HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN 1

THỰC TẬP CƠ SỞ

ĐỀ TÀI: NHẬN DIỆN NGÔN NGỮ KÍ HIỆU SỬ DỤNG TRÍ TUỆ NHÂN TẠO

Giảng viên hướng dẫn Đặng Hoàng Long

Sinh viên thực hiện: Nguyễn Thành An - B22DCCN006

Lê Xuân Dũng - B22DCCN127

Nguyễn Việt Hoàng - B22DCVT214

Nguyễn Ngọc Nhật - B22DCAT208

Bùi Minh Tùng - B22DCDT293

Hà Nội - 2025

Contents

DANH MỤC KÝ HIỆU VÀ CHỮ VIẾT TẮT

STT	Từ viết tắt	Tiếng Anh	Tiếng Việt (giải thích)
1	AI	Artificial Intelligence	Trí tuệ nhân tạo
2	ML	Machine Learning	Học máy
3	DL	Deep Learning	Học sâu
4	Seq2Seq	Sequence-to-Sequence	
5	RELU	Rectified Linear Unit Layer	Lớp Relu
6	GRU	Gated Recurrent Unit	Nút Hồi tiếp có Cổng
7	RNN	Recurrent Neural Network	Mạng nơ-ron hồi tiếp

List of Figures

CHƯƠNG 1. GIỚi THIỆU VỀ NHẬN DIỆN NGÔN NGỮ KÝ HIỆU

Chương đầu giới thiệu tổng quan về ngôn ngữ ký hiệu, nhấn mạnh vai trò quan trọng của nó trong giao tiếp và hòa nhập xã hội của cộng đồng người khiếm thính. Nội dung trình bày các rào cản hiện tại trong giao tiếp giữa người khiếm thính và cộng đồng, từ đó làm rõ tầm quan trọng của việc xây dựng hệ thống nhận diện ngôn ngữ ký hiệu tự động. Chương này cũng tổng quan các phương pháp truyền thống và hiện đại trong xử lý dữ liệu ký hiệu, đồng thời giới thiệu mô hình LSTM – một giải pháp phù hợp cho xử lý chuỗi cử chỉ tay. Cuối cùng, chương nêu rõ mục tiêu đề tài là phát triển một hệ thống nhận diện chính xác và có tính ứng dụng cao, góp phần thúc đẩy việc hòa nhập xã hội của người khiếm thính thông qua hỗ trợ công nghệ.

1.1 Giới thiệu chung về ngôn ngữ ký hiệu

Ngôn ngữ ký hiệu là một hệ thống giao tiếp bằng tay, cử chỉ, biểu cảm khuôn mặt và chuyển động cơ thể, được sử dụng rộng rãi bởi cộng đồng người khiếm thính để truyền đạt thông tin và biểu đạt ý nghĩ. Khác với ngôn ngữ nói và viết, ngôn ngữ ký hiệu không dựa vào âm thanh mà hoàn toàn sử dụng các yếu tố thị giác. Mỗi quốc gia, thậm chí mỗi khu vực, có thể có một hệ thống ngôn ngữ ký hiệu riêng biệt — ví dụ như ASL (American Sign Language) ở Mỹ hay VSL (Vietnamese Sign Language) ở Việt Nam.

Ngôn ngữ ký hiệu đóng vai trò cực kỳ quan trọng trong việc xây dựng cầu nối giữa người khiếm thính và cộng đồng nói chung, góp phần tạo nên một xã hội hòa nhập và công bằng hơn. Tuy nhiên, rào cản lớn nhất là phần lớn người bình thường không biết ngôn ngữ ký hiệu, dẫn đến khó khăn trong giao tiếp. Chính vì vậy, việc phát triển các hệ thống nhận diện ngôn ngữ ký hiệu bằng trí tuệ nhân tạo không chỉ mang ý nghĩa công nghệ mà còn có giá trị xã hội sâu sắc, giúp thu hẹp khoảng cách giao tiếp và tạo điều kiện cho người khiếm thính hòa nhập tốt hơn với cộng đồng.

1.1.1 Tầm quan trọng của việc nhận diện ngôn ngữ ký hiệu

Ngôn ngữ ký hiệu không chỉ là phương tiện giao tiếp quan trọng của cộng đồng người khiếm thính mà còn là một phần không thể thiếu trong việc đảm bảo quyền được tiếp cận thông tin và tham gia xã hội của họ. Trong khi ngôn ngữ nói được sử dụng phổ biến trong các hoạt động hàng ngày như học tập, lao động, y tế và pháp luật, thì người khiếm thính thường gặp nhiều rào cản do không thể nghe hoặc nói như người bình thường. Việc không thể sử dụng hoặc hiểu ngôn ngữ ký hiệu gây ra khoảng cách lớn trong giao tiếp, từ đó ảnh hưởng nghiêm trọng đến cơ hội học tập, nghề nghiệp, và sự hòa nhập xã hội của họ.

Tầm quan trọng của ngôn ngữ ký hiệu thể hiện rõ ở việc nó không chỉ giúp người

khiếm thính trao đổi với nhau, mà còn mở ra cơ hội để họ tương tác với xã hội nếu có sự hỗ trợ phù hợp từ phía công nghệ và cộng đồng. Trong bối cảnh phát triển mạnh mẽ của trí tuệ nhân tạo (AI) và học sâu (deep learning), các ứng dụng tự động nhận diện và phiên dịch ngôn ngữ ký hiệu đang trở thành công cụ hữu ích để phá vỡ rào cản giao tiếp này. Những hệ thống như vậy không chỉ hỗ trợ người khiếm thính trong môi trường học đường hay công sở mà còn giúp nâng cao nhận thức của xã hội về tầm quan trọng của giao tiếp toàn diện và công bằng. Do đó, việc nghiên cứu và phát triển công nghệ nhận diện ngôn ngữ ký hiệu không chỉ là một bài toán kỹ thuật mà còn mang ý nghĩa nhân văn sâu sắc.

1.1.2 Ứng dụng công nghệ AI trong nhận diện ngôn ngữ ký hiệu

AI đang thay đổi nhanh chóng cách chúng ta tương tác với các hệ thống thông minh. Trong lĩnh vực nhận diện ngôn ngữ ký hiệu, AI giúp tự động hóa việc phiên dịch các cử chỉ tay sang dạng văn bản hoặc âm thanh. Nhờ vào khả năng học từ dữ liệu lớn, AI có thể nhận dạng và tổng hợp thông tin từ nhiều nguồn khác nhau nhằm đạt độ chính xác cao.

1.2 Các nghiên cứu liên quan

1.2.1 Dịch Thuật Ngôn Ngữ Ký Hiệu Sử Dụng của Pooya Fayyazsanavi, Antonios Anastasopoulos, Jana Košecká

Nghiên cứu này [1] trình bày một phương pháp tiên tiến cho việc dịch thuật ngôn ngữ ký hiệu từ video sang văn bản nói, tập trung đặc biệt vào giai đoạn Gloss2Text và đạt được hiệu suất vượt trội so với các phương pháp hiện tại thông qua việc kết hợp các mô hình ngôn ngữ lớn tiền huấn luyện, kỹ thuật tăng cường dữ liệu và hàm mất mát làm mượt nhãn mới.

Bối Cảnh và Thách Thức Nghiên Cứu

Đặc Điểm của Bài Toán Dịch Thuật Ngôn Ngữ Ký Hiệu

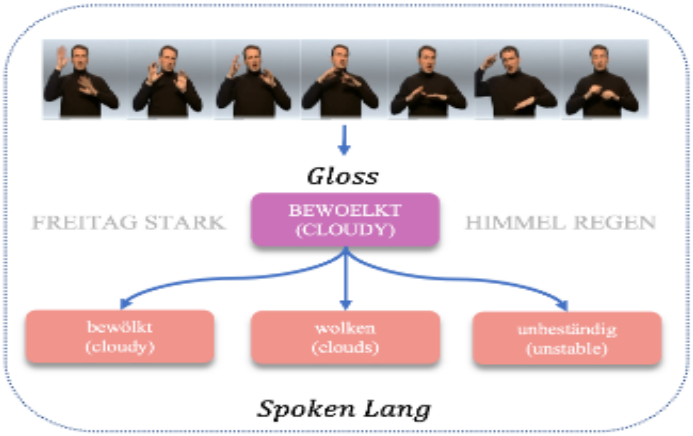
Dịch thuật ngôn ngữ ký hiệu từ video sang văn bản nói thường được chia thành hai giai đoạn chính: Sign2Gloss và Gloss2Text. Trong giai đoạn Sign2Gloss, các chú thích gloss được dự đoán từ video đầu vào, thiết lập mối liên kết giữa các biểu hiện thị giác và ý nghĩa tương ứng. Giai đoạn Gloss2Text tiếp theo sẽ dịch các chú thích gloss này thành ngôn ngữ nói.

Việc dịch thuật ngôn ngữ ký hiệu đối mặt với những thách thức độc đáo do ngữ pháp riêng biệt, sự tinh tế trong biểu hiện và sự biến thiên cao về hình thức thị giác giữa các người ký hiệu và bối cảnh khác nhau. Một thách thức lớn khác là sự khan hiếm dữ liệu - trong khi một mô hình dịch máy thần kinh (NMT) điển hình cần khoảng 1 triệu mẫu song song để huấn luyện hiệu quả, bộ dữ liệu ngôn ngữ ký hiệu hiện có nhỏ hơn nhiều bậc độ lớn. Ví dụ, bộ dữ liệu PHOENIX-2014T cho ngôn ngữ ký hiệu Đức chỉ có

8.257 cặp gloss-văn bản.

Tính Mơ Hồ trong Dịch Thuật Gloss

Một đặc điểm quan trọng của ngôn ngữ ký hiệu là tính mơ hồ trong dịch thuật gloss. Như được minh họa trong nghiên cứu, gloss "BEWOELKT (CLOUDY)" có thể được dịch thành nhiều cách khác nhau trong cùng một bộ dữ liệu. Tính mơ hồ này có thể chia sẻ cùng ý nghĩa nhưng khác về hình thức, chẳng hạn như "wolken (cloudy)", hoặc gloss có thể đại diện cho ý nghĩa khái niệm như "unbeständig (unstable)".



Hình 1.1 Ví dụ về sự mơ hồ khi dịch từ "BEWOELKT (CLOUDY)"

Phương Pháp Tiếp Cận Đề Xuất

Kiến Trúc Tổng Quan

Mục tiêu của hệ thống dịch gloss là chuyển đổi một chuỗi chú thích gloss G = $g_1, g_2, ..., g_T$ thành một chuỗi từ ngữ nói T = $t_1, t_2, ..., t_L$. Phương pháp này bao gồm việc tinh chỉnh các mô hình ngôn ngữ lớn được thiết kế đặc biệt cho nhiệm vụ này. Nghiên cứu sử dụng mô hình NLLB-200, một mô hình ngôn ngữ đa ngôn ngữ được Meta phát triển, được huấn luyện trên 200 ngôn ngữ với 3.6 tỷ câu từ các ngôn ngữ ít tài nguyên và 40.1 tỷ câu từ các ngôn ngữ nhiều tài nguyên.

