### TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN KHOA CÔNG NGHỆ THÔNG TIN

**BỘ MÔN CÔNG NGHỆ PHẦN MỀM**

**TRẦN KHÁNH LINH – TRẦN THANH TỊNH**

**XÂY DỰNG HỆ THỐNG CUNG CẤP DỊCH VỤ NHẬN DẠNG ÂM THANH TIẾNG VIỆT**

# KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN CNTT

### TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN KHOA CÔNG NGHỆ THÔNG TIN

**BỘ MÔN CÔNG NGHỆ PHẦN MỀM**

**TRẦN KHÁNH LINH - 1612339 TRẦN THANH TỊNH - 1612754**

**XÂY DỰNG HỆ THỐNG CUNG CẤP DỊCH VỤ NHẬN DẠNG ÂM THANH TIẾNG VIỆT**

**KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN CNTT**

GIÁO VIÊN HƯỚNG DẪN

## TS. NGÔ HUY BIÊN

**NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

............................................................................................................................

............................................................................................................................

............................................................................................................................

............................................................................................................................

............................................................................................................................

............................................................................................................................

............................................................................................................................

............................................................................................................................

............................................................................................................................

............................................................................................................................

............................................................................................................................

............................................................................................................................

TpHCM, ngày . . . tháng . . . năm 2020 Giáo viên hướng dẫn

[Kí tên và ghi rõ họ tên]

# NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN

............................................................................................................................

............................................................................................................................

............................................................................................................................

............................................................................................................................

............................................................................................................................

............................................................................................................................

............................................................................................................................

............................................................................................................................

............................................................................................................................

............................................................................................................................

............................................................................................................................

............................................................................................................................

Khoá luận đáp ứng yêu cầu của Khoá luận cử nhân CNTT.

TpHCM, ngày . . . tháng . . . năm 2020 Giáo viên phản biện

[Kí tên và ghi rõ họ tên]

# LỜI CẢM ƠN

Tri ân thầy - Tiến sĩ Ngô Huy Biên, người đã luôn trực tiếp hướng dẫn, định hướng cho hướng đi của luận văn, góp ý giúp đỡ nhiệt tình chúng em trong các vấn đề về kiến thức, nội dung, cách thức trình bày đồng thời luôn tạo điều kiện thoải mái nhất để chúng em có thể hoàn thành khóa luận, chúng em xin gửi đến thầy lời cảm ơn chân thành và sâu sắc nhất.

Chúng em xin gửi lời cảm ơn đến quý Thầy Cô trong khoa công nghệ thông tin của trường đại học Khoa Học Tự Nhiên đã tận tình giảng dạy nâng bước chúng em trong suốt gần 4 năm học vừa qua. Em xin chân thành cảm ơn Khoa Công Nghệ Thông Tin, trường Đại học Khoa Học Tự Nhiên, Đại học Quốc gia Tp. Hồ Chí Minh đã tạo điều kiện thuận lợi cho chúng em trong quá trình học tập và thực hiện đề tài tốt nghiệp. Đồng thời chúng em cũng không quên gửi những lời cảm ơn chân thành đến những người thân trong gia đình và bạn bè đã giúp đỡ chúng em trong quá trình thực hiện luận văn, đặc biệt là quá trình thu thập dữ liệu tiếng nói được sự đóng góp giúp đỡ của mọi người.

Do trình độ nghiên cứu và thời gian có hạn, chúng em đã cố gắng hết sức nhưng chắc chắn không tránh khỏi những thiếu sót và hạn chế. Rất mong nhận được sự góp ý và chỉ dẫn của quý Thầy Cô.

Cuối cùng, chúng em xin trân trọng cảm ơn và chúc sức khỏe quý Thầy Cô!

TpHCM, ngày . . . tháng . . . năm 2020 Sinh viên

Khoa Công Nghệ Thông Tin Bộ môn Công Nghệ Phần Mềm

**ĐỀ CƯƠNG CHI TIẾT**

|  |
| --- |
| **Tên đề tài:** Xây dựng hệ thống cung cấp dịch vụ nhận dạng âm thanh tiếng Việt |
| **Giáo viên hướng dẫn:** TS. Ngô Huy Biên |
| **Thời gian thực hiện:** 15/10/2018 đến 30/06/2019 |
| **Sinh viên thực hiện:** Phan Thị Mỹ Dung - 1512062, Nguyễn Thanh Trí - 1512605 |
| **Loại đề tài:** Tìm hiểu công nghệ có ứng dụng minh hoạ |
| **Nội dung đề tài:**   1. Trình bày lý do xây dựng hệ thống cung cấp dịch vụ web nhận dạng âm thanh tiếng Việt. 2. Xây dựng dịch vụ web (API) để nhận 1 file âm thanh tiếng Việt và xuất ra nội dung ở dạng văn bản (có thể sử dụng các công cụ hay thư viện có sẵn). 3. Quản lý người đăng ký, khóa truy cập, số lượng truy cập vào dịch vụ. 4. Tạo thư viện SDK, ví dụ mẫu và tài liệu hỗ trợ lập trình viên sử dụng dịch vụ, với các ngôn ngữ khác nhau. 5. Thiết kế và hiện thực hóa khả năng mở rộng dịch vụ, khi số lượng truy cập cao. 6. Xây dựng một trang web demo việc sử dụng API để tải lên 1 file âm thanh tiếng Việt và xuất ra nội dung ở dạng văn bản. 7. Xây dựng một ứng dụng demo việc sử dụng SDK. |
| **Kế hoạch thực hiện:** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Thời gian thực hiện** | **Công việc thực hiện** | | **Người thực hiện** |  |
|  | 21/10/2019 - 27/10/2019 | * Nhận đề tài * Xây dựng bản kế hoạch sơ bộ cho các công việc cần thực hiện | | Linh, Tịnh |  |
|  | 28/10/2019 - 10/11/2019 | - Tìm hiểu và phân tích các yêu cầu về kiến thức cho đề tài | | Linh, Tịnh |  |
|  | 11/11/2019 - 24/11/2019 | * Khảo sát và dùng thử các hệ thống cung cấp dịch vụ mẫu có sẵn trên thị trường * Xây dựng 1 bản mẫu prototype, proof of concept (POC) | | Linh, Tịnh |  |
|  | 25/11/2019 - 01/12/2019 | * Viết chương 1 luận văn | | Linh |  |
|  | 02/12/2019 - 08/12/2019 | * Chỉnh sửa chương 1 luận văn | | Tịnh |  |
|  | 09/12/2019 - 15/12/2019 | - Tìm hiểu lý thuyết nền tảng về Mircoservice | | Linh, Tịnh |  |
|  | 16/12/2019 - 29/12/2019 | * Tìm hiểu lý thuyết nền tảng về kiến trúc hệ thống CQRS * Tìm hiểu lý thuyết nền tảng về Event Sourcing | | Linh, Tịnh |  |
|  | 30/12/2019 - 05/01/2020 | * Viết chương 2 luận văn | | Linh |  |
|  | 06/01/2020 - 12/01/2020 | * Xây dựng mô hình mẫu thực hiện việc nhận dạng tập tin âm thanh và xuất ra văn bản cho một từ | | Linh, Tịnh |  |
|  | 13/01/2020 - 19/01/2020 | * Chỉnh sửa chương 2 luận văn | | Tịnh |  |
|  | 20/01/2020 - 26/01/2020 | * Thiết kế kiến trúc hệ thống cung cấp dịch vụ nhận dạng âm thanh tiếng Việt | | Linh, Tịnh |  |
|  | 27/01/2020 - 02/02/2020 | * Chỉnh sửa kiến trúc hệ thống * Xây dựng cơ bản hệ thống cung cấp dịch vụ web (API) * Thiết kế giao diện hệ thống | | Linh, Tịnh |  |
|  | 03/02/2020 - 09/02/2020 | * Viết chương 3 luận văn * Chỉnh sửa chương 3 luận văn | | Linh, Tịnh |  |
|  | 10/02/2020 - 22/03/2020 | * Xây dựng và triển khai hệ thống cung cấp dịch vụ web (API) | | Linh, Tịnh |  |
|  | 23/03/2020 - 29/03/2020 | * Tạo thư viện SDK | | Linh, Tịnh |  |
|  | 30/03/2020 - 05/04/2020 | * Xây dựng một trang web demo việc sử dụng API để tải lên 1 file âm thanh tiếng Việt và xuất ra nội dung ở dạng văn bản | | Linh, Tịnh |  |
|  | 06/04/2020 - 12/04/2020 | * Viết chương 4 luận văn * Chỉnh sửa chương 4 luận văn | | Linh, Tịnh |  |
|  | 13/04/2020 - 26/04/2020 | * Xây dựng một ứng dụng demo việc sử dụng SDK | | Linh, Tịnh |  |
|  | 27/04/2020 - 03/05/2020 | * Viết chương 5 luận văn * Chỉnh sửa chương 5 luận văn | | Linh, Tịnh |  |
|  | 04/05/2020 - 24/05/2020 | * Chỉnh sửa và cải thiện khả năng mở rộng dịch vụ khi số lượng truy cập cao cho hệ thống cung cấp dịch vụ web (API) * Hoàn thiện trang web demo và ứng dụng demo | | Linh, Tịnh |  |
|  | 25/05/2020 - 31/05/2020 | * Hoàn chỉnh cuốn luận văn * Thiết kế slide trình bày luận văn | | Linh, Tịnh |  |
|  | 01/06/2020 - 19/06/2020 | * Hoàn chỉnh slide trình bày * Hoàn chỉnh sản phẩm khoá luận | | Linh, Tịnh |  |
| **Xác nhận của giáo viên hướng dẫn** | | | **Ngày . . . tháng . . . năm 2020 Sinh viên thực hiện** | | |

**MỤC LỤC**

[CHƯƠNG 1: GIỚI THIỆU LUẬN VĂN 6](#_bookmark0)

* 1. [GIỚI THIỆU ĐỀ TÀI 6](#_bookmark1)
     1. [Trợ giúp tự nhiên nhất cho người sử dụng 7](#_bookmark2)
     2. [Sự đa dạng phong phú của tiếng nói làm nên sự đa dạng của cách dịch vụ nhận dạng giọng nói 7](#_bookmark3)
     3. [Tiềm năng phát triển ứng dụng đa dạng 8](#_bookmark4)
     4. [Khả năng tích hợp dịch vụ vào các ứng dụng và thiết bị công nghệ cao 9](#_bookmark5)
     5. [Tiềm năng kinh doanh lợi nhuận cao 10](#_bookmark6)
  2. [LÝ DO LỰA CHỌN ĐỀ TÀI 11](#_bookmark7)
  3. [HƯỚNG PHÁT TRIỂN CỦA LUẬN VĂN 12](#_bookmark8)
  4. [MỤC TIÊU CỦA LUẬN VĂN 13](#_bookmark9)
  5. [PHẠM VI ĐỀ TÀI 14](#_bookmark10)

[CHƯƠNG 2: LÝ THUYẾT NỀN TẢNG 15](#_bookmark11)

* 1. [LÝ THUYẾT NỀN TẢNG CỦA NHẬN DẠNG TIẾNG NÓI 15](#_bookmark12)
     1. [Định nghĩa 15](#_bookmark13)
        1. [Định nghĩa âm thanh số 15](#_bookmark14)
        2. [Định nghĩa nhận dạng tiếng nói 17](#_bookmark15)
     2. [Lý thuyết nền tảng mạng nơ-ron (Neural Networks) 17](#_bookmark16)
        1. [Giải phẫu tế bào mạng nơ-ron 17](#_bookmark17)
        2. [Lan truyền ngược (Backpropagation) 19](#_bookmark18)
        3. [Học với lan truyền ngược 20](#_bookmark19)
        4. [Hàm kích hoạt (Activation function) 22](#_bookmark20)
        5. [Độ giảm dốc theo lô nhỏ (Mini-batch Gradient Descent) 24](#_bookmark21)
        6. [Phương pháp độ giảm dốc với động lượng (Momentum) 25](#_bookmark22)
     3. [Các phương pháp huấn luyện mạng nơ-ron đương đại 27](#_bookmark23)
        1. [Hàm kích hoạt đơn vị tuyến tính chỉnh lưu (ReLu - Rectified Linear Activation Function) 28](#_bookmark24)
        2. [Phương pháp chuẩn hóa theo lô (Batch Normalization) 29](#_bookmark25)
        3. [Phương pháp cắt giảm (Dropout) 31](#_bookmark26)
     4. [Các kiến trúc mạng nơ-ron hồi quy 31](#_bookmark27)
        1. [Mạng nơ-ron hồi quy (RNN - Recurrent Neural Network) 31](#_bookmark28)
        2. [Mạng bộ nhớ dài-ngắn (LSTM - Long Short Term Memory) 35](#_bookmark29)
        3. [Mạng nơ-ron hồi quy hai chiều (BiRNN - Bidirectional recurrent neural network) 37](#_bookmark30)
        4. [Mạng nơ-ron hồi quy học sâu (Deep RNN - Deep Recurrent Neural Network)](#_bookmark31)

[....................................................................................................................................39](#_bookmark31)

* + 1. [Các kỹ thuật trong nhận dạng tiếng nói 39](#_bookmark32)
       1. [Trích chọn đặc trưng (Feature Extraction) và kỹ thuật trích chọn đặc trưng hệ số cepstral tần số Mel (Mel Frequency Cepstral Coefficients - MFCC) 40](#_bookmark33)
       2. [Phân loại thời gian kết nối (CTC - Connectionist Temporal Classification) .41](#_bookmark34)
  1. [HỆ THỐNG NHẬN DẠNG TIẾNG NÓI 43](#_bookmark35)
     1. [Hệ thống nhận dạng tiếng nói thông thường 43](#_bookmark36)
        1. [Trích chọn đặc trưng (Feature Extraction) 44](#_bookmark37)
        2. [Mô hình âm thanh (Acoustic Model) 44](#_bookmark38)
        3. [Mô hình ngôn ngữ (Language Model) 45](#_bookmark39)
        4. [Từ điển phát âm (Lexicon/Pronunciation Dictionary) 46](#_bookmark40)
        5. [Bộ giải mã (Decoding) 46](#_bookmark41)
     2. [Hệ thống nhận dạng tiếng nói hướng đầu cuối (End-to-End) 46](#_bookmark42)
        1. [Giới thiệu và đặt vấn đề 46](#_bookmark43)
        2. [Kiến trúc hệ thống nhận dạng giọng nói đầu cuối (end-to-end) 48](#_bookmark44)

[CHƯƠNG 3: GIẢI PHÁP ĐỀ TÀI 50](#_bookmark45)

* 1. [TỔNG QUAN GIẢI PHÁP KIẾN TRÚC MÔ HÌNH 50](#_bookmark46)
  2. [GIẢI PHÁP RÚT TRÍCH ĐẶC TRƯNG 51](#_bookmark47)
     1. [Tổng quan về giải pháp 51](#_bookmark48)
     2. [Chi tiết giải pháp 51](#_bookmark49)
  3. [GIẢI PHÁP XÂY DỰNG MÔ HÌNH ÂM THANH 54](#_bookmark50)
     1. [Tổng quan về giải pháp 54](#_bookmark51)
     2. [Mô hình mạng nơ-ron hồi quy (RNN) và khung huấn luyện 55](#_bookmark52)
  4. [GIẢI PHÁP TÍCH HỢP MÔ HÌNH NGÔN NGỮ 57](#_bookmark53)
  5. [GIẢI PHÁP XÂY DỰNG MÁY CHỦ 58](#_bookmark54)
  6. [GIẢI PHÁP XÂY DỰNG ỨNG DỤNG 58](#_bookmark55)
     1. [Thiết kế giao diện ứng dụng 59](#_bookmark56)
     2. [Thiết kế kiến trúc ứng dụng 61](#_bookmark57)
  7. [TỔNG KẾT 62](#_bookmark58)

[CHƯƠNG 4: CÀI ĐẶT VÀ TRIỂN KHAI 64](#_bookmark59)

* 1. [GIỚI THIỆU VỀ PYTHON VÀ THƯ VIỆN TENSORFLOW 64](#_bookmark60)
     1. [Python 64](#_bookmark61)
     2. [Tensorflow 65](#_bookmark62)
  2. [CÀI ĐẶT DEEP SPEECH 66](#_bookmark63)
     1. [Giới thiệu 66](#_bookmark64)
     2. [Cài đặt 66](#_bookmark65)
  3. [DỮ LIỆU HUẤN LUYỆN MÔ HÌNH 67](#_bookmark66)
  4. [HUẤN LUYỆN MÔ HÌNH 69](#_bookmark67)
     1. [Điều chỉnh n\_hidden 70](#_bookmark68)
     2. [Điều chỉnh epoch 72](#_bookmark69)
     3. [Điều chỉnh batch\_size 73](#_bookmark70)
     4. [Điều chỉnh dropout\_rate 75](#_bookmark71)
     5. [Điều chỉnh validation\_step 77](#_bookmark72)
  5. [ĐÓNG GÓI MÔ HÌNH 78](#_bookmark73)
  6. [XÂY DỰNG MÁY CHỦ (SERVER) 79](#_bookmark74)
  7. [CÀI ĐẶT CHỨC NĂNG GHI ÂM TRÊN ỨNG DỤNG 80](#_bookmark75)
  8. [CÀI ĐẶT CHỨC NĂNG THÊM BÀI HỌC TRÊN ỨNG DỤNG 81](#_bookmark76)
  9. [CÀI ĐẶT CHỨC NĂNG XUẤT TẬP TIN VĂN BẢN TRÊN ỨNG DỤNG 82](#_bookmark77)
  10. [TỔNG KẾT 82](#_bookmark78)

[CHƯƠNG 5: TỔNG KẾT VÀ ĐÁNH GIÁ 83](#_bookmark79)

* 1. [KIẾN THỨC ĐẠT ĐƯỢC 83](#_bookmark80)
  2. [KẾT QUẢ MÔ HÌNH HUẤN LUYỆN 83](#_bookmark81)
  3. [KẾT QUẢ HỆ THỐNG 84](#_bookmark82)
     1. [Môi trường phát triển 84](#_bookmark83)
     2. [Môi trường triển khai 84](#_bookmark84)

[5.3.4 Chức năng chưa cài đặt 85](#_bookmark85)

* 1. [KẾT QUẢ ỨNG DỤNG 85](#_bookmark86)
     1. [Môi trường phát triển 85](#_bookmark87)
     2. [Môi trường triển khai 85](#_bookmark88)
     3. [Chức năng đã cài đặt 86](#_bookmark89)
  2. [ĐỊNH HƯỚNG PHÁT TRIỂN VÀ NGHIÊN CỨU TRONG TƯƠNG LAI 86](#_bookmark90)
     1. [Công nghệ 86](#_bookmark91)
     2. [Ý tưởng hệ thống 86](#_bookmark92)
  3. [LỜI KẾT 86](#_bookmark93)

[TÀI LIỆU THAM KHẢO 88](#_bookmark94)

[PHỤ LỤC 90](#_bookmark95)

[PHỤ LỤC 1: KIẾN TRÚC MVC 90](#_bookmark96)

[PHỤ LỤC 2: WEB API 92](#_bookmark97)

[PHỤ LỤC 3: GIAO DIỆN CHỨC NĂNG ỨNG DỤNG 94](#_bookmark98)

[Phụ lục 3.1: Màn hình xem danh sách bài giảng 94](#_bookmark99)

[Phụ lục 3.2: Màn hình thêm mới bài giảng 94](#_bookmark100)

[Phụ lục 3.3 Màn hình xem danh sách ghi chú của một bài giảng 95](#_bookmark101)

[Phụ lục 3.4: Màn hình thêm mới ghi chú 96](#_bookmark102)

[Phụ lục 3.5 Màn hình chia sẻ bài giảng 96](#_bookmark103)

[Phụ lục 3.6 Bài giảng sau khi được xuất pdf 97](#_bookmark104)

# CHƯƠNG 1: GIỚI THIỆU LUẬN VĂN

## GIỚI THIỆU ĐỀ TÀI

Trong thời đại bùng nổ công nghệ thông tin hiện nay, nhu cầu người dùng sử dụng các thiết bị công nghệ phục vụ cho nhiều mục đích khác nhau trong đời sống là vô cùng to lớn. Càng ngày thời gian rảnh tay của con người càng bị thu hẹp, con người mong muốn có thêm những công cụ hỗ trợ họ một cách tự động và tự nhiên hơn thay vì họ phải trực tiếp sử dụng ứng dụng và tìm kiếm chức năng phục vụ tương ứng.

Âm thanh hay cụ thể hơn là tiếng nói là một trong những phát minh vĩ đại nhất của con người, giúp con người tách biệt so với phần còn lại. Con người sử dụng tiếng nói để giao tiếp với nhau dễ dàng hơn. Tiếng nói ẩn chứa trong đó một lượng thông tin vô cùng phong phú và phức tạp. Trong những năm gần đây, các nhà phát triển đã dần chuyển đổi cách thức giao tiếp giữa con người và máy tính từ việc gõ phím cứng rồi đến giao diện cảm ứng và nay là chuyển sang giao tiếp qua giọng nói. Nhận thấy được tiềm năng phát triển to lớn như vậy, hàng loạt các dịch vụ nhận dạng âm thanh tiếng nói cũng như là ứng dụng sử dụng dịch vụ này đã ra đời.

Bằng những đặc tính hấp dẫn của mình, dịch vụ nhận dạng tiếng nói cũng như ứng dụng sử dụng dịch vụ này không chỉ hấp dẫn người sử dụng mà còn thu hút nhiều nhà đầu tư công nghệ trên thế giới. Một dịch vụ nhận dạng tiếng nói sở hữu những đặc điểm thu hút sau:

* + - Trợ giúp tự nhiên nhất cho người sử dụng.
    - Sự đa dạng phong phú của tiếng nói làm nên sự đa dạng của cách dịch vụ nhận dạng giọng nói.
    - Khả năng tích hợp dịch vụ vào các ứng dụng và thiết bị công nghệ khác là rất cao.
    - Tiềm năng kinh doanh lợi nhuận là rất to lớn.

Sau đây sinh viên sẽ điểm qua chi tiết về các đặc điểm hấp dẫn của dịch vụ loại hình này để làm rõ lý do khiến nó trở thành mối quan tâm của các nhà đầu tư và người sử dụng hiện nay.

### Trợ giúp tự nhiên nhất cho người sử dụng

Trong cuộc sống, đôi khi con người cần phải làm rất nhiều việc một lúc cùng nhau, ví dụ như một nhân viên văn phòng khi đi họp, ngoài việc nói chuyện thảo luận với khách hàng, họ cũng cần phải liên tục ghi chú lại những điểm quan trọng trong cuộc nói chuyện. Hay một người nội trợ muốn vừa chuẩn bị bữa ăn vừa nhắn tin với bạn bè, hoặc một người lái xe hơi muốn bật nhạc trong lúc họ đang lái xe. Tất cả những tác vụ đó nếu phải thực hiện bằng tay sẽ gây nên nhiều trở ngại.

Với một dịch vụ nhận dạng giọng nói có chất lượng tốt, nếu được tích hợp vào các ứng dụng trợ lý ảo, nhắn tin, nghe nhạc... các nhà phát triển có thể giải quyết tất cả những trở ngại nói trên cho người dùng và dần đem lại cho họ những trải nghiệm tự động tự nhiên nhất trên mọi công việc thường nhật của đời sống.

### Sự đa dạng phong phú của tiếng nói làm nên sự đa dạng của cách dịch vụ nhận dạng giọng nói

Có tới gần ba nghìn thứ tiếng trên toàn thế giới, mỗi tiếng nói lại bị phân hóa theo từng vùng miền, quốc gia mà các cách đọc khác nhau dẫn đến sự hình thành của nhiều phương ngữ lẫn các biến thể ngôn ngữ, dịch vụ nhận dạng tiếng nói vì thế mà bị giới hạn đi rất nhiều. Ở châu Âu và châu Mỹ đã xuất hiện rất nhiều các dịch vụ nhận dạng tiếng nói, tiêu biểu như Google Speech, Microsoft Azure (Speech Service). Tuy nhiên, các dịch vụ hiện có này chỉ thực sự thành công và đạt được kết quả chính xác cao trong nhận dạng tiếng Anh Mỹ, còn những thứ tiếng khác hoặc những biến thể khác thì độ chính xác thấp hơn nhiều.

Vấn đề đặt ra khi sử dụng dịch vụ này, nhà phát triển ứng dụng chỉ quan tâm về độ hiệu quả của dịch vụ cho loại ngôn ngữ chính đang sử dụng tại thị trường của họ. Ví dụ như ở thị trường Việt Nam, ứng dụng làm ra là chủ yếu cho người Việt Nam sử dụng, nhà đầu tư cần một dịch vụ phục vụ hiệu quả nhất khi nhận dạng ngôn ngữ tiếng Việt. Điều này đặt ra một vách ngăn vô hình, các dịch vụ được phát triển bởi các nước phương Tây không thể đạt được độ hiệu quả mong đợi do họ không có đủ nguồn dữ liệu và vốn kiến thức về giọng nói tiếng Việt. Nhưng nếu dịch vụ do chính người Việt Nam phát triển thì đây hoàn toàn là lợi thế. Như vậy sự phong phú của tiếng nói là nên sự đa dạng cho các dịch vụ nhận dạng tiếng nói.

### Tiềm năng phát triển ứng dụng đa dạng

Dữ liệu âm thanh là một loại dữ liệu ẩn chứa trong đó nhiều thông tin riêng biệt, mà thông qua những thông tin được trích xuất ra từ các loại hình dữ liệu này, ta có thể vận dụng chúng vào trong những mục đích khác nhau. Sau đây sinh viên sẽ điểm qua các sản phẩm nổi bật để làm rõ sự đa dạng của loại hình ứng dụng này trên nhiều lĩnh vực.

* + - * Amazon Echo

Amazon Echo là một thiết bị gia đình thông minh có cơ chế hoạt động giống một chiếc loa cầm tay. Bạn có thể thực hiện nhiều việc với Echo bằng cách ra lệnh giọng nói như: nghe nhạc, thiết lập lịch trình, cập nhật tin tức, … Tuy nhiên mục tiêu của Amazon Echo ra đời là không phải là để cạnh tranh trong lĩnh vực nhận diện giọng nói, mà là để phục vụ cho việc bán hàng qua mạng. Thay vì người dùng phải lên trang web tìm kiếm sản phẩm, thì nay có thể ra lệnh cho Echo thực hiện việc đó thông qua giọng nói.

