vị ngữ âm đã được xác định, hệ thống sử dụng một mô hình ngôn ngữ để xác định chuỗi các từ có khả năng nhất mà người nói dự định nói. Mô hình ngôn ngữ tính đến khả năng thống kê của các từ và kết hợp từ khác nhau dựa trên bối cảnh của bài phát biểu. Cuối cùng, hệ thống áp dụng các thuật toán thống kê để chọn phiên âm có khả năng nhất của các từ được nói dựa trên các mô hình ngôn ngữ và âm thanh. Điều này có thể liên quan đến việc thực hiện các điều chỉnh và điều chỉnh dựa trên bối cảnh và bất kỳ kiến thức trước đây về người nói hoặc chủ đề. Trong trường hợp của Việt Nam, quá trình giải mã sẽ liên quan đến các bước cơ bản tương tự, nhưng với sự phức tạp thêm của kế toán cho các âm và âm thanh độc đáo của ngôn ngữ. Điều này có thể yêu cầu các mô hình ngôn ngữ và âm thanh bổ sung được thiết kế riêng cho người Việt Nam.

Trong quá trình decoding, các mô hình xử lý ngôn ngữ tự nhiên (Natural Language Processing) sẽ phân tích các đặc trưng của tín hiệu âm thanh. Các kĩ thuật decoding trong bài toán Speech to text với tiếng Việt đang được nghiên cứu và cải thiện về độ chính xác. Bao gồm việc sử dụng các mô hình học sâu (deep learning) như mạng nơ-ron nhân tạo (Convolutional Neural Network - CNN) và mạng nơ-ron hồi quy (Recurrent Neural Network - RNN)

để phân tích và mã hóa tín hiệu âm thanh giọng nói, sử dụng các mô hình ngôn ngữ để cải thiện khả năng dự đoán của mô hình.

## Chương IV: Các hướng tiếp cận của bài toán Speech to text

Học sâu (deep learning) đã trở thành một công cụ mạnh mẽ trong xử lý giọng nói vì nó có thể học được các biểu diễn phức tạp của dữ liệu giọng nói có được từ âm thanh thô (raw audio). Do đó, những tiến bộ đáng kể đã được thực hiện trong các nhiệm vụ xử lý âm thanh khác nhau, đặc biệt với âm thanh giọng nói bao gồm nhận dạng giọng nói, nhận dạng người nói, tổng hợp giọng nói…

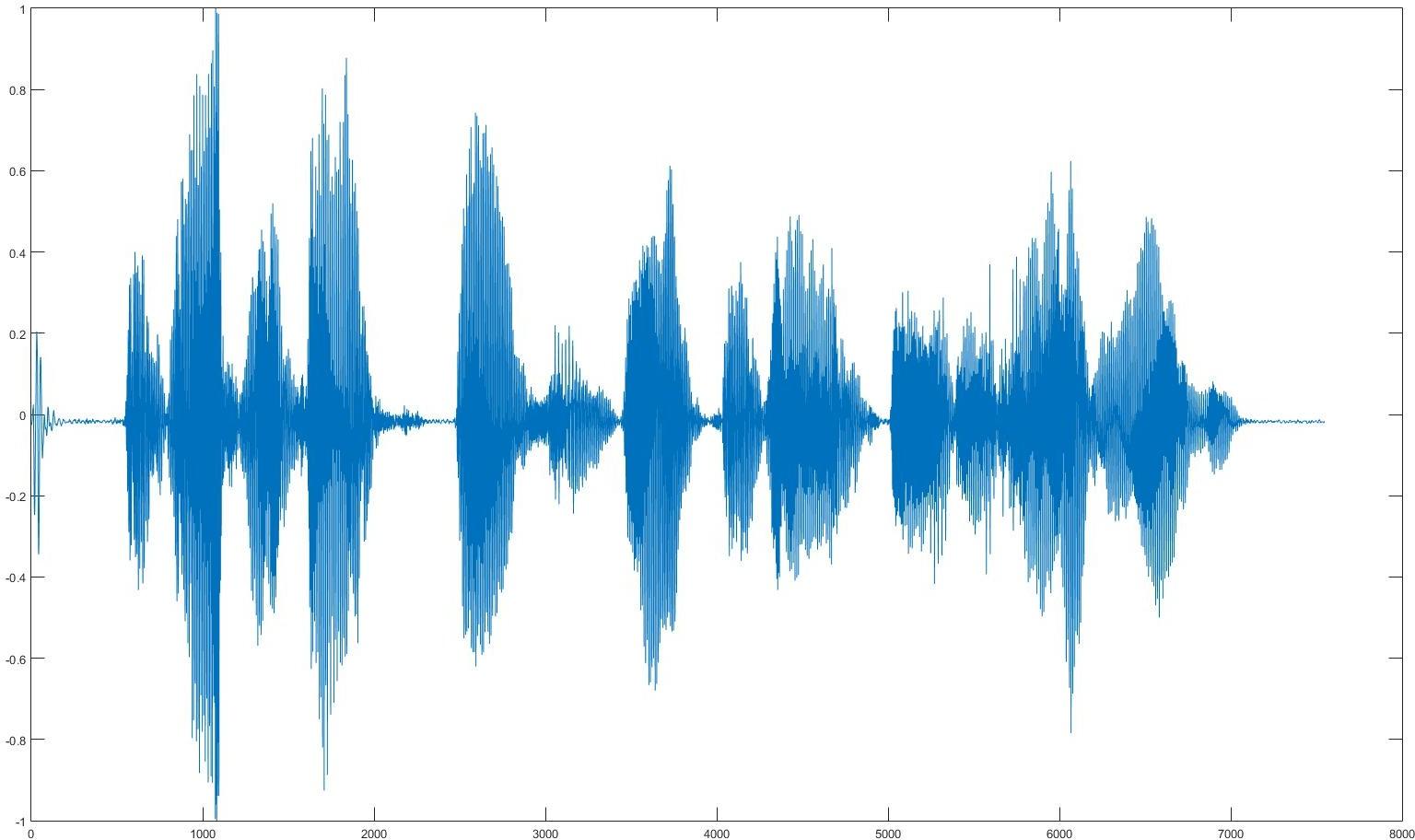
### Tín hiệu âm thanh giọng nói và đặc trưng âm thanh giọng nói (Signal and Feature of speech).

#### Tín hiệu âm thanh giọng nói.

Trong bài báo *“A Review of Deep Learning Techniques for Speech Processing”* của *AMBUJ MEHRISH* có định nghĩa như sau tính hiệu âm thanh là sự thay đổi của áp lực trong không khí, có thể được con người coi là âm thanh. Khi các tín hiệu như vậy được tạo ra bởi con người để tạo điều kiện cho giao tiếp nói, chúng được gọi là tín hiệu lời nói. Để máy tính xử lý tín hiệu lời nói, chúng cần được chuyển đổi từ một lượng, áp suất không khí, sang một lượng khác, thường là tín hiệu điện.

Chuyển đổi này đạt được với sự trợ giúp của đầu dò. Trong bối cảnh xử lý lời nói, hình dạng của tín hiệu định kỳ, lặp lại sau một thời gian cố định, được gọi là dạng sóng của nó. Nghịch đảo của giai đoạn được gọi là tần số của tín hiệu. Mặt khác, đề cập đến cách con người cảm nhận về chất lượng âm thanh.

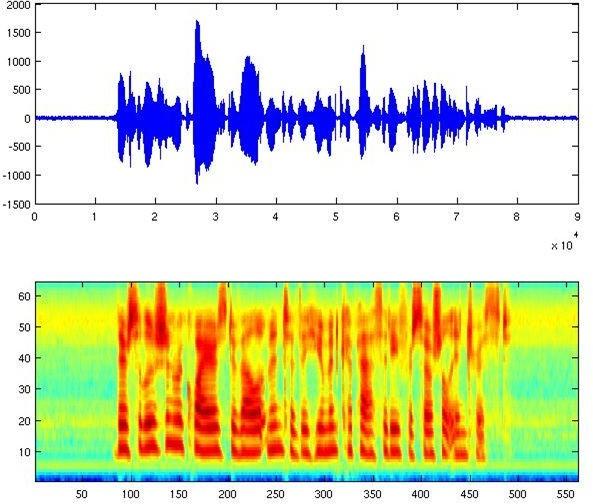
Tín hiệu lời nói thường được số hóa để tạo điều kiện xử lý. Điều này liên quan đến việc chuyển đổi chúng thành một loạt các số bằng cách đo biên độ của tín hiệu trong một khoảng thời gian cố định. Số lượng mẫu được thu thập mỗi giây xác định tỷ lệ lấy mẫu.



Hình 4.1: Tín hiệu âm thanh giọng nói.

#### Đặc trưng âm thanh giọng nói

Các đặc trưng âm thanh giọng nói là các biểu diễn bằng số của các tín hiệu âm thanh giọng nói được sử dụng để phân tích, nhận dạng và tổng hợp. Nhìn chung, tín hiệu âm thanh giọng nói có thể phân thành 2 loại: các tính năng miền thời gian (time-domain) và các tính năng miền tần số (frequency-domain).



Hình 4.2: Time-domain và frequency-domain của âm thanh giọng nói

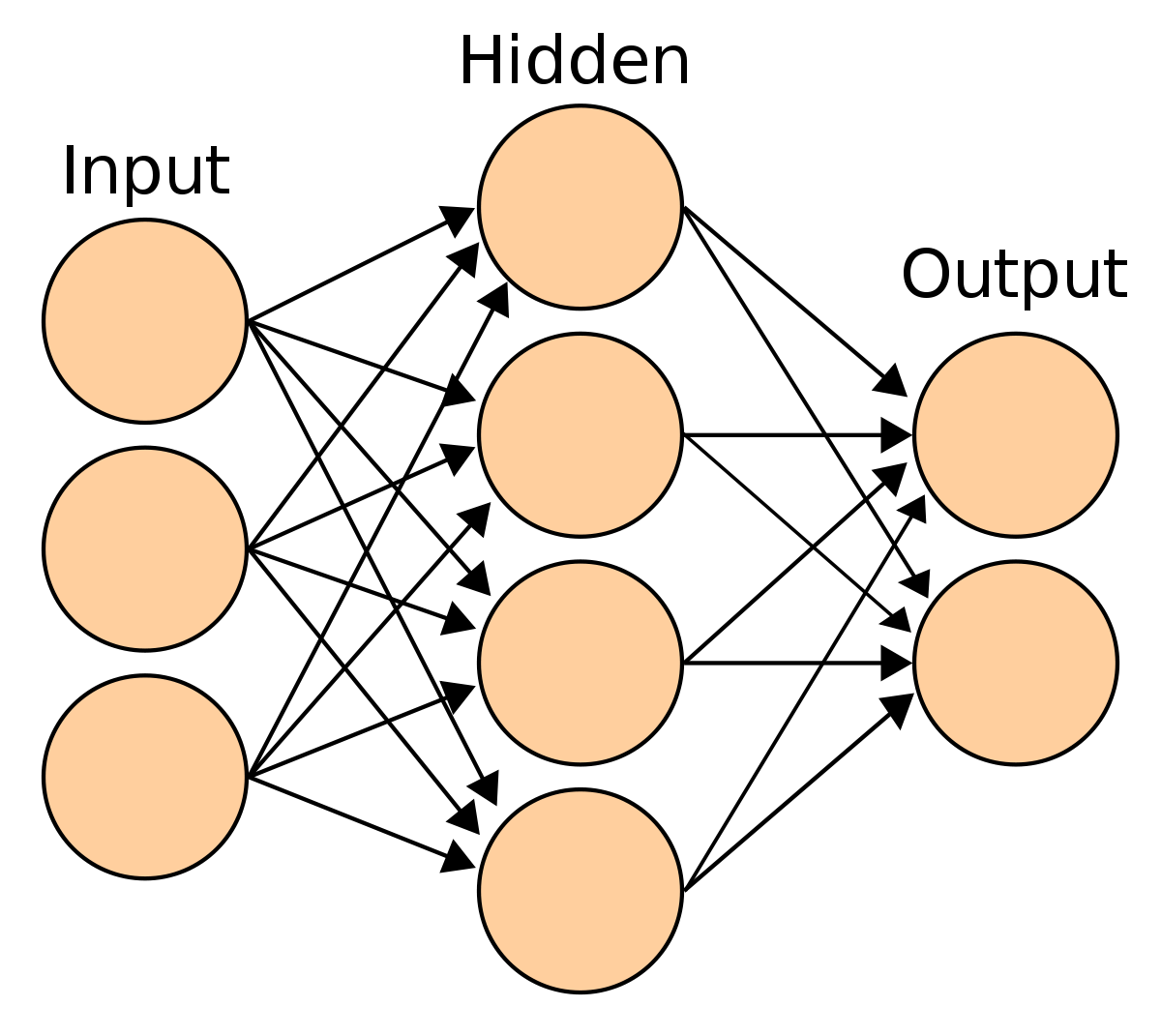
Các tính năng miền thời gian (time-domain) được lấy trực tiếp từ biên độ của tín hiệu âm thanh giọng nói theo thời gian. Đây là phương pháp đơn giản trong ứng dụng xử lý âm thanh giọng nói và thường được áp dụng trong xử lý âm thanh giọng nói theo thời gian thực.

Các tính năng miền tần số (frequency-domain) được lấy từ tín hiệu được biểu thị trong miền tần số còn được gọi là phổ của nó. Một phổ nắm bắt sự phân bố năng lượng như là một hàm của tần số. Phổ là các biểu diễn trực quan hai chiều nắm bắt các biến thể trong phổ tín hiệu theo thời gian. Khi so sánh với các tính năng miền thời gian, thường sẽ phức tạp hơn khi tính toán các tính năng miền tần số vì chúng có xu hướng liên quan đến các hoạt động biến đổi tần số thời gian như Fourier Transform. Phép biến đổi Fourier (FT) là một phép biến đổi toán học quan trọng trong lĩnh vực toán học, vật lý, kỹ thuật và các lĩnh vực khác. Phép biến đổi Fourier cho phép chuyển đổi một tín hiệu từ miền thời gian sang miền tần số.

### Kiến trúc mô hình học sâu và các nhiệm vụ xử lý âm thanh giọng nói.

#### Mạng nơ-ron nhân tạo (Neural Networks)

Mạng Neural network (Mạng nơ-ron) là một mô hình tính toán được lấy cảm hứng từ cách hoạt động của hệ thần kinh sinh học. Nó là một loại mô hình máy học có khả năng tự động học và áp dụng kiến thức từ dữ liệu.



Hình 4.3: Hình ảnh biểu diễn mạng nơ-ron nhân tạo

Mạng Neural network gồm một tập hợp các đơn vị tính toán gọi là nơ-ron, được tổ chức thành các lớp. Mỗi nơ-ron trong mạng nhận đầu vào từ các nơ-ron trong lớp trước đó, thực hiện một phép tính và truyền đầu ra đến các nơ-ron trong lớp tiếp theo. Quá trình này được thực hiện thông qua các liên kết trọng số, cho phép mạng học cách điều chỉnh trọng số để tạo ra các đầu ra phù hợp với đầu vào.

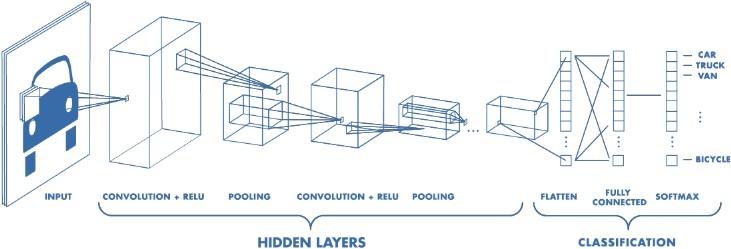
Mạng Neural network có thể có nhiều lớp, gọi là lớp ẩn, giữa lớp đầu vào và lớp đầu ra. Những lớp ẩn này giúp mạng học các mức độ biểu diễn trừu tượng và phức tạp hơn từ dữ liệu đầu vào. Các lớp này cùng với các hàm kích hoạt (activation functions) giúp mạng học các mô hình phi tuyến.

Quá trình huấn luyện mạng Neural network thường bao gồm việc cung cấp dữ liệu huấn luyện đã được gán nhãn và điều chỉnh các trọng số của mạng để giảm thiểu sai số giữa đầu ra dự đoán và đầu ra thực tế. Quá trình này được thực hiện thông qua thuật toán backpropagation, trong đó đạo hàm của hàm mất mát được tính toán và sử dụng để cập nhật trọng số.

Mạng Neural network đã có những ứng dụng rất rộng trong lĩnh vực nhận dạng hình ảnh, xử lý ngôn ngữ tự nhiên, dự báo, điều khiển tự động, và nhiều lĩnh vực khác.

#### Mạng tích chập nơ-ron nhân tạo (Convolutional Neural Network - CNN)

Mạng CNN (Convolutional Neural Network) là một loại mạng nơ-ron nhân tạo được sử dụng rộng rãi trong lĩnh vực xử lý ảnh và thị giác máy tính. Nó được thiết kế để tự động học và phân tích các đặc trưng từ dữ liệu hình ảnh một cách hiệu quả. Cấu trúc của mạng CNN được lấy cảm hứng từ cách mà thị giác của con người hoạt động. Mạng này bao gồm các lớp tích chập (convolutional layers) và các lớp tổng hợp (pooling layers), kết hợp với các lớp kết nối đầy đủ (fully connected layers) ở cuối để thực hiện phân loại hoặc dự đoán.



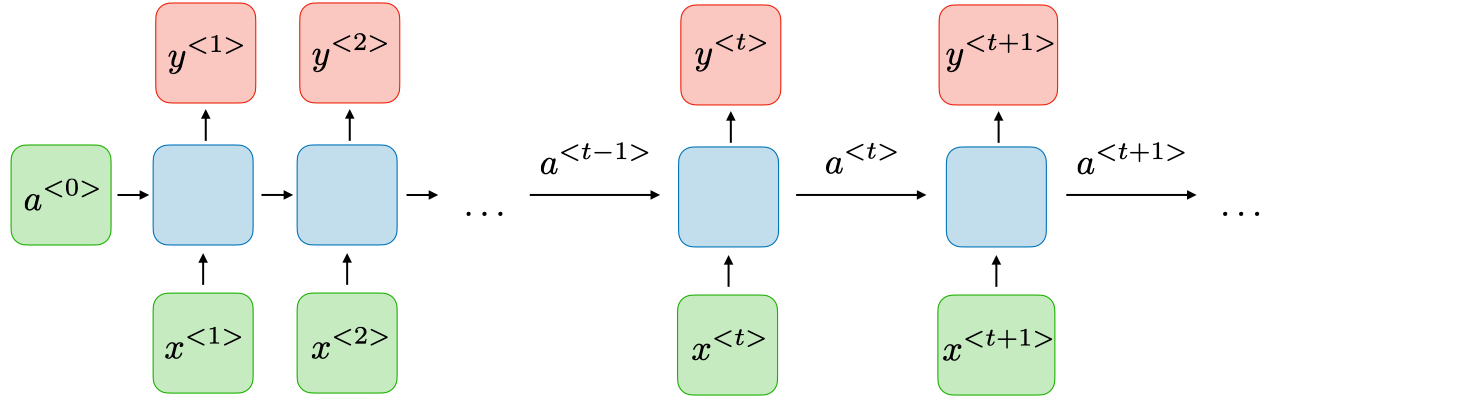
Hình 4.4: Minh họa cấu trúc của mạng CNN

Các lớp tích chập trong mạng CNN thực hiện các phép tích chập giữa các bộ lọc (filters) và các vùng của ảnh đầu vào. Bằng cách áp dụng các bộ lọc này, mạng CNN có khả năng tự động trích xuất các đặc trưng như cạnh, góc, texture, hoặc các đặc trưng phức tạp hơn từ ảnh. Các lớp tổng hợp được sử dụng để giảm kích thước không gian của đặc trưng, đồng thời giữ lại thông tin quan trọng. Điều này giúp giảm số lượng tham số và tính toán của mạng, đồng thời giúp ngăn chặn overfitting.

Sau khi qua các lớp tích chập và lớp tổng hợp, các đặc trưng được đưa vào các lớp kết nối đầy đủ để thực hiện phân loại hoặc dự đoán. Cuối cùng, mạng CNN tạo ra đầu ra là một phân phối xác suất trên các lớp đối tượng mà nó được huấn luyện để nhận dạng. Mạng CNN đã đạt được những thành tựu đáng kể trong các ứng dụng như nhận dạng đối tượng, phân loại ảnh, nhận dạng khuôn mặt, xe tự lái và nhiều lĩnh vực khác.

