

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC - KỸ THUẬT MÁY TÍNH



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

Bài tập lớn

Sentiment analysis

Giảng viên hướng dẫn: PGS.TS. Quản Thành Thơ
Sinh viên: Bùi Đức Anh

2112751

TP. HỒ CHÍ MINH, THÁNG 04/2025

Mục lục

1	Giới thiệu	2
2	Kiến thức nền tảng	3
2.1	Convolutional neural network	3
2.2	Long-short term memory	5
2.2.1	Mạng nơ ron truy hồi (RNN - Recurrent Neural Network)	5
2.2.2	Mạng LSTM – Long Short-Term Memory	6
2.3	Transformer	8
2.3.1	Transformer: Kiến trúc nền tảng cho xử lý ngôn ngữ tự nhiên hiện đại	8
2.3.2	BERT: Biểu diễn ngữ nghĩa hai chiều mạnh mẽ	9
2.3.3	PhoBERT: BERT dành riêng cho tiếng Việt	10
2.4	Biểu diễn từ và kỹ thuật Word Embedding	11
3	Kiến trúc đề xuất và thực nghiệm	13
3.1	CNN	13
3.2	LSTM	16
3.3	Hybrid CNN+LSTM	20
3.4	BERT-based	24
4	Đánh giá	31
5	Kết luận	32
5.1	Tổng kết	32
5.2	Định hướng phát triển trong tương lai	32
	TÀI LIỆU THAM KHẢO	34

1 Giới thiệu

Phân tích cảm xúc (Sentiment Analysis) là một trong những bài toán cơ bản và quan trọng trong lĩnh vực xử lý ngôn ngữ tự nhiên (Natural Language Processing – NLP). Mục tiêu chính của bài toán là xác định cảm xúc, thái độ hoặc quan điểm của người viết thông qua nội dung văn bản. Ứng dụng của phân tích cảm xúc rất đa dạng và thực tế, có thể kể đến như đánh giá sản phẩm, phản hồi khách hàng, theo dõi dư luận xã hội, hỗ trợ chăm sóc khách hàng tự động, hoặc thậm chí dự đoán xu hướng hành vi của người dùng. Trong bài toán này, mỗi đoạn văn bản sẽ được gán một nhãn thể hiện cảm xúc, thường là tích cực, tiêu cực hoặc trung lập. Chẳng hạn, câu “Sản phẩm này rất tuyệt vời” sẽ được gán nhãn tích cực, trong khi câu “Dịch vụ quá tệ” sẽ nhận nhãn tiêu cực.

Tuy nhiên, bài toán phân tích cảm xúc không đơn giản như việc nhận diện từ khóa cảm xúc mà đòi hỏi mô hình phải hiểu được ngữ cảnh, cấu trúc câu, sự mơ hồ về ngữ nghĩa, hiện tượng đảo nghĩa do phủ định và các yếu tố ngôn ngữ học khác. Ví dụ, cụm từ “không tệ” mang nghĩa tích cực dù bản thân từ “tệ” là tiêu cực. Điều này đặt ra yêu cầu rất cao về khả năng xử lý ngôn ngữ tự nhiên của hệ thống phân tích cảm xúc, đặc biệt trong tiếng Việt – một ngôn ngữ giàu sắc thái và nhiều biến thể biểu đạt.

Tùy vào mục tiêu ứng dụng, bài toán phân tích cảm xúc có thể được triển khai theo nhiều cấp độ như phân loại nhị phân (tích cực/tiêu cực), phân loại ba lớp (tích cực/trung lập/tiêu cực) hoặc phân loại đa mức (rất tiêu cực, tiêu cực, trung lập, tích cực, rất tích cực). Việc lựa chọn mức độ phân loại phù hợp phụ thuộc vào yêu cầu thực tế và độ phức tạp của dữ liệu đầu vào.