* + - * Google Assistant

Google Assistant là một trợ lý ảo cá nhân được phát triển bởi Google. Đây là một sản phẩm sinh ra cho cuộc cạnh tranh trong lĩnh vực nhận diện giọng nói, người dùng có thể tìm kiếm trên Internet, đặt lịch sự kiện và báo thức, tham gia vào các cuộc trò chuyện hai chiều giữa phần mềm và người dùng. Sự kết hợp giữa Google Assistant với Google

Maps, Google Photos, Yotube và một loạt các dịch vụ tiện ích khác tạo nên hệ sinh thái vô cùng mạnh mẽ.

* + - * Cortana

Cortana là một trợ lý ảo được phát triển bởi Microsoft. Nhưng thay vì bước vào cuộc cạnh tranh trong lĩnh vực nhận diện giọng nói với Google Assistant, hay phục vụ bán hàng như Amazon Echo, thì Microsoft hướng Cortana là một công cụ tăng năng suất trong phần mềm và tập trung mạnh vào các doanh nghiệp. Tương lai của Cortana có thể không phải là một sản phẩm tiêu dùng thú vị mà là xương sống cho các giải pháp trợ lý ảo mà các doanh nghiệp có thể tùy chỉnh thành trợ lý ảo hoặc chatbox riêng của họ.

* + - * Siri

Siri là một ứng dụng trợ lý ảo phát triển bởi Apple cho phép nhận diện giọng nói để làm các công việc mà bình thường phải làm bằng tay ví dụ như: gọi điện, soạn và gửi tin nhắn, mở đèn flash, mở ứng dụng cài trên máy, kiểm tra thời tiết, tạo các ghi chú, tìm đường, gửi email, bật nhạc hay điều chỉnh độ sáng màn hình… Người sử dụng có thể đưa ra câu hỏi và Siri sẽ tìm ra câu trả lời, hoặc người sử dụng có thể ra yêu cầu để Siri thực hiện trong một phạm vi cho phép.

Thông qua các ứng dụng đã trình bày, ta có thể thấy được nguồn dữ liệu âm thanh là một nguồn dữ liệu bao hàm rất nhiều thông tin, và bằng những kĩ thuật khai thác riêng biệt, ta có thể tận dụng nguồn thông tin này để phục vụ cho nhiều lĩnh vực kinh doanh chiến lược khác nhau.

### Khả năng tích hợp dịch vụ vào các ứng dụng và thiết bị công nghệ cao

Hiện nay, với nhu cầu sử dụng ngày càng nhiều ở thị trường di động, việc phát triển ra một ý tưởng mới cho các nhà phát triển phần mềm ngày càng trở nên khó khăn. Dưới sự ra đời của một hệ thống nhận dạng tiếng nói hợp lý, các nhà phát triển phần mềm không chỉ có thể sáng tạo ra nhiều ý tưởng sản phẩm mới mà còn có thể nâng cấp cải tiến cách

thức sử dụng các sản phẩm có sẵn, từ đó ngày càng tạo ra sự tiện dụng và thu hút khách hàng nhiều hơn.

Ví dụ như Google Maps trước đây khi người dùng muốn tìm đường đến một địa điểm nào đó, họ phải nhập địa điểm và thực hiện các thao tác khá phức tạp khác, nhưng với việc tích hợp Google Assistant, Google Maps hiện nay đã cho phép người dùng điều khiển được việc chuyển hướng, chỉ đường bằng giọng nói trong khi họ đang bận rộn với việc lái xe và không thể sử dụng tay để điều khiển ứng dụng.

Trong tháng 12 vừa qua, ở Việt Nam, tập đoàn Zalo cũng giới thiệu tới cộng đồng công nghệ sự ra đời của một phần mềm trợ lý ảo mang tên Ki-Ki. Sự ra đời của trợ lý ảo này hứa hẹn là một cổng kết nối toàn bộ hệ sinh thái các sản phẩm phần mềm của Zalo (như Zalo, Zing Mp3, Báo Mới, …) nhằm đem lại những trải nghiệm tự động nhất cho người dùng trên mọi phương diện.

### Tiềm năng kinh doanh lợi nhuận cao

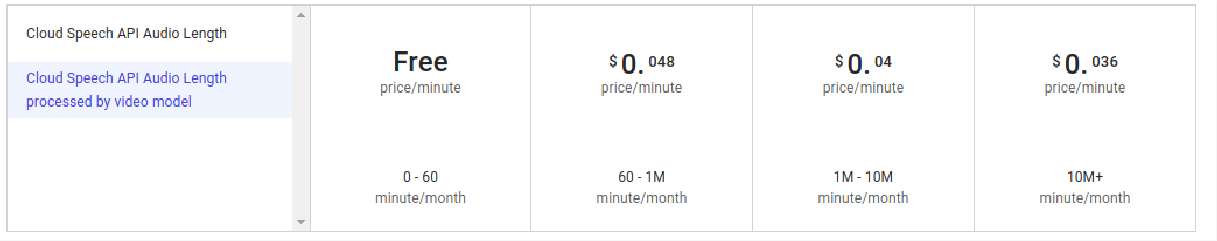
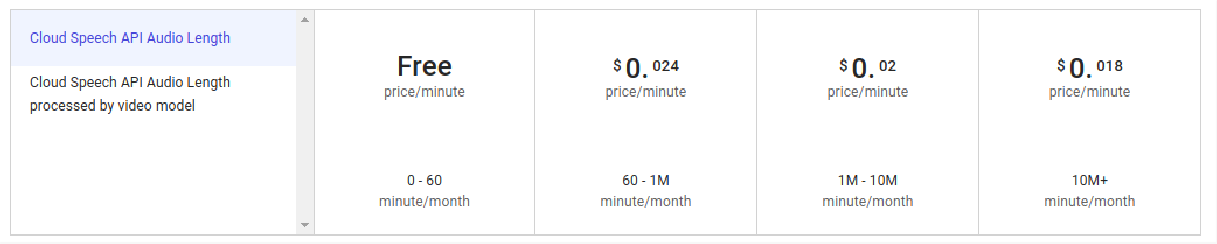
Ứng dụng cho phép người dùng giao tiếp bằng giọng nói đang thu hút một lượng lớn người dùng trên nhiều độ tuổi, đặc biệt là nhóm người dùng bận rộn với nhiều việc cùng phải làm trong cùng lúc. Đối với những ứng dụng đã có sẵn trên thị trường hoàn toàn có thể sử dụng dịch vụ này để nâng cấp chức năng cho phép người dùng ra lệnh bằng giọng nói, còn đối với những ứng dụng mới, hoàn toàn có thể dựa vào việc khai thác dịch vụ này để phục vụ ý tưởng cho nhà phát triển phần mềm.

Với những công ty công nghệ phần mềm và thiết bị phần cứng, có những sự lựa chọn như sử dụng tiếp dịch vụ nhận dạng tiếng nói hoặc tùy chỉnh lại dịch vụ để phục vụ cho mục đích riêng của họ. Khả năng tích hợp dịch vụ này vào các ứng dụng phần mềm và các thiết bị phần cứng là rất cao, làm cho thị trường nhận dạng giọng nói phong phú, đa dạng và vì thế lợi nhuận đem lại từ thị trường này là rất to lớn.

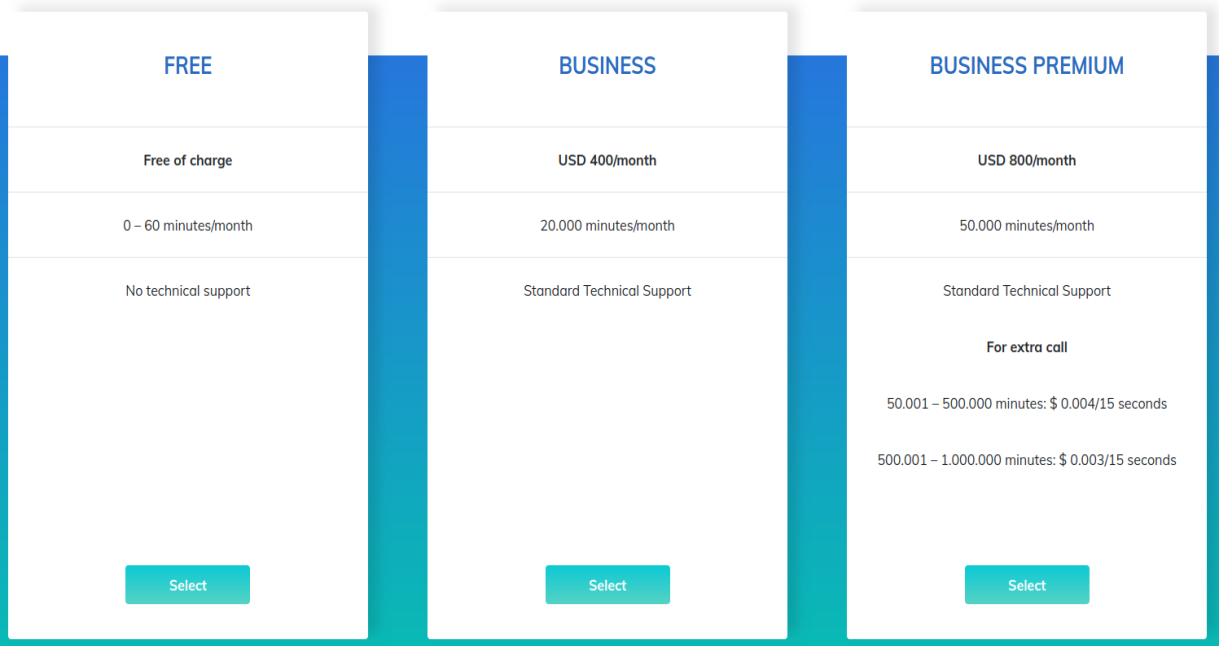
## LÝ DO LỰA CHỌN ĐỀ TÀI

Nhằm mục đích hệ thống hóa các kiến thức của bản thân trong quá trình học tập vào một sản phẩm có ý nghĩa thực tế, có tiềm năng đầu tư cao trong tương lai, nhóm sinh viên lựa chọn đề tài “Xây dựng hệ thống cung cấp dịch vụ nhận dạng âm thanh tiếng Việt”.

Như đã trình bày ở trên, ứng dụng sử dụng dịch vụ nhận diện giọng nói ngày càng thu hút các nhà phát triển phần mềm lẫn người tiêu dùng phần mềm. Thị trường Việt Nam đang là thị trường tiềm năng vì đây là lĩnh vực mới. Hiện nay hầu hết các ứng dụng nhận dạng giọng nói tiếng Việt đang sử dụng dịch vụ từ Google Cloud Speech API và một phần nhỏ sử dụng dịch vụ từ OpenFPT Speech nhưng độ chính xác từ hai dịch vụ trên là không cao. Độ chính xác không cao là do tiếng Việt có cấu trúc văn phạm không rõ ràng và sự phong phú trong từ vựng. Nhưng điều đáng nói hơn là chi phí chi trả cho việc sử dụng các dịch vụ được phát triển sẵn như vậy là khá cao. Lấy ví dụ như Google Cloud Speech API, thì mức phí là:



Hình 1.5: Mức phí cho dịch vụ của Google (Nguồn: [cloud.google.com]) Chi phí cho dịch vụ được cung cấp bởi OpenFPT Speech là:



Hình 1.6: Mức phí cho dịch vụ của FPT (Nguồn: [openfpt.vn])

Xét trong phân khúc thị trường ở Việt Nam, một dịch vụ nhận diện giọng nói tiếng Việt với độ chính xác chưa cao và mức phí khá đắt đỏ đang vô hình làm chậm lại sự phát triển của các phần mềm tự động. Các nhà đầu tư mới bắt đầu sẽ không có đủ nguồn lực và nguồn vốn để chi trả cho việc phát triển một ứng dụng có tích hợp dịch vụ nhận diện giọng nói tiếng Việt.

Với hai lý do kể trên, nhóm sinh viên quyết định chọn đề tài “Xây dựng hệ thống cung cấp dịch vụ nhận dạng tiếng Việt” để tạo ra một dịch vụ có mức phí thấp hơn (hoặc miễn phí) và với độ chính xác chấp nhận được.

## HƯỚNG PHÁT TRIỂN CỦA LUẬN VĂN

Tuy dịch vụ nhận dạng âm thanh trên thị trường khá nhiều, nhưng dịch vụ hỗ trợ nhận dạng âm thanh tiếng Việt lại thực sự hiếm hoi và còn nhiều hạn chế. Độ chính xác về khả năng nhận dạng âm thanh tiếng Việt chưa được tối ưu hóa và đồng thời mức chi phí cho dịch vụ lại khá tốn kém. Điều này làm cho việc tích hợp dịch vụ nhận dạng âm thanh (ASR) vào ứng dụng ở Việt Nam còn gặp nhiều hạn chế nên chưa tận dụng và phát huy

được tối ưu các tiềm năng trong lĩnh vực này. Vì vậy, để hoàn thành tốt đề tài luận văn, nhóm sinh viên tiến hành xây dựng một hệ thống cung cấp dịch vụ nhận dạng âm thanh tiếng Việt sẽ hỗ trợ tối thiểu những tiêu chí sau:

* + - Hỗ trợ nhận dạng âm thanh tiếng Việt từ tập âm thanh tiếng Việt đã thu được với độ chính xác trên 75%.
    - Hỗ trợ nhận dạng âm thanh tiếng Việt phương ngữ từ 70-80%.
    - Mức chi phí để sử dụng dịch vụ là thấp nhất có thể hoặc miễn phí.

### MỤC TIÊU CỦA LUẬN VĂN

Để hoàn thành tốt đề tài luận văn, bản luận văn và sản phẩm hệ thống dịch vụ cuối cùng của sinh viên sẽ đảm bảo tối thiểu các mục tiêu sau đây:

* + - Bản luận văn trình bày chi tiết, rõ ràng về lý do xây dựng hệ thống cung cấp dịch vụ web (API) nhận dạng âm thanh tiếng Việt.
    - Bản luận văn trình bày các lý thuyết nền tảng và giải pháp để xử lý một tập tin âm thanh tiếng Việt ở đầu vào và trả về nội dung ở dạng văn bản ở đầu ra. Từ đó làm rõ các tính năng của hệ thống dịch vụ mẫu cần được xây dựng lại và trình bày được các ưu khuyết điểm của tính năng mà sinh viên lựa chọn để tập trung phát triển cho sản phẩm luận văn của nhóm. Từ các khuyết điểm của hệ thống dịch vụ mẫu, luận văn sẽ trình bày về các giải pháp cũng như những lý thuyết liên quan để cải tiến những tiêu chí mà hệ thống dịch vụ mẫu còn hạn chế hay chưa tối ưu. Bên cạnh đó, luận văn cũng trình bày các quy trình và cơ cấu cụ thể của sản phẩm luận văn để đạt được sản phẩm tốt nhất.
    - Sản phẩm hệ thống cung cấp dịch vụ web của luận văn sẽ bảo đảm hai tính năng cơ bản nhất là nhận diện âm thanh tiếng Việt phổ thông từ tệp âm thanh đã được

thu âm sẵn, hoặc từ nguồn âm thanh đang được phát trực tiếp với độ chính xác trên 80%. Đối với tiếng Việt phương ngữ, hệ thống dịch vụ sẽ cố gắng hỗ trợ độ chính xác từ 70-80%. Đồng thời chi phí khi sử dụng hệ thống sẽ được giảm xuống mức tối thiểu.

* + - Xây dựng một ứng dụng di động trên nền tảng Android để sử dụng các chức năng phía hệ thống cung cấp dịch vụ web. Giao diện của ứng dụng di động sẽ được chọn lọc lại từ những ứng dụng mẫu trên thị trường và nâng cấp cho cần thiết với các cải tiến chức năng được thêm vào nhằm nâng cao trải nghiệm cho người dùng.

### PHẠM VI ĐỀ TÀI

Yêu cầu ban đầu của đề tài “Xây dựng hệ thống cung cấp dịch vụ nhận dạng âm thanh tiếng Việt”. Qua quá trình khảo sát, việc phát triển hệ thống một cách thuần túy là một nhiệm vụ khó khăn quá sức. Nên sản phẩm của khóa luận sẽ tập trung vào việc phát triển, hoàn thiện một cách đầy đủ dịch vụ nhận dạng giọng nói tiếng Việt dựa trên core engine tốt trên thị trường. Hệ thống cung cấp khả năng nhận dạng âm thanh tiếng Việt phổ thông và tiếng Việt phương ngữ từ tệp âm thanh được thu sẵn hoặc từ nguồn âm thanh đang phát trực tiếp. Sản phẩm luận văn được thực hiện một cách toàn diện nên sẽ áp dụng và mở rộng các thư viện có sẵn cũng như cài đặt thô để đáp ứng một số yêu cầu đặc thù nhằm đạt được các mục tiêu và yêu cầu chức năng đã đề ra.

# CHƯƠNG 2: LÝ THUYẾT NỀN TẢNG

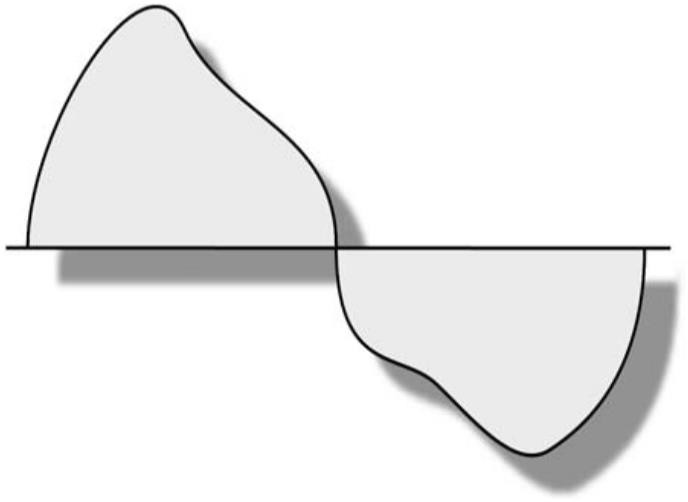
## LÝ THUYẾT NỀN TẢNG CỦA NHẬN DẠNG TIẾNG NÓI

### Định nghĩa

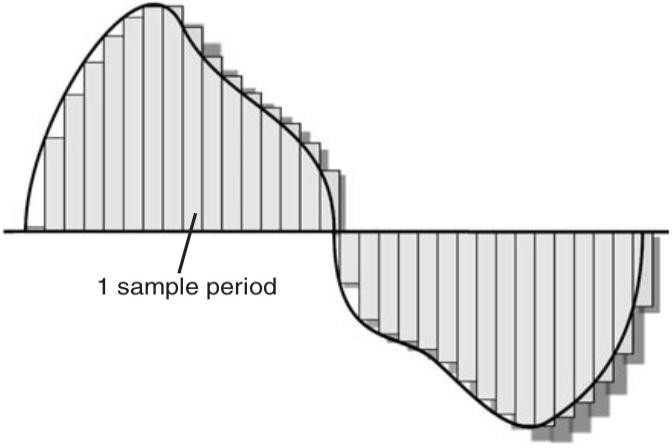
Phần này trình bày một số khái niệm cốt lõi về âm thanh kỹ thuật số (digital audio), các lý thuyết kỹ thuật trong mạng nơ-ron (neural network) nhận dạng giọng nói, để phục vụ cho việc thiết lập cấu trúc hệ thống nhận dạng tiếng nói cơ bản và hệ thống hiện đại phức tạp hơn được trình bày ở các phần sau.

* + - 1. Định nghĩa âm thanh số

Trong thế giới thực, âm thanh là các sóng cơ học, có dạng hình sin tuần hoàn liên tục (analog audio), được truyền, ghi, lưu trữ và tái tạo như là sự thay đổi của các mức điện áp theo thời gian như hình 2.1. Âm thanh số là những xung điện từ rời rạc, được chuyển đổi từ sóng âm thanh liên tục thành tín hiệu số bằng kỹ thuật lấy mẫu định kỳ (periodic samples) và biến đổi các mức tín hiệu được lấy mẫu này thành một luồng đại diện nhị phân có thể được xử lý, lưu trữ trên máy tính [12] như hình 2.2.



Hình 2.1 Tín hiệu âm thanh là liên tục trong tự nhiên (Nguồn: [12])



Hình 2.2 Tín hiệu âm thanh số sử dụng lấy mẫu định kỳ để mã hóa thông tin (Nguồn: [12])

Do đó, bằng những mẫu rời rạc, âm thanh số có thể mô phỏng gần giống với âm thanh thực tế, việc mô phỏng được đặc trưng bởi các thông số sau.

* + - * + Mẫu (sample)

Là thành phần nhỏ nhất của một bản âm thanh số, là giá trị biên độ của tần số sóng âm tại thời điểm lấy mẫu. Để có các xung điện tử rời rạc, cần phải tiến hành rất nhiều lần lấy mẫu. Càng lấy nhiều mẫu, tín hiệu số thu được càng chính xác.

* + - * + Tỉ lệ lấy mẫu (sample rate)

Là số lượng phép lấy mẫu trên tín hiệu tuần hoàn liên tục trong một giây, đơn vị Hz. Ví dụ một bản nhạc có tỉ lệ lấy mẫu là 16000 Hz thì mỗi giây nhạc sẽ được lấy mẫu 16000 lần. Bởi vì việc lấy mẫu được gắn trực tiếp với thành phần thời gian, tỉ lệ lấy mẫu của một hệ thống xác định băng thông tổng thể của nó, vì vậy một hệ thống có tỉ lệ lấy mẫu cao càng có khả năng lưu trữ nhiều tần số ở giới hạn trên.

* + - * + Độ nét (bit depth)

Để lưu lại dưới dạng số, mỗi mẫu được biểu diễn bằng một lượng bit dữ liệu nhất định, nó liên quan trực tiếp tới số bước lượng tử hóa được mã hóa vào dòng bit (bit stream). Vì

vậy, độ nét ảnh hưởng trực tiếp tới độ sắc nét, trung thực của âm thanh. Nếu số lượng bit dùng để mã hóa mẫu quá thấp thì độ phân giải không đạt tiêu chuẩn, thậm chí là biến dạng âm thanh. Mặt khác, nếu độ nét quá cao làm cho tệp tin lưu trữ âm thanh số sẽ rất lớn.

* + - * + Kênh (channel)

Tín hiệu số sẽ được tách ra thành nhiều kênh sao cho khi nghe bằng hệ thống loa thích hợp sẽ có cảm giác như khi đang nghe nhạc trong không gian thực tế.

Từ bốn thông số cơ bản trên, ta có thể biết được chất lượng và dung lượng của một bản âm thanh số. Ví dụ một phút bản âm thanh số có: tỉ lệ lấy mẫu bằng 16000Hz, độ nét bằng 16 bits, kênh bằng 1, sẽ có dung lượng 16000 × 16 × 1 × 60 = 14,65 Kb. Trong phần huấn luyện mô hình nhận dạng âm thanh tiếng Việt, nhóm sinh viên lựa chọn âm thanh số có các thông số như ví dụ trên làm dữ liệu đầu vào.

* + - 1. Định nghĩa nhận dạng tiếng nói

Nhận dạng tiếng nói là một quá trình nhận dạng mẫu, với mục đích phân lớp thông tin đầu vào là tín hiệu tiếng nói thành một dãy tuần tự các mẫu đã được học trước đó và lưu trữ trong bộ nhớ.

### Lý thuyết nền tảng mạng nơ-ron (Neural Networks)

Mạng nơ-ron là một tập hợp các mô hình thống kê được đặc trưng bởi một tập các nút kết nối với các hàm kích hoạt phi tuyến tính cùng các tham số có khả năng học. Mạng nơ-ron hiện là mô hình thống kê phổ biến nhất được sử dụng cho các ứng dụng máy học trong một loạt các lĩnh vực bao gồm: thị giác máy tính, nhận dạng giọng nói, xạ hình y tế, ...

* + - 1. Giải phẫu tế bào mạng nơ-ron

Một tế nào (nút) của mạng nơ-ron là một hàm của tập các trọng số tương ứng với các giá trị đầu vào (inputs) {𝑥0, … , 𝑥𝑁}.

𝑁

𝑦 = 𝑎 (∑ 𝑤𝑖𝑥𝑖 + 𝑏)

𝑖

Trong đó:

* 𝑤𝑖 : trọng số của đầu vào 𝑥𝑖
* 𝑎: hàm kích hoạt (activation function)
* 𝑏: độ sai lệch (bias)

Ta sẽ sử dụng kí hiệu ma trận để làm đơn giản cách thể hiện, trong đó mỗi tế bào nơ-ron bao gồm một vector đầu vào 𝑥 = {𝑥0, … , 𝑥𝑁}, một vector trọng số 𝑤 = {𝑤0, … , 𝑤𝑁} và một vector sai lệch 𝑏, khi đó đầu ra là:

𝑦 = 𝑎(𝑤𝑇𝑥 + 𝑏)

Nếu hàm kích hoạt a là một biến thể của hàm Heaviside,

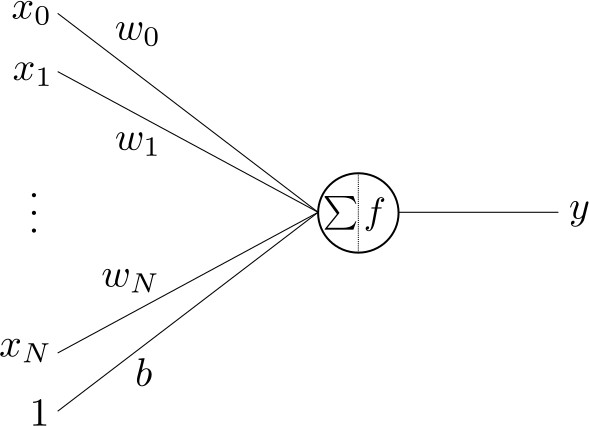
𝑎(𝑥) = { 1

𝑛ế𝑢 𝑥 ≥ 0

0 ℎ𝑜ặ𝑐 − 1 𝑛ế𝑢 𝑥 < 0

thì tế bào nơ-ron này được gọi là một perceptron, một bộ phân loại nhị phân đơn giản, là một trong những phương pháp học kết nối sớm nhất được phát minh bởi Rosenblatt. Hình

2.3 minh họa kiến trúc điển hình của một tế bào mạng nơ-ron.