#### Mạng nơ-ron hồi quy (Recurrent Neural Networks - RNNs)

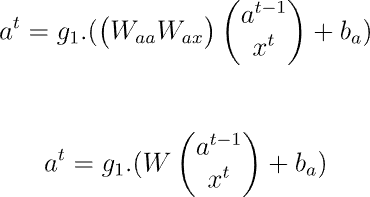
Ý tưởng cốt lõi của RNN: Con người chúng ta không thể bắt đầu suy nghĩ của mình tại tất cả các thời điểm, cũng giống như việc bạn đang đọc bài viết này, bạn hiểu mỗi chữ ở đây dựa vào các chữ mà bạn đã đọc và hiểu trước đó, chứ không phải đọc xong cái là vứt chữ đó đi rồi đến lúc gặp thì lại phải đọc và tiếp thu lại. Nói nôm na ra là kiểu học xong quên luôn xong đến

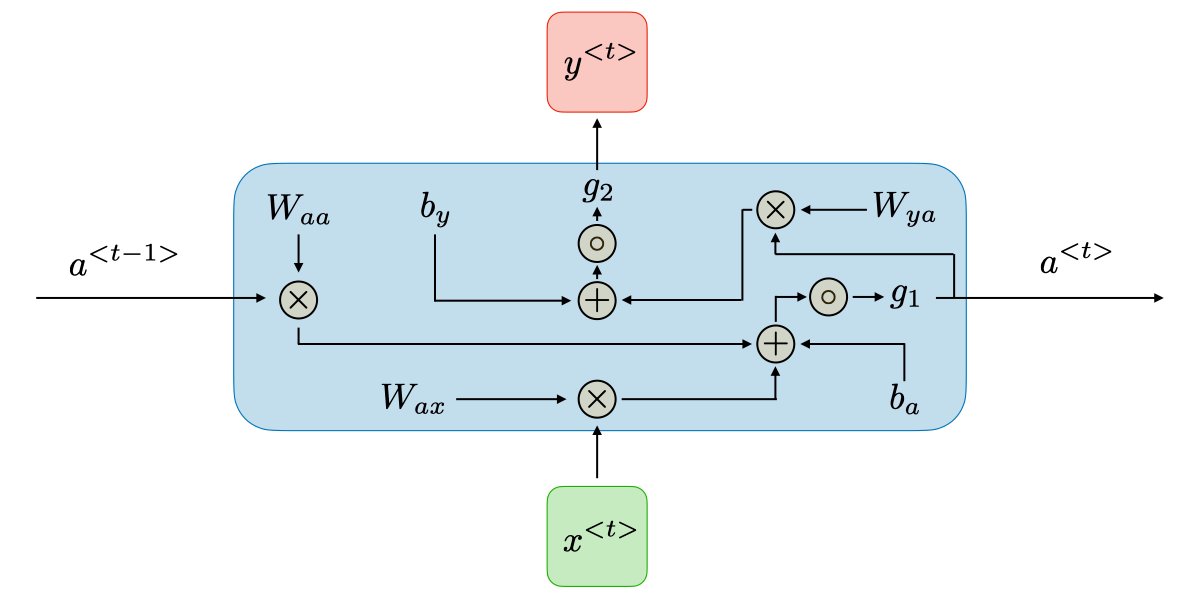
lúc gặp lại là lại học lại. Các mô hình mạng nơ-ron truyền thống lại không thể làm được việc trên. Vì vậy mạng nơ-ron hồi quy (RNN) được sinh ra để giải quyết việc đó. Mạng này chứa các vòng lặp bên trong cho phép nó lưu lại các thông tin đã nhận được. RNN là một thuật toán quan trọng trong xử lý thông tin dạng chuỗi hay nói cách khác là dạng xử lý tuần tự. Xử lý tuần tự là mỗi block sẽ lấy thông tin của block trước và input hiện tại làm đầu vào.

Hình 4.5: Cấu trúc cơ bản của RNN

Tại mỗi bước t, giá trị kích hoạt 𝑎𝑡 và đầu ra 𝑦𝑡 được biểu diễn như sau:







Hình 4.6: Biểu diễn các công thức tính toán của một khối trong RNN

Cùng với đó ta có công thức tính đầu ra tương ứng 𝑡:

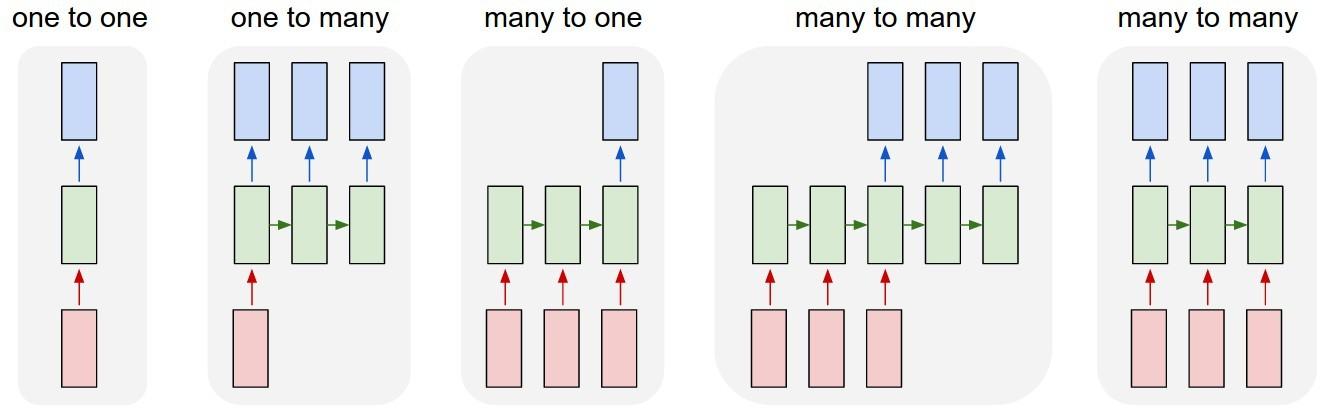


Trong đó 𝑊 , 𝑊 , 𝑊 , 𝑏 , 𝑏 là các trọng số chia sẻ tạm thời trong

𝑎𝑎 𝑎𝑥 𝑦𝑎 𝑎 𝑦

từng lớp ẩn (hidden state) tương ứng. Mạng RNN có nhiều cấu trúc khác nhau:

* One to one: một đầu vào, một đầu ra.
* One to many: một đầu vào, nhiều đầu ra.
* Many to one: nhiều đầu vào, một đầu ra.
* Many to many: nhiều đầu vào, nhiều đầu ra.



Hình 4.7: Các cấu trúc của một số dạng RNN

Ưu điểm và nhược điểm của mạng nơ-ron hồi quy:

| Ưu điểm | Nhược điểm |
| --- | --- |
| Khả năng xử lý các chuỗi đầu vào có độ dài khác nhau | Tính toán khá chậm |
| Kích cỡ mô hình không bị tăng lên theo kích thước đầu vào | Khó truy cập lại thông tin đã đi qua ở một khoảng thời gian dài trước đó - hay còn gọi là bị quên thông tin khi gặp nhiều thông tin mới |
| Quá trình tính toán có sử dụng thông tin trước đo | Phải thực hiện tuần tự nên không tận dụng triệt để được khả năng tính toán song song của GPU |
| Trọng số được chia sẻ trong suốt quá trình học | Vanishing gradient |

Vấn đề của RNN:

Một điểm nổi bật của RNN là nó có thể lấy các thông tin trước đó để dự đoán cho hiện tại. Ví dụ như “Trời hôm nay có mưa nặng” thì RNN có thể dự đoán được chữ tiếp theo sau là “hạt” để hoàn thiện câu hoàn chỉnh “Trời hôm nay có mưa nặng hạt”. Nhưng RNN chỉ hoạt động tốt với chuỗi ngắn, còn đối với chuỗi dài thì khả năng dự đoán của nó bị kém đi. Nguyên nhân là khi chuỗi quá dài thì càng về sau nó sẽ quên các thông tin của các từ trước đó, thậm chí khi càng về cuối thì nó có thể quên luôn thông tin của các từ ở đầu

câu.

#### Bộ nhớ dài- ngắn hạn (Long-Short Term Memory - LSTM)

LSTM là một dạng đặc biệt của RNN, nó có khả năng học các thông tin ở xa. Về cơ bản thì LSTM và RNN không khác nhau là mấy nhưng LSTM có cải tiến một số phép tính trong 1 hidden state và nó đã hiểu quả. Cấu trúc của LSTM không khác gì RNN, những sự cải tiến ở đây nằm ở phần tính toán trong từng hidden state như sau: Thay vì chỉ có một tầng mạng nơ-ron, LSTM thiết kế với 4 tầng mạng nơ-ron tương tác với nhau một các rất đặc biệt.

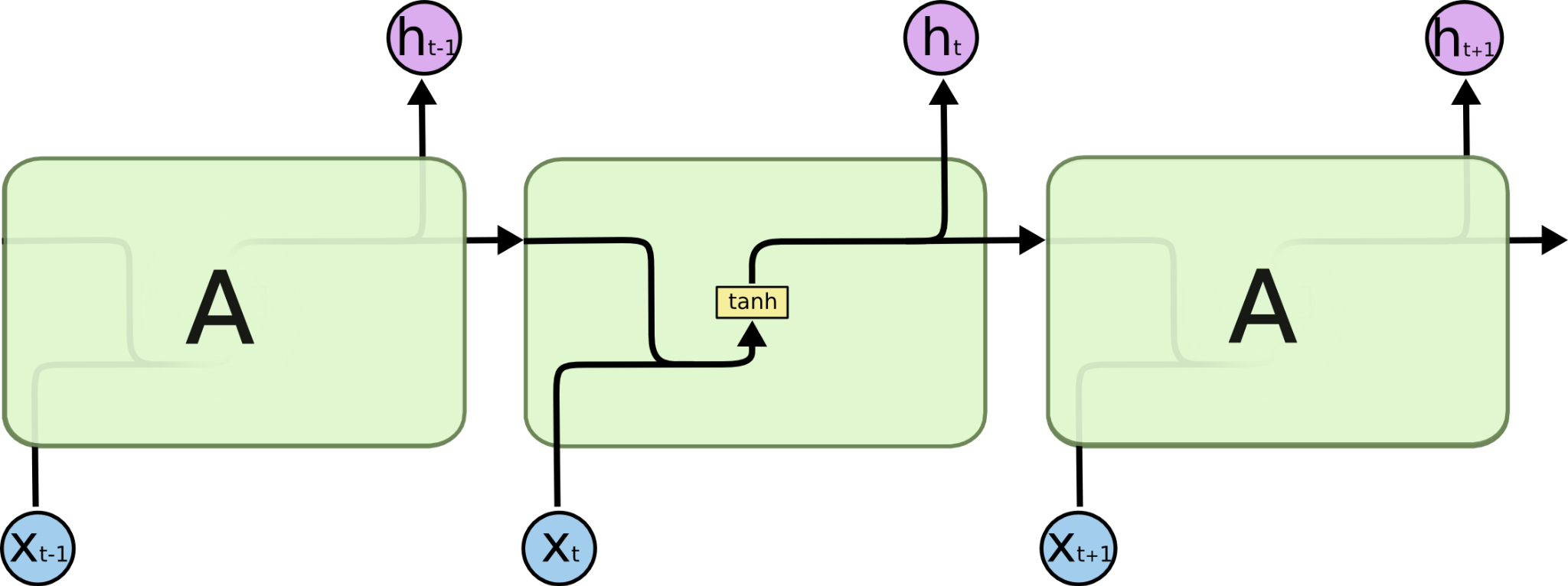
Với kiến trúc đặc biệt, LSTM có khả năng giữ và quản lý thông tin trong một khoảng thời gian dài, cho phép nó xử lý các chuỗi dữ liệu dài và xử lý các phụ thuộc xa. Điều này khác biệt so với mạng RNN truyền thống, có khả năng mất mát thông tin trong quá trình lan truyền ngược.

Mạng LSTM sử dụng các đơn vị LSTM, trong đó mỗi đơn vị có các thành phần chính sau đây:

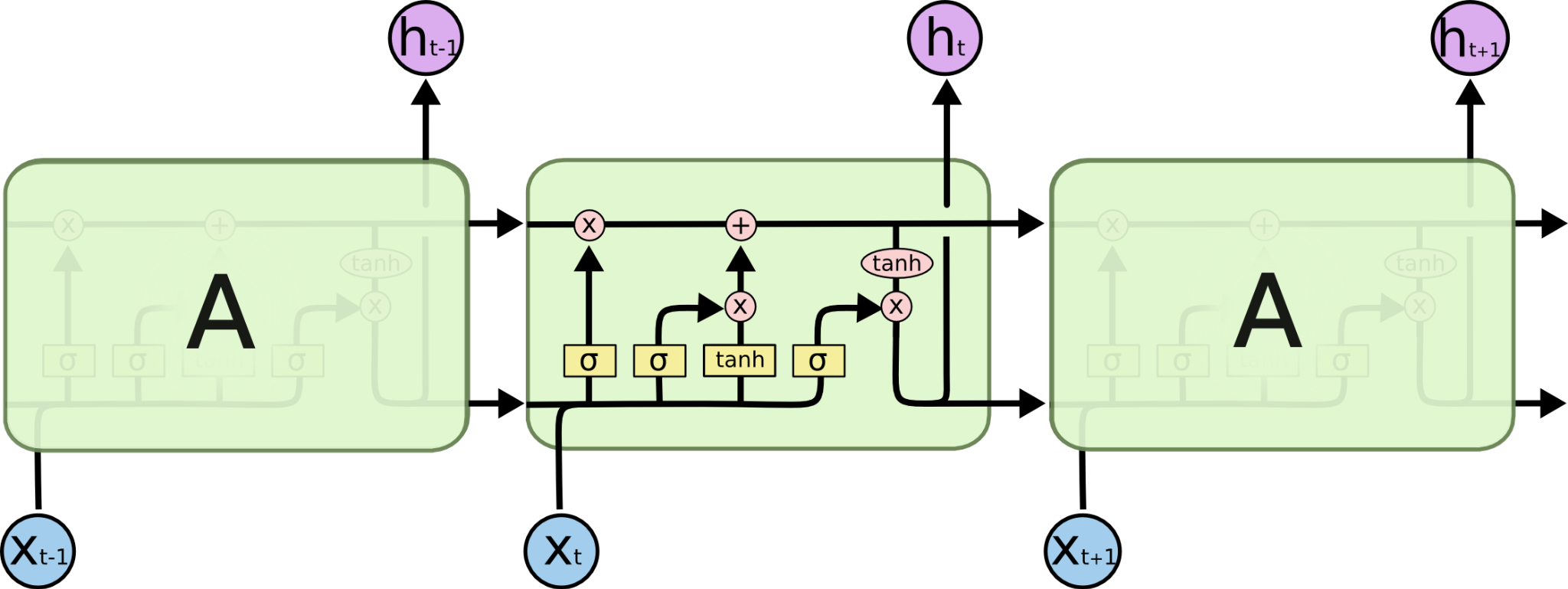
* Cổng quên (Forget gate): Xác định xem thông tin nào nên được giữ lại và thông tin nào nên được bỏ qua từ quá khứ.
* Cổng đầu vào (Input gate): Quyết định thông tin nào nên được cập nhật từ đầu vào mới.
* Cổng ra (Output gate): Quyết định thông tin nào nên được truyền tiếp làm đầu ra của đơn vị LSTM.
* Ô tình trạng (Cell state): Lưu trữ và truyền thông tin dọc theo chuỗi thời gian, giúp duy trì thông tin quan trọng trong quá trình xử lý chuỗi.

Các cổng trong LSTM được điều chỉnh bởi hàm kích hoạt (ví dụ: hàm sigmoid) để quyết định thông tin nào nên được đi qua và thông tin nào nên bị lọc hoặc bỏ qua. Mạng LSTM đã chứng minh hiệu quả của nó trong nhiều ứng dụng như xử lý ngôn ngữ tự nhiên, dịch máy, nhận dạng giọng nói, dự báo chuỗi thời gian và nhiều lĩnh vực khác đòi hỏi xử lý dữ liệu có tính thời gian hoặc phụ thuộc dài hạn.

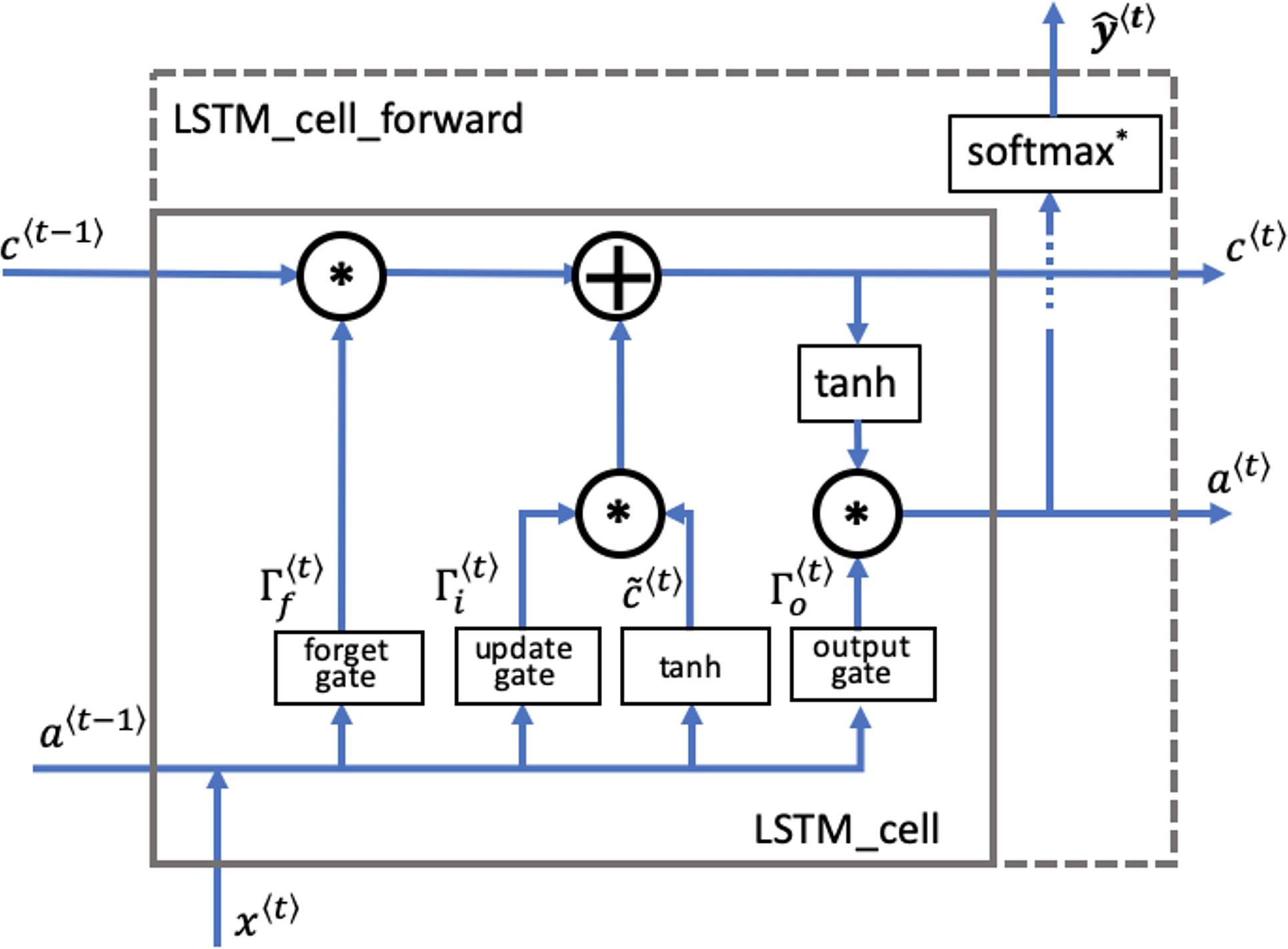
Dưới đây là 2 hình ảnh biểu diễn sự khác nhau giữa RNN và LSTM:



Hình 4.8: Cấu trúc của RNN với hàm kích hoạt tanh



Hình 4.9: Cấu trúc của LSTM với hàm kích hoạt sigmoid và tanh



Hình 4.10: Minh họa cấu trúc của một cell state

Chìa khóa để giúp LSTM có thể truyền tải thông tin giữa các hidden state một các xuyên suốt chính là cell state (hình 4.8)

Trong đó:

* Forget gate Γ **:**

𝑓

+ Đầu ra là hàm *sigmoid* chứa các giá trị từ 0 đến 1.

+ Nếu forget gate có giá trị bằng 0, LSTM sẽ "quên" trạng thái được lưu trữ trong đơn vị tương ứng của trạng thái cell trước đó.

+ Nếu cổng quên có giá trị bằng 1, LSTM sẽ chủ yếu ghi nhớ giá trị tương ứng ở trạng thái được lưu trữ.