Trong những năm gần đây, sự phát triển mạnh mẽ của các kỹ thuật học sâu (deep learning) đã góp phần nâng cao độ chính xác trong phân tích cảm xúc. Các mô hình như RNN, LSTM, GRU, CNN, cũng như các mô hình ngôn ngữ tiền huấn luyện như BERT, PhoBERT đã chứng minh hiệu quả vượt trội so với các phương pháp học máy truyền thống nhờ khả năng học được biểu diễn ngữ nghĩa sâu và mối liên kết ngữ cảnh dài hạn trong câu. Trong phạm vi đề tài lần này, em xin tập trung tìm hiểu và áp dụng một số mô hình deep learning tiêu biểu như LSTM, TextCNN và PhoBERT cho bài toán phân tích cảm xúc văn bản tiếng Việt. Việc so sánh, đánh giá và phân tích hiệu quả giữa các mô hình sẽ là nội dung chính của nghiên cứu.

2 Kiến thức nền tảng

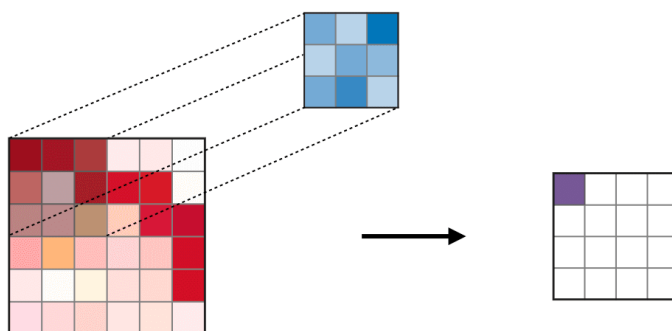
2.1 Convolutional neural network

Tổng quan

Mạng neural tích chập (Convolutional neural networks), còn được biết đến với tên CNNs, là một dạng mạng neural được cấu thành bởi các tầng sau:

Các kiểu tầng

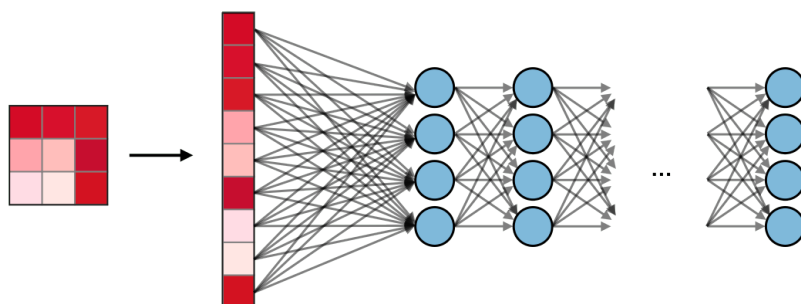
Tầng tích chập (CONV) - Tầng tích chập (CONV) sử dụng các bộ lọc để thực hiện phép tích chập khi đưa chúng đi qua đầu vào I theo các chiều của nó. Các siêu tham số của các bộ lọc này bao gồm kích thước bộ lọc F và độ trượt (stride) S . Kết quả đầu ra O được gọi là feature map hay activation map.



Hình 1: Tầng tích chập

Pooling (POOL) - Tầng pooling (POOL) là một phép downsampling, thường được sử dụng sau tầng tích chập, giúp tăng tính bất biến không gian. Cụ thể, max pooling và average pooling là những dạng pooling đặc biệt, mà tương ứng là trong đó giá trị lớn nhất và giá trị trung bình được lấy ra.

Fully Connected (FC) - Tầng kết nối đầy đủ (FC) nhận đầu vào là các dữ liệu đã được làm phẳng, mà mỗi đầu vào đó được kết nối đến tất cả neuron. Trong mô hình mạng CNNs, các tầng kết nối đầy đủ thường được tìm thấy ở cuối mạng và được dùng để tối ưu hóa mục tiêu của mạng ví dụ như độ chính xác của lớp.