Kỹ Thuật Tăng Cường Dữ Liệu

Để cải thiện độ bền vững của phương pháp dịch thuật cơ sở, nghiên cứu khám phá hai kỹ thuật tăng cường dữ liệu riêng biệt:

- Paraphrasing dịch câu đích gốc sang một ngôn ngữ trung gian (tiếng Anh) và sau đó dịch ngược về ngôn ngữ gốc (tiếng Đức). Chu trình này tạo ra sự đa dạng ngôn ngữ ở phía đích trong khi lý tưởng là vẫn bảo toàn ý nghĩa ban đầu trong các chú thích gloss.
- Back Translation bao gồm việc huấn luyện một mô hình dịch thuật ngược với dữ

liệu ngôn ngữ nói làm đầu vào, tạo ra chuỗi gloss tương ứng.

Kỹ Thuật Làm Mượt Nhãn Nhận Thức Ngữ Nghĩa (SALS)

Đóng góp chính của nghiên cứu là đề xuất hàm mất mát làm mượt nhãn mới được tối ưu hóa cho các tính mơ hồ cụ thể của dịch thuật gloss. Trong phương pháp làm mượt nhãn truyền thống, vector nhãn one-hot được thay thế bằng hỗn hợp của nó với phân phối đều. Tuy nhiên, phương pháp này gán xác suất khác không cho tất cả các từ trong từ vựng, bao gồm cả những từ không có trong từ vựng đích.

SALS đề xuất một vector xác suất mới trong đó đầu tiên đặt giá trị của các từ không phải đích về không. Trong số các từ trong từ vựng đích, họ tính toán độ tương tự ngữ nghĩa của mỗi từ với các từ khác trong từ vựng đích. Họ sử dụng FastText để tạo vector nhúng từ cho mỗi từ và sau đó tính độ tương tự cosine. Ba tình huống có thể xảy ra: đối với các từ trong ngôn ngữ đích có độ tương tự cao, họ sử dụng độ tương tự cosine; đối với các từ có độ tương tự ngữ nghĩa thấp nhưng có trong ngôn ngữ đích, họ áp dụng làm mượt nhãn tiêu chuẩn; các từ ngoài ngôn ngữ đích nhận được làm mượt bằng không.

Tối Ưu Hóa với LoRA

Để tối ưu hóa hiệu suất mô hình hơn nữa, nghiên cứu sử dụng kỹ thuật LoRA (Low-Rank Adaptation) để tinh chỉnh. Trong phương pháp này, họ đóng băng các trọng số mô hình gốc và chỉ huấn luyện bộ điều hợp ngôn ngữ ký hiệu cho nhiệm vụ đích. Điều này cho phép họ duy trì các chức năng ban đầu của LLM. Mô hình bộ điều hợp cuối cùng tiết kiệm bộ nhớ, chỉ chiếm khoảng 100 Megabyte không gian.

Thí Nghiệm và Kết Quả

Thiết Lập Thí Nghiệm

Nghiên cứu đánh giá phương pháp của họ trên bộ dữ liệu PHOENIX-2014T, tập trung vào các video ngôn ngữ ký hiệu Đức về dự báo thời tiết. Với 8.257 chuỗi chứa 1.066 gloss và 2.887 từ tiếng Đức, bộ dữ liệu này cung cấp một chuẩn mực cụ thể cho lĩnh vực để đánh giá các mô hình được tinh chỉnh bằng điểm BLEU.

Đối với kỹ thuật SALS, họ đặt ngưỡng độ tương tự cosine là 0.6 để đảm bảo chỉ xem xét các từ có độ tương tự ngữ nghĩa đủ cao. Họ cũng đặt $\beta = 0.1$. Đối với phương pháp cuối cùng, họ sử dụng NLLB với 3.3 tỷ tham số, kiến trúc bao gồm 24 lớp encoder và decoder.

Kết Quả Hiệu Suất

Kết quả cho thấy phương pháp của họ đạt hiệu suất vượt trội với cải thiện tương đối 3.75% trong BLEU-1 (1.98 điểm chênh lệch), 6.69% trong BLEU-4 (1.77 điểm chênh lệch), 5.38% trong ROUGE (2.78 điểm chênh lệch), và 2.07% trong CHRF++ (1.03 điểm

chênh lệch), với số lượng tham số có thể huấn luyện ít hơn đáng kể. Phương pháp của họ cũng cho thấy sự giảm hiệu suất trung bình chỉ 0.22 trên các chỉ số BLEU, ROUGE và CHRF++ khi chuyển từ tập phát triển sang tập kiểm tra.

	BLEU-1	BLEU-4
600M	52.7	26.5
1.3B	53.4	27.3
3.3B	53.1	27.1
3.3B+LoRA	53.8	27.5

Hình 1.2 Điểm số trên thang điểm BLEU của model với số lượng tham số khác nhau

Hệ thống NLLB-SALSloss cho thấy cải thiện trung bình 1.68 điểm trên tập phát triển và 1.06 điểm trên tập kiểm tra so với hệ thống NLLB-FineTuned trên các chỉ số BLEU, ROUGE và CHRF++. Khi so sánh với mô hình zero-shot mà không có tinh chỉnh, kết quả tỏ ra không tối ưu, nhấn mạnh vai trò chính của tinh chỉnh trong việc tối ưu hóa LLM cho dịch thuật Gloss2Text.

Phân Tích Chi Tiết

Phân tích so sánh với phương pháp trước đó cho thấy phương pháp này tạo ra các câu ngắn hơn so với các câu được tạo bởi phương pháp trước đó. Tỷ lệ tạo của phương pháp này là 0.9808, trong khi tỷ lệ của Chen et al. (2022b) là 1.0233.

Phương pháp này cho thấy cải thiện nhất quán trên hầu hết các nhóm tần suất so với phương pháp trước, cho thấy phương pháp này hiệu quả hơn trong việc dự đoán chính xác cả từ thông thường và từ hiếm. Tuy nhiên, phương pháp này vượt trội trong việc dự đoán các câu ngắn hơn nhưng tụt hậu ở các câu dài hơn.

Hạn Chế và Thách Thức

Hạn Chế của Biểu Diễn Gloss

Mặc dù gloss cung cấp chú thích có cấu trúc bổ sung cho video ngôn ngữ ký hiệu, chúng không nắm bắt đầy đủ sự phức tạp của giao tiếp ngôn ngữ ký hiệu. Biểu hiện khuôn mặt, là một phần của việc truyền đạt ý nghĩa trong ngôn ngữ ký hiệu, thường không được thể hiện trong các chú thích gloss. Ngoài ra, các cử chỉ liên quan đến việc chỉ vào các vị trí hoặc vật thể cụ thể thường bị bỏ qua trong biểu diễn gloss.

Chi Phí Tính Toán

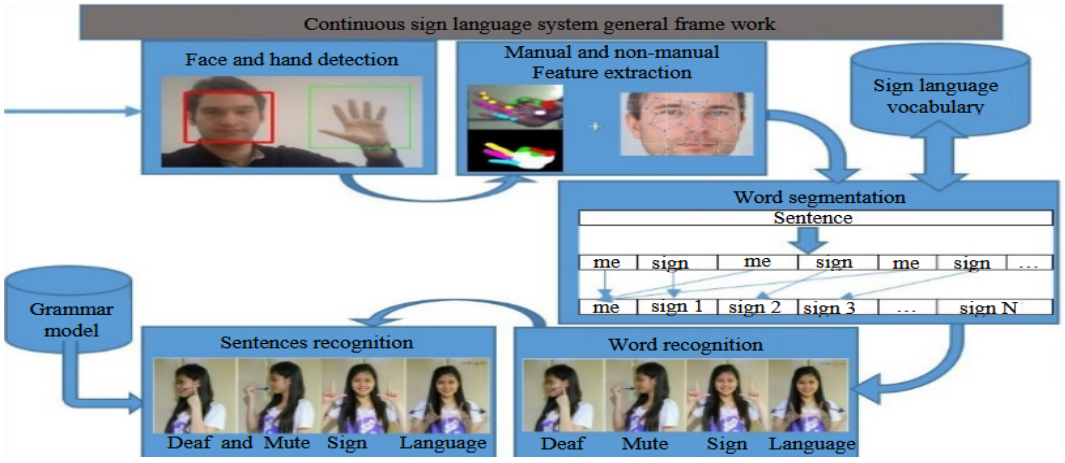
Một hạn chế khác là chi phí tính toán được giới thiệu bởi phương pháp này. Phương pháp yêu cầu tính toán độ tương tự giữa gloss và từ vựng đích, có thể làm chậm quá trình. Vấn đề này trở nên rõ ràng hơn khi kích thước từ vựng tăng, khiến giai đoạn huấn luyện vừa chậm hơn vừa tốn nhiều tài nguyên hơn.

Giới Hạn Về Miền Dữ Liệu

Cũng quan trọng là thừa nhận rằng các bộ dữ liệu được sử dụng trong ngôn ngữ ký hiệu thường có từ vựng cụ thể cho miền. Từ vựng này có thể không phản ánh các hoạt động và tương tác hàng ngày phổ biến trong cộng đồng người điếc. Điều này có thể giới hạn phạm vi và khả năng áp dụng của các hệ thống ngôn ngữ ký hiệu được phát triển bằng các bộ dữ liệu như vậy.

1.3 Tổng quan phương pháp tiếp cận

Trước đây, các hệ thống nhận diện ngôn ngữ ký hiệu chủ yếu dựa vào các thuật toán thị giác máy tính truyền thống, sử dụng các đặc trưng hình ảnh thủ công (handcrafted features) như hình dạng bàn tay, hướng tay, hoặc điểm nối cơ thể. Tuy nhiên, những phương pháp này gặp nhiều hạn chế về độ chính xác, khả năng mở rộng và khả năng thích nghi với các biến thể trong biểu hiện ngôn ngữ ký hiệu của từng người dùng khác nhau.[2]



Hình 1.3 Phương pháp tiếp cận dựa trên đặc trưng thủ công[2]

Ngày nay, với sự phát triển của học sâu và mô hình tuần tự như LSTM, các hệ thống nhận diện hiện đại có thể học được đặc trưng trực tiếp từ dữ liệu đầu vào, đặc biệt hiệu quả với dữ liệu dạng chuỗi thời gian như video. Mô hình LSTM nổi bật với khả năng lưu trữ và xử lý thông tin trong khoảng thời gian dài, giúp nhận diện chính xác các chuỗi động tác phức tạp và liên tục trong ngôn ngữ ký hiệu. Trong nghiên cứu này, chúng tôi lựa chọn tiếp cận sử dụng mô hình LSTM kết hợp với thư viện PyTorch để xây dựng hệ thống nhận diện ngôn ngữ ký hiệu tự động, với kỳ vọng cải thiện độ chính xác và tính linh hoạt so với các phương pháp trước đó.

1.4 Công cụ và kiến trúc sử dụng

Để hiện thực hóa hệ thống nhận diện, chúng tôi sử dụng thư viện PyTorch – một công cụ mã nguồn mở mạnh mẽ và linh hoạt cho phát triển các mô hình học sâu. PyTorch cho phép xây dựng mô hình một cách trực quan và dễ dàng kiểm soát trong quá trình huấn luyện, điều chỉnh siêu tham số, cũng như xử lý dữ liệu đầu vào.

Mô hình chính được sử dụng là LSTM, thuộc nhóm mạng nơ-ron hồi tiếp (Recurrent Neural Network – RNN), được thiết kế để xử lý chuỗi dữ liệu có tính thời gian như ngôn ngữ ký hiệu. Ngoài ra, hệ thống còn bao gồm các thành phần như bộ tiền xử lý (preprocessing), chia tách dữ liệu, huấn luyện, và đánh giá hiệu suất. Dữ liệu đầu vào có thể là chuỗi khung hình video được quay sẵn hoặc trực tiếp từ webcam, sau đó chuỗi tọa độ trên các điểm trên cơ thể sẽ được trích xuất bằng MediaPipe.

1.5 Mục tiêu của đồ án

Mục tiêu chính của đề tài là xây dựng một hệ thống nhận diện ngôn ngữ ký hiệu sử dụng mô hình học sâu LSTM, có khả năng chuyển đổi các cử chỉ tay thành văn bản tương ứng một cách tự động và chính xác. Đề tài tập trung vào việc áp dụng các kỹ thuật học sâu hiện đại trong xử lý chuỗi thời gian, đồng thời tận dụng sức mạnh của thư viện PyTorch để phát triển mô hình linh hoạt, dễ huấn luyện và mở rộng.

Bên cạnh đó, chúng tôi cũng đặt mục tiêu đánh giá hiệu quả của mô hình trong các điều kiện khác nhau, từ dữ liệu huấn luyện giới hạn đến việc thử nghiệm trên các biểu hiện ký hiệu không chuẩn hóa. Kết quả của đề tài kỳ vọng sẽ đóng góp vào kho nghiên cứu ứng dụng trí tuệ nhân tạo trong lĩnh vực hỗ trợ người khuyết tật, đồng thời là cơ sở để phát triển các hệ thống phiên dịch ngôn ngữ ký hiệu thời gian thực trong tương lai.

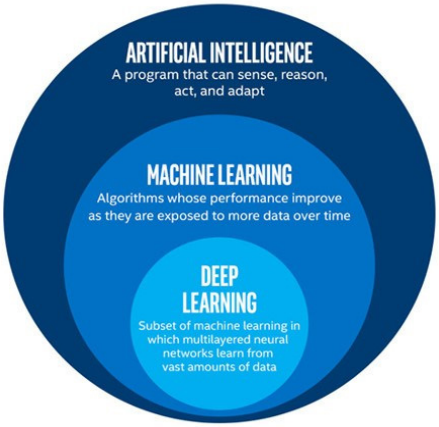
CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

Chương này đã trình bày nền tảng lý thuyết cho đề tài, bao gồm giới thiệu về học sâu, mô hình mạng LSTM, cơ chế trích xuất đặc trưng từ cử chỉ tay, và ứng dụng xử lý chuỗi trong xử lý ngôn ngữ tự nhiên. Những kiến thức này tạo cơ sở cho việc thiết kế và triển khai hệ thống nhận diện ngôn ngữ ký hiệu mà nhóm đề xuất.

2.1 Giới thiệu về Deep Learning

Trong những năm vừa qua, cùng với sự bùng nổ của cuộc cách mạng công nghiệp 4.0, các thuật ngữ như trí tuệ nhân tạo (AI), học máy (Machine Learning) và học sâu (Deep Learning) đang dần trở nên phổ biến và trở thành những khái niệm mà các công dân của kỷ nguyên 4.0 buộc phải nắm được.

Có thể giải thích mối liên hệ giữa 3 khái niệm này bằng cách tưởng tượng chúng như những vòng tròn, trong đó AI – ý tưởng xuất hiện sớm nhất – là vòng tròn lớn nhất, tiếp đến là machine learning – khái niệm xuất hiện sau, và cuối cùng là deep learning – thứ đang thúc đẩy sự bùng phát của AI hiện nay – là vòng tròn nhỏ nhất.



Hình 2.4 Mối liên hệ giữa AI, Machine Learning và Deep Learning

Khái niệm cơ bản về trí tuệ nhân tạo

AI có thể được định nghĩa như một ngành của khoa học máy tính liên quan đến việc tự động hóa các hành vi thông minh. AI là một bộ phận của khoa học máy tính và do đó nó phải được đặt trên những nguyên lý lý thuyết vững chắc, có khả năng ứng dụng được của lĩnh vực này. Nói nôm na cho dễ hiểu: đó là trí tuệ của máy móc được tạo ra bởi con người. Trí tuệ này có thể tư duy, suy nghĩ, học hỏi,... như trí tuệ con người. Xử lý dữ liệu ở rộng hơn tầm, quy mô, hệ thống, khoa học và nhanh hơn so với con người. Khái niệm cơ bản về học máy

Khái niệm cơ bản về học máy

Machine Learning là một thuật ngữ rộng để chỉ hành động bạn dạy máy tính cải thiện một nhiệm vụ nào đó đang thực hiện. Cụ thể hơn, Machine Learning đề cập tới bất kỳ hệ thống mà hiệu suất của máy tính khi thực hiện một nhiệm vụ sẽ trở nên tốt hơn sau khi hoàn thành nhiệm vụ đó nhiều lần. Hay nói cách khác, khả năng cơ bản nhất của Machine Learning là sử dụng thuật toán để phân tích thông tin có sẵn, học hỏi từ nó rồi đưa ra quyết định hoặc dự đoán về một đối tượng có dữ liệu liên quan. Thay vì tạo ra một phần mềm với những hành động, hướng dẫn chi tiết để thực hiện một nhiệm vụ cụ thể, máy tính được "huấn luyện" bằng cách sử dụng lượng dữ liệu và các thuật toán để học cách thực hiện nhiệm vụ.

Ví dụ: Với dự án một loại Machine Learning, giả sử bạn muốn một chương trình có thể xác định được mèo trong các bức ảnh:

- Đầu tiên, bạn cung cấp cho AI một tập hợp các đặc điểm của loài mèo để máy nhận dạng, ví dụ như sắc lông, hình dáng, cơ thể, kích thước, lông mũi, hình tai...
- Tiếp theo, bạn cung cấp một số hình ảnh cho AI, trong đó một số hoặc tất cả có hình mèo, thậm chí có cả ảnh "méo" để máy có thể cho hiệu quả hơn các chi tiết, đặc điểm để diễn đoán hình mèo.
- Sau khi máy đã được cho dữ liệu cần thiết về mèo, nó phải biết cách tìm ra con mèo trong một bức tranh – "Nếu trong hình ảnh có chứa các chi tiết như X, Y, hoặc Z nào đó, thì 95% khả năng đó là một con mèo".

Khái niệm học sâu

Deep Learning là một phương thức trong lĩnh vực trí tuệ nhân tạo (AI), được sử dụng để dạy cho máy tính xử lý dữ liệu một cách được lấy cảm hứng từ con người. Mô hình học sâu có thể nhận diện nhiều hình mẫu phức tạp trong hình ảnh, văn bản, âm thanh và các dữ liệu khác để tạo ra thông tin chuyên sâu và dự đoán chính xác.

Ví dụ: Khi huấn luyện Deep Learning cho việc nhận diện mèo, cung cấp cho nó dữ hình ảnh cần thiết về loài mèo, và nó sẽ tự mình hình dung, tự học. Các bước cần làm như sau:

- Cung cấp cho máy rất nhiều ảnh về mèo.
- Thuật toán sẽ kiểm tra ảnh để xem các đặc điểm, chi tiết chung giữa các bức ảnh.
- Mỗi bức ảnh sẽ được giải mã chi tiết dưới nhiều cấp độ, từ các hình dạng lớn, chung đến các ô nhỏ và nhỏ hơn nữa. Nếu một hình dạng hoặc các đường được lặp lại nhiều lần, thuật toán sẽ ghi nhận nó như là một đặc tính quan trọng.

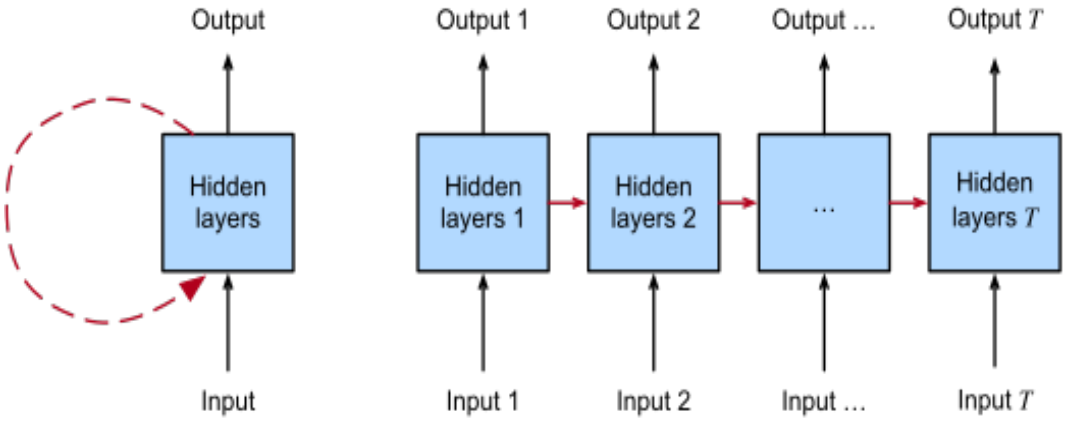
- Sau khi phân tích đủ hình ảnh cần thiết, thuật toán giờ đây sẽ biết được các mẫu nào cung cấp bằng chứng rõ ràng nhất về mèo và tất cả những gì con người phải làm chỉ là cung cấp các dữ liệu thô.

Tóm lại: Deep Learning là loại machine learning mà trong đó máy tự đào tạo chính nó. Deep Learning đòi hỏi rất nhiều dữ liệu đầu vào và sức mạnh tính toán hơn là Machine Learning, nhưng nó đã bắt đầu được triển khai bởi các công ty công nghệ lớn như Facebook, Amazon. Trong đó, một trong những cái tên nổi tiếng nhất về Machine Learning là AlphaGo, một máy tính có thể chơi cờ vây với chính bản thân mà cho đến khi nó có thể dự đoán những đường đi nước bước chính xác nhất để dễ đánh bại nhiều nhà vô địch trên thế giới.

2.2 Mạng RNN

Cho đến nay, các mô hình học sâu chủ yếu xử lý dữ liệu có độ dài cố định như dữ liệu dạng bảng (tabular) hay hình ảnh cố định kích thước (ví dụ: Fashion-MNIST). Tuy nhiên, nhiều tác vụ trong thực tế yêu cầu xử lý dữ liệu tuần tự (sequence data) như: phân tích video, dịch máy, tạo chú thích ảnh, tổng hợp giọng nói, hay điều khiển robot. Những tác vụ này cần mô hình vừa đọc, vừa sinh ra chuỗi dữ liệu đầu ra.

RNN (Recurrent Neural Network) là một kiến trúc học sâu giúp mô hình hóa động lực học của chuỗi dữ liệu bằng các kết nối hồi tiếp. Mặc dù nhìn có vẻ mâu thuẫn với bản chất mạng truyền thẳng, nhưng khi "trải" RNN theo thời gian, ta thấy đó là mạng truyền thẳng với tham số được chia sẻ tại mỗi bước thời gian. Điều này cho phép RNN giữ thông tin từ các bước trước để xử lý các bước sau.



Hình 2.5 Mạng RNN

RNN có lịch sử lâu đời, bắt đầu từ các mô hình mô phỏng não bộ, và trở nên phổ biến nhờ thành công trong các bài toán như nhận diện chữ viết tay, dịch máy, chẩn đoán y tế... Dù hiện tại đã nhường sân chơi cho Transformer, RNN vẫn giữ vai trò quan trọng

trong bài toán mô hình hóa chuỗi.

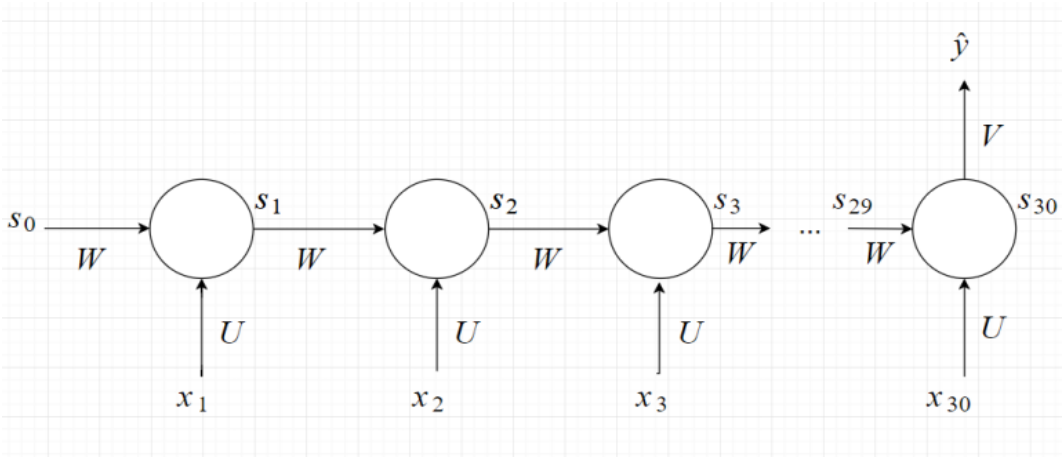
Một phát hiện quan trọng thúc đẩy sự phát triển của mô hình chuỗi là: dù đầu vào và đầu ra không thể biểu diễn bằng vector độ dài cố định, chúng vẫn có thể biểu diễn bằng chuỗi các vector độ dài cố định (ví dụ: văn bản = chuỗi từ; video = chuỗi ảnh...).

2.3 Mạng LSTM

RNN là một loại mạng nơ-ron đặc biệt thích hợp với dữ liệu chuỗi như văn bản, âm thanh hoặc chuyển động tay theo thời gian. Khác với mạng nơ-ron truyền thống (feedforward neural networks), RNN có khả năng duy trì trạng thái theo thời gian nhờ cơ chế lan truyền thông tin qua các bước thời gian.

Tuy nhiên, RNN truyền thống gặp khó khăn trong việc ghi nhớ thông tin dài hạn do hiện tượng "biến mất đạo hàm" (vanishing gradient). Để khắc phục điều này, Hochreiter và Schmidhuber đã đề xuất mô hình Long Short-Term Memory (LSTM)[3]. LSTM sử dụng các cổng (gates) để điều khiển dòng thông tin đi qua các trạng thái ẩn, từ đó cải thiện đáng kể khả năng ghi nhớ và học các phụ thuộc dài hạn.

Trong đề tài của chúng tôi, RNN có thể mang thông tin của frame (ảnh) từ state trước tới các state sau, rồi ở state cuối là sự kết hợp của tất cả các ảnh để dự đoán hành động trong video.



Hình 2.6 Mô hình RNN

Đạo hàm của L với W ở state thứ i: $\frac{\partial L}{\partial W} = \frac{\partial L}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial s_{30}} * \frac{\partial s_{30}}{\partial s_i} * \frac{\partial s_i}{\partial W}$, trong đó $\frac{\partial s_{30}}{\partial s_i} = \prod_{j=i}^{29} \frac{\partial s_{j+1}}{\partial s_j}$

Giả sử activation là tanh function, $s_t = \tanh(Ux_t + Ws_{t-1})$

$\frac{\partial s_t}{\partial s_{t-1}} = (1 - s_t^2) * W \Rightarrow \frac{\partial s_{30}}{\partial s_i} = W^{30-i} * \prod_{j=i}^{29}(1 - s_j^2)$

Ta có $s_j < 1, W < 1 \Rightarrow$ Ở những state xa thì $\frac{\partial s_{30}}{\partial s_i} \approx 0$ hay $\frac{\partial L}{\partial W} \approx 0$, hiện tượng vanishing gradient.

Ta có thể thấy là các state càng xa ở trước đó thì càng bị vanishing gradient và các hệ số không được update với các frame ở xa. Hay nói cách khác là RNN không học được từ các thông tin ở trước đó xa do vanishing gradient.

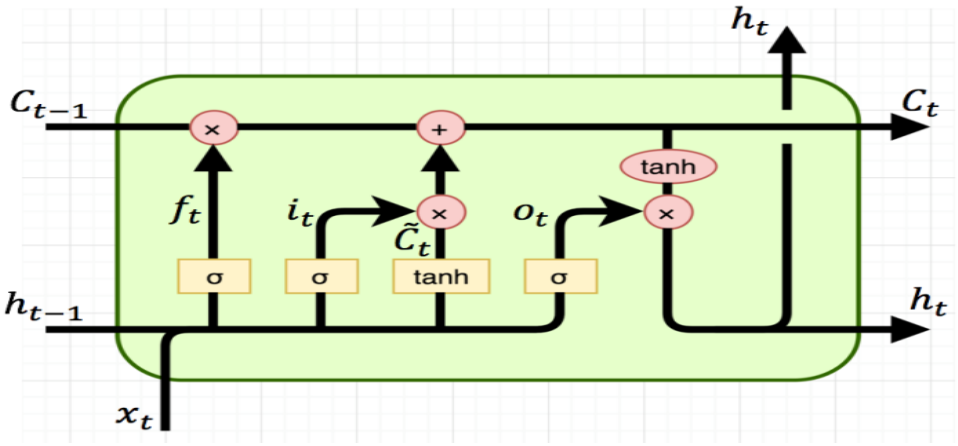
Như vậy về lý thuyết là RNN có thể mang thông tin từ các layer trước đến các layer sau, nhưng thực tế là thông tin chỉ mang được qua một số lượng state nhất định, sau đó thì sẽ bị vanishing gradient, hay nói cách khác là model chỉ học được từ các state gần nó => short term memory.

Thử lấy ví dụ về short term memory: Bài toán là dự đoán từ tiếp theo trong đoạn văn. Đoạn đầu tiên "Mặt trời mọc ở hướng . . . ", ta có thể chỉ sử dụng các từ trước trong câu để đoán là đông. Tuy nhiên, với đoạn, "Tôi là người Việt Nam. Tôi đang sống ở nước ngoài. Tôi có thể nói trôi chảy tiếng . . . " thì rõ ràng là chỉ sử dụng từ trong câu đấy hoặc câu trước là không thể dự đoán được từ cần điền là Việt. Ta cần các thông tin từ state ở trước đó rất xa => cần long term memory điều mà RNN không làm được => Cần một mô hình mới để giải quyết vấn đề này => Long short term memory (LSTM) ra đời.

Mô hình LSTM

Ở state thứ t của mô hình LSTM:

- Output: c_t, h_t, ta gọi c là cell state, h là hidden state.
- Input: c_{t-1}, h_{t-1}, x_t Trong đó x_t là input ở state thứ t của model. c_{t-1}, h_{t-1} là output của layer trước. h đóng vai trò khá giống như s ở RNN, trong khi c là điểm mới của LSTM.



Hình 2.7 Mô hình LSTM

f_t, i_t, o_t tương ứng với forget gate, input gate và output gate.

- Forget gate: $f_t = \sigma(U_f * x_t + W_f * h_{t-1} + b_f)$

- Input gate: $i_t = \sigma(U_i * x_t + W_i * h_{t-1} + b_i)$
- Output gate: $o_t = \sigma(U_o * x_t + W_o * h_{t-1} + b_o)$

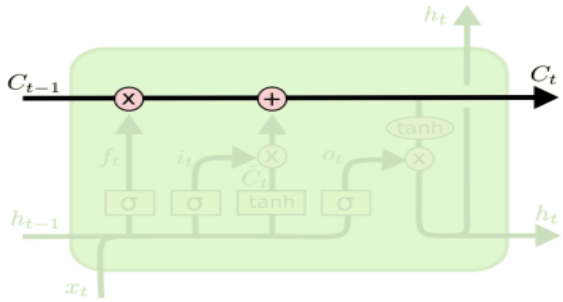
Nhận xét: $0 < f_t, i_t, o_t < 1; b_f, b_i, b_o$ là các hệ số bias; hệ số W, U giống như trong RNN.

$\tilde{c}_t = \tanh(U_c * x_t + W_c * h_{t-1} + b_c)$, bước này giống như tính s_t trong RNN.

$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t$, forget gate quyết định xem cần lấy bao nhiêu từ cell state trước và input gate sẽ quyết định lấy bao nhiêu từ input của state và hidden layer của layer trước.

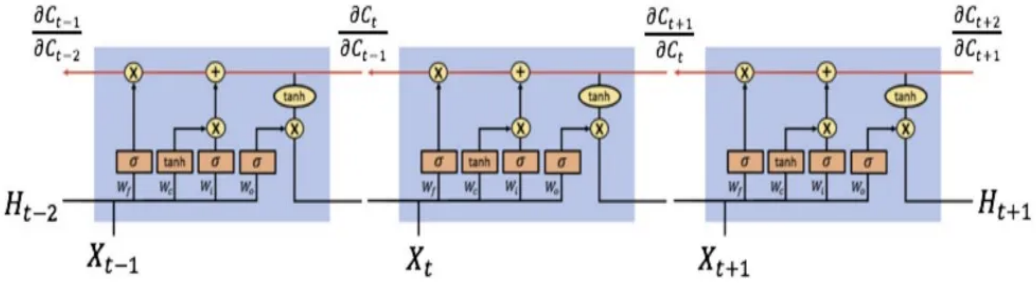
$h_t = o_t * \tanh(c_t)$, output gate quyết định xem cần lấy bao nhiêu từ cell state để trở thành output của hidden state. Ngoài ra h_t cũng được dùng để tính ra output y_t cho state t.