* Candidate value  : Chứa thông tin có thể lưu trữ từ step hiện tại



* Update gate

Γ **:** Quyết định xem phần thông tin nào của  có thể

𝑖

thêm vào .



* Cell state : Là bộ nhớ trong của LSTM. Cell state như 1 băng tải truyền các thông tin cần thiết xuyết suất cả quá trình, qua các nút mạng và chỉ tương tác tuyển tính 1 chút. Vì vậy thông tin có thể truyền đi thông suốt mà không bị thay đổi.



* Output gate Γ𝑜**:** Cổng điều chỉnh lượng thông tin đầu ra của cell hiện tại và lượng thông tin truyền tới trạng thái tiếp theo.



* Hidden state  : Được sử dụng để xác định ba cổng  của time step tiếp theo.



* Prediction  : Dự đoán trong trường hợp sử dụng này là phân loại, vì vậy bạn sẽ sử dụng softmax.



Nhìn chung, LSTM khá giống với RNN hay nói cách khác thì RNN là một dạng đặc biệt của LSTM. LSTM giải quyết phần nào Vanishing gradient so với RNN, nhưng chỉ một phần.

Cuối cùng, một biến thể khá nổi tiếng của LSTM, đó chính là Gated Recurrent Units hay GRU. Kiến trúc của GRU kết hợp cổng quên và cổng đầu vào một cổng cập nhật duy nhất. Ngoài ra, GRU cũng kết hợp cell state và hidden state để đưa ra các thay đổi khác.

Hình 4.11: Minh họa cấu trúc của GRU

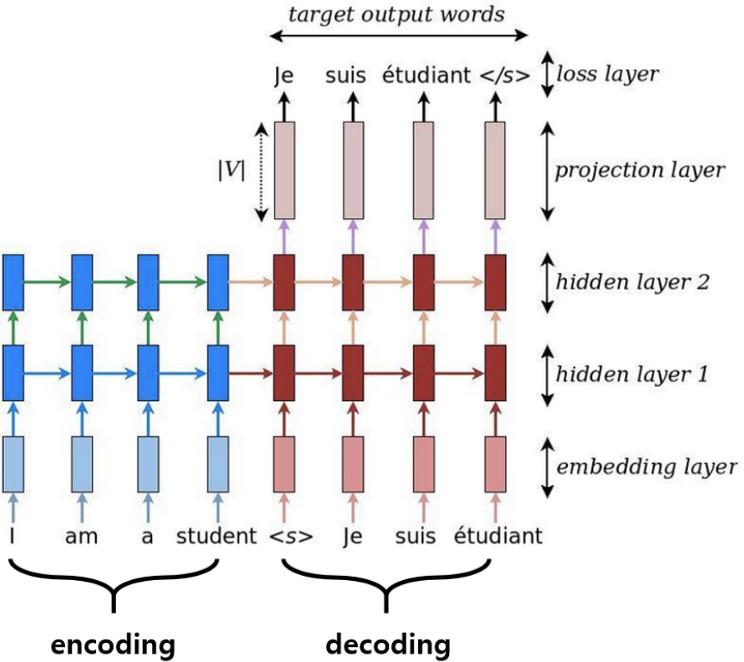
Ngược lại ,với lượng tính toán như trên thì RNN đã chậm rồi, LSTM còn chậm hơn nữa.

* RNN và LSTM vẫn bị xảy ra trường hợp vanishing gradient và thậm chị còn mất khá nhiều thời gian. Bên cạnh đó việc nó không tương thích với dữ liệu có cấu trúc khiến cho việc mô hình bị mất đi cơ chế học được sự liên quan giữa các từ trong câu với nhau
* Vì vậy, Attention đã ra đời, để tăng cường thông tin giữa các hidden sate - thể hiện sự tương quan giữa các hidden state với nhau, với việc loại bỏ hoàn toàn tính tuần tự , cái thiện với việc song song hóa, tăng tốc độ xử lý. Mục đích cuối cùng là để tinh toán mối tương quan giữa input và output.

#### Mô hình Sequence to Sequence (Seq2seq)

#### Cấu trúc, cơ chế hoạt động và hạn chế của mô hình Seq2seq

Mô hình Seq2Seq (Sequence-to-Sequence) là một kiến trúc mạng nơ-ron đặc biệt được sử dụng để xử lý các tác vụ dự đoán, chuyển đổi hoặc tạo ra các chuỗi dữ liệu. Nó được sử dụng rộng rãi trong các ứng dụng như dịch máy, tổng hợp văn bản, trả lời tự động và nhiều tác vụ khác.



Hình 4.12: Hình ảnh cấu trúc của mô hình seq2seq

Mô hình Seq2Seq bao gồm hai thành phần chính: encoder (bộ mã hóa) và decoder (bộ giải mã). Cả hai thành phần này thường được thể hiện bằng các mạng nơ-ron LSTM hoặc GRU (Gated Recurrent Unit).

* Encoder: Bộ mã hóa nhận đầu vào là một chuỗi dữ liệu và trích xuất các đặc trưng quan trọng của chuỗi đó. Đầu vào có thể là văn bản, âm thanh hoặc bất kỳ dạng chuỗi nào khác.
* Mỗi phần tử trong chuỗi được truyền qua một mạng LSTM/GRU trong encoder để tạo ra một vector đại diện cho toàn bộ chuỗi đầu vào. Vector này thường được gọi là vector ngữ cảnh (context vector) hoặc vector biểu diễn (representation vector).
* Decoder: Bộ giải mã nhận vector ngữ cảnh từ encoder và tạo ra chuỗi đầu ra. Nó cũng sử dụng một mạng LSTM/GRU để dự đoán các phần tử trong chuỗi đầu ra. Trong quá trình huấn luyện, đầu ra thực tế được sử dụng để so sánh với đầu ra dự đoán và điều chỉnh các trọng số của mạng để giảm thiểu sai số.

Ví dụ về mô hình Seq2Seq: dịch máy

* Bước 1: Đầu vào (Input): "Hello, how are you?"
* Bước 2: Bộ mã hóa (Encoder): Mạng LSTM nhận chuỗi đầu vào và tạo ra một vector ngữ cảnh.
* Bước 3: Vector ngữ cảnh được truyền vào bộ giải mã (Decoder).
* Bước 4: Bộ giải mã (Decoder): Mạng LSTM nhận vector ngữ cảnh và bắt đầu tạo ra chuỗi đầu ra.
* Bước 5: Chuỗi đầu ra được dự đoán từ bộ giải mã: "Xin chào, bạn khỏe không?"

Quá trình này được lặp lại cho từng phần tử trong chuỗi đầu vào và dự đoán các phần tử tương ứng trong chuỗi đầu ra. Mô hình Seq2Seq cho phép mô hình học cách ánh xạ từ một ngôn ngữ nhất định.

Mục tiêu của chúng ta trong bài toán này là xây dựng một trình tóm tắt văn bản, với đầu vào là một chuỗi dài các từ (trong nội dung văn bản) và đầu ra là một bản tóm tắt ngắn (cũng là một chuỗi). Vì vậy chúng ta có thể thực hiện bài toán này bằng mô hình Many-to-many Seq2seq được thể hiện ở hình 4.5.

Vấn đề của mô hình seq2seq:

* Vấn đề mất thông tin ngữ cảnh dài hạn: Mô hình Seq2Seq sử dụng mạng nơ-ron RNN để mã hóa và giải mã chuỗi. RNN có khả năng ghi nhớ thông tin trước đó trong trạng thái ẩn của nó, nhưng nó dễ bị mất thông tin trong quá trình lan truyền ngược. Điều này có nghĩa là các phụ thuộc dài hạn trong chuỗi đầu vào có thể không được xử lý tốt.
* Vấn đề độ dài chuỗi biến đổi: Mô hình Seq2Seq không xử lý tốt các chuỗi đầu vào và đầu ra có độ dài khác nhau. Nếu các chuỗi đầu vào và đầu ra có độ dài khác nhau, việc đồng bộ giữa encoder và decoder sẽ trở nên khó khăn và có thể dẫn đến kết quả không chính xác.
* Khả năng mô hình hóa thông tin phi tuyến: Mạng LSTM/GRU trong mô hình Seq2Seq có khả năng mô hình hóa thông tin phi tuyến, nhưng nó cũng có giới hạn. Đối với các tác vụ phức tạp yêu cầu khả năng mô hình hóa phi tuyến cao, mô hình Seq2Seq có thể gặp khó khăn.
* Độ phức tạp tính toán: Mô hình Seq2Seq có độ phức tạp tính toán cao, đặc biệt là khi xử lý các chuỗi dữ liệu dài. Điều này có thể gây khó khăn trong việc huấn luyện và triển khai mô hình trên các tài nguyên tính toán hạn chế.
* Thiếu khả năng xử lý từ vựng lớn: Mô hình Seq2Seq có thể gặp khó khăn trong việc xử lý các từ vựng lớn. Việc mã hóa và giải mã các từ trong các vector số có thể trở nên tốn kém về mặt tính toán và không hiệu quả đối với từ vựng lớn.
* Thiếu khả năng xử lý thông tin ngoại lệ: Mô hình Seq2Seq có xu hướng

không xử lý tốt các thông tin ngoại lệ hoặc không thường xuyên trong

dữ liệu đầu vào, vì nó dựa vào các phép tính trung bình của chuỗi đầu vào.

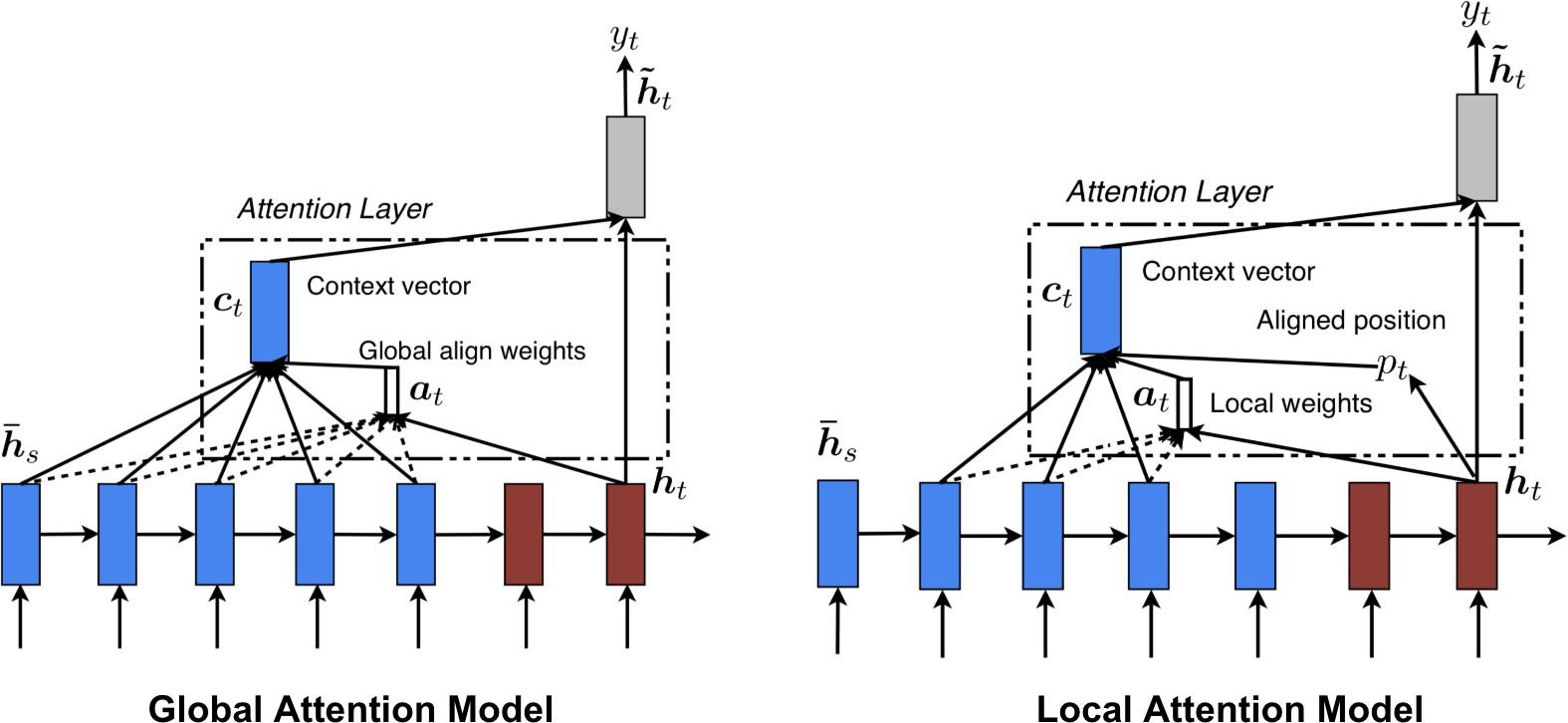
#### Cơ chế Attention trong mô hình seq2seq

Trong phần 4.2.5 có nêu lên rõ ràng hạn chế của mô hình seq2seq, Cơ chế Attention sinh ra để khắc phục vấn đề này. Nó nhằm mục đích dự đoán một từ bằng cách chỉ xem xét một vài phần cụ thể của chuỗi, thay vì toàn bộ chuỗi. Xét ví dụ với chuỗi đầu vào là “"Which sport do you like most?”

và output là “I love cricket”.

Từ "I" trong câu output có mối liên hệ với từ "you" trong input, tương tự từ "love" trong câu output lại có liên kết với từ "like" trong câu input. Vậy, thay vì nhìn vào tất cả những từ trong đầu vào thì chúng ta có thể tăng tầm quan trọng của một vài từ cụ thể của đầu vào có ý nghĩa đối với đầu ra. Đó là ý tưởng cơ bản của cơ chế Attention.

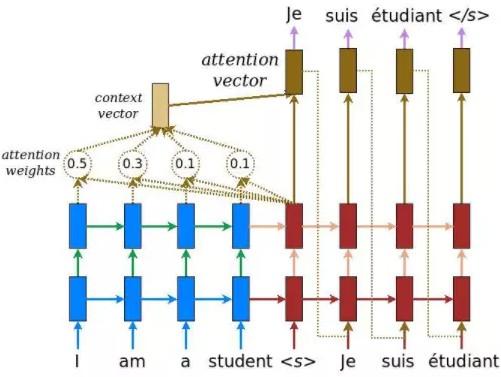
Có hai cơ chế Attention khác nhau phụ thuộc vào vector trung gian:



Hình 4.13: Cơ chế dựa vào weight global và weight local

* Global Attention: Tất cả các trạng thái ẩn của bộ mã hóa được xem xét để tạo ra vector trung gian.
* Local Attention: Chỉ một vài trạng thái ẩn của bộ mã hóa được xem xét và lấy ra để tạo vector trung gian.

Mô hình seq2seq khi áp dụng cơ chế attention vào sẽ có cấu trúc như sau (các khối màu xanh dương là encoder, màu đỏ là decoder):



Hình 4.14: Cấu trúc của mô hình seq2seq có sử dụng attention

Chi tiết các bước, tại mỗi time-step ℎ ở phía decoder:

𝑡

* Bước 1: Nhận vector trạng thái ẩn của decoder ℎ và tất cả các vector

𝑡

trạng thái ẩn của encoder ℎ .

𝑠

* Bước 2: Tính trọng số attention. Với mỗi vector trạng thái ẩn (hidden state) của encoder thì ta cần tính toán sự liên quan với vector trạng thái ẩn của decoder. Cụ thể, ta sẽ áp dụng một phương trình tính score của

"attention" với đầu vào là vector trạng thái ẩn decoder - ℎ

𝑡

một vector

trạng thái ẩn của encoder - ℎ

𝑠

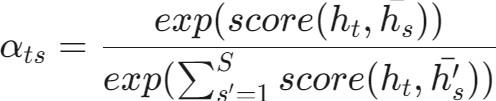


*score(*ℎ , ℎ *)*

𝑡 𝑠

trả về một giá trị vô hướng được gọi là

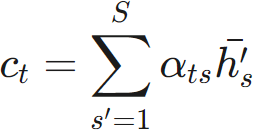
* Bước 3: Tính trọng số Attention. Áp dụng hàm softmax với đầu vào là score attention



* Bước 4: Tính toán vector bối cảnh 𝑐 (context vector) là tổng của các

𝑡

trọng số attention nhân với vector trạng thái ẩn của decoder tại time-step tương ứng



Như vậy cơ chế attention được ra đời để giải quyết các vấn đề của mô hình seq2seq (transformer và cơ chế attention ra đời để thay cho seq2seq không cần đến các mạng nơ-ron hồi tiếp) , với ý tưởng sử dụng một vector bối cảnh có thể tương tác với toàn bộ vector trạng thái ẩn của encoder thay vì chỉ sử dụng vector trạng thái ẩn cuối cùng để tạo ra vector biểu diễn cho decoder.

#### Mô hình Transformer

#### Cơ chế Attention trong mô hình Transformer

Với cơ chế Attention, Transformer sẽ không cần tới các phép toán convolutional và recurrence, với cơ chế xử lý song song rõ ràng cho thấy được cơ chế này cung cấp một hiệu suất vượt trội và thời gian huấn luyện được tối ưu đang kể. Việc ở các mạng trước đó, khi sử dụng với chuỗi đầu vào là một chuỗi dài, sẽ khiến cho việc tìm hiểu mối tương quan giữa các từ trở nên khó khăn hơn rất nhiều.

Trong Transformer vấn đề này đã được giải quyết nhưng sẽ phải giảm effective resolution do lấy trung bình các attention-weighted position, những vấn để này cũng sẽ được giải quyết ở phần sau với Multi-head attention.

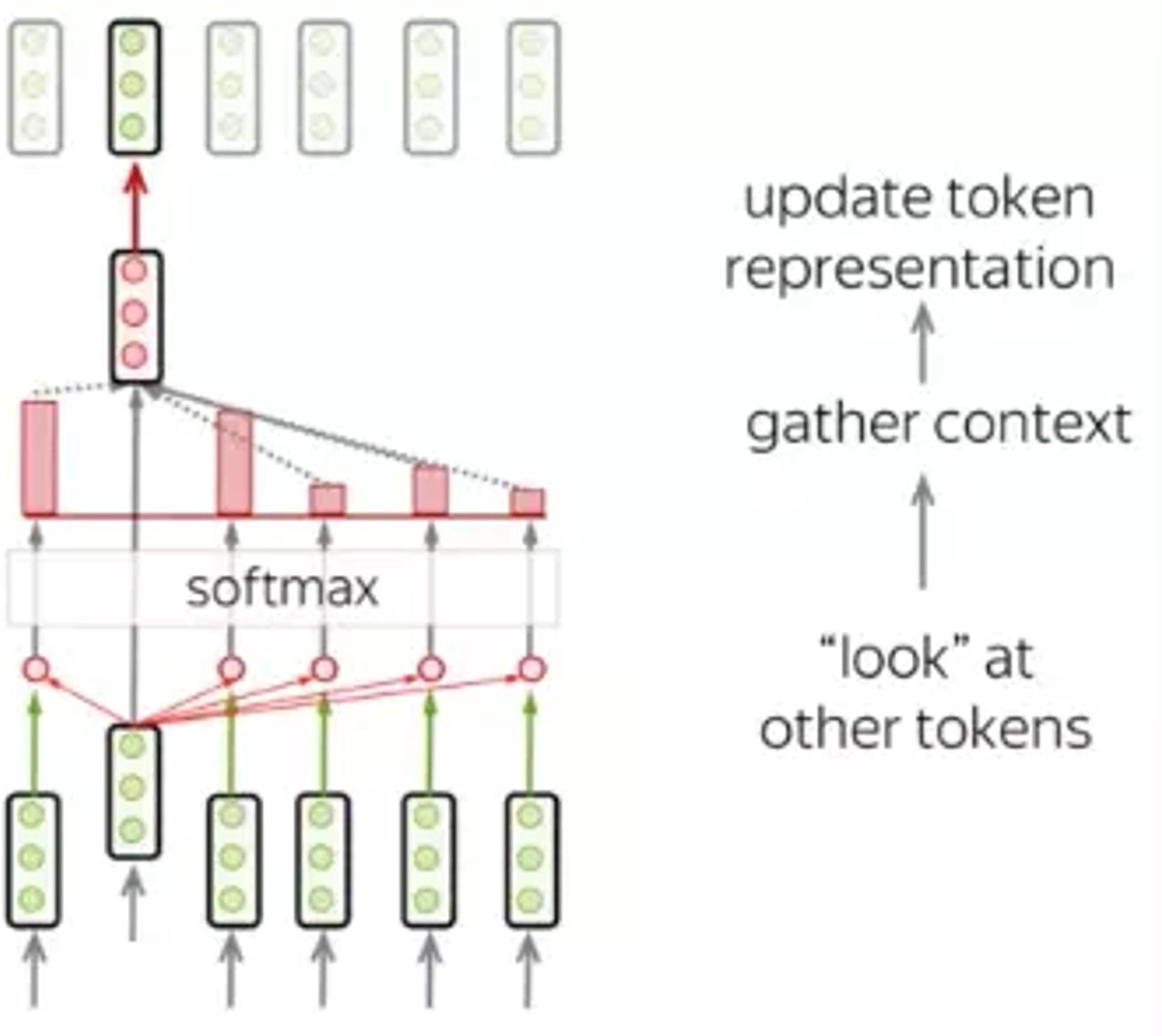
Trong bài toán NLP, đối với trường hợp cụ thể như ta cần mã hóa(encode) một chuỗi đầu vào thành một vector chứa ngữ cảnh rồi sau đó

giải mã (decode) dự đoán các từ trước đó. Nhưng chúng ta sẽ gặp một số vấn đề như sau:

* Vanishing gradient: mất mát đạo hàm ở những hidden state cuối khi đầu vào là một chuỗi dài như đoạn văn
* Exploding gradient: Hiện tượng gradient tăng mất kiểm soát do dồn ở những lớp cuối và đặc biệt xảy ra với những câu dài như đoạn văn.
* Nén bộ nhớ (Memory compression): Do việc embed chuỗi đầu vào thành vector cố định kích thước, qua nhiều lần thử nghiệm cho thấy được mô hình này học với các câu dài rất kém trong khi đó lại rất lãng phí bộ nhớ với những câu ngắn. Vấn đề này vẫn còn xuất hiện trong LSTM và GRU

Không phù hợp với dạng dữ liệu có mối quan hệ liên kết ngữ nghĩa: Ví dụ với câu: “She is doing homework”. Nhận thấy “homework” có mối quan hệ nhiều hơn với “doing”. Đối với RNN - học tuần tự từ trái sang phải nhưng không có cơ chế để mô hình học được sự liên quan giữa các từ. Attention sẽ giúp chúng ta thực hiện được việc đó.

Cơ chế self-attention là thành phần quan trọng nhất của Transformer. Sự khác biệt giữa RNN Attention và self-attention là: Trong attention sẽ tính toán dựa trên trạng thái của decoder ở time-step hiện tại và tất cả các trạng thái ẩn của encoder.



Hình 4.15: Miêu tả hoạt động của self-attention

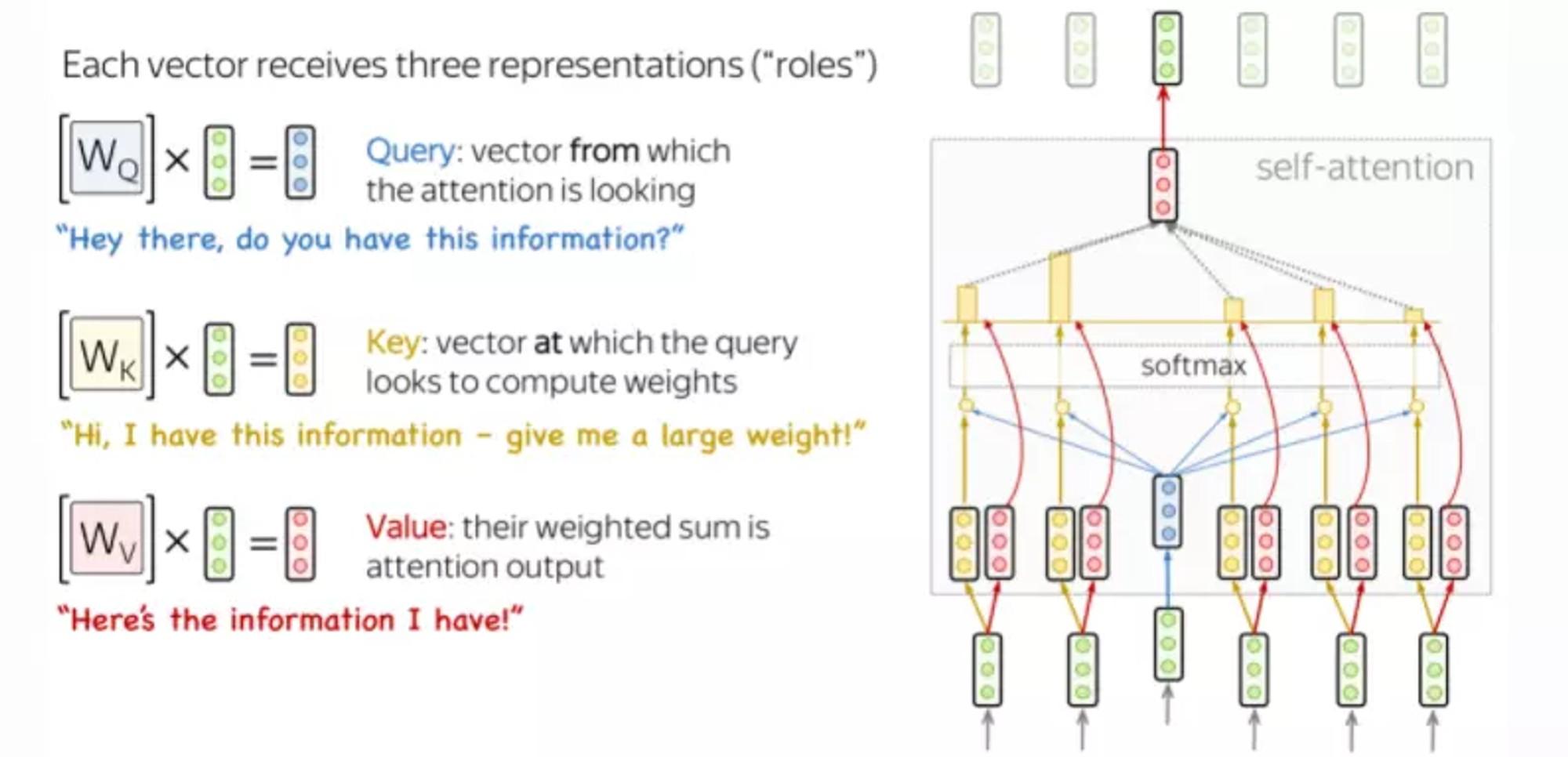
Còn self-attention có thể hiểu là attention trong 1 câu, khi từng thành phần trong câu sẽ tương tác với nhau. Từng token sẽ quan sát các tokens còn lại, thu thập nhữ cảnh của câu và cật nhập vào vector biểu diễn. Để xây dựng cơ chế self-attention ta cần chú ý đến hoạt động của vector biểu diễn cho mỗi từ lần lượt là:

* Query: được sử dụng để đặt câu hỏi từ này ở trong câu có ý nghĩa gì.
* Key: trả lời từ hiện tại có liên quan thế nào đến từ mà query đang hỏi.
* Value: chứa thông tin của từ đang hỏi.



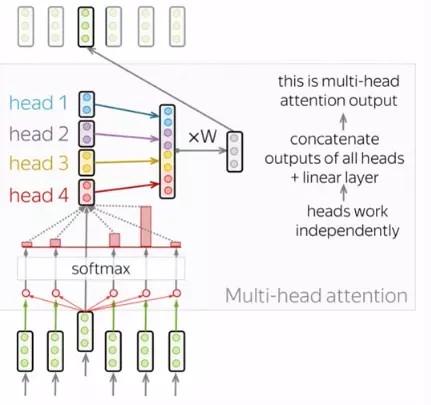
Dựa vào ý nghĩa của *q, k, v* có thể giải thích được công thức của Transformers Attention:

* *q.k* qua hàm softmax để đưa ra xác suất của từ liên quan nhất với từng từ để hỏi tương ứng.
* Sau đó nhân với *v* để đưa ra giá trị dựa vào sự tương quan đó.



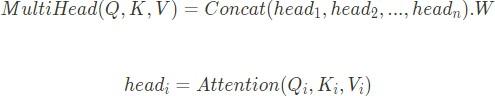
Hình 4.16: Mô tả nhiệm vụ và chức năng của các vector query, key, value

Query được sử dụng khi một token quan sát những tokens còn lại, nó sẽ tìm kiếm thông tin xung quanh để hiểu được ngữ cảnh và mối quan hệ của nó với tokens còn lại. Key sẽ phản hồi yêu cầu của Query và được sử dụng để tính trọng số attention. Cuối cùng, Value được sử dụng trong số attention vừa rồi để tính ra vector đại diện (attention vector).



Hình 4.17: Mô tả một multi-head attention

Mỗi quá trình như vậy được gọi là 1 head của attention. Khi lặp lại quá trình này nhiều lần (trong bài báo là 3 heads) ta sẽ thu được quá trình Multi-head Attention như biến đổi bên dưới:



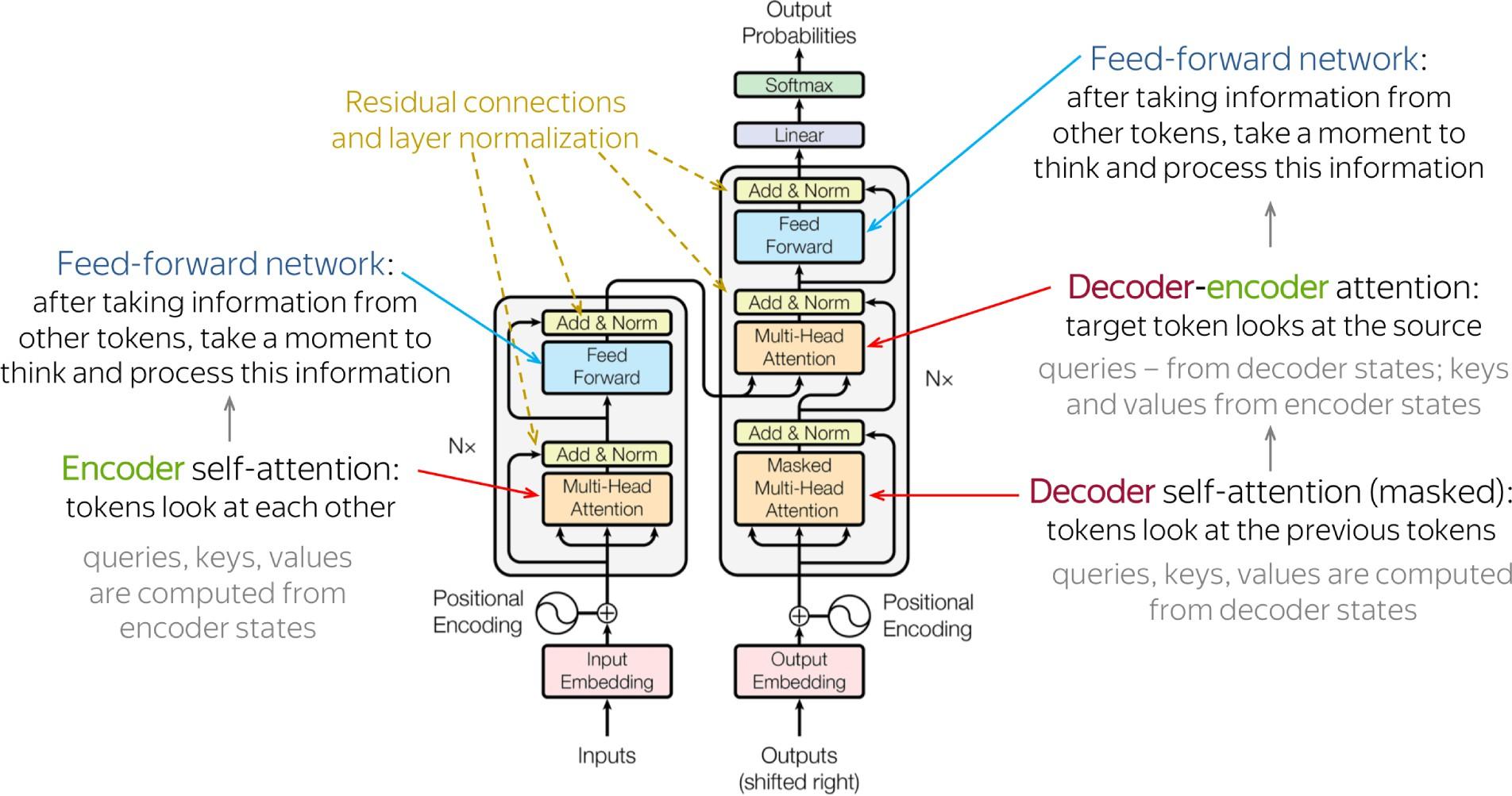
Để trả về output có cùng kích thước với ma trận input chúng ta chỉ cần nhân với ma trận *W* để chiều rộng bằng với chiều rộng của ma trận input.

#### Cấu trúc của mô hình Transformer

Tiếp theo ta sẽ đi tìm hiểu về các thành phần chính cấu tạo thành transformer. Đây là cấu trúc của transformer được giới thiệu trong bài báo “Attention is all you need”. Mô hình thực hiện chính xác những gì đã được giới thiệu ở trên.

Ở bên trái là encoder, thông thường có Nx = 6 layers chồng lên nhau. Mỗi layer sẽ có multi-head attention như đã tìm hiểu và khối feed-forward. Ngoài ra còn các các kết nối residual giống như trong mạng Resnet. Ở bên phải là decoder, tương tự cũng có Nx = 6 layers chồng lên nhau.

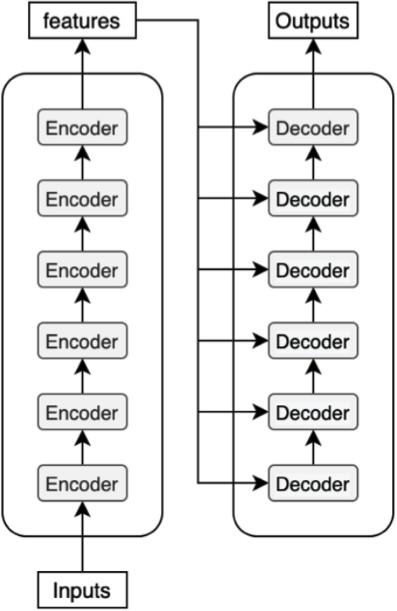
Kiến trúc thì khá giống encoder những chỉ có thêm khối masked multi-head attention ở vị trí đầu tiên. Ta sẽ tìm hiểu sâu hơn các thành phần trong transformer.



Hình 4.18: Hình ảnh mô tả cấu trúc của mô hình Transformer

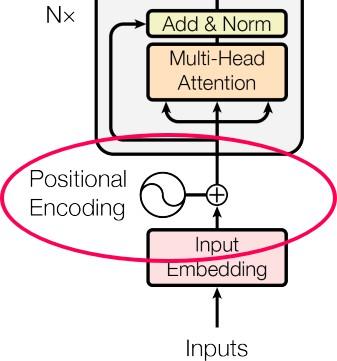
Với kiến trúc của Transformer có 2 phần chính là encoders và decoders:

* Encoders: gồm bộ lớp encoder giống hệt nhau và xếp chồng lên nhau. Mỗi lớp có 2 khối chính đó là: attention block và feed-forward block, feed-forward network được kết nối đầy đủ theo vị trí ( positionwide fully connected feed-forward network ).
* Decoders: Mỗi khối decoder nhận được các features từ encoder. Chúng ta có thể hình dung được sự liên kết giữa decoder và encoder như hình 4.16.



Hình 4.19: Hình ảnh mô tả sự liên kết giữa decoder và encoder

#### Positional Encoding:



Hình 4.20: Vị trí của positional encoder trong mô hình Transformer

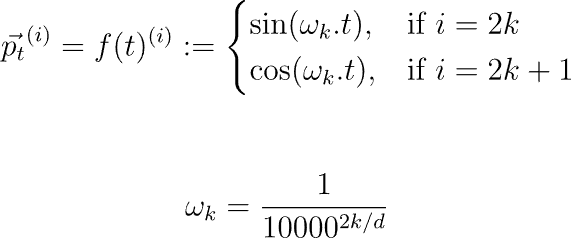
Khác với việc học tuần tự của RNN, Transformer loại bỏ cơ chế học tuần tự để học song song dựa trên multi-head self-attention, nên máy tính không hiểu được cấu trúc của câu dẫn đến việc sai về mặt ngữ nghĩa. Do đó cần một cách để giúp máy tính hiểu được thứ tự các từ trong câu, ý tưởng là ta sẽ đánh trọng số cho các từ trong câu.

Cách tiếp cận ngây thơ để đánh trọng trọng số là thêm giá trị mỗi từ theo tuyến tính. Có nghĩa là với từ đầu tiên thêm 1, từ thứ hai thêm hai, cứ như vậy cho đến hết. Với cách tiếp cần này sẽ có vài vấn đề như: các giá trị sẽ trở lên quá lớn, mô hình mất nhiều thời gian để hội tụ hơn (lý do các bạn có thể tìm kiếm với từ khóa “feature scaling”)

Một cách tiếp cận khác là thêm các giá trị trong khoảng [0, 1]; với giá trị 0 cho từ đầu tiên và 1 cho từ cuối cùng. Một trong những vấn đề đối với cách tiếp cần này là bạn không thể tìm ra có bao nhiêu từ hiện diện trong một phạm vi cụ thể.

Nói cách khác, giá trị cộng thêm không có ý nghĩa nhất quán trong các câu khác nhau. Cách tiếp cận tốt nhất nên thỏa mãn các yêu cầu sau:

* Giá trị được thêm nên là duy nhất với mỗi time-step.
* Khoảng cách giữa các từ trong câu nên thống nhất với mọi câu.
* Giá trị thêm vào cần phải được giới hạn.
* Giá trị thêm vào cần phải làm rõ được vị trí các từ trong câu. Công thức tính toán positional encoding trong Transformer là:



Trong đó:

* *d* là số chiều của vector từ .
* *k* là chỉ số của chiều trong vector từ.
* 𝑓(𝑡) là vị trí từ trong câu.

Mỗi thành phần trong ma trận positional encoding sẽ có giá trị sin hoặc cos của một hàm sóng với tần số khác nhau. Khi kết hợp với ma trận từ

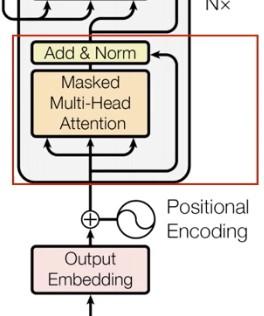
nhúng, positional encoding giúp mô hình Transformer biết được vị trí tương đối của từng từ trong câu.

Ví dụ: Giả sử chúng ta có một câu gồm các từ "I", "love", "cats". Đầu tiên, mỗi từ sẽ được biểu diễn bằng vector từ nhúng. Tiếp theo, positional encoding sẽ được thêm vào các vector từ để đưa thông tin vị trí vào mô hình Transformer.

* Đầu vào ban đầu: [vector\_embed("I"), vector\_embed("love"), vector\_embed("cats")]
* Sau khi thêm positional encoding: [vector\_embed("I")+positional\_encoding("I"), vector\_embed("love")+positional\_encoding("love"), vector\_embed("cats")+positional\_encoding("cats")]

Positional encoding giúp mô hình Transformer phân biệt được vị trí và thứ tự các từ trong câu, làm cơ sở cho việc xử lý thông tin chuỗi trong mô hình.

#### Masked Multi-Head Attention

Một layer tương tự như Multi-Head Attention, điểm khác biệt là ở lúc train, khi đã biết rõ output mong muốn để đưa vào làm input của time-step tiếp theo. Để hiểu rõ điều này trước tiên bạn cần biết cách Decoder vận hành, tôi sẽ lấy ví dụ trong bài toán text generate với câu “Tall house with wide door”

Hình 4.21: Hình ảnh vị trí của lớp masked multi-head attention

Cũng giống như RNN, Decoder sử dụng output của time-step trước làm input của time-step hiện tại. Tức với ví dụ trên ta sẽ có các bước decode sau:

* input = “” qua Masked Multi-Head Attention, nhận thêm thông tin từ Encoder để đưa ra thông tin đầu tiên output = “Tall”.
* input = “Tall” → output = “Tall house”.
* input = “Tall house” → output = “Tall house with”.
* input = “Tall house with” → output = “Tall house with wide”.
* input = “Tall house with wide” → output = “Tall house with wide door”.

Mọi thứ sẽ như trên nếu mô hình đã được train xong, nhưng trong quá trình train output ở time-step trước có thể sai khiến output của time-step hiện tại cũng có thể sai theo. Và đương nhiên việc này có thể dần cải thiện trong quá trình train, nhưng sẽ rất tốn tài nguyên. Vì vậy giải pháp được đưa ra là sử dụng output mà chúng ta muốn đưa vào làm input của từng time-step. Mỗi time-step sẽ nhập input tương ứng nên có ở thời điểm đó, tức sẽ cho biết các từ nên được generate ở thời đó và che đi phần còn lại. Để máy tính hiểu được cách làm trên ta sử dụng Look-ahead Mask là ma trận tam giác dưới với các giá trị 1 biểu thị từ không muốn che, và 0 cho từ muốn che.