Hình 2: Tầng kết nối đầy đủ

Hàm kích hoạt

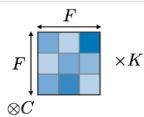

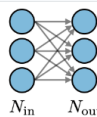
Các hàm kích hoạt (activation functions) đóng vai trò cực kỳ quan trọng trong các mạng nơ-ron tích chập (CNN) và các mạng nơ-ron nói chung. Chúng được sử dụng để tạo ra các đặc tính phi tuyến tính, giúp mạng có khả năng học các mô hình phức tạp hơn. Cụ thể hơn:

1. Tạo phi tuyến tính: Giúp mạng học các mối quan hệ phức tạp hơn thay vì bị giới hạn ở các phép toán tuyến tính.
2. Học đặc trưng: Giúp các lớp trong mạng trích xuất được đặc trưng từ cơ bản đến trừu tượng.
3. Hội tụ nhanh hơn: Giảm các vấn đề như vanishing gradients, tăng hiệu quả huấn luyện.
4. Tăng tính linh hoạt: Cho phép mạng xử lý các ánh xạ phức tạp từ đầu vào đến đầu ra.
5. Lọc thông tin: Nhấn mạnh đặc trưng quan trọng, giảm nhiễu (như ReLU chỉ giữ giá trị dương).

Các hàm phổ biến: ReLU, Sigmoid, Tanh, Leaky ReLU, Softmax.

Độ phức tạp của mô hình

Để đánh giá độ phức tạp của một mô hình, cách hữu hiệu là xác định số tham số mà mô hình đó sẽ có. Trong một tầng của mạng neural tích chập, nó sẽ được tính toán như sau:

	CONV	POOL	FC
Minh họa			
Kích thước đầu vào	$I \times I \times C$	$I \times I \times C$	N_{in}
Kích thước đầu ra	$O \times O \times K$	$O \times O \times C$	N_{out}
Số lượng tham số	$(F \times F \times C + 1) \cdot K$	0	$(N_{in} + 1) \times N_{out}$
Lưu ý	<ul style="list-style-type: none"> Một tham số bias với mỗi bộ lọc Trong đa số trường hợp, $S < F$ Một lựa chọn phổ biến cho K là $2C$ 	<ul style="list-style-type: none"> Phép pooling được áp dụng lên từng kênh (channel-wise) Trong đa số trường hợp, $S = F$ 	<ul style="list-style-type: none"> Đầu vào được làm phẳng Mỗi neuron có một tham số bias Số neuron trong một tầng FC phụ thuộc vào ràng buộc kết cấu

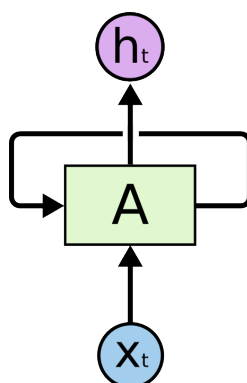
Hình 3: Tham số mạng tích chập

2.2 Long-short term memory

2.2.1 Mạng nơ ron truy hồi (RNN - Recurrent Neural Network)

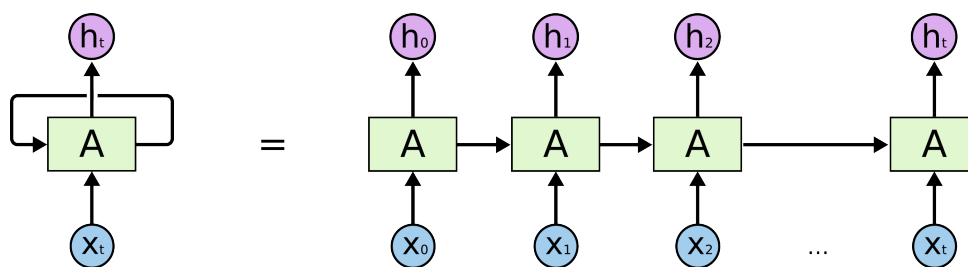
Khác với các mạng nơ-ron tích chập (CNN) thông thường – nơi toàn bộ dữ liệu đầu vào được xử lý đồng thời – mạng nơ-ron hồi tiếp (RNN – Recurrent Neural Network) được thiết kế đặc biệt để xử lý dữ liệu có tính chất tuần tự theo thời gian. Thay vì coi các mẫu dữ liệu là độc lập, RNN tận dụng thông tin từ các bước thời gian trước đó để đưa ra dự đoán tại bước hiện tại. Nhờ đó, mô hình có thể học được ngữ cảnh và mối quan hệ giữa các phần tử trong chuỗi.