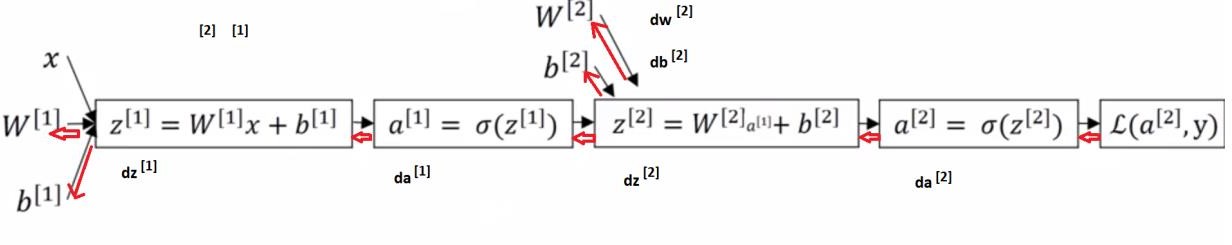


Hình 2.3: Minh họa kiến trúc điển hình của một tế bào mạng nơ-ron. (Nguồn: [13])

* + - 1. Lan truyền ngược (Backpropagation)

Các thuật toán học sâu tương phản với các thuật toán học nông bởi số biến đổi được tham số hóa một tín hiệu gặp phải khi nó lan truyền từ các lớp đầu vào đến các lớp đầu ra. Mỗi chuỗi các biến đổi từ đầu vào đến đầu ra gọi là một đường gán kế thừa (CAP - Credit Assignment Path). Vấn đề gán kế thừa (Credit Assignment Problem) được giải quyết với khám phá lan truyền ngược (backpropagation), cho phép học với mạng nơ-ron nhiều lớp. Sau đây, nhóm sinh viên sẽ trình bày ý tưởng của quá trình lan truyền ngược.

Hình 2.4 minh họa một mạng nơ-ron 2 lớp, gồm một lớp đầu vào, một lớp ẩn và một lớp đầu ra.



Hình 2.4: Minh họa một mạng nơ-ron 2 lớp (Nguồn: Coursera Sequence Model)

Lưu ý ở lan truyền tiến, các bước tính toán là như sau: đầu tiên ta tính toán 𝑧[1], sau đó tính toán 𝑎[1], rồi tính toán 𝑧[2] ghi chú là 𝑧[2] cũng phụ thuộc vào các tham số 𝑊[2] và

𝑏[2], sau đó dựa vào 𝑧[2] tính toán 𝑎[2] và cuối cùng là tính toán chi phí. Đối với lan truyền ngược, ta sẽ đi tính toán theo chiều ngược lại, cụ thể là tính toán 𝑑𝑎[2], sau đó tính

𝑑𝑧[2], quay ngược lên tính 𝑑𝑤[2] và 𝑑𝑏[2], tương tự ta tính toán tiếp cho các biến 𝑑𝑎[1],

𝑑𝑧[1], 𝑑𝑊[1], 𝑑𝑏[1]. Thông thường, ta sẽ bỏ qua tính đạo hàm của 𝑑𝑎[2], thay vào đó nhập lại thành một bước là tính trên 𝑑𝑧[2]. Sau cùng ta rút ra các công thức sau:

𝑑𝑎[2] = 𝑎[2] − 𝑦

𝑑𝑊[2] = 𝑑𝑧[2]𝑎[1]𝑇

𝑑𝑏[2] = 𝑑𝑧[2]

𝑑𝑧[1] = 𝑊[2]𝑇𝑑𝑧[2] ∗ 𝑔[1]′(𝑧[1])

𝑑𝑊[1] = 𝑑𝑧[1]𝑥𝑇

𝑑𝑏[1] = 𝑑𝑧[1]

* + - 1. Học với lan truyền ngược

Học với lan truyền ngược giống như quy tắc delta, các độ nhạy được dùng để điều chỉnh trọng số tỷ lệ với một hằng số tỉ lệ học (learning rate) 𝛼. Việc cập nhật trọng số thực hiện theo công thức sau:

∆𝑤𝑛 = −𝛼

𝑖𝑗

𝜕𝐸𝑛

𝜕𝑤𝑖𝑗

= −𝛼𝛿𝑛𝑦𝑖

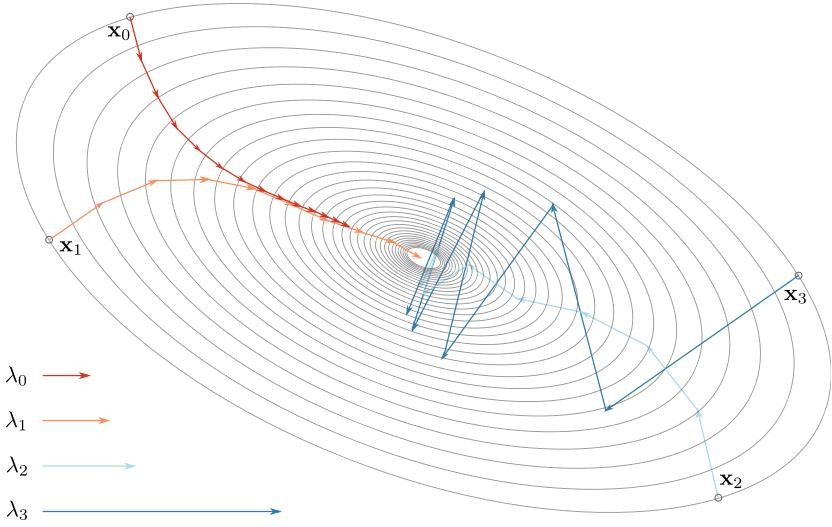
𝑗

Trong đó:

* 𝛿𝑛: đã được định nghĩa ở công thức

𝑗

* 𝑦𝑖 : đầu ra của nơ-ron i



Hình 2.5 Minh họa sự ảnh hưởng của tỉ lệ học và chính sách học lên độ hội tụ với lan truyền ngược. (Nguồn: [13])

Lan truyền ngược là một phương pháp gốc dốc nhất (steepest descent). Hình 2.5 minh họa quy tắc học với lan truyền ngược, đặc trưng cho kích thước mỗi bước là tham số tỉ lệ học. Tham số tỉ lệ học thay đổi kích thước bước hay độ lớn của vector thay đổi trọng số. Hình 2.5 cũng minh họa độ ảnh hưởng của tỉ lệ học trên độ giảm dốc. Tỉ lệ học quá nhỏ làm cho kết quả học rất chậm như 𝛼0, tuy nhiên tỉ lệ học quá lớn lại làm cho bước nhảy

xung quanh khu vực điểm cực tiểu và mất thời gian dài để tiếp cận điểm cực tiểu này như

𝛼2 và 𝛼3.

Để đạt được điểm cực tiểu cục bộ, tỉ lệ học cũng phải được giảm trong quá trình huấn luyện. Tuy nhiên, nếu giảm quá nhanh, nó có thể không bao giờ đạt được lưu vực gần với điểm cực tiểu, như với 𝛼0, ngược lại nếu giảm quá chậm, nó có thể mất một thời gian dài

để tiến vào khu vực này, như với 𝛼3.

Cân bằng việc cố gắng tìm một tỉ lệ học và chính sách học phù hợp không may là một phần "ma thuật đen" đằng sau việc huấn luyện DNN đến từ kinh nghiệm. Tuy nhiên Bottou (2012) và I. Goodfellow, Y.Bengio, và Courville (2016) là tài liệu tham khảo tuyệt vời về một số các tiếp cận phổ biến để làm công việc này dễ dàng hơn.

* + - 1. Hàm kích hoạt (Activation function)

Hàm kích hoạt là phần rất quan trọng trong mạng nơ-ron, đặc biệt là mạng nơ-ron nhiều lớp ẩn. Nếu không có hàm kích hoạt phi tuyến tính, cho dù mạng nơ-ron có nhiều lớp ẩn đến cỡ nào thì cũng chỉ có sức mạnh đại diện cho phân loại tuyến tính, điều này tương đương với một mạng mà không có lớp ẩn nào. Vì bản chất tổng hợp các hàm tuyến tính là một hàm tuyến tính. Do đó, hàm kích hoạt 𝑎 là một hàm phi tuyến tính được áp dụng cho đầu ra tại mỗi nút, cho phép mạng nơ-ron nhiều lớp ẩn học các hàm phi tuyến phức tạp.

Trong lĩnh vực mạng nơ-ron, hàm kích hoạt được chọn một cách cổ điển là hàm sigmoid, ánh xạ các giá trị đầu vào vào khoảng biên từ 0 đến 1, phù hợp cho các mạng phân loại nhị phân. Có công thức theo phương trình:

𝑧[𝑖] = 𝑊[𝑖]𝑥 + 𝑏[𝑖]

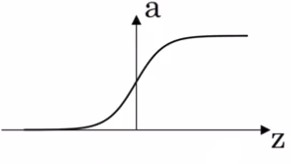
𝑎 =

1

1 + 𝑒−𝑧

Trong đó:

* 𝑊[𝑖]: trọng số tại lớp thứ i
* 𝑏[𝑖]: tham số sai lệch tại lớp thứ i
* 𝑎: hàm kích hoạt



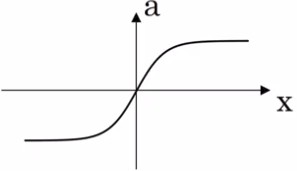
Hình 2.6 Minh họa hàm kích hoạt sigmoid. (Nguồn: Coursera Sequence Models)

Ngoài ra, có một hàm kích hoạt luôn hoạt động tốt hơn hàm sigmoid là hàm tiếp tuyến hyperbolic (hyperbolic tangent function), ánh xạ các giá trị đầu vào vào khoảng biên từ -1 đến 1. Có công thức là:

𝑎 =𝑡𝑎𝑛ℎ (𝑧) =

𝑒𝑧 − 𝑒−𝑧

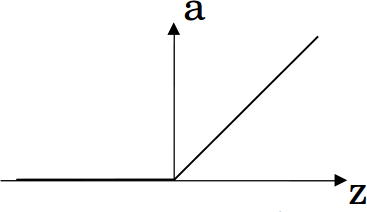
𝑒𝑧 + 𝑒−𝑧



Hình 2.7: Minh họa hàm kích hoạt tanh. (Nguồn: Coursera Sequence Models)

Tuy nhiên, một vấn đề với hàm kích hoạt sigmoid và tanh nếu z quá lớn hoặc quá nhỏ thì độ dốc của hàm sẽ rất nhỏ, điều này làm chậm quá trình tìm điểm cực tiểu của hàm chi phí, dẫn đến làm chậm quá trình học. Vì lý do này, dựa vào các kết quả thực nghiệm được cải thiện, mạng nơ-ron hiện đại có xu hướng sử dụng hàm kích hoạt đơn vị tuyến tính chỉnh lưu (ReLu - Rectified Linear Unit). Có công thức như sau:

𝑎 = 𝑚𝑎𝑥 (0, 𝑧)



Hình 2.8 Minh họa hàm kích hoạt ReLu. (Nguồn: Coursera Sequence Models)

Vì vậy, đạo hàm luôn bằng 1 nếu z dương, và bằng 0 nếu z âm. Dựa trên thực nghiệm, sử dụng hàm kích hoạt ReLu, mạng nơ-ron sẽ học nhanh hơn so với khi dùng với hàm sigmoid hoặc hàm tanh. Lý do chính là có ít hơn sự ảnh hưởng của độ dốc hàm bằng 0 làm chậm việc học. Vì mặc dù, có một nửa phạm vi của z làm độ dốc hàm ReLu bằng 0, nhưng trong thực tế, đủ các đơn vị ẩn thì ta sẽ có z lớn hơn 0, vì vậy việc học vẫn khá nhanh với hầu hết các ví dụ đào tạo.

* + - 1. Độ giảm dốc theo lô nhỏ (Mini-batch Gradient Descent)

Quá trình áp dụng máy học là một quá trình thực nghiệm và có sự lặp lại cao, nghĩa là ta phải huấn luyện rất nhiều mô hình để tìm ra mô hình tốt nhất cho vấn đề giải quyết. Một điều khó khăn hơn là việc học sâu không hoạt động tốt cho chế độ dữ liệu lớn, việc huấn luyện để có mô hình trên tập dữ liệu lớn là rất chậm và tốn kém. Một trong những thuật toán tối ưu hóa tốc độ huấn luyện mô hình là giảm độ dốc theo lô nhỏ (mini-batch gradient descent).

* Giảm độ dốc theo lô nhỏ

Thay vì tính toán độ dốc trên toàn bộ tập huấn luyện, ta có thể thay thế bằng một tập con đặc trưng của tập huấn luyện - một lô nhỏ (mini-batch). Lấy mẫu ngẫu nhiên (không thay thế) một tập con của tập huấn luyện 𝑋𝑚𝑏 ∈ 𝑋, sao cho

1

∆𝑤𝑖𝑗 = −𝛼

|𝑋

𝑚𝑏

| ∑

{𝑛|𝑥𝑛 𝜖𝑋𝑚𝑏}

𝜕𝐸𝑛

𝜕𝑤𝑖𝑗

Trong đó kích thước lô nhỏ 𝑋𝑚𝑏 phải đủ ý nghĩa để thể hiện số liệu thống kê của phân phối tập huấn luyện. Với kích thước lô nhỏ bằng 1, ta có thuật toán Stochastic Gradient

Descent, nhưng thường thuật toán này khá nhiễu và không đại diện được cho tập dữ liệu chung.

* Giảm độ dốc theo lô (Batch Gradient Descent)

Thuật toán độ giảm dốc được thực hiện trên toàn bộ tập huấn luyện cùng một thời điểm,

𝑁

1 𝜕𝐸𝑛

∆𝑤𝑖𝑗 = −𝛼 𝑁 ∑ 𝜕𝑤

𝑛=0

Trong đó:

* N: số lượng mẫu huấn luyện trong X.
* 𝛼: tỉ lệ học (learning rate)
* ∆𝑤𝑖𝑗: giá trị delta cho mỗi làn cập nhật tham số

𝑖𝑗

Huấn luyện với kích thước là toàn bộ bộ huấn luyện là rất tốn kém và chậm vì nó yêu cầu thực hiện chuyển tiếp qua tất cả các mẫu đào tạo cho mỗi lần cập nhật tham số.

* + - 1. Phương pháp độ giảm dốc với động lượng (Momentum)

Lý do cơ bản của tỉ lệ học (learning rate) và chính sách học có ảnh hưởng lớn là do giảm độ dốc là một phương pháp tối ưu hóa bậc nhất và chỉ xem xét các đạo hàm riêng bậc một.

∇𝐸(𝑥, 𝑦) = (

𝜕𝐸

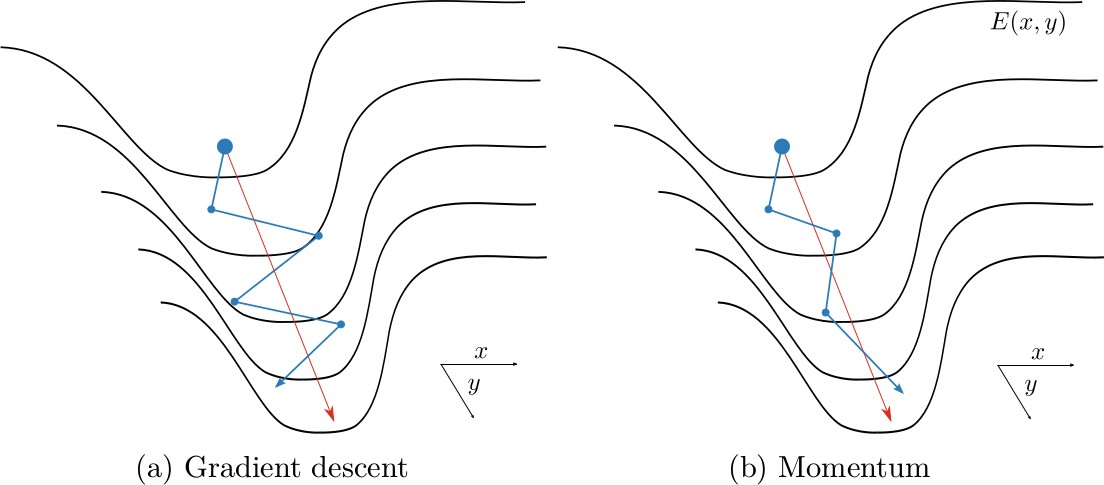
𝜕𝑥

𝜕𝐸

, )

𝜕𝑦

Độ dốc cho biết hướng tăng tối đa tại một điểm cho trước trên bề mặt hàm lỗi, nhưng nó không cho bất kỳ thông tin nào về độ cong của bề mặt tại điểm đó. Độ cong của bề mặt được mô tả bởi các đạo hàm bậc hai như đạo hàm riêng bậc hai. Các đạo hàm này cung cấp thông tin quan trọng về độ cong của bề mặt hàm lỗi 𝐸. Ví dụ trong hình 2.5 bề mặt hàm lỗi có hình ellipse, gây ra vấn đề khi chúng ta chỉ xem xét hướng giảm tối đa. Ví dụ điển hình về bề mặt hàm lỗi bệnh lý cho các hàm bậc nhất là một bề mặt trông giống như một thung lũng hẹp, được thể hiện ở hình 2.9a. Với một khởi tạo bên ngoài đáy thung lũng, độ dốc sẽ nảy dọc theo các bức tường của thung lũng dẫn đến sự hội tụ học rất chậm. Hình 2.9 minh họa vấn đề độ cong bệnh lý của bề mặt hàm lỗi 𝐸(𝑥, 𝑦), thể hiện một thung lũng hẹp và đường tối ưu từ điểm xuất phát đến điểm cực tiểu được vẽ bằng mũi tên màu đỏ. Trong bề mặt hàm lỗi như vậy, các hàm bậc nhất không thể dùng thông tin được cung cấp bởi Hessian trên bề mặt cong để tránh nảy dọc theo các bức tường của thung lũng và làm chậm độ giảm dốc.



Hình 2.9 Minh họa vấn đề bề mặt độ cong bệnh lý và các tiếp cân của độ giảm dốc và độ giảm dốc với động lượng. (Nguồn: [13])

Đối với các bề mặt hoạt động tốt, trong đó tỉ lệ các tham số là tương tự nhau, các lưu vực thu hút xung quanh cực tiểu có dạng tròn và do đó tránh được vấn đề này, vì các độ dốc bậc một sẽ chỉ gần như trực tiếp tại cực tiểu cho bất kỳ vị trí nào trên bề mặt hàm lỗi.

Độ dốc giảm dần với động lượng (Gradient descent with momentum) là một thuật toán cải tiến và học nhanh hơn so với thuật toán độ giảm dốc tiêu chuẩn, đây là một cách để giảm thiểu ảnh hưởng độ cong bệnh lý tới độ giảm dốc, điều này cũng giúp làm thay đổi độ dốc.

Trong động lượng, độ dốc trên nhiều lần lặp được tính lũy thành một vận tốc độ giảm dốc (velocity gradient),

𝑣𝑑𝑊 = 𝛽𝑣𝑑𝑊 + (1 − 𝛽)𝑑𝑊

𝑣𝑑𝑏 = 𝛽𝑣𝑑𝑏 + (1 − 𝛽)𝑑𝑏

𝑊 = 𝑊 − 𝛼𝑣𝑑𝑊

𝑏 = 𝑏 − 𝛼𝑣𝑑𝑊

Trong đó:

* 𝛽: có giá trị phổ biến = 0.9

Động lượng có khả năng lưu trữ một số thông tin về độ dốc của các lần lặp trước và sử dụng thông tin này để làm giảm hiệu ứng của độ dốc mới trên hướng tìm kiếm như được minh họa ở hình. Đối với các bề mặt lỗi có độ cong bệnh lý, điều này có thể tăng tốc đáng kể việc học.

### Các phương pháp huấn luyện mạng nơ-ron đương đại

Ở đây, nhóm sinh viên sẽ phác thảo những khác biệt có liên quan nhất trong việc huấn luyện các mô hình mạng nơ-ron học sâu (DNN - Deep Neural Network) đương đại so với

trước khi làm việc của Krizhevsky, Sutskever và Geoffrey E. Hinton (2012), cách đây chưa tới mười năm.

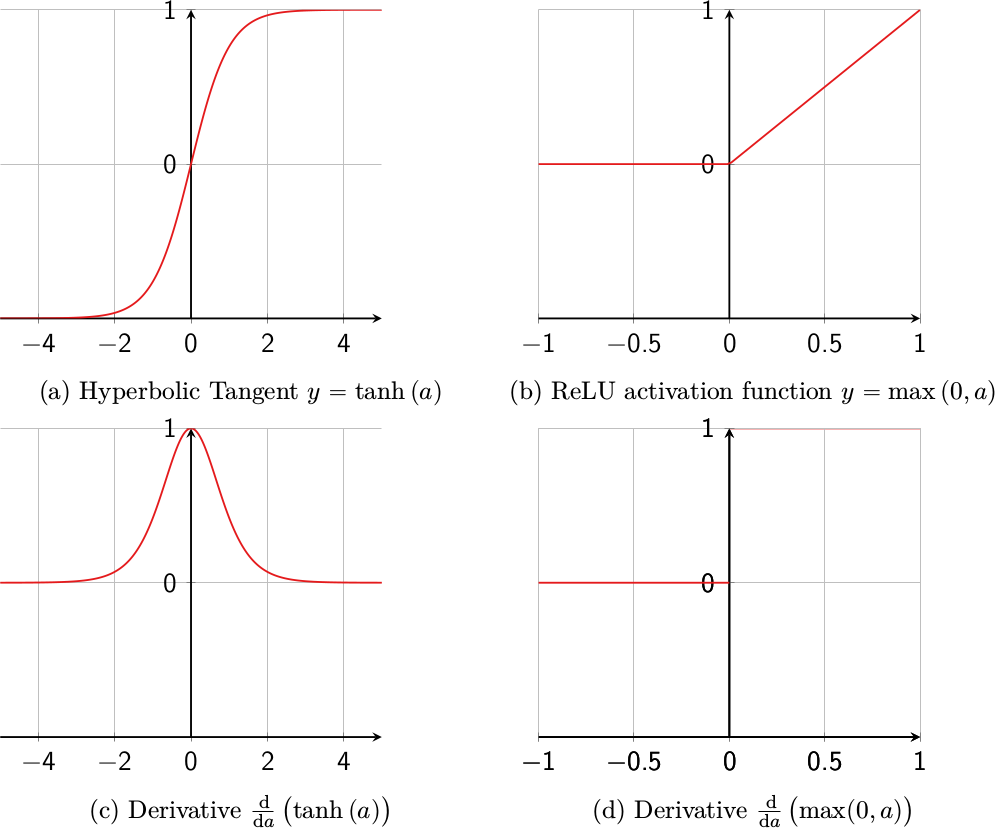
* + - 1. Hàm kích hoạt đơn vị tuyến tính chỉnh lưu (ReLu - Rectified Linear Activation Function)

Một phần không thể thiếu của bất kỳ mạng nơ-ron nào là hàm kích hoạt phi tuyến tính. Trong lịch sử, mạng nơ-ron đã từng sử dụng chức năng kích hoạt dạng sigmoid. Tuy nhiên một vấn đề lớn với các hàm kích hoạt dạng sigmoid là độ dốc bên ngoài một vùng tương đối hẹp trên miền hàm là một số rất nhỏ. Khi huấn luyện với lan truyền ngược (backpropagation) điều này có nghĩa là hầu hết độ dốc có độ lớn rất nhỏ và việc huấn luyện có thể sẽ tốn một thời gian rất dài, hoặc thậm chí bị đình trệ hoàn toàn - tình huống này được gọi là độ dốc biến mất (vanishing gradient).

ReLu được đề xuất như một giải pháp, đầu tiên cho các máy Boltzmann bị hạn chế (Nair và Geoffrey E. Hinton, 2010), sau đó là cho các mạng nơ-ron ((Glorot và Y. Bengio, 2010), trong đó về mặt thực nghiệm, nó được chứng minh là cho phép đào tạo dễ dàng hơn với lan truyền ngược.

𝑧[𝑖] = 𝑊[𝑖]𝑥 + 𝑏[𝑖]

𝑎 = 𝑚𝑎𝑥 (0, 𝑧)



Hình 2.10: Minh họa các hàm kích hoạt thường dùng trong mạng nơ-ron và đạo hàm tương ứng của nó. (Nguồn: [13])

ReLu không thể hiện độ bảo hòa như các hàm dạng sigmoid, nó luôn cho độ dốc có giá trị 0 hoặc 1, được minh họa ở hình 2.10. Trong thực tế, điều này có thể làm tăng tốc độ đào tạo thậm chí cho phép các mạng mà không thể đào tạo thực tế với hàm kích hoạt dạng sigmoid chẳng hạn như mạng học sâu Krizhevsky, Sutskever và Geoffrey E. Hinton (2012).

* + - 1. Phương pháp chuẩn hóa theo lô (Batch Normalization)

Một số kiến trúc mạng đủ phức tạp, tức là các mạng có nơ-ron với số lượng kết nối đi/đến khác nhau đáng kể thì thậm chí việc khởi tạo cẩn thận cũng không ngăn được việc bùng

nổ hay biến mất độ dốc. Thay vào đó, Ioffe and Szegedy (2015) đã đề xuất một cách tiếp cận mới là chuẩn hóa theo lô (Batch normalization).

Chuẩn hóa theo lô tính toán giá trị trung bình, phương sai của lô nhỏ (mini-batch) và ngăn chặn biến mất độ dốc bằng chuẩn hóa các phản hồi/độ dốc theo thống kê lô (batch). Sử dụng phương pháp này trong quá trình đào tạo có thể tạo ra khác biệt lớn, trong nhiều trường hợp các mạng mà trước đây không hội tụ, sẽ hội tụ và thậm chí có thể tăng tốc độ đào tạo.