#### Add & Norm layers và Feed-forward layers

Normalization layers: Trong cấu trúc của transformer có các lớp “Add và Norm” thì “Norm” ở đây thể hiện cho Normalization layer. Lớp này đơn giản là sẽ chuẩn hóa lại đầu ra của multi-head attention, mang lại hiệu quả cho việc nâng cao khả năng hội tụ.

Khối Feed-forward: Sau khi thực hiện tính toán ở khối attention cho mỗi lớp thì khối tiếp theo là FFN. Có thể hiểu là cơ chế attention giúp thu thập thông tin từ những tokens đầu vào thì FFN là khối xử lý những thông tin đó.

Dưới đây là một phác thảo về cách áp dụng Transformer trong bài toán ASR:

* Chuẩn bị dữ liệu: Dữ liệu đầu vào trong ASR là tín hiệu giọng nói. Trước khi áp dụng Transformer, tín hiệu giọng nói thường được chuyển đổi thành các đặc trưng âm thanh như Mel-frequency cepstral coefficients (MFCCs) hoặc filter bank energies. Các đặc trưng này sau đó được chia thành các khung thời gian nhỏ và tiếp tục được sử dụng trong mô hình Transformer.
* Mã hóa âm thanh: Đầu tiên, mô hình sẽ có một lớp mã hóa âm thanh để xử lý đặc trưng âm thanh từ tín hiệu giọng nói. Mã hóa âm thanh thường bao gồm một hoặc nhiều lớp mạng nơ-ron tích chập (CNN) hoặc lớp mạng nơ-ron dài ngắn (LSTM) để trích xuất thông tin từ đặc trưng âm thanh.
* Mã hóa ngôn ngữ: Sau khi âm thanh được mã hóa, mô hình sử dụng mã hóa ngôn ngữ để biểu diễn các từ trong chuỗi văn bản. Mã hóa ngôn ngữ thường bao gồm một hoặc nhiều lớp mạng nơ-ron dài ngắn dài (LSTM) hoặc lớp mã hóa biến thể Transformer. Mô hình học cách ánh xạ từng từ trong chuỗi văn bản đến một biểu diễn vector.
* Attention và Decoder: Sau khi được mã hóa, cả hai biểu diễn âm thanh và ngôn ngữ được đưa vào một bộ giải mã Transformer. Đây là nơi mà attention mechanism (cơ chế chú ý) được sử dụng để đồng bộ hóa thông tin âm thanh và ngôn ngữ. Attention cho phép mô hình tập trung vào các phần quan trọng của âm thanh và từ vựng để đưa ra các dự đoán chính xác cho từng thời điểm trong chuỗi.
* Mã hóa và dự đoán chuỗi ký tự: Cuối cùng, chuỗi ký tự được dự đoán bằng cách sử dụng một lớp mạng nơ-ron hoàn toàn kết nối (fully connected layer) và một hàm softmax. Lớp mạng nơ-ron hoàn toàn kết nối sẽ chuyển đổi các biểu diễn từ bộ giải mã thành một vector đầu ra

có kích thước bằng với số lượng từ vựng. Hàm softmax sau đó sẽ áp dụng để chuẩn hóa vector đầu ra thành một phân phối xác suất trên các từ vựng.

* Trong quá trình huấn luyện, mô hình sẽ được điều chỉnh bằng cách so sánh đầu ra dự đoán với chuỗi ký tự đúng sử dụng một hàm mất mát CTC Loss (sẽ được đề cập ở mục 4.3). Mục tiêu là tối thiểu hóa mất mát để đạt được độ chính xác cao trong việc dự đoán chuỗi ký tự.

Tổng quan, việc áp dụng Transformer trong bài toán ASR giúp mô hình có khả năng mô hình hóa quan hệ phức tạp giữa tín hiệu giọng nói và văn bản. Các cơ chế attention trong Transformer giúp mô hình tập trung vào các phần quan trọng của dữ liệu đầu vào và giúp cải thiện độ chính xác của hệ thống nhận dạng giọng nói.

### Hàm mất mát (Loss function)

#### Định nghĩa

Loss function (hàm mất mát) là một hàm toán học được sử dụng để đo lường sự sai khác giữa giá trị dự đoán của mô hình và giá trị đích (ground truth). Mục tiêu của hàm mất mát là tối thiểu hóa sai khác này, từ đó cung cấp cho mô hình một phản hồi về chất lượng của dự đoán và hướng dẫn cho mô hình để cải thiện hiệu suất dự đoán.

Trong quá trình huấn luyện mô hình học máy, hàm mất mát được sử dụng để đo lường mức độ "mất mát" của dự đoán so với giá trị đích. Hàm mất mát thường phụ thuộc vào loại bài toán và mục tiêu mong muốn. Một số ví dụ phổ biến của hàm mất mát bao gồm:

* Mean Squared Error (MSE): Hàm mất mát này thường được sử dụng trong các bài toán dự đoán giá trị số. Nó tính toán sai khác bình phương trung bình giữa giá trị dự đoán và giá trị đích. MSE là một hàm lồi và đặc biệt nhạy cảm với các giá trị dự đoán xa giá trị đích.
* Cross-Entropy Loss: Đây là một hàm mất mát thường được sử dụng trong bài toán phân loại và xác suất. Nó đo lường sự khác biệt giữa phân phối xác suất dự đoán và phân phối xác suất thực tế. Hàm cross-entropy đặc biệt hữu ích khi áp dụng trong bài toán phân loại nhị phân hoặc phân loại đa lớp.
* Binary Cross-Entropy Loss: Tương tự như cross-entropy loss, hàm mất mát này được sử dụng trong bài toán phân loại nhị phân, trong đó chỉ có hai lớp đích. Nó đo lường sự khác biệt giữa xác suất dự đoán cho lớp dương và xác suất thực tế của lớp dương.
* Categorical Cross-Entropy Loss: Đây là một hàm mất mát được sử dụng trong bài toán phân loại đa lớp, trong đó có nhiều hơn hai lớp đích. Nó tính toán sai khác giữa phân phối xác suất dự đoán và phân phối xác suất thực tế cho từng lớp.

Đây chỉ là một số ví dụ phổ biến của hàm mất mát trong machine learning. Có nhiều loại hàm mất mát khác nhau tùy thuộc vào loại bài toán và mục tiêu cụ thể. Một số hàm mất mát khác bao gồm hàm Huber loss, hàm hinge loss, hàm log loss, hàm softmax loss, và nhiều hàm mất mát khác.

Khi huấn luyện mô hình, mục tiêu là tìm cách điều chỉnh các tham số của mô hình để giảm thiểu giá trị của hàm mất mát. Quá trình này thường được thực hiện bằng cách sử dụng phương pháp tối ưu hóa như gradient descent để tìm ra các tham số tối ưu nhất. Chọn hàm mất mát phù hợp là một phần quan trọng trong quá trình huấn luyện mô hình. Hàm mất mát phải phù hợp với bài toán cụ thể và đáp ứng các yêu cầu của mục tiêu dự đoán. Ngoài ra, hàm mất mát cũng có thể được kết hợp với các thành phần khác như regularization để kiểm soát overfitting hoặc các hạn chế khác của mô hình.

Với bài toán Speech to text, hàm mất mát được sử dụng là Connectionist Temporal Classification loss (CTC loss)

#### Connectionist Temporal Classification (CTC loss) trong bài toán speech to text

CTC loss được thiết kế để huấn luyện mô hình S2T trong trường hợp không có các đầu ra trung gian (intermediate outputs) cụ thể giữa âm thanh đầu vào và chuỗi ký tự đầu ra. Điều này giúp mô hình có khả năng dự đoán các chuỗi ký tự có độ dài khác nhau mà không cần biết chính xác vị trí của từng ký tự trong âm thanh đầu vào.

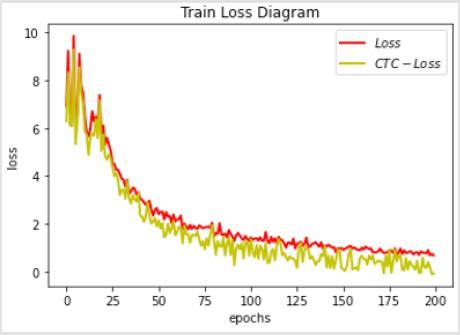
CTC loss tính toán mất mát dựa trên sự khác biệt giữa chuỗi ký tự đầu ra dự đoán của mô hình và chuỗi ký tự thực tế. Nó cũng xem xét các chuyển đổi và các ký tự không hợp lệ trong quá trình chuyển đổi từ âm thanh đến văn bản. Mục tiêu của hàm mất mát CTC là tối thiểu hóa khoảng cách giữa chuỗi ký tự dự đoán và chuỗi ký tự thực tế**.** CTC loss không chỉ định nghĩa một cách tự động hàm mất mát mà còn thực hiện việc giảm bớt độ phức tạp trong việc tính toán đạo hàm và cung cấp khả năng biểu diễn các chuỗi đầu ra có độ dài khác nhau. Để huấn luyện mô hình S2T sử dụng mô hình Transformer, chúng ta tính toán CTC loss dựa trên các chuỗi ký tự đầu ra dự đoán và chuỗi ký tự thực tế, sau đó tối thiểu hóa giá trị mất mát này thông qua quá trình lan truyền ngược (backpropagation) và cập nhật trọng số của mô hình.

CTC loss giúp giải quyết các vấn đề như phân biệt giữa các ký tự liền kề, xác định khoảng cách giữa các ký tự trống và các quy tắc biểu diễn đầu ra khác. Nó cũng cho phép mô hình tạo ra các chuỗi đầu ra có độ dài khác nhau từ các chuỗi dự đoán ban đầu.

Trong bài toán ASR (Automatic Speech Recognition), CTC (Connectionist Temporal Classification) loss được sử dụng để huấn luyện mô hình nhận dạng giọng nói để dự đoán chuỗi ký tự (văn bản) tương ứng với âm thanh đầu vào.

Cơ chế hoạt động của CTC loss trong bài toán ASR có các bước chính như sau:

* Chuẩn bị dữ liệu: Đầu tiên, âm thanh đầu vào được chia thành các khung (frame) nhỏ có độ dài cố định. Mỗi khung âm thanh được ánh xạ vào một tập hợp các ký tự hoặc đơn vị văn bản (ví dụ: alphabet, phonemes) tương ứng.
* Xác định chuỗi đầu ra có thể: Từ âm thanh đầu vào, ta tạo ra một chuỗi ký tự có thể bằng cách loại bỏ các ký tự trống (blank) và các ký tự lặp lại liền kề.
* Định nghĩa ký tự trống và ký tự lặp lại: Trong CTC loss, một ký tự trống được sử dụng để biểu thị các ký tự không hợp lệ hoặc khoảng trắng. Một ký tự lặp lại được sử dụng để biểu thị sự lặp lại của các ký tự liền kề trong chuỗi đầu ra.
* Tạo ra các chuỗi con có thể: Từ chuỗi đầu ra có thể, tất cả các chuỗi con có thể được tạo ra bằng cách loại bỏ các ký tự trống và ký tự lặp lại. Các chuỗi con này tương ứng với các chuỗi dự đoán có thể từ mô hình.
* Tính toán xác suất của chuỗi thực tế: Xác suất của chuỗi thực tế được tính toán bằng cách tổng hợp xác suất của tất cả các chuỗi con có thể tương ứng với chuỗi thực tế. Quá trình này dựa trên mô hình nhận dạng giọng nói.
* Tính toán hàm mất mát CTC: Hàm mất mát CTC được tính toán bằng cách so sánh xác suất của chuỗi thực tế và chuỗi dự đoán. Mục tiêu là tối thiểu hóa khoảng cách giữa chuỗi dự đoán và chuỗi thực tế.



Hình 4.22: Hình ảnh biểu diễn kết quả của minh họa của CTC - loss

Quá trình lan truyền ngược và cập nhật: Quá trình lan truyền ngược (backpropagation) được sử dụng để tính toán đạo hàm của hàm mất mát CTC theo các tham số của mô hình. Sau đó, các tham số được cập nhật bằng phương pháp tối ưu hóa gradient descent để tối thiểu hóa hàm mất mát.

### Phương pháp đánh giá

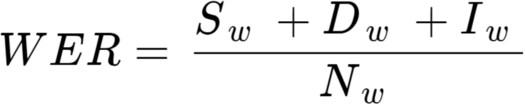
Trong bài toán ASR (Automatic Speech Recognition), để đánh giá độ chính xác của hệ thống nhận dạng giọng nói, các phương pháp đánh giá sau đây thường được sử dụng: Word Error Rate (WER), Character Error Rate (CER), Sentence Error Rate (SER), Phoneme Error Rate (PER).

Các phương pháp đánh giá này đều cung cấp thông tin quan trọng về độ chính xác của hệ thống nhận dạng giọng nói. Thông thường, càng thấp các chỉ số WER, CER thì hệ thống ASR càng chính xác. Đối với mỗi bài toán cụ thể, phương pháp đánh giá phù hợp sẽ được lựa chọn tùy thuộc vào yêu cầu và đặc điểm của tập dữ liệu.

#### Word Error Rate (WER).

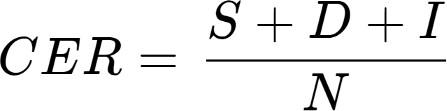
* Word Error Rate (WER): WER là một phương pháp đánh giá phổ biến trong ASR. Nó đo lường tỷ lệ lỗi giữa chuỗi văn bản được nhận dạng và chuỗi văn bản đúng. WER được tính bằng tổng số từ được chèn (insertions), xóa (deletions) và thay đổi (substitutions) chia cho tổng số từ trong chuỗi đúng. Mục tiêu là tìm cách giảm WER càng thấp càng tốt.
* Ví dụ: Văn bản tham chiếu gốc: “My name is kenneth” và đầu ra OCR: “Myy nime iz kenneth” thì ở đây, tỷ lệ lỗi ký tự CER là 16.67%, trong khi tỷ lệ lỗi từ WER là 75%. Rõ ràng WER là 75% vì 3 trong số 4 từ trong câu đã bị phiên âm sai.

Dưới đây là công thức tính tỉ lệ lỗi WER:



#### Character Error Rate (CER).

* Character Error Rate (CER): CER là một phương pháp đánh giá tương tự như WER, nhưng thay vì đánh giá trên từng từ, nó đo lường tỷ lệ lỗi giữa chuỗi ký tự nhận dạng và chuỗi ký tự đúng. CER cũng được tính bằng tổng số ký tự chèn, xóa và thay đổi chia cho tổng số ký tự trong chuỗi đúng.



Trong đó:

* S = Số lần thay thế
* D = Số lần xóa
* I = Số lần chèn
* N = Tổng số ký tự trong văn bản tham chiếu

Mẫu số N có thể được tính theo công thức: N = S + D + C (trong đó C = Số ký tự đúng). Kết quả của phương trình này đại diện cho tỷ lệ phần trăm ký tự trong đầu ra OCR không chính xác so với văn bản tham chiếu đầu vào. Giá trị CER càng thấp (mô hình hoàn hảo khi CER=0), hiệu suất của mô hình OCR càng tốt.

Tùy thuộc trường hợp sử dụng và điều kiện liên quan, tỷ lệ lỗi ký tự CER được sử dụng để đánh giá mức độ hiệu quả của OCR tương ứng. Các tình huống và độ phức tạp khác nhau (ví dụ: văn bản in so với văn bản viết tay, loại nội dung, v.v.) có thể dẫn đến các hiệu suất OCR khác nhau. Tuy nhiên, chúng ta có thể tham khảo đánh giá độ chính xác của một số tổ chức uy tín tại Úc như sau (đối với văn bản in):

* Độ chính xác OCR tốt: CER 1‐2% (tức là chính xác 98–99%)
* Độ chính xác OCR trung bình: CER 2-10%
* Độ chính xác OCR kém: CER> 10% (tức là độ chính xác dưới 90%)

Đối với các trường hợp phức tạp liên quan đến văn bản viết tay có nội dung không đồng nhất và xa rời từ vựng (ví dụ: đơn đăng ký), giá trị CER cao khoảng 20% có thể được coi là đạt yêu cầu.

#### Beam-search output

Beam search là một thuật toán được sử dụng trong các tác vụ tạo ngôn ngữ tự nhiên, bao gồm cả bài toán chuyển đổi giọng nói thành văn bản (ASR). Beam search được sử dụng để tìm kiếm qua không gian đề xuất của các chuỗi kết quả có thể, với mục tiêu tìm ra chuỗi có xác suất cao nhất.

Khi áp dụng beam search trong bài toán ASR, thuật toán sẽ duyệt qua các chuỗi đầu ra có khả năng từ các mô hình âm thành chữ (ASR models) và tìm ra chuỗi có xác suất cao nhất dựa trên mô hình đã huấn luyện.

Beam search hoạt động theo cách sau:

* Ban đầu, thuật toán khởi tạo một số lượng k chuỗi đầu ra (gọi là beam width) với các chuỗi rỗng.
* Tiếp theo, thuật toán duyệt qua các chuỗi trong beam hiện tại và mở rộng chúng bằng cách thêm các từ mới từ không gian từ vựng hoặc thông qua các quy tắc ngữ cảnh.
* Mỗi chuỗi được gắn kèm với một giá trị xác suất dự đoán dựa trên mô hình ASR.
* Beam search tiếp tục lựa chọn và mở rộng các chuỗi có xác suất cao nhất và cắt giảm beam width lại để chỉ giữ lại k chuỗi có xác suất cao nhất.
* Quá trình lựa chọn và mở rộng được lặp lại cho đến khi đạt được điều kiện dừng, chẳng hạn khi đạt tới độ dài tối đa cho chuỗi kết quả hoặc khi không còn chuỗi nào có thể cải thiện đáng kể xác suất.

Beam search sẽ trả về kết quả cuối cùng là chuỗi có xác suất cao nhất trong beam hoặc sau khi đạt được điều kiện dừng. Điều này giúp cải thiện chất lượng của kết quả đầu ra so với việc chỉ lựa chọn từ có xác suất cao nhất tại mỗi bước thời gian như trong một thuật toán Greedy search.

#### Greedy search

Greedy search là một thuật toán đơn giản và phổ biến trong các tác vụ tạo ngôn ngữ tự nhiên, bao gồm cả bài toán chuyển đổi giọng nói thành văn bản (ASR). Thuật toán Greedy search được sử dụng để tạo ra chuỗi kết quả dựa trên các dự đoán có xác suất cao nhất tại mỗi bước thời gian.

Khi áp dụng Greedy search trong bài toán ASR, thuật toán sẽ lựa chọn từ có xác suất dự đoán cao nhất tại mỗi thời điểm và tiếp tục dự đoán cho đến khi đạt được điều kiện dừng, chẳng hạn khi đạt tới độ dài tối đa cho chuỗi kết quả hoặc khi gặp từ kết thúc (như dấu câu chấm, dấu chấm hỏi, dấu chấm than, vv.).

Thuật toán Greedy search có ưu điểm là đơn giản và hiệu quả về mặt tính toán, vì chỉ cần chọn từ có xác suất cao nhất tại mỗi bước. Tuy nhiên, nó có

thể dẫn đến kết quả không tối ưu do việc không xem xét các lựa chọn từ khác có thể tạo ra chuỗi kết quả tốt hơn. Greedy search không đảm bảo tìm được chuỗi có xác suất cao nhất toàn cục, mà chỉ tối ưu hóa từng bước theo xác suất tại thời điểm hiện tại.

Để cải thiện chất lượng của kết quả, thuật toán Beam search và các phương pháp tìm kiếm khác nhưng có quan tâm đến nhiều lựa chọn từ khác nhau đã được phát triển và sử dụng trong các tác vụ tạo ngôn ngữ tự nhiên.

## Chương V: Thực nghiệm

### Môi trường Anaconda.

Anaconda là một môi trường phát triển và quản lý gói phần mềm mã nguồn mở, được sử dụng rộng rãi trong việc phân tích dữ liệu và phát triển các ứng dụng khoa học dữ liệu. Nó cung cấp một bộ công cụ linh hoạt và mạnh mẽ để làm việc với Python và các ngôn ngữ khác như R và Julia.