Nhận xét: $h_t, \tilde{c}_t$ khá giống với RNN, nên model có short term memory. Trong khi đó c_t giống như một băng chuyền ở trên mô hình RNN vậy, thông tin nào cần quan trọng và dùng ở sau sẽ được gửi vào và dùng khi cần => có thể mang thông tin từ đi xa => long term memory. Do đó mô hình LSTM có cả short term memory và long term memory.



Hình 2.8 Cell state trong LSTM

LSTM chống vanishing gradient



Hình 2.9 LSTM chống vanishing gradient

Ta cũng áp dụng thuật toán back propagation through time cho LSTM tương tự như

RNN.

Thành phần chính gây là vanishing gradient trong RNN là $\frac{\partial s_{t+1}}{\partial s_t} = (1 - s_t^2) * W$, trong đó s_t,W<1.

Tương tự trong LSTM ta quan tâm đến $\frac{\partial c_t}{\partial c_{t-1}} = f_t$. Do $0 < f_t < 1$ nên về cơ bản thì LSTM vẫn bị vanishing gradient nhưng bị ít hơn so với RNN. Hơn thế nữa, khi mang thông tin trên cell state thì ít khi cần phải quên giá trị cell cũ, nên $f_t \approx 1 \Rightarrow$ Tránh được vanishing gradient.

Do đó LSTM được dùng phổ biến hơn RNN cho các toán thông tin dạng chuỗi.[4]

2.4 Mạng GRU

GRU là một biến thể gọn hơn của LSTM, thường có chất lượng tương đương và tính toán nhanh hơn đáng kể. [5]

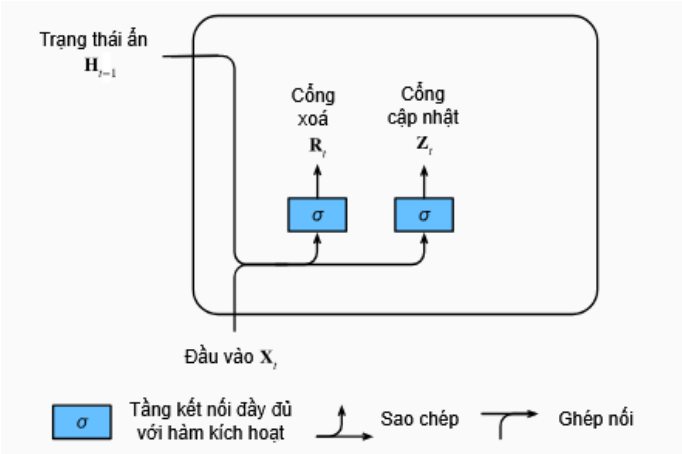
Kiểm soát Trạng thái Ẩn

Sự khác biệt chính giữa RNN thông thường và GRU là GRU hỗ trợ việc kiểm soát trạng thái ẩn. Điều này có nghĩa là ta có các cơ chế được học để quyết định khi nào nên cập nhật và khi nào nên xóa trạng thái ẩn. Ví dụ, nếu ký tự đầu tiên có mức độ quan trọng cao, mô hình sẽ học để không cập nhật trạng thái ẩn sau lần quan sát đầu tiên. Tương tự, mô hình sẽ học cách bỏ qua những quan sát tạm thời không liên quan, cũng như cách xóa trạng thái ẩn khi cần thiết.

Cổng Xóa và Cổng Cập nhật

Đầu tiên ta giới thiệu cổng xóa và cổng cập nhật. Ta thiết kế chúng thành các vector có các phần tử trong khoảng (0,1) để có thể biểu diễn các tổ hợp lồi. Chẳng hạn, một biến xóa cho phép kiểm soát bao nhiêu phần của trạng thái trước đây được giữ lại. Tương tự, một biến cập nhật cho phép kiểm soát bao nhiêu phần của trạng thái mới sẽ giống trạng thái cũ.

Ta bắt đầu bằng việc thiết kế các cổng tạo ra các biến này. hình dưới minh họa các đầu vào cho cả cổng xóa và cổng cập nhật trong GRU, với đầu vào ở bước thời gian hiện tại X_t và trạng thái ẩn ở bước thời gian trước đó H_{t-1}. Đầu ra được tạo bởi một tầng kết nối đầy đủ với hàm kích hoạt sigmoid.



Hình 2.10 Cổng xóa và cổng cập nhật trong GRU

Tại bước thời gian t, với đầu vào minibatch là $X_t \in R^{n*d}$ (số lượng mẫu: n, số lượng đầu vào: d) và trạng thái ẩn ở bước thời gian gần nhất là $H_{t-1} \in R^{n*h}$ (số lượng trạng thái ẩn: h), cổng xóa $R_t \in R^{n*h}$ và cổng cập nhật $Z_t \in R^{n*h}$ được tính như sau:

- $R_t = \sigma(X_t * W_{xr} + H_{t-1} * W_{hr} + b_r)$
- $Z_t = \sigma(X_t * W_{xz} + H_{t-1} * W_{hz} + b_z)$

Ở đây, $W_{xr}, W_{xz} \in R^{d*h}$ và $W_{hr}, W_{hz} \in R^{d*h}$ là các tham số trọng số và $b_r, b_z \in R_{1*h}$ là các hệ số điều chỉnh. Ta sẽ sử dụng hàm sigmoid để biến đổi các giá trị đầu vào nằm trong khoảng (0,1).

Hoạt động của Cổng Xóa

Ta bắt đầu bằng việc tích hợp cổng xóa với một cơ chế cập nhật trạng thái tiềm ẩn thông thường. Trong RNN thông thường, ta cập nhật trạng thái ẩn theo công thức:

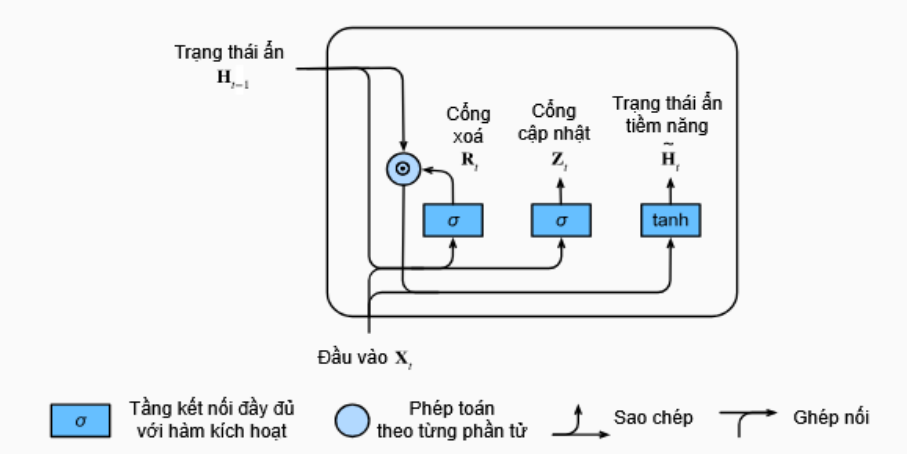
- $H_t = \tanh(X_t W_{xh} + H_{t-1} W_{hh} + b_h)$

Nếu các phần tử trong cổng xóa R_t có giá trị gần với 1, kết quả sẽ giống với RNN thông thường. Nếu tất cả các phần tử của R_t gần với 0, trạng thái ẩn sẽ là đầu ra của một perceptron đa tầng với đầu vào là X_t. Bất kỳ trạng thái ẩn nào tồn tại trước đó đều được đặt lại về giá trị mặc định. Tại đây nó được gọi là trạng thái ẩn tiềm năng, và chỉ là tiềm năng vì ta vẫn cần kết hợp thêm đầu ra của cổng cập nhật.

Trạng thái ẩn tiềm năng được tính như sau:

- $H_t = \tanh(X_t W_{xh} + (R_t \odot H_{t-1}) W_{hh} + b_h)$

trong đó ký hiệu $\odot$ biểu thị phép nhân theo từng phần tử (element-wise) giữa các tensor.

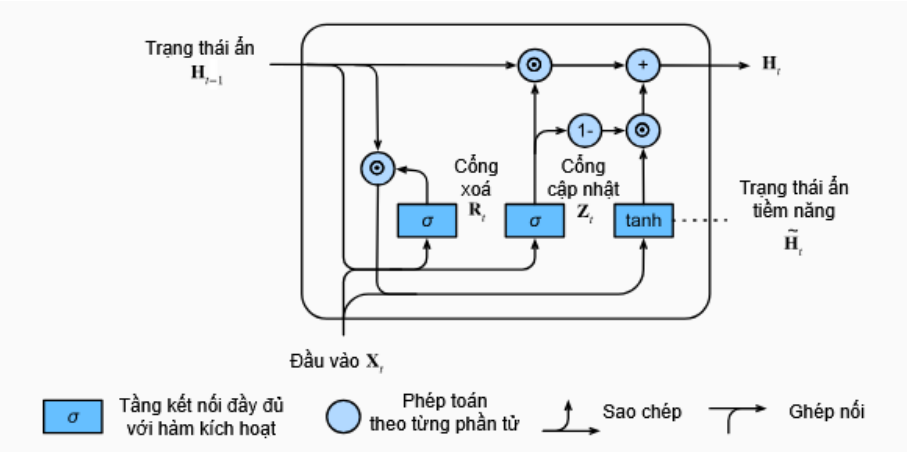


Hình 2.11 Tính toán của trạng thái ẩn tiềm năng trong một GRU

Hoạt động của Cổng Cập nhật

Tiếp theo ta sẽ kết hợp hiệu ứng của cổng cập nhật Z_t như trong hình dưới. Cổng này xác định mức độ giống nhau giữa trạng thái mới H_t và trạng thái cũ H_{t-1}, cũng như mức độ trạng thái ẩn tiềm năng $\tilde{H}_t$ được sử dụng. Biến cổng (gating variable) Z_t được sử dụng cho mục đích này, bằng cách áp dụng tổ hợp lồi giữa trạng thái cũ và trạng thái tiềm năng. Ta có phương trình cập nhật cuối cùng cho GRU.

- $H_t = Z_t \odot H_{t-1} + (1 - Z_t) \odot \tilde{H}_t$



Hình 2.12 Tính toán trạng thái ẩn trong GRU

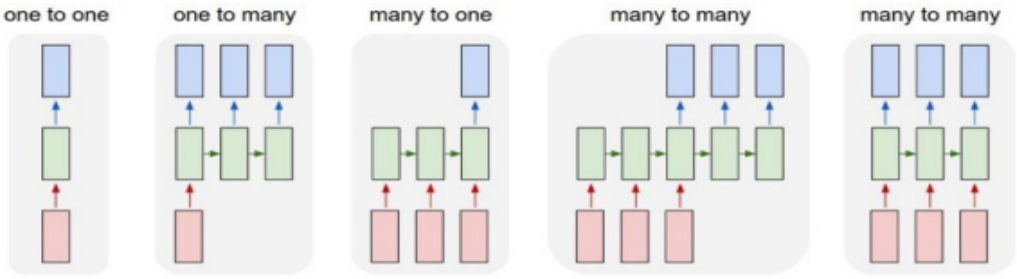
Nếu các giá trị trong cổng cập nhật Z_t bằng 1, chúng ta chỉ đơn giản giữ lại trạng thái cũ. Trong trường hợp này, thông tin từ X_t về cơ bản được bỏ qua, tương đương với

việc bỏ qua bước thời gian t trong chuỗi phụ thuộc. Ngược lại, nếu Z_t gần giá trị 0, trạng thái ẩn H_t sẽ gần với trạng thái ẩn tiềm năng $\tilde{H}_t$. Những thiết kế trên có thể giúp chúng ta giải quyết vấn đề tiêu biến gradient trong các mạng RNN và nắm bắt tốt hơn sự phụ thuộc xa trong chuỗi thời gian. Tóm lại, các mạng GRU có hai tính chất nổi bật sau:

- Cổng xóa giúp nắm bắt các phụ thuộc ngắn hạn trong chuỗi thời gian.
- Cổng cập nhật giúp nắm bắt các phụ thuộc dài hạn trong chuỗi thời gian.

2.5 Mạng Seq2Seq

Mô hình RNN ra đời để xử lý các dữ liệu dạng chuỗi (sequence) như text, video.



Hình 2.13 Các dạng bài toán RNN

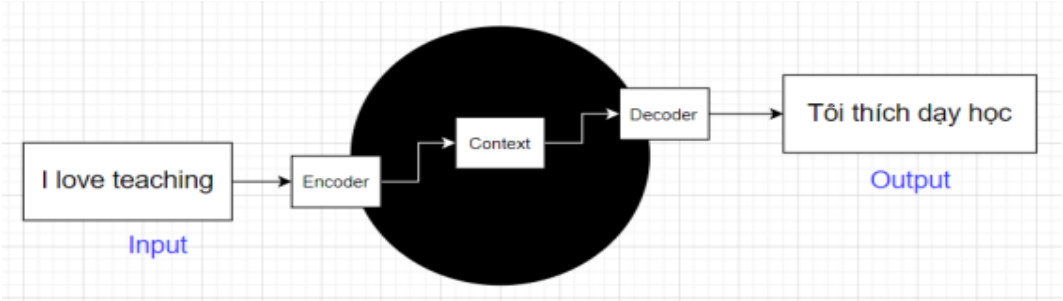
Bài toán RNN được phân làm một số dạng:

- One to one: mẫu bài toán cho Neural Network (NN) và Convolutional Neural Network (CNN), 1 input và 1 output, ví dụ với bài toán phân loại ảnh MNIST input là ảnh và output ảnh đấy là số nào.
- One to many: bài toán có 1 input nhưng nhiều output, ví dụ với bài toán caption cho ảnh, input là 1 ảnh nhưng output là nhiều chữ mô tả cho ảnh, dưới dạng một câu.
- Many to one: bài toán có nhiều input nhưng chỉ có 1 output, ví dụ bài toán phân loại hành động trong video, input là nhiều ảnh (frame) tách ra từ video, ouptut là hành động trong video.
- Many to many: bài toán có nhiều input và nhiều output, ví dụ bài toán dịch từ tiếng anh sang tiếng việt, input là 1 câu gồm nhiều chữ: "I love Vietnam" và output cũng là 1 câu gồm nhiều chữ "Tôi yêu Việt Nam". Độ dài sequence của input và output có thể khác nhau.

Mô hình Seq2Seq sinh ra để giải quyết bài toán many to many và rất hiệu quả trong các bài toán: dịch, tóm tắt đoạn văn.

Mô hình seq2seq

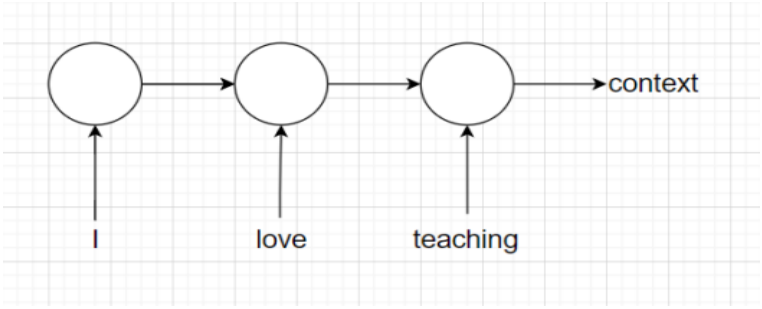
Input của mô hình seq2seq là một câu tiếng anh và output là câu dịch tiếng việt tương ứng, độ dài hai câu này có thể khác nhau. Ví dụ: input: I love teaching -> output: Tôi thích dạy học, input 1 câu 3 từ, output 1 câu 4 từ.



Hình 2.14 Seq2Seq Model

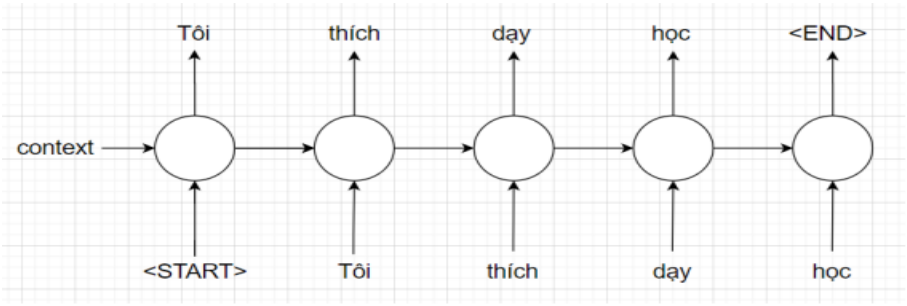
Mô hình seq2seq gồm 2 thành phần là encoder và decoder.

Encoder nhận input là câu tiếng anh và output ra context vector, còn decoder nhận input là context vector và output ra câu tiếng việt tương ứng. Phần encoder sử dụng mô hình RNN (hoặc có thể là các mô hình cải tiến như GRU, LSTM) và context vector được dùng là hidden states ở node cuối cùng. Phần decoder cũng là một mô hình RNN với s_0 chính là context vector rồi dần dần sinh ra các từ ở câu dịch.



Hình 2.15 Mô hình Encoder

Các từ trong câu tiếng anh sẽ được embedding thành vector và cho vào mô hình RNN, hidden state ở node cuối cùng sẽ được dùng làm context vector. Về mặt lý thuyết thì context vector sẽ mang đủ thông tin của câu tiếng anh cần dịch và sẽ được làm input cho decoder.



Hình 2.16 Mô hình Decoder

2 tag <SOS> và <EOS> được thêm vào câu output để chỉ từ bắt đầu và kết thúc của câu dịch. Mô hình decoder nhận input là context vector. Ở node đầu tiên context vector và tag <SOS> sẽ output ra chữ đầu tiên trong câu dịch, rồi tiếp tục mô hình sinh chữ tiếp theo cho đến khi gặp tag <EOS> hoặc đến max_length của câu output thì dừng lại.

2.6 Trích xuất tọa độ với MediaPipe Hands

Trước khi đưa dữ liệu vào mạng LSTM, hệ thống cần trích xuất tọa độ từ video đầu vào. Nhóm sử dụng MediaPipe Hands – một thư viện mã nguồn mở của Google – để phát hiện và theo dõi tay trong thời gian thực. Mỗi khung hình đầu ra là một tập hợp gồm 21 điểm mốc với tọa độ 3 chiều (x, y, z), đại diện cho các khớp và điểm đặc trưng trên mỗi bàn tay trái phải, và 33 điểm mốc với 4 thông số (x, y, z, visibility) đại điện cho các điểm đặc trưng trên cơ thể.

Chuỗi điểm mốc này tạo thành dữ liệu đầu vào động theo thời gian (temporal data), phù hợp với mô hình RNN/LSTM. Việc trích xuất đặc trưng này giúp giảm độ phức tạp so với xử lý hình ảnh thô toàn bộ, đồng thời tăng tính tổng quát của mô hình.

2.7 Xử lý chuỗi và ngôn ngữ tự nhiên

Sau khi nhận diện được các từ riêng lẻ từ từng chuỗi động tác tay, bài toán được chuyển thành một bài toán xử lý chuỗi – trong đó hệ thống cần ghép các từ theo đúng ngữ pháp và ngữ nghĩa để tạo thành câu hoàn chỉnh. Ở giai đoạn này, nhóm sử dụng mô hình Seq2Seq gồm hai thành phần chính: Encoder và Decoder, cả hai đều được xây dựng dựa trên GRU. Encoder có nhiệm vụ tiếp nhận chuỗi các từ đã nhận diện và nén thông tin vào một vector ngữ cảnh (context vector). Sau đó, Decoder sử dụng vector này để sinh ra từng từ trong câu đầu ra, theo cách tuần tự.

Mô hình này giúp hệ thống chuyển đổi các chuỗi hành động tay thành câu văn hoàn chỉnh, tăng tính ứng dụng trong các tình huống giao tiếp hoặc hỗ trợ người dùng trong đời sống thực tế.

2.8 Framework và công cụ triển khai

Dự án được triển khai chủ yếu bằng ngôn ngữ Python, sử dụng thư viện PyTorch – một trong những framework học sâu phổ biến và linh hoạt hiện nay. PyTorch cung cấp API rõ ràng và hỗ trợ GPU tốt, giúp đẩy nhanh quá trình huấn luyện mô hình.

Ngoài ra, nhóm sử dụng thư viện MediaPipe cho việc trích xuất đặc trưng tay, NumPy và Pandas cho xử lý dữ liệu, Matplotlib để trực quan hóa và các tiện ích như scikit-learn cho đánh giá mô hình.

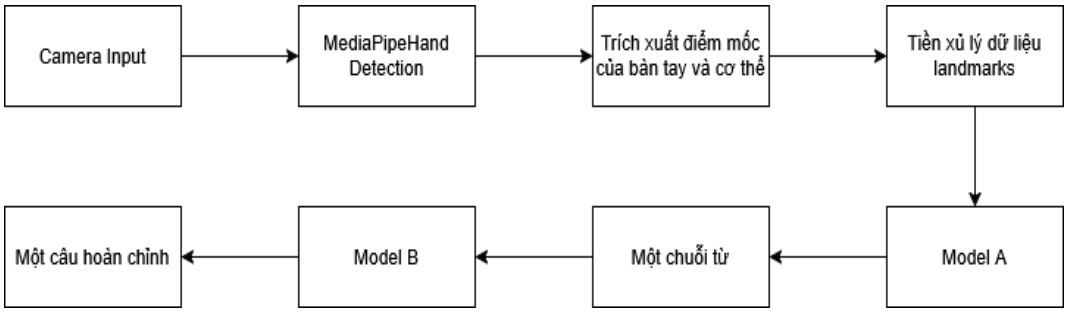
2.9 Phương pháp tiếp cận bài toán

Trong đề tài này, nhóm lựa chọn cách tiếp cận bài toán nhận diện ngôn ngữ ký hiệu thông qua hai giai đoạn chính: (1) nhận diện và trích tọa độ từ hình ảnh, và (2) xử lý chuỗi và suy diễn câu hoàn chỉnh bằng mô hình học sâu.

Sau khi thu được chuỗi landmark từ một chuỗi video ngắn hoặc tập hợp các hình ảnh, nhóm tiến hành tiền xử lý dữ liệu và ánh xạ từng chuỗi landmark tương ứng với một từ trong ngôn ngữ ký hiệu. Các từ này có thể là các hành động rời rạc, như "tôi", "yêu", "bạn", được biểu diễn qua một đoạn chuyển động tay ngắn.

Tiếp theo, để kết nối các từ riêng lẻ thành một câu hoàn chỉnh có nghĩa, nhóm sử dụng một mạng LSTM (Long Short-Term Memory). Mô hình LSTM có khả năng ghi nhớ mối liên hệ ngữ cảnh giữa các từ theo thời gian, từ đó giúp hệ thống hiểu được cấu trúc và trật tự câu trong ngôn ngữ tự nhiên. Nhờ vậy, mô hình không chỉ dừng lại ở việc nhận diện từ đơn lẻ, mà còn có thể tái tạo lại một câu đầy đủ và hợp ngữ pháp từ chuỗi các động tác ký hiệu.

Phương pháp này giúp giảm thiểu độ phức tạp so với nhận diện trực tiếp toàn bộ câu ký hiệu, đồng thời tận dụng được lợi thế của LSTM trong xử lý chuỗi. Cách tiếp cận này cũng mở ra khả năng mở rộng hệ thống về sau, khi cần tăng số lượng từ vựng hoặc cải thiện độ chính xác của câu đầu ra.



Hình 2.17 Pipeline hệ thống

CHƯƠNG 3. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

Chương này sẽ trình bày chi tiết thiết kế hệ thống phiên dịch ngôn ngữ ký hiệu thành văn bản, gồm hai mô hình chính: Model A dùng để nhận diện từ từ cử chỉ tay, và Model B dùng để xây dựng câu hoàn chỉnh từ chuỗi từ đã nhận diện. Thiết kế hai tầng giúp tăng khả năng mở rộng, quản lý mô hình hiệu quả, đồng thời phù hợp với cách tiếp cận học sâu hiện đại trong xử lý ngôn ngữ và chuỗi thời gian.

3.1 Mục tiêu thiết kế

Mục tiêu của hệ thống là xây dựng một mô hình có khả năng phiên dịch ngôn ngữ ký hiệu thành câu hoàn chỉnh bằng văn bản, từ đó hỗ trợ người khiếm thính trong giao tiếp với cộng đồng. Để giải quyết bài toán này, nhóm chia hệ thống thành hai giai đoạn xử lý chính, tương ứng với hai mô hình riêng biệt:

- Mô hình A: nhận diện từng từ hoặc kí tự trong ngôn ngữ ký hiệu thông qua trích xuất đặc trưng từ cử chỉ tay.
- Mô hình B: xử lý chuỗi các từ đầu ra từ mô hình A để xây dựng câu hoàn chỉnh dựa trên ngữ cảnh và trật tự thời gian.

Việc tách biệt hai mô hình giúp dễ quản lý dữ liệu, huấn luyện mô hình hiệu quả hơn, và có thể nâng cấp từng phần riêng lẻ trong tương lai.

3.2 Kiến trúc tổng thể hệ thống

Hệ thống được thiết kế thành ba tầng chức năng chính:

- Tầng thu thập và xử lý dữ liệu đầu vào:
  - Sử dụng thư viện MediaPipe Hands để phát hiện tay và trích xuất 21 điểm landmark trên mỗi khung hình video.
  - Mỗi cử chỉ tay trong một khoảng thời gian tương ứng với một từ hoặc một kí tự trong ngôn ngữ ký hiệu.
- Tầng nhận diện từ:
  - Đầu vào là chuỗi tọa độ landmark của tay theo thời gian.
  - Mô hình học sâu LSTM sẽ phân loại chuỗi đó thành một từ cụ thể (ví dụ: “xin”, “chào”, “bạn”,...) hoặc các kí tự bảng chữ cái (ví dụ: "a", "b", "c",...).
  - Đầu ra là một chuỗi từ đơn.
- Tầng tổng hợp câu:

- Nhận đầu vào là chuỗi các từ đã nhận diện được từ mô hình A.
- Mô hình GRU được huấn luyện để hiểu ngữ cảnh và trật tự, từ đó xâu chuỗi các từ thành câu hoàn chỉnh có ngữ pháp.

3.3 Ưu điểm của thiết kế hai mô hình

Việc chia hệ thống nhận diện ngôn ngữ ký hiệu thành hai mô hình riêng biệt – mô hình A chuyên nhận diện từ vựng qua cử chỉ tay và mô hình B chuyên tổng hợp câu từ chuỗi các từ – mang lại nhiều lợi ích vượt trội về mặt kỹ thuật và triển khai thực tế. Thiết kế phân tầng này không chỉ giúp giảm thiểu độ phức tạp tổng thể mà còn nâng cao tính mô đun, khả năng mở rộng, bảo trì và tái sử dụng trong tương lai.

3.3.1 Phân chia nhiệm vụ rõ ràng, giảm độ phức tạp

Thay vì xây dựng một mô hình duy nhất có nhiệm vụ từ nhận diện cử chỉ đầu vào đến sinh câu hoàn chỉnh, cách tiếp cận hai mô hình cho phép tách biệt quy trình xử lý thành hai giai đoạn chuyên biệt. Mô hình A chỉ cần tập trung vào việc học biểu diễn và phân loại cử chỉ tay thành các đơn vị từ vựng rời rạc. Mô hình B tiếp nhận chuỗi từ này và học cách cấu trúc thành câu hợp ngữ cảnh.

Cách phân chia này giúp mỗi mô hình trở nên đơn nhiệm, dễ tối ưu và huấn luyện hơn, đồng thời giảm đáng kể độ phức tạp tổng thể của hệ thống.

3.3.2 Tăng tính mô đun, dễ kiểm soát và bảo trì

Khi các thành phần được thiết kế mô đun và hoạt động độc lập, việc kiểm soát và bảo trì trở nên dễ dàng hơn. Nếu mô hình A gặp lỗi, chẳng hạn như sai trong phân loại cử chỉ, nhóm phát triển chỉ cần tập trung sửa chữa hoặc cải tiến mô hình đó mà không ảnh hưởng đến mô hình B. Tương tự, nếu có nhu cầu thay đổi cách tổng hợp ngôn ngữ, mô hình B có thể được cập nhật mà không đòi hỏi tái huấn luyện mô hình A.

Đây là một điểm mạnh quan trọng trong quá trình phát triển phần mềm và triển khai hệ thống thực tế, khi mà việc nâng cấp từng thành phần một cách an toàn là nhu cầu thường xuyên.

3.3.3 Tối ưu hóa tài nguyên huấn luyện và triển khai

Về mặt tính toán, hai mô hình đơn nhiệm thường yêu cầu ít tài nguyên hơn so với một mô hình tích hợp phức tạp. Việc huấn luyện mô hình A có thể thực hiện riêng lẻ với các tập dữ liệu cử chỉ, trong khi mô hình B có thể huấn luyện với dữ liệu văn bản – giảm đáng kể chi phí và thời gian so với việc thu thập và huấn luyện dữ liệu đầu-cuối.

Thêm vào đó, mô hình A có thể được huấn luyện trước trên các tập dữ liệu về nhận diện cử chỉ, còn mô hình B có thể tái sử dụng các mô hình ngôn ngữ có sẵn như GPT,

BERT, hoặc LSTM.

3.3.4 Khả năng mở rộng hệ thống trong tương lai

Kiến trúc hai tầng tạo điều kiện thuận lợi cho việc mở rộng hệ thống. Một số hướng mở rộng bao gồm:

- Mở rộng từ vựng: mô hình A có thể được huấn luyện để nhận diện thêm nhiều loại cử chỉ tay, từ đó tăng độ phong phú của ngôn ngữ ký hiệu.
- Nâng cấp chất lượng ngôn ngữ: mô hình B có thể được thay thế bằng các kiến trúc hiện đại như Transformer, BERT hoặc các mô hình ngôn ngữ lớn (LLM) để cải thiện khả năng tổng hợp câu, biểu cảm, cú pháp và ngữ cảnh.
- Tích hợp thêm chức năng mới: chẳng hạn như hệ thống dịch đa ngôn ngữ (dịch từ ký hiệu sang nhiều ngôn ngữ), hoặc tích hợp công nghệ chuyển văn bản thành giọng nói (Text-to-Speech) để tạo nên một hệ thống giao tiếp hoàn chỉnh.

3.3.5 Tăng cường hỗ trợ nghiên cứu và cải tiến chuyên sâu

Khi hai mô hình hoạt động độc lập, các nhà nghiên cứu có thể tự do thử nghiệm các thuật toán, kỹ thuật học sâu khác nhau trên từng phần. Chẳng hạn:

- Thử nghiệm các kiến trúc mạng CNN hoặc Transformer cho mô hình A để cải thiện độ chính xác trong nhận diện cử chỉ phức tạp.
- Áp dụng kỹ thuật sinh văn bản có điều kiện (conditional generation) hoặc học tăng cường (reinforcement learning) vào mô hình B để tạo ra các câu ngôn ngữ mượt mà hơn.

Sự tách biệt này hỗ trợ tốt cho việc nghiên cứu chuyên sâu trong hai lĩnh vực riêng biệt là Gesture Recognition và Natural Language Processing, từ đó nâng cao chất lượng toàn diện của hệ thống.

3.4 Mô hình A – Nhận diện chữ ký hiệu

Sau khi mô hình học sâu được huấn luyện thành công, hệ thống được triển khai để thực hiện nhận diện cử chỉ tay trong thời gian thực từ nguồn dữ liệu là video webcam. Quá trình này bao gồm các bước chính: trích xuất đặc trưng, chuẩn hóa, suy luận với mô hình đã huấn luyện, và hiển thị kết quả dự đoán.

3.4.1 Cấu trúc mô hình LSTM

Mô hình sử dụng trong hệ thống là một mạng LSTM, được thiết kế để xử lý chuỗi dữ liệu các khung hình trích xuất từ video. Kiến trúc cụ thể bao gồm:

- Lớp LSTM (nn.LSTM) với một tầng và đầu vào dạng chuỗi có kích thước batch_first = True.
- Lớp Dropout với tỷ lệ 0.6 nhằm giảm hiện tượng overfitting.
- Hai lớp fully-connected:
  - Lớp thứ nhất nhận đầu vào là hidden_size và trả về hidden_size, sau đó được chuẩn hóa bởi BatchNorm1d và kích hoạt bằng ReLU.
  - Lớp thứ hai cho đầu ra phân loại ứng với số lớp cử chỉ (9 lớp).
- Hàm kích hoạt cuối cùng là softmax, dùng trong bước suy luận để đánh giá xác suất.

```python
class LSTMModel(nn.Module):
    def __init__(self, input_size=258, hidden_size=160, output_size=9):
        super(LSTMModel, self).__init__()
        self.lstm = nn.LSTM(input_size, hidden_size, num_layers=1, batch_first=True)
        self.dropout = nn.Dropout(0.6)
        self.fc1 = nn.Linear(hidden_size, hidden_size)
        self.bn = nn.BatchNorm1d(hidden_size)
        self.relu = nn.ReLU()
        self.fc2 = nn.Linear(hidden_size, output_size)

    def forward(self, x):
        x, _ = self.lstm(x)
        x = torch.max(x, dim=1).values
        x = self.dropout(x)
        x = self.fc1(x)
        x = self.bn(x)
        x = self.relu(x)
        x = self.fc2(x)
        return x
```

Hình 3.18 Cấu trúc mô hình LSTM

3.4.2 Tiền xử lý dữ liệu từ webcam

Để trích xuất dữ liệu đầu vào cho mô hình, hệ thống sử dụng thư viện MediaPipe Holistic để phát hiện và thu thập các điểm mốc (landmarks) từ:

- Tư thế toàn thân (pose_landmarks): 33 điểm, mỗi điểm có 4 thông số (x, y, z, visibility)
- Bàn tay trái và phải (left_hand_landmarks, right_hand_landmarks): mỗi tay 21 điểm, mỗi điểm có 3 thông số (x, y, z)

```python
mp_holistic = mp.solutions.holistic
holistic = mp_holistic.Holistic(
    static_image_mode=False,
    model_complexity=0,
    smooth_landmarks=True,
    min_detection_confidence=0.5,
    min_tracking_confidence=0.5
)
```

Hình 3.19 Tiền xử lí dữ liệu từ Webcam

Tổng số đặc trưng mỗi khung hình: 33*4 + 21*3 + 21*3 = 258 đặc trưng.

Các đặc trưng từ mỗi khung hình được chuẩn hóa theo từng khối (pose, left, right) bằng cách trừ trung bình và chia cho độ lệch chuẩn của từng phần, nhằm đảm bảo tính nhất quán và giảm ảnh hưởng của nhiễu trong quá trình suy luận.

3.4.3 Bộ đệm dữ liệu và ổn định kết quả đầu ra

Hệ thống sử dụng một deque (bộ đệm vòng) để lưu trữ 80 khung hình gần nhất, đại diện cho một đoạn chuyển động hoàn chỉnh của cử chỉ tay.

Tọa độ các điểm trên bàn tay và cơ thể của mỗi frame được chuẩn hóa trước khi xử lí:

```python
def normalize_frames(frame):
    pose = frame[:132].reshape(33, 4)
    left = frame[132:195].reshape(21, 3)
    right = frame[195:258].reshape(21, 3)

    pose_norm = normalize_keypoint_block(pose, dim=4)
    left_norm = normalize_keypoint_block(left, dim=3)
    right_norm = normalize_keypoint_block(right, dim=3)

    return np.concatenate([pose_norm, left_norm, right_norm])
```

Hình 3.20 Chuẩn hóa tọa độ từ frame

Cứ sau mỗi 5 khung hình, nếu bộ đệm đủ 80 mẫu, mô hình sẽ thực hiện suy luận:

- Chuỗi 80 khung hình được chuẩn hóa và đưa vào mô hình để dự đoán.
- Nếu xác suất dự đoán lớn hơn 0.7, nhãn dự đoán được đưa vào một bộ đệm thứ hai pred_history để đánh giá sự ổn định.

- Nếu một nhãn xuất hiện ít nhất 3 lần trong 5 lần gần nhất, hệ thống mới chính thức hiển thị nhãn đó lên màn hình.

```
if len(buffer) == 80 and frame_count % 5 == 0:
        x = np.array(buffer)
        x = normalize_keypoints(x)
        x_tensor = torch.from_numpy(x).float().unsqueeze(0).to(device)

        with torch.no_grad():
            output = model(x_tensor)
            pred = torch.argmax(output, dim=1).item()
            conf = torch.softmax(output, dim=1)[0][pred].item()

        if conf > 0.7:
            pred_history.append(pred)
            most_common, count = Counter(pred_history).most_common(1)[0]
            if count >= 3:
                label = f"{classes[most_common]} ({conf:.2f})"
        else:
            label = ""
```

Hình 3.21 Khi có đủ 80 frames, hệ thống tiến hành dự đoán cử chỉ

3.4.4 Kết quả thử nghiệm thực tế

Hệ thống hoạt động hiệu quả với các đoạn cử chỉ rõ ràng và tốc độ xử lý nhanh trên cả CPU và GPU. Thời gian nhận diện mỗi chuỗi là ngắn (dưới 1 giây) và độ chính xác đạt cao nếu người dùng biểu diễn cử chỉ đúng chuẩn.

Tuy nhiên, trong môi trường thực tế, độ sáng, góc quay camera và chuyển động nền có thể ảnh hưởng đến chất lượng nhận diện. Việc tối ưu thêm các tham số của MediaPipe, cũng như mở rộng dữ liệu huấn luyện, sẽ giúp cải thiện hệ thống.

3.5 Mô hình B – Tổng hợp câu hoàn chỉnh

Mô hình B được xây dựng dựa trên kiến trúc Seq2Seq, một kiến trúc nổi bật dùng để xử lý dữ liệu tuần tự. Kiến trúc này bao gồm hai thành phần chính: Encoder và Decoder, với vai trò lần lượt là mã hóa chuỗi đầu vào và sinh ra chuỗi đầu ra tương ứng. Cả hai thành phần trong mô hình đều sử dụng mạng hồi tiếp GRU, một biến thể hiệu quả và nhẹ hơn của LSTM.

3.5.1 Kiến trúc tổng thể

Trong mô hình này:

- Encoder GRU tiếp nhận một chuỗi các token đã được số hóa (chuyển thành chỉ số từ vựng), ánh xạ sang không gian vector thông qua lớp embedding, rồi truyền qua một tầng GRU để tạo ra các hidden state.
- Decoder GRU sử dụng hidden state cuối cùng của encoder như một biểu diễn ngữ

nghĩa của toàn bộ câu đầu vào. Tại mỗi bước thời gian, decoder nhận đầu vào là token trước đó (trong huấn luyện có thể dùng ground truth – kỹ thuật teacher forcing) và hidden state hiện tại để dự đoán token tiếp theo trong chuỗi đầu ra.

3.5.2 Encoder

Encoder có nhiệm vụ đọc toàn bộ chuỗi đầu vào và mã hóa thành một vector biểu diễn. Lớp này gồm hai thành phần:

- Embedding Layer: Biến mỗi token trong chuỗi đầu vào thành một vector liên tục có chiều cố định embed_size.
- GRU Layer: Duyệt qua chuỗi embedding và tạo ra một chuỗi các hidden state. Hidden state cuối cùng được sử dụng làm đầu vào khởi tạo cho decoder.

```python
class Encoder(nn.Module):
    def __init__(self, input_vocab_size, embed_size, hidden_size):
        super(Encoder, self).__init__()
        self.embedding = nn.Embedding(input_vocab_size, embed_size)
        self.gru = nn.GRU(embed_size, hidden_size, batch_first=True)

    def forward(self, x, lengths):
        embedded = self.embedding(x)
        packed = nn.utils.rnn.pack_padded_sequence(embedded, lengths.cpu(), batch_first=True, enforce_sorted=False)
        outputs, hidden = self.gru(packed)
        outputs, _ = nn.utils.rnn.pad_packed_sequence(outputs, batch_first=True)
        return outputs, hidden
```

Hình 3.22 Mô hình Encoder

- x: tensor đầu vào chứa các chỉ số token (shape: [batch_size, seq_len]).
- lengths: độ dài thực tế của mỗi chuỗi trong batch, phục vụ cho xử lý packing sequence.
- hidden: tensor biểu diễn trạng thái ngữ nghĩa của cả câu, được truyền sang decoder.

3.5.3 Decoder

Decoder có vai trò sinh ra chuỗi đầu ra từ thông tin mã hóa. Các thành phần chính:

- Lớp Embedding: chuyển đổi token hiện tại thành vector.
- Lớp GRU: cập nhật trạng thái dựa trên embedding đầu vào và hidden state trước đó.
- Lớp Linear: ánh xạ từ hidden state sang không gian từ vựng đầu ra, tạo phân phối xác suất cho token tiếp theo.

```python
class Decoder(nn.Module):
    def __init__(self, target_vocab_size, embed_size, hidden_size):
        super(Decoder, self).__init__()
        self.embedding = nn.Embedding(target_vocab_size, embed_size)
        self.gru = nn.GRU(embed_size, hidden_size, batch_first=True)
        self.fc = nn.Linear(hidden_size, target_vocab_size)

    def forward(self, x, hidden):
        x = x.unsqueeze(1)  # batch_size x 1
        embedded = self.embedding(x)
        output, hidden = self.gru(embedded, hidden)
        prediction = self.fc(output.squeeze(1))
        return prediction, hidden
```

Hình 3.23 Mô hình Decoder

- Mỗi bước thời gian, decoder chỉ tạo ra một token, nên đầu vào có shape (batch_size, 1).
- hidden được cập nhật sau mỗi bước, giữ trạng thái để tạo token tiếp theo.

3.5.4 Ưu điểm và hạn chế

Ưu điểm:

- Có thể học các ánh xạ từ chuỗi sang chuỗi bất kỳ, bất kể độ dài.
- Dễ huấn luyện nhờ cấu trúc GRU gọn nhẹ.
- Hoạt động tốt cho nhiều bài toán tuần tự như sinh mô tả ảnh, dịch máy, sinh văn bản.

Hạn chế:

- Không có cơ chế attention, nên với chuỗi dài, việc nén toàn bộ thông tin vào một hidden state cuối cùng có thể làm mất mát thông tin.
- Mô hình khó xử lý các quan hệ dài hạn hoặc cấu trúc phức tạp trong câu nếu không có kỹ thuật bổ sung như attention hoặc copy mechanism.

TÀI LIỆU THAM KHẢO

[1] P. Fayyazsanavi, A. Anastasopoulos, and J. Košecká, "Gloss2text: Sign language gloss translation using llms and semantically aware label smoothing," *arXiv preprint arXiv:2407.01394*, 2024.

[2] N. B. Ibrahim, H. H. Zayed, and M. M. Selim, "Advances, challenges and opportunities in continuous sign language recognition," *J. Eng. Appl. Sci*, vol. 15, no. 5, pp. 1205–1227, 2020.

[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[4] T. Nguyễn Thanh, *Deep learning cơ bản*. 2020.

[5] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, *Dive into deep learning*. Cambridge University Press, 2023.