Thống kê lô được tính theo chuẩn hóa lô là:

𝑀

1

𝜇𝑏 =

𝑀 ∑ 𝑥𝑖

𝑖=0

𝑀

𝜎2 = 1 ∑

𝑏 𝑀

𝑖=0

𝑥̂ = 𝑥𝑖 − 𝜇𝑏

𝑖

√𝜎2 + 𝜀

𝑏

Trong đó:

* 𝑏: lô nhỏ đang xét
* 𝜇𝑏: giá trị trung bình
* 𝜎2: phương sai

𝑏

* 𝜀: số ổn định (epsilon)
* 𝑥𝑖: phản hồi của lô nhỏ
* 𝑥̂𝑖: phản hồi đã chuẩn hóa

Chuẩn hóa theo lô sau đó sử dụng các thống kê này cùng với hai tham số đã học trên các lô nhỏ là 𝛾và 𝛽để chia tỉ lệ và thay đổi các phản hồi.

𝑦𝑖 = 𝛾𝑥̂𝑖+ 𝛽

* + - 1. Phương pháp cắt giảm (Dropout)

Geoffrey E. Hinton, Srivastava, et al. (2012a) và Srivastava et al. (2014) đã giới thiệu phương pháp cắt giảm (dropout), một phương pháp ngăn chặn vượt mức (overfit) trong các mạng lớn khi đào tạo. Ý tưởng chính là: Trong quá trình huấn luyện ta loại bỏ một tập các nút nơ-ron, được lấy mẫu ngẫu nhiên từ mỗi lớp với xác suất cố định p.

Cơ chế ảnh hưởng của phương pháp cắt giảm được giải thích theo nhiều cách khác nhau, nhưng đáng chú ý nhất là các cách giải thích được đưa ra bởi Geoffrey E. Hinton, Srivastava, et al. (2012a) và Srivastava et al. (2014).

Giải thích của Geoffrey E. Hinton, Srivastava, et al. (2012a) là phương pháp cắt giảm là một hình thức chính quy hóa bằng độ ồn, ngăn chặn sự thích nghi của các tế bào nơ-ron. Giải thích chính được đưa ra bởi Srivastava et al. (2014) rằng phương pháp cắt giảm là một hình thức tích hợp mô hình, tính trung bình trên một số lượng lớn các kiến trúc mô hình "mỏng hơn" ngẫu nhiên tại thời điểm đào tạo để cải thiện khái quát hóa. Tuy nhiên, việc tính trung bình trên tất cả các mô hình được xem xét trong suốt quá trình đào tạo là cực kỳ tốn kém, bởi vì số lượng mô hình có thể có tăng theo cấp số mũ.

### Các kiến trúc mạng nơ-ron hồi quy

Một danh sách nghiên cứu đầy đủ của mọi kiến trúc học sâu DNN là điều không khả thi và nằm ngoài phạm vi của luận án này, tuy nhiên ở đây, nhóm sinh viên đã nỗ lực khái quát về những kiến trúc nơ-ron hồi quy nổi bật trong những năm gần đây đồng thời truyền cảm hứng cho việc thiết lập kiến trúc hệ thống nhận diện giọng nói trong các chương sau của nhóm.

* + - 1. Mạng nơ-ron hồi quy (RNN - Recurrent Neural Network)

Mạng nơ-ron hồi quy (RNN - Recurrent Neural Network) là mạng thần kinh lan truyền tới (Feedforward Neural Networks) [11] được tăng cường bằng cách bao gồm các cạnh mà kéo dài các bước thời gian liền kề, đưa ra khái niệm về thời gian cho mô hình. Giống như mạng lan truyền tới, mạng RNN không có chu trình giữa các cạnh thông thường. Tuy nhiên, các cạnh mà kết nối các bước thời gian liền kề, được gọi là cạnh hồi quy, có thể hình thành chu trình có độ dài bằng một, tự kết nối từ một nút (node) tới chính nó theo thời gian. Tại thời điểm 𝑡, nút với cạnh hồi quy nhận đầu vào từ điểm dữ liệu hiện tại 𝑥(𝑡) và từ các giá trị nút ẩn ℎ(𝑡−1) trong trạng thái trước đó của mạng. Đầu ra 𝑦̂(𝑡) cho mỗi thời điểm 𝑡 được tính bằng các giá trị nút ẩn ℎ(𝑡) tại thời điểm 𝑡. Đầu vào 𝑥(𝑡−1) tại thời điểm 𝑡 - 1 có thể ảnh hưởng đến đầu ra 𝑦̂(𝑡) tại thời điểm 𝑡 và sau đó, bằng các kết nối hồi quy.

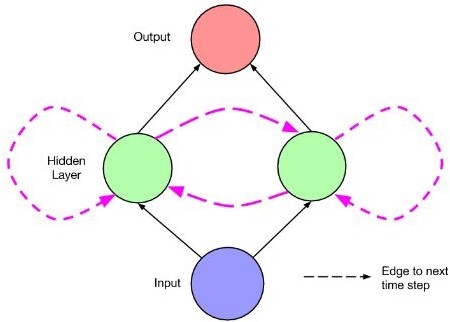
Các tính toán cần thiết tại mỗi bước thời gian trên đường là truyền tới của mạng RNN đơn giản (hình 2.11), được thể hiện như sau:

ℎ(𝑡) = 𝜎(𝑊ℎ𝑥 + 𝑊ℎℎ ℎ(𝑡−1) + 𝑏ℎ)

𝑦̂(𝑡) = 𝑠𝑜𝑓𝑡𝑚𝑎𝑥(𝑊𝑦ℎ ℎ(𝑡) + 𝑏𝑦)

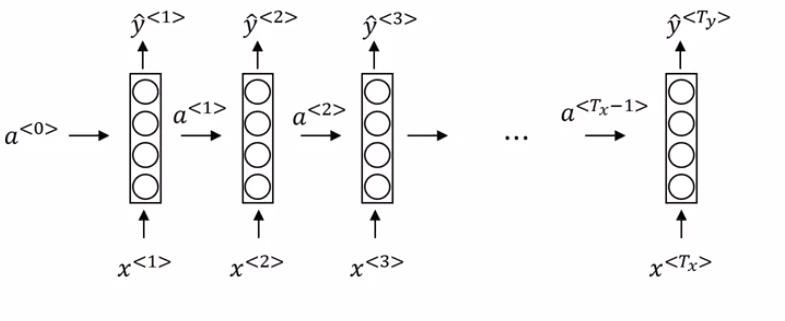
Trong đó:

* 𝑊ℎ𝑥 là ma trận trọng số thông thường giữa đầu vào và lớp ẩn
* 𝑊ℎℎ là ma trận trọng số hồi quy giữa lớp ẩn và chính nó ở các bước thời gian liền kề.
* Các vector 𝑏ℎvà 𝑏𝑦 là các tham số sai lệch.



Hình 2.11 Mạng RNN đơn giản (Nguồn: [11])

Một biểu diễn dễ hiểu cho hình 2.11 được thể hiện ở hình 2.12, trong đó các bước thời gian được mở ra. Với hình ảnh này, mạng được hiểu không phải là chu trình, mà là một mạng học sâu với mỗi lớp tương ứng với mỗi bước thời gian, được chia sẻ trọng số qua các lớp.



Hình 2.12 Minh họa mạng RNN được mở ra từ hình 2.11 (Nguồn: Coursera Sequence Models)

Bây giờ lấy ví dụ, xét bài toán: Xác định chữ nào là một phần của tên người trong một câu.

Đọc một câu từ trái sang phải, đánh số mỗi từ trong câu là 𝑥𝑖(1 ≤ 𝑖 ≤ 𝑁). Tiến hành nạp từ thứ nhất 𝑥1vào một lớp ẩn mạng nơ-ron, lớp này sẽ dự đoán kết quả 𝑦̂<1> xem 𝑥1 có phải là một phần của tên người không. Xét từ thứ 2 (𝑥2) thay vì chỉ dự đoán 𝑦̂<2> từ 𝑥2, thì lớp này nhận thêm giá trị kích hoạt 𝑎1 từ bước 1. Các bước sau được thực hiện tương

tự, cho đến khi kết thúc câu. Thông thường ở bước đầu tiên cũng được truyền thêm vào giá trị kích hoạt với 𝑎<0> (là một vector không). Hình 2.12 minh họa mạng RNN lan truyền tới với giá trị 𝑎<𝑡>, 𝑦̂<𝑡> được tính theo công thức sau:

𝑎<𝑡> = 𝑔(𝑊𝑎𝑎 𝑎<𝑡−1> + 𝑊𝑎𝑥 𝑥<𝑡> + 𝑏𝑎)

𝑦̂<𝑡> = 𝑔(𝑊𝑦𝑎𝑎<𝑡> + 𝑏𝑦)

RNN duyệt dữ liệu từ trái qua phải và tham số dùng cho mỗi bước là được chia sẻ. Vì vậy, khi dự đoán kết quả 𝑦̂<3>, RNN không chỉ sử dụng đầu vào là 𝑥3, mà còn sử dụng thông tin từ 𝑥1và 𝑥2.

Xét hai trường hợp đầu vào sau:

Trường hợp một:

1. Anh ấy nói *"Teddy Roosevelt là một tổng thống tuyệt vời".*
2. Anh ấy nói *"Teddy là loại gấu bông được mua nhiều nhất ở cửa hàng này"*.

Trường hợp hai:

*Thời thơ ấu, tôi thường nghe bố tôi nhắc tới một người anh hùng, ... bố tôi đang nhắc tới Phan Đình Giót.*

Ở trường hợp một, *"Teddy"* là một phần tên người trong câu (1), còn câu (2) thì không phải. Như vậy, một điểm yếu của mạng RNN là chỉ dùng thông tin từ các bước phía trước

trong chuỗi để thực hiện dự đoán kết quả, thông tin dùng để dự đoán này là không đủ. Để khắc phục nhược điểm này, mạng nơ-ron hồi quy hai chiều (BiRNN - Bidirectional recurrent neural network) ra đời.

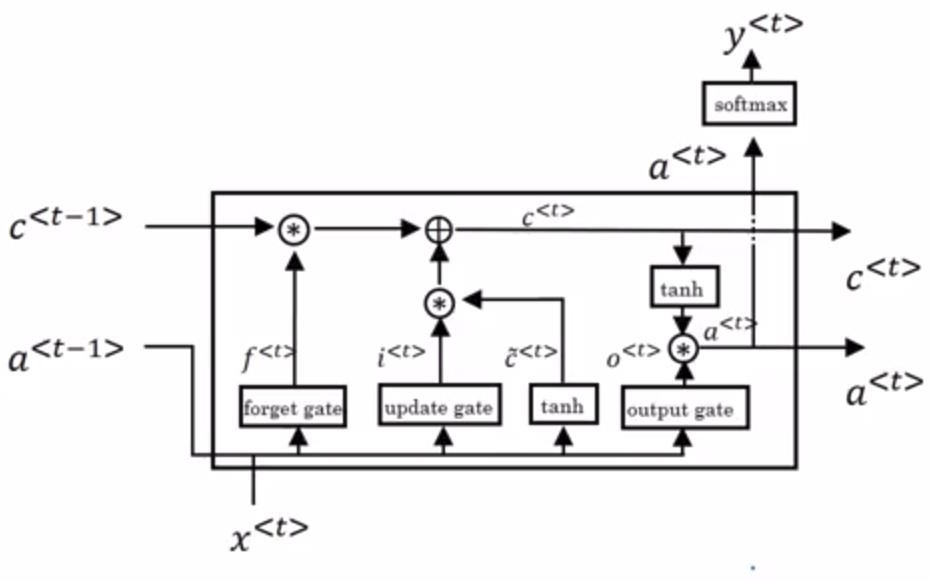
Còn ở trường hợp hai, cụm từ *"Phan Đình Giót"* có được dự đoán là tên của người hay không phụ thuộc thông tin được mang đến từ cụm *"người anh hùng"*. Nhưng mạng RNN bị hạn chế trong duy trì phụ thuộc tầm xa, nghĩa là nếu chuỗi đầu vào đủ dài, mạng đủ sâu, thì RNN khó khăn trong việc mang thông tin từ các bước trước tới các bước sau, thậm chí RNN có thể bỏ qua thông tin quan trọng đến từ những bước đầu tiên. Đây là vấn đề biến mất độ dốc (vanishing gradient).

* + - 1. Mạng bộ nhớ dài-ngắn (LSTM - Long Short Term Memory)

Như đã trình bày ở mục 2.1.4.1, mạng RNN bị vấn đề độ dốc biến mất (vanishing gradient). Bản chất mô hình RNN có nhiều ảnh hưởng cục bộ, nghĩa là đầu ra 𝑦̂<3> sẽ bị ảnh hưởng bởi những giá trị gần 𝑦̂<3> như 𝑥<1> 𝑥<2> 𝑥<3>, đầu ra vị trí cuối chuỗi

𝑦̂<𝑇> sẽ bị ảnh hưởng bởi những giá trị gần vị trí T. Điều này gây nên khó khăn nếu muốn đầu ra 𝑦̂<𝑇> bị ảnh hưởng mạnh mẽ bởi những giá trị ở đầu chuỗi (hình 2.12).

Mạng bộ nhớ dài-ngắn (Long Short-Term Memory - LSTM) được tạo ra để giải quyết vấn đề duy trì phụ thuộc tầm xa. Mạng LSTM giống với mạng RNN tiêu chuẩn, nhưng với mỗi nút gốc trong lớp ẩn sẽ được thay thế bằng một ô nhớ (memory cell) và sử dụng thêm ba cổng riêng biệt là cổng quên (forget gate), cổng vào (input gate), cổng ra (output gate) để điều chỉnh luồng thông tin trong một ô LSTM.



Hình 2.13 Mình họa một đơn vị LSTM (Nguồn: Coursera Sequence Models)

Cơ chế điều chỉnh luồng thông tin và cập nhật trạng thái ô nhớ được thể hiện như sau:

* + - * + Cổng quên (forget gate)

Cổng này quyết định thông tin nào là quan trọng để giữ lại từ bước phía trước. Thông tin từ lớp ẩn trước kết hợp với thông tin đầu vào hiện tại, sau đó được truyền qua hàm sigmoid, giá trị được chuyển đổi vào đường biên từ 0 đến 1. Giá trị càng gần 0 thì bị bỏ qua, càng gần 1 thì được giữ lại. Công thức tính toán giá trị cổng quên:

𝛤𝑓 = 𝜎(𝑊𝑓 [𝑎<𝑡−1>, 𝑥<𝑡>] + 𝑏𝑓)

* + - * + Cổng vào (input gate)

Cổng này quyết định thông tin nào là quan trọng để thêm vào từ bước hiện tại. Kết hợp thông tin từ lớp ẩn trước và thông tin đầu vào hiện tại cho truyền qua hàm sigmoid. Hàm này sẽ quyết định giá trị nào sẽ được cập nhật, bằng cách chuyển giá trị về khoảng biên từ 0 đến 1, gần 0 là không quan trọng, gần 1 là quan trọng. Đồng thời cũng truyền thông tin từ lớp ẩn trước và thông tin đầu vào hiện tại qua hàm tanh, giá trị được chuyển về khoảng

biên từ -1 đến 1. Sau đó nhân đầu ra sigmoid với đầu ra tanh, để quyết định thông tin nào quan trọng để giữ lại từ đầu ra hàm tanh. Công thức tính giá trị cổng vào:

𝛤𝑖 = 𝜎(𝑊𝑖 [𝑎<𝑡−1>, 𝑥<𝑡>] + 𝑏𝑖)

* + - * + Trạng thái ô (cell status)

Bây giờ, ta đã đủ thông tin để tính toán trạng thái mới của ô. Trạng thái mới được tính toán bằng trạng thái ô đang xét nhân với vector đầu ra của cổng quên để loại bỏ một số giá trị của ô trạng thái nếu nhân với giá trị gần 0, sau đó cộng giá trị này với đầu ra của cổng vào. Công thức để tính trạng thái ô hiện tại:

𝑐̃<𝑡> = 𝑡𝑎𝑛ℎ(𝑊𝑐 [𝑎<𝑡−1>, 𝑥<𝑡>] + 𝑏𝑐)

𝑐<𝑡> = 𝛤𝑖 ∗ 𝑐̃<𝑡> + 𝛤𝑓 ∗ 𝑐<𝑡−1>

Cổng ra (output gate)

Cổng này quyết định trạng thái ẩn truyền cho bước tiếp theo là gì. Truyền trạng thái lớp ẩn bước trước và giá trị đầu vào hiện tại vào hàm sigmoid. Sau đó đưa trạng thái mới của ô đã cập nhật qua hàm tanh, nhân giá trị đầu ra của hàm sigmoid và giá trị đầu ra của hàm tanh để quyết định những thông tin trạng thái ẩn nào nên mang theo, kết quả này và trạng thái mới của ô được truyền tới bước tiếp theo. Công thức tính giá trị cổng ra:

𝛤𝑜 = 𝜎(𝑊𝑜 [𝑎<𝑡−1>, 𝑥<𝑡>] + 𝑏𝑜)

* + - 1. Mạng nơ-ron hồi quy hai chiều (BiRNN - Bidirectional recurrent neural network)

Cùng với mạng LSTM, một trong những kiến trúc RNN được sử dụng nhiều nhất là mạng nơ-ron hồi quy hai chiều (BiRNN - Bidirectional recurrent neural network), khắc phục nhược điểm chỉ nhận thông tin từ các bước thời gian phía trước để dự đoán kết quả bước hiện tại.

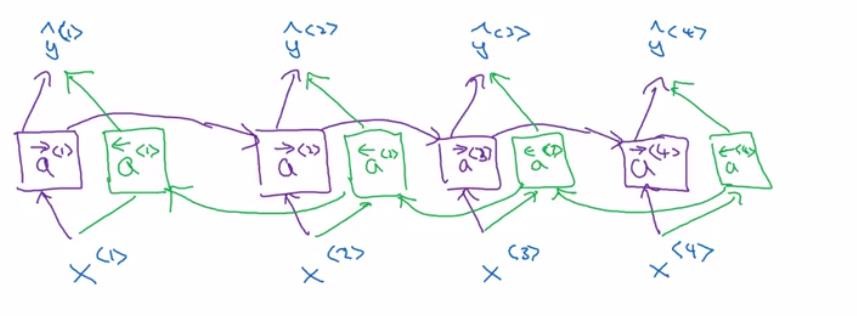
Trong kiến trúc này, có hai lớp nút ẩn. Cả hai lớp ẩn đều được kết nối với đầu vào và đầu ra. Hai lớp này được phân biệt ở chỗ, lớp đầu tiên có các kết nối hồi quy đến từ các bước thời gian trước, trong khi ở lớp thứ hai, hướng hồi quy của các kết nối bị đảo ngược, nghĩa là truyền giá trị kích hoạt ngược theo chiều chuỗi (hình 2.14). Ba phương trình sau mô tả một BiRNN.

ℎ(𝑡) = 𝜎(𝑊ℎ𝑥𝑥(𝑡) + 𝑊ℎℎ ℎ(𝑡−1) + 𝑏ℎ)

𝑧(𝑡) = 𝜎(𝑊𝑧𝑥 𝑥(𝑡) + 𝑊𝑧𝑧 𝑧(𝑡+1) + 𝑏𝑧)

𝑦̂(𝑡) = 𝑠𝑜𝑓𝑡𝑚𝑎𝑥(𝑊𝑦ℎ ℎ(𝑡) + 𝑊𝑦𝑧 𝑧(𝑡) + 𝑏𝑦)

Trong đó ℎ(𝑡) và 𝑧(𝑡) là các giá trị của lớp ẩn theo hai hướng lan truyền tới và lan truyền ngược tương ứng.



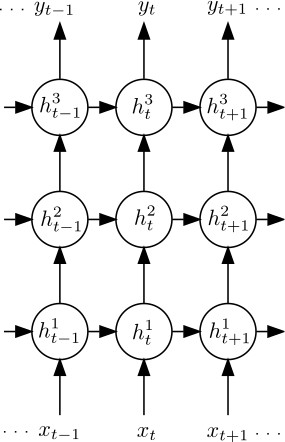
Hình 2.14: Minh họa mạng BiRNN (Nguồn: Coursera Sequence Models)

Tuy nhiên BiRNN có một hạn chế là không thể chạy liên tục, nó yêu cầu một điểm cuối cố định cho cả tương lai và quá khứ, vì BiRNN vẫn mang bản chất của một mạng RNN cơ bản. Trên thực tế LSTM và BiRNN khá tương thích với nhau, vì LSTM giới thiệu một đơn vị cơ bản mới để tạo thành một lớp ẩn, còn BiRNN liên quan đến việc kết nối giữa các lớp ẩn, bất kể chúng được cấu thành từ loại đơn vị nào. Sự kết hợp này, đưa đến khái

niệm BiLSTM. Dựa vào thực nhiệm [11], cách tiếp cận này đã đạt được kết quả hiện đại về nhận dạng chữ viết tay và phân loại âm vị.

* + - 1. Mạng nơ-ron hồi quy học sâu (Deep RNN - Deep Recurrent Neural Network)

Một yếu tố quan trọng cho sự thành công gần đây của các hệ thống lai là sử dụng kiến trúc học sâu. Deep RNN có thể được tạo bằng cách xếp chồng nhiều lớp ẩn RNN lên nhau, với đầu ra của một lớp trở thành đầu vào của lớp tiếp theo, được biểu diễn như hình 2.15.



Hình 2.15: Mạng nơ-ron hồi quy học sâu (Nguồn: [8])

### Các kỹ thuật trong nhận dạng tiếng nói

* + - 1. Trích chọn đặc trưng (Feature Extraction) và kỹ thuật trích chọn đặc trưng hệ số cepstral tần số Mel (Mel Frequency Cepstral Coefficients - MFCC)

Thành phần trích xuất đặc trưng dùng để tạo chuỗi vector đặc trưng giọng nói *X={x}* từ tín hiệu giọng nói dạng sóng *s*. Mục đích của thành phần này là trích xuất các thông tin hữu ích và loại bỏ những thông tin không liên quan (nhiễu) từ tín hiệu giọng nói, để từ đó áp dụng các công cụ toán học xử lý lên âm thanh mà không mất tính tổng quát. Các vector đặc trưng thường được tính toán trên cơ sở từng khung (frame), với thời lượng mỗi khung là 25ms, khoảng cách mỗi khung với nhau là 10ms. Trong thời lượng khung ngắn như vậy, tiếng nói được xem là đứng yên. Với mỗi một khung, trích chọn ra một vector đặc trưng, thông dụng nhất là hệ số cepstral tần số Mel (MFCC - Mel Frequency Cepstral Coefficients).

MFCC dựa trên hệ thống thính giác ngoại vi của con người. Nhận thức của con người về nội dung tần số âm thanh về tín hiệu giọng nói không tuân theo thang đo tuyến tính. Do đó, với mỗi âm có tần số thực 𝑡, được đo bằng Hz thì cường độ chủ quan được đo trên thang đo Mel. Thang đo tần số Mel là khoảng cách tần số tuyến tính dưới 1000 Hz và khoảng cách logarit trên 1 kHz. Như một điểm tham chiếu, cường độ của âm 1 kHz, cao hơn 40 dB so với ngưỡng nghe nhận thức, được xác định là 1000 Mels.

Việc tính toán MFCC bao gồm những bước sau.

* + - * + Đóng gói tần số Mel

Đối với mỗi âm có tần số thực tế (f), được đo bằng Hz, cường độ chủ quan được đo trên thang đo được gọi là thang âm 'mel'. Do đó, chúng ta có thể áp dụng công thức xấp xỉ sau để tính mels từ tần số f được đo bằng Hz.

𝑀𝑒𝑙(𝑓) = 2595 ×𝑙𝑜𝑔 𝑙𝑜𝑔 10 (1 +

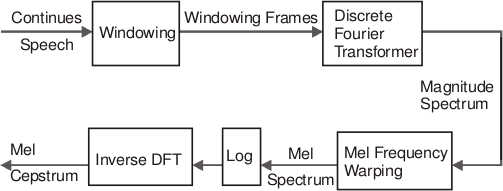
𝑓

)

700

Cách tiếp cận này mô phỏng phổ chủ quan sử dụng ngân hàng bộ lọc, một bộ lọc cho mỗi thành phần tần số Mel mong muốn. Ngân hàng bộ lọc thang đo Mel là một chuỗi các bộ lọc thông dải tần hình tam giác được thiết kế để mô phỏng bộ lọc thông dải được cho là xảy ra trong hệ thống thính giác con người.

* + - * + Tính cepstrum

Trong bước cuối cùng này, ta chuyển đổi phổ Mel trở lại thời gian, kết quả nhận được là MFCC. Biểu diễn cepstral của phổ giọng nói cung cấp một biểu diễn tốt về đặc tính phổ cục bộ của tín hiệu phân tích khung đã cho. Vì hệ số phổ Mel (và logarit của chúng) là số thực, nên ta có thể chuyển chúng thành miền thời gian bằng cách sử dụng biến đổi cosin rời rạc (DCT). Sơ đồ hoàn chỉnh cho tính toán MFCC được thể hiện ở hình 2.16.

Hình 2.16: Sơ đồ hoàn chỉnh cho tính toán MFCC (Nguồn: [2]) 2.1.5.2 Phân loại thời gian kết nối (CTC - Connectionist Temporal Classification)

Theo [8], mạng thần kinh thường được huấn luyện dựa trên phân loại mức khung (frame- level) trong nhận dạng giọng nói. Điều này yêu cầu một mục tiêu huấn luyện riêng cho mỗi khung, sự căn chỉnh giữa âm thanh và chuỗi lời dịch được xác định bởi HMM. Tuy nhiên việc căn chỉnh chỉ đáng tin cậy khi bộ phân loại được đào tạo, dẫn tới sự phụ thuộc vòng giữa phân đoạn và nhận dạng. Hơn nữa, việc căn chỉnh là không liên quan đến phần lớn các nhiệm vụ nhận dạng giọng nói. Phân loại thời gian kết nối (CTC - Connectionist

Temporal Classification) là một hàm mục tiêu cho phép một RNN được huấn luyện cho các nhiệm vụ dịch chuỗi mà không yêu cầu bất cứ sự căn chỉnh nào giữa chuỗi đầu vào và chuỗi đích.