Trong kiến trúc RNN, tại mỗi bước thời gian, một véc tơ đầu vào sẽ được truyền vào mạng, cùng với đầu ra (hay trạng thái ẩn) từ bước trước đó. Cơ chế "vòng lặp" đặc trưng trong thân mạng RNN chính là chìa khóa cho phép mô hình duy trì thông tin quá khứ và truyền qua các bước tiếp theo trong chuỗi.



Hình 4: Mạng nơ ron truy hồi với vòng lặp

Khi "trải phẳng" kiến trúc RNN theo trục thời gian, ta sẽ thấy các bước thời gian được xếp nối tiếp nhau, chia sẻ cùng trọng số mạng. Mỗi bước có thể được hiểu như một bản sao của cùng một mạng nơ-ron, liên kết với nhau theo chuỗi thời gian.



Hình 5: Cấu trúc trải phẳng của mạng nơ ron truy hồi

Ứng dụng thực tế của RNN rất đa dạng, từ nhận diện giọng nói, mô hình ngôn ngữ, dịch máy đến chú thích hình ảnh – những bài toán đều yêu cầu xử lý dữ liệu tuần tự và dựa trên ngữ cảnh xung quanh.

Một điểm mạnh rõ rệt của RNN là khả năng kết nối thông tin ngắn hạn, ví dụ như trong câu: “Học sinh đang tối trường học”, từ “trường học” có thể được dự đoán dễ dàng từ các từ ngay trước đó.

Tuy nhiên, khi ngữ cảnh kéo dài, như trong đoạn văn: “Hôm qua Bống đi học nhưng không mang áo mưa. Trên đường đi học trời mưa. Cặp sách của Bống bị ướt”, thì mô hình cần ghi nhớ các chi tiết từ nhiều câu trước như “không mang áo mưa” và “trời mưa” để dự đoán từ “ướt”. Đây là ví dụ điển hình của phụ thuộc dài hạn (long-term dependencies).

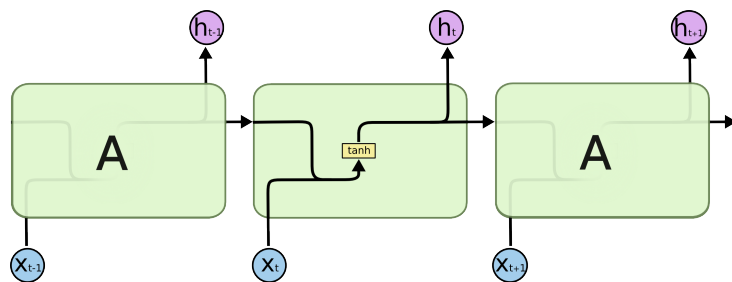
Mặc dù về lý thuyết RNN có thể học được các phụ thuộc dài hạn, nhưng trên thực tế mô hình thường gặp khó khăn vì hiện tượng triệt tiêu đạo hàm (vanishing gradient) khi lan truyền ngược qua nhiều bước thời gian. Điều này khiến những thông tin quan trọng từ quá khứ xa bị mờ nhạt dần.

Bài báo “Learning Long-Term Dependencies with Gradient Descent is Difficult” đã phân tích rõ ràng vấn đề này. Để khắc phục hạn chế đó, mô hình LSTM (Long Short-Term Memory) ra đời với một kiến trúc đặc biệt giúp lưu giữ thông tin lâu dài nhờ vào cơ chế cổng điều khiển.

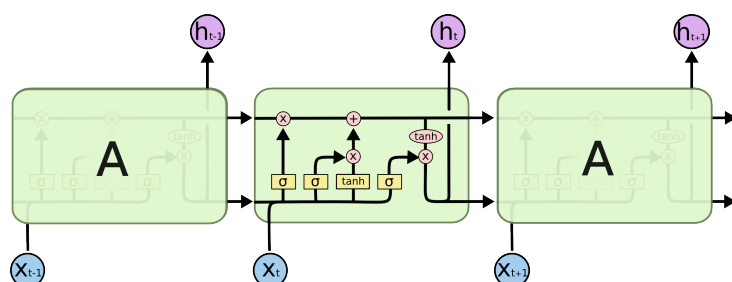
2.2.2 Mạng LSTM – Long Short-Term Memory

Mạng trí nhớ ngắn hạn định hướng dài hạn, còn gọi là LSTM (Long Short-Term Memory), là một kiến trúc đặc biệt của mạng nơ-ron hồi tiếp (RNN) được giới thiệu bởi Hochreiter và Schmidhuber vào năm 1997. LSTM được thiết kế để khắc phục những hạn chế của RNN truyền thống trong việc học các phụ thuộc dài hạn — cụ thể là hiện tượng triệt tiêu đạo hàm khiến mô hình khó học được thông tin từ các bước thời gian xa. Nhờ vào thiết kế đặc biệt, LSTM đã trở thành một trong những kiến trúc phổ biến nhất trong các bài toán liên quan đến dữ liệu chuỗi như dịch máy, tổng hợp văn bản, nhận dạng tiếng nói, và phân tích cảm xúc.

Trong khi RNN tiêu chuẩn chỉ có một tầng ẩn duy nhất, thường là hàm tanh, được lặp lại qua từng bước thời gian (xem Hình 6), thì LSTM mở rộng mỗi bước lặp này thành một cấu trúc phức tạp hơn gồm 4 tầng ẩn: 3 tầng sử dụng hàm kích hoạt sigmoid và 1 tầng sử dụng tanh (xem Hình 7). Điều này cho phép mô hình kiểm soát tốt hơn việc lưu giữ, ghi nhớ hay xóa bỏ thông tin tại mỗi thời điểm.



Hình 6: Kiến trúc RNN đơn giản

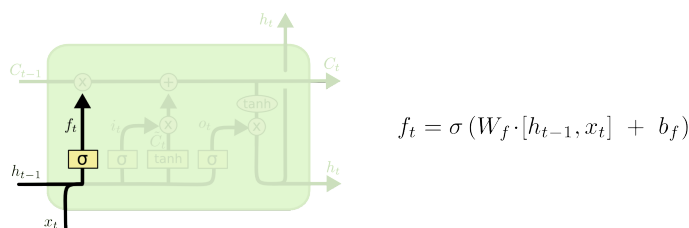


Hình 7: Kiến trúc một khối LSTM với 3 sigmoid và 1 tanh

LSTM được xây dựng dựa trên một ý tưởng cốt lõi: sử dụng một thành phần gọi là ô trạng thái (cell state) như một băng chuyền chạy xuyên suốt chuỗi thời gian (xem Hình 6). Đây là nơi chứa các thông tin quan trọng mà mô hình cần ghi nhớ lâu dài. Việc thêm hoặc xóa thông tin khỏi ô trạng thái được kiểm soát bởi các cổng (gates). Mỗi cổng là một tầng nơ-ron có hàm kích hoạt sigmoid và một phép nhân điểm, cho phép mô hình kiểm soát dòng chảy của thông tin (xem Hình 7). Giá trị sigmoid nằm trong khoảng từ 0 đến 1 sẽ quyết định tỉ lệ thông tin được truyền qua. Ví dụ, nếu đầu ra của sigmoid là 0.8, tức là 80% thông tin sẽ được giữ lại.

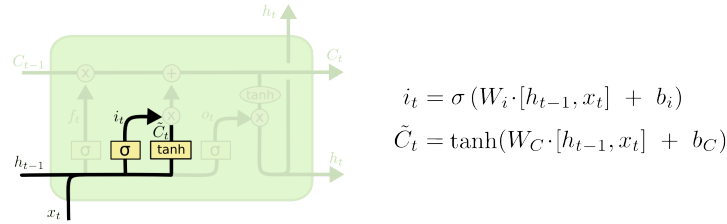
LSTM bao gồm ba cổng chính:

- Cổng quên (forget gate) quyết định phần nào của ô trạng thái cũ sẽ bị loại bỏ. Nó nhận đầu vào hiện tại và trạng thái ẩn trước đó, rồi tính ra một véc-tơ giá trị trong khoảng từ 0 đến 1 tương ứng với mỗi phần tử trong ô trạng thái (xem Hình 8).



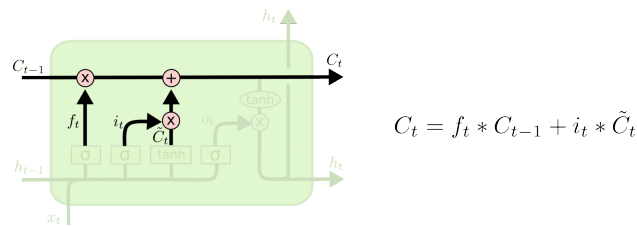
Hình 8: Tầng cổng quên (forget gate layer)

- Cổng vào (input gate) gồm hai thành phần: một tầng sigmoid để xác định thông tin mới nào nên được cập nhật, và một tầng tanh để tạo ra véc-tơ ứng viên trạng thái mới. Hai đầu ra này sẽ được kết hợp để quyết định thông tin nào sẽ được thêm vào ô trạng thái (xem Hình 9).



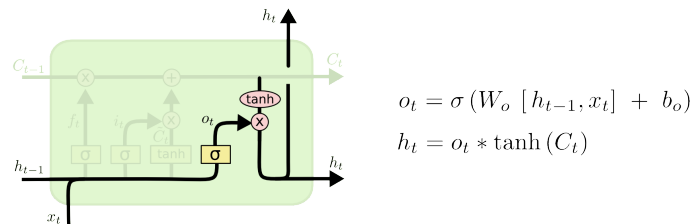
Hình 9: Cập nhật giá trị cho ô trạng thái bằng cách kết hợp 2 kết quả từ tầng cổng vào và tầng ẩn hàm tanh

- Sau đó, trạng thái mới của ô được tính bằng cách lấy trạng thái cũ nhân với giá trị từ cổng quên, cộng với phần ứng viên trạng thái mới nhân với giá trị từ cổng vào. Kết quả là một ô trạng thái đã được cập nhật (xem Hình 10).



Hình 10: Ô trạng thái mới

- Cuối cùng, cổng đầu ra (output gate) quyết định phần nào của ô trạng thái sẽ được xuất ra làm đầu ra của LSTM. Tầng sigmoid xác định vùng nào sẽ được đưa ra ngoài, rồi nhân với phiên bản đã được tanh hóa của ô trạng thái để tạo thành đầu ra (xem Hình 11).



Hình 11: Điều chỉnh thông tin ở đầu ra thông qua hàm tanh

Nhờ vào thiết kế thông minh và có kiểm soát này, LSTM có thể học được các mối liên hệ xa trong chuỗi dữ liệu mà không gặp vấn đề như RNN truyền thống. Nó chủ động quên đi thông tin không cần thiết và chỉ lưu giữ những gì có ý nghĩa cho việc dự đoán tiếp theo. Đây là lý do tại sao LSTM vẫn là một lựa chọn mạnh mẽ trong rất nhiều ứng dụng hiện đại liên quan đến dữ liệu thời gian.

2.3 Transformer

2.3.1 Transformer: Kiến trúc nền tảng cho xử lý ngôn ngữ tự nhiên hiện đại

Trước năm 2017, hầu hết các mô hình xử lý ngôn ngữ tự nhiên đều dựa vào kiến trúc tuần tự như RNN và LSTM để mô hình hóa mối quan hệ giữa các từ trong câu. Tuy

nhiên, các mô hình này gặp hạn chế trong việc học các quan hệ dài hạn và khