ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KĨ THUẬT MÁY TÍNH

BÀI TẬP LỚN MÔN

CÁCH TIẾP CẬN HIỆN ĐẠI TRONG XỬ LÝ NGÔN NGỮ TỰ NHIÊN

XÂY DỰNG MÔ HÌNH CHO BÀI TOÁN PHÂN TÍCH CẢM XÚC DỰA TRÊN HỌC SÂU

Giảng viên hướng dẫn: PGS. Quản Thành Thơ

Học viên: Võ Thị Thanh Kiều

MSHV: 2370122

TP. HỒ CHÍ MINH, 2025

Lời cảm ơn

Trước tiên, em xin bày tỏ lòng biết ơn sâu sắc đến Thầy **PGS.TS. Quản Thành Thơ**, giảng viên hướng dẫn, đã tận tình đồng hành, theo sát và hỗ trợ em trong suốt quá trình thực hiện bài tập lớn. Nhờ vào sự chỉ bảo tận tâm, sự quan tâm và những định hướng quý báu từ Thầy, em đã có thể hoàn thành tốt nội dung nghiên cứu của mình. Đối với em, Thầy không chỉ là người truyền đạt tri thức mà còn là nguồn cảm hứng lớn lao, truyền lửa đam mê, sự tự tin và quyết tâm để em vượt qua mọi khó khăn và kiên trì theo đuổi đến cùng.

Tiếp theo, em xin gửi lời cảm ơn chân thành đến quý Thầy, Cô Trường **Đại học Bách Khoa – Đại học Quốc gia Thành phố Hồ Chí Minh** nói chung, và đặc biệt là các Thầy, Cô thuộc **Khoa Khoa học và Kỹ thuật Máy tính** nói riêng. Những kiến thức quý báu mà em đã được lĩnh hội trong suốt quá trình học tập tại trường chính là nền tảng vững chắc giúp em thực hiện bài tập lớn này một cách hiệu quả và có chiều sâu.

Bên cạnh đó, em cũng xin được chân thành cảm ơn gia đình và các bạn bè thân thiết – những người luôn ở bên, cổ vũ và động viên tinh thần em trong suốt quá trình học tập và thực hiện bài tập lớn. Hơn hết, em vô cùng biết ơn Trường Đại học Bách Khoa – Đại học Quốc gia Thành phố Hồ Chí Minh vì đã tạo nên một môi trường học tập năng động, chuyên nghiệp và giàu cảm hứng, giúp em không ngừng phát triển bản thân.

Em xin chân thành cảm ơn!

TP. Hồ Chí Minh, ngày 12 tháng 04 năm 2025

Võ Thị Thanh Kiều

Mục lục

Danh sách thuật ngữ tiếng Anh

Computer Vision	Thị giác máy tính
Deep Learning	Học sâu
Machine Learning	Học máy
Feature	Nét đặc trưng
Bidirectional Long Short-Term Memory	Mạng LSTM hai chiều
Machine Learning	Học máy
Gated Recurrent Unit	Đơn vị hồi tiếp có cổng
Threshold	Ngưỡng
Natural Language Processing	Xử lý ngôn ngữ tự nhiên

Danh sách từ viết tắt

DL	Deep Learning
ML	Machine Leaning
LSTM	Long Short Term Memory
BiLSTM	Bidirectional Long Short-Term Memory
GRU	Gated Recurrent Unit
NLP	Natural Language Processing
BERT	Bidirectional Encoder Representations from Transformers

Danh sách hình vẽ

Danh sách bảng

TÓM TẮT ĐỀ TÀI

Cuộc Cách mạng Công nghiệp lần thứ tư, hay còn gọi là Cách mạng Công nghiệp 4.0, đang từng ngày tác động sâu sắc đến mọi mặt của đời sống xã hội, từ kinh tế, y tế, giáo dục cho đến cách con người tương tác và làm việc. Một trong những biểu hiện rõ nét nhất của sự thay đổi này là sự phát triển mạnh mẽ của các thiết bị điện tử thông minh như điện thoại di động, máy tính bảng, đồng hồ thông minh, và các hệ thống nhà thông minh. Những thiết bị này không chỉ góp phần nâng cao chất lượng cuộc sống mà còn giúp kết nối con người với thế giới một cách nhanh chóng và tiện lợi hơn bao giờ hết. Tuy nhiên, song hành với những lợi ích vượt trội mà công nghệ mang lại, vấn đề an ninh mạng đang nổi lên như một thách thức cấp bách, đặc biệt trong bối cảnh các nền tảng thương mại điện tử và mạng xã hội ngày càng phát triển mạnh mẽ. Những nền tảng này không chỉ là nơi diễn ra các hoạt động mua bán trực tuyến, mà còn là môi trường để người dùng chia sẻ thông tin, bày tỏ quan điểm cá nhân, và giao lưu với cộng đồng toàn cầu. Chính vì thế, việc đảm bảo an toàn thông tin cá nhân, bảo mật tài khoản và phòng ngừa các hành vi tấn công mạng đã trở thành một yêu cầu thiết yếu đối với cả người dùng và nhà cung cấp dịch vụ.

Đáng chú ý, các đối tượng người dùng dễ bị tổn thương như trẻ em, người cao tuổi và các nhóm yếu thế trong xã hội đang ngày càng trở thành mục tiêu của nhiều hành vi tiêu cực trên không gian mạng. Những đối tượng này thường thiếu kiến thức hoặc kỹ năng để tự bảo vệ bản thân trước các nguy cơ như lừa đảo trực tuyến, quấy rối qua mạng, hay tiếp cận với nội dung không phù hợp. Không chỉ dừng lại ở đó, môi trường mạng hiện nay đang tồn tại một lượng lớn các nội dung độc hại và sai lệch như bình luận mang tính kích động, thông tin giả mạo, và những hình ảnh, video vi phạm chuẩn mực đạo đức xã hội. Những nội dung này không chỉ ảnh hưởng xấu đến tâm lý và nhận thức của người dùng, mà còn gây áp lực lớn cho đội ngũ kiểm duyệt nội dung, vốn phải đối mặt với khối lượng công việc ngày càng khổng lồ và phức tạp.

Tóm lại, trong khi Cách mạng Công nghiệp 4.0 mở ra nhiều cơ hội để nâng cao chất lượng cuộc sống, thì việc bảo vệ không gian mạng an toàn, lành mạnh và nhân văn đang trở thành một yêu cầu cấp thiết. Điều này đòi hỏi sự phối hợp chặt chẽ giữa các cơ quan chức năng, doanh nghiệp công nghệ và chính người dùng – nhằm xây dựng một môi trường số không chỉ tiện ích, hiện đại mà còn văn minh và an toàn cho mọi người, đặc biệt là thế hệ trẻ và những nhóm người dễ bị tổn thương trong xã hội.

Cụ thể, trong khuôn khổ đề tài này, em đã có những đóng góp chính yếu thông qua việc thiết kế và xây dựng một khung xử lý dữ liệu đầu vào dành riêng cho bài toán phân tích cảm xúc trong tiếng Việt, được triển khai trên tập dữ liệu của cuộc thi VLSP (Vietnamese Language and Speech Processing). Hệ thống tiền xử lý này bao gồm các bước lọc nhiễu, chuẩn hóa ngôn ngữ, xử lý từ viết tắt, ký tự đặc biệt, cũng như tích hợp các kỹ thuật tách từ và mã hóa phù hợp với ngữ cảnh tiếng Việt, vốn mang tính đặc thù và phức tạp hơn so với nhiều ngôn ngữ khác.

Trên nền tảng dữ liệu đã được xử lý, em triển khai và so sánh hiệu quả của các mô hình học sâu khác nhau, bao gồm các mô hình đơn như CNN, LSTM, RNN, GRU và BERT cũng như các mô hình kết hợp giữa CNN, LSTM, RNN và GRU – đây là hướng tiếp cận có khả năng tận dụng đồng thời ưu điểm của cả mạng nơ-ron tích chập trong trích xuất đặc trưng không gian và mạng LSTM/RNN/GRU trong xử lý chuỗi dữ liệu thời gian. Các mô hình này được huấn luyện và đánh giá cẩn thận trên tập dữ liệu bình luận tiếng Việt, qua đó tìm ra kiến trúc tối ưu nhằm đạt được độ chính xác cao trong phân loại cảm xúc.

Nghiên cứu này mang lại nhiều giá trị đáng kể cả về mặt khoa học lẫn thực tiễn. Về mặt học thuật, nghiên cứu đã góp phần đề xuất một quy trình tiền xử lý dữ liệu có hệ thống và hiệu quả, phù hợp với đặc điểm ngôn ngữ tiếng Việt trong bối cảnh mạng xã hội và thương mại điện tử. Đồng thời, việc xây dựng thành công các mô hình phân loại cảm xúc với hiệu suất cao đã góp phần mở rộng khả năng ứng dụng của các phương pháp học sâu trong xử lý ngôn ngữ tự nhiên tiếng Việt – một lĩnh vực vẫn đang trong quá trình hoàn thiện và phát triển mạnh mẽ.

Về mặt xã hội, hệ thống phân tích cảm xúc do em xây dựng có tiềm năng được ứng dụng rộng rãi trong việc sàng lọc và đánh giá nội dung bình luận trên các nền tảng thương mại điện tử, mạng xã hội hoặc cổng dịch vụ công. Điều này không chỉ hỗ trợ việc xây dựng một môi trường trực tuyến lành mạnh, văn minh mà còn giúp các doanh nghiệp, tổ chức và cơ quan chức năng kịp thời phát hiện, kiểm soát các nội dung tiêu cực, qua đó góp phần nâng cao văn hóa ứng xử trên không gian mạng trong bối cảnh chuyển đổi số hiện nay.

Chương 1

Tổng quan đề tài

1.1 Giới thiệu tổng quan và lý do chọn đề tài

Sự phát triển vượt bậc của thương mại điện tử trong những năm gần đây đã góp phần định hình lại thói quen mua sắm của người tiêu dùng, đặc biệt là trên các nền tảng trực tuyến. Những sàn thương mại điện tử như Shopee, Tiki, Lazada hay các nền tảng quốc tế như Amazon, eBay,... đã và đang tạo ra một "sân chơi" rộng lớn, nơi người tiêu dùng có thể dễ dàng tiếp cận, lựa chọn và mua sắm hàng hóa chỉ bằng vài thao tác chạm đơn giản trên thiết bị di động. Với tính tiện lợi, nhanh chóng và khả năng truy cập mọi lúc mọi nơi, việc mua sắm trực tuyến đang dần thay thế các hình thức truyền thống, trở thành xu hướng chủ đạo trong thời đại số.

Trong bối cảnh đó, việc phân tích cảm xúc từ các bình luận của người dùng trên nền tảng thương mại điện tử, đặc biệt là thông qua thiết bị di động, đóng vai trò vô cùng quan trọng. Những phản hồi, đánh giá và bình luận được người tiêu dùng để lại sau mỗi giao dịch không chỉ đơn thuần là những ý kiến cá nhân, mà còn là nguồn dữ liệu quý giá phản ánh mức độ hài lòng, kỳ vọng, cũng như những vấn đề tồn đọng trong quá trình mua sắm và sử dụng sản phẩm. Khi được xử lý và phân tích một cách khoa học, những dữ liệu này có thể cung cấp cái nhìn sâu sắc về cảm nhận thực sự của người tiêu dùng, từ đó hỗ trợ các doanh nghiệp điều chỉnh chiến lược kinh doanh nhằm cải thiện trải nghiệm người dùng một cách hiệu quả.

Cách mạng công nghệ, đặc biệt là sự phổ cập của smartphone và kết nối Internet di động, đã giúp người tiêu dùng dễ dàng truy cập các nền tảng thương mại điện tử ở bất kỳ thời điểm nào và tại bất kỳ đâu. Điều này đồng nghĩa với việc họ có thể nhanh chóng chia sẻ cảm xúc, để lại bình luận, đánh giá sản phẩm ngay sau khi nhận hàng hoặc trong quá trình sử dụng. Các cảm xúc này dù là tích cực hay tiêu cực thì đều là những tín hiệu quan trọng mà doanh nghiệp cần lắng nghe và phân tích để hiểu rõ hơn nhu cầu thực sự của khách hàng.

Việc áp dụng các kỹ thuật phân tích cảm xúc hiện đại như học sâu (deep learning), xử

lý ngôn ngữ tự nhiên (NLP), và các mô hình phân loại cảm xúc trên dữ liệu bình luận từ thiết bị di động sẽ mở ra nhiều cơ hội mới trong việc tối ưu hóa chiến lược tiếp thị, nâng cao chất lượng dịch vụ và cá nhân hóa trải nghiệm người dùng. Các doanh nghiệp có thể từ đó xây dựng chân dung khách hàng chính xác hơn, dự đoán hành vi tiêu dùng, phát hiện kịp thời các vấn đề tiềm ẩn, và nhanh chóng đưa ra giải pháp xử lý phù hợp.

Không những vậy, thông qua quá trình phân tích hàng loạt cảm xúc người dùng, doanh nghiệp còn có thể xây dựng được một cơ sở dữ liệu cảm xúc quy mô lớn phản ánh đa dạng các chiều cảm xúc xoay quanh sản phẩm, dịch vụ, hoặc thương hiệu. Đây sẽ là một nguồn tài nguyên chiến lược giúp định hướng phát triển sản phẩm, tối ưu dịch vụ chăm sóc khách hàng và tạo lợi thế cạnh tranh bền vững trên thị trường.

Từ tất cả những lý do trên, có thể khẳng định rằng việc lựa chọn nghiên cứu đề tài phân tích cảm xúc từ bình luận người dùng trên thiết bị di động là một hướng đi mang tính chiến lược và có giá trị ứng dụng cao. Nó không chỉ đáp ứng nhu cầu thực tiễn của ngành thương mại điện tử, mà còn góp phần thúc đẩy quá trình chuyển đổi số và nâng cao trải nghiệm tiêu dùng trong thời đại công nghệ 4.0.

1.2 Mục tiêu nghiên cứu

Trong khuôn khổ đề tài nghiên cứu này, em tập trung vào việc xây dựng các mô hình phân tích cảm xúc cho bình luận tiếng Việt, với trọng tâm là nghiên cứu các kỹ thuật tiền xử lý dữ liệu và ứng dụng các thuật toán học máy hiện đại. Mục tiêu tổng quát của đề tài là nâng cao hiệu quả phân loại cảm xúc thông qua việc xử lý dữ liệu một cách tối ưu và lựa chọn mô hình phù hợp với ngữ liệu tiếng Việt đặc thù. Để đạt được mục tiêu đó, em đã xác định và thực hiện các nội dung chính như sau:

• Tiền xử lý dữ liệu: Em đã áp dụng một chuỗi các kỹ thuật tiền xử lý dữ liệu nghiêm ngặt, bao gồm loại bỏ ký tự đặc biệt, chuẩn hóa văn bản, xử lý từ viết tắt, và tách từ tiếng Việt với sự hỗ trợ của thư viện PyVi. Ngoài ra, em còn sử dụng mô hình CBOW (Continuous Bag of Words) để ánh xạ các từ thành vector biểu diễn ngữ nghĩa trong không gian liên tục. Việc này không chỉ giúp giảm nhiễu dữ liệu mà còn làm nổi bật các đặc trưng ngữ nghĩa tiềm ẩn trong văn bản, hỗ trợ hiệu quả cho các mô hình học sâu phía sau. Quá trình này nhằm làm sạch và biểu diễn tốt hơn tập dữ liệu bình luận tiếng Việt từ cuộc thi VLSP, từ đó cải thiện chất lượng đầu vào cho các mô hình huấn luyện. Việc xử lý ngôn ngữ tự nhiên tiếng Việt đặt ra nhiều thách thức do đặc điểm cú pháp và ngữ nghĩa riêng biệt, nên quá trình tiền xử lý đóng vai trò then chốt trong toàn bộ hệ thống.

• Xây dựng và huấn luyện mô hình: em đã triển khai và so sánh hiệu quả của nhiều mô hình học máy hiện đại cho bài toán phân tích cảm xúc, bao gồm:

- **CNN (Convolutional Neural Network):** Mô hình này được sử dụng để trích xuất các đặc trưng không gian từ văn bản. CNN có khả năng nhận diện các cụm từ giàu

cảm xúc thông qua các bộ lọc (filter) hoạt động trên chuỗi từ, nhờ đó hỗ trợ phân loại cảm xúc một cách hiệu quả.

- **LSTM (Long Short-Term Memory):** Là một loại mạng nơ-ron hồi tiếp (RNN), LSTM có khả năng ghi nhớ thông tin trong chuỗi dữ liệu dài, giúp nắm bắt các quan hệ ngữ nghĩa phức tạp giữa các từ trong câu, từ đó nâng cao độ chính xác của việc phân tích cảm xúc.

- **GRU (Gated Recurrent Unit):** Tương tự như LSTM nhưng có kiến trúc gọn nhẹ hơn, GRU giúp giảm thời gian huấn luyện trong khi vẫn đảm bảo khả năng ghi nhớ các mối liên kết dài hạn trong chuỗi văn bản.

- **RNN (Recurrent Neural Network):** Mô hình học chuỗi cơ bản được sử dụng làm baseline để so sánh với các mô hình nâng cao như LSTM và GRU.

- **BERT (Bidirectional Encoder Representations from Transformers):** Là mô hình ngôn ngữ mạnh mẽ dựa trên kiến trúc Transformer, BERT có khả năng hiểu ngữ cảnh hai chiều, từ đó cải thiện rõ rệt hiệu suất phân loại cảm xúc so với các mô hình truyền thống.

- **Các mô hình kết hợp:** em cũng tiến hành xây dựng các mô hình kết hợp như CNN_LSTM, CNN_GRU, LSTM_GRU, RNN_GRU... nhằm tận dụng ưu điểm của từng kiến trúc trong việc trích xuất đặc trưng không gian và ghi nhớ ngữ cảnh thời gian.

Thông qua các hoạt động nghiên cứu và thực nghiệm nêu trên, đề tài không chỉ góp phần hệ thống hóa các kỹ thuật tiền xử lý dữ liệu cho văn bản tiếng Việt trong lĩnh vực phân tích cảm xúc, mà còn mở ra hướng tiếp cận khả thi trong việc ứng dụng học sâu và học kết hợp cho các bài toán xử lý ngôn ngữ tự nhiên. Kết quả của nghiên cứu này kỳ vọng sẽ đóng góp tích cực vào việc xây dựng các hệ thống thông minh có khả năng hiểu và phản hồi cảm xúc của người dùng trong các nền tảng trực tuyến, đặc biệt là thương mại điện tử và mạng xã hội.

1.3 Đối tượng và phương pháp nghiên cứu

Các bình luận từ các trang thương mại điện tử dần tạo nên một nguồn dữ liệu lớn có sức ảnh hưởng. Trong đó, vấn đề nội dung bình luận có thể giúp hoặc làm ảnh hưởng đến quyết định mua sản phẩm của người dùng. Điều này thúc đẩy em đề xuất giải pháp ứng dụng các kỹ thuật xử lý ngôn ngữ tiên tiến và hiệu quả để bóc tách các nội dung bình luận, phân tích cảm xúc từ các bình luận để giúp các hệt hống thương mại điện tử xử lý được các bình luận không phù hợp. Tuy giải pháp đề xuất chỉ ở pha nền tảng, sơ khởi, nhưng phạm vi ứng dụng

của nó là vô cùng lớn. Giải pháp đề xuất có thể áp dụng từ các đơn vị báo điện tử hay trang thông tin điện tử có lưu lượng thấp bình luận nhưng cần sự kiểm duyệt cao, cho đến các nền tảng bình luận lớn như mạng xã hội hay diễn đàn hoặc các trang mua sắm thương mại điện tử. Từ đó, xây dựng môi trường tích cực, văn minh cho không gian mạng. Không những vậy, ứng dụng còn là cơ sở để các cơ quan, tổ chức đánh giá, theo dõi các đối tượng nhằm mục đích quản lý, nghiên cứu, giáo dục, ... Từ việc xác định được mục tiêu, đối tượng và phạm vi đề tài. Điều này giúp em rất nhiều trong việc xác định được các việc cần phải làm và đối tượng mà đề tài hướng đến. Hơn thế nữa, phạm vi nghiên cứu giúp em khoanh vùng được các việc mà mình cần làm giúp không vượt khỏi và đi xa hơn so với mục tiêu đề ra.

Phương pháp được áp dụng để triển khai ý tưởng:

- Bước đầu nghiên cứu lý thuyết về các mô hình và phương pháp dựa trên các bài báo có liên quan để phân tích tổng hợp các thuật toán được sử dụng trước.
- Tiếp đến thực nghiệm trên bộ dữ liệu.
- Thống kê kết quả để so sánh với các phương pháp được thực nghiệm.
- Chọn ra những thuật toán cho kết quả khả quan để tiếp tục nghiên cứu.

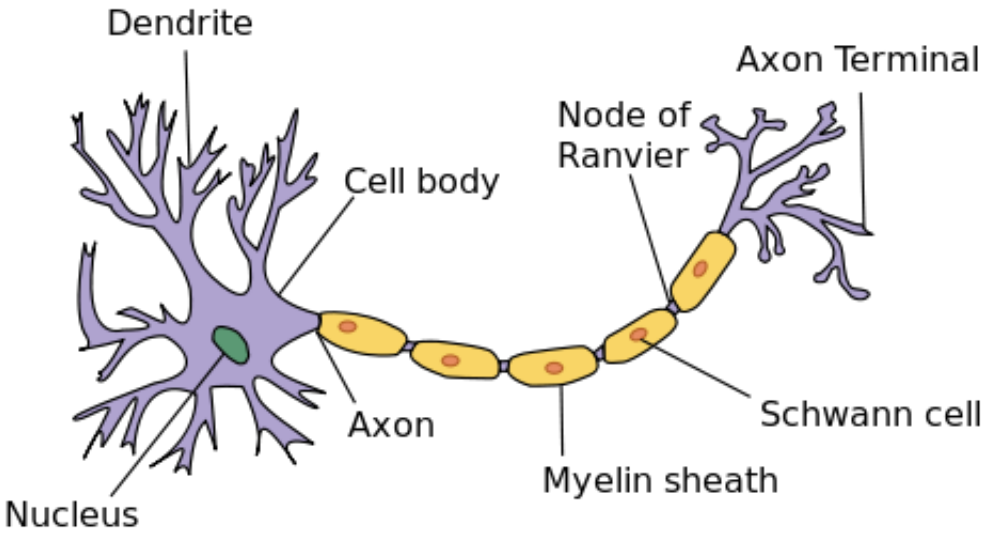
Chương 2

Cơ sở lý thuyết

2.1 Mạng thần kinh nhân tạo (Artificial Neural Networks – ANN)

2.1.1 Định nghĩa

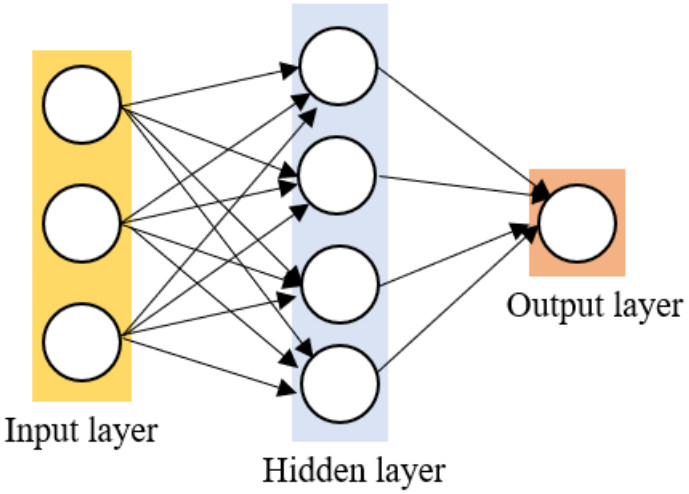
Mạng thần kinh nhân tạo – ANN là một thuật toán của học máy – Machine Learning, được lấy cảm hứng từ mạng lưới thần kinh tự nhiên của hệ thần kinh con người. Hình 2.6 mô tả một neuron - tế bào trong bộ não, mỗi neuron có phần thân (soma) chứa nhân, các tín hiệu đầu vào qua các sợ nhánh (dendrite) và các tín hiệu đầu ra qua sợi trục (axon), mỗi sợi trục chia thành các nhánh con kết nối với các sợ nhánh của nơ-ron khác để truyền tín hiêu. Như vậy, được hiểu đơn giản thì neuron là một đơn vị tính toán mà nó lấy tính hiệu vào thông qua sợi nhánh và thực hiện tính toán sau đó gửi giá trị đầu ra qua sợi trục đến các node khác hoặc neuron khác trong bộ não. Trong bộ não, các neuron giao tiếp với nhau qua các xung điện nhỏ được gửi qua sợi trục. Nếu các xung điện này đủ lớn để kích hoạt neuron, thì tín hiệu sẽ được đi qua sợi trục đến các sợi nhánh của các neuron khác. Mỗi neuron cần quyết định có kích hoạt neuron đấy hay không, nó tương tự các hoạt động của hàm sigmoid được trình bày ở phần tiếp theo.



Hình 2.1: Hình mô tả tế bào thần kinh
(nguôn)

Quá trình học của mạng nơ-ron bao gồm việc điều chỉnh các tham số của mạng (trọng số và bias) để giảm thiểu sai số giữa dự đoán của mạng và giá trị thực tế. Điều này được thực hiện thông qua một quá trình gọi là huấn luyện, trong đó mạng được cung cấp một lượng lớn dữ liệu huấn luyện có nhãn. Qua quá trình huấn luyện, mạng sẽ tự động học được các quy tắc để ánh xạ từ dữ liệu đầu vào đến đầu ra mong muốn.

Kiến trúc chung của mạng neuron nhân tạo cơ bản gồm 3 phần: Input, Hidden Layer và Output Layer. Input là lớp nhận dữ liệu đầu vào, được biểu diễn dưới dạng các vector số liệu. Mỗi phần tử trong vector tương ứng với một đặc trưng của dữ liệu. Hidden Layer (lớp ẩn) nằm giữa lớp đầu vào và lớp đầu ra, các lớp ẩn thực hiện quá trình xử lý thông tin phức tạp. Mỗi neuron trong lớp ẩn nhận đầu vào từ các neuron ở lớp trước, thực hiện phép biến đổi tuyến tính và phi tuyến tính (qua hàm kích hoạt) và truyền kết quả đến các neuron ở lớp tiếp theo. Số lượng lớp ẩn và số lượng neuron trong mỗi lớp ẩn có thể thay đổi tùy thuộc vào độ phức tạp của bài toán. Trong một mạng neuron nhân tạo có thể có nhiều Hidden Layer. Output Layer là lớp cung cấp kết quả cuối cùng của mạng. Các neuron trong lớp đầu ra nhận đầu vào từ các neuron ở Hidden Layer cuối cùng và thực hiện phép biến đổi cuối cùng để đưa ra dự đoán hoặc quyết định.