Lớp đầu ra chứa một đơn vị đơn cho mỗi nhãn phiên âm (ký tự, âm vị, nốt nhạc, ...) cộng thêm một ký tự phụ gọi là "trống" tương ứng với vị trị rỗng.

Cho một chuỗi đầu vào *x* có chiều dài *T*, vector đầu ra 𝑦𝑡 được chuẩn hoá với hàm softmax, thì xác suất sinh ra nhãn (label) hay ký tự trống với chỉ số *k* tại thời gian *t* là:

𝑒𝑥𝑝(𝑦𝑘)

𝑡

𝑃𝑟(𝑘, 𝑡|𝒙) =

∑𝑘′ 𝑒𝑥𝑝(𝑦𝑘′)

𝑡

trong đó 𝑦𝑘 là phần tử thứ *k* của 𝑦 . Một CTC căn chỉnh ***a*** là một chuỗi các chỉ số kí tự

𝑡 𝑡

nhãn và ký tự trống có chiều dài *T*. Xác suất Pr(***a***|***x***) của ***a*** là tích các xác suất phát xạ tại các lần lặp:

𝑇

𝑃𝑟(𝑎|𝑥) = ∏ 𝑃𝑟(𝑎𝑡, 𝑡|𝑥)

𝑡=1

Cho một chuỗi phiên âm, ở đây có rất nhiều khả năng căn chỉnh cho chỉ số ký tự nhãn và ký tự trống. Ví dụ, giả sử kí tự "\_" dùng biểu diễn kí tự trống, thì các căn chỉnh (a, −, b, c,

−, −) và (−, −, a, −, b, c) đều tương ứng với phiên âm (a, b, c). Khi cùng một ký tự nhãn xuất hiện lặp lại liên tiếp trong một căn chỉnh, sự lặp lại sẽ bị xóa, vì vậy (a, b, b, b, c, c) và (a, −, b, −, c, c) đều tương ứng với (a, b, c). Kí hiệu B là toán tử xóa nhãn lặp lại, thì tổng xác suất của phiên âm đầu ra y sẽ bằng tổng các xác suất của các căn chỉnh tương ứng với nó:

𝑃𝑟(𝑦|𝑥) = ∑ 𝑃𝑟(𝑎|𝑥)

𝑎∈ℬ−1(𝑦)

Điều này tích hợp trên tất cả các căn chỉnh có thể có, cho phép mạng được huấn luyện với dữ liệu chưa được phân tách.

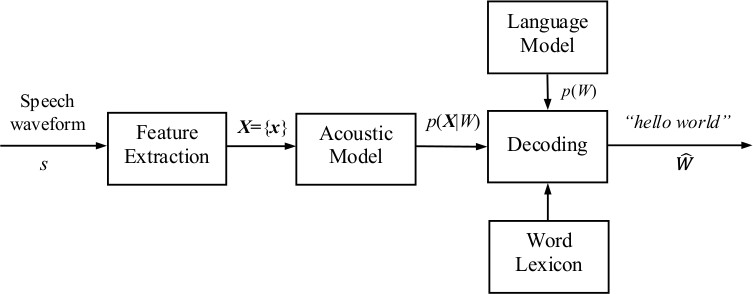
Dựa theo những kiến thức đã trình bày, phần tiếp theo nhóm sinh viên sẽ trình kiến trúc hệ thống nhận dạng tiếng nói nói chung, làm tiền đề cho thiết lập các giải pháp cho bài toán cụ thể là nhận dạng âm thanh tiếng Việt sẽ được trình bày ở những mục sau.

## HỆ THỐNG NHẬN DẠNG TIẾNG NÓI

Các mô hình nhận dạng tiếng nói mẫu được chia thành ba thành phần chính: trích xuất và lựa chọn vector tính năng, mô hình âm thanh và mô hình ngôn ngữ. Tùy theo bài toán tiếp cận mà giải pháp mỗi mô hình nhận dạng tiếng nói sẽ được tùy chỉnh để đưa cài đặt phù hợp. Dưới đây, nhóm sinh viên sẽ trình bày về hai giải pháp hệ thống nhận dạng tiếng nói điển hình là: hệ thống nhận dạng tiếng nói thông thường và hệ thống nhận dạng tiếng nói hiện đại đầu cuối (end-to-end). Đồng thời đưa ra lý giải tại sao hệ thống nhận dạng tiếng nói hiện đại lại chiếm ưu thế hơn trong những năm gần đây.

### Hệ thống nhận dạng tiếng nói thông thường

Trước khi đi qua phần hệ thống nhận dạng tiếng nói đầu cuối (End-to-End) và kiến trúc hệ thống mà nhóm sinh viên sẽ sử dụng trong luận văn này, thì mục này sẽ trình bày tóm tắt thông tin về hệ thống nhận dạng tiếng nói thông thường. Hình 2.17 là sơ đồ khối của một hệ thống nhận dạng tiếng nói thông thường, gồm năm thành phần chính: trích chọn đặc trưng, mô hình âm thanh, mô hình ngôn ngữ, từ điển phát âm (word lexicon) và bộ giải mã.



Hình 2.17: Sơ đồ khối của một hệ thống nhận dạng tiếng nói thông thường (Nguồn: [9])

* + - 1. Trích chọn đặc trưng (Feature Extraction)

Thành phần này dùng để trích xuất vector đặc trưng của tín hiệu âm thanh đầu vào, giữ lại những thông tin có ích để làm đầu vào cho khối mô hình âm thanh phía sau và loại đi những thông tin dư thừa (ví dụ nhiễu).

* + - 1. Mô hình âm thanh (Acoustic Model)

Mô hình âm thanh dùng để mô hình thống kê các tính năng cho mỗi đơn vị tiếng nói, ví dụ như một từ hoặc một âm vị. Đầu vào là đặc trưng của tiếng nói X, đầu ra là xác suất có điều kiện của từ (word) hay âm vị (phoneme), 𝑃(𝑊). Các tham số của mô hình âm thanh được xác định thông qua quá trình huấn luyện trên tập mẫu có trước. Mô hình xác suất được sử dụng phổ biến cho mô hình âm thanh là mô hình Markov ẩn (HMM - Hidden Markov Model) bởi mô hình này cung cấp một khung mô hình đơn giản và hiệu quả để mô hình hóa chuỗi vector theo thời gian. Để mô tả phân bố xác suất của các đặc trưng đầu vào về không gian người ta thường dùng mô hình Gaussian Mixture Model (GMM) [9].

Trong mô hình âm thanh của hệ thống nhận dạng âm thanh liên tục bộ từ vựng lớn, một HMM thường được dùng để mô hình đơn vị cơ bản của tiếng nói gọi là ngữ âm hay âm vị. Việc ánh xạ từ từ sang chuỗi âm vị được thực hiện bởi từ điển phát âm.

Phát âm của âm vị bị ảnh hưởng đáng kể bởi hiệu ứng cộng tác, vì vậy các âm vị thường được mở rộng để bao gồm ngữ cảnh của những âm vị xung quanh. Mô hình âm học thường sử dụng bộ ba âm vị (triphones) là đơn vị tiếng nói.

Một vấn đề khi sử dụng mô hình phụ thuộc vào ngữ cảnh như bộ ba âm vị là nhận thức dữ liệu. Độ phức tạp của mô hình tăng theo cấp lũy thừa so với số âm vị trong ngữ cảnh. Ví dụ trong tiếng Anh, có 40 đơn âm, suy ra số bộ ba âm vị bằng 403 = 64000.

* + - 1. Mô hình ngôn ngữ (Language Model)

Mô hình ngôn ngữ lưu trữ những những tri thức (prior knowledge) về từ ngữ, ngữ pháp và những thông tin liên quan đến một ngôn ngữ. Những kiến thức này được thay đổi liên tục theo thời gian, do đó mô hình ngôn ngữ cần được cập nhật thường xuyên. Đôi khi chúng ta không nghe rõ từ (do mô hình âm thanh yếu) nhưng ta vẫn đoán được gần như chính xác từ ta không nghe được. Đó là do mô hình ngôn ngữ dựa vào những thông tin trước đó như về ngữ pháp. Ví dụ, mô hình âm thanh khó khăn để phân biệt hai câu sau do cách phát âm gần giống nhau.

Câu một: "Cậu bé ấy là chú tiểu ở chùa Linh Quy Pháp Ẩn" Câu hai: "Cậu bé ấy là chú tiều ở chùa Linh Quy Pháp Ẩn"

Mô hình ngôn ngữ đưa ra xác suất câu một cao hơn câu hai, nên kết quả xuất ra là câu một.

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−1 … 𝑊1). N thường bằng 3 hoặc 4 cho các hệ thống nhận dạng tiếng nói. Mô hình ngôn ngữ n-gram

tuy đơn giản nhưng có nhược điểm là không mô tả được sự phụ thuộc dài do giới hạn của n.

* + - 1. Từ điển phát âm (Lexicon/Pronunciation Dictionary)

Mô hình âm học dùng để mô hình hóa những phần tử nhỏ nhất của tiếng nói gọi là âm vị (phoneme). Trong khi đó, mô hình ngôn ngữ lại thường sử dụng từ để mô hình hóa. Do vậy cần có một cầu nối giữa hai mô hình này đó là từ điển phát âm. Từ điển phát âm mô tả cách phát âm của một từ bằng cách biểu diễn từ đó dưới dạng một chuỗi các âm vị.

* + - 1. Bộ giải mã (Decoding)

Bộ giải mã có nhiệm vụ giải mã chuỗi các vector đặc trưng thành biểu diễn tượng trưng. Nó sử dụng mô hình âm thanh và từ điển phát âm để cung cấp một điểm xác suất 𝑃(𝑊) cho từng chuỗi từ giả thuyết (𝑊). Mô hình ngôn ngữ được áp dụng đồng thời để tính điểm xác xuất mô hình ngôn ngữ là 𝑃(𝑊) cho mỗi chuỗi từ giả thuyết. Nhiệm vụ của bộ giải mã là xác định chuỗi giả thuyết tốt nhất mà được chọn dựa trên sự kết hợp điểm xác xuất âm thanh và ngôn ngữ cho tín hiệu đầu vào. Quá trình này được thực hiện theo phương trình sau:

𝑊̂ = arg max 𝑃(𝑋)

Trong đó W là chuỗi từ giả thuyết mà có xác xuất lớn nhất tương ứng với chuỗi đầu vào vector đặc trưng X = {x}. 𝑃(𝑋) có thể được tính bằng công thức Bayes, nên công thức trên được viết lại như sau:

𝑊̂ = arg max 𝑃(𝑊) 𝑃(𝑊)

𝑃(𝑋)

### Hệ thống nhận dạng tiếng nói hướng đầu cuối (End-to-End)

* + - 1. Giới thiệu và đặt vấn đề

Mạng nơ-ron là mô hình học mạnh mẽ, đã đạt được những kết quả tiên tiến trong nhiều nhiệm vụ học máy có giám sát và không giám sát. Những tiến bộ gần đây về thuật toán và phần cứng máy tính đã giúp chúng ta huấn luyện mạng thần kinh đầu cuối (end-to-

end) hỗ trợ cho các nhiệm vụ mà trước đây đòi hỏi trình độ chuyên môn đáng kể từ con người. So với cách tiếp cận truyền thống, mạng thần kinh đầu cuối đòi hỏi ít sự nỗ lực hơn từ con người và mang lại kết quả cao hơn. Nó cho phép dịch trực tiếp dữ liệu âm thanh ra chuỗi chữ mà không cần đại diện ngữ âm trung gian. Những kết quả khả quan này có thể đạt được là nhờ sự phong phú đa dạng của tập dữ liệu huấn luyện.

Đặt vấn đề: Tại sao không dùng mô hình ẩn Markov tiếp tục trong hệ thống đầu cuối.

Mạng nơ-ron hồi quy không phải là mô hình duy nhất có khả năng đại diện cho sự phụ thuộc thời gian. Mô hình Makov ẩn (HMM) là mô hình một chuỗi được quan sát là phụ thuộc chính xác vào một chuỗi các trạng thái chưa được quan sát. Tuy nhiên, cách tiếp cận mô hình Makov truyền thống bị hạn chế do trạng thái của chúng phải được rút ra từ không gian trạng thái rời rạc có kích thước khiêm tốn S. Thuật toán lập trình động được sử dụng để thực hiện suy luận hiệu quả với các mô hình Makov ẩn trong thời gian

𝑂(|𝑆|2). Hơn nữa, bảng chuyển đổi xác xuất di chuyển giữa hai trạng thái thời gian liền kề bất kỳ có kích thước |𝑆|2. Do đó, các tính toán tiêu chuẩn trở nên không khả thi với HMM khi tập các trạng thái ẩn lớn dần. Hơn nữa, mỗi trạng thái ẩn có thể chỉ phụ thuộc vào trạng thái ngay trước đó. Mặc dù có thể mở rộng mô hình HMM để chiếm một cửa sổ ngữ cảnh lớn hơn bằng cách tạo một không gian trạng thái mới bằng với sản phẩm chéo của các trạng thái có thể có tại mỗi thời điểm trong cửa sổ. Điều này làm tăng không gian trạng thái theo cấp số nhân theo kích thước cửa sổ, đồng thời không thực tế về mặt tính toán các phụ thuộc tầm xa.

Với những hạn chế của mô hình Makov, mô hình nơ-ron hồi quy (RNN) dần mang lại các kết quả khả quan hơn. RNN và các phiên bản cải tiến của nó duy trì được các phụ thuộc tầm xa.

Ngoài ra, trong hệ thống nhận dạng tiếng nói cũ, lời dịch (câu chữ tương ứng với âm thanh đầu vào) dùng để huấn luyện hệ thống nhận dạng tiếng nói là từ vựng, thì biểu diễn trong các mạng thần kinh là dựa vào ngữ âm (phonetic). Vì vậy cần một từ điển cách phát

âm để ánh xạ từ từ sang chuỗi các âm vị (phoneme). Việc tạo những từ điển như vậy đòi hỏi nỗ lực đáng kể từ con người và được chứng minh là ảnh hưởng tới hiệu suất toàn cục.

* + - 1. Kiến trúc hệ thống nhận dạng giọng nói đầu cuối (end-to-end)

Hệ thống nhận dạng giọng nói đầu cuối (end-to-end) là hệ thống trực tiếp ánh xạ một chuỗi các tính năng âm thanh đầu vào tới một chuỗi các biểu đồ hoặc từ (word) [10].

Tùy vào mục đích của từng loại bài toán mà kiến trúc hệ thống nhận dạng tiếng nói đầu cuối được tùy chỉnh cài đặt cho phù hợp. Mặc dù hệ thống này có thể dịch trực tiếp từ âm thanh thô thành chuỗi chữ tương ứng, nhưng chi phí cao và hiệu suất không tốt, nên nhìn chung kiến trúc tổng thể của mô hình đầu cuối này cũng gồm ba thành phần chính: trích chọn đặc trưng, mô hình âm thanh và mô hình ngôn ngữ.

Ở mô hình âm thanh sẽ thay thế kiến trúc mô hình Markov ẩn thành sự kết hợp giữa mạng nơ-ron và bộ phân loại kết nối thời gian (CTC) để tận dụng thông tin từ bộ nhớ và giảm tác vụ căn chỉnh đầu vào tín hiệu và đầu ra kết quả. Cụ thể như sau, cho một chuỗi đầu vào 𝑥 = (𝑥1,…, 𝑥𝑇), một mạng hồi quy (RNN) tính toán trên chuỗi vector lớp ẩn ℎ =

(ℎ1,…, ℎ𝑇 ) và chuỗi vector đầu ra 𝑦 = (𝑦,…, 𝑦𝑇 ) bằng cách lặp lại hai phương trình sau

với 1 ≤ 𝑡 ≤ 𝑇:

ℎ𝑡 = 𝐻(𝑊𝑖ℎ𝑥𝑡 + 𝑊ℎℎ ℎ𝑡−1 + 𝑏ℎ)

𝑦𝑡 = 𝑊ℎ𝑜ℎ𝑡 + 𝑏𝑜

Trong đó W, b lần lượt là các tham số ma trận trọng lượng và vector sai lệch (bias vector), H là hàm kích hoạt của các lớp ẩn. Tuy nhiên, bản thân mạng RNN có hai điểm hạn chế (đã đề cập trong mục 2.1.4.1) nên tùy vào loại bài toán giải quyết mà người ta có thể đề xuất giải pháp sử dụng mạng LSTM hoặc mạng BiRNN, thậm chí là mạng BiLSTM để thay thế mạng RNN cơ bản.

Ngoài ra trong hệ thống nhận dạng tiếng nói thông thường, cần có một từ điển phát âm để làm cầu nối giữa mô hình âm thanh và mô hình ngôn ngữ, thì ở kiến trúc đầu cuối, không cần dùng tới thành phần này nữa, do mô hình ngôn ngữ và mô hình âm thanh đều sử dụng đơn vị từ (word) để mô hình hóa.

Trên thực tế, mô hình âm thanh thường được tích hợp với mô hình ngôn ngữ N-gram để tìm được chuỗi có xác suất cao nhất. Nguyên nhân là dữ liệu được đào tạo từ tập huấn luyện là không bao giờ đủ, nên cần có một bộ tri thức đầy đủ về ngữ pháp, từ ngữ của một ngôn ngữ để hiệu chỉnh lại kết quả.

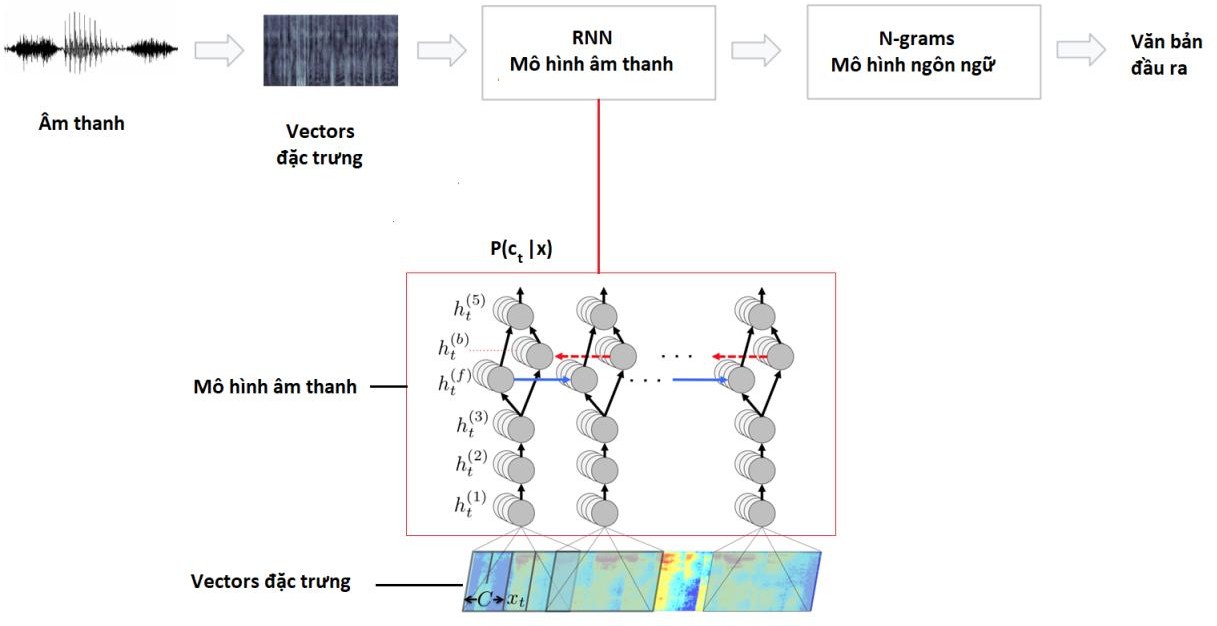
Thông qua chương 2, nhóm sinh viên đã trình bày các kiến thức cơ bản về âm thanh kỹ thuật số (digital audio), các lý thuyết kỹ thuật trong mạng nơ-ron (neural network) nhận dạng giọng nói, đồng thời so sánh và thiết lập các kiến trúc hệ thống nhận dạng giọng nói phổ thông đang được sử dụng. Chương tiếp theo, nhóm sinh viên sẽ làm rõ các giải pháp cụ thể cho từng thành phần trong hệ thống nhận dạng âm thanh tiếng Việt, hướng xây dựng máy chủ và cả ứng dụng trên nền tảng di động.

# CHƯƠNG 3: GIẢI PHÁP ĐỀ TÀI

## TỔNG QUAN GIẢI PHÁP KIẾN TRÚC MÔ HÌNH

Nhóm sinh viên đề xuất sử dụng kiến trúc mô hình nhận dạng đầu cuối (end-to-end) kết hợp cùng ý tưởng của bài báo Baidu. Bài báo được công bố vào tháng 12/2014 tại Silicon Valley AI Lab đã trình bày ý tưởng về một kiến trúc cụ thể để xây dựng một mô hình mạng nơ-ron hồi quy (RNN) tối ưu với hướng đi mới so với các hệ thống nhận dạng âm thanh truyền thống.

Kiến trúc tổng thể cho việc kết hợp 2 kiến trúc trên được minh họa cụ thể ở hình 3.1. Trong đó, nhóm nhấn mạnh chính vào phần mô hình âm thanh.



Hình 3.1: Tổng quan kiến trúc mô hình nhận dạng âm thanh

Sau đây, nhóm sẽ trình bày các giải pháp giải quyết vấn đề cho từng thành phần trong hệ thống nhận dạng âm thanh tiếng Việt, hướng xây dựng máy chủ và ứng dụng áp dụng thực tế trên nền tảng di động.

## GIẢI PHÁP RÚT TRÍCH ĐẶC TRƯNG

### Tổng quan về giải pháp

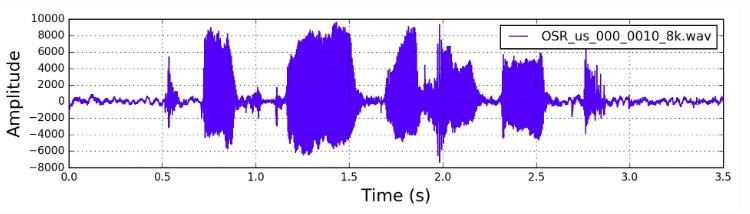
Dựa trên các kết quả thực nghiệm của [1] và [2], nhóm sinh viên đề xuất sử dụng kỹ thuật Mel-frequency cepstral coefficients (MFCC) với mục đích kết xuất được các đặc trưng của tiếng nói tiếng Việt dưới dạng một chuỗi các vector đặc trưng n chiều (mỗi chiều là một giá trị thực) để phục vụ cho quá trình huấn luyện.

MFCC hoạt động dựa trên hệ thống thính giác ngoại vi của con người. Nhận thức của con người về nội dung tần số của âm thanh đối với tín hiệu giọng nói không tuân theo thang đo tuyến tính. Vì vậy nên MFCC hoạt động tốt cho cả việc xử lý âm thanh liên tục lẫn âm thanh rời rạc.

### Chi tiết giải pháp

Nhóm sinh viên sử dụng đầu vào là tập tin WAV PCM 16 bit, với tần số lấy mẫu 16000 Hz, thời lượng không vượt quá 10s. Tập tin WAV là tín hiệu giọng nói rõ ràng (hoặc bị nhiễu) bao gồm một giọng nói phát ra từ một câu và xen lẫn một số đoạn tạm dừng ở giữa.

Tín hiệu thô sẽ có dạng như sau trong miền thời gian:



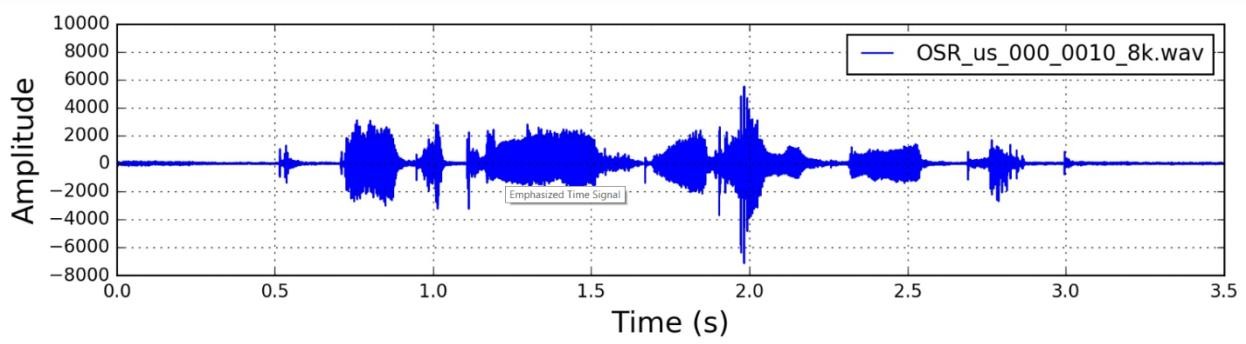
Hình 3.2: Tín hiệu thô trong miền thời gian (Nguồn: Haytham Fayek)

* + - * Bước 1

Bước đầu tiên là áp dụng bộ lọc tiền nhấn mạnh vào tín hiệu để khuếch đại tần số cao, điều này giúp cân bằng phổ tần số vì tần số cao thường có cường độ nhỏ hơn tần số thấp. Bộ lọc tiền nhấn mạnh có thể được áp dụng vào tín hiệu x với công thức sau:

𝑦(𝑡) = 𝑥(𝑡) − 𝛼𝑥(𝑡 − 1)

trong đó hệ số 𝛼 = 0.97. Sau quá trình này ta sẽ thu được tín hiệu có biểu đồ như hình 3.3.