Hình 5.1: Logo Anaconda

Dưới đây là một số đặc điểm chính của môi trường Anaconda:

* Quản lý gói phần mềm: Anaconda cung cấp một hệ thống quản lý gói phần mềm mạnh mẽ cho Python và các ngôn ngữ khác. Người dùng có thể dễ dàng cài đặt, cập nhật và xóa bỏ các gói phần mềm từ một kho lưu trữ trực tuyến.
* Môi trường ảo: Anaconda cho phép bạn tạo và quản lý các môi trường ảo, giúp tách biệt các phiên bản và phụ thuộc của các gói phần mềm. Điều này cho phép bạn cài đặt các phiên bản khác nhau của các gói và thử nghiệm ứng dụng mà không ảnh hưởng đến môi trường hệ thống chính.
* Công cụ tích hợp: Anaconda đi kèm với một số công cụ tích hợp hữu ích như Jupyter Notebook, JupyterLab và Spyder. Điều này cho phép

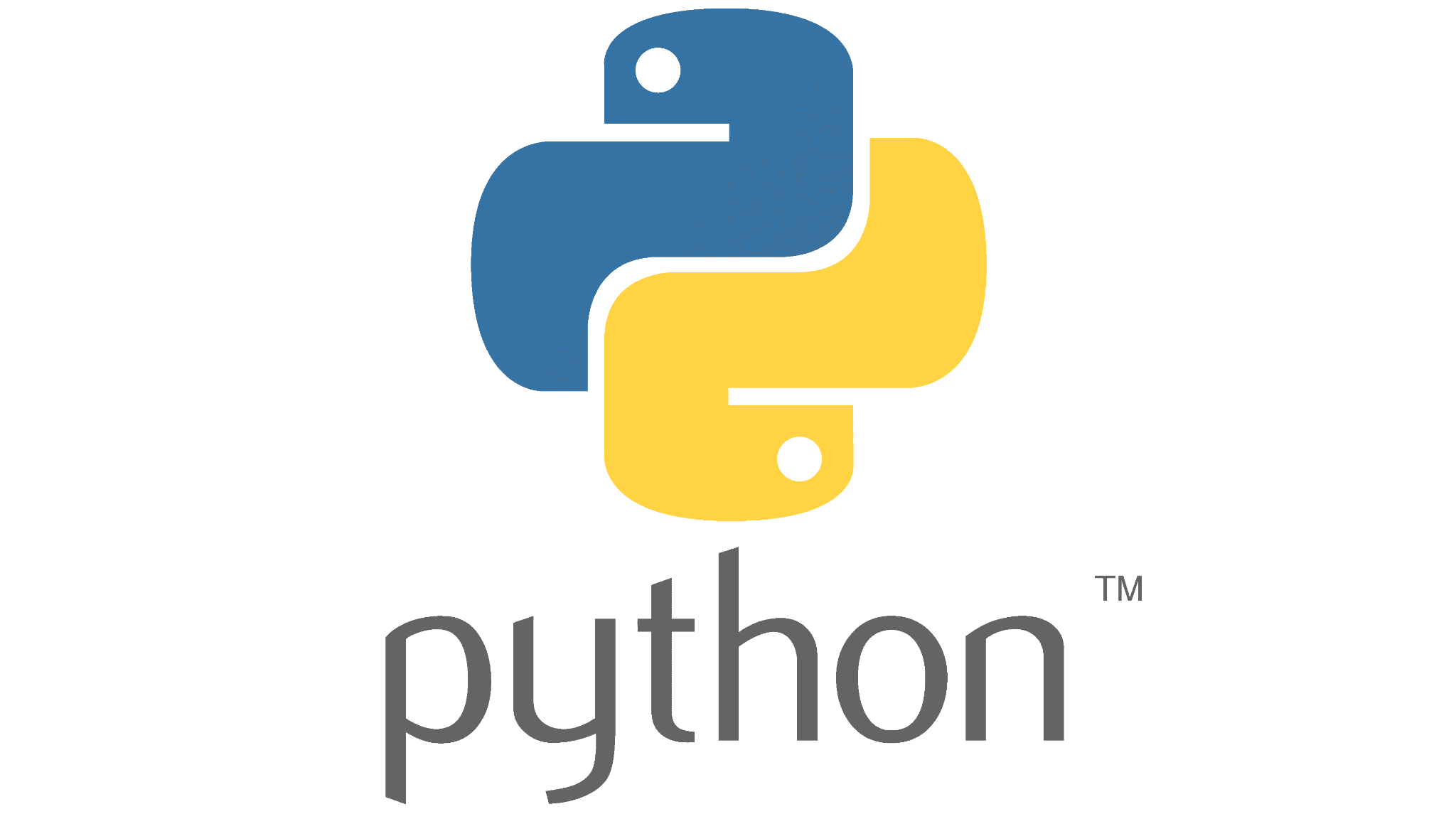
bạn làm việc trực tiếp trong môi trường phát triển tích hợp, cung cấp khả năng tương tác và khám phá dữ liệu thuận tiện.

* Hỗ trợ đa nền tảng: Anaconda hỗ trợ đa nền tảng, cho phép bạn sử dụng môi trường và các công cụ của nó trên các hệ điều hành phổ biến như Windows, macOS và Linux.

Conda giúp chúng ta quản lý thư viện một cách dễ dàng hơn, kiểm tra các phiên bản sao cho phù hợp và có thể tách mỗi project là một môi trường hoàn toàn độc lập với nhau.

### Ngôn ngữ Python

Python là một ngôn ngữ lập trình thông dịch, được phát triển bởi Guido van Rossum và ra mắt lần đầu vào năm 1991. Nó có cú pháp đơn giản, dễ đọc và dễ hiểu, làm cho việc viết mã Python trở nên dễ dàng cho cả người mới học lập trình và những người có kinh nghiệm.



Hình 5.2: Logo python

Dưới đây là một số đặc điểm và tính năng chính của Python:

* Đa mục đích: Python là một ngôn ngữ đa mục đích, có thể được sử dụng để phát triển ứng dụng web, ứng dụng di động, phân tích dữ liệu, máy học (machine learning), trí tuệ nhân tạo (artificial intelligence), và nhiều lĩnh vực lập trình khác.
* Cú pháp đơn giản: Python có cú pháp rõ ràng và đơn giản, với việc sử dụng khoảng trắng để định dạng mã nguồn. Điều này làm cho mã Python dễ đọc và dễ hiểu, giúp tăng tính tương tác và sáng tạo của các nhà lập trình.
* Hỗ trợ thư viện phong phú: Python đi kèm với một số thư viện và module chuẩn mạnh mẽ, cho phép bạn thực hiện nhiều tác vụ thông qua các hàm sẵn có mà không cần phải viết lại từ đầu. Ngoài ra, Python còn có cộng đồng lớn và năng động, sản sinh ra nhiều thư viện bên thứ ba phong phú và mạnh mẽ, từ đó hỗ trợ cho nhiều lĩnh vực và ứng dụng khác nhau.
* Hỗ trợ đa nền tảng: Python là ngôn ngữ đa nền tảng, có thể chạy trên các hệ điều hành phổ biến như Windows, macOS và Linux. Điều này cho phép bạn phát triển ứng dụng Python một cách linh hoạt trên nhiều môi trường.
* Mạnh trong phân tích dữ liệu và máy học: Python là một trong những ngôn ngữ phổ biến nhất trong lĩnh vực phân tích dữ liệu và máy học. Có các thư viện mạnh mẽ như NumPy, Pandas, Matplotlib, scikit-learn và TensorFlow hỗ trợ việc xử lý dữ liệu, phân tích, và xây dựng mô hình máy học.
* Học và sử dụng dễ dàng: Python là một ngôn ngữ lập trình rất dễ học và sử dụng. Cú pháp đơn giản và rõ ràng giúp người mới học nắm bắt nhanh chóng, và cộng đồng Python rất hướng dẫn và hỗ trợ. Ngoài ra, Python cũng có tài liệu phong phú và rất nhiều nguồn tài nguyên trực tuyến, giúp bạn giải quyết vấn đề và tìm kiếm thông tin dễ dàng.
* Tích hợp và mở rộng: Python có khả năng tích hợp với các ngôn ngữ lập trình khác như C/C++, Java, và .NET. Bạn có thể sử dụng Python để viết các phần mở rộng cho ứng dụng sử dụng các ngôn ngữ khác hoặc sử dụng các thư viện có sẵn để tận dụng các tính năng mạnh mẽ từ các ngôn ngữ khác.
* Độ tin cậy và ổn định: Python là một ngôn ngữ ổn định và đáng tin cậy. Nó có cộng đồng lớn và sự hỗ trợ liên tục, giúp người dùng giải quyết các vấn đề và tìm kiếm giải pháp khi gặp lỗi.

Tổng quan, Python là một ngôn ngữ lập trình phổ biến và mạnh mẽ, với cú pháp đơn giản, hỗ trợ thư viện phong phú, và rất dễ học và sử dụng. Nó đã trở thành một công cụ quan trọng trong nhiều lĩnh vực, từ phân tích dữ liệu đến phát triển web và trí tuệ nhân tạo.

### Các thư viện chính sử dụng trong bài toán

#### Thư viện Pytorch



Hình 5.3: Logo Pytorch

PyTorch là một thư viện mã nguồn mở cho việc xây dựng và huấn luyện mô hình học sâu. Nó được phát triển bởi Facebook AI Research và hỗ trợ các tính năng quan trọng để triển khai các mô hình học máy và trí tuệ nhân tạo.

Dưới đây là một tổng quan về các tính năng chính của PyTorch:

* Dynamic Computation Graph: PyTorch sử dụng đồ thị tính toán động, cho phép người dùng định nghĩa và thay đổi các phép tính trong quá trình chạy mà không cần xây dựng lại đồ thị từ đầu. Điều này tạo điều kiện thuận lợi cho việc debug và tạo ra các mô hình phức tạp.
* Tích hợp với Python: PyTorch được thiết kế để tương tác mượt mà với Python và các thư viện phổ biến khác như NumPy. Điều này giúp người dùng dễ dàng sử dụng và tích hợp PyTorch vào quy trình làm việc hiện có của mình.
* Cung cấp các lớp và hàm cơ bản: PyTorch cung cấp một loạt các lớp và hàm cơ bản để xây dựng mô hình học sâu, bao gồm các lớp và hàm cho việc tạo mạng nơ-ron, hàm kích hoạt, lớp tích chập, lớp pooling, và nhiều hơn nữa. Điều này giúp tiết kiệm thời gian và công sức trong việc triển khai mô hình.
* Hỗ trợ GPU: PyTorch hỗ trợ tính toán trên GPU, cho phép tăng tốc độ huấn luyện và triển khai mô hình lên GPU. Người dùng có thể dễ dàng di chuyển dữ liệu và mô hình giữa CPU và GPU sử dụng PyTorch.

PyTorch cung cấp một giao diện trực quan và dễ sử dụng cho việc xây dựng và huấn luyện mô hình học sâu. Nó đã trở thành một trong những thư viện phổ biến và được ưa chuộng trong cộng đồng nghiên cứu và ứng dụng học sâu. Điểm mạnh của PyTorch bao gồm:

* Tính linh hoạt: PyTorch cho phép người dùng tạo và tùy chỉnh các mô hình học sâu theo ý muốn. Người dùng có thể dễ dàng xác định các lớp, hàm kích hoạt, hàm mất mát và tối ưu hóa, và thậm chí tạo ra các kiến trúc mô hình phức tạp.
* Hỗ trợ tính toán tự động (Autograd): PyTorch tự động tính đạo hàm cho các phép tính trong mô hình, giúp người dùng dễ dàng thực hiện lan truyền ngược và cập nhật trọng số mô hình. Điều này giúp rút ngắn quá trình triển khai mô hình và tối ưu hóa.
* Giao diện dễ sử dụng: PyTorch có cú pháp đơn giản và trực quan, giúp người dùng dễ dàng hiểu và sử dụng các hàm và lớp của thư viện. Nó cũng cung cấp nhiều hàm tiện ích và công cụ hỗ trợ cho việc xử lý dữ liệu và triển khai mô hình.
* Tích hợp với các thư viện khác: PyTorch được tích hợp tốt với các thư viện phổ biến khác trong hệ sinh thái Python như NumPy và SciPy. Điều này giúp người dùng dễ dàng làm việc với dữ liệu số, thực hiện

các phép tính mảng và tận dụng các công cụ phân tích dữ liệu mạnh mẽ.

* Hỗ trợ đa nền tảng: PyTorch có thể hoạt động trên nhiều nền tảng khác nhau, bao gồm Linux, Windows và macOS. Nó cũng hỗ trợ các kiến trúc máy tính khác nhau như CPU và GPU.

Tổng quan, PyTorch là một thư viện mạnh mẽ và linh hoạt cho việc xây dựng và huấn luyện mô hình học sâu. Nó đã thu hút sự quan tâm và đóng góp của một cộng đồng lớn, và tiếp tục phát triển và cải tiến để đáp ứng các yêu cầu của người dùng.

#### Thư viện torch audio



Hình 5.4: Logo torch audio

Thư viện Torch Audio là một thư viện mã nguồn mở trong hệ sinh thái PyTorch, được thiết kế để làm việc với âm thanh và dữ liệu âm thanh trong các nhiệm vụ học sâu và xử lý tín hiệu âm thanh. Nó cung cấp các công cụ và chức năng cho việc xử lý, biến đổi và tạo dữ liệu âm thanh, làm giàu dữ liệu và trích xuất đặc trưng âm thanh.

Dưới đây là một tổng quan về các tính năng chính của Torch Audio:

* Các phép biến đổi âm thanh: Torch Audio cung cấp nhiều phép biến đổi âm thanh như thu nhỏ, kéo dài, cắt, nhân đôi, thay đổi âm lượng, chuyển đổi âm thanh thành dạng tín hiệu và ngược lại. Điều này giúp người dùng tiền xử lý và chuẩn bị dữ liệu âm thanh cho quá trình huấn luyện và kiểm thử mô hình.
* Trích xuất đặc trưng âm thanh: Torch Audio cung cấp các phương pháp trích xuất đặc trưng âm thanh như MFCC (Mel-frequency cepstral coefficients), spectrogram, melspectrogram, và nhiều hơn nữa. Điều này cho phép người dùng chuyển đổi dữ liệu âm thanh thành biểu diễn số học có thể được sử dụng trong quá trình huấn luyện mô hình.
* Đa dạng định dạng âm thanh: Torch Audio hỗ trợ nhiều định dạng âm thanh phổ biến như WAV, MP3, FLAC và OGG. Điều này giúp người dùng làm việc với các tệp âm thanh từ các nguồn khác nhau mà không cần phải lo lắng về việc chuyển đổi định dạng.
* Hỗ trợ tích hợp: TorchAudio được tích hợp chặt chẽ với PyTorch, cho phép người dùng dễ dàng tích hợp các phép biến đổi và trích xuất đặc trưng âm thanh vào luồng công việc PyTorch hiện có. Nó cung cấp các lớp và hàm hỗ trợ để xử lý dữ liệu âm thanh trong các mô hình học sâu.

Tổng quan về TorchAudio chỉ ra rằng đây là một thư viện quan trọng và hữu ích trong lĩnh vực xử lý âm thanh và học sâu. Nó cung cấp các công cụ và chức năng cần thiết để xử lý, biến đổi và trích xuất đặc trưng âm thanh, giúp tăng cường quá trình chuẩn bị dữ liệu và xây dựng mô hình cho các nhiệm vụ âm thanh. Kết hợp nó với PyTorch và các thư viện khác, bạn có thể xây dựng và huấn luyện các mô hình học sâu hiệu quả trong các ứng dụng âm thanh.

#### Thư viện transformer

Thư viện Transformer là một thư viện Python được phát triển để triển khai và sử dụng các mô hình Transformer trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP) và học sâu. Nó cung cấp các công cụ và chức năng để xây dựng, huấn luyện và sử dụng các mô hình Transformer phổ biến như BERT, GPT và Transformer Encoder/Decoder.



Hình 5.5: Logo transformer

Dưới đây là một số tính năng chính của thư viện Transformer:

* Xây dựng mô hình Transformer: Thư viện Transformer cung cấp các lớp và công cụ để xây dựng các kiến trúc mô hình Transformer, bao gồm Transformer Encoder và Transformer Decoder. Bạn có thể tùy chỉnh số lớp, số đầu ra, kích thước embedding và các siêu tham số khác của mô hình.
* Tích hợp với các kiến trúc mô hình phổ biến: Thư viện hỗ trợ các kiến trúc mô hình phổ biến dựa trên Transformer như BERT, GPT và Transformer Encoder/Decoder. Bạn có thể sử dụng các kiến trúc này để thực hiện các tác vụ như phân loại văn bản, sinh văn bản và dịch máy.
* Hỗ trợ huấn luyện mô hình: Thư viện cung cấp các công cụ để huấn luyện mô hình Transformer trên các tập dữ liệu. Bạn có thể tùy chỉnh hàm mất mát, thuật toán tối ưu và các tham số huấn luyện khác. Nó cũng hỗ trợ huấn luyện mô hình trên nhiều GPU để tăng tốc quá trình huấn luyện.
* Tích hợp với ngôn ngữ tự nhiên: Thư viện Transformer cung cấp các công cụ và chức năng để tiền xử lý và xử lý ngôn ngữ tự nhiên. Bạn có thể sử dụng các công cụ này để biến đổi văn bản thành dạng vector, thực hiện mã hóa và giải mã, và thực hiện các tác vụ NLP khác như dịch máy và phân loại văn bản.
* Tích hợp với PyTorch và TensorFlow: Thư viện Transformer được thiết kế để tích hợp tốt với các thư viện học sâu phổ biến như PyTorch và TensorFlow. Bạn có thể sử dụng Transformer cùng với các tính năng và công cụ của các thư viện này, bao gồm tích hợp với các lớp mô hình khác, tối ưu hóa và tri
* Tích hợp với các tác vụ NLP: Thư viện Transformer cung cấp các chức năng tiện ích để thực hiện các tác vụ NLP thông qua các mô hình Transformer. Bạn có thể sử dụng nó để thực hiện phân loại văn bản, phân tích cảm xúc, trích xuất thông tin, dịch máy, tự động tạo tiêu đề và nhiều tác vụ NLP khác.
* Tích hợp với mô hình ngôn ngữ tiền huấn luyện: Thư viện Transformer hỗ trợ tích hợp với các mô hình ngôn ngữ tiền huấn luyện như GPT và BERT. Bạn có thể tải xuống và sử dụng các mô hình này trong các tác vụ xử lý ngôn ngữ tự nhiên và học sâu.
* Tài liệu và ví dụ mã mẫu: Thư viện Transformer cung cấp tài liệu và ví dụ mã mẫu để giúp bạn bắt đầu sử dụng và hiểu rõ hơn về cách sử dụng các mô hình Transformer trong các tác vụ NLP. Có các hướng dẫn chi tiết, ví dụ mã và tài liệu tham khảo để bạn học tập và triển khai các ứng dụng của riêng mình.
* Cộng đồng và hỗ trợ: Thư viện Transformer được hỗ trợ bởi một cộng đồng lớn và sôi nổi, trong đó bạn có thể nhận được sự giúp đỡ, chia sẻ kiến thức và thảo luận với những người khác sử dụng thư viện này. Có các diễn đàn, nhóm thảo luận và nhiều nguồn tài nguyên trực tuyến khác để bạn có thể tương tác và học tập từ cộng đồng này.

Tổng quan về thư viện Transformer chỉ ra rằng nó là một công cụ mạnh mẽ để triển khai và sử dụng các mô hình Transformer trong lĩnh vực xử lý ngôn ngữ tự nhiên và học sâu. Bằng cách sử dụng thư viện này, bạn có thể xây dựng các mô hình mạnh mẽ và hiệu quả cho các tác vụ NLP và khám phá nhiều ứng dụng mới trong lĩnh vực này.

#### Thư viện FastAPI

FastAPI là một framework phát triển ứng dụng web nhanh và dễ sử dụng, được xây dựng trên nền tảng Starlette và hỗ trợ Python 3.7+.



Hình 5.6: Logo FastAPI

Dưới đây là một tổng quan về một số tính năng chính của FastAPI:

* Tích hợp dễ dàng: FastAPI tích hợp tốt với các thư viện và công cụ phổ biến như Pydantic, SQLAlchemy, Tortoise ORM, OAuth, etc. Điều này giúp việc phát triển ứng dụng web trở nên dễ dàng và tiện lợi.
* Hiệu suất cao: FastAPI được xây dựng với hiệu suất cao như giao thức ASGI và asyncio. Điều này cho phép xử lý các yêu cầu đồng thời một cách hiệu quả và đáp ứng nhanh chóng.
* Validation và Serialization tự động: FastAPI sử dụng Pydantic, một thư viện validation và serialization mạnh mẽ, để kiểm tra và xử lý dữ liệu đầu vào và đầu ra của API tự động. Việc này giúp giảm lỗi và tăng tính đáng tin cậy của ứng dụng.
* Tài liệu tự động và giao diện API tương tác: FastAPI tự động tạo tài liệu API từ các chú thích trong mã nguồn và cung cấp giao diện tương tác để thử nghiệm và tài liệu hóa API. Điều này giúp giảm thời gian phát triển và cung cấp một cách dễ dàng để khám phá và sử dụng API.
* Hỗ trợ WebSockets: FastAPI hỗ trợ giao thức WebSockets, cho phép xây dựng ứng dụng thời gian thực và truyền tải dữ liệu hai chiều giữa

máy khách và máy chủ.

* Cấu hình đơn giản: FastAPI sử dụng Python Typing để kiểm tra kiểu dữ liệu đầu vào và đầu ra, giúp tạo ra mã nguồn rõ ràng và dễ đọc. Cấu hình đơn giản và ngắn gọn giúp làm tăng tính bảo mật và khả năng bảo trì của ứng dụng.

Với những tính năng mạnh mẽ này, FastAPI đã trở thành một lựa chọn phổ biến cho việc phát triển các ứng dụng web Python nhanh chóng và hiệu quả.

### Bộ dữ liệu huấn luyện

#### Bộ dữ liệu âm thanh LJSpeech

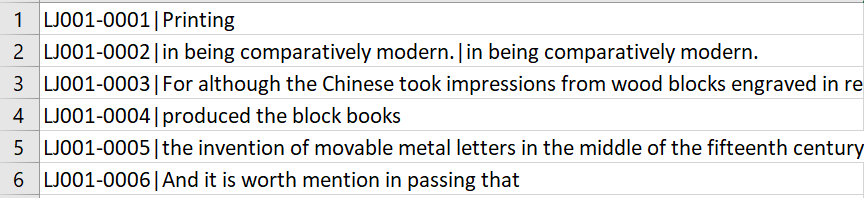
Bộ dữ liệu LJSpeech là một bộ dữ liệu âm thanh được sử dụng phổ biến trong lĩnh vực hợp thành giọng nói tự động (speech-to-text). Nó bao gồm một tập hợp các câu tiếng Anh gốc được phát hiện từ cuốn sách The LJ Speech Dataset, được viết bởi Janet L. Baker. Bộ dữ liệu này đã được chia thành các file âm thanh tương ứng.



Hình 5.7: Hình ảnh bộ dữ liệu LJSpeech

Dưới đây là một tổng quan về bộ dữ liệu LJSpeech:

* Kích thước: Bộ dữ liệu LJSpeech có kích thước khoảng 13GB.
* Số lượng mẫu: Bộ dữ liệu bao gồm 13.100 mẫu câu tiếng Anh.
* Định dạng âm thanh: Các file âm thanh trong bộ dữ liệu thường có định dạng WAV và tỷ lệ mẫu 16kHz.
* Văn bản gốc: Mỗi mẫu âm thanh được gắn kết với văn bản tương ứng trong cuốn sách. Văn bản được cung cấp dưới dạng file văn bản đơn giản (.txt) đi kèm với bộ dữ liệu.
* Tính đa dạng: Bộ dữ liệu bao gồm các loại văn bản khác nhau như tiếng nói thông thường, câu hỏi, trích dẫn và đoạn văn. Điều này giúp đảm bảo tính đa dạng và khả năng ứng dụng của bộ dữ liệu.
* Chất lượng âm thanh: Bộ dữ liệu LJSpeech được ghi âm với chất lượng tương đối cao và cung cấp âm thanh rõ ràng và chân thực. Tuy nhiên, cần lưu ý rằng không phải tất cả các mẫu âm thanh đều có cùng chất lượng và đôi khi có thể xuất hiện nhiễu hoặc sai sót nhỏ trong quá trình ghi âm.
* Bản quyền: Bộ dữ liệu LJSpeech được phát hành dưới giấy phép Creative Commons Attribution 4.0 International (CC BY 4.0). Điều này cho phép người sử dụng sử dụng, sao chép, phân phối và chỉnh sửa bộ dữ liệu với điều kiện cung cấp thông tin về tác giả ban đầu (Janet L. Baker) và công trình gốc.
* Ứng dụng: Bộ dữ liệu LJSpeech có thể được sử dụng để huấn luyện các mô hình hợp thành giọng nói tự động, như Tacotron và WaveNet, để tạo ra giọng nói tổng hợp chất lượng cao. Nó cũng có thể được sử dụng trong các nhiệm vụ nhận dạng giọng nói, dịch thuật giọng nói và các ứng dụng khác liên quan đến xử lý ngôn ngữ tự nhiên.

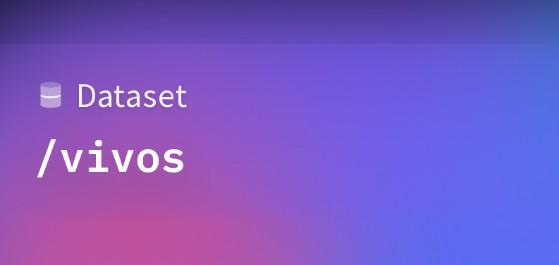


Hình 5.8: Chi tiết về bộ dữ liệu LJSpeech

Bộ dữ liệu LJSpeech đã được sử dụng rộng rãi trong các nhiệm vụ text-to-speech như hợp thành giọng nói tự động và nhận dạng giọng nói. Nó cung cấp một tập hợp lớn các mẫu âm thanh với văn bản tương ứng, tạo điều kiện thuận lợi cho việc huấn luyện và nghiên cứu các mô hình nhận dạng giọng nói và hợp thành giọng nói. LJSpeech cho thấy nó là một nguồn tài nguyên hữu ích và quan trọng trong lĩnh vực hợp thành giọng nói tự động và xử lý giọng nói. Sự đa dạng của văn bản và chất lượng âm thanh trong bộ dữ liệu tạo điều kiện thuận lợi cho việc nghiên cứu và phát triển các mô hình và ứng dụng liên quan.

#### Bộ dữ liệu VIVOS

Bộ dữ liệu VIVOS (Vietnamese Voice Speech Corpus) là một bộ dữ liệu giọng nói tiếng Việt, được sử dụng rộng rãi trong các nhiệm vụ xử lý giọng nói và hợp thành giọng nói tự động. Đây là bộ dữ liệu lớn và chất lượng cao, được phát triển bởi nhóm nghiên cứu VIVOS của Đại học Công nghệ Giao thông Vận tải, Việt Nam.

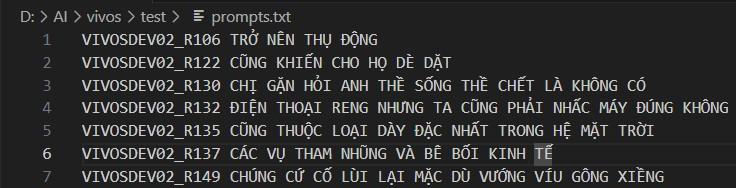


Hình 5.9: Hình ảnh bộ dữ liệu VIVOS

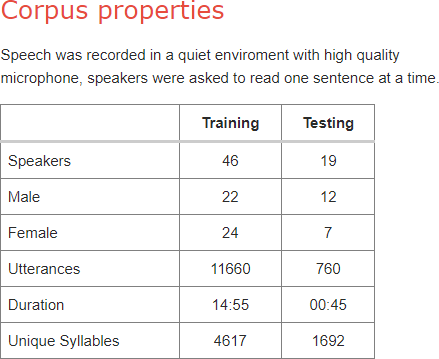
Dưới đây là một tổng quan về bộ dữ liệu VIVOS:

* Kích thước: Bộ dữ liệu VIVOS có kích thước lớn, gồm hàng ngàn câu và hàng giờ giọng nói.
* Số lượng người nói: Bộ dữ liệu bao gồm dữ liệu từ nhiều người nói khác nhau, đảm bảo đa dạng về giọng điệu và phong cách.
* Chất lượng âm thanh: Các file âm thanh trong bộ dữ liệu VIVOS được ghi âm với chất lượng cao, đảm bảo âm thanh rõ ràng và chân thực. Các tài liệu trong bộ dữ liệu bao gồm cả ngữ điệu và đặc điểm ngữ cảnh.
* Đa dạng ngữ cảnh: Bộ dữ liệu cung cấp sự đa dạng về ngữ cảnh, bao gồm các loại văn bản khác nhau như tin tức, tiểu thuyết, truyện cười, bình luận, diễn thuyết và nhiều nội dung khác.
* Bản quyền: Tương tự như LJSpeech, bộ dữ liệu VIVOS được phát hành dưới giấy phép Creative Commons Attribution 4.0 International (CC BY 4.0), cho phép sử dụng, sao chép, phân phối và chỉnh sửa bộ dữ liệu với điều kiện cung cấp thông tin về nguồn gốc.
* Định dạng dữ liệu: Bộ dữ liệu VIVOS cung cấp âm thanh được lưu trữ dưới định dạng WAV, đây là định dạng phổ biến và tương thích với hầu hết các công cụ và thư viện xử lý âm thanh.
* Ghi chú và phụ đề: Mỗi mẫu giọng nói trong bộ dữ liệu VIVOS thường được kèm theo ghi chú và phụ đề tương ứng. Ghi chú cung cấp thông tin về nội dung và ngữ cảnh của mẫu giọng nói, trong khi phụ đề có thể chứa văn bản gốc hoặc trích dẫn từ mẫu giọng nói.
* Ứng dụng và nghiên cứu: Bộ dữ liệu VIVOS có thể được sử dụng trong các ứng dụng và nghiên cứu về xử lý giọng nói tiếng Việt, bao gồm hợp thành giọng nói tự động, nhận dạng giọng nói, chuyển đổi giọng nói và các nhiệm vụ liên quan khác. Nó cung cấp một nguồn tài nguyên quan trọng để phát triển và đánh giá các mô hình và thuật toán xử lý giọng nói.

Bộ dữ liệu VIVOS đã tạo điều kiện thuận lợi cho nghiên cứu và phát triển các ứng dụng về xử lý giọng nói tiếng Việt, bao gồm hợp thành giọng nói tự động, nhận dạng giọng nói và dịch thuật giọng nói. Với chất lượng âm thanh cao và sự đa dạng của ngữ cảnh, nó là một nguồn tài nguyên quý giá trong lĩnh vực xử lý ngôn ngữ tự nhiên tiếng Việt.



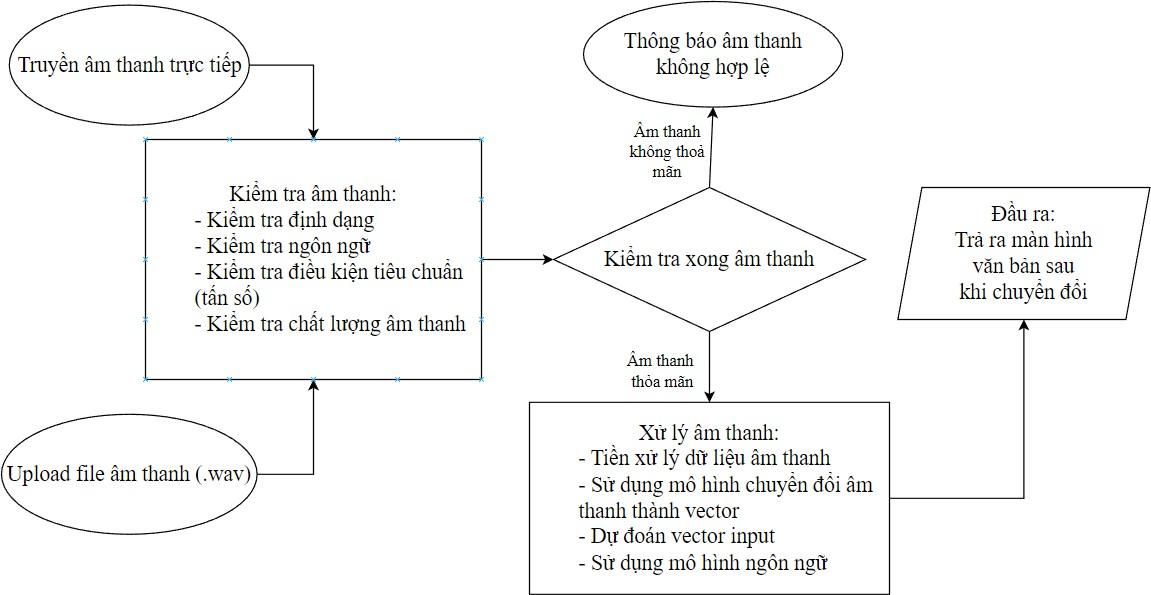
Hình 5.10: Chi tiết về bộ dữ liệu VIVOS



Hình 5.11: Thông tin về bộ dữ liệu VIVOS

### Xây dựng hệ thống

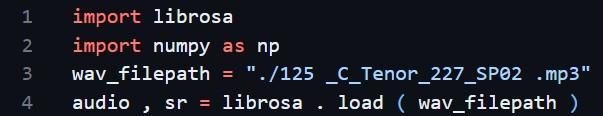
Nhiệm vụ chính của bài toán này là nhận diện âm thanh để từ đó có thể chuyển đổi thành văn bản, nhận biết giọng của người nói tương ứng… Với điểm nổi bật chính là dữ liệu âm thanh, phân tích các kỹ năng xử lý cũng như xây dựng các chức năng và mô hình cho bài toán speech to text



Hình 5.12: Hình ảnh minh họa luồng hoạt động của bài toán

#### Huấn luyện mô hình Transformer với hàm mất mát là CTC loss:

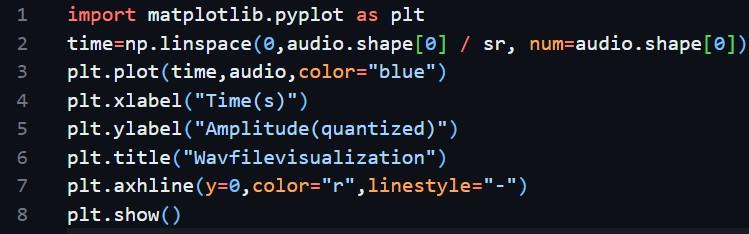
Bước đầu tiên, từ một file âm thanh, ta sẽ đọc file âm thanh (các file âm thanh thường có đuôi .mp3, .wav...) lên thành một vector. Trong python, có rất nhiều thư viện hỗ trợ chúng ta việc này như scipy, pydub... song trong bài này ta sẽ chọn thư viện librosa. Để đọc một file âm thanh, ta dùng lệnh sau:

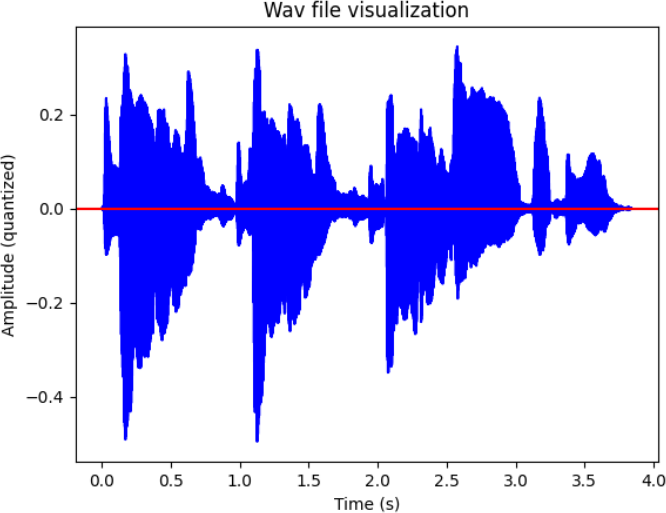


Hàm librosa.load() sẽ trả về 2 biến bao gồm:

* + - * audio: Vector 1D biểu diễn file âm thanh trong time-domain (giá trị amplitude theo thời gian) có shape = (n\_samples,). Đây chính là toàn bộ nội dung của file 125\_C\_Tenor\_227\_SP02.mp3 được đặt trong một

numpy array. Để dễ hình dung hơn, ta có thể trực quan hóa file âm thanh này như sau:





Hình 5.13: Hình ảnh đồ thị trực quan âm thanh dưới dạng time-domain

* + - * sr (sample rate): Tần số lấy mẫu của file âm thanh (mặc định sr=22050). Các giá trị trong file âm thanh (trong audio) được ghi lại theo một chu kỳ tần số nhất định, gọi là tần số lấy mẫu (sampling rate). Theo định nghĩa, tần số lấy mẫu là số mẫu dữ liệu được thu thập trong một giây, đơn vị là Hertz (Hz).

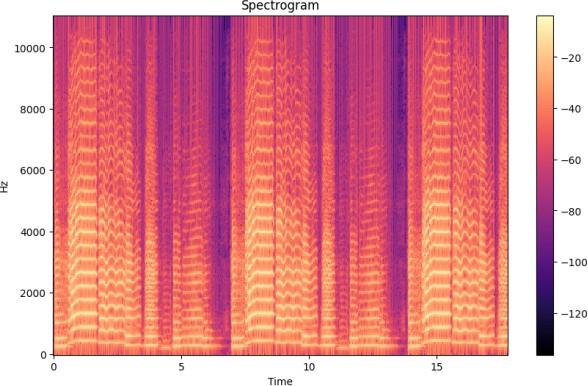


Hình 5.14: Hình ảnh minh họa về sample rate

* + - * Các cột trong hình 5.13 (có thể coi là các điểm dữ liệu) xuất hiện dày đặc hơn khi tần số lấy mẫu tăng.Trong thư viện librosa, ta có thể thay đổi tần số lấy mẫu của file âm thanh thông qua tham số sr trong librosa.load().
      * Và cũng từ tần số lấy mẫu, ta có thể tính được thời lượng của một file âm thanh bằng công thức:

*audio\_time = n\_samples / sampling\_rate*

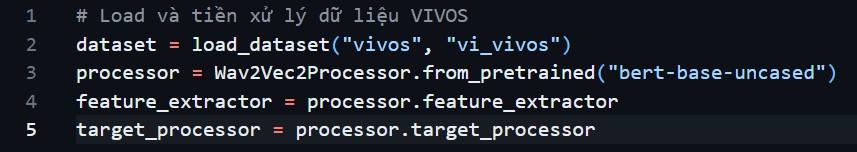
Một vector âm thanh với dạng biểu diễn time-domain, ta sẽ chuyển đổi sang đặc trưng ở dạng ảnh spectrogram. Với spectrogram, ta có thể xem như một bức ảnh đầu vào, sau đó đưa qua các mạng tích chập để tìm các pattern tương tự như với các loại ảnh khác.



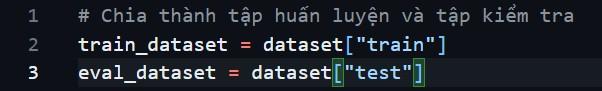
Hình 5.15: Spectrogram của file âm thanh

Bộ dữ liệu gốc không có một file .csv tổng hợp cặp đường dẫn file âm thanh và label tương ứng. Vì vậy, để thuận tiện trong việc xử lý dữ liệu, ta sẽ tạo các file .csv để lưu trữ thông tin này. Code triển khai như sau:

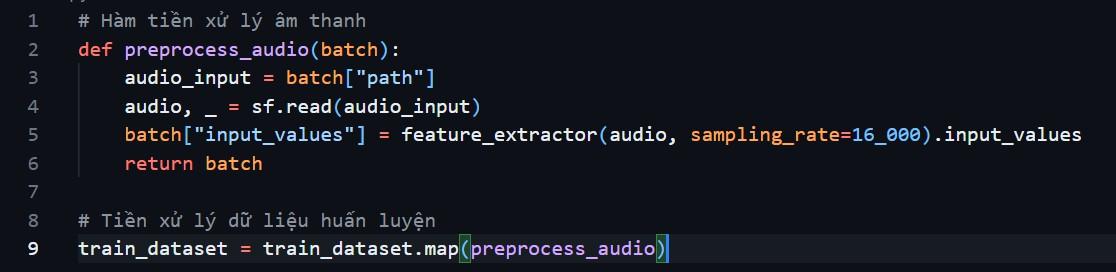
* + - * Load và tiền xử lý bộ dữ liệu:



* + - * Chia tập train/test:



* + - * Khởi tạo hàm tiền xử lý dữ liệu:



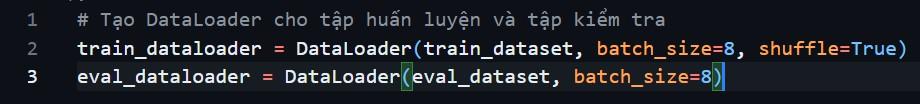
* + - * Khởi tạo hàm tiền xử lý văn bản:



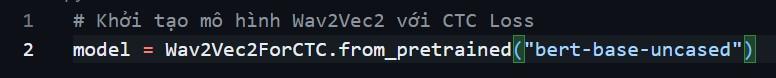
* + - * Tiền xử lý dữ liệu test:



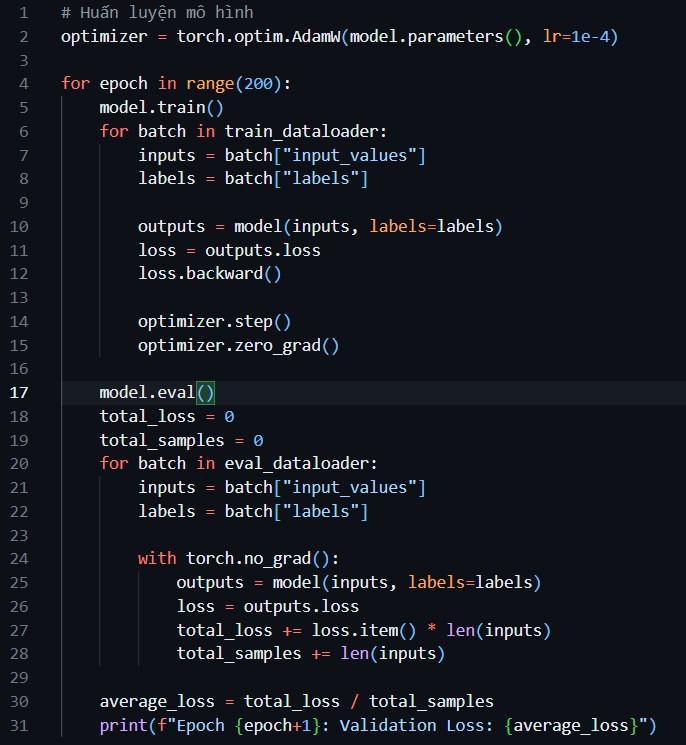
* + - * Load dữ liệu với dataloader:



* + - * Khởi tạo mô hình Wave2vec và CTC loss:



* + - * Huấn luyện mô hình:



#### Kết quả huấn luyện của mô hình



Hình 5.16: Kết quả huấn luyện mô hình

Trong một hệ thống ASR chính thức, cần có hai thành phần: mô hình âm thanh và mô hình ngôn ngữ. Ở đây, mô hình tinh chỉnh ctc-wav2vec hoạt động như một mô hình âm thanh. Đối với mô hình ngôn ngữ, em đã cùng thử nghiệm với 2 phương pháp là có sử dụng mô hình ngôn ngữ và không sử dụng mô hình ngôn ngữ:

* + - * Mô hình có sử dụng mô hình ngôn ngữ, cụ thể là mô hình 4-grams LM thì đạt được WER là 6.15.
      * Mô hình không sử dụng mô hình ngôn ngữ thì có WER là 10.77.

Bảng 1: Bảng kết quả WER khi có mô hình ngôn ngữ và khi không có mô hình ngôn ngữ

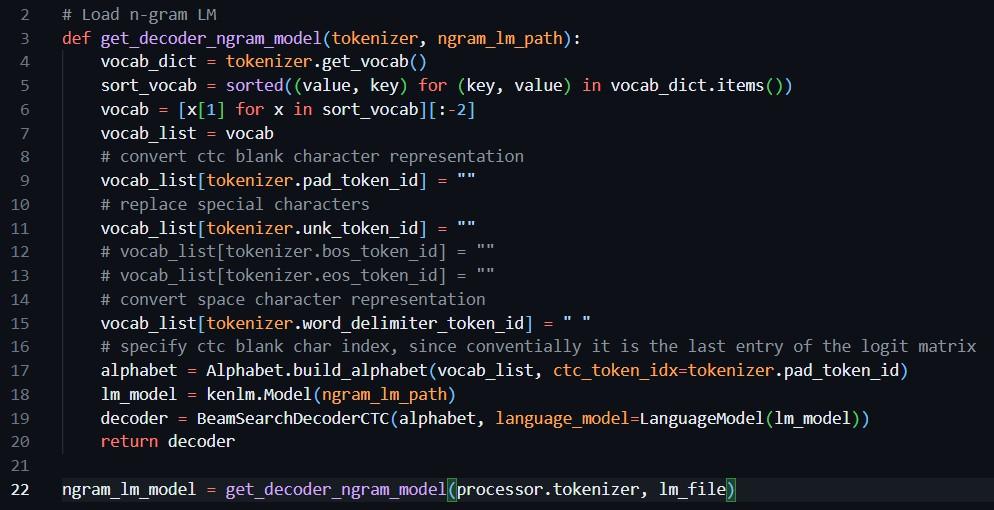
| Mô hình | WER |
| --- | --- |
| có 4-grams LM | 6.15 |
| không có LM | 10.77 |

Từ kết quả trên cho ta thấy rõ rệt tầm quan trọng của mô hình ngôn ngữ trong hệ thống dự đoán âm thanh thành văn bản.

Do giới hạn về tài nguyên phần cứng, việc huấn luyện mô hình diễn ra không được quá lâu. Trong tương lai gần, em sẽ cố gắng cải thiện WER sao cho có kết quả tốt.

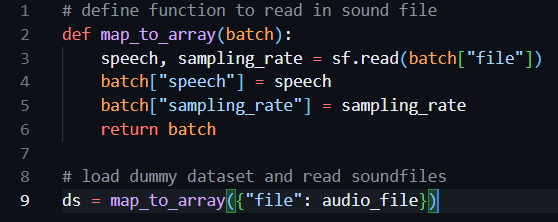
#### Thử nghiệm mô hình và triển khai mô hình.

Tiến hành load mô hình ngôn ngữ, n-grams model:

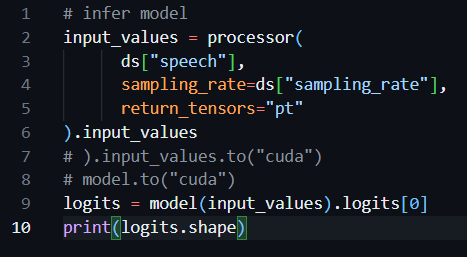


Trong đó:

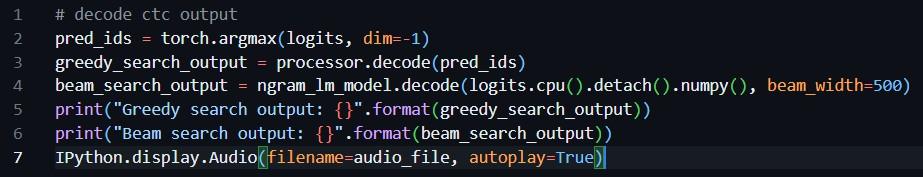
* + - * processor là mô hình xử lý âm thanh giọng nói
      * lm\_model là mô hình ngôn ngữ, cụ thể là vi\_lm\_4grams Thiết lập hàm đọc âm thanh:



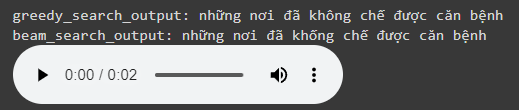
Thực hiện load mô hình và tiến hành dự đoán âm thanh đầu vào



Sau khi thu được output khi đã cho đi qua mô hình, ta tiến hành giải mã



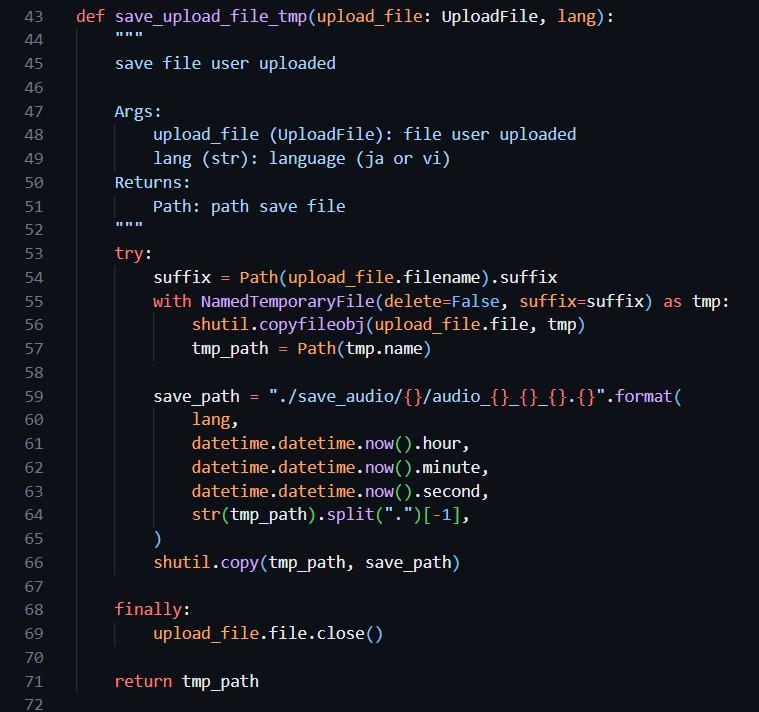
Thu được đầu ra cuối cùng



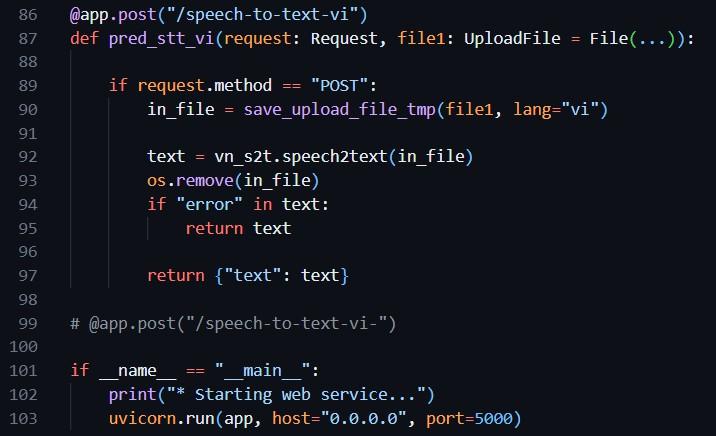
Hình 5.16: Output cuối cùng của mô hình

#### Triển khai mô hình với FastAPI

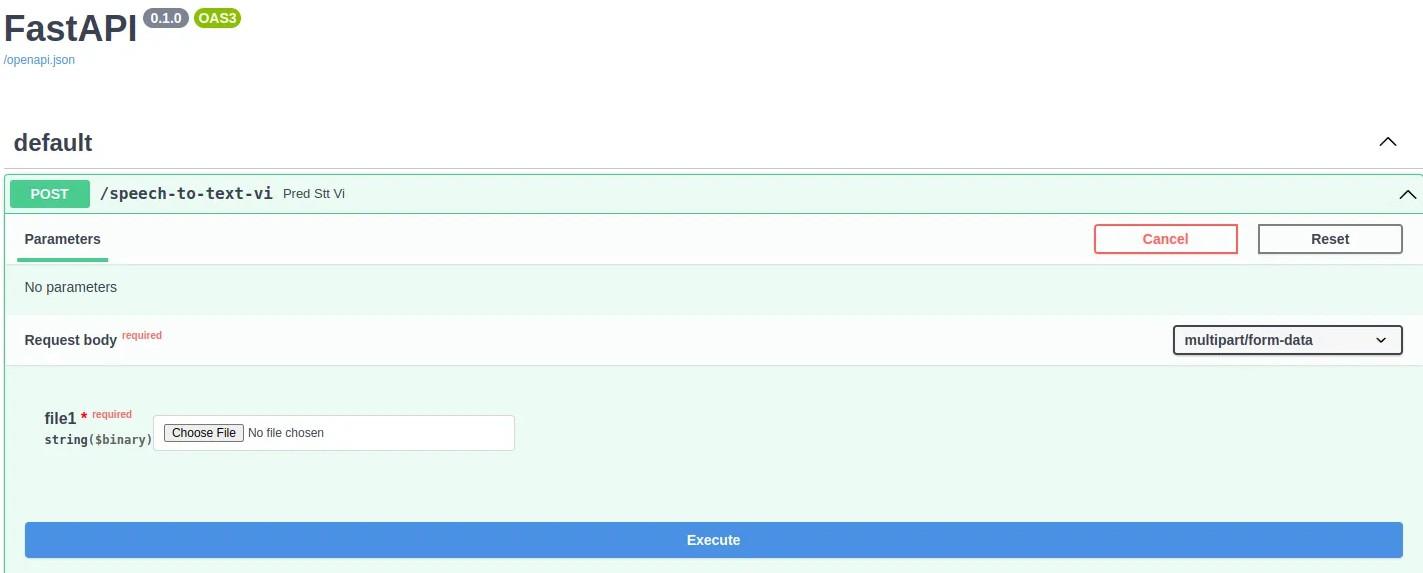
Khởi tạo hàm lưu file: với đầu vào là một file, sau đó tiến hành lưu về máy và trả ra đường dẫn của file đã lưu.



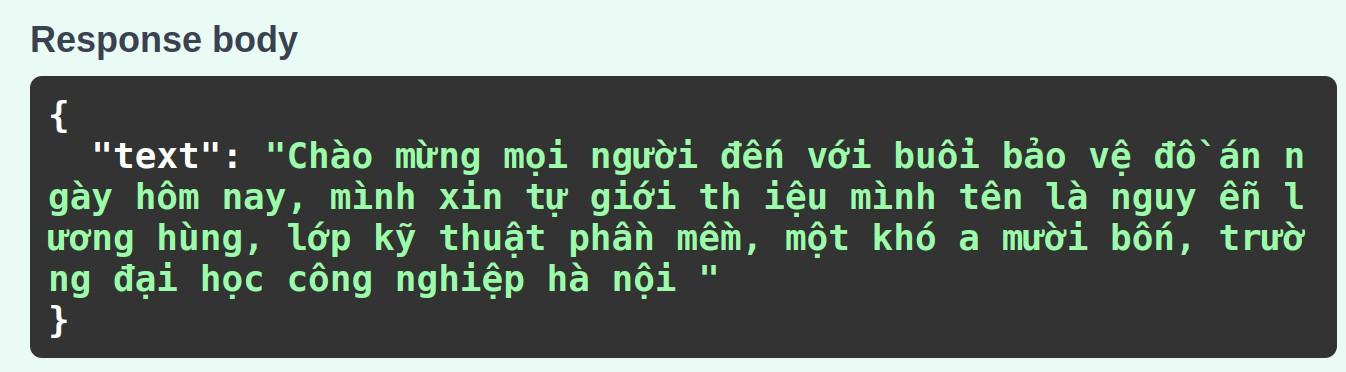
Khởi tạo hàm api chuyển đổi âm thanh thành văn bản: đầu vào là file, trong hàm viết các câu lệnh xử lý. Output là một dictionary chứa câu đã được chuyển từ âm thanh thành văn bản



Giao diện Swagger FastAPI



Hình 5.17: Giao diện Fast API



Hình 5.18: Output cuối cùng nhận được của API

Nội dung của phần output như sau: “Chào mừng mọi người đến với buổi bảo vệ đồ án ngày hôm nay, mình xin tự giới th iệu mình tên là nguy ễn lương hùng, lớp kỹ thuật phần mềm, một khó a mười bốn, trường đại học công nghiệp hà nội”. Nhận thấy kết quả thu được khá tốt trong môi trường ít tiếng ồn và âm thanh đủ lớn. Ta sẽ thu được kết quả thu được đúng 41/41 từ được thể hiện trong file âm thanh.

# KẾT LUẬN

### Kết quả đạt được

Trong thời đại 4.0, AI đang là một xu hướng cực kì mạnh mẽ và đang rất được quan tâm. Công nghệ AI đang được ứng dụng khắp mọi nơi và bài toán Speech to text cũng đang được đầu tư rất nhiều để cho bài toán trở nên hoàn thiện và khả thi hơn. Với mục tiêu học tập và nghiên cứu, các hướng giải quyết để bài toán trở nên gần gũi và dễ tiếp cận hơn. Đề tài “*Xây dựng mô hình chuyển đổi ngôn ngữ từ âm thanh thành văn bản tiếng Việt với cơ chế Attention trong Transformer* ” đã được em nghiên cứu và ấp ủ từ lâu, vì mức độ thử thách của bài toán này khiến em trở nên thích thú và muốn chinh phục.

Đề tài hoàn toàn có thể ứng dụng được trong các buổi họp, đóng vai trò như một thư ký ghi lại lời phát biểu của từng cá nhân. Qua quá trình học tập và rèn luyện tại công ty cổ phần VTI đã giúp em trau dồi kiến thức, bên cạnh đó là sự giúp đỡ tận tình của cô Vũ Thị Tuyết Mai, đã hướng dẫn cho em tận tình để hoàn thành môn đồ án một cách tốt nhất. Và chính bản thân em cũng đã có cho mình được lượng kiến thức nhất định: Hoàn thiện được mục tiêu đặt ra ban đầu của đề tài, xây dựng mô hình, cải tiến, triển khai với API. Hoàn thiện kĩ năng đọc hiểu và làm chủ sản phẩm của mình, tự giác chủ động trong công việc. Nắm chắc về kiến thức Deep Learning nói chung và NLP nói riêng. Tính thực tiễn cao, bước đầu trong quá trình đưa lên sử dụng.

### Hướng phát triển

Vì kĩ năng còn hạn chế và kinh nghiệm chưa đủ nhiều, còn một số chức năng mà em chưa kịp hoàn thiện, em rất mong được quý thầy cô đóng góp ý kiến để giúp em hoàn thiện đề tài của mình trở nên tốt hơn. Hướng phát triển ra các thiết bị edge device, cải tiến tốc độ dự đoán của mô hình và nhúng source code và thiết bị, cải thiện độ chính xác, cũng như xây dựng mô hình dịch âm thanh theo thời gian thực.

# TÀI LIỆU THAM KHẢO

1. Ashish,Vaswani. “Attention Is All You Need.” *arXiv*, 6 12 2017,

https://arxiv.org/pdf/1706.03762.pdf. Accessed 15 May 2023.

1. Chahuneau, Victor. “kpu/kenlm: KenLM: Faster and Smaller Language Model Queries.” *GitHub*, https://github.com/kpu/kenlm. Accessed 15 May 2023.
2. Graves, Alex Graves. “Connectionist Temporal Classification: Labeling Unsegmented Sequence Data with Recurrent Neural Networks.” *Department of Computer Science*, https://[www.cs.toronto.edu/~graves/icml\_2006.pdf.](http://www.cs.toronto.edu/~graves/icml_2006.pdf) Accessed 15 May 2023.
3. Gu, Jiuxiang. “*Recent Advances in Convolutional Neural Networks*” *“arXiv”*, [http://arxiv.org/pdf/1512.07108.pdf.](http://arxiv.org/pdf/1512.07108.pdf) Accessed 15 May 2023.
4. Hever, Gal. “Neural Machine Translation with Transformers | by Gal Hever | Medium.” *Gal Hever*, 2020, https://galhever.medium.com/neural-machine-translation-with-transfor mers-69d4bf918299. Accessed 16 May 2023.
5. IBM. “What are Recurrent Neural Networks?” *IBM*, https://[www.ibm.com/topics/recurrent-neural-networks.](http://www.ibm.com/topics/recurrent-neural-networks) Accessed 15

May 2023.

1. Johnson, Linda, and Keith Ito. “The LJ Speech Dataset.” *Keith Ito*, https://keithito.com/LJ-Speech-Dataset/. Accessed 15 May 2023.

“QUARTZNET: DEEP AUTOMATIC SPEECH RECOGNITION WITH 1D TIME-CHANNEL SEPARABLE CONVOLUTIONS.” *arXiv*, 22 10

2019, https://arxiv.org/pdf/1910.10261.pdf. Accessed 15 May 2023.

1. Tiệp, Vũ Hữu. “Machine Learning cơ bản.” *Machine Learning cơ bản*, https://machinelearningcoban.com. Accessed 15 May 2023.
2. Trường Đại học Công nghệ Giao thông Vận tải. “Download.” *AILab*, [http://ailab.hcmus.edu.vn/vivos.](http://ailab.hcmus.edu.vn/vivos) Accessed 15 May 2023.
3. Tuấn, Nguyễn Thanh. “Sách Deep Learning cơ bản.” *Deep Learning cơ bản*, https://nttuan8.com/sach-deep-learning-co-ban/. Accessed 15 May 2023.
4. VBD. “Word embedding: So sánh các cách tiếp cận truyền thống và hiện đại - Công ty cổ phần Vina Big Data.” *Vina Big Data*, 27 October 2022, https://vinbigdata.com/kham-pha/word-embedding-so-sanh-cac-cach-ti ep-can-truyen-thong-va-hien-dai.html. Accessed 15 May 2023.
5. Zajchowski, Matthew. “Automatic Speech Recognition (ASR) Software - An Introduction.” *Usability Geek*,

https://usabilitygeek.com/automatic-speech-recognition-asr-software-an

-introduction/. Accessed 15 May 2023.