Hình 2.2: Hình mô tả một mô hình mạng neuron đơn giản

2.1.2 Các thuật ngữ thường gặp

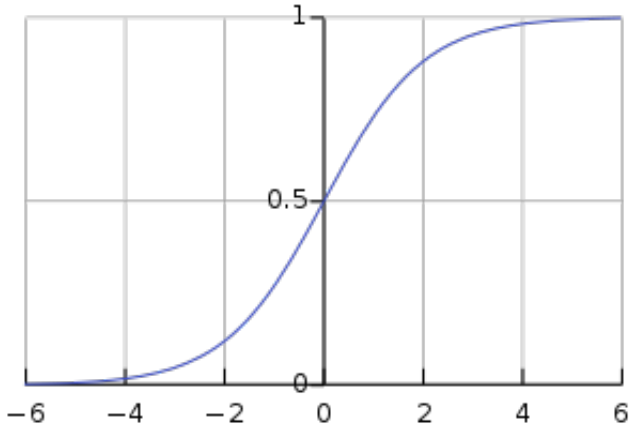
2.1.2.1 Hàm Sigmoid

Hàm Sigmoid là hàm kích hoạt phi tuyến tính.Công thức:

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{2.1}$$

Trong đó, e là cơ số của logarit tự nhiên, có giá trị sấp sĩ là $2,71828$.

Hàm Sigmoid nhận đầu vào là một số thực và đầu ra là một giá trị trong khoảng $(0,1)$ (hình 2.8). Nếu đầu vào là số thực âm rất nhỏ thì đầu ra sẽ tiệm cận 0 và ngược lại. Trước kia, hàm Sigmoid được dùng nhiều vì có đạo hàm tốt. Tuy nhiên, hiện tại hàm Sigmoid rất ít khi dùng vì nó bão hòa , triệt tiêu gradient và nó không có trung tâm là 0 gây ra khó khăn cho việc hội tụ.



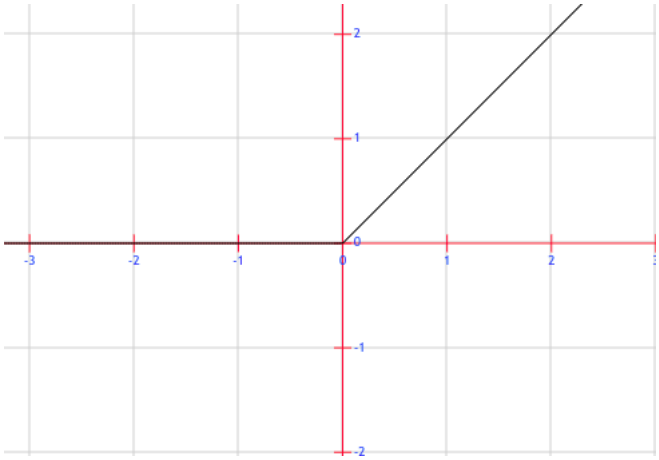
Hình 2.3: Đồ thị hàm Sigmoid

2.1.2.2 Hàm ReLU

Trong những năm gần đây, hàm ReLU đang được sử dụng khá nhiều khi huấn luyện các mạng neuron. Công thức của ReLU:

$$f(x) = max(0,x) \tag{2.2}$$

Từ công thức trên, ta thấy thực chất ReLU là hàm phân ngưỡng tại 0, nghĩa là các giá trị bé hơn 0 sẽ được gán giá trị bằng 0 và những giá trị lớn hơn hoặc bằng 0 sẽ được giữ nguyên giá trị. Hàm ReLU có ưu điểm là tính toan nhanh và giảm mất gradient. Đồ thị hàm ReLU được mô tả trong hình 2.9.



Hình 2.4: Đồ thị hàm ReLU

2.1.2.3 Hàm ngưỡng

Hàm ngưỡng là hàm không liên tục và miền giá trị của nó chỉ mang hai giá trị 0 và 1. Hàm tạo ra trạng thái tắt và mở của các neuron. Hàm ngưỡng được mô tả theo công thức:

$$\varphi(v) = \begin{cases} 1 & \texttt{if } x \geq 0 \\ 0 & \texttt{if } x < 0 \end{cases} \tag{2.3}$$

Hiện nay, hàm ngưỡng ít được sử dụng vì làm mất gradient do hàm này không có đạo hàm tại điểm 0 và đạo hàm tại các điểm còn lại đều bằng 0.

2.1.2.4 Hàm Softmax

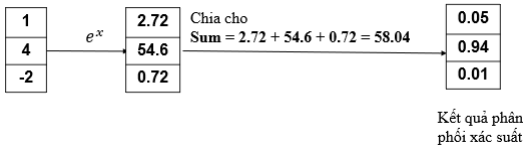
Trong bài toan phân loại K lớp, hàm Softmax là được dùng để tính phân bố xác suất của dữ liệu đầu vào trên mỗi lớp cho trước. Hàm thường được dùng làm hàm kích hoạt cuối cùng của mạng neuron để chuẩn hóa đầu ra của mạng thành phân phối sác xuất trên các lớp đầu ra được dự đoán. Hàm Softmax nhận đầu vào là một vector z của K số thực và chuẩn

hóa nó thành phân thối xác xuất bao gồm K xác suất tỉ lệ với cấp số nhân của các số đầu vào. Công hàm Softmax để tính P_i (xác suất của lớp i) từ giá trị x_i gồm n giá trị:

$$P_i = \frac{e^x}{\sum_{j=1}^{n} e^j} \tag{2.4}$$

Trong đó e là cơ số của logarit tự nhiên, $i = 1, \cdots, n$.

Ví dụ ta có input đầu vào $[1, 4, -4]$ ta sẽ được kết quả phân phối xác suất là $[0.05, 0.94, 0.01]$.



Hình 2.5: Tính phân phối xác suất tính bằng hàm Softmax

2.1.2.5 Thuật toán Backpropagation

Thuật toán Backpropagation là một trong những thuật toán cốt lõi trong việc huấn luyện các mạng nơ-ron nhân tạo. Nó được sử dụng để tính toán gradient của hàm mất mát (loss function) đối với các tham số của mạng (trọng số và bias). Từ đó, chúng ta có thể điều chỉnh các tham số này theo hướng giảm thiểu hàm mất mát, giúp mạng học được các quy tắc ánh xạ từ dữ liệu đầu vào đến đầu ra mong muốn. Quá trình học này diễn ra thông qua việc lặp đi lặp lại hai giai đoạn chính: lan truyền xuôi và lan truyền ngược.

- Lan truyền xuôi là quá trình truyền dữ liệu đầu vào qua các lớp của mạng nơ-ron. Tại mỗi lớp, các neuron thực hiện phép biến đổi tuyến tính và phi tuyến tính để tính toán đầu ra. Cuối cùng, mạng đưa ra một dự đoán.

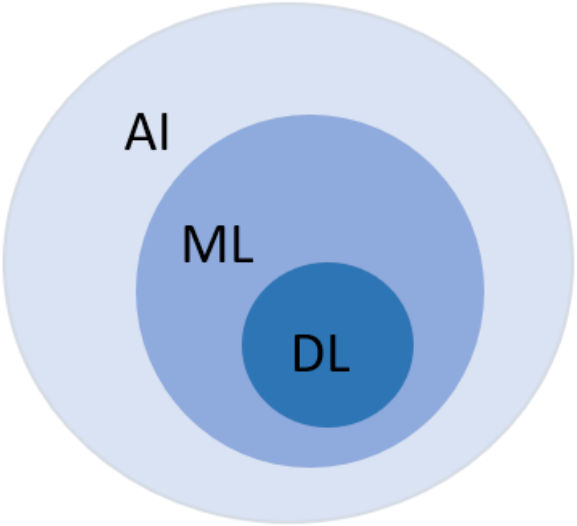
- Lan truyền ngược là quá trình toán vtínhà điều chỉnh các tham số của mạng. Đầu tiên, ta tính toán sai số giữa dự đoán của mạng và giá trị thực tế. Sau đó, sai số này được truyền ngược trở lại các lớp trước đó để tính toán gradient của hàm mất mát đối với từng tham số. Gradient này chỉ ra hướng mà chúng ta cần điều chỉnh các tham số để giảm thiểu sai số. Cuối cùng, các tham số được cập nhật theo hướng giảm gradient.

2.2 Học sâu (Deep Learning)

2.2.1 Định nghĩa

Deep Learning – học sâu là một tập con của Machine Learning. Học sâu là một mô hình học sâu dựa trên các ý tưởng từ não bộ với việc tiếp thu rất nhiều tầng trừu tượng để

biểu diễn dữ liệu. Dữ liệu Deep Learning thường yêu cầu lớn và nguồn tài nguyên sử dụng nhiều hơn, nhờ đó cho độ chính xác cao.



Hình 2.6: Hình miêu tả quan hệ của Artificial Intelligence – AI, Machine Learning – ML và Deep Learning – DL

2.2.2 Lịch sử phát triển

Giai đoạn Perceptron: Đây là một trong những nền móng đầu tiên của Deep Learning. Năm 1957, Frank Posenblatt là người đầu tiên khởi nguồn về Perceptron trong một bài nghiên cứu của mình. Perceptron là một thuật toán học có giám sát giúp giải quyết vấn đề phân lớp nhị phân. Mặc dù mang nhiều kì vọng, thuật toán này đã nhanh chóng bị chứng minh không giải quyết vấn đề đơn giản. Năm 1969, Marvin Minsky và Seymour Papert đã chứng minh rằng không thể "học" được hàm số XOR khi sử dụng perceptron. Phát hiện này khiến cho việc nghiên cứu về các mô hình học sâu gian đoạn gần 20 năm. Thuật toán K-means sẽ hoạt động như sau:

Giai đoạn sự ra đời của MLP và Backpropagation: Năm 1986, Geoffrey Hinton cùng với hai tác giả xuất bản bài báo khoa học chứng minh rằng mạng neuron với nhiều lớp (multi-layer perceptron - MLP) có thể được huấn luyện một cách hiệu quả dựa trên thuật toán lan truyền ngược. Việc này giúp mạng neuron thoát được hạn chế của perceptron là chỉ có thể biểu diễn được các quan hệ tuyến tính. Thuật toan mạng vài lại thành công ban đầu, nổi trội nhất là CNN cho bài viết nhận dạng chữ số viết tay. Để huân luyện được mô hình mạng neuron ta cần rất nhiều dữ liệu huấn luyện. Tuy nhiên vào thời điểm đó,dữ liệu ảnh và ảnh được gán nhãn rất hiếm. Ngay cả khi có đủ dữ liệu, một vấn đề nan giải lúc bấy giờ là khả năng tính toán của máy tính không đủ. Điều này, làm cho việc nghiên cứu mạng neuron dần được thay thế bởi Support Vector Machine – SVM.

Giai đoạn Deep Learning ra đời: Năm 2006, Geoffrey Hinton tiếp tục giới hiệu ý

tưởng tiền huấn luyện không giám sát (Unsupervised Pretraining) thông qua Deep Belief Networts (DBN) để tạo các mô hình. Sử dụng ý tưởng này, mọi người đã đào tạo các mạng có nhiều lớp hơn so với trước đây. Kể từ đó, mạng neuron với nhiều lớp được đổi tên thành Deep Learning.

Giai đoạn phát triển của Deep Learning: Năm 2012, tại cuộc thi ImageNet Large Scale Visual Recognition Challenge (ILSVRC), Alex Krizhevsky, Ilya Sutskever, và Geoffrey Hinton tham gia và đạt kết quả top-5 error rate là 16%. Mô hình này là là một mạng neuron tích chập sau này được gọi là Alexnet. Sau mô hình Alexnet, nhiều mô hình dành giải cao trong những giải cao trong những năm tiếp theo là: ZFNet (2013), GoogLeNet (2014), VGG (2014), ResNet (2015) InceptionResNet (2016).

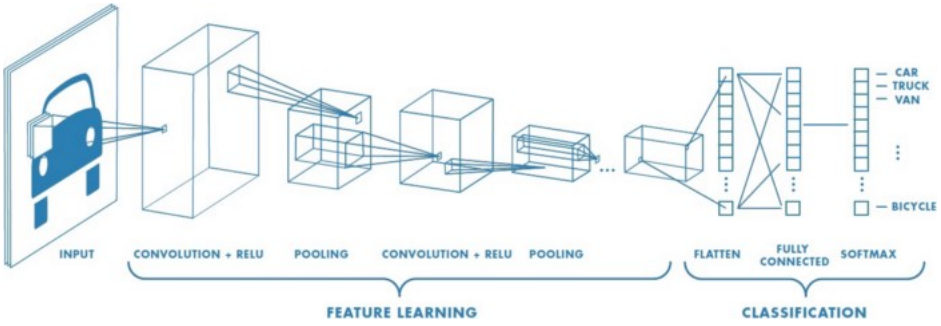
Việc thành công của Deep Learning hiện nay nhờ vào các điều kiện sau:

- Sự ra đời của nhiều bộ dữ liệu lớn được gắn nhãn.
- Khả năng tính toán song song với tốc độ cao của của GPU.
- Sự ra đời của hàm ReLU và các hàm kích hoạt.
- Có nhiều kiến trực mạng mới được ra đời.
- Có nhiều thư viện hỗ trợ việc huấn luyện các mô hình học sâu: caffe, tensorflow, mxnet, pytorch, keras, ...
- Xuất hiện nhiều kĩ thuật tối ưu mới: Adagrad, RMSProp, Adam, ...

2.2.3 Mạng neuron tích chập (Convolutional Neural Network - CNN)

2.2.3.1 Định nghĩa

Mạng neuron tích chập - Convolutional Neural Network (CNN) [1] là một mạng neuron học sâu được ứng dụng để giải quyết các bài toán nhận dạng hình ảnh, phân loại ảnh, phân tích ảnh y tế, xử lý ngôn ngữ tự nhiên và hệ thống khuyến nghị. Đầu vào của CNN là một tensor có dạng $NxHxWxD$, với N là số ảnh, H là chiều cao, W là chiều rộng và D là chiều sâu của ảnh (số kênh của ảnh). Một mô hình CNN gồm dãy các lớp cơ bản gồm lớp Convolutional (Conv), lớp Pooling (Pool) và lớp Fully-Connected (FC). Các lớp này liên kết với nhau theo một thứ tự nhất định. Thông thường, một ảnh sẽ được lan truyền qua lớp Convolutional (Conv) đầu tiên, sau đó các giá trị đầu ra sẽ lan truyền qua lớp Pooling (Pool). Cuối cùng, được lan truyền qua tầng fully connected layer và softmax để tính sác xuất ảnh đó chứa vật thế gì.



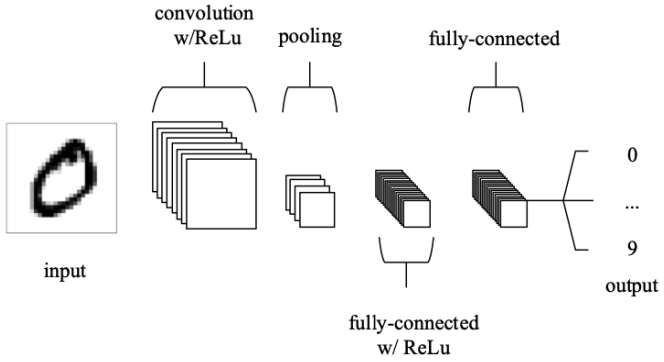
Hình 2.7: Mạng neuron với nhiều lớp chập
(nguồn **??**)

2.2.3.2 Kiến trúc

Như đã đề cập trước đó, mạng nơ-ron tích chập (**Convolutional Neural Networks - CNNs**) chủ yếu được xây dựng dựa trên giả định rằng đầu vào là dữ liệu hình ảnh. Điều này dẫn đến việc thiết kế kiến trúc của CNN được tối ưu hóa để xử lý loại dữ liệu đặc thù này. Một trong những điểm khác biệt quan trọng giữa CNN và các mạng nơ-ron truyền thống là cách tổ chức các nơ-ron trong mỗi lớp. Cụ thể, các nơ-ron trong các lớp của CNN được sắp xếp theo ba chiều: chiều không gian của đầu vào (chiều cao và chiều rộng), và chiều sâu. Ở đây, *chiều sâu* không phải là số lớp trong mạng, mà là chiều thứ ba của **khối kích hoạt** (*activation volume*). Không giống như các mạng nơ-ron truyền thống (ANNs), nơi mà mỗi nơ-ron trong một lớp kết nối với toàn bộ các nơ-ron ở lớp trước, các nơ-ron trong CNN chỉ kết nối với một *vùng cục bộ nhỏ* ở lớp trước đó. Điều này giúp mạng học được các đặc trưng cục bộ quan trọng trong ảnh như cạnh, góc, và hoa văn. Cụ thể, giả sử đầu vào là một ảnh màu RGB có kích thước $64 \times 64 \times 3$ (chiều cao, chiều rộng và ba kênh màu). Qua nhiều lớp tích chập và xử lý, đầu ra cuối cùng sẽ có kích thước $1 \times 1 \times n$, với n là số lớp phân loại. Điều này có nghĩa là CNN đã nén toàn bộ thông tin từ ảnh đầu vào thành một khối nhỏ chứa điểm số dự đoán (*class scores*) được tổ chức theo chiều sâu.

Mạng nơ-ron tích chập (**Convolutional Neural Networks - CNNs**) bao gồm ba loại lớp chính: **lớp tích chập (convolutional layer)**, **lớp gộp (pooling layer)** và **lớp kết nối đầy đủ (fully-connected layer)**. Khi các lớp này được xếp chồng lên nhau theo một trật tự hợp lý, chúng tạo thành một kiến trúc CNN hoàn chỉnh.

Một kiến trúc CNN đơn giản dùng để phân loại tập dữ liệu MNIST được minh họa trong Hình 2.8 dưới đây:



Hình 2.8: Một kiến trúc CNN đơn giản bao gồm năm lớp chính: tích chập (có ReLU), gộp, hai lớp fully-connected và lớp đầu ra.

Chức năng của các thành phần chính trong CNN cho NLP được mô tả như sau:

1. **Lớp đầu vào (Input Layer)**: Đầu vào là một chuỗi các từ trong văn bản. Mỗi từ được ánh xạ thành một vector nhúng có kích thước cố định (ví dụ, 300 chiều). Do đó, đầu vào của mạng có thể được biểu diễn như một ma trận có kích thước $n \times d$, trong đó n là số từ trong câu, và d là kích thước vector nhúng.

2. **Lớp tích chập (Convolutional Layer)**: Các bộ lọc (filters) trượt qua từng cụm từ (n-gram) trong chuỗi để phát hiện các đặc trưng quan trọng. Ví dụ, một filter có kích thước 3 sẽ học được các đặc trưng từ các cụm 3 từ liên tiếp trong câu. Các hàm kích hoạt như **ReLU** thường được áp dụng để tăng tính phi tuyến.

3. **Lớp gộp (Pooling Layer)**: Lớp này thực hiện **max-pooling** hoặc **average-pooling** trên đầu ra của lớp tích chập để giảm chiều dữ liệu và chọn lọc các đặc trưng nổi bật nhất từ toàn bộ văn bản. Điều này giúp mô hình học các đặc trưng ngữ nghĩa quan trọng bất kể vị trí xuất hiện trong câu.

4. **Lớp kết nối đầy đủ (Fully-connected Layer)**: Các đặc trưng đã được gộp sẽ được đưa vào một hoặc nhiều lớp fully-connected nhằm mục đích suy diễn (inference) và phân loại. Trong bài toán phân loại văn bản, đầu ra cuối cùng thường là một lớp **softmax** cho các nhãn lớp.

Ưu điểm của CNNs trong NLP là khả năng trích xuất đặc trưng cục bộ (local patterns) hiệu quả, đồng thời giúp mô hình nhận biết các cụm từ hoặc cấu trúc ngữ pháp quan trọng trong câu. Khi được kết hợp với kỹ thuật embedding hiện đại như BERT hoặc Word2Vec, CNNs có thể đạt hiệu quả rất tốt trong các tác vụ như phân loại văn bản, nhận diện thực thể, hoặc phát hiện cảm xúc. Tuy nhiên, việc thiết kế và tối ưu hóa các siêu tham số (filter size, số filters, padding, stride, v.v.) là yếu tố then chốt quyết định đến hiệu suất mô hình.

2.2.3.3 Convolutional layer

Như tên gọi của nó, tầng tích chập (convolutional layer) đóng vai trò cốt lõi trong cách mạng nơ-ron tích chập (CNN) hoạt động. Trong bối cảnh xử lý ngôn ngữ tự nhiên (NLP), tầng này sử dụng các *kernel* (bộ lọc) có thể học được để trích xuất đặc trưng từ chuỗi văn bản đầu vào. Các kernel trong NLP thường có kích thước nhỏ theo chiều thời gian (ví dụ: độ dài n-gram), nhưng trải dài toàn bộ chiều đặc trưng của đầu vào (embedding dimension). Khi văn bản được đưa vào mạng, mỗi kernel sẽ trượt dọc theo chuỗi từ và thực hiện tích vô hướng (dot product) giữa trọng số của nó và đoạn văn bản tương ứng, tạo ra một *bản đồ kích hoạt* (activation map) dạng một chiều.

Khi kernel trượt qua chuỗi văn bản, mỗi vị trí sẽ được tính toán bằng tổng có trọng số giữa kernel và phần tử đầu vào tương ứng. Từ đó, mạng sẽ học được những kernel có khả năng "kích hoạt" khi phát hiện đặc trưng quan trọng tại một vị trí cụ thể trong câu (ví dụ: sự xuất hiện của một cụm từ biểu cảm trong bài phân tích cảm xúc). Mỗi kernel tương ứng với một bản đồ kích hoạt riêng, và các bản đồ này sẽ được xếp chồng lên nhau theo chiều sâu để tạo thành đầu ra tổng thể của tầng tích chập.

So với mạng nơ-ron truyền thống (fully-connected ANN), CNN có ưu điểm là giảm số lượng tham số cần học nhờ hai cơ chế:

- **Kết nối cục bộ:** Mỗi nơ-ron ở tầng tích chập chỉ kết nối với một vùng nhỏ (receptive field) của đầu vào thay vì toàn bộ chuỗi.
- **Chia sẻ tham số (parameter sharing):** Các vị trí khác nhau trong chuỗi sử dụng cùng một kernel, tức là cùng bộ trọng số, nhờ vậy làm giảm đáng kể số lượng tham số cần cập nhật trong quá trình lan truyền ngược (backpropagation).

Tầng tích chập trong NLP có thể được điều chỉnh thông qua các siêu tham số:

- **Số lượng kernel:** quyết định số chiều sâu đầu ra.
- **Kích thước kernel:** thường đại diện cho độ dài n-gram cần trích xuất.
- **Bước trượt (stride):** điều khiển mức độ di chuyển của kernel trên chuỗi văn bản.
- **Zero-padding:** giúp kiểm soát kích thước đầu ra bằng cách thêm giá trị 0 ở đầu/cuối chuỗi.

Cho đầu vào có chiều dài V, kích thước kernel R, padding Z, và stride S, thì chiều dài đầu ra L được tính theo công thức:

$$L = \left\lfloor \frac{(V - R + 2Z)}{S} \right\rfloor + 1$$

Nếu kết quả không phải là số nguyên, tức là stride không hợp lý và cần được điều chỉnh lại. Tầng tích chập trong NLP giúp mạng học được các đặc trưng quan trọng trong chuỗi văn

bản như n-gram biểu cảm, chủ đề, hoặc cú pháp. Thông qua việc chia sẻ tham số và kết nối cục bộ, CNN không chỉ tiết kiệm bộ nhớ và thời gian huấn luyện mà còn mang lại khả năng tổng quát hóa tốt hơn trên tập dữ liệu đầu vào.

2.2.3.4 Pooling layer

Trong kiến trúc mạng nơ-ron tích chập (CNN), tầng **pooling** giữ một vai trò vô cùng quan trọng trong việc giảm thiểu độ phức tạp của mô hình mà vẫn bảo toàn được những đặc trưng quan trọng của dữ liệu. Đây là một trong ba loại tầng chính của CNN (gồm tầng tích chập – convolutional, tầng pooling – gộp, và tầng fully-connected – kết nối đầy đủ), và nó thường xuất hiện sau các tầng tích chập. Về bản chất, tầng pooling được thiết kế để **giảm dần kích thước không gian** (spatial dimensions) của các biểu diễn đặc trưng (feature maps) sau khi đã được trích xuất thông qua các tầng tích chập. Việc giảm kích thước này không chỉ giúp tiết kiệm bộ nhớ và thời gian tính toán, mà còn đóng vai trò như một kỹ thuật regularization nhằm **hạn chế hiện tượng overfitting** bằng cách loại bỏ thông tin dư thừa hoặc nhiễu không cần thiết.

Hình thức pooling phổ biến nhất là **max-pooling**. Với cách tiếp cận này, bản đồ đặc trưng đầu vào được chia thành các vùng nhỏ (thường là 2×2), và mỗi vùng này sẽ được thay thế bằng giá trị lớn nhất trong vùng. Thao tác này giúp giữ lại đặc trưng mạnh nhất, từ đó cải thiện khả năng nhận diện và phân biệt đặc trưng quan trọng. Max-pooling đặc biệt hiệu quả trong việc bảo toàn sự hiện diện của các đặc trưng quan trọng bất kể vị trí của chúng trong không gian đầu vào. Ngoài ra còn có các lớp pooling như average pooling tính giá trị trung bình trong vùng thay vì giá trị lớn nhất và L1/L2 pooling sử dụng các phép chuẩn hóa để làm mượt đặc trưng.

Tùy vào tác vụ, kích thước kernel và stride (bước trượt) cũng có thể thay đổi. Một kernel 2×2 với stride 2 sẽ giảm kích thước bản đồ đặc trưng xuống còn 25%, giúp tiết kiệm tài nguyên tính toán mà không mất quá nhiều thông tin. Trong một số trường hợp, **overlapping pooling** (stride nhỏ hơn kích thước kernel, như kernel 3×3 với stride 2) được sử dụng để giảm mất mát thông tin nhưng đồng thời tăng độ phức tạp.

Tầng pooling có tính chất **phá hủy thông tin** (destructive). Việc rút gọn biểu diễn đầu vào đồng nghĩa với việc một lượng thông tin nào đó sẽ bị mất đi. Tuy nhiên, nếu được sử dụng hợp lý, pooling sẽ tạo ra **biểu diễn đặc trưng cô đọng, ổn định**, và góp phần làm mô hình tổng quát tốt hơn.

Pooling cũng tạo nên tính chất **dịch bất biến (translation invariance)** trong CNN, nghĩa là mô hình có thể nhận diện đặc trưng ngay cả khi chúng bị lệch nhẹ về vị trí — điều này rất quan trọng trong cả xử lý ảnh và văn bản. Tầng pooling đóng vai trò như một "bộ lọc chọn lọc thông minh", loại bỏ phần nhiễu và giữ lại cốt lõi của thông tin. Nhờ đó, mạng CNN không chỉ trở nên nhẹ hơn, dễ huấn luyện hơn, mà còn đạt được độ chính xác và khả năng tổng quát cao hơn trong nhiều tác vụ học máy. Việc nắm rõ cơ chế và ảnh hưởng

của tầng pooling sẽ giúp ta xây dựng các kiến trúc CNN mạnh mẽ và tối ưu hơn trong các lĩnh vực như thị giác máy tính, xử lý ngôn ngữ tự nhiên, phân tích chuỗi thời gian hoặc âm thanh.

2.2.3.5 Fully Connected Layer

Trong kiến trúc mạng nơ-ron tích chập (CNN), tầng **fully-connected (FC)** hay còn gọi là tầng kết nối đầy đủ, là một thành phần quan trọng, đặc biệt ở giai đoạn cuối của mô hình. Không giống như các tầng tích chập hay pooling vốn hoạt động cục bộ, tầng fully-connected có cơ chế kết nối toàn cục: mỗi nơ-ron trong tầng này được kết nối với **toàn bộ** các nơ-ron trong tầng trước và tầng sau. Tầng fully-connected được lấy cảm hứng từ cách tổ chức nơ-ron trong mạng nơ-ron truyền thống (Artificial Neural Network – ANN). Ở đó, các nơ-ron giữa hai tầng liên tiếp đều được liên kết với nhau, cho phép mô hình học được các mối quan hệ toàn diện giữa các đặc trưng đầu vào.

Trong CNN, các tầng tích chập và pooling chủ yếu đảm nhiệm nhiệm vụ **trích xuất đặc trưng**, đặc biệt là các đặc trưng cục bộ (local features). Tuy nhiên, để thực hiện các tác vụ như **phân loại (classification)** hoặc **hồi quy (regression)**, ta cần một tầng có khả năng tổng hợp toàn bộ đặc trưng đã trích xuất để đưa ra quyết định cuối cùng — đó chính là vai trò của tầng fully-connected. Sau khi dữ liệu đầu vào đi qua nhiều tầng tích chập và pooling, nó được "làm phẳng" (flatten) thành một vector 1 chiều, sau đó được đưa vào một hoặc nhiều tầng fully-connected. Tại đây, toàn bộ thông tin từ các bản đồ đặc trưng sẽ được tổng hợp lại và truyền qua một hàm kích hoạt như ReLU hoặc softmax, từ đó tạo ra **đầu ra phân lớp cuối cùng**.

Đặc điểm kỹ thuật và tính chất

- **Số lượng tham số lớn**: Do mỗi nơ-ron kết nối với toàn bộ nơ-ron của tầng trước, số lượng trọng số có thể rất lớn.
- **Nguy cơ overfitting cao**: Tầng FC dễ dẫn đến hiện tượng overfitting nếu không có kỹ thuật regularization như dropout.
- **Tổng hợp thông tin toàn cục**: Đây là lợi thế lớn của tầng FC, giúp mô hình có cái nhìn tổng thể thay vì chỉ xử lý thông tin cục bộ.

Tầng fully-connected cũng đóng vai trò quan trọng trong các lĩnh vực khác như:

- **Xử lý ngôn ngữ tự nhiên (NLP)**: Ánh xạ vector embedding sang xác suất nhãn.
- **Chuỗi thời gian (Time-series)**: Tổng hợp thông tin từ các bước thời gian để đưa ra dự đoán.

Tầng fully-connected trong CNN là cầu nối quan trọng giữa phần trích xuất đặc trưng và phần ra quyết định. Tuy không có tính chất cục bộ như tầng tích chập, nhưng nó là thành phần cần thiết để mô hình hóa mối quan hệ phi tuyến giữa các đặc trưng và mục tiêu đầu ra. Việc thiết kế số lượng tầng FC, số lượng nơ-ron trong mỗi tầng, cũng như sử dụng các kỹ thuật giảm overfitting sẽ quyết định chất lượng của toàn bộ mô hình.

2.2.4 Mạng hồi quy (Recurrent Neural Network - RNN)

2.2.4.1 Định nghĩa

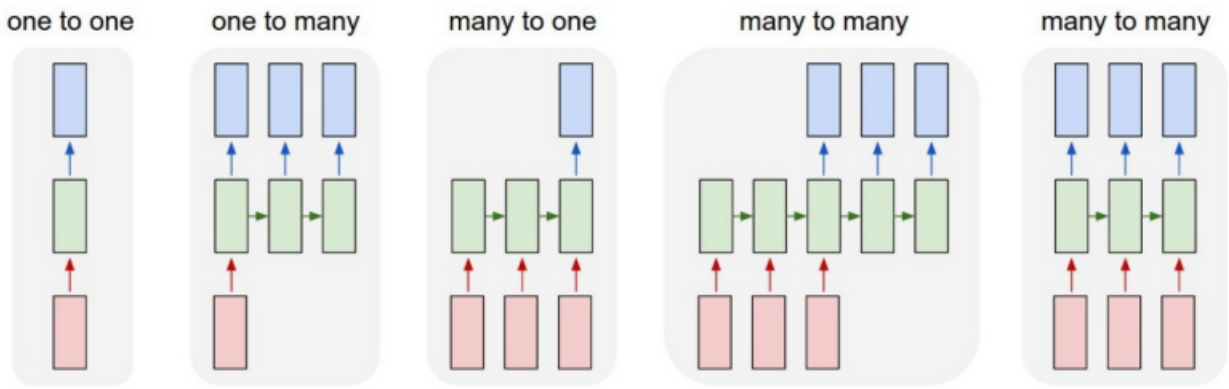
Trong lĩnh vực xử lý video, một bài toán quan trọng là phân loại hành động của con người. Ví dụ, chúng ta muốn xác định một đoạn video 30 giây có chứa hành động "đứng", "ngồi", "chạy" hay "đánh nhau". Để giải quyết bài toán này, chúng ta cần một mô hình có khả năng xử lý chuỗi dữ liệu, tức là một chuỗi các hình ảnh liên tiếp theo thời gian. Video là một chuỗi các hình ảnh được sắp xếp theo thứ tự thời gian. Thứ tự của các hình ảnh này rất quan trọng, vì nó thể hiện sự phát triển của hành động theo thời gian. CNN rất hiệu quả trong việc phân loại hình ảnh tĩnh, nhưng lại khó khăn trong việc nắm bắt thông tin liên quan đến thứ tự của các hình ảnh trong một video.

RNN là một loại mạng thần kinh nhân tạo được thiết kế đặc biệt để xử lý dữ liệu chuỗi. RNN có khả năng "nhớ" thông tin từ các bước thời gian trước đó và sử dụng thông tin này để dự đoán ở bước thời gian hiện tại.

2.2.4.2 Dữ liệu dạng chuỗi

Dữ liệu dạng chuỗi (sequence data) là những dữ liệu có thứ tự, trong đó mỗi phần tử trong chuỗi đều có mối liên hệ chặt chẽ với các phần tử trước và sau nó. Ví dụ điển hình là các khung hình liên tiếp trong một video hoặc các từ trong một câu. Ví dụ, trong bài toán dự đoán đột quỵ tim, các chỉ số sức khỏe của bệnh nhân được thu thập theo thời gian tạo thành một chuỗi dữ liệu. RNN có thể học hỏi từ chuỗi dữ liệu này để dự đoán khả năng xảy ra đột quỵ. Tương tự, trong bài toán dịch máy, thứ tự các từ trong một câu quyết định ý nghĩa của câu. RNN cũng được sử dụng để xử lý các chuỗi từ này và tạo ra bản dịch tương ứng. Một điểm chung của các bài toán trên là dữ liệu đầu vào đều là các chuỗi có thứ tự. Việc nắm bắt được mối liên hệ giữa các phần tử trong chuỗi là rất quan trọng để giải quyết các bài toán này. RNN với khả năng "nhớ" thông tin từ các bước thời gian trước đó, là một công cụ phù hợp để xử lý loại dữ liệu này.

2.2.4.3 Phân loại bài toán RNN



Hình 2.9: ác dạng bài toán RNN

Mạng nơ-ron hồi quy có những loại bài toán sau:

- One-to-one: Một đầu vào tương ứng với một đầu ra duy nhất. Ví dụ: Dự đoán giá nhà dựa trên diện tích.
- One-to-many: Một đầu vào tạo ra nhiều đầu ra. Ví dụ: Tạo một bản nhạc dựa trên một nốt nhạc đầu tiên.
- Many-to-one: Nhiều đầu vào kết hợp để tạo ra một đầu ra. Ví dụ: Phân loại cảm xúc của người dùng dựa trên một đoạn văn bản.
- Many-to-many (độ dài sequence bằng nhau): Một chuỗi đầu vào tương ứng với một chuỗi đầu ra có cùng độ dài. Ví dụ: Dịch máy từ tiếng Anh sang tiếng Việt.
- Many-to-many (độ dài sequence khác nhau): Một chuỗi đầu vào tương ứng với một chuỗi đầu ra có độ dài khác. Ví dụ: Tóm tắt một bài báo dài thành một đoạn văn ngắn.

2.2.5 Long Short-Term Memory - LSTM)

2.2.5.1 Lý do ra đời của LSTM

Long Short-Term Memory (LSTM) [2] là một biến thể nổi bật của mạng nơ-ron hồi tiếp (Recurrent Neural Networks – RNN), được Hochreiter và Schmidhuber giới thiệu vào năm 1997 nhằm khắc phục một trong những hạn chế nghiêm trọng nhất của RNN truyền thống: hiện tượng *gradient biến mất* (vanishing gradient). Khi làm việc với các chuỗi dữ liệu dài, mạng RNN thường gặp khó khăn trong việc duy trì và học các mối quan hệ phụ thuộc dài hạn, do tín hiệu lan truyền ngược suy giảm đáng kể khiến cho việc cập nhật trọng số trở nên vô hiệu.

LSTM được thiết kế với một kiến trúc đặc biệt cho phép duy trì và kiểm soát dòng chảy thông tin trong thời gian dài hơn. Không giống như các tế bào RNN cơ bản chỉ sử dụng trạng thái ẩn h_t để ghi nhớ thông tin ngắn hạn, tế bào LSTM còn duy trì thêm một thành phần quan trọng là trạng thái ô (cell state) c_t, đóng vai trò như một bộ nhớ dài hạn. Trạng thái này có thể được hình dung như một băng chuyền thông tin chạy xuyên suốt qua chuỗi thời gian, nơi dữ liệu được truyền đi và điều chỉnh một cách tối thiểu nhờ vào các cổng điều khiển.

LSTM sử dụng ba cổng chính: cổng quên (forget gate), cổng đầu vào (input gate), và cổng đầu ra (output gate), nhằm kiểm soát việc loại bỏ thông tin cũ, ghi nhớ thông tin mới, và quyết định phần thông tin nào sẽ được sử dụng để tạo đầu ra tại mỗi bước thời gian. Nhờ vào cơ chế kiểm soát mềm dẻo này, LSTM có thể học các phụ thuộc dài hạn mà không bị ảnh hưởng nghiêm trọng bởi vấn đề gradient.

2.2.5.2 Trực quan hóa hoạt động của LSTM

Một cách hình dung trực quan về khả năng lưu giữ và điều tiết thông tin của LSTM[2] là qua ví dụ về một mô hình dự đoán nội dung tiếp theo trong một bộ phim có hai nhân vật chính là Alice và Bob. Nếu trong một đoạn dài của bộ phim chỉ xuất hiện các phân cảnh liên quan đến Bob, mô hình cần có khả năng lờ đi các thay đổi không liên quan và vẫn duy trì thông tin về Alice để đưa ra dự đoán phù hợp khi cô ấy xuất hiện trở lại. Cấu trúc cổng của LSTM cho phép điều đó: cổng quên giúp loại bỏ những phần thông tin không còn quan trọng, cổng đầu vào chọn lọc thông tin mới đáng lưu giữ, trong khi cổng đầu ra điều phối giữa trí nhớ ngắn hạn và dài hạn để tạo nên phản hồi tại mỗi bước thời gian.

2.2.5.3 Cấu trúc và tính toán trong LSTM

Về mặt kỹ thuật, mỗi tế bào LSTM thực hiện một chuỗi các phép tính nhằm cập nhật và duy trì hai trạng thái chính là h_t và c_t. Tại thời điểm t, với đầu vào là x_t và trạng thái ẩn từ bước trước h_{t-1}, ta tiến hành như sau:

Trước tiên, *cổng quên* f_t xác định phần nào trong trạng thái ô trước đó c_{t-1} nên được giữ lại, thông qua hàm kích hoạt sigmoid:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.5}$$

$$c'_t = f_t \odot c_{t-1} \tag{2.6}$$

Tiếp theo, *cổng đầu vào* quyết định phần thông tin mới nào sẽ được ghi nhớ vào bộ nhớ. Trước hết, trạng thái ứng viên mới $\tilde{c}_t$ được tính thông qua hàm tanh, phản ánh nội dung mới có thể lưu trữ, sau đó được điều chỉnh bởi một cổng sigmoid i_t để xác định trọng số cập nhật:

$$\tilde{c}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{2.7}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.8}$$

$$c_t = c'_t + i_t \odot \tilde{c}_t \tag{2.9}$$

Cuối cùng, *cổng đầu ra* o_t điều chỉnh thông tin nào sẽ được sử dụng để tạo ra trạng thái ẩn mới h_t, dựa trên trạng thái ô vừa được cập nhật:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.10}$$

$$h_t = o_t \odot \tanh(c_t) \tag{2.11}$$

Trong các công thức trên, $\odot$ là phép nhân từng phần tử (element-wise), còn σ là hàm sigmoid để bảo đảm đầu ra nằm trong khoảng $[0, 1]$.

LSTM có thể được xem như một đột phá trong thiết kế kiến trúc mạng hồi tiếp, cho phép xử lý hiệu quả các chuỗi dữ liệu có độ dài lớn mà vẫn duy trì được khả năng học thông tin ngữ cảnh từ xa. Với cơ chế cổng tinh vi và khả năng lưu trữ thông tin trong thời gian dài, LSTM đã chứng minh được hiệu quả vượt trội trong nhiều bài toán học chuỗi như dịch máy, nhận diện giọng nói, phân tích ngữ nghĩa văn bản, và các ứng dụng đòi hỏi xử lý thông tin thời gian.

2.2.6 Gated recurrent unit - GRU

Mạng hồi tiếp (Recurrent Neural Networks – RNNs)[3] từ lâu đã được xem là một công cụ mạnh mẽ trong việc xử lý dữ liệu tuần tự như chuỗi thời gian và ngôn ngữ tự nhiên. Tuy nhiên, RNN truyền thống bộc lộ nhiều điểm yếu khi phải học các phụ thuộc dài hạn trong chuỗi. Một trong những vấn đề nghiêm trọng là hiện tượng *gradient biến mất*, xảy ra khi tín hiệu lan truyền ngược qua nhiều bước thời gian trở nên quá nhỏ, dẫn đến việc cập nhật trọng số không hiệu quả. Ngược lại, trong một số trường hợp khác, mạng có thể gặp phải hiện tượng *gradient bùng nổ*, khi gradient tăng quá mức làm cho quá trình huấn luyện trở nên bất ổn định. Thêm vào đó, do chỉ lưu trữ thông tin thông qua một trạng thái ẩn duy nhất h_t, RNN thường gặp khó khăn trong việc ghi nhớ thông tin từ xa, dẫn đến khả năng ghi nhớ bị giới hạn. Tất cả những yếu tố này khiến việc huấn luyện RNN trở nên phức tạp và ít hiệu quả trong thực tiễn.

Để giải quyết những hạn chế nêu trên, kiến trúc Gated Recurrent Unit (GRU) đã được đề xuất như một biến thể đơn giản hơn so với LSTM nhưng vẫn giữ được hiệu quả trong việc xử lý các chuỗi dài. GRU loại bỏ trạng thái ô riêng biệt như trong LSTM và thay vào đó, chỉ duy trì một trạng thái ẩn h_t. Cốt lõi trong kiến trúc GRU là việc sử dụng hai cổng điều khiển chính: cổng đặt lại (reset gate) và cổng cập nhật (update gate).

Cổng đặt lại có vai trò điều chỉnh mức độ ảnh hưởng của thông tin từ trạng thái ẩn trước đó đối với trạng thái hiện tại. Toán học mô tả cổng này như sau:

$$r_t = \sigma(W_r x_t + U_r h_{t-1})$$

Trong khi đó, cổng cập nhật đảm nhiệm việc xác định thông tin nào nên được giữ lại từ quá khứ và thông tin nào nên được cập nhật từ đầu vào hiện tại. Cổng này được tính bằng:

$$z_t = \sigma(W_z x_t + U_z h_{t-1})$$

Dựa trên hai cổng này, GRU tính toán ra trạng thái ẩn ứng viên, vốn là một biểu diễn trung gian thể hiện trạng thái mới tiềm năng. Trạng thái này được điều khiển bởi cổng đặt lại để điều chỉnh ảnh hưởng của trạng thái trước đó:

$$\tilde{h}_t = \tanh(W_h x_t + U_h(r_t \odot h_{t-1}))$$

Cuối cùng, trạng thái ẩn mới h_t được tính bằng cách kết hợp giữa trạng thái cũ và trạng thái ứng viên thông qua cổng cập nhật:

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$

Biểu thức này cho thấy rằng khi giá trị của z_t gần 0, mạng ưu tiên giữ lại thông tin từ quá khứ, trong khi khi z_t gần 1, thông tin từ trạng thái ứng viên sẽ chiếm ưu thế. Cách phối hợp mềm mại này giúp GRU thích ứng linh hoạt với độ dài phụ thuộc trong chuỗi.

GRU mang lại nhiều lợi ích nổi bật so với RNN truyền thống. Nhờ vào cơ chế cổng, GRU có khả năng kiểm soát và bảo toàn gradient tốt hơn, giúp giảm thiểu hiện tượng gradient biến mất – một trong những yếu tố then chốt để học được các phụ thuộc dài hạn trong chuỗi dữ liệu. Ngoài ra, khả năng ghi nhớ thông tin trong thời gian dài được cải thiện đáng kể nhờ vào việc cập nhật chọn lọc thông tin mới và thông tin cũ thông qua cổng cập nhật. Một lợi thế khác là GRU có cấu trúc đơn giản hơn so với LSTM vì không sử dụng trạng thái ô riêng biệt và số lượng cổng điều khiển ít hơn, từ đó giúp giảm số lượng tham số và tăng tốc độ huấn luyện.

Gated Recurrent Unit (GRU) là một cải tiến đáng kể trong lĩnh vực mạng nơ-ron hồi tiếp, giúp khắc phục các vấn đề nghiêm trọng của RNN truyền thống như gradient biến mất, khả năng ghi nhớ giới hạn và huấn luyện kém hiệu quả. Nhờ vào cơ chế cổng thông minh và kiến trúc tinh gọn, GRU không chỉ đạt hiệu năng cao trong nhiều tác vụ mà còn đảm bảo tính linh hoạt và tốc độ trong quá trình huấn luyện. Do đó, GRU đang ngày càng trở thành lựa chọn phổ biến trong các mô hình học sâu hiện đại xử lý dữ liệu tuần tự.

2.2.7 BERT

BERT (Bi-directional Encoder Representations from Transformers) là một mô hình ngôn ngữ được đề xuất bởi Jacob Devlin và cộng sự vào năm 2018 [4]. Đây là mô hình học sâu đầu tiên trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP) sử dụng cơ chế mã hóa hai chiều (deeply bidirectional) kết hợp với phương pháp học chuyển tiếp (transfer learning) trên quy mô lớn. Nhờ thiết kế này, BERT đạt được kết quả tiên tiến (state-of-the-art) trên 11 tác vụ NLP khác nhau, bao gồm phân loại văn bản, nhận diện thực thể và trả lời câu hỏi.

Khác với các mô hình truyền thống chỉ xử lý ngữ cảnh theo một chiều (trái–phải hoặc phải–trái), BERT có khả năng hiểu ngữ cảnh cả trước và sau từ đang xét trong một lần xử lý duy nhất. Kiến trúc của BERT bao gồm nhiều tầng mã hóa (Transformer encoders) được xếp chồng lên nhau theo dạng chuỗi truyền thẳng (feed-forward), trong đó đầu ra của tầng này sẽ là đầu vào cho tầng kế tiếp.

Mô hình BERT được huấn luyện với hai kích thước chính: $\textbf{BERT}_{\textbf{BASE}}$ (12 tầng Transformer, 768 đơn vị ẩn, 12 đầu chú ý) và $\textbf{BERT}_{\textbf{LARGE}}$ (24 tầng, 1024 đơn vị ẩn, 16 đầu chú ý). Dữ liệu huấn luyện bao gồm toàn bộ văn bản Wikipedia và BookCorpus của Google, nhằm đảm bảo phạm vi bao phủ rộng và đa dạng ngữ nghĩa.

Quá trình huấn luyện BERT diễn ra qua hai giai đoạn chính: **tiền huấn luyện (pre-training)** và **tinh chỉnh (fine-tuning)**.

2.2.7.1 Giai đoạn tiền huấn luyện

Tiền huấn luyện BERT dựa trên hai tác vụ không giám sát:

- **Masked Language Modeling (MLM)**: Một tỷ lệ 15% từ trong chuỗi đầu vào sẽ bị che khuất (mask) và mô hình được yêu cầu dự đoán từ bị ẩn. Cụ thể, 80% số từ bị thay bằng token đặc biệt [MASK], 10% bị thay bằng từ ngẫu nhiên và 10% được giữ nguyên. Mục tiêu của phương pháp này là buộc mô hình phải học ngữ cảnh đầy đủ hai chiều của mỗi từ.

- **Next Sentence Prediction (NSP)**: Với hai câu liên tiếp, mô hình cần dự đoán liệu câu thứ hai có thực sự tiếp nối câu thứ nhất trong ngữ cảnh ban đầu hay không. Việc này giúp mô hình học được quan hệ giữa các câu trong văn bản.

Trong quá trình mã hóa, BERT sử dụng ba loại embedding chính: token embedding, position embedding và segment embedding. Ngoài ra, ba token đặc biệt cũng được đưa vào chuỗi đầu vào: [CLS] (cho tác vụ phân loại), [SEP] (đánh dấu ranh giới giữa hai câu), và [MASK] (cho tác vụ MLM).

2.2.7.2 Giai đoạn tinh chỉnh mô hình

Sau khi hoàn tất tiền huấn luyện, mô hình có thể được tinh chỉnh dễ dàng cho các tác vụ cụ thể bằng cách thêm một tầng đầu ra tương ứng với bài toán. Mô hình BERT giữ nguyên

trọng số của các tầng Transformer và chỉ tinh chỉnh một số tham số ở tầng trên cùng. Một số ví dụ điển hình về tinh chỉnh như sau:

- **GLUE (General Language Understanding Evaluation)**: Đây là tập hợp nhiều tác vụ NLP khác nhau. Trong bài toán phân loại, đầu ra của token [CLS] sẽ được đưa qua một mạng neural nhiều tầng và hàm softmax để suy luận lớp phân loại.
- **SQuAD v1.1 và v2.0**: Đây là các tác vụ trả lời câu hỏi. Mô hình dự đoán vị trí bắt đầu và kết thúc câu trả lời trong đoạn văn dựa trên biểu diễn của các token. Riêng với SQuAD v2.0, một token [CLS] được chèn thêm giữa token bắt đầu và kết thúc để mô hình có thể học khả năng nhận biết các câu hỏi không có đáp án.
- **SWAG (Situations with Adversarial Generations)**: Trong tác vụ dự đoán câu tiếp theo, đầu vào bao gồm một câu dẫn và bốn câu ứng viên. Token [CLS] được mã hóa để đại diện cho toàn bộ chuỗi và sau đó được đưa vào tầng phân loại.

Nhờ thiết kế linh hoạt, khả năng mã hóa ngữ cảnh hai chiều, cùng với cơ chế huấn luyện hiệu quả, BERT đã tạo ra một bước tiến đột phá trong nhiều lĩnh vực NLP, trở thành nền tảng cho hàng loạt mô hình sau này như RoBERTa, ALBERT, DistilBERT và nhiều biến thể khác.

PhoBERT là một mô hình ngôn ngữ tiền huấn luyện theo kiến trúc BERT, được phát triển riêng cho tiếng Việt bởi nhóm nghiên cứu của VinAI Research [5]. Đây là mô hình đầu tiên áp dụng kỹ thuật học không giám sát trên tập dữ liệu lớn tiếng Việt để huấn luyện các phiên bản BERT nhằm phục vụ cho nhiều tác vụ xử lý ngôn ngữ tự nhiên. Khác với BERT gốc được huấn luyện trên dữ liệu tiếng Anh với phương pháp phân tách từ bằng tokenizer WordPiece, PhoBERT sử dụng **SentencePiece** với mã hóa **Byte-Pair Encoding (BPE)**, một phương pháp phân tách từ hiệu quả hơn đối với các ngôn ngữ có tính chất hình thái học như tiếng Việt. Trước khi huấn luyện, văn bản tiếng Việt được tách từ bằng trình tách từ `pyvi` để xử lý các cụm từ đa từ (multi-word expressions), sau đó áp dụng BPE trên toàn bộ tập văn bản. PhoBERT[6] được huấn luyện trên một tập dữ liệu tiếng Việt rất lớn, bao gồm khoảng **20GB văn bản** thu thập từ các nguồn như Wikipedia tiếng Việt và Common Crawl. Hai phiên bản của PhoBERT được công bố, tương tự như các phiên bản của BERT:

- **PhoBERT$_{\text{base}}$**: Bao gồm 12 lớp Transformer, 768 đơn vị ẩn và 12 đầu chú ý (attention heads). Đây là phiên bản có kiến trúc tương đương với BERT$_{\text{BASE}}$.
- **PhoBERT$_{\text{large}}$**: Bao gồm 24 lớp Transformer, 1024 đơn vị ẩn và 16 đầu chú ý, tương ứng với kiến trúc của BERT$_{\text{LARGE}}$.

Trong quá trình tiền huấn luyện, PhoBERT cũng sử dụng hai tác vụ chính là **Masked Language Modeling (MLM)** và **Next Sentence Prediction (NSP)**, tương tự như BERT gốc. Sau khi huấn luyện, PhoBERT được tinh chỉnh (fine-tuned) trên nhiều tác vụ NLP

tiếng Việt như phân loại văn bản, gán nhãn thực thể (NER), phân tích cú pháp phụ thuộc, và phân tích cảm xúc. Kết quả đánh giá cho thấy PhoBERT đạt hiệu năng vượt trội so với các mô hình trước đó như VnCoreNLP hay các mô hình đa ngôn ngữ như mBERT (multilingual BERT). Trên bộ tiêu chuẩn đánh giá VLSP (Vietnamese Language and Speech Processing), PhoBERT thiết lập nhiều kết quả state-of-the-art, đặc biệt trong các tác vụ như gán nhãn thực thể (F1-score cao hơn mBERT đến 5%) và phân loại cảm xúc. PhoBERT không chỉ cho thấy hiệu quả vượt trội trên dữ liệu tiếng Việt mà còn đóng vai trò quan trọng trong việc thúc đẩy nghiên cứu NLP cho các ngôn ngữ có ít tài nguyên (low-resource languages). Mô hình đã được công bố và triển khai công khai trên Hugging Face, cho phép cộng đồng dễ dàng sử dụng và tinh chỉnh cho các tác vụ cụ thể.

2.2.8 Bidirectional Long Short-Term Memory - BiLSTM

Bidirectional Long Short-Term Memory (BiLSTM)[7] là một mở rộng của kiến trúc LSTM truyền thống, cho phép mạng nơ-ron học được thông tin ngữ cảnh từ cả hai chiều – quá khứ (forward) và tương lai (backward) – trong chuỗi dữ liệu đầu vào. Kiến trúc này lần đầu tiên được giới thiệu bởi Schuster và Paliwal [**?**] và đã được áp dụng rộng rãi trong nhiều bài toán xử lý ngôn ngữ tự nhiên (NLP) như gán nhãn từ loại (POS tagging), nhận dạng thực thể tên (NER), phân tích cảm xúc, và dịch máy.

2.2.8.1 Động cơ ra đời

Mặc dù LSTM có khả năng ghi nhớ các phụ thuộc dài hạn trong chuỗi nhờ cơ chế bộ nhớ và cổng điều khiển, nó vẫn xử lý dữ liệu theo một hướng thời gian – thường là từ trái sang phải. Trong nhiều bài toán NLP, việc hiểu đúng ngữ nghĩa của một từ hoặc một cụm từ phụ thuộc không chỉ vào ngữ cảnh phía trước mà còn vào ngữ cảnh phía sau. BiLSTM giải quyết điểm hạn chế này bằng cách sử dụng hai mạng LSTM độc lập: một xử lý chuỗi theo chiều thuận (forward), và một xử lý chuỗi theo chiều nghịch (backward).

2.2.8.2 Kiến trúc mô hình

Trong BiLSTM, mỗi đầu vào x_t tại thời điểm t sẽ được xử lý đồng thời bởi hai mạng LSTM:

- Mạng **LSTM tiến** (forward LSTM): xử lý chuỗi từ $x_1 \rightarrow x_t \rightarrow x_T$
- Mạng **LSTM lùi** (backward LSTM): xử lý chuỗi từ $x_T \rightarrow x_t \rightarrow x_1$

Kết quả của hai mạng LSTM được kết hợp lại để tạo ra biểu diễn cuối cùng:

$$\overrightarrow{h_t} = \text{LSTM}_{\text{forward}}(x_t), \quad \overleftarrow{h_t} = \text{LSTM}_{\text{backward}}(x_t)$$

$$h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}]$$

Trong đó, h_t là vector biểu diễn đầu ra tại thời điểm t, được tạo thành bằng cách nối (concatenate) hai vector trạng thái ẩn từ hai chiều.

2.2.8.3 Ưu điểm và ứng dụng

Việc kết hợp thông tin từ cả hai chiều giúp BiLSTM vượt trội hơn so với LSTM đơn hướng trong nhiều tác vụ NLP. Một số ưu điểm chính bao gồm:

- **Ngữ cảnh đầy đủ hơn:** BiLSTM có khả năng nắm bắt mối liên hệ giữa các từ trong cả quá khứ và tương lai của chuỗi, đặc biệt hiệu quả với các câu có cấu trúc ngữ pháp phức tạp.
- **Hiệu quả cao trong NER, POS, Chunking, và Parsing:** Do khả năng nhận diện mô hình và cấu trúc ngôn ngữ tốt hơn, BiLSTM thường được sử dụng làm tầng nền trong các hệ thống học sâu hiện đại cho xử lý ngôn ngữ.
- **Tích hợp tốt với Attention và CRF:** BiLSTM có thể kết hợp với các lớp như Attention hoặc Conditional Random Fields (CRF) để nâng cao độ chính xác trong các bài toán gán nhãn tuần tự.

2.2.9 Continuous Bag of Words - CBOW

ontinuous Bag of Words (CBOW)[8] là một trong hai kiến trúc chính được đề xuất trong mô hình Word2Vec, bên cạnh mô hình Skip-Gram. CBOW được thiết kế để học biểu diễn từ (word embeddings) bằng cách khai thác ngữ cảnh xung quanh của một từ mục tiêu trong câu.

2.2.9.1 Ý tưởng chính

CBOW hoạt động theo hướng dự đoán từ trung tâm (target word) dựa trên các từ ngữ cảnh (context words) xung quanh nó. Cụ thể, với một chuỗi từ đầu vào $w_1, w_2, ..., w_T$, tại một thời điểm t, mục tiêu của mô hình là dự đoán từ w_t dựa trên các từ lân cận $\{w_{t-c}, ..., w_{t-1}, w_{t+1}, ..., w_{t+c}\}$, trong đó c là kích thước cửa sổ ngữ cảnh (context window size).

2.2.9.2 Kiến trúc mô hình

Kiến trúc của CBOW tương đối đơn giản và hiệu quả. Các bước chính bao gồm:

- **Nhúng ngữ cảnh:** Mỗi từ ngữ cảnh được ánh xạ thành một vector nhúng (embedding vector) thông qua một ma trận tra cứu từ điển (lookup table).
- **Tổng hợp ngữ cảnh:** Các vector nhúng của các từ ngữ cảnh được tính trung bình hoặc cộng lại để tạo ra một vector duy nhất biểu diễn ngữ cảnh.

- **Dự đoán từ mục tiêu:** Vector tổng hợp ngữ cảnh được đưa qua một lớp tuyến tính và hàm softmax để dự đoán xác suất phân phối trên toàn bộ từ vựng, với mục tiêu là tối đa hóa xác suất của từ mục tiêu thực sự.

Toán học của hàm mất mát trong CBOW được định nghĩa như sau:

$$\mathscr{L} = -\log P(w_t | w_{t-c}, ..., w_{t-1}, w_{t+1}, ..., w_{t+c})$$

Trong đó xác suất điều kiện được tính bằng:

$$P(w_t|\text{context}) = \frac{\exp(\mathbf{v}_{w_t}^\top \mathbf{h})}{\sum_{w \in V} \exp(\mathbf{v}_w^\top \mathbf{h})}$$

Với:

- $\mathbf{h}$ là vector ngữ cảnh tổng hợp.
- $\mathbf{v}_w$ là vector nhúng của từ w.
- V là tập từ vựng.

2.2.9.3 Ưu điểm và ứng dụng

CBOW có một số ưu điểm nổi bật như:

- Tốc độ huấn luyện nhanh do kiến trúc đơn giản.
- Hiệu quả trong việc học các biểu diễn từ mượt mà và liên tục (continuous representations).
- Phù hợp với các bộ dữ liệu lớn, đặc biệt khi sử dụng kỹ thuật negative sampling hoặc hierarchical softmax để tối ưu hóa quá trình tính toán.

CBOW được ứng dụng rộng rãi trong nhiều hệ thống xử lý ngôn ngữ tự nhiên như phân loại văn bản, gán nhãn thực thể (NER), phân tích cảm xúc, và đặc biệt là khởi tạo vector từ trong các mô hình học sâu phức tạp hơn như LSTM hoặc Transformer.

Chương 3

Các nghiên cứu về phân tích cảm xúc

3.1 Giới thiệu

Phân tích cảm xúc (Sentiment Analysis) là một lĩnh vực quan trọng trong xử lý ngôn ngữ tự nhiên (NLP), nhằm xác định và phân loại cảm xúc thể hiện trong văn bản thành các loại như tích cực, tiêu cực hoặc trung lập. Đối với tiếng Việt, việc phân tích cảm xúc gặp nhiều thách thức do đặc điểm ngôn ngữ phức tạp, cấu trúc cú pháp đa dạng và sự phong phú về từ vựng. Trong những năm gần đây, nhiều nghiên cứu đã được thực hiện để cải thiện hiệu quả của các mô hình phân tích cảm xúc tiếng Việt.Phân tích cảm xúc (Sentiment Analysis) là một lĩnh vực quan trọng trong xử lý ngôn ngữ tự nhiên (NLP), nhằm xác định và phân loại cảm xúc thể hiện trong văn bản thành các loại như tích cực, tiêu cực hoặc trung lập. Đối với tiếng Việt, việc phân tích cảm xúc gặp nhiều thách thức do đặc điểm ngôn ngữ phức tạp, cấu trúc cú pháp đa dạng và sự phong phú về từ vựng. Các yếu tố như ngữ nghĩa, ngữ cảnh và các biểu thức cảm xúc đặc trưng của văn hóa Việt Nam cũng làm cho việc phát triển các mô hình phân tích cảm xúc trở nên khó khăn hơn.

Trong những năm gần đây, nhiều nghiên cứu đã được thực hiện để cải thiện hiệu quả của các mô hình phân tích cảm xúc tiếng Việt. Các phương pháp hiện tại bao gồm việc sử dụng từ điển cảm xúc, mô hình học máy và học sâu (deep learning) để nâng cao độ chính xác trong việc phân loại cảm xúc. Một số nghiên cứu đã áp dụng các mô hình như Recurrent Neural Networks (RNN) và Long Short-Term Memory (LSTM) để xử lý các văn bản dài và phức tạp, cho phép nhận diện các sắc thái cảm xúc tinh tế hơn trong ngôn ngữ. Ngoài ra, việc phát triển các bộ dữ liệu lớn và đa dạng cũng là một yếu tố quan trọng giúp cải thiện khả năng của các mô hình phân tích cảm xúc. Các bộ dữ liệu này không chỉ bao gồm các bình luận trên mạng xã hội mà còn các đánh giá sản phẩm, phản hồi từ khách hàng và các bài viết trên blog, từ đó cung cấp một cái nhìn toàn diện hơn về cảm xúc của người dùng. Tóm lại, mặc dù phân tích cảm xúc trong tiếng Việt còn nhiều thách thức, nhưng với sự phát triển không ngừng của công nghệ và nghiên cứu, chúng ta có thể kỳ vọng vào những bước tiến đáng kể trong lĩnh vực này trong tương lai gần.

3.2 Các công trình nghiên cứu về tiền xử lý

Tiền xử lý dữ liệu đóng vai trò quan trọng trong phân tích cảm xúc, đặc biệt với dữ liệu tiếng Việt chứa nhiều biến thể ngôn ngữ và ký tự đặc biệt. Khang và cộng sự (2020)[9] đã nghiên cứu tác động của tiền xử lý đối với dữ liệu từ mạng xã hội Việt Nam, cho thấy việc áp dụng kỹ thuật này giúp cải thiện độ chính xác lên 4.66% so với các phương pháp trước đó. Các bộ dữ liệu phổ biến như VLSP[10], UIT-VSFC[11], và HSA cũng được nhấn mạnh trong các nghiên cứu gần đây, đặc biệt trong việc xử lý văn bản ngắn và đa dạng về cảm xúc. Ví dụ, bộ dữ liệu UIT-VSFC chứa phản hồi của sinh viên, trong khi VLSP và HSA tập trung vào đánh giá sản phẩm và dịch vụ.

Ngoài ra, việc áp dụng các kỹ thuật tiền xử lý dữ liệu còn được kết hợp với các phương pháp tăng cường dữ liệu để nâng cao hiệu suất của mô hình phân tích cảm xúc. Luu và cộng sự (2020)[12] đã áp dụng các kỹ thuật tăng cường dữ liệu như chèn từ, thay thế từ và xóa từ để mở rộng tập dữ liệu huấn luyện, nhằm giảm chi phí gán nhãn và cải thiện hiệu suất mô hình phân tích cảm xúc tiếng Việt. Kết quả cho thấy việc áp dụng các kỹ thuật này giúp tăng khoảng 1.5% điểm F1-macro trên cả hai tập dữ liệu VLSP2019 và UIT-VSFC. Điều này chứng minh rằng sự kết hợp giữa tiền xử lý dữ liệu và các kỹ thuật tăng cường dữ liệu có thể mang lại những cải thiện đáng kể trong việc phân tích cảm xúc tiếng Việt.

Trong khóa luận của Lê Si Lắc (2021) tại Trường Đại học Công nghệ Thông tin TP.HCM: "Nghiên cứu bài toán phân tích cảm xúc của người dùng" đề cập đến quy trình tiền xử lý dữ liệu như chuẩn hóa văn bản, loại bỏ từ dừng, và ứng dụng các mô hình ngôn ngữ như BERT để phân tích cảm xúc trên các bộ dữ liệu VLSP, HSA, UIT-VSFC. Nghiên cứu cũng khảo sát hiệu quả của các kỹ thuật làm giàu dữ liệu (data augmentation) như thay thế từ đồng nghĩa và kết hợp với word embeddings.

Tóm lại, tiền xử lý dữ liệu và việc áp dụng các kỹ thuật tăng cường dữ liệu là hai yếu tố quan trọng quyết định đến độ chính xác và hiệu quả của quá trình phân tích cảm xúc tiếng Việt. Thông qua việc áp dụng các kỹ thuật này, các nhà nghiên cứu có thể đạt được kết quả tốt hơn trong việc phân tích cảm xúc, đặc biệt là trong bối cảnh ngôn ngữ tiếng Việt đa dạng và phong phú. Sự phát triển của các kỹ thuật tiền xử lý dữ liệu mới sẽ tiếp tục đóng góp vào việc nâng cao chất lượng của các nghiên cứu trong lĩnh vực này.

3.3 Công trình nghiên cứu về các mô hình giải quyết bài toán phân tích cảm xúc trên miền dữ liệu tiếng Việt

Phân tích cảm xúc trong văn bản là một lĩnh vực quan trọng trong xử lý ngôn ngữ tự nhiên (NLP), đặc biệt trong bối cảnh bùng nổ dữ liệu từ mạng xã hội và các nền tảng trực tuyến. Việc hiểu và phân loại cảm xúc từ văn bản giúp các tổ chức và doanh nghiệp nắm bắt được phản hồi của người dùng, từ đó cải thiện sản phẩm và dịch vụ. Trong những năm gần

đây, nhiều nghiên cứu đã được thực hiện nhằm phát triển các mô hình hiệu quả cho phân tích cảm xúc tiếng Việt. Một trong những hướng tiếp cận phổ biến là sử dụng các mô hình học sâu như CNN, LSTM, BERT và các biến thể của chúng.

Nghiên cứu của Xi Ouyang và cộng sự [5] đã đề xuất một khung mô hình kết hợp Word2Vec với mạng nơ-ron tích chập (CNN), đạt độ chính xác 45,4% trên tập dữ liệu tiếng Việt, vượt trội so với các mô hình RNN[13] và MV-RNN[14]. Mạng CNN được ưa chuộng trong các nhiệm vụ phân loại văn bản ngắn nhờ khả năng trích xuất các đặc trưng quan trọng thông qua các lớp tích chập và tổng hợp. Tuy nhiên, CNN có hạn chế trong việc xử lý các mối quan hệ tuần tự dài hạn trong văn bản. Để khắc phục điều này, các mô hình kết hợp như ConvLSTM đã được đề xuất, tận dụng ưu điểm của cả CNN và LSTM để cải thiện hiệu suất phân tích cảm xúc.

Bên cạnh đó, sự xuất hiện của các mô hình ngôn ngữ tiền huấn luyện như BERT[15] và các biến thể dành riêng cho tiếng Việt như PhoBERT[6] đã mở ra những cơ hội mới cho phân tích cảm xúc. Nghiên cứu của Trần Khải Thiện và Tiếu Phùng Mai Sương đã khảo sát và phân tích các hướng tiếp cận phân tích cảm xúc trong tiếng Việt, nhấn mạnh vai trò của các mô hình ngôn ngữ hiện đại trong việc nâng cao hiệu quả phân tích.

Gần đây, một nghiên cứu khác đã áp dụng mô hình BERT kết hợp với kiến trúc đa kênh CNN-GRU [16] để phân tích cảm xúc trên phản hồi học viên, cho thấy sự cải thiện hiệu suất tại chỉ số F1-Score (Macro) so với các nghiên cứu trước đó.

Ngoài ra, các kỹ thuật làm tăng dữ liệu (data augmentation) cũng được áp dụng để cải thiện hiệu suất của mô hình phân tích cảm xúc. Hương và cộng sự [17] đã đề xuất mô hình làm tăng dữ liệu văn bản dựa trên các câu bình luận áp dụng cho ngôn ngữ tiếng Việt, nhằm giải quyết vấn đề thiếu hụt dữ liệu huấn luyện.

Những phát hiện từ các nghiên cứu trên đã cung cấp cơ sở vững chắc cho việc phát triển các mô hình phân tích cảm xúc tiếng Việt, đặc biệt trong việc xử lý các bình luận và phản hồi từ người dùng. Việc kết hợp các kỹ thuật tiền xử lý dữ liệu, mô hình học sâu, và các phương pháp làm tăng dữ liệu hứa hẹn sẽ nâng cao khả năng phân tích cảm xúc trong các ứng dụng thực tiễn.

Chương 4

Phân tích tập dữ liệu

4.1 Định nghĩa bộ dữ liệu

Bộ dữ liệu VLSP (Vietnamese Language and Speech Processing) là một trong những tập dữ liệu quan trọng và được sử dụng rộng rãi trong nghiên cứu phân tích cảm xúc tiếng Việt. Thông qua quá trình tìm hiểu và phân tích trên dữ liệu, em nhận thấy rằng: bộ dữ liệu VLSP bao gồm các bình luận ngắn được thu thập từ mạng xã hội, tập trung vào các sản phẩm điện tử. Mỗi bình luận được gán một nhãn cảm xúc tổng thể, thuộc một trong ba loại:

- Tích cực (Positive): Được gán nhãn 1
- Trung tính (Neutral): Được gán nhãn 0
- Tiêu cực (Negative): Được gán nhãn -1

Mỗi mục dữ liệu trong bộ này bao gồm hai thuộc tính chính:

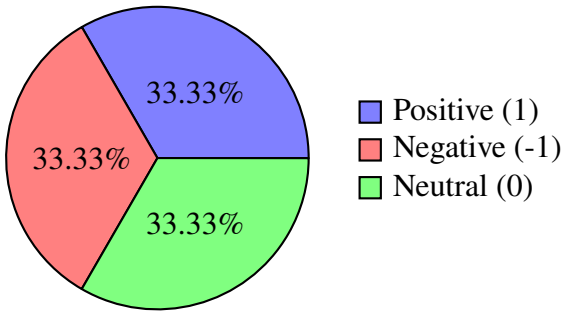
- Class: Nhãn cảm xúc của bình luận
- Data: Nội dung văn bản của bình luận

Tổng số lượng bình luận trong bộ dữ liệu này được chia thành hai tập: tập huấn luyện (Train) và tập kiểm tra (Test), với tổng số lượng nhãn cụ thể như bảng 4.3:

Nhãn	1	-1	0
Số lượng	2050	2050	2050

Bảng 4.1: Thông tin bộ dữ liệu VLSP

Từ thống kê số lượng nhãn trong bộ dữ liệu, chúng ta có thể thấy rõ ràng rằng các nhãn được phân bố rất cân bằng nhau. Điều này mang lại lợi ích đáng kể trong quá trình huấn luyện mô hình, vì không có sự chênh lệch đáng kể giữa các loại dữ liệu. Khi các dữ liệu



Hình 4.1: Biểu đồ tròn thể hiện phân bố cảm xúc trong bộ dữ liệu VLSP

được cân bằng, việc huấn luyện mô hình trở nên hiệu quả hơn, vì mô hình có thể học được các mẫu và mối quan hệ từ mỗi loại dữ liệu một cách đồng đều.

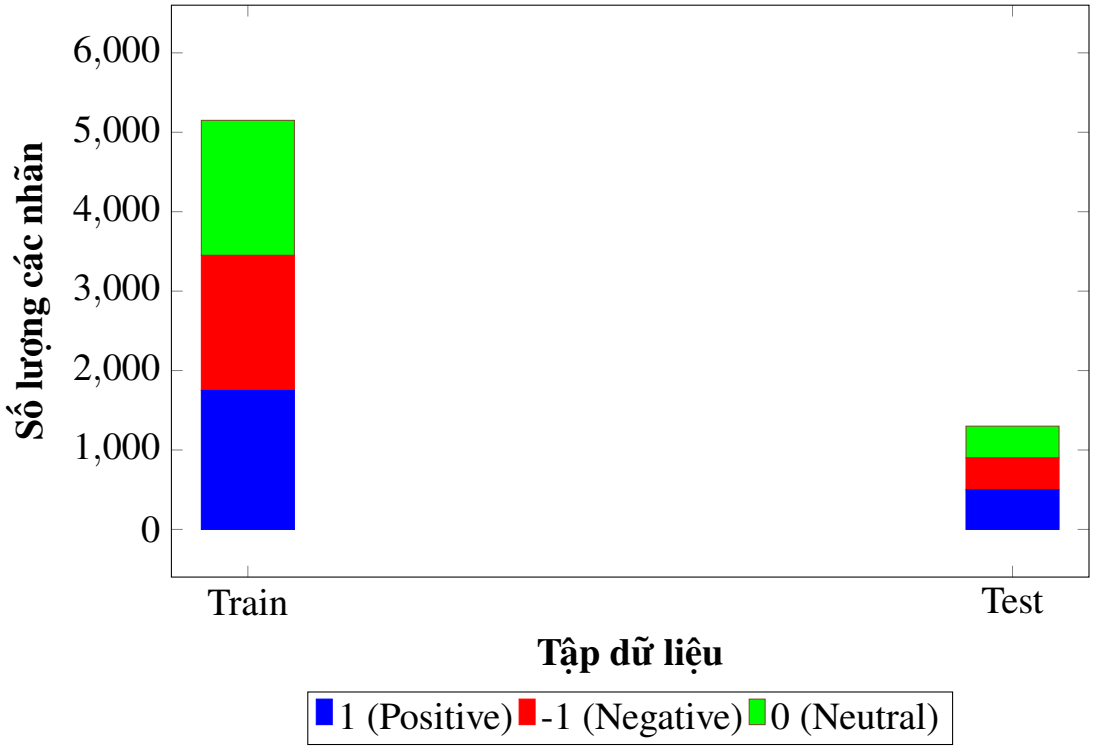
Ngoài ra, các dữ liệu cân bằng cũng thuận lợi hơn trong việc lựa chọn các chỉ số đánh giá mô hình. Khi các chỉ số đánh giá được tính toán trên các dữ liệu không cân bằng, chúng có thể bị ảnh hưởng bởi sự chênh lệch giữa các loại dữ liệu, dẫn đến kết quả đánh giá không chính xác. Do đó, việc có các dữ liệu cân bằng giúp chúng ta có thể đánh giá mô hình một cách công bằng và chính xác hơn. Tập dữ liệu được chia thành hai tập: tập huấn luyện (Train) và tập kiểm tra (Test). Việc chia tập dữ liệu này giúp chúng ta có thể đánh giá mô hình trên các dữ liệu chưa từng được mô hình nhìn thấy trước đó, giúp kiểm tra khả năng tổng quát hóa của mô hình. Thông tin chi tiết về việc chia tập dữ liệu sẽ được trình bày ở bảng 4.2:

Nhãn	**Tổng số nhãn**	**Train**	**Test**
1	2050	1700	350
-1	2050	1700	350
0	2050	1700	350
Tổng cộng	6150	5100	1050

Bảng 4.2: Số nhãn được chia ở tập Train và Test

Từ biểu đồ, ta có thể thấy được trực quan về bộ dữ liệu được chia ra thành 3 tập Train:Test đều nhau về số lượng các nhãn. Cụ thể, tập dữ liệu được chia thành 2 phần chính: tập huấn luyện (Train) và tập kiểm tra (Test). Mỗi phần đều có số lượng các nhãn tương đương nhau, bao gồm các nhãn Positive (1), Negative (-1) và Neutral (0). Sự cân bằng này trong bộ dữ liệu giúp ích rất nhiều trong quá trình huấn luyện mô hình. Khi các nhãn được phân bố đều nhau, mô hình có thể học được các mẫu và mối quan hệ từ mỗi loại dữ liệu một cách đồng đều. Điều này giúp mô hình tránh bị thiên vị về một loại dữ liệu nhất định và tăng khả năng tổng quát hóa. Ngoài ra, số lượng các nhãn trong bộ dữ liệu đều bằng nhau cũng thuận lợi cho quá trình huấn luyện. Khi các nhãn có số lượng bằng nhau, việc tính toán các chỉ số đánh giá mô hình như độ chính xác, độ nhớ, độ chính xác trung bình,... sẽ trở nên dễ dàng

và chính xác hơn. Điều này giúp chúng ta có thể đánh giá mô hình một cách công bằng và chính xác hơn. Tập dữ liệu được chia thành 2 phần đều nhau về số lượng các nhãn cũng giúp chúng ta có thể kiểm tra khả năng tổng quát hóa của mô hình. Khi mô hình được huấn luyện trên tập dữ liệu Train và kiểm tra trên tập dữ liệu Test, chúng ta có thể đánh giá khả năng của mô hình trong việc dự đoán các dữ liệu mới mà chưa từng được mô hình nhìn thấy trước đó.



Hình 4.2: Biểu đồ thể hiện số lượng nhãn trong tập dữ liệu

4.2 Phân tích đặc điểm bộ dữ liệu

Với mong muốn tìm hiểu về các tính chất đặc trưng của bộ dữ liệu VLSP, em đã thực hiện các phân tích cơ bản về đặc điểm của bộ dữ liệu như: độ dài câu, độ dài câu trung bình, các từ có sự xuất hiện nhiều, . . . Những phân tích này giúp em có cái nhìn tổng quan về đặc điểm của bộ dữ liệu VLSP và góp phần tìm ra các cách tiếp cận và lựa chọn các mô hình thử nghiệm phù hợp với tính chất của bộ dữ liệu.

Đầu tiên, em tiến hành những thống kê cơ bản về bộ dữ liệu VLSP. em thực hiện kiểm tra các thống kê về độ dài trung bình của câu, kích thước bộ từ vựng (từ) và các đặc điểm khác của dữ liệu. Những thống kê này giúp em hiểu rõ hơn về cấu trúc và nội dung của bộ dữ liệu. Một trong những phân tích quan trọng đầu tiên là kiểm tra độ dài trung bình của câu trong bộ dữ liệu. Độ dài trung bình của câu giúp em hiểu rõ hơn về mức độ phức tạp của ngôn ngữ sử dụng trong bộ dữ liệu. Nếu độ dài trung bình của câu quá ngắn hoặc quá dài, điều này có thể ảnh hưởng đến việc lựa chọn mô hình và cách tiếp cận phù hợp.

Tiếp theo, em tiến hành phân tích kích thước bộ từ vựng (từ) trong bộ dữ liệu. Kích thước bộ từ vựng giúp em hiểu rõ hơn về mức độ đa dạng của ngôn ngữ sử dụng trong bộ dữ liệu. Nếu bộ từ vựng quá nhỏ hoặc quá lớn, điều này có thể ảnh hưởng đến việc lựa chọn mô hình và cách tiếp cận phù hợp. Ngoài ra, em cũng thực hiện phân tích về các từ có sự xuất hiện nhiều trong bộ dữ liệu. Những từ có sự xuất hiện nhiều giúp em hiểu rõ hơn về chủ đề và nội dung chính của bộ dữ liệu. Điều này cũng giúp em xác định các từ khóa quan trọng và có thể được sử dụng để cải thiện hiệu suất của mô hình. Sau khi đã thực hiện những phân tích cơ bản này, em đã có cái nhìn tổng quan về đặc điểm của bộ dữ liệu VLSP. Những thông tin này đã giúp em tìm ra các cách tiếp cận và lựa chọn các mô hình thử nghiệm phù hợp với tính chất của bộ dữ liệu. Điều này giúp em có thể thực hiện các thí nghiệm và đánh giá hiệu suất của mô hình một cách chính xác và hiệu quả hơn.

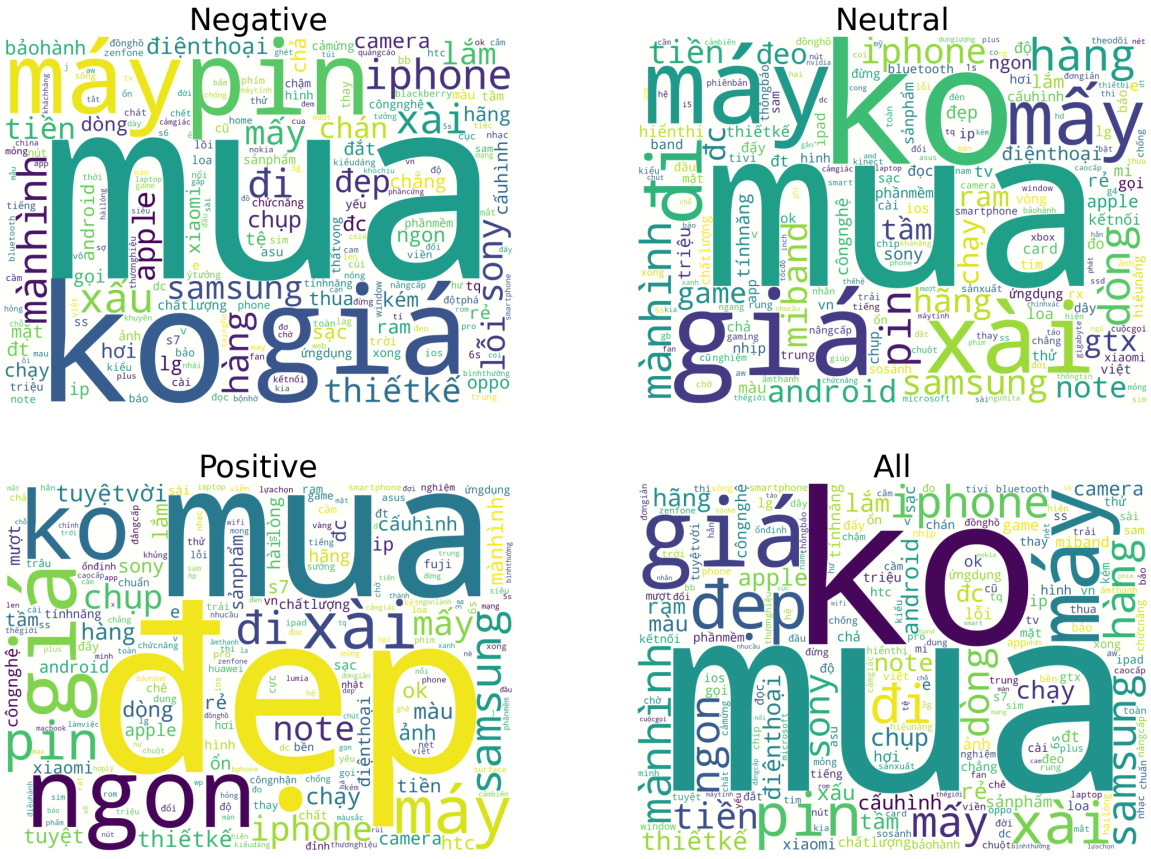
Độ dài trung bình	64.88
Kích thước bộ từ vựng	8,795

Bảng 4.3: Thống kê độ dài trung bình câu và kích thước bộ dữ liệu

Độ dài câu trung bình của bộ dữ liệu là 64.88 từ, cho thấy rằng độ dài câu bình luận tương đối dài. Điều này có thể giúp cho các mô hình học máy "hiểu" và "học" được hết thông tin trong câu một cách dễ dàng hơn. Tuy nhiên, mặc dù độ dài câu trung bình là một yếu tố thuận lợi, nhưng việc huấn luyện các mô hình học máy vẫn đòi hỏi một số lượng khổng lồ các phép tính toán và số lần cập nhật tham số. Do đó, em chắc chắn sẽ phải tốn nhiều thời gian và tài nguyên cho giai đoạn xây dựng mô hình.

Một đặc điểm quan trọng của tiếng Việt là các từ không phân biệt bằng khoảng trắng, và có những cụm từ phải đi liền với nhau mới thể hiện được ý nghĩa, chẳng hạn như "khách hàng", "ứng dụng", "thông tin". Chính vì điều đó, em tiến hành tách từ, sau đó mới thống kê và đo đạc kích thước bộ từ vựng của bộ dữ liệu. Kết quả cho thấy kích thước từ vựng trong bộ dữ liệu VLSP gồm 8.795 từ và cụm từ.

Ngoài việc đánh giá các đặc trưng cơ bản trên bộ dữ liệu, em còn thực hiện phân tích về tần suất xuất hiện của các từ trên phương diện nhãn. Mục đích của phân tích này là nhằm tìm ra đặc trưng của các nhãn hiện có trong bộ dữ liệu. Bằng cách phân tích tần suất xuất hiện của các từ, em có thể xác định được các từ khóa quan trọng và mối quan hệ giữa các từ với các nhãn. Điều này sẽ giúp em có cái nhìn sâu sắc hơn về bộ dữ liệu và có thể lựa chọn mô hình học máy phù hợp nhất cho nhiệm vụ phân tích cảm xúc trên bộ dữ liệu VLSP.



Hình 4.3: Word cloud của bộ dữ liệu VLSP

Từ các dữ liệu được phân tích, có thể thấy rằng các từ thường xuất hiện như “đẹp”, “ngon”, “giá”, . . . là các từ xuất hiện thường xuyên và được nhắc đến nhiều nhất ở cả ba nhãn và toàn bộ dữ liệu. Sự xuất hiện thường xuyên của các từ này cho thấy rằng bộ dữ liệu VLSP tập trung vào các nhận xét về sản phẩm, cụ thể là thiết bị di động. Các từ như “đẹp” và “ngon” thường được sử dụng để mô tả các đặc điểm về hình dạng, thiết kế và chất lượng của sản phẩm. Trong khi đó, từ “giá” thường được sử dụng để đề cập đến giá cả của sản phẩm. Sự xuất hiện thường xuyên của các từ này cho thấy rằng người dùng thường đánh giá sản phẩm dựa trên các yếu tố như thiết kế, chất lượng và giá cả. Điều này cũng cho thấy rằng bộ dữ liệu VLSP là một bộ dữ liệu về phân tích cảm xúc của người dùng đối với sản phẩm thiết bị di động. Các nhận xét trong bộ dữ liệu này thường xoay quanh các đánh giá về sản phẩm, bao gồm cả các đánh giá tích cực và tiêu cực. Bằng cách phân tích các từ thường xuất hiện trong bộ dữ liệu, có thể xác định được các chủ đề chính và các yếu tố quan trọng mà người dùng quan tâm khi đánh giá sản phẩm. Điều này sẽ giúp ích trong việc xây dựng các mô hình phân tích cảm xúc hiệu quả và chính xác hơn.

Chương 5

Thực nghiệm và đánh giá

5.1 Môi trường thực nghiệm

Em thực hiện đề tài sử Anaconda, Python 3.11 và IDE là Pycharm. Một số thư viện Deep Learning quan trọng chính là keras, tensorflow 2.2.0 và ngoài ra còn môt số thư viện cơ bản khác như numpy, pandas, cv2, os...

5.2 Tiền xử lý dữ liệu

Trong quá trình xử lý văn bản đầu vào, đặc biệt là các mô tả và chính sách sản phẩm, em nhận thấy dữ liệu có chứa nhiều điểm bất thường và thiếu nhất quán về mặt định dạng. Do đó, em tiến hành một loạt bước tiền xử lý nhằm chuẩn hóa và làm sạch dữ liệu, giúp mô hình học máy tiếp cận thông tin một cách chính xác và hiệu quả hơn. Các bước cụ thể như sau:

5.2.1 Xóa khoảng trắng dư thừa

Khi phân tích cấu trúc câu để thực hiện việc tách câu trong văn bản, em phát hiện có nhiều trường hợp văn bản chứa các khoảng trắng dư thừa – đặc biệt là do người dùng nhập liệu không thống nhất. Những khoảng trắng không cần thiết này có thể gây nhiễu trong quá trình phân tích và trích xuất thông tin, đồng thời làm giảm hiệu quả của các mô hình xử lý ngôn ngữ tự nhiên. Vì vậy, em đã triển khai bước làm sạch để loại bỏ các khoảng trắng dư, đảm bảo tính đồng nhất cho đầu vào.

Ví dụ:

Trước: “...điều quý vị muốn biết...”
Sau: “...điều quý vị muốn biết...”

5.2.2 Chuẩn hóa văn bản về chữ thường

Tất cả các từ trong tập dữ liệu được chuyển về dạng viết thường nhằm tránh tình trạng cùng một từ nhưng lại bị xử lý thành hai thực thể khác nhau do sự khác biệt về chữ hoa – chữ thường. Đây là một bước quan trọng trong quá trình chuẩn hóa vì Python (hoặc các công cụ NLP) thường phân biệt chữ hoa và chữ thường khi xử lý từ vựng.

Ví dụ: "An Giang" $\rightarrow$ "an giang"

3. Chuẩn hóa Unicode

Với các văn bản tiếng Việt có dấu, việc không đồng nhất mã hóa Unicode có thể dẫn đến lỗi trong quá trình so sánh, phân tích hoặc trích xuất dữ liệu. Chẳng hạn, hai chuỗi có vẻ giống nhau về mặt hiển thị nhưng lại khác về mã hóa bên trong, khiến việc kiểm tra tính bằng nhau trả về kết quả sai.

Ví dụ: "hiểu" (mã hóa 1) $\neq$ "hiểu" (mã hóa 2) $\rightarrow$ False khi so sánh

Do đó, em đã chuyển đổi toàn bộ văn bản về dạng Unicode chuẩn (*NFC – Normal Form Composed*) để đảm bảo tính nhất quán trong toàn bộ quá trình xử lý.

5.2.3 Chuẩn hóa kiểu gõ dấu tiếng Việt

Tiếng Việt hiện có nhiều cách gõ dấu khác nhau, trong đó phổ biến nhất là kiểu gõ dấu cũ và kiểu mới (theo chuẩn Unicode tổ hợp). Để đảm bảo đồng nhất văn bản đầu vào, em thực hiện chuyển đổi tất cả các kiểu gõ dấu về một chuẩn duy nhất – cụ thể là kiểu gõ dấu cũ. Việc này không chỉ giúp mô hình hiểu và xử lý tốt hơn, mà còn tránh được các lỗi phân tách từ không mong muốn.

5.2.4 Tách từ (Tokenization)

Do đặc trưng ngôn ngữ tiếng Việt là từ ghép (2–3 từ đơn tạo thành một nghĩa hoàn chỉnh), nên em đã sử dụng thư viện `PyVi` để tách từ chính xác hơn so với phương pháp tách theo khoảng trắng thông thường. Việc này rất quan trọng khi sử dụng các mô hình học sâu hoặc vector hóa từ.

Ví dụ:

Câu gốc: "em là sinh viên"
Sau tách từ: ["em", "là", "sinh_viên"]

5.2.5 Loại bỏ stopwords (từ dừng)

Stopwords là những từ xuất hiện với tần suất cao nhưng không mang nhiều ý nghĩa ngữ nghĩa, chẳng hạn như "bị", "của", "biết bao", v.v. Việc giữ lại các từ này có thể làm nhiễu

mô hình phân loại và làm giảm hiệu quả của mô hình học máy. Do đó, em sử dụng một danh sách từ dừng tiếng Việt phổ biến để loại bỏ chúng khỏi câu.

Ví dụ:

Trước: “em bị đau chân”
Sau: “em đau chân”

Để đảm bảo tính nhất quán giữa dữ liệu và danh sách từ dừng, em đã thực hiện tách từ trước khi loại bỏ stopwords. Đồng thời, các từ trong danh sách stopwords cũng được chuyển về dạng nối từ bằng dấu gạch dưới (VD: “học_sinh”), tương tự như dữ liệu sau khi token hóa. Nhờ đó, việc đối chiếu và loại bỏ trở nên chính xác và hiệu quả hơn.

xTrong quá trình xử lý dữ liệu cho một bài toán về NLP, em đã lựa chọn áp dụng mô hình CBOW (Continuous Bag of Words) nhằm tạo ra embedding matrix một cách hiệu quả và phù hợp với đặc trưng của tập dữ liệu đang sử dụng. Mô hình CBOW này đã được huấn luyện từ trước bởi thầy, và em chỉ việc tải lại để sử dụng, điều này không những tiết kiệm thời gian huấn luyện mà còn đảm bảo chất lượng của vector từ đã được học.

Việc xây dựng embedding matrix bắt đầu từ công đoạn tiền xử lý dữ liệu. em tiến hành làm sạch văn bản bằng cách loại bỏ các ký tự đặc biệt, con số và đưa toàn bộ nội dung về chữ thường nhằm thống nhất định dạng. Đồng thời, những từ dừng (stopwords) không mang nhiều ý nghĩa ngữ nghĩa cũng được loại bỏ, giúp dữ liệu trở nên gọn gàng và tập trung hơn vào các từ khóa quan trọng. Khi đã có một tập văn bản sạch và chuẩn hóa, em đưa dữ liệu này vào mô hình CBOW để tiến hành huấn luyện.

CBOW hoạt động bằng cách dự đoán một từ dựa vào ngữ cảnh xung quanh nó, từ đó hình thành các vector đại diện cho từng từ một cách có ý nghĩa ngữ nghĩa. Điều đáng chú ý là các vector được sinh ra từ mô hình này không chỉ đơn thuần là những dãy số ngẫu nhiên, mà thực sự mang trong mình thông tin về mối liên hệ giữa các từ trong văn cảnh cụ thể. Sau khi mô hình hoàn tất quá trình học, em tiến hành trích xuất các vector từ đã học được để xây dựng nên embedding matrix. Mỗi từ trong từ vựng được biểu diễn bằng một vector liên tục và có khả năng mô tả phần nào vai trò và ý nghĩa của từ đó trong ngôn ngữ.

Embedding matrix tạo ra theo cách này mang lại nhiều lợi ích thiết thực. Trước hết, vì các vector được học từ chính tập dữ liệu huấn luyện, nên chúng phản ánh rất sát ngữ cảnh thực tế của dữ liệu, làm tăng độ phù hợp khi áp dụng vào các mô hình học máy sau này. Ngoài ra, việc biểu diễn từ bằng vector còn giúp mô hình hiểu rõ hơn các mối quan hệ ngữ nghĩa giữa các từ, kể cả những từ không xuất hiện thường xuyên, từ đó nâng cao khả năng khái quát hóa. Ma trận embedding chỉ được em dùng ở CNN, LSTM, RGU, RNN và các mô hình kết hợp giữa các kiến trúc trên.

5.2.6 Chuẩn bị dữ liệu để train cho model handwriting

Từ bộ dữ liệu đã gộp em chia bộ dữ liệu ngày làm hai với tỉ lệ 8:2 tương tứng cho tập train và tập test.

Tên tập dữ liệu	Tổng số ảnh
Tập train	74,160
Tập test	18,540

Bảng 5.1: Thông tin dữ liệu để huấn luyện model handwriting

5.3 Quá trình thực nghiệm và kiến trúc mô hình

5.3.1 Các mô hình học sâu truyền thống

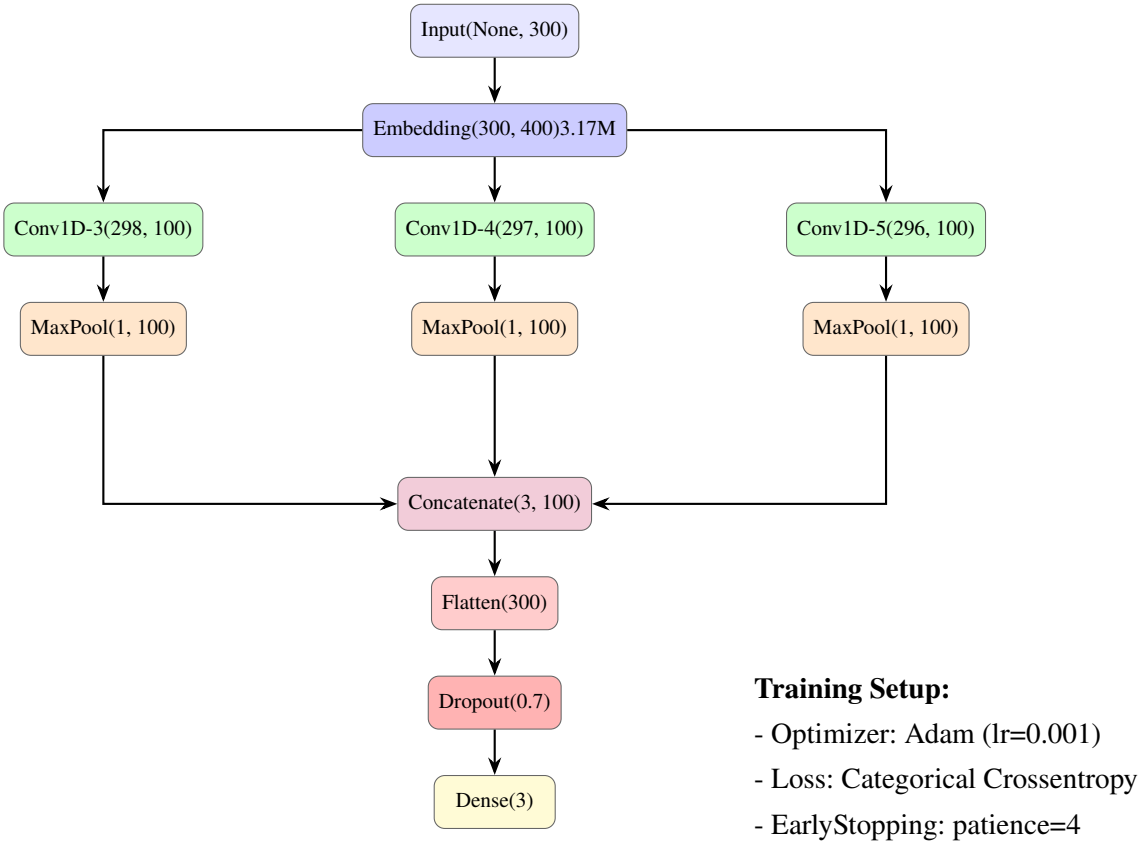
Trước khi đưa vào quá trình huấn luyện, dữ liệu được xử lý qua một bước tiền xử lý bao gồm chuẩn hoá văn bản, tách từ, và mã hoá theo mô hình **CBOW**. Trong phương pháp này, mỗi từ trong dữ liệu được biểu diễn bằng một vector thông qua việc học các ngữ cảnh xung quanh. CBOW giúp mô hình học được mối quan hệ ngữ nghĩa giữa các từ, đồng thời tạo ra các vector nhúng (embedding) có ý nghĩa trước khi đưa vào mạng nơ-ron. Các vector từ đầu ra của mô hình CBOW sau đó được sử dụng như đầu vào cho các mô hình học sâu như CNN, LSTM và các biến thể kết hợp nhằm khai thác các đặc trưng không gian và ngữ cảnh của chuỗi văn bản.

5.3.1.1 CNN (code gốc của thầy) - CNN_Org

Mô hình được xây dựng dựa trên kiến trúc CNN, triển khai bằng `TensorFlow` và `Keras`. Tại tầng đầu vào, dữ liệu có kích thước `(sequence_length,)` được đưa qua lớp `Embedding`, giúp ánh xạ mỗi token thành một vector số để học đặc trưng ngữ nghĩa.

Tiếp theo, mô hình sử dụng ba lớp `Conv1D` song song với kích thước kernel lần lượt là 3, 4 và 5. Mỗi lớp tích chập gồm 100 bộ lọc, sử dụng hàm kích hoạt ReLU và được áp dụng chính quy hóa L2 với hệ số 0.01. Sau mỗi lớp tích chập là một lớp `MaxPooling1D` để giảm kích thước và giữ lại đặc trưng quan trọng.

Đầu ra của các lớp pooling được nối lại bằng lớp `Concatenate`, sau đó được làm phẳng (`Flatten`) và chuyển đổi thành vector một chiều (`Reshape`). Để tránh overfitting, mô hình áp dụng một lớp `Dropout` với tỷ lệ 0.7. Cuối cùng, lớp đầu ra là một `Dense` layer gồm 3 đơn vị với hàm kích hoạt `softmax`, phục vụ cho bài toán phân loại 3 lớp.



Kiến trúc của CNN_Org

5.3.1.2 LSTM - LSTM_Simple

Mô hình được triển khai dựa trên kiến trúc LSTM, sử dụng thư viện TensorFlow và Keras. Mục tiêu của mô hình là xử lý dữ liệu chuỗi có chiều dài xác định và học được các mối quan hệ phụ thuộc dài hạn giữa các phần tử trong chuỗi.

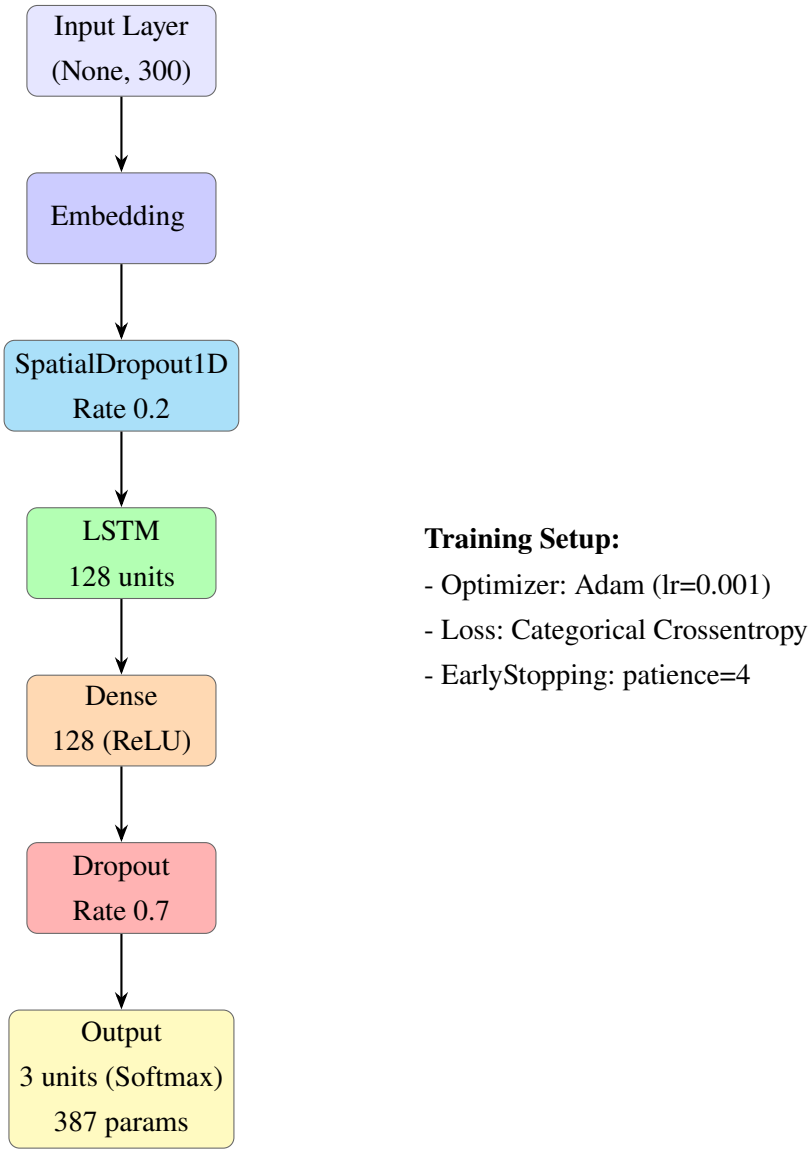
Tại tầng đầu vào (Input Layer), dữ liệu được đưa vào với kích thước (sequence_length,), trong đó sequence_length đại diện cho độ dài của mỗi chuỗi văn bản sau khi đã được mã hóa. Dữ liệu sau đó được truyền qua một lớp nhúng (Embedding Layer), được định nghĩa bởi embedding_layer, nơi mỗi token trong chuỗi được ánh xạ thành một vector số thực 400 chiều. Lớp nhúng này giúp mô hình học được các đặc trưng ngữ nghĩa từ các từ trong dữ liệu đầu vào.