Hình 3.3: Tín hiệu trong miền thời gian sau quá trình nhấn mạnh (Nguồn: Haytham Fayek)

* + - * Bước 2

Sau giai đoạn tiền nhấn mạnh, ta tách tín hiệu thành những khung ngắn với thông số chiều dài của cửa sổ phân tích (winlen) là 0.032s và chiều dài bước nhảy giữa các cửa sổ liên tiếp (winstep) là 0.02s, sau đó áp dụng hàm phân tích Hamming lên từng khung này. Sau giai đoạn tách khung cho tín hiệu, ta thực hiện biến đổi Fourier trên mỗi khung để tính phổ tần số với NFFT = 512, sau đó tính toán phổ công suất bằng phương trình sau:

𝑃 = |𝐹𝐹𝑇(𝑥𝑖)|2

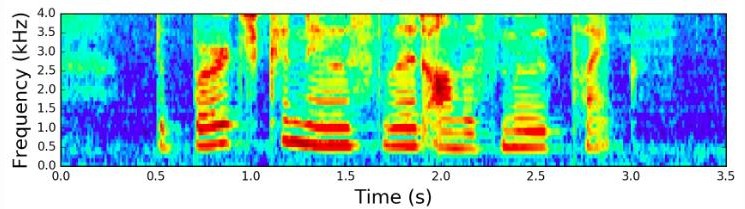
𝑁

Trong đó, 𝑥𝑖 là khung thứ i của tín hiệu 𝑥.

* + - * Bước 3

Bước cuối cùng để tính toán các bộ lọc ngân hàng là áp dụng các bộ lọc hình tam giác, cụ thể là 26 bộ lọc. Thang đo Mel nhằm bắt chước nhận thức về âm thanh của tai người, bằng cách phân biệt đối xử hơn ở tần số thấp và ít phân biệt đối xử hơn ở tần số cao.

Sau khi áp dụng ngân hàng bộ lọc cho phổ công suất của tín hiệu, ta thu được phổ như hình 3.4.

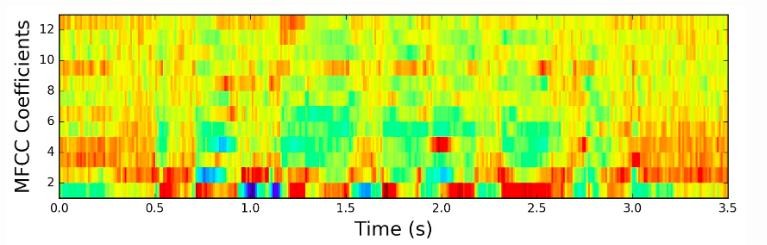


Hình 3.4: Minh họa khung phổ của tín hiệu (Nguồn: Haytham Fayker)

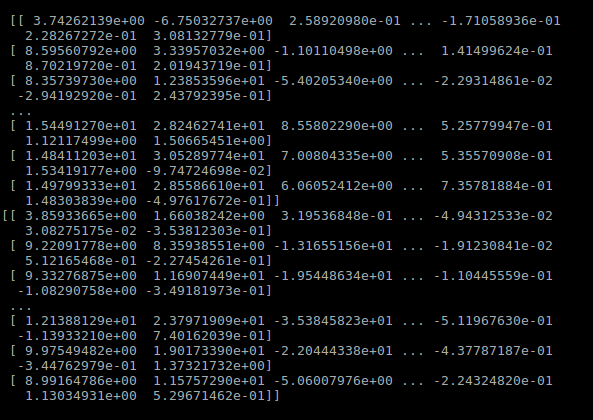
* + - * Bước 4

Ở đây, ta áp dụng biến đổi Cosine rời rạc (DCT - Discrete Cosine Transform) để giải mã các hệ số ngân hàng bộ lọc, mang lại một biểu diễn nén của các ngân hàng bộ lọc. Thông thường, đối với nhận dạng giọng nói tự động, các hệ số cepstrals từ 2 đến 13 được giữ lại và phần còn lại bị loại bỏ, cụ thể là số lượng cepstrum trả về (numcep) là 13.

Kết quả MFCC như được biểu diễn ở hình 3.5 và hình 3.6.



Hình 3.5: MFCCs (Nguồn: Haytham Fayker)



Hình 3.6: Biểu diễn số thực của ma trận MFCCs

## GIẢI PHÁP XÂY DỰNG MÔ HÌNH ÂM THANH

### Tổng quan về giải pháp

Dựa trên các đánh giá thực tế và điều kiện phần cứng (GPUs) lẫn lượng dữ liệu (data) cho phép, nhóm sinh viên chọn phương pháp học sâu (deep learning) để xây dựng hệ

thống mạng nơ-ron hồi quy (Recurrent neural network) trong mô hình âm thanh (acoustic model). Mô hình được đào tạo từ đầu đến cuối từ những âm thanh đã được rút trích để tạo ra các bản phiên âm, do đó với đủ dữ liệu và khả năng tính toán, mô hình có thể tự học được cả sự biển đổi tiếng ồn hoặc biến đổi loa.

### Mô hình mạng nơ-ron hồi quy (RNN) và khung huấn luyện

Cốt lõi của quá trình đào tạo một RNN là để nhận vào các phổ âm (speech spectrograms) và tạo ra phiên âm văn bản (text) tương ứng và cụ thể ở đây là văn bản tiếng Việt. Để dễ hình dung ta lấy ví dụ một tập huấn luyện 𝒳*= {*(𝑥(1), 𝑦(1)), (𝑥(2), 𝑦(2)), . . . } với *x* là một cách nói ở đầu vào và *y* là một nhãn tức văn bản tiếng Việt tương ứng ở đầu ra. Mỗi cách nói 𝑥(𝑖) là một chuỗi thời gian có độ dài 𝑇(𝑖), trong đó mỗi đoạn thời gian nhất định là

một một vector tính năng âm thanh 𝑥(𝑖), *t = 1, ...,* 𝑇(𝑖). Tiếp tục sử dụng phổ để làm các tính năng, vì vậy 𝑥(𝑖) biểu diễn cho độ lớn của khoảng tần số thứ *p* trong khung âm thanh tại thời điểm *t*. Mục tiêu của RNN là chuyển đổi một chuỗi đầu vào *x* thành một chuỗi xác suất ký tự cho nhãn *y*, với 𝑦̂𝑡 = ℙ(𝑐𝑡|𝑥), trong đó 𝑐𝑡 thuộc tập các kí tự chữ cái trong

𝑡

𝑡,𝑝

bảng chữ cái tiếng Việt và một vài kí tự khác như dấu cách, dấu nháy đơn.

Mô hình RNN được tạo thành từ 5 lớp ẩn (hidden layers). Đối với đầu vào *x*, các đơn vị ẩn ở lớp thứ *l* được kí hiệu là ℎ(𝑙) với quy ước ℎ(0) là đầu vào. Ba lớp đầu tiên trong mô hình không hồi quy. Đối với lớp đầu tiên, tại mỗi thời điểm *t*, đầu ra phụ thuộc vào khung

phổ 𝑥𝑡 cùng với bối cảnh khung *C* ở mỗi bên. Các lớp không hồi quy còn lại hoạt động trên dữ liệu độc lập cho mỗi bước thời gian. Do đó, với mỗi lần *t*, ba lớp đầu tiên được tính theo công thức:

ℎ(𝑙) = 𝑔(𝑊(𝑙)ℎ(𝑙−1) + 𝑏(𝑙))

𝑡 𝑡

Trong đó, *g(z)=min{max{0,z), 20}* là hàm kích hoạt cắt xén tuyến tính (ReLU-Rectified Linear Unit) với 𝑊(𝑙) là ma trận trọng số (weight matrix), 𝑏(𝑙) là tham số sai lệch (bias) cho lớp *l* và 20 là giá trị cắt (clipping value) trong ReLU. Lớp thứ tư trong mô hình RNN

là lớp hồi quy hai chiều (BiRNN-Bidirectional recurrent layer) được trình bày trong [3]. Hàm kích hoạt Tanh (Tanh activation) được sử dụng ở lơp này kết hợp với mạng Long Short Term Memory (LSTM) [4]. Lớp này có hai bộ đơn vị ẩn bao gồm một tập hồi quy thuận (foward) ℎ(𝑓) và một hồi quy ngược (backward) ℎ(𝑏) :

ℎ(𝑓) = 𝑔(𝑊(4)ℎ(3) + 𝑊(𝑓)ℎ(𝑓) + 𝑏(4))

𝑡 𝑡 𝑟 𝑡−1

ℎ(𝑏) = 𝑔(𝑊(4)ℎ(3) + 𝑊(𝑏)ℎ(𝑏) + 𝑏(4))

𝑡 𝑡 𝑟 𝑡+1

(Lưu ý rằng ℎ(𝑓) phải được tính toán tuần tự từ *t=1* đến *t=*𝑇(𝑖) cho cách nói thứ *i*, trong khi đó ℎ(𝑏) cũng sẽ được tính toán tuần tự nhưng theo chiều ngược lại tức từ *t=*𝑇(𝑖) đến *t=1*.)

Lớp thứ năm trong mô hình RNN là lớp không hồi quy (tương tự ba lớp đầu tiên), lớp này sẽ nhận cả đơn vị thuận và ngược được truyền từ lớp thứ tư để làm đầu vào:

ℎ(5) = 𝑔(𝑊(5)ℎ(4) + 𝑏(5)) , với ℎ(4) = ℎ(𝑓) + ℎ(𝑏)

𝑡 𝑡 𝑡 𝑡 𝑡

Lớp đầu ra của mô hình là một hàm softmax tiêu chuẩn mang lại xác suất ký tự dự đoán cho mỗi lần cắt *t* và ký tự *k* trong bảng chữ cái:

ℎ(6) = 𝑦̂

≡ ℙ(𝑐

𝑒𝑥𝑝(𝑊(6)ℎ(5) + 𝑏(6))

= 𝑘|𝑥) = 𝑘 𝑡 𝑘

𝑡,𝑘

𝑡,𝑘

∑ 𝑒𝑥𝑝(𝑊(6)ℎ(5) + 𝑏(6))

𝑗 𝑗 𝑡 𝑗

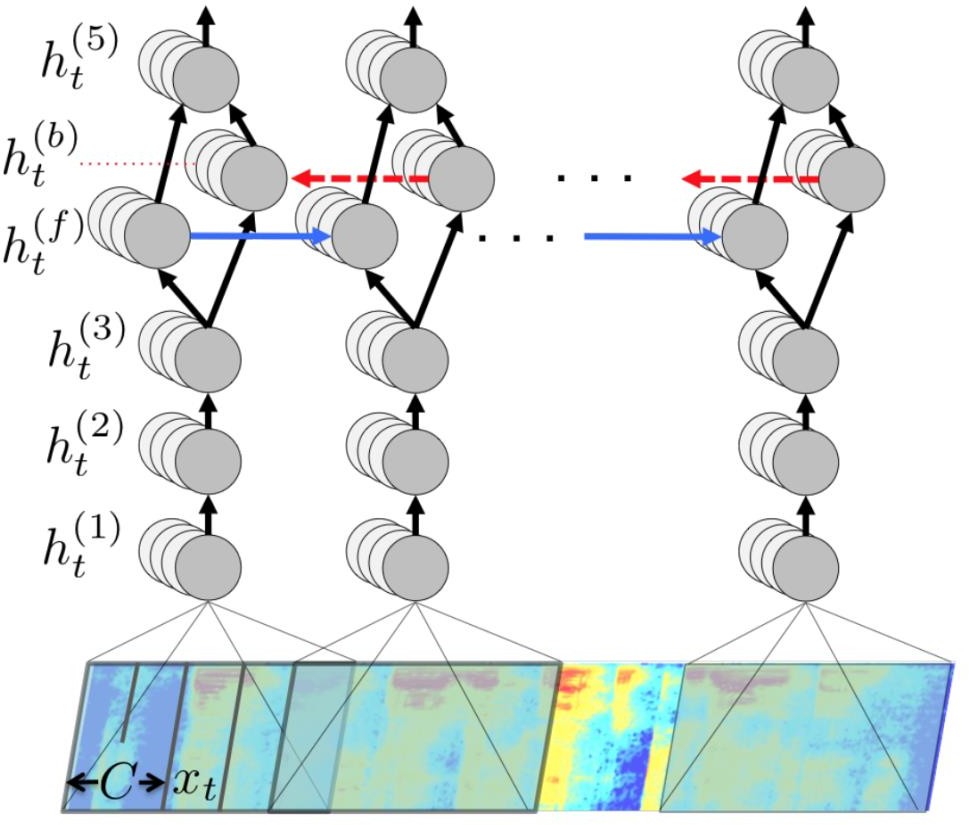
(Trong đó, 𝑊(6) đại diện cho cột thứ *k* trong ma trận trọng số và 𝑏(6) đại diện cho độ sai

𝑘 𝑘

lệch thứ *k*.)

Khi đã tính toán được dự đoán cho ℙ(𝑐𝑡|𝑥), ta tiếp tục tính toán tổn thất CTC (CTC-loss) ℒ(𝑦̂, 𝑦) [5] để đo độ lỗi trong việc dự đoán. Trong quá trình đào tạo, ta có thể đánh giá độ dốc ∇𝑦̂ ℒ(𝑦̂, 𝑦) liên quan đến đầu ra của mạng được đưa ra theo chuỗi ký tự cơ sở *y*. Từ đây, việc tính toán độ dốc đối với tất cả các tham số mô hình có thể được thực hiện thông

qua việc truyền ngược lại qua phần còn lại của mạng. Sau cùng, ta sẽ có được một mô hình RNN hoàn chỉnh như hình 3.7:



Hình 3.7: Kiến trúc hoàn chỉnh của mô hình RNN được xây dựng (Nguồn: [6])

## GIẢI PHÁP TÍCH HỢP MÔ HÌNH NGÔN NGỮ

Trên thực tế, mô hình âm thanh ở phần trước vẫn có khả năng cao xảy ra những lỗi đối với những từ hiếm hoặc không xuất hiện trong tập dữ liệu huấn luyện. Nguyên nhân là để đào tạo từ đủ dữ liệu lời nói để nghe tất cả các từ hoặc cấu trúc ngôn ngữ của tiếng Việt nói riêng và các ngôn ngữ khác nói chung là điều gần như bất khả thi. Vì vậy, nhóm đề xuất việc tích hợp hệ thống với mô hình ngôn ngữ N-grams (N-grams language model) trong khâu giải mã để tìm được chuỗi ký tự có độ chính xác cao nhất về mặt tương thích lẫn mặt ngữ nghĩa.

Với đầu ra ℙ(𝑐|𝑥) của RNN, ta tiến hành việc tìm kiếm để tìm ra một chuỗi ký tự

𝑐1, 𝑐2, 𝑐3,... có khả năng nhất dựa trên cả đầu ra của RNN lẫn mô hình ngôn ngữ, trong đó mô hình ngôn ngữ có vai trò diễn giải chuỗi ký tự thành từ. Cụ thể, ta hướng đến việc tìm một chuỗi *c* tối đa hóa mục tiêu kết hợp:

𝑄(𝑐) = 𝑙𝑜𝑔(ℙ(𝑐|𝑥)) + 𝛼𝑙𝑜𝑔(ℙ𝑙𝑚(𝑐)) + 𝛽𝑤𝑜𝑟𝑑\_𝑐𝑜𝑢𝑛𝑡(𝑐)

Trong đó α và β là các tham số có thể điều chỉnh nhằm điều khiển sự cân bằng giữa RNN, ràng buộc mô hình ngôn ngữ và độ dài của câu *c*. Thuật ngữ ℙ𝑙𝑚 đại diện cho xác suất của chuỗi *c* theo mô hình N-grams. Ta có thể tối đa hóa mục tiêu này bằng thuật toán

tìm kiếm chùm tia (Beam search), với kích thước chùm tia (Beam width) điển hình trong phạm vi từ 1000 đến 8000 (giá trị càng lớn đòi hỏi thời gian và tài nguyên phần cứng để huấn luyện càng nhiều), tương tự như cách tiếp cận được mô tả trong [7].

## GIẢI PHÁP XÂY DỰNG MÁY CHỦ

Máy chủ (server) được nhóm sinh viên xây dựng với mục đích tạo ra một cầu nối giữa mô hình đã được huấn luyện (model) và phía ứng dụng sản phẩm (client). Vì vậy trong giới hạn của khoá luận, máy chủ sẽ chỉ cung cấp duy nhất một giao diện lập trình (API) với chức năng chuyển đổi từ một file âm thanh (wav) giọng nói tiếng Việt thành một văn bản (text) tương ứng.

## GIẢI PHÁP XÂY DỰNG ỨNG DỤNG

Để ứng dụng hoá hệ thống nhận dạng âm thanh tiếng Việt, nhóm sinh viên quyết định xây dựng một ứng dụng di động lớp học thông minh trên nền tảng Android để ứng dụng kết quả của hệ thống vào một tình huống cụ thể có thể ứng dụng và thương mại hoá tốt.

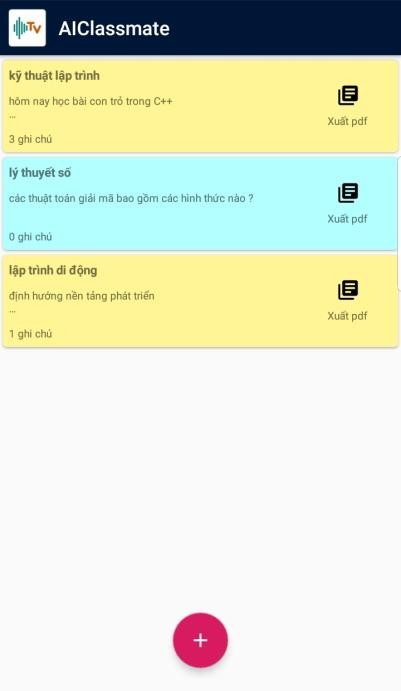
Ứng dụng lớp học thông minh do nhóm sinh viên xây dựng có chức năng chính là tạo một bài giảng thông qua việc chuyển đổi âm thanh tiếng Việt được ghi âm sang một văn bản tương ứng và cho phép người dùng lưu nội dung văn bản đó dưới dạng nội dung của

một bài giảng. Đồng thời ứng dụng cũng hỗ trợ một số chức năng bổ trợ khác như xuất bài giảng ở định dạng pdf và chia sẻ bài giảng.

### Thiết kế giao diện ứng dụng

Giao diện ứng dụng bao gồm 3 màn hình tương ứng với các công việc cụ thể:

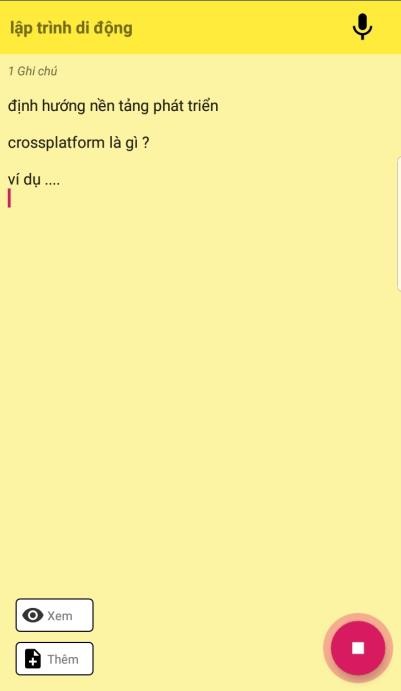
* + - * Giao diện màn hình danh sách bài giảng



Hình 3.8: Màn hình danh sách bài giảng

Danh sách các bài giảng cho phép người dùng quản lý và thao tác với các bài giảng. Cụ thể hỗ trợ thao tác xuất tập tin định dạng pdf cho bài giảng tương ứng. Nội dung trong tập tin pdf được căn chỉnh theo định dạng bố cục từng phần hợp lý bao gồm tiêu đề bài giảng, nội dung bài giảng và cuối cùng là phần ghi chú thêm. Các bài giảng sẽ được sắp xếp theo thứ tự từ cũ nhất đến mới nhất theo hướng từ trên xuống nhằm giảm thời gian tìm kiếm cho người dùng.

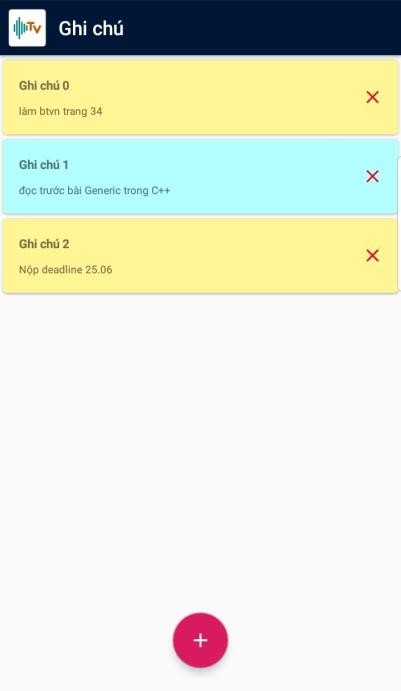
* + - * Giao diện màn hình thêm mới/ chỉnh sửa bài giảng

Hình 3.9: Màn hình thêm mới/ chỉnh sửa bài giảng

Giao diện đơn giản trực quan giúp định hướng người dùng xác định tính năng thêm mới hoặc chỉnh sửa một bài giảng. Tiêu đề và nội dung bài giảng hỗ trợ cho người dùng hai cách thức để thao tác bao gồm nhập trực tiếp thông qua bàn phím và thao tác ghi âm lời nói để chuyển đổi thành văn bản tương ứng. Góc trái dưới của màn hình là khu vực thao tác với ghi chú, người dùng có thể thêm mới hoặc xem các ghi chú trong bài giảng hiện tại nếu có.

* + - * Giao diện màn hình kết quả



Hình 3.10: Màn hình danh sách ghi chú

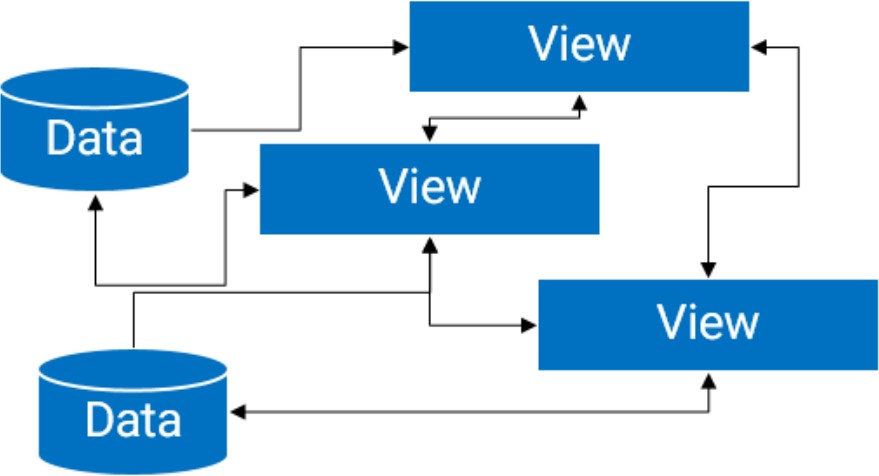
Danh sách các ghi chú cho phép người dùng quản lý và thao tác với các ghi chú. Cụ thể hỗ trợ thao tác xoá và chỉnh sửa nội dung của một ghi chú có trong danh sách. Các ghi chú được sắp xếp theo thứ tự từ cũ nhất đến mới nhất theo hướng từ trên xuống nhằm giảm thời gian tìm kiếm cho người dùng.

### Thiết kế kiến trúc ứng dụng

Để cài đặt các chức năng như trong thiết kế, ứng dụng AIClassmate cần đáp ứng các yêu cầu sau:

* + - * Khi người dùng thêm mới một bài giảng, hoặc thêm/chỉnh sửa một ghi chú, ứng dụng cần đảm bảo tất cả giao diện đều được cập nhật lại dữ liệu hiển thị.
      * Tiến trình thu âm và xử lý âm thanh không làm ảnh hưởng tới độ phản hồi của ứng dụng.

Như vậy, nhìn chung AIClassmate là ứng dụng đơn giản minh họa thực tế cho mô hình nhận dạng giọng nói mà nhóm sinh viên đã thiết kế, các tác vụ cũng không quá phức tạp và cồng kềnh. Tận dụng tính logic của khung làm việc Android (Android Framework), nhóm quyết định đề xuất một kiến trúc biến thể từ kiến trúc MVC (chi tiết lý thuyết có thể xem ở Phụ lục) tạo ra một khối logic chương trình tự nhiên hơn trong việc cài đặt giao diện người dùng là kiến trúc Model-View được minh họa như hình 3.11.



Hình 3.11: Minh họa kiến trúc Model-View

## TỔNG KẾT

Thông qua chương 3, sinh viên đã làm rõ được các giải pháp cụ thể cho từng thành phần trong hệ thống nhận dạng âm thanh tiếng Việt, hướng xây dựng máy chủ và cả ứng dụng trên nền tảng di động.

Nhóm sinh viên đã trình bày một hệ thống nhận dạng âm thanh tiếng Việt dựa trên việc học tập sâu (deep learning) từ đầu đến cuối có khả năng vượt trội hơn các hệ thống nhận dạng hiện đại trong hai môi trường đầy thách thức là nói chuyện và nói trong môi trường có tiếng ồn. Kết hợp lại, các giải pháp này cho phép một hệ thống nhận dạng tiếng nói

dựa trên dữ liệu có hiệu suất tốt hơn các phương pháp hiện có trong khi không còn phụ thuộc vào các giai đoạn xử lý phức tạp đã cản trở tiến trình tiếp theo. Nhóm sinh viên tin rằng phương pháp này sẽ tiếp tục được cải thiện khi tận dụng được sức mạnh tính toán phần cứng và kích thước tập dữ liệu tăng lên trong tương lai.

Chương kế tiếp sẽ trình bày về các thư viện, công cụ và những cách khắc phục khó khăn cụ thể nếu có cho các giải pháp đã trình bày ở chương này.

# CHƯƠNG 4: CÀI ĐẶT VÀ TRIỂN KHAI

## GIỚI THIỆU VỀ PYTHON VÀ THƯ VIỆN TENSORFLOW

### Python

Mã nguồn xây dựng mô hình huấn luyện của đề tài được phát triển dựa trên Python. Python là một ngôn ngữ lập trình thông dịch được sử dụng rất phổ biến trong lĩnh vực khoa học máy tính nhờ những ưu điểm sau:

* + - * Đa nền tảng

Python có thể chạy trên nhiều hệ điều hành như Windows, MacOS, Linux/Unix và một số hệ điều hành khác trên máy tính. Ngoài ra, Python còn có cả những phiên bản chạy được trên .NET, máy ảo Java. Tất cả chỉ với cùng một mã nguồn cho một công việc.