Tiếp theo, mô hình áp dụng lớp SpatialDropout1D với tỷ lệ dropout là 0.2. Lớp này có tác dụng làm giảm overfitting bằng cách loại ngẫu nhiên toàn bộ các đặc trưng tại một thời điểm nhất định, thay vì loại ngẫu nhiên từng phần tử riêng lẻ như trong Dropout thông thường.

Dữ liệu sau đó được đưa vào một lớp LSTM với 128 đơn vị. Lớp LSTM này không trả về toàn bộ chuỗi đầu ra (return_sequences=False), mà chỉ giữ lại vector trạng thái cuối cùng, đại diện cho toàn bộ ngữ cảnh của chuỗi. Dropout được áp dụng cả ở input và recurrent

state thông qua các tham số dropout và recurrent_dropout, giúp tăng khả năng tổng quát hóa.

Sau lớp LSTM, đầu ra được truyền qua một lớp Dense với 128 đơn vị và hàm kích hoạt ReLU để học thêm các biểu diễn trừu tượng. Một lớp Dropout bổ sung được áp dụng sau lớp Dense để tiếp tục giảm thiểu overfitting. Cuối cùng, mô hình sử dụng một lớp Dense đầu ra với số đơn vị bằng số lớp là 3 và hàm kích hoạt softmax, nhằm đưa ra phân phối xác suất cho mỗi lớp trong bài toán phân loại đa lớp.



Kiến trúc của LSTM_Simple

5.3.1.3 LSTM nối tiếp LSTM (Stacked LSTM) - Stacked_LSTM

Mô hình được triển khai dựa trên kiến trúc **LSTM** sử dụng thư viện `TensorFlow` và `Keras`. Ở tầng đầu vào (*Input Layer*), dữ liệu được đưa vào với kích thước `(sequence_length,)`,

trong đó `sequence_length` đại diện cho độ dài của mỗi chuỗi dữ liệu.

Dữ liệu sau đó được truyền qua một lớp nhúng (**Embedding Layer**), do `embedding_layer` định nghĩa, nơi mỗi từ hoặc token trong chuỗi được ánh xạ thành một vector số thực. Việc sử dụng lớp nhúng giúp mô hình học được các đặc trưng ngữ nghĩa từ dữ liệu đầu vào. Tiếp theo, một lớp **SpatialDropout1D** với tỷ lệ 0.2 được áp dụng để giảm thiểu hiện tượng overfitting bằng cách loại bỏ ngẫu nhiên toàn bộ vector đặc trưng tại một số bước thời gian trong chuỗi.

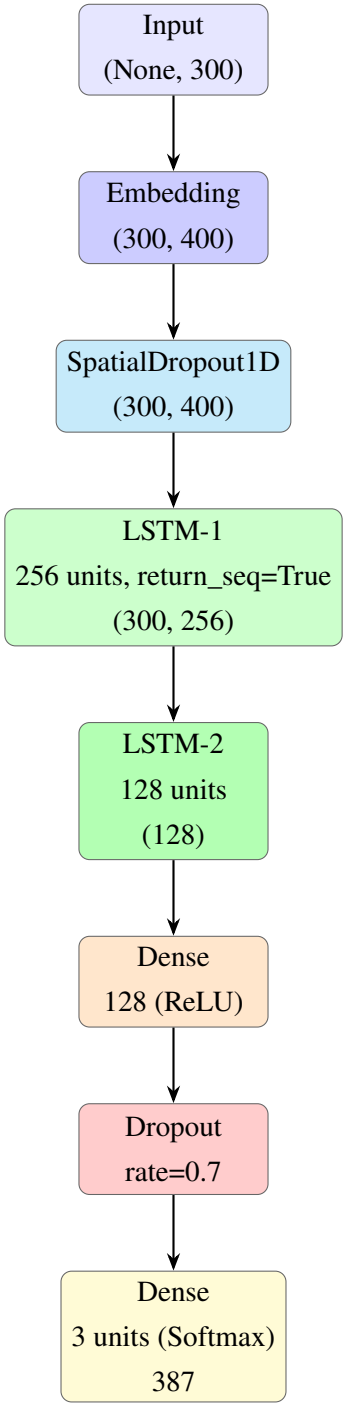
Phần chính của mô hình là hai lớp **LSTM được xếp chồng liên tiếp (stacked)**:

- Lớp `LSTM-1` có 256 đơn vị, với `return_sequences=True`, cho phép trả về toàn bộ chuỗi đầu ra để tiếp tục truyền vào lớp LSTM tiếp theo.
- Lớp `LSTM-2` có 128 đơn vị, với `return_sequences=False`, chỉ giữ lại trạng thái cuối cùng để đưa vào các lớp phía sau.

Cả hai lớp LSTM đều sử dụng `dropout` và `recurrent_dropout` nhằm nâng cao tính ổn định và khả năng tổng quát hóa của mô hình trong quá trình huấn luyện.

Sau khi trích xuất đặc trưng từ các tầng LSTM, đầu ra được truyền qua một lớp **Dense** với 128 đơn vị và hàm kích hoạt `ReLU`, giúp tiếp tục học các đặc trưng phi tuyến. Một lớp **Dropout** tiếp theo được áp dụng để giảm thiểu overfitting bằng cách ngẫu nhiên bỏ qua một phần đầu ra của lớp Dense trong quá trình huấn luyện.

Cuối cùng, mô hình sử dụng một lớp **Dense đầu ra** với số đơn vị bằng số lớp cần phân loại (`units=num_classes`) và hàm kích hoạt `softmax`, dùng để đưa ra phân phối xác suất cho mỗi lớp trong bài toán phân loại đa lớp.



Thông số huấn luyện:

- Optimizer: Adam (lr = 0.001)
- Loss: Categorical Crossentropy
- EarlyStopping: patience = 4

Kiến trúc của Stacked_LSTM

5.3.1.4 Parallel LSTM - Parallel_LSTM

Mô hình được triển khai dựa trên kiến trúc **LSTM** sử dụng thư viện `TensorFlow` và `Keras`, với mục tiêu xử lý dữ liệu chuỗi có độ dài cố định. Ở tầng đầu vào (*Input Layer*), dữ liệu được đưa vào với kích thước `(sequence_length,)`, trong đó `sequence_length`

đại diện cho độ dài của chuỗi văn bản sau khi được mã hóa.

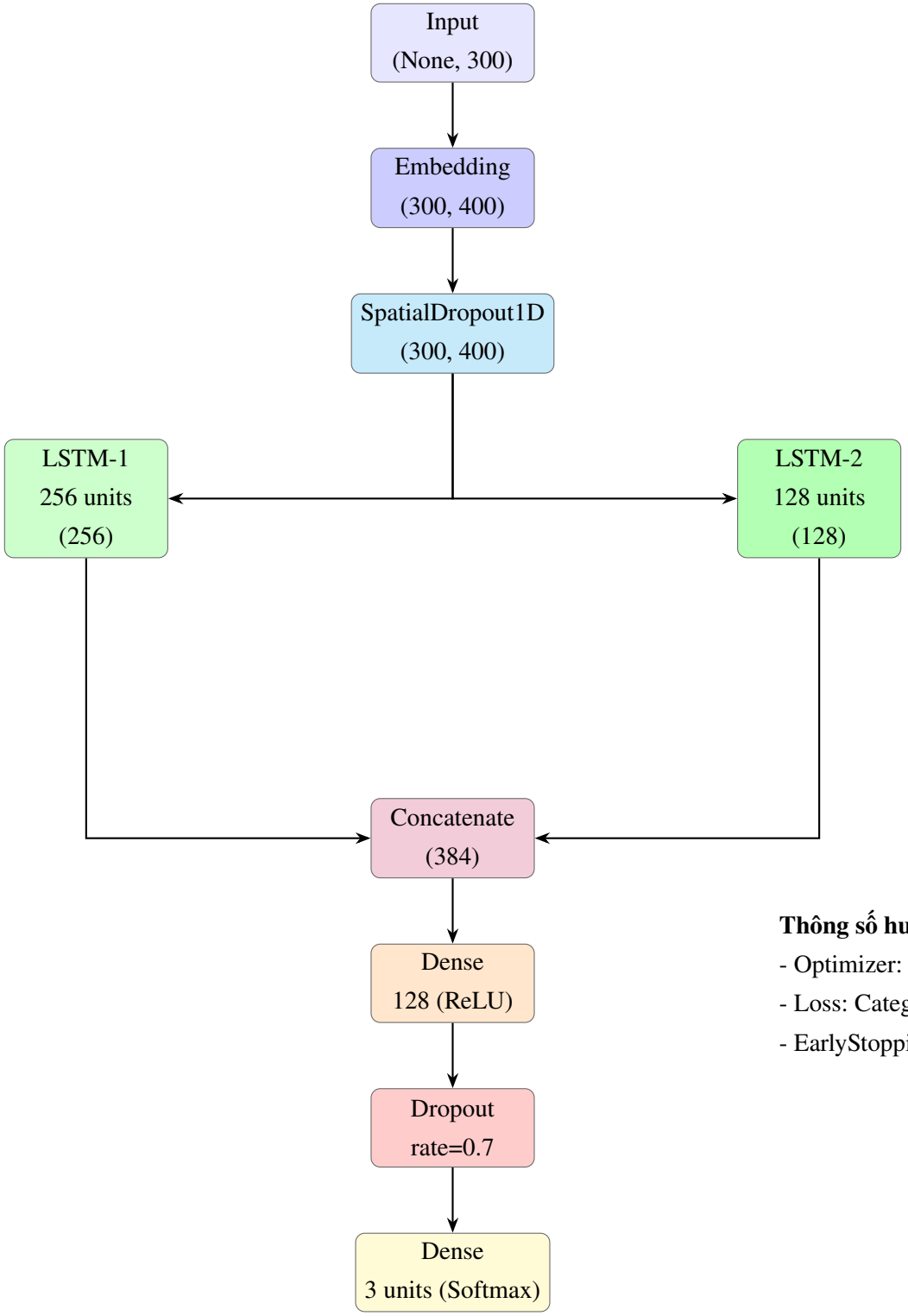
Dữ liệu đầu vào sau đó được truyền qua một **lớp nhúng (Embedding Layer)**, do `embedding_layer` định nghĩa, nơi mỗi token trong chuỗi được ánh xạ thành một vector số thực. Lớp nhúng này giúp mô hình học được các đặc trưng ngữ nghĩa của từ ngữ trong dữ liệu đầu vào. Sau lớp nhúng, một lớp **SpatialDropout1D** với tỷ lệ 0.2 được áp dụng để giảm thiểu hiện tượng quá khớp (overfitting) bằng cách loại bỏ ngẫu nhiên toàn bộ vector đặc trưng tại một số bước thời gian trong chuỗi.

Điểm nổi bật trong kiến trúc này là việc sử dụng **hai nhánh LSTM hoạt động song song**:

- Nhánh thứ nhất sử dụng `LSTM` với 256 đơn vị, có nhiệm vụ học các đặc trưng tổng quát hơn từ toàn bộ chuỗi.
- Nhánh thứ hai sử dụng `LSTM` với 128 đơn vị, cung cấp thông tin bổ sung ở mức thấp hơn hoặc hỗ trợ tăng tính đa dạng trong biểu diễn chuỗi.

Cả hai nhánh đều không trả về chuỗi đầy đủ (`return_sequences=False`), mà chỉ giữ lại **trạng thái cuối cùng**, đại diện cho ngữ cảnh của toàn bộ chuỗi. Đầu ra từ hai nhánh được **kết hợp bằng lớp** `concatenate`, tạo thành một tensor hợp nhất chứa đặc trưng từ cả hai nhánh.

Tiếp theo, tensor này được truyền qua một lớp **Dense** với 128 đơn vị và hàm kích hoạt `ReLU`, nhằm trích xuất thêm đặc trưng phi tuyến. Sau đó, mô hình áp dụng một lớp **Dropout** với tỷ lệ được xác định (`dropout_rate`) để tiếp tục giảm thiểu overfitting trong quá trình huấn luyện. Cuối cùng, mô hình sử dụng một lớp **Dense đầu ra** với số đơn vị bằng số lớp cần phân loại 3 (`units=num_classes`) và hàm kích hoạt `softmax`.



Thông số huấn luyện:

- Optimizer: Adam (lr = 0.001)
- Loss: Categorical Crossentropy
- EarlyStopping: patience = 4

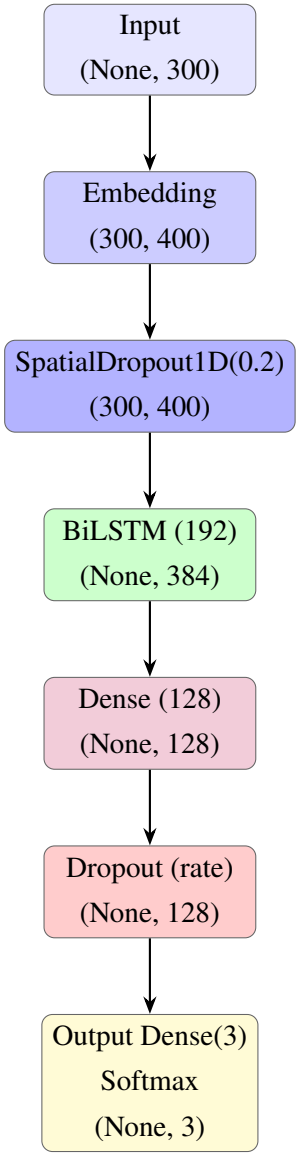
Kiến trúc của Parallel_LSTM

5.3.1.5 BiLSTM - Single_BiLSTM

Mô hình được triển khai dựa trên kiến trúc BiLSTM sử dụng thư viện **TensorFlow** và **Keras**. Tại tầng đầu vào (*Input Layer*), dữ liệu được đưa vào với kích thước là `(sequence_length,)`, trong đó `sequence_length` đại diện cho độ dài của mỗi chuỗi dữ liệu. Sau đó, dữ liệu được truyền qua một lớp nhúng (*Embedding Layer*), được định nghĩa bởi `embedding_layer`, nơi mỗi từ hoặc token trong chuỗi được biểu diễn thành một vector số.

Sau lớp nhúng, mô hình sử dụng một lớp `SpatialDropout1D` với tỷ lệ dropout là 0.2 nhằm ngăn ngừa hiện tượng overfitting và tăng khả năng tổng quát hóa bằng cách loại bỏ ngẫu nhiên một số chiều của vector đặc trưng. Tiếp theo, dữ liệu được truyền qua một lớp **BiLSTM** với 192 đơn vị ẩn, cho phép mô hình học thông tin ngữ cảnh theo cả hai chiều của chuỗi – từ trái sang phải và từ phải sang trái. Lớp BiLSTM được cấu hình với `dropout` và `recurrent_dropout` nhằm nâng cao khả năng khái quát hóa trong quá trình huấn luyện.

Sau lớp BiLSTM, đầu ra được chuyển đến một lớp `Dense` với 128 đơn vị và hàm kích hoạt ReLU để tiếp tục trích xuất đặc trưng phi tuyến. Một lớp `Dropout` được áp dụng sau đó với tỷ lệ dropout = 0.7, tiếp tục giúp giảm nguy cơ overfitting. Cuối cùng, mô hình sử dụng một lớp `Dense` đầu ra với số đơn vị tương ứng với số lớp phân loại (`num_classes`) và hàm kích hoạt `softmax` để đưa ra xác suất phân bố của các lớp mục tiêu.



Thông số huấn luyện:
- Optimizer: Adam (lr = 0.001)
- Loss: Categorical Crossentropy
- EarlyStopping: patience = 4

Kiến trúc của Single_BiLSTM

5.3.1.6 Parallel BiLSTM - Parallel_BiLSTM

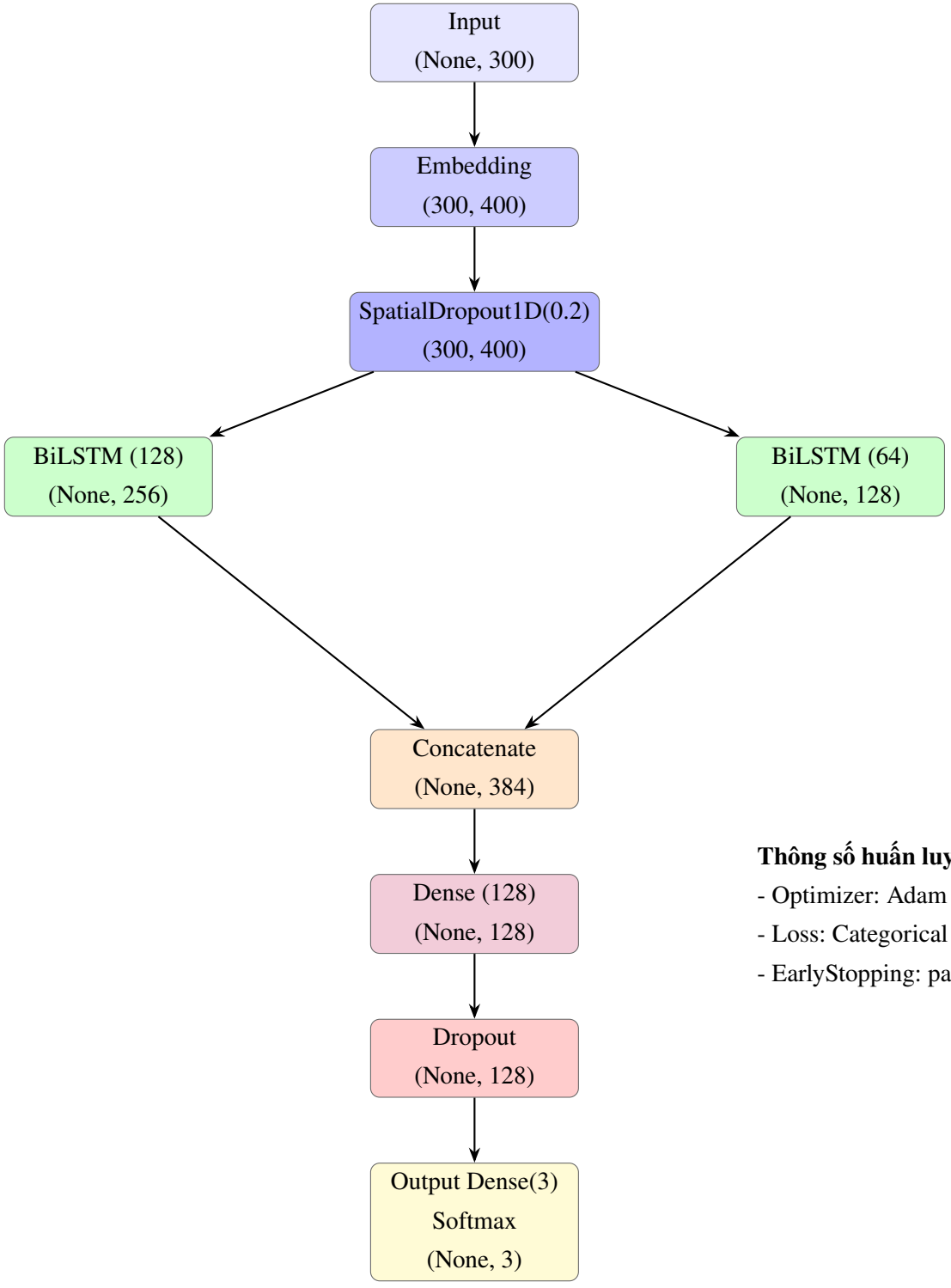
Mô hình được triển khai dựa trên kiến trúc BiLSTM sử dụng thư viện **TensorFlow** và **Keras**, với mục tiêu xử lý dữ liệu dạng chuỗi có chiều dài xác định. Tại tầng đầu vào (*Input Layer*), dữ liệu được đưa vào với kích thước là `(sequence_length,)`, trong đó `sequence_length` đại diện cho độ dài của mỗi chuỗi dữ liệu. Sau đó, dữ liệu được truyền qua một lớp nhúng (*Embedding Layer*), nơi mỗi từ hoặc token trong chuỗi được biểu diễn thành một vector số có kích thước xác định. Việc sử dụng lớp nhúng giúp mô hình học được các đặc trưng ngữ nghĩa từ dữ liệu đầu vào.

Sau lớp nhúng, mô hình áp dụng lớp `SpatialDropout1D` với tỷ lệ dropout là 0.2 nhằm làm giảm sự phụ thuộc của mô hình vào các đặc trưng cụ thể và tăng khả năng tổng quát

hóa. Tiếp theo, kiến trúc sử dụng hai lớp **Bidirectional LSTM** độc lập để trích xuất đặc trưng theo hai hướng từ dữ liệu đã qua dropout. Lớp BiLSTM thứ nhất có 128 đơn vị ẩn cho mỗi hướng (tổng đầu ra là 256), trong khi lớp BiLSTM thứ hai sử dụng 64 đơn vị ẩn cho mỗi hướng (tổng đầu ra là 128). Cả hai lớp BiLSTM đều không trả về chuỗi (`return_sequences=False`), do đó chỉ giữ lại thông tin ở bước cuối cùng trong chuỗi.

Kết quả từ hai lớp BiLSTM được kết hợp bằng cách sử dụng lớp `concatenate`, tạo thành một tensor duy nhất có kích thước tổng hợp. Tensor này sau đó được đưa qua một lớp `Dense` với 128 đơn vị và hàm kích hoạt `ReLU` nhằm tăng khả năng biểu diễn phi tuyến. Một lớp `Dropout` tiếp theo với tỷ lệ 0.7 được áp dụng để tránh *overfitting*.

Cuối cùng, đầu ra của mô hình là một lớp `Dense` với ba đơn vị (`units=3`) và hàm kích hoạt `softmax`, dùng để dự đoán xác suất của ba lớp phân loại đầu ra tương ứng với các nhãn cảm xúc trong bài toán phân tích cảm xúc.

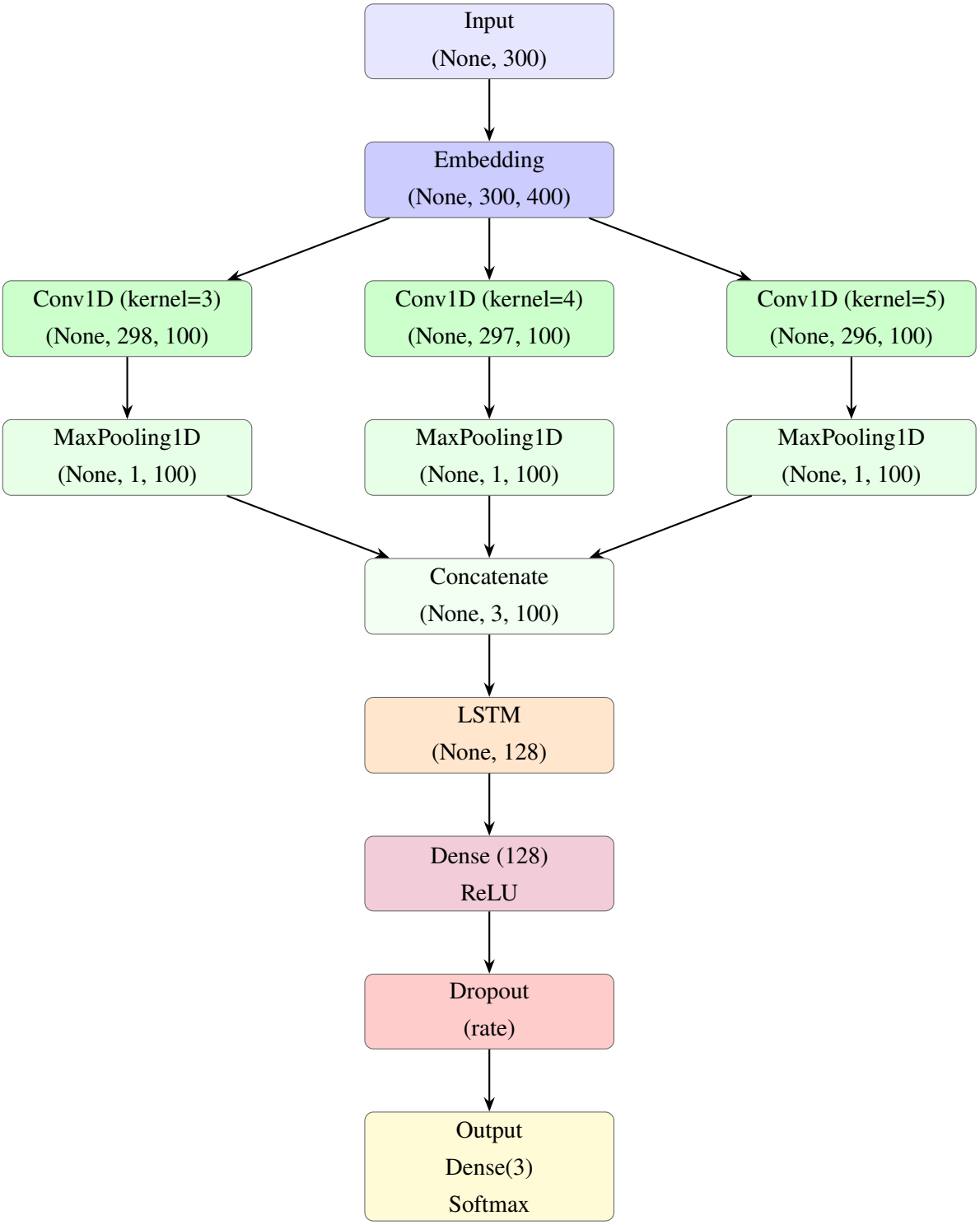


Kiến trúc của Parallel_BiLSTM

5.3.1.7 CNN kết hợp LSTM xếp chồng (CNN trước LSTM) - CNN_LSTM

Mô hình được triển khai dựa trên sự kết hợp giữa **CNN** và **LSTM**, sử dụng thư viện **TensorFlow** và **Keras**, nhằm khai thác đồng thời đặc trưng cục bộ từ CNN và quan hệ

ngữ cảnh dài hạn từ LSTM. Mô hình sử dụng ba lớp `Conv1D` song song với các kích thước kernel khác nhau (3, 4, 5), mỗi lớp có 100 bộ lọc và áp dụng regularization L2. Sau đó, đầu ra được gộp cực đại qua lớp `MaxPooling1D` và kết hợp bằng `concatenate`. Kết quả hợp nhất được đưa qua hai lớp `LSTM` song song (128 units) để học các mối quan hệ ngữ cảnh, rồi tiếp tục được kết hợp trước khi đi vào lớp phân loại. Kiến trúc tổng quát như sau:



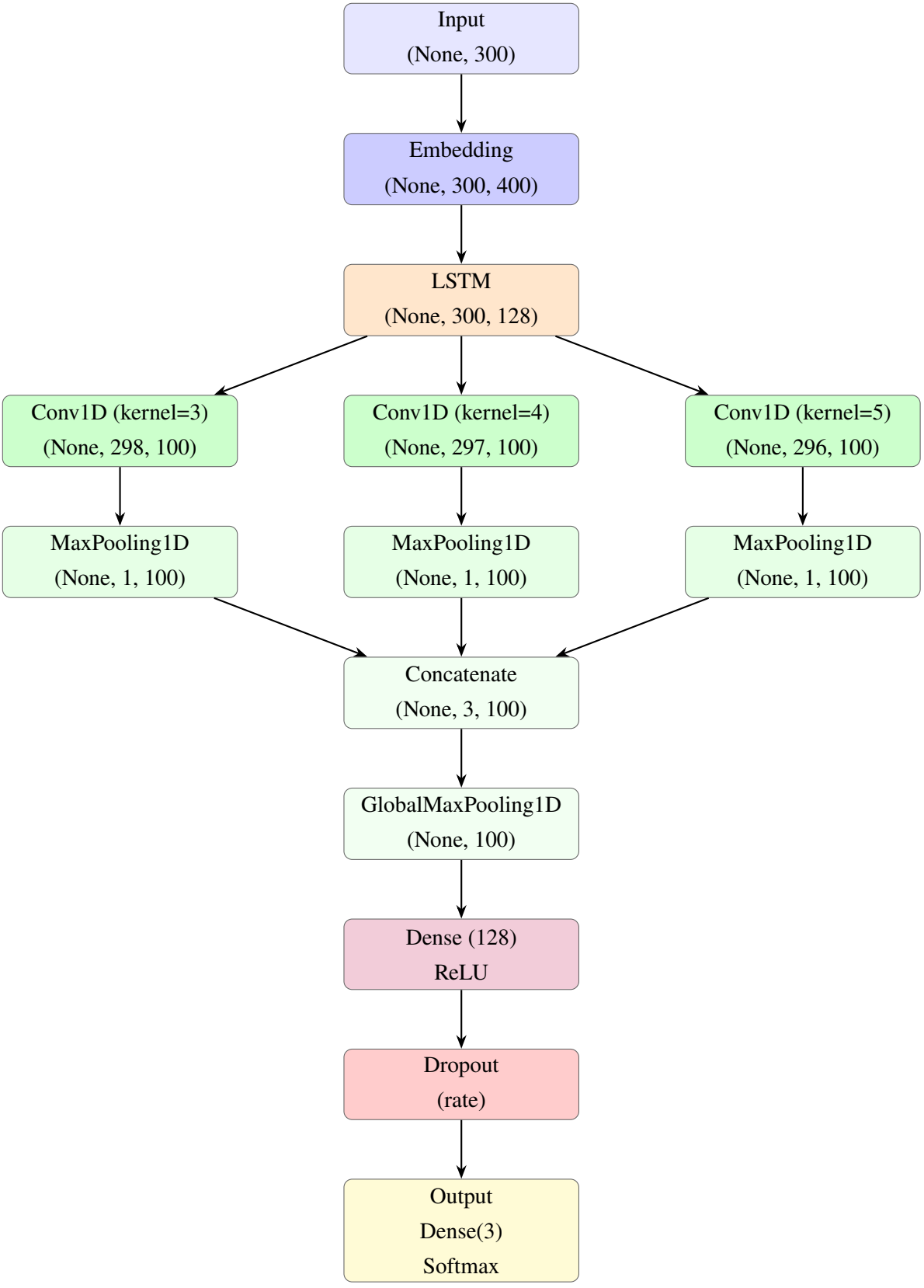
Kiến trúc của mô hình CNN_LSTM

5.3.1.8 LSTM kết hợp CNN xếp chồng (LSTM trước CNN) - LSTM_CNN

Tương tự mô hình trước, kiến trúc này được xây dựng trên sự kết hợp giữa **LSTM** và **CNN**, sử dụng thư viện **TensorFlow** và **Keras**. Mục tiêu của mô hình là tận dụng khả năng

học các phụ thuộc dài hạn trong chuỗi của LSTM, đồng thời khai thác các đặc trưng cục bộ từ văn bản thông qua các lớp tích chập (CNN). Ở tầng đầu vào (*Input Layer*), dữ liệu được biểu diễn dưới dạng chuỗi có độ dài cố định là `sequence_length = 300`. Mỗi token được ánh xạ sang một vector thông qua lớp *Embedding*, cho đầu ra có kích thước `(None, 300, 400)`, trong đó 400 là chiều không gian của vector nhúng. Lớp `LSTM` với 128 đơn vị ẩn tiếp theo được áp dụng, có cấu hình `return_sequences=True` nhằm giữ lại thông tin theo thời gian tại mỗi bước. Tham số `dropout` và `recurrent_dropout` được sử dụng nhằm giảm thiểu hiện tượng quá khớp (*overfitting*). Sau đó, đầu ra từ LSTM được đưa đồng thời vào ba lớp `Conv1D` với cùng số lượng bộ lọc (100 filters) nhưng khác nhau về kích thước kernel (tương ứng là 3, 4 và 5). Các lớp này sử dụng hàm kích hoạt `ReLU` và có thêm regularization L2 để giảm nguy cơ quá khớp. Mục đích của việc sử dụng nhiều kernel là để học các mẫu ngữ nghĩa ngắn có độ dài khác nhau (n-gram đa dạng) từ chuỗi đã được mã hóa ngữ cảnh.

Tiếp theo, mỗi đầu ra từ lớp `Conv1D` được đưa qua một lớp `MaxPooling1D` với kích thước phù hợp để rút trích đặc trưng quan trọng nhất theo từng dòng. Sau bước pooling, các tensor đầu ra được kết hợp (concatenate) trên trục thời gian, tạo thành một tensor duy nhất. Lớp `GlobalMaxPooling1D` sau đó được sử dụng để làm phẳng đầu ra, tạo thành một vector đặc trưng đại diện cho toàn bộ chuỗi đầu vào. Vector đặc trưng này được truyền qua một lớp `Dense` với 128 đơn vị và hàm kích hoạt `ReLU`, tiếp theo là lớp `Dropout` nhằm tăng khả năng khái quát hóa của mô hình. Cuối cùng, lớp đầu ra (`Dense(3)`) sử dụng hàm kích hoạt `softmax` để đưa ra xác suất thuộc về một trong ba lớp cảm xúc.



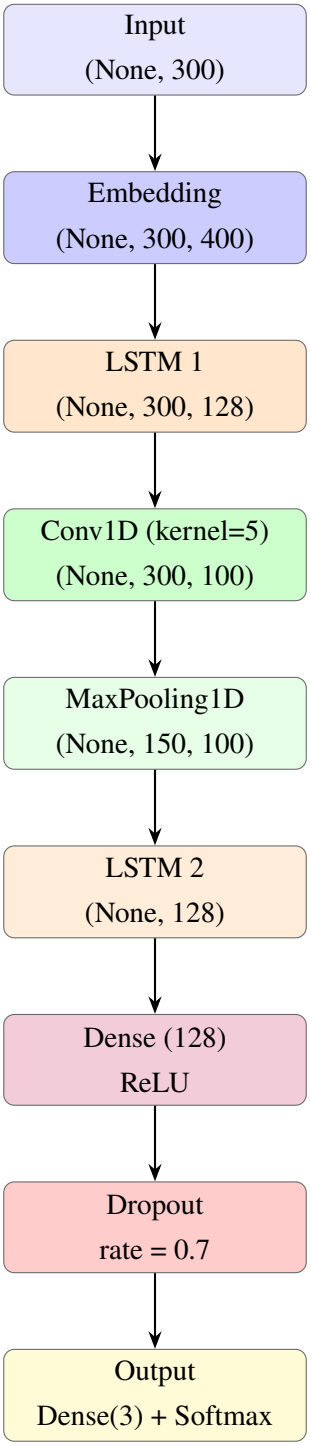
Kiến trúc của mô hình LSTM_CNN

5.3.1.9 Hai LSTM kết hợp với CNN theo thứ tự LSTM_CNN_LSTM

Mô hình **LSTM_CNN_LSTM** được xây dựng nhằm kết hợp khả năng học phụ thuộc dài hạn của **LSTM** với khả năng trích xuất đặc trưng cục bộ của **CNN**. Kiến trúc được triển khai bằng **TensorFlow** và **Keras** theo trình tự sau:

Chuỗi đầu vào được ánh xạ thành vector liên tục thông qua lớp `Embedding` với kích thước `(300, 400)`. Tiếp theo, lớp `LSTM` đầu tiên (128 đơn vị ẩn, `return_sequences=True`) tạo ra chuỗi đầu ra để phục vụ lớp `Conv1D` kế tiếp. Lớp tích chập sử dụng 100 bộ lọc (kernel=5, `ReLU`, `padding='same'`) kết hợp với lớp `MaxPooling1D` (pool size=2) để giảm chiều và làm nổi bật đặc trưng.

Đầu ra từ pooling được đưa vào lớp `LSTM` thứ hai (128 đơn vị ẩn, `return_sequences=False`) nhằm thu được biểu diễn toàn cục của chuỗi. Sau đó, tín hiệu được xử lý qua lớp `Dense` (128 đơn vị, `ReLU`) và `Dropout` (rate=0.7) để giảm overfitting. Cuối cùng, lớp `Dense` đầu ra sử dụng `softmax` để phân loại cảm xúc thành ba lớp.



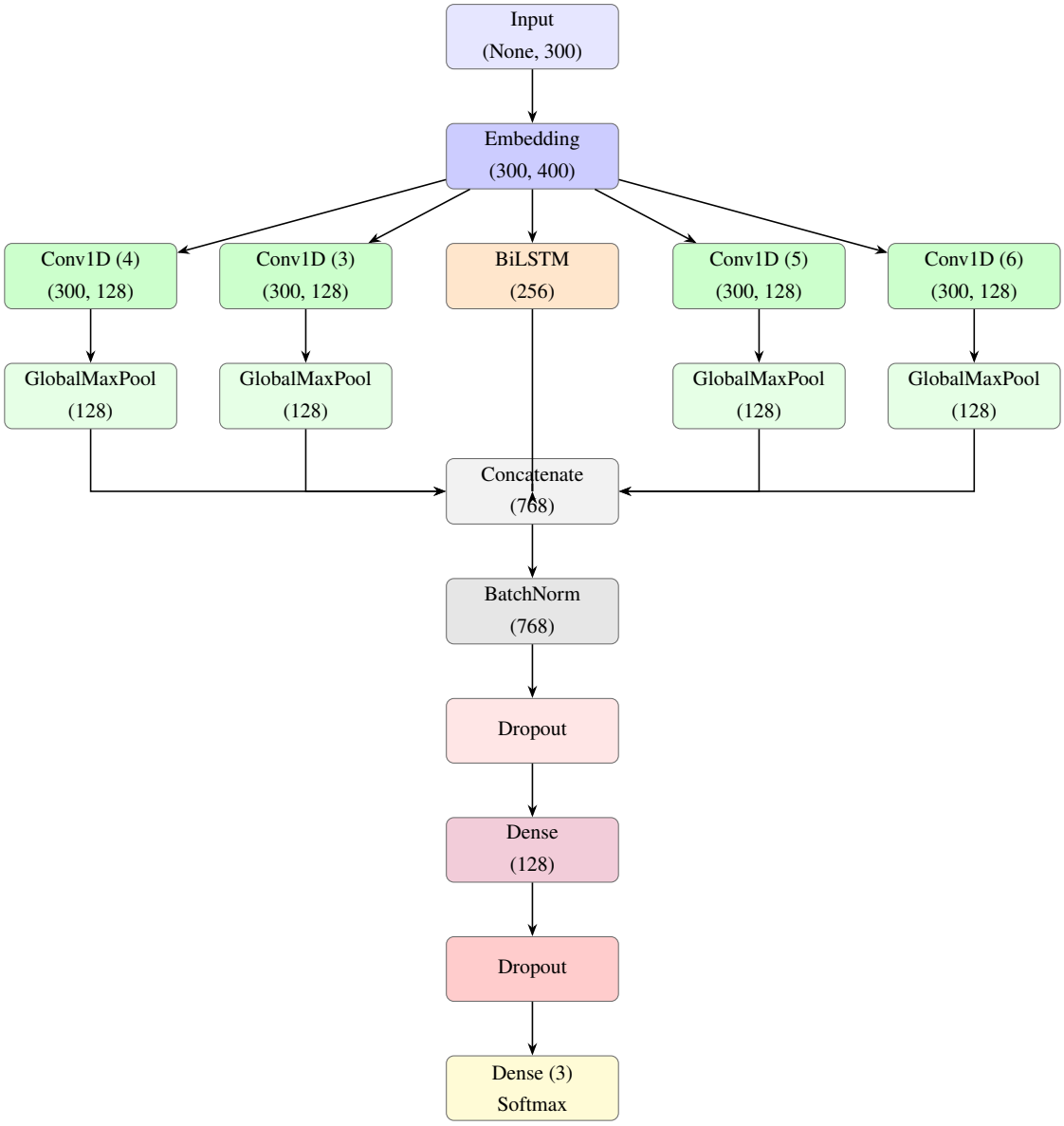
Kiến trúc của mô hình LSTM_CNN_LSTM

5.3.1.10 CNNs kết hợp BiLSTM chạy song song - CNN_BiLSTM

Mô hình **CNN_BiLSTM** được xây dựng nhằm kết hợp khả năng học đặc trưng cục bộ đa tỷ lệ từ **CNN** và khả năng mô hình hóa ngữ cảnh hai chiều của **BiLSTM**. Kiến trúc mô hình được hiện thực bằng **TensorFlow** và **Keras**. Dữ liệu đầu vào sau khi được ánh xạ qua lớp `Embedding` sẽ được xử lý song song qua bốn lớp `Conv1D` với các kích thước

kernel lần lượt là 3, 4, 5 và 6. Mỗi lớp tích chập sử dụng 128 bộ lọc, hàm kích hoạt ReLU, padding='same' và áp dụng L2 regularization (0.01) để tránh overfitting. Đầu ra từ các lớp này được đưa vào các lớp GlobalMaxPooling1D nhằm chọn ra đặc trưng quan trọng nhất từ mỗi nhánh tích chập. Đồng thời, đầu vào cũng được đưa vào lớp Bidirectional LSTM với 128 đơn vị ẩn và return_sequences=False để thu được biểu diễn tổng thể của chuỗi từ cả hai chiều.

Các đặc trưng thu được từ BiLSTM và các nhánh CNN được kết hợp thông qua hàm concatenate, tạo thành một vector đặc trưng tổng hợp, giàu ngữ nghĩa và cục bộ. Sau đó, mô hình áp dụng lớp BatchNormalization để chuẩn hóa đầu ra, tiếp theo là hai lớp Dropout (tỷ lệ 0.7) đan xen với lớp Dense (128 đơn vị, ReLU) nhằm giảm overfitting và học biểu diễn trung gian hiệu quả hơn. Cuối cùng, lớp Dense đầu ra với hàm kích hoạt softmax được sử dụng để đưa ra dự đoán phân phối xác suất trên ba lớp cảm xúc.



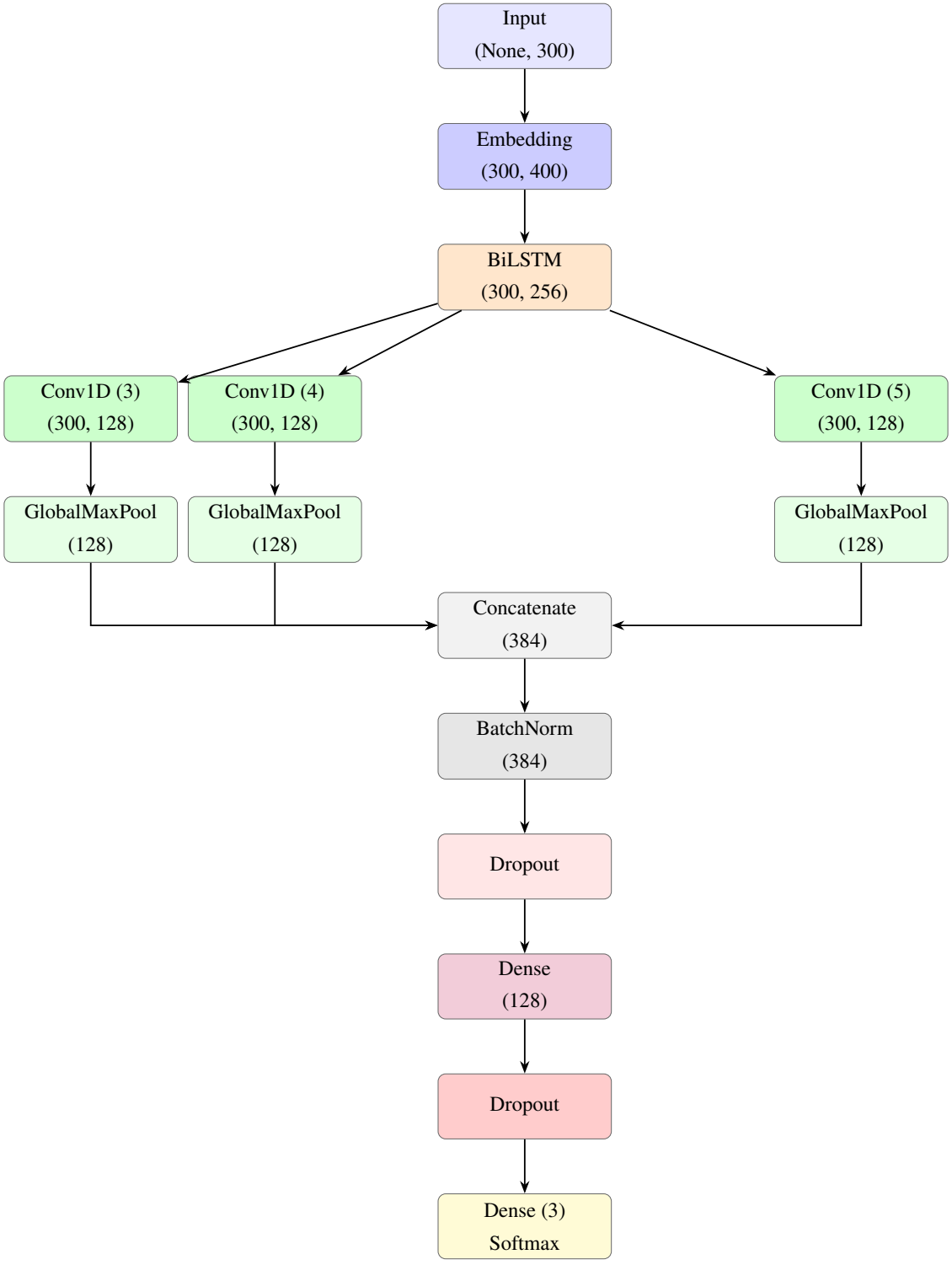
Kiến trúc của mô hình CNN_BiLSTM

5.3.1.11 CNNs kết hợp BiLSTM chạy tuần tự - Stacked_BiLSTN_CNN

Mô hình **Stacked_BiLSTN_CNN** được thiết kế nhằm khai thác hiệu quả thông tin ngữ cảnh hai chiều của chuỗi văn bản và các đặc trưng cục bộ ở nhiều mức độ. Kiến trúc mô hình kết hợp lớp **BiLSTM** với các nhánh **CNN đa kernel**, được triển khai bằng **TensorFlow** và **Keras**. Đầu vào sau lớp `Embedding` được đưa vào một lớp `BiLSTM` với `128` đơn vị ẩn và thiết lập `return_sequences=True`, nhằm giữ lại toàn bộ chuỗi đầu ra để phục vụ cho bước trích xuất đặc trưng bằng CNN sau đó.

Ba lớp `Conv1D` song song được áp dụng trên đầu ra của BiLSTM với kích thước kernel lần lượt là `3`, `4`, và `5`. Mỗi lớp sử dụng `128` bộ lọc, hàm kích hoạt `ReLU`, `padding='same'`, và `L2 regularization (0.01)` để tăng tính khái quát. Sau đó, các đầu ra tích chập được đưa qua các lớp `GlobalMaxPooling1D` nhằm trích xuất đặc trưng nổi bật nhất từ mỗi nhánh.

Các đặc trưng đã rút gọn được nối lại qua lớp `concatenate`, tạo thành một biểu diễn tổng hợp. Biểu diễn này tiếp tục được chuẩn hóa bởi `BatchNormalization`, và xử lý qua hai lớp `Dropout` (tỷ lệ `0.7`) nhằm giảm hiện tượng overfitting, xen kẽ với một lớp `Dense` (`128` đơn vị, `ReLU`) để tăng cường khả năng học. Cuối cùng, lớp `Dense` đầu ra sử dụng hàm `softmax` để dự đoán phân phối xác suất trên ba lớp cảm xúc.



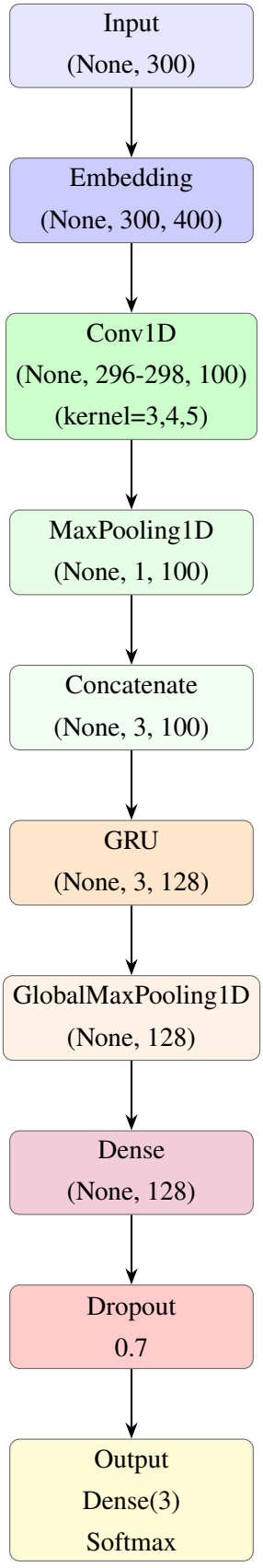
Kiến trúc của mô hình Stacked_BiLSTM_CNN

5.3.1.12 GRU kết hợp với CNN (liên tiếp GRU trước CNN) - CNN_GRU

Mô hình **CNN_GRU** được xây dựng nhằm khai thác đặc trưng cục bộ của chuỗi văn bản thông qua các lớp `CNN` đa kernel, kết hợp với khả năng ghi nhớ ngữ cảnh theo chuỗi của lớp `GRU`. Đầu vào từ lớp `Embedding` được xử lý song song qua ba lớp `Conv1D` với

các kích thước kernel khác nhau, được quy định bởi tham số `filter_sizes`. Mỗi lớp sử dụng `num_filters` bộ lọc, hàm kích hoạt `ReLU`, và chuẩn hóa L2 (`L2 regularization = 0.01`) nhằm tăng tính khái quát.

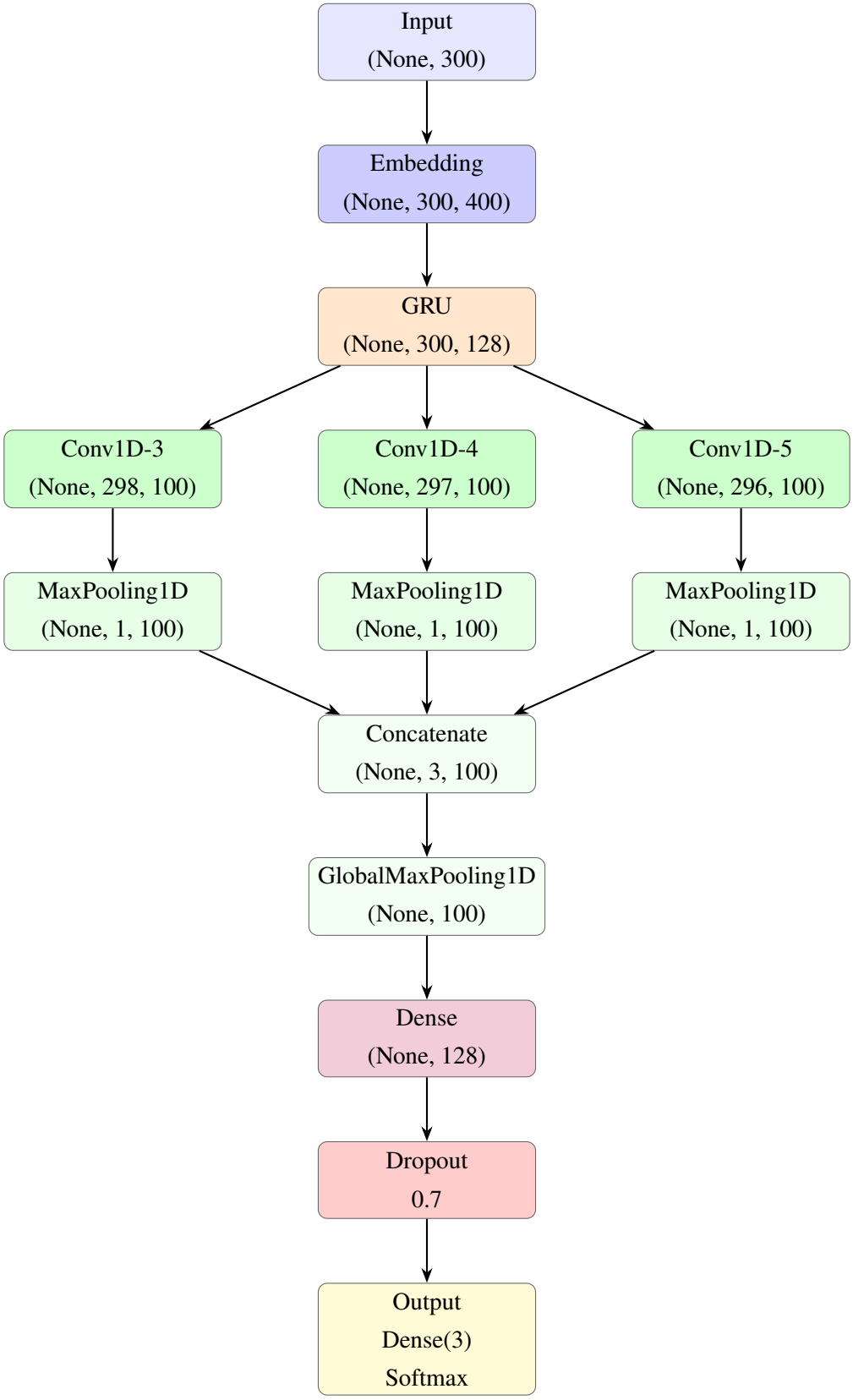
Tiếp theo, đầu ra từ mỗi lớp tích chập được đưa qua lớp `MaxPooling1D` với kích thước tương ứng để giảm chiều và giữ lại đặc trưng nổi bật nhất. Các đặc trưng từ ba nhánh được nối lại bằng lớp `concatenate`, tạo thành một tensor tổng hợp đặc trưng. Tensor này được đưa vào một lớp `GRU` với `128` đơn vị ẩn, có thiết lập `return_sequences=True`, cùng với `dropout` và `recurrent_dropout` nhằm hạn chế overfitting và tăng khả năng khái quát. Lớp `GlobalMaxPooling1D` sau đó được sử dụng để rút trích đặc trưng toàn cục từ chuỗi đầu ra của GRU, tạo thành một vector biểu diễn cố định. Vector này được đưa qua một lớp `Dense` với `128` đơn vị và hàm kích hoạt `ReLU`, tiếp theo là lớp `Dropout` (tỷ lệ `0.7`). Cuối cùng, lớp `Dense` đầu ra sử dụng hàm `softmax` để phân loại văn bản thành ba lớp cảm xúc.



Kiến trúc của mô hình CNN_GRU

5.3.1.13 CNN kết hợp với GRU (liên tiếp CNN trước GRU) - GPR_CNN

Mô hình GRU_CNN được thiết kế với mục tiêu kết hợp khả năng học thông tin tuần tự từ GRU và khả năng trích xuất đặc trưng cục bộ của CNN. Đầu vào được ánh xạ qua lớp nhúng, sau đó truyền qua lớp GRU với 128 đơn vị ẩn để học các phụ thuộc dài hạn. Đầu ra từ GRU được xử lý song song qua ba lớp Conv1D với kích thước kernel lần lượt là 3, 4, và 5. Mỗi nhánh được nối tiếp bởi lớp MaxPooling1D để giảm chiều và giữ lại thông tin quan trọng. Các đặc trưng sau đó được gộp lại bằng lớp concatenate, rồi đưa qua lớp GlobalMaxPooling1D để thu được véc-tơ đầu ra. Cuối cùng, véc-tơ này được truyền qua lớp Dense 128 đơn vị, Dropout 0.7, và lớp Dense softmax để dự đoán xác suất thuộc về ba lớp cảm xúc. Kiến trúc này tận dụng cả thông tin ngữ cảnh tuần tự và đặc trưng ngữ pháp cục bộ trong văn bản.

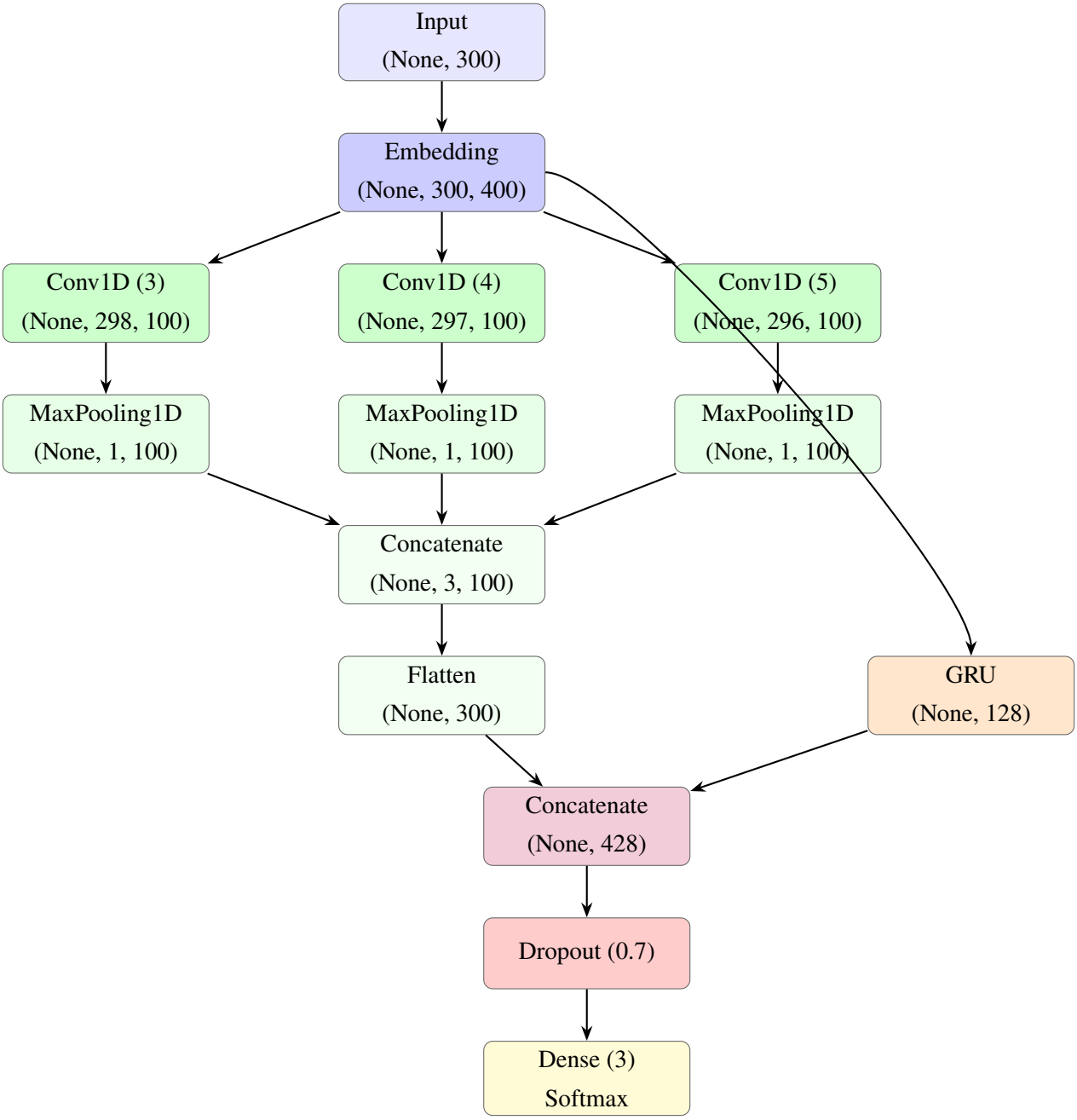


Kiến trúc của mô hình GPR_CNN

5.3.1.14 GRU và CNN kết hợp song song - CNN_GRU_Parallel

Mô hình được xây dựng với cấu trúc kết hợp song song giữa mạng **CNN** và **GRU**. Cụ thể, kiến trúc mô hình được thiết kế như sau:

- **Lớp nhúng (Embedding)**: Đầu vào là chuỗi có độ dài cố định, được ánh xạ sang không gian vector liên tục thông qua lớp *Embedding*, tạo ra biểu diễn ngữ nghĩa cho từng từ trong chuỗi.
- **Nhánh CNN**: Đầu ra từ lớp nhúng được đưa song song vào ba lớp `Conv1D` với kích thước kernel lần lượt là 3, 4 và 5, mỗi lớp có 100 bộ lọc. Các lớp này sử dụng hàm kích hoạt `ReLU` và được regular hóa bằng chuẩn L2. Sau mỗi lớp `Conv1D` là một lớp `MaxPooling1D` có kích thước cửa sổ bằng độ dài đầu ra còn lại, nhằm rút trích đặc trưng quan trọng nhất của từng nhánh. Các đầu ra được nối lại bằng lớp `Concatenate` và đưa qua lớp `Flatten` để chuyển thành vector đặc trưng một chiều.
- **Nhánh GRU**: Song song với nhánh CNN, đầu ra từ lớp nhúng cũng được đưa vào một lớp `GRU` với 128 đơn vị ẩn. Lớp này được regular hóa bằng chuẩn L2 và không trả về chuỗi (`return_sequences=False`), nhằm rút gọn thông tin theo thời gian thành một vector duy nhất thể hiện trạng thái toàn chuỗi.
- **Kết hợp và phân loại**: Đầu ra từ hai nhánh CNN và GRU được kết hợp bằng lớp `Concatenate`, sau đó được đưa qua một lớp `Dropout` với tỷ lệ 0.7 nhằm giảm overfitting. Cuối cùng, lớp `Dense` đầu ra sử dụng hàm kích hoạt `softmax` để phân loại chuỗi đầu vào thành một trong ba lớp cảm xúc.



Kiến trúc của mô hình CNN_GRU_Parallel

5.3.2 Chiến lược huấn luyện mô hình

Tất cả các mô hình được huấn luyện trong vòng tối đa **10 Epoch** với **batch size = 256**, sử dụng hàm mất mát `categorical crossentropy` và thuật toán tối ưu `Adam`. Dữ liệu được chia theo tỷ lệ **80% cho huấn luyện** và **20% cho kiểm định (validation)**.

Để tránh hiện tượng overfitting, chiến lược **dừng sớm (Early Stopping)** được áp dụng, theo dõi giá trị hàm mất mát trên tập kiểm định (`val_loss`). Mô hình sẽ dừng huấn luyện nếu không có sự cải thiện đáng kể nào (với ngưỡng `min_delta = 0.01`) sau **4 epoch liên**

tiếp. Tất cả mô hình đều được huấn luyện với dữ liệu đã được xáo trộn ngẫu nhiên trước mỗi epoch (`shuffle=True`).

Quá trình huấn luyện được tổ chức thông qua một vòng lặp, trong đó từng mô hình được biên dịch (`compile`) và huấn luyện tuần tự. Kết quả huấn luyện của từng mô hình được lưu trữ lại để phục vụ cho việc so sánh và đánh giá hiệu quả sau này.

5.3.3 BERT

Trong nghiên cứu này, chúng em tiến hành huấn luyện ba mô hình học sâu dựa trên kiến trúc Transformer cho bài toán phân loại cảm xúc tiếng Việt: **BERT-base-multilingual-cased**, **PhoBERT-base**, và **PhoBERT-large**. Các mô hình được triển khai bằng thư viện Transformers của Hugging Face, với quy trình huấn luyện sử dụng API `Trainer` kết hợp bộ dữ liệu VLSP.

5.3.3.1 Tiền xử lý dữ liệu

Tập dữ liệu gốc được nạp từ tệp văn bản định dạng CSV. Mỗi dòng dữ liệu bao gồm một nhãn cảm xúc `Class` và một câu văn bản `Data`. Các bước tiền xử lý bao gồm:

- Loại bỏ các dòng bị thiếu giá trị.
- Đổi tên các cột thành `text` và `label`.
- Chuẩn hóa nhãn theo ánh xạ: $-1 \rightarrow 0$, $0 \rightarrow 1$, $1 \rightarrow 2$.
- Với PhoBERT, văn bản được tách từ bằng công cụ PyVi (`ViTokenizer`).

5.3.3.2 Token hóa và chuẩn hóa định dạng

Sau khi tiền xử lý, dữ liệu được token hóa bằng bộ `AutoTokenizer` tương ứng với từng mô hình. Các tham số token hóa bao gồm:

- `padding=max_length`,
- `truncation=True`,
- `max_length=256`.

Sau đó, định dạng dữ liệu được chuẩn hóa sang `PyTorch Tensor` với các cột đầu vào: `input_ids`, `attention_mask`, và `labels`.

5.3.3.3 Kiến trúc mô hình và thiết lập huấn luyện

BERT-base-multilingual-cased là phiên bản đa ngôn ngữ gồm 12 tầng Transformer, 768 chiều ẩn và 12 đầu attention. Mô hình được tải sẵn từ `bert-base-multilingual-cased` với `num_labels=3`.

PhoBERT-base và PhoBERT-large là hai mô hình ngôn ngữ tiếng Việt do nhóm VinAI phát triển.

- PhoBERT-base: 12 tầng Transformer, 768 chiều ẩn, 12 đầu attention.
- PhoBERT-large: 24 tầng Transformer, 1024 chiều ẩn, 16 đầu attention.

Mô hình PhoBERT yêu cầu văn bản được tách từ trước, do đó bước tiền xử lý sử dụng `ViTokenizer` là bắt buộc.

5.3.3.4 Huấn luyện và đánh giá

Quá trình huấn luyện sử dụng lớp `Trainer` với các tham số:

Tham số	**BERT-base-multilingual**	**PhoBERT-base**	**PhoBERT-large**
Tốc độ học (learning rate)	2×10^{-5}	2×10^{-5}	2×10^{-5}
Batch size	16	16	4
Số epoch	5	10	10
Trọng số suy giảm (weight decay)	0.01	0.01	0.01
Chiều dài tối đa (max length)	256	256	256
Tokenizer sử dụng nhanh (use_fast)	Mặc định (True)	False	False
Tiền xử lý PyVi	Không	Có	Có
Thiết bị huấn luyện	MPS (Apple Silicon)	MPS (Apple Silicon)	MPS (Apple Silicon)

Bảng 5.2: Thống kê các tham số huấn luyện cho từng mô hình BERTS

Mô hình được huấn luyện trên Apple Silicon (sử dụng `MPS`) nếu khả dụng. Sau huấn luyện, mô hình được đánh giá trên tập kiểm tra với các chỉ số:

- **Accuracy**,
- **Precision**
- **Recall**,
- **F1-score** (macro average).

Báo cáo phân loại (classification report) được trích xuất từ thư viện `sklearn.metrics`, phản ánh chi tiết hiệu suất của mô hình trên ba lớp nhãn cảm xúc: tiêu cực (-1), trung lập (0), và tích cực (1).

5.3.3.5 Tài nguyên sử dụng

- Bộ dữ liệu VLSP.
- Mô hình BERT đa ngôn ngữ: `bert-base-multilingual-cased`.
- Mô hình PhoBERT: `vinai/phobert-base`, `vinai/phobert-large`.
- Thư viện: `transformers`, `datasets`, `sklearn`, `pyvi`.

Chương 6

Kết quả

6.1 Một số metric được sử dụng

6.1.1 Precision

Trong đánh giá mô hình phân loại, Precision được định nghĩa là tỷ lệ các mẫu được dự đoán đúng trên tổng số mẫu được dự đoán là dương. Công thức tính Precision như sau:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Trong đó:

- TP (True Positive): Số lượng mẫu dương được dự đoán đúng.
- FP (False Positive): Số lượng mẫu âm được dự đoán sai thành dương.

6.1.2 Recall

Trong đánh giá mô hình phân loại, Recall được định nghĩa là tỷ lệ số mẫu dương được dự đoán đúng trên tổng số mẫu thực sự là dương. Công thức tính Recall như sau:

$$\text{Recall} = \frac{TP}{TP + FN}$$

Trong đó:

- TP (True Positive): Số lượng mẫu dương được dự đoán đúng.
- FN (False Negative): Số lượng mẫu dương bị dự đoán sai thành âm.

6.1.3 F1-score

F1-score là một thước đo hài hòa giữa Precision và Recall, thường được sử dụng để đánh giá hiệu suất của mô hình phân loại. Công thức tính F1-score như sau:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

6.1.4 Accuracy

Độ chính xác là một chỉ số đánh giá hiệu suất của mô hình phân loại. Nó được tính bằng công thức:

$$\text{Accuracy} = \frac{\text{Số lượng dự đoán đúng}}{\text{Tổng số dự đoán}}$$

6.2 Kết quả thực nghiệm

Bảng 6.1 trình bày kết quả so sánh hiệu suất giữa các mô hình học sâu truyền thống và các mô hình ngôn ngữ tiền huấn luyện trong tác vụ phân loại cảm xúc tiếng Việt. Các metric số được xem xét bao gồm **Accuracy**, **Precision**, **Recall** và **F1-score**.

Mô hình	**Accuracy (↑)**	**Precision (↑)**	**Recall (↑)**	**F1-score (↑)**
Stacked_LSTM	54.57%	59.91%	48.10%	51.89%
LSTM_Simple	60.86%	62.79%	56.57%	58.70%
Parallel_BiLSTM	60.76%	61.98%	59.14%	59.92%
Parallel_LSTM	62.10%	63.84%	60.19%	61.34%
LSTM_CNN	62.76%	63.55%	62.10%	62.52%
LSTM_CNN_LSTM	62.67%	62.72%	62.00%	61.99%
GRU_CNN	63.62%	64.17%	61.90%	62.69%
CNN_GRU_Parallel	64.10%	64.36%	62.76%	63.10%
CNN_LSTM	64.10%	65.26%	63.33%	63.73%
Stacked_BiLSTN_CNN	64.29%	65.57%	62.76%	63.55%
CNN_Org	64.76%	67.04%	62.95%	64.15%
Single_BiLSTM	65.24%	66.24%	63.52%	64.15%
CNN_BiLSTM	65.62%	66.11%	64.67%	64.79%
CNN_GRU	66.48%	68.05%	64.10%	65.32%
BERT-base-multilingual	69.90%	70.00%	70.00%	70.00%
PhoBERT-base	76.29%	76.00%	76.00%	76.00%
PhoBERT-large	**76.57%**	**77.00%**	**77.00%**	**77.00%**

Bảng 6.1: So sánh hiệu suất các mô hình

6.2.1 Hiệu suất của các mô hình học sâu truyền thống

Các mô hình học sâu như LSTM, GRU, BiLSTM và CNN (hoặc kết hợp giữa chúng) thể hiện hiệu suất đa dạng, phụ thuộc vào kiến trúc cụ thể. Trong nhóm này, mô hình `Stacked_LSTM` cho thấy kết quả thấp nhất, với độ chính xác chỉ đạt **54.57%** và F1-score là **51.89%**. Điều này cho thấy rằng việc chỉ đơn thuần tăng chiều sâu của mạng LSTM mà không có thêm cơ chế tăng cường biểu diễn đặc trưng có thể dẫn đến quá khớp hoặc khó hội tụ.

Ngược lại, một số mô hình kết hợp như CNN_GRU và CNN_BiLSTM đạt hiệu suất cao hơn đáng kể, lần lượt với Accuracy là **66.48%** và **65.62%**, cùng F1-score đạt **65.32%** và **64.79%**. Những kết quả này cho thấy sự kết hợp giữa CNN và các mạng tuần tự như GRU hoặc BiLSTM giúp tăng cường khả năng trích xuất đặc trưng theo cả chiều không gian và chiều thời gian.

Ngoài ra, các mô hình LSTM_Simple, Parallel_LSTM, hay LSTM_CNN cũng đạt kết quả khá ổn định trong khoảng từ 60% đến 63%, tuy nhiên vẫn thấp hơn đáng kể so với nhóm mô hình Transformer. Nhìn chung, nhóm mô hình học sâu truyền thống mặc dù đã có những cải tiến trong kiến trúc và cách kết hợp tầng, nhưng vẫn còn hạn chế trong việc mô hình hóa ngữ nghĩa phức tạp và ngữ cảnh rộng.

6.2.2 Hiệu suất của các mô hình Transformer

Các mô hình ngôn ngữ tiền huấn luyện (pre-trained language models), đặc biệt là BERT và PhoBERT, đã chứng minh sự vượt trội rõ rệt so với các mô hình học sâu truyền thống. Mô hình BERT-base-multilingual, mặc dù không được huấn luyện đặc thù cho tiếng Việt, vẫn đạt được Accuracy lên tới **69.90%** cùng với Precision, Recall và F1-score đều đạt **70.00%**. Điều này cho thấy sức mạnh tổng quát hóa của mô hình đa ngôn ngữ BERT, dù chưa được tối ưu hoá cho dữ liệu tiếng Việt.

Tuy nhiên, sự cải thiện lớn nhất được ghi nhận khi sử dụng các phiên bản của PhoBERT - mô hình được huấn luyện chuyên biệt cho tiếng Việt. Phiên bản PhoBERT-base đạt Accuracy **76.29%** và F1-score **76.00%**, cao hơn gần 10% so với BERT-base-multilingual. Phiên bản nâng cấp hơn là PhoBERT-large đạt hiệu suất vượt trội nhất trong toàn bộ bảng, với Accuracy lên đến **76.57%**, và các chỉ số Precision, Recall, F1-score đều đồng đều ở mức **77.00%**. Điều này phản ánh rằng việc mở rộng quy mô kiến trúc mô hình (với nhiều tầng transformer hơn) cùng với việc huấn luyện trên tập dữ liệu tiếng Việt lớn đã giúp mô hình nắm bắt tốt hơn các đặc trưng ngữ nghĩa và ngữ pháp của tiếng Việt.

6.2.3 So sánh tổng quan và đánh giá xu hướng

Một xu hướng rõ ràng có thể quan sát được là hiệu suất tăng dần từ các mô hình học sâu truyền thống đến các mô hình ngôn ngữ tiền huấn luyện. Không chỉ đạt kết quả cao hơn về Accuracy, các mô hình Transformer còn cho thấy sự cân bằng giữa Precision và Recall, điều này đặc biệt quan trọng trong bài toán phân loại đa lớp, nơi mỗi nhãn đều có ý nghĩa riêng biệt và cần được xử lý công bằng.

Mặc dù các mô hình học sâu như CNN_GRU hay CNN_BiLSTM đã đạt đến mức F1-score trên 65%, nhưng vẫn thấp hơn gần 13% so với mô hình PhoBERT-large. Do đó, ối với bài toán phân loại cảm xúc tiếng Việt, các mô hình ngôn ngữ tiền huấn luyện như PhoBERT là lựa chọn ưu việt cả về độ chính xác, độ tin cậy lẫn khả năng tổng quát hoá.

6.2.4 Kết luận

Kết quả từ Bảng 6.1 cho thấy rằng mô hình `PhoBERT-large` là phương án tối ưu nhất trong việc phân loại cảm xúc tiếng Việt, vượt trội hơn rõ rệt so với các mô hình học sâu truyền thống. Những kết quả này khẳng định vai trò quan trọng của các mô hình ngôn ngữ tiền huấn luyện trong xử lý ngôn ngữ tự nhiên hiện đại, đặc biệt là khi được huấn luyện chuyên biệt cho từng ngôn ngữ.

Tài liệu tham khảo

[1] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 International Conference on Engineering and Technology (ICET)*, 2017, pp. 1–6.

[2] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, 11 1997.

[3] C. Gulcehre, K. Cho, R. Pascanu, and Y. Bengio, "Learned-norm pooling for deep feedforward and recurrent neural networks," 09 2014, pp. 530–546.

[4] V. Chakkarwar, S. Tamane, and A. Thombre, *A Review on BERT and Its Implementation in Various NLP Tasks*, 05 2023, pp. 112–121.

[5] W. Yue and L. Li, "Sentiment analysis using word2vec-cnn-bilstm classification," 12 2020, pp. 1–5.

[6] D. Q. Nguyen and A. Nguyen, "Phobert: Pre-trained language models for vietnamese," 01 2020, pp. 1037–1042.

[7] A. Aziz Sharfuddin, M. Nafis Tihami, and M. Saiful Islam, "A deep recurrent neural network with bilstm model for sentiment classification," in *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*, 2018, pp. 1–4.

[8] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *Proceedings of Workshop at ICLR*, vol. 2013, 01 2013.

[9] K. Nguyen and K. Nguyen, "Exploiting vietnamese social media characteristics for textual emotion recognition in vietnamese," 09 2020.

[10] H. D. Huynh, H. T.-T. Do, K. V. Nguyen, and N. L.-T. Nguyen, "A simple and efficient ensemble classifier combining multiple neural network models on social media datasets in vietnamese," 2020. [Online]. Available: https://arxiv.org/abs/2009.13060

[11] K. Nguyen, D.-V. Nguyen, P. Nguyen, T. Truong, and N. Nguyen, "Uit-vsfc: Vietnamese students' feedback corpus for sentiment analysis," 11 2018, pp. 19–24.

[12] S. Luu, K. Nguyen, and N. Nguyen, “Empirical study of text augmentation on social media text in vietnamese,” 09 2020.

[13] Z. Lipton, “A critical review of recurrent neural networks for sequence learning,” 05 2015.

[14] Q. Cui, S. Wu, Q. Liu, W. Zhong, and L. Wang, “Mv-rnn: A multi-view recurrent neural network for sequential recommendation,” *IEEE Transactions on Knowledge and Data Engineering*, vol. PP, pp. 1–1, 11 2018.

[15] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” 10 2018.

[16] S. Trần, K. Thái, and V. Phạm, “Phân tích cảm xúc trên phản hồi học viên bằng mô hình bert kết hợp kiến trúc đa kênh cnn-gru,” *Tạp chí Nghiên cứu Tài chính - Marketing*, vol. 16, pp. 92–109, 02 2025.

[17] T. Ho Huong, “Kỹ thuật làm tăng dữ liệu trong phân tích cảm xúc trên ngôn ngữ tiếng việt,” *TẠP CHÍ KHOA HỌC ĐẠI HỌC MỞ THÀNH PHỐ HỒ CHÍ MINH - KỸ THUẬT VÀ CÔNG NGHỆ*, vol. 17, pp. 20–27, 04 2022.