* + - * Đơn giản

Python có cú pháp rất đơn giản, rõ ràng. Cú pháp của Python dễ viết và dễ đọc hơn rất nhiều khi so sánh với những ngôn ngữ lập trình khác như Java, C/C++, C#, JavaScript, ... Điều này cũng giúp cho nhà phát triển tập trung vào việc phát triển giải pháp thay vì cú pháp.

* + - * Mã nguồn mở

Python là một dự án mã nguồn mở nên nhà phát triển có thể thoải mái sử dụng cho các mục đích cá nhân và vì vậy nên cộng đồng phát triển Python thường xuyên đưa ra những bản cập nhật mới nhằm tăng trải nghiệm cũng như tối ưu hoá Python.

* + - * Nhiều thư viện hỗ trợ

Python có một khối lượng lớn các thư viện tiêu chuẩn giúp cho công việc của nhà phát triển trở nên dễ dàng hơn rất nhiều, đặc biệt là các thư viện xử lý toán học của Python cực kỳ đa dạng và mạnh mẽ.

### Tensorflow

Thư viện Tensorflow được sử dụng trong việc tính toán các biểu đồ và các dữ liệu dưới dạng số hoá trong sản phẩm khoá luận. Là một thư viện mã nguồn mở hỗ trợ mạnh mẽ các phép toán học để tính toán trong máy học. Để xây dựng một mô hình huấn luyện cho đề tài, nhóm sử dụng các giao diện lập trình cấp thấp (low level APIs) mà Tensorflow cung cấp:

* + - * Tensor

Đây là một sự khái quát hóa các vectơ và ma trận cho các kích thước có khả năng cao hơn. Là cấu trúc dữ liệu đại diện cho tất cả các loại dữ liệu trong Tensorflow. Một tensor sẽ có 3 thuộc tính cơ bản nhất bao gồm:

* + - * + Số bậc (rank): giúp phân loại dữ liệu của tensor. (Scalar, Vector, Matrix, N- Tensor)
        + Số chiều (shape): giúp xác định mức độ tương hợp giữa các tensor khi thực hiện tính toán.
        + Kiểu dữ liệu (type): kiểu dữ liệu cho toàn bộ các thành phần (elements) trong tensor.
      * Graph

Đây là một loại đồ thị với các đỉnh (node) là đại diện cho biến đầu vào hoặc một phép tính toán và các cạnh (edge) là đại diện cho dữ liệu truyền bên trong đồ thị tức dữ liệu đầu vào và đầu ra của các phép tính tại một đỉnh. Và trong tensorflow, tất cả thành phần bên trong một đồ thị đều ở dạng tensor. Cách xử lý tính toán theo hướng đồ thị này có thể giúp tensorflow tận dụng được khả năng tính toán song song bằng việc chia tách các phép toán độc lập và khả năng phân tán khi chia nhỏ công việc xử lý cho nhiều CPU, GPU khác nhau.

* + - * Session

Đây là một phiên xử lý được định nghĩa trong thư viện tensorflow. Một đối tượng phiên (session) cung cấp quyền truy cập vào các thiết bị trong máy cục bộ và các thiết bị từ xa bằng cách sử dụng thời gian chạy phân tán. Nó cũng lưu trữ thông tin về đồ thị (graph) để có thể chạy cùng một tính toán hiệu quả nhiều lần. Nếu không có phiên (session), mọi tính toán trong đồ thị (graph) sẽ gần như không được triển khai.

## CÀI ĐẶT DEEP SPEECH

### Giới thiệu

Vào tháng 12/2014, bộ phận nghiên cứu Baidu tại Silicon Valley AI Lab đã công bố một bài báo Deep Speech: mở rộng quy mô nhận dạng tiếng nói theo hướng đầu cuối (end-to- end). Bài báo này trình bày ý tưởng về một kiến trúc cụ thể để xây dựng một mô hình mạng nơ-ron hồi quy (RNN) tối ưu với hướng đi mới so với các hệ thống nhận dạng âm thanh truyền thống.

Ý tưởng của bài báo này nhanh chóng được triển khai bởi Mozilla và sớm được công bố dưới dạng mã nguồn mở vào năm 2016 và vẫn đang tiếp tục phát triển, cải tiến từng ngày. Nhóm sinh viên sẽ sử dụng mã nguồn DeepSpeech của Mozilla và chỉnh sửa lại để tối ưu tốt hơn trên giọng nói tiếng Việt với nền tảng hỗ trợ việc tính toán chính bên dưới là thư viện Tensorflow.

### Cài đặt

Đầu tiên ta cần tải về mã nguồn DeepSpeech từ github của Mozilla. Phần hướng dẫn cài đặt của Mozilla cũng có yêu cầu bắt buộc về những thư viện và công cụ để có thể chạy DeepSpeech:

* + - * Python 3.6
      * Hệ điều hành MacOS hoặc Linux.

Và để huấn luyện riêng một mô hình cụ thể từ một tập dữ liệu ngôn ngữ riêng, ta cần phải cài thêm các thư viện được liệt kê sẵn trong tập tin *requirements.txt* có sẵn trong mã nguồn của DeepSpeech (Lưu ý: cần cài đặt phiên bản của các thư viện phù hợp và tương ứng với yêu cầu).

Sau khi hoàn thành quá trình cài đặt, nhóm sinh viên sẽ chỉnh sửa mã nguồn và điều chỉnh một vài siêu tham số (hyperparameters) để đạt kết quả tốt nhất khi huấn luyện trên tập dữ liệu tiếng Việt.

## DỮ LIỆU HUẤN LUYỆN MÔ HÌNH

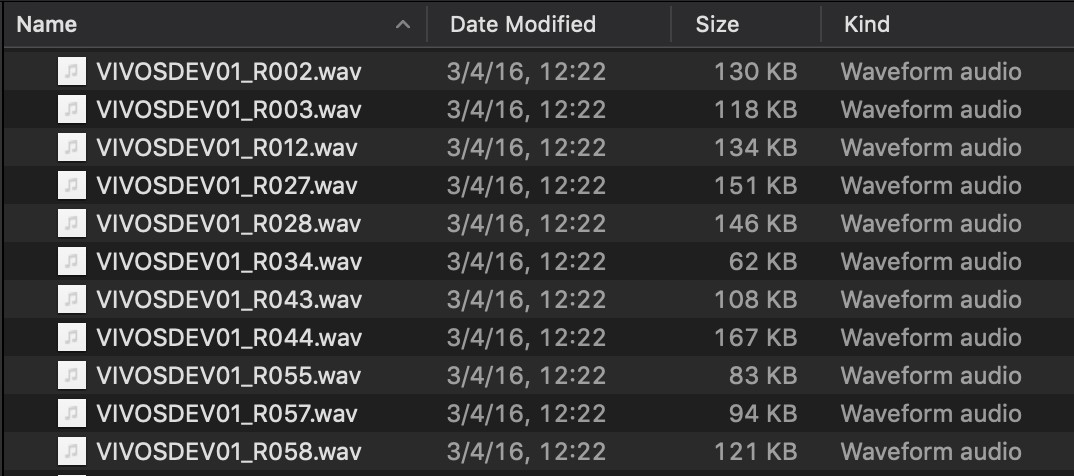
Để huấn luyện một mô hình nhận dạng âm thanh tiếng Việt dựa trên DeepSpeech thì bộ dữ liệu phải có hai thành phần chính bao gồm:

* Âm thanh

Để huấn luyện một hệ thống nhận dạng âm thanh tiếng Việt đủ tốt thì lượng dữ liệu âm thanh dùng để huấn luyện cũng phải đủ nhiều và đủ tốt. Nhóm sinh viên đã thu thập được khoảng 50 giờ dữ liệu thô (bao gồm khoảng 15 giờ dữ liệu từ phòng thí nghiệm AILab của Trường Đại học Khoa Học Tự Nhiên và khoảng 35 giờ dữ liệu từ dự án mở của FPT) để tiến hành huấn luyện. Ngoài ra dữ liệu âm thanh dùng để huấn luyện cũng cần thoả mãn các điều kiện:

* + Độ dài tối đa của mỗi tập tin âm thanh là 10 giây.
  + Tập tin âm thanh cần chuyển về định dạng \*.wav nhằm đồng nhất định dạng đầu vào và tăng chất lượng âm thanh.
  + Giá trị số lần lấy mẫu trên 1 giây (sample rate) là 16000 Hz.
  + Giá trị độ nét (bit depth) là 16 bit.

Bên cạnh những điều kiện về mặt kĩ thuật, chất lượng tiếng nói trong các tập tin âm thanh cần phải rõ ràng và cần phải hạn chế tạp âm.



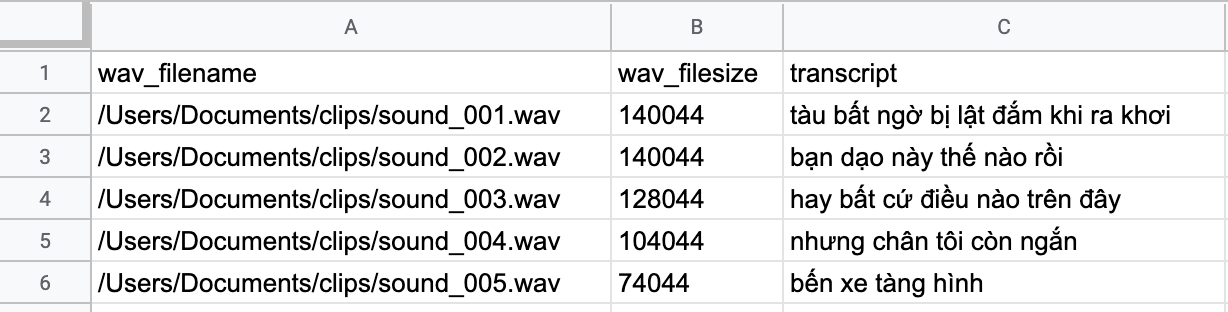
Hình 4.1: Minh hoạ bộ dữ liệu âm thanh

* Văn bản

Là bản dịch (transcript) tương ứng với nội dung lời nói trong các tập tin âm thanh.

Dữ liệu âm thanh và văn bản sau khi đã được kiểm chứng và sàn lọc sẽ được tổng hợp trong một tập tin định dạng \*.csv với nội dung được chia thành 3 cột theo thứ tự sau:

* wav\_filename: Nội dung cột này chứa đường dẫn trực tiếp đến một tập tin âm thanh.
* wav\_filesize: Nội dung cột này chứa dung lượng của tập tin âm thanh tương ứng tính theo byte.
* transcript: Nội dung cột này chứa bản dịch tương ứng với nội dung lời nói trong tập tin âm thanh.



Hình 4.2: Minh hoạ dữ liệu bản dịch văn bản tương ứng với mỗi tập tin âm thanh

Trong quá trình thu thập dữ liệu, nhóm sinh viên đã gặp rất nhiều vấn đề về chất lượng dữ liệu như: mất tiếng, tiếng tạp âm quá lớn, tiếng quá nhỏ, ... Và giải pháp nhóm sinh viên đưa ra là chỉnh sửa dữ liệu trực tiếp thông qua các phần mềm xử lý âm thanh (vd: Audacity, OceanAudio, ...) đối với những mẫu có mức độ sai lệch nhỏ hoặc trong tình trạng tiếng ồn lớn. Những mẫu bị sai lệch nhiều hoặc chất lượng quá thấp buộc nhóm sinh viên phải bỏ. Việc này một phần sẽ giảm bớt tình trạng gây nhiễu cho mô hình trong quá trình huấn luyện. Điều này dẫn đến việc chỉ còn khoảng hơn 43 giờ dữ liệu âm thanh tương ứng với 30000 mẫu để huấn luyện. Trong đó, dữ liệu được chia nhỏ thành 3 bộ train, dev, test với kích thước như sau:

* Bộ train: chứa khoảng 27600 mẫu ước tính tương đương khoảng 38 giờ dữ liệu.
* Bộ dev: chứa khoảng 700 mẫu ước tính tương đương khoảng 1 giờ dữ liệu.
* Bộ test: chứa khoảng 2300 mẫu ước tính tương đương khoảng 3 giờ dữ liệu.

## HUẤN LUYỆN MÔ HÌNH

Để tìm ra các tham số phù hợp cho mô hình, đặc thù ngôn ngữ của bộ dữ liệu tiếng Việt đã thu thập được, nhóm sinh viên sẽ tiến hành công việc huấn luyện mô hình hay cụ thể hơn là quá trình điều chỉnh các siêu tham số (hyperparameters) để tìm ra những giá trị tối ưu nhất có ảnh hưởng nhiều đến độ chính xác của mô hình.

Quá trình huấn luyện được nhóm triển khai trên máy tính có cấu hình bao gồm 1 GPU NVidia GTX 1060 12GB và chạy trên hệ điều hành Ubuntu 16.04. Sau một khoảng thời gian huấn luyện và ghi nhận kết quả, nhóm sinh viên đã thống kê được những siêu tham số có ảnh hưởng khá nhiều đến kết quả dự đoán của mô hình bao gồm các siêu tham số sau:

* Epoch: một epoch là một lần duyệt qua hết tất cả số lượng các mẫu trong trong tập huấn luyện.
* Batch\_size: số lượng mẫu được sử dụng cho một lần cập nhật trọng số.
* N\_hidden: độ rộng của một lớp được dùng khi khởi tạo các lớp trong mô hình.
* Validation\_step: Tần suất thực hiện việc xác thực mô hình trên mỗi đơn vị epoch.
* Dropout\_rate: tỷ lệ lược bỏ cho các quá trình chuyển tiếp giữa các lớp trong mô hình.

### Điều chỉnh n\_hidden

* + - * Chọn n\_hidden = 1024

|  |  |  |
| --- | --- | --- |
| **Tên tham số** | **Giá trị** | **Ghi chú** |
| n\_hidden | 1024 |  |
| train\_batch\_size | 1 | Giá trị mặc định |
| dev\_batch\_size | 1 | Giá trị mặc định |
| test\_batch\_size | 1 | Giá trị mặc định |
| epoch | 75 | Giá trị mặc định |

|  |  |  |
| --- | --- | --- |
| dropout\_rate | 0.05 | Giá trị mặc định |
| validation\_step | 0 | Giá trị mặc định |
| **WER** | **0.5731** | |

* + - * Chọn n\_hidden = 512

|  |  |  |
| --- | --- | --- |
| **Tên tham số** | **Giá trị** | **Ghi chú** |
| n\_hidden | 512 |  |
| train\_batch\_size | 1 | Giá trị mặc định |
| dev\_batch\_size | 1 | Giá trị mặc định |
| test\_batch\_size | 1 | Giá trị mặc định |
| epoch | 75 | Giá trị mặc định |
| dropout\_rate | 0.05 | Giá trị mặc định |
| validation\_step | 0 | Giá trị mặc định |
| **WER** | **0.4137** | |

* + - * Chọn n\_hidden = 256

|  |  |  |
| --- | --- | --- |
| **Tên tham số** | **Giá trị** | **Ghi chú** |
| n\_hidden | 256 |  |
| train\_batch\_size | 1 | Giá trị mặc định |
| dev\_batch\_size | 1 | Giá trị mặc định |
| test\_batch\_size | 1 | Giá trị mặc định |
| epoch | 75 | Giá trị mặc định |

|  |  |  |
| --- | --- | --- |
| dropout\_rate | 0.05 | Giá trị mặc định |
| validation\_step | 0 | Giá trị mặc định |
| **WER** | **0.4366** | |

Kết luận: chọn n\_hidden = 512 cho các lần huấn luyện sau.

### Điều chỉnh epoch

* + - * Chọn epoch = 60

|  |  |  |
| --- | --- | --- |
| **Tên tham số** | **Giá trị** | **Ghi chú** |
| n\_hidden | 512 |  |
| train\_batch\_size | 1 | Giá trị mặc định |
| dev\_batch\_size | 1 | Giá trị mặc định |
| test\_batch\_size | 1 | Giá trị mặc định |
| epoch | 60 |  |
| dropout\_rate | 0.05 | Giá trị mặc định |
| validation\_step | 0 | Giá trị mặc định |
| **WER** | **0.4441** | |

* + - * Chọn epoch = 70

|  |  |  |
| --- | --- | --- |
| **Tên tham số** | **Giá trị** | **Ghi chú** |
| n\_hidden | 512 |  |
| train\_batch\_size | 1 | Giá trị mặc định |

|  |  |  |
| --- | --- | --- |
| dev\_batch\_size | 1 | Giá trị mặc định |
| test\_batch\_size | 1 | Giá trị mặc định |
| epoch | 70 |  |
| dropout\_rate | 0.05 | Giá trị mặc định |
| validation\_step | 0 | Giá trị mặc định |
| **WER** | **0.4002** | |

* + - * Chọn epoch = 80

|  |  |  |
| --- | --- | --- |
| **Tên tham số** | **Giá trị** | **Ghi chú** |
| n\_hidden | 512 |  |
| train\_batch\_size | 1 | Giá trị mặc định |
| dev\_batch\_size | 1 | Giá trị mặc định |
| test\_batch\_size | 1 | Giá trị mặc định |
| epoch | 80 |  |
| dropout\_rate | 0.05 | Giá trị mặc định |
| validation\_step | 0 | Giá trị mặc định |
| **WER** | **0.4193** | |

Kết luận: chọn epoch = 70 cho các lần huấn luyện sau.

### Điều chỉnh batch\_size

* + - * Chọn batch\_size cho 3 bộ (train, dev, test) = (100, 20, 50)

|  |  |  |
| --- | --- | --- |
| **Tên tham số** | **Giá trị** | **Ghi chú** |
| n\_hidden | 512 |  |
| train\_batch\_size | 100 |  |
| dev\_batch\_size | 20 |  |
| test\_batch\_size | 50 |  |
| epoch | 70 |  |
| dropout\_rate | 0.05 | Giá trị mặc định |
| validation\_step | 0 | Giá trị mặc định |
| **WER** | **0.3722** | |

* + - * Chọn batch\_size cho 3 bộ (train, dev, test) = (200, 40, 100)

|  |  |  |
| --- | --- | --- |
| **Tên tham số** | **Giá trị** | **Ghi chú** |
| n\_hidden | 512 |  |
| train\_batch\_size | 200 |  |
| dev\_batch\_size | 40 |  |
| test\_batch\_size | 100 |  |
| epoch | 70 |  |
| dropout\_rate | 0.05 | Giá trị mặc định |
| validation\_step | 0 | Giá trị mặc định |
| **WER** | **0.3323** | |

* + - * Chọn batch\_size cho 3 bộ (train, dev, test) = (300, 60, 150)

|  |  |  |
| --- | --- | --- |
| **Tên tham số** | **Giá trị** | **Ghi chú** |
| n\_hidden | 512 |  |
| train\_batch\_size | 300 |  |
| dev\_batch\_size | 60 |  |
| test\_batch\_size | 150 |  |
| epoch | 70 |  |
| dropout\_rate | 0.05 | Giá trị mặc định |
| validation\_step | 0 | Giá trị mặc định |
| **WER** | **0.3109** | |

Kết luận: chọn batch\_size cho 3 bộ (train, dev, test) = (300, 60, 150) cho các lần huấn luyện sau.

### Điều chỉnh dropout\_rate

* + - * Chọn dropout\_rate = 0.15

|  |  |  |
| --- | --- | --- |
| **Tên tham số** | **Giá trị** | **Ghi chú** |
| n\_hidden | 512 |  |
| train\_batch\_size | 100 |  |
| dev\_batch\_size | 20 |  |
| test\_batch\_size | 50 |  |
| epoch | 70 |  |
| dropout\_rate | 0.15 |  |
| validation\_step | 0 | Giá trị mặc định |

|  |  |
| --- | --- |
| **WER** | **0.2983** |

* + - * Chọn dropout\_rate = 0.2

|  |  |  |
| --- | --- | --- |
| **Tên tham số** | **Giá trị** | **Ghi chú** |
| n\_hidden | 512 |  |
| train\_batch\_size | 200 |  |
| dev\_batch\_size | 40 |  |
| test\_batch\_size | 100 |  |
| epoch | 70 |  |
| dropout\_rate | 0.05 |  |
| validation\_step | 0 | Giá trị mặc định |
| **WER** | **0.2659** | |

* + - * Chọn dropout\_rate = 0.25

|  |  |  |
| --- | --- | --- |
| **Tên tham số** | **Giá trị** | **Ghi chú** |
| n\_hidden | 512 |  |
| train\_batch\_size | 300 |  |
| dev\_batch\_size | 60 |  |
| test\_batch\_size | 150 |  |
| epoch | 70 |  |
| dropout\_rate | 0.25 |  |
| validation\_step | 0 | Giá trị mặc định |

|  |  |
| --- | --- |
| **WER** | **0.2934** |

Kết luận: chọn dropout\_rate = 0.2 cho các lần huấn luyện sau.

### Điều chỉnh validation\_step

* + - * Chọn validation\_step = 1

|  |  |  |
| --- | --- | --- |
| **Tên tham số** | **Giá trị** | **Ghi chú** |
| n\_hidden | 512 |  |
| train\_batch\_size | 100 |  |
| dev\_batch\_size | 20 |  |
| test\_batch\_size | 50 |  |
| epoch | 70 |  |
| dropout\_rate | 0.15 |  |
| validation\_step | 1 |  |
| **WER** | **0.2584** | |

* + - * Chọn validation\_step = 5

|  |  |  |
| --- | --- | --- |
| **Tên tham số** | **Giá trị** | **Ghi chú** |
| n\_hidden | 512 |  |
| train\_batch\_size | 200 |  |
| dev\_batch\_size | 40 |  |
| test\_batch\_size | 100 |  |

|  |  |  |
| --- | --- | --- |
| epoch | 70 |  |
| dropout\_rate | 0.05 |  |
| validation\_step | 5 |  |
| **WER** | **0.2613** | |

* + - * Chọn validation\_step = 10

|  |  |  |
| --- | --- | --- |
| **Tên tham số** | **Giá trị** | **Ghi chú** |
| n\_hidden | 512 |  |
| train\_batch\_size | 300 |  |
| dev\_batch\_size | 60 |  |
| test\_batch\_size | 150 |  |
| epoch | 70 |  |
| dropout\_rate | 0.25 |  |
| validation\_step | 10 |  |
| **WER** | **0.2647** | |

Kết luận: chọn validation\_step = 1 cho các lần huấn luyện sau.

## ĐÓNG GÓI MÔ HÌNH

Để xuất mô hình sau khi hoàn tất quá trình huấn luyện, ta cần gán đường dẫn sẽ lưu mô hình vào cờ “*export\_dir*” khi chạy DeepSpeech. DeepSpeech sẽ xuất mô hình đã được huấn luyện dưới dạng một tập tin có định dạng \*.pb. Tập tin này có thể hiểu là mô hình được đóng gói từ đồ thị (graph) của tensorflow với các tham số được chọn lọc trong quá trình huấn luyện.

Tuy nhiên tập tin mô hình đóng gói ở định dạng \*.pb này sẽ có thời gian nạp (loading time) cao và tiêu thụ khá nhiều vùng nhớ do hệ thống cần phải nạp vào toàn bộ các khối (blocks) lên bộ nhớ (RAM) để thực hiện một bước xử lý hậu kỳ (post-processing) để dữ liệu có thể sử dụng được. Và để giải quyết vấn đề này ta có thể xuất ra một tập tin mới có định dạng \*.pbmm, định dạng này là các phiên bản ánh xạ bộ nhớ (memory mapped) của đồ thị để làm cho chúng hiệu quả hơn. Cụ thể tập tin ở định dạng \*.pbmm sẽ chỉ nạp các khối cần thiết của dữ liệu là có thể sử dụng được ngay lập tức.

## XÂY DỰNG MÁY CHỦ (SERVER)

Flask Framework và Heroku là hai nền tảng được nhóm sinh viên chọn để xây dựng hệ thống máy chủ nhằm đóng vai trò làm cầu nối giữa ứng dụng và mô hình nhận dạng âm thanh. Với các yếu tô như tốc độ triển khai nhanh gọn, sự tiện ích và tính thông dụng nên việc chọn hai nền tảng này để xây dựng máy chủ là quyết định phù hợp với nhu cầu đặt ra của nhóm sinh viên.

Hệ thống máy chủ trong giới hạn luận văn này sẽ cung cấp ra bên ngoài duy nhất một giao diện lập trình ứng dụng (Application Programming Interface-API) để chuyển đổi từ âm thanh tiếng nói nhận được và trả về dữ liệu văn bản tiếng Việt tương ứng. Tuy nhiên khi tiến hành xây dựng API nhóm sinh viên gặp hai vấn đề cần giải quyết:

* Đảm bảo dữ liệu được bảo mật.
* Kích thước dữ liệu âm thanh gửi đi có thể lớn.

Để giải quyết được cả hai vấn đề trên, nhóm sinh viên quyết định dùng phương thức đăng (POST) cho API nhóm xây dựng. Tuy nhiên việc chọn phương thức này cũng tạo ra một khuyết điểm cho hệ thống máy chủ, đó chính là về vấn đề tốc độ. Sau khi xây dựng API, ta có thể triển khai trực tiếp lên Heroku để hoàn tất việc triển khai máy chủ.

## CÀI ĐẶT CHỨC NĂNG GHI ÂM TRÊN ỨNG DỤNG

Một trong những bước quan trọng cho ứng dụng luận văn là cài đặt giải pháp ghi âm, để giải quyết vấn đề đầu tiên này nhóm sinh viên sử dụng giao diện lập trình ghi âm của Android để cài đặt chức năng này.

Khi sử dụng giao diện lập trình ghi âm của Android cung cấp, nhóm sinh viên đã rút ra được một số lưu ý sau:

* Để có thể sử dụng micro của thiết bị và tạo bản ghi âm, ta phải yêu cầu quyền truy cập đến micro và quyền truy cập vào bộ nhớ của thiết bị người dùng, việc yêu cầu quyền truy cập này có thể được thực hiện thông qua tập tin cài đặt đi kèm khi cài đặt ứng dụng Android. Tuy nhiên, từ phiên bản Android 6.0 trở đi, ứng dụng nhắm tới SDK 23 hoặc cao hơn nên ứng dụng sẽ yêu cầu người dùng cấp quyền truy cập micro và bộ nhớ của thiết bị khi ứng dụng đang chạy.
* Cấu hình các thông số cần thiết cho bản ghi âm để có thể tương thích với máy chủ (server), bao gồm các thông số như sau:
  + Tỉ lệ lấy mẫu trên một giây (sample rate): 16000 Hz.
  + Độ nét (bit depth): 16 bit.
  + Kênh (channel): Mono.
* Việc ghi âm cần được chạy trên một luồng phụ để đảm bảo tính tương tác phản hồi của ứng dụng trong lúc ghi âm.

Để thực hiện chức năng ghi âm, nhóm sinh viên sử dụng lớp AudioRecord do Android cung cấp cho các ứng dụng Java. Lớp này cung cấp khả năng chuyển đổi âm thanh từ phần cứng của thiết bị sang dạng dữ liệu mà người phát triển cần.

Do chức năng ghi âm hoạt động liên tục nên khi sử dụng lớp AudioRecord, nhóm sinh viên sử dụng định dạng tập tin PCM (Pulse Code Modulation) - là loại tập tin âm thanh không nén, không có phần tiêu đề (header) - để ứng dụng có thể lưu trữ đối với các dữ liệu âm thanh mới trước khi lưu xuống một bản ghi âm với định dạng WAV.

## CÀI ĐẶT CHỨC NĂNG THÊM BÀI HỌC TRÊN ỨNG DỤNG

Chức năng thêm bài học mới của ứng dụng AIClassmate cung cấp cho người dùng các lựa chọn sau:

* Thêm tiêu đề bài học bằng cách đọc vào micro hoặc nhập trực tiếp.
* Thêm nội dung bài học bằng cách đọc vào micro hoặc nhập trực tiếp.
* Thêm ghi chú cho bài học.
* Xem các ghi chú thuộc về bài học đó.

Ở lựa chọn thêm tiêu đề bằng cách đọc vào micro, ứng dụng tự động thu âm trong khoảng ba giây, sau đó gửi tập tin âm thanh lên máy chủ và đợi kết quả chữ trả về. Ở lựa chọn thêm nội dung bài học bằng cách đọc vào micro, ứng dụng sẽ chờ người dùng thực hiện thu âm xong, sau đó mới tiến hành gửi tập tin lên máy chủ để nhận dữ liệu chữ trả về. Cả hai lựa chọn này đều tuân thủ giải pháp ghi âm mà nhóm sinh viên đã trình bày ở mục 4.7.

Để quản lý bài học cùng với ghi chú thuộc về bài học đó, nhóm sinh viên lựa chọn SQLite là cơ sở dữ liệu dùng để lưu trữ với lược đồ lưu trữ đơn giản như hình 4.3.



Hình 4.3: Mô tả lược đồ lưu trữ bài giảng và ghi chú trong ứng dụng AIClassmate

## CÀI ĐẶT CHỨC NĂNG XUẤT TẬP TIN VĂN BẢN TRÊN ỨNG DỤNG

Hiện tại ứng dụng AIClassmate hỗ trợ người dùng xuất một bài học ra tập tin định dạng pdf tương ứng. Để có thể tạo một tập tin trên nền tảng Android, đầu tiên ta cần yêu cầu quyền truy cập vào bộ nhớ của thiết bị người dùng như đã trình bày trong mục 4.7. Đối với tập tin văn bản thông thường, nhóm sinh viên đã chọn sử dụng lớp CreatePdf thuộc gói com.uttampanchasara.pdfgenerator để thao tác với việc ghi dữ liệu lên các tập tin.

## TỔNG KẾT

Trong chương 4, nhóm sinh viên đã trình bày về cách thức cài đặt và triển khai cho các thành phần bao gồm hệ thống nhận dạng âm thanh, hệ thống máy chủ và các chức năng của ứng dụng. Nội dung chi tiết cho một số phần cài đặt được nhóm sinh viên trình bày chi tiết ở phần phụ lục, chương 5 sẽ là các tổng kết về quá trình thực hiện luận văn của nhóm sinh viên.

# CHƯƠNG 5: TỔNG KẾT VÀ ĐÁNH GIÁ

## KIẾN THỨC ĐẠT ĐƯỢC

* + - Lĩnh hội được kiến thức tổng quan về nhận dạng tiếng nói và các kỹ thuật trong mạng nơ-ron để nhận dạng tiếng nói.
    - Lĩnh hội được kiến thức và kinh nghiệm thiết kế, xây dựng, triển khai và kiểm thử khi phát triển hệ thống cung cấp dịch vụ.
    - Tiếp thu được kiến trúc và luồng xử lý của hệ nhận dạng giọng nói tiên tiến hướng đầu cuối nói chung và dự án DeepSpeech nói riêng, tạo nền tảng cho việc tìm hiểu và phát triển mô hình nhận dạng giọng nói tiếng Việt.
    - Học hỏi được kiến thức về mẫu thiết kế và áp dụng được các mẫu thiết kế vào ứng dụng sản phẩm nhằm giảm nhẹ hao phí khi chỉnh sửa và mở rộng các lớp giao diện, các lớp mô hình dữ liệu cũng như các quy trình xử lý của ứng dụng.
    - Học hỏi được cách hệ thống hóa kiến thức để áp dụng vào việc viết luận văn.

## KẾT QUẢ MÔ HÌNH HUẤN LUYỆN

* + - Dữ liệu huấn luyện: 28500 mẫu (~35 giờ)
    - Dữ liệu kiểm tra: 2300 mẫu (~4 giờ)
    - Mô hình ngôn ngữ: N-gram (N = 5)

|  |  |  |
| --- | --- | --- |
| Số từ kiểm tra | 33455 | |
| Số từ nhận đúng | 24809 | |
| Số từ nhận sai | 8646 | |
| Tỉ lệ sai từ (WER) | 25.84 % | |
| Phân loại lỗi | Số lỗi | Tỉ lệ gây lỗi |
| Sai chính tả | 177 | 2.05 % |
| Sai từ | 7787 | 90.06 % |
| Mất từ | 682 | 7.89 % |

## KẾT QUẢ HỆ THỐNG

### Môi trường phát triển

* + - * Hệ điều hành, Ubuntu 16.04 LTS, MacOS. X
      * Công cụ phân tích thiết kế: Astah
      * Công cụ xây dựng hệ thống: Visual Code
      * Công cụ kiểm thử API: Postman
      * Thư viện và nền tảng (framework) sử dụng:
        + Flask
        + Python 3

### Môi trường triển khai

* + - * Nền tảng đám mây Heroku.
      * Nền tảng đám mây Google Cloud (dự phòng).

### Chức năng đã cài đặt

* + - * Cung cấp dịch vụ nhận vào một tập tin âm thanh dạng wav và trả về đối tượng Json chứa văn bản tương ứng.

### Chức năng chưa cài đặt

* + - * Chưa hỗ trợ nhiều định dạng âm thanh.

## KẾT QUẢ ỨNG DỤNG

### Môi trường phát triển

* + - * Hệ điều hành: Ubuntu 16.04 LTS, MacOS. X
      * Hệ quản trị cơ sở dữ liệu: SQLite
      * Công cụ phân tích thiết kế: Astah
      * Công cụ xây dựng ứng dụng: Android Studio
      * Công cụ kiểm thử: Samsung Galaxy J7 Prime, Samsung Galaxy S7 EDGE.
      * Thư viện và nền tảng (framework) sử dụng:
        + Android Framework: Xây dựng giao diện, thao tác ghi âm trên microphone.
        + Okhttp3: Hỗ trợ tạo kết nối với hệ thống qua giao thức http.
        + PDFgenerator: Định dạng văn bản và xuất tập tin pdf.

### Môi trường triển khai

* + - * Thiết bị: Samsung Galaxy J7 Prime, Samsung Galaxy S7 EDGE
      * Hệ điều hành: Android 7.0 Nougat, Android 8.0 Orange

### Chức năng đã cài đặt

* + - * Sử dụng microphone của thiết bị để ghi âm.
      * Chỉnh sửa tiêu đề và nội dung bài giảng.
      * Thêm, xoá, sửa các ghi chú cho một bài giảng.
      * Xuất bài giảng dưới định dạng tập tin pdf.

## ĐỊNH HƯỚNG PHÁT TRIỂN VÀ NGHIÊN CỨU TRONG TƯƠNG LAI

### Công nghệ

* + - * Để cải tiến và nâng cấp hệ thống, nhóm sinh viên sẽ nghiên cứu nhằm chuyển giao việc triển khai hệ thống từ Heroku sang Google Cloud VPS.

### Ý tưởng hệ thống

* + - * Phát triển hệ thống cho phép chấp nhận và xử lý nhiều yêu cầu từ người sử dụng đồng thời.
      * Hỗ trợ nhận dạng thêm nhiều ngôn ngữ trên thế giới.
      * Đáp ứng tốc độ phản hồi trong thời gian thực (real-time).

## LỜI KẾT

Luận văn, hệ thống cung cấp dịch vụ và ứng dụng được xây dựng là sản phẩm kết tinh của một quá trình nghiên cứu, làm việc và học tập nghiêm túc của nhóm sinh viên. Mặc dù còn nhiều hạn chế về hệ thống và khả năng xử lý của hệ thống, song sản phẩm hệ thống cung cấp dịch vụ nhận dạng tiếng Việt VASR đã đem lại cho bản thân nhóm sinh viên được nhiều kiến thức cũng như phương hướng để triển khai dự án khởi nghiệp trong tương lai. Các hệ thống học sâu nói chung và nhận dạng tiếng nói đặc biệt là tiếng Việt

nói riêng đang là những chuyên ngành nổi bật trên thế giới hiện nay vì những trải nghiệm tuyệt vời mà chúng đem lại cho người dùng. Thông qua một quá trình học tập và nghiên cứu lâu dài, nhóm sinh viên sẽ sử dụng những kiến thức đã tích lũy được để tiếp tục nghiên cứu và phát triển sự nghiệp của bản thân.

# TÀI LIỆU THAM KHẢO

1. Syed Sibtain Khalid, Safdar Tanweer, Abdul Mobin, Afshar Alam, “*A comparative Performance Analysis of LPC and MFCC for Noise Estimation in Speech Recognition Task*”, International Journal of Electronics Engineering Research, ISSN 0975-6450 Volume 9, Number 3 (2017) pp. 377-390.
2. Vibha Tiwari*,* “*MFCC and its applications in speaker recognition*”, International Journal on Emerging Technologies 1(1): pp. 19-22, 2010.
3. M. Schuster, K. K. Paliwal, “*Bidirectional recurrent neural networks*”. IEEE Transactions on Signal Processing, 45(11): pp. 2673-2681, 1997.
4. Github blog Website. “Understanding LSTM Networks”

Internet: https://colah.github.io/posts/2015-08-Understanding-LSTMs/, [August 27, 2015]

1. A. Graves, S. Fernandez, F. Gomez, J. Schmidhuber. “Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks”. In ICML, pp. 369-376. ACM, 2006.
2. Nvidia developer blog Website. “*Deep Speech: Accurate Speech Recognition with GPU-Accelerated Deep Learning*”

Internet: https://devblogs.nvidia.com/deep-speech-accurate-speech-recognition-gpu- accelerated-deep-learning/, [February 25, 2015]

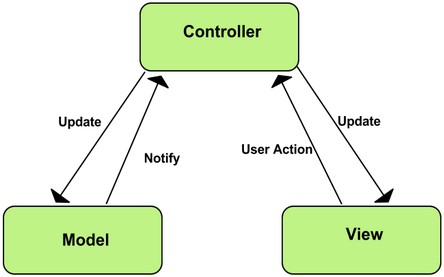
1. A. Y. Hannun, A. L. Maas, D. Jurafsky, A. Y. Ng. “*First-pass large vocabulary continuous speech recognition using bi-directional recurrent DNNs*”. abs/1408.2873, 2014.
2. Alex Graves, Navdeep Jaitly. “*Towards End-to-End Speech Recognition with Recurrent Neural Networks*”, pp.3-4.
3. Do Van Hai, “*Acoustic modeling for speech recognition under limited training data conditions*” PhD Thesis, Nanyang Technological University, 2015.
4. Bo Li, Yanzhang He, Shuo-Yiin Chang. “*Towards End-to-End Speech Recognition*” ISCSLP Tutorial 4, Nov. 26, 2018.
5. Zachary C. Lipton, John Berkowitz, Charles Elkan. “*A Critical Review of Recurrent Neural Networks for Sequence Learning*” June 5th, 2015, pp.10-11.
6. Heitaro Nakajima. “*Digital Audio Technology*” Hardcover – 1983, pp. 218-220.
7. Yani Andrew Ioannou. “Structural Priors in Deep Neural Networks” September 2017.

# PHỤ LỤC

## PHỤ LỤC 1: KIẾN TRÚC MVC

Kiến trúc MVC (Model - View - Controller) là kiến trúc phần mềm được ứng dụng rộng rãi trong việc phát triển các ứng dụng trang web và máy tính để bàn (Desktop PC). MVC chia một ứng dụng thành ba phần tương tác được với nhau để tách biệt giữa cách thức mà thông tin được xử lý nội hàm và phần thông tin được trình bày và tiếp nhận từ phía người dùng.

* + Mô hình (Model) là nơi chứa những logic xử lý nghiệp vụ tương tác với dữ liệu hoặc hệ quản trị cơ sở dữ liệu bên dưới.
  + Khung nhìn (View) là những thể hiển giao diện của ứng dụng và không tồn tại bất cứ logic nào.
  + Bộ điều khiển (Controller) là nơi tiếp nhận những yêu cầu xử lý được gửi từ người dùng, nó sẽ gồm những xử lý nhiều nghiệp vụ logic giúp lấy đúng dữ liệu thông tin cần thiết nhờ các nghiệp vụ lớp Model cung cấp và hiển thị dữ liệu đó ra cho người dùng nhờ lớp View.

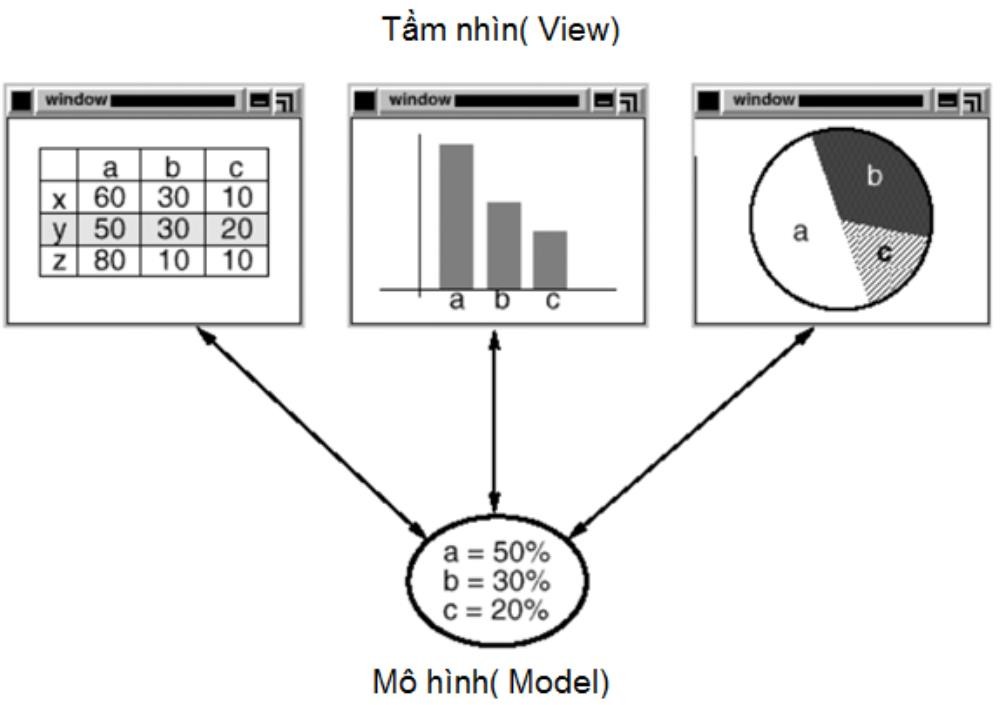


Phụ lục 1.1: Mô hình kiến trúc MVC (Nguồn: Internet)

Phụ lục 1.1 minh họa kiến trúc tổng thể của MVC, ở đây không có kênh giao tiếp trực tiếp nào nữa mô hình và khung nhìn, tất cả mọi hoạt động giao tiếp giữa hai thành phần này đều phải được thông qua bộ điều khiển. Bộ điều khiển là cầu nối giữa giao diện và hệ thống ứng dụng, khi người dùng tương tác với giao diện chương trình, bộ điều khiển sẽ nhận được tương tác này và gọi xử lý ở mô hình sau đó quyết định trả ra giao diện tương ứng dựa trên các phản hồi từ mô hình, hoặc cũng có thể trả ra giao diện một cách trực tiếp mà không thông qua mô hình. Ví dụ, người dùng mong muốn xem danh sách học sinh hiện có, thì bộ điều khiển nhận được tín hiệu này, truyền vào lấy danh sách dữ liệu từ các nghiệp vụ xử lý của mô hình. Mô hình thực hiện truy vấn và trả kết quả về cho bộ điều khiển, bộ điều khiển trả kết quả về cho khung nhìn, khung nhìn hiển thị danh sách học sinh lên cho người dùng.

Các khung nhìn phải đảm bảo sự thể hiện của nó là phản ánh lại trạng thái của mô hình bên dưới. Như vậy, khi có sự thay đổi dữ liệu phía dưới mô hình, các khung nhìn phải đảm bảo được thể hiện dữ liệu chính xác ngay tại thời điểm đó. Những yêu cầu trên đều được bộ điều khiển thông báo đến các khung nhìn. Việc tách biệt như vậy, tạo nên tính

mở rộng cho chương trình, ta có thể tạo ra nhiều khung nhìn khác nhau dựa trên một mô hình có sẵn mà không cần phải viết lại nó.

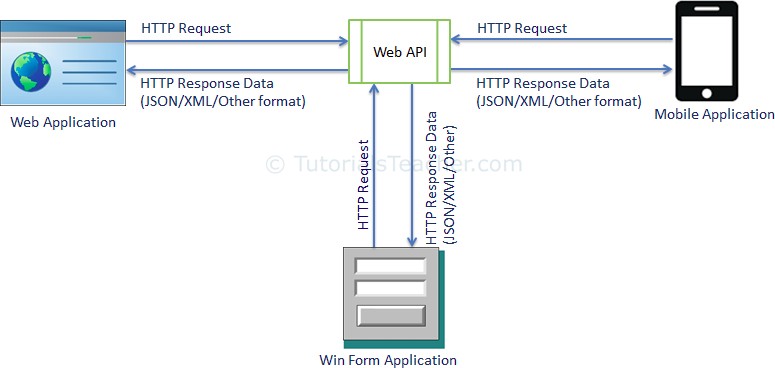


Phụ lục 1.2: Sự tương tác giữa View và Model (Nguồn: Design Pattern Gang of Four)

## PHỤ LỤC 2: WEB API

API là các là các giao diện lập trình ứng dụng (Application Programming Interface) cho phép lập trình viên giao tiếp với các cài đặt chức năng của một thư viện mà không cần phải biết rõ cài đặt ở mức thấp. Với sự phát triển của Internet, API không còn nhất thiết phải là một thư viện hay một khung làm việc bên trong thiết bị mà chương trình đang chạy, các nhà phát triển và thiết kế đưa ra khái niệm Web API, đem các chức năng cài đặt ở một thiết bị khác và cung cấp các lời gọi đến các API cho các máy khách thông qua mạng Internet.

Web API là một API trên web có thể được truy cập bằng giao thức HTTP, công nghệ mới này loại bỏ hoàn toàn những phức tạp và nhọc nhằn trong việc cài đặt hạ tầng và hướng người lập trình viên đến một môi trường lập trình linh hoạt và dễ dàng hơn. Sự dễ dàng khi cài đặt giao diện lập trình ứng dụng web còn thể hiện ở chỗ người lập trình có thể sử dụng định dạng JSON để trao đổi dữ liệu, như ta đã biết, định dạng XML sẽ rất tốn thời gian cho các bộ dịch trích xuất dữ liệu, thay vào đó, dữ liệu JSON khiến cho việc đọc hiểu cấu trúc dữ liệu và các tiến trình dịch trở nên dễ dàng và nhanh chóng hơn nên dần dần JSON đã dần có xu hướng thay thế cho XML truyền thống trong việc trao đổi dữ liệu trên Internet. Chúng ta có thể xây dựng API Web bằng các công nghệ khác nhau như Java, .NET, ... Ví dụ API REST của Twitter cung cấp quyền truy cập theo chương trình để đọc và ghi dữ liệu bằng cách ta có thể tích hợp khả năng của Twitter vào ứng dụng của mình.

Ngoài khả năng mở rộng, tính linh hoạt và dễ dàng tương thích, việc sử dụng giao diện lập trình ứng dụng Web còn giúp chương trình cài đặt trở nên nhẹ nhàng hơn vì đẩy các xử lý nghiệp vụ phức tạp lên máy chủ, từ đó độ hiệu quả về cả mặt thời gian và hiệu năng được nâng lên đáng kể. Hình phụ lục 2.1 mô tả cách giao tiếp của các ứng dụng tới giao diện lập trình ứng dụng Web.

Phụ lục 2.1: Mô tả cách giao tiếp của các ứng dụng tới giao diện lập trình ứng dụng Web (nguồn: Bizcoder)

## PHỤ LỤC 3: GIAO DIỆN CHỨC NĂNG ỨNG DỤNG

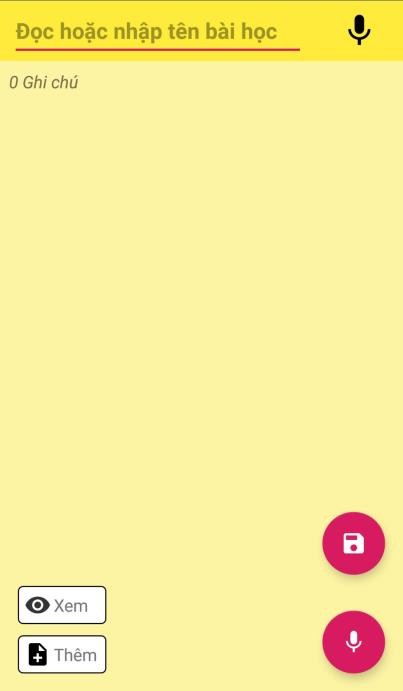
Dưới đây là hình ảnh giao diện các màn hình chức năng cụ thể của ứng dụng AIClassmate.

### Phụ lục 3.1: Màn hình xem danh sách bài giảng



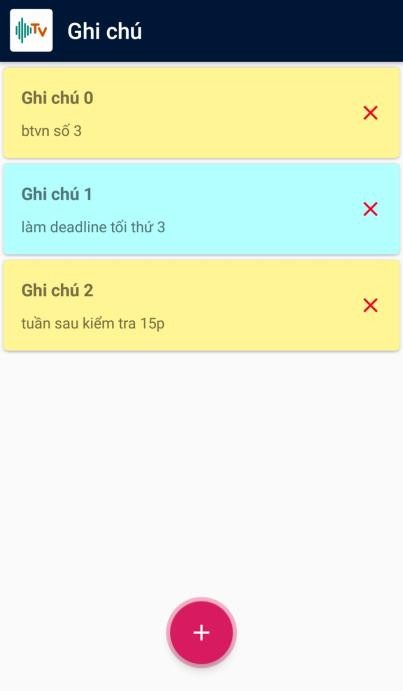
Phụ lục 3.1: Màn hình xem danh sách bài giảng

### Phụ lục 3.2: Màn hình thêm mới bài giảng



Phụ lục 3.2: Màn hình thêm mới bài giảng

### Phụ lục 3.3 Màn hình xem danh sách ghi chú của một bài giảng



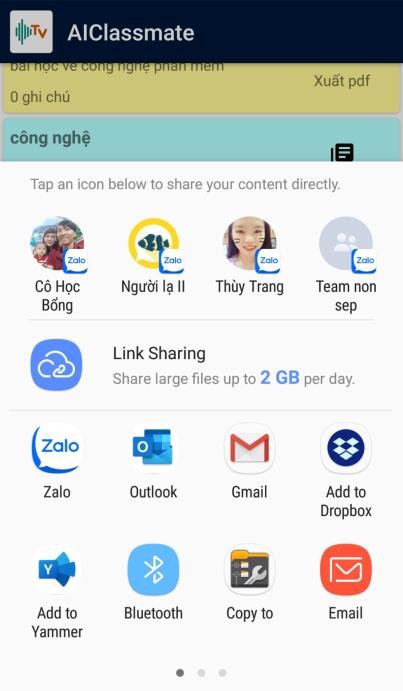
Phụ lục 3.3: Màn hình xem danh sách bài giảng

### Phụ lục 3.4: Màn hình thêm mới ghi chú



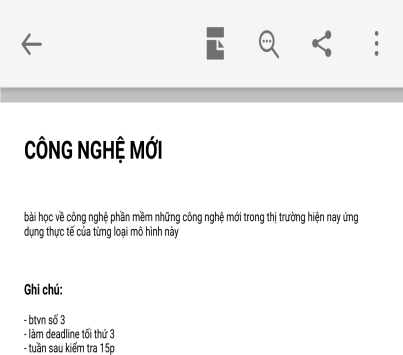
Phụ lục 3.4: Màn hình thêm mới ghi chú

### Phụ lục 3.5 Màn hình chia sẻ bài giảng



Phụ lục 3.5: Màn hình chia sẻ bài giảng

### Phụ lục 3.6 Bài giảng sau khi được xuất pdf



Phụ lục 3.6 Bài giảng sau khi được xuất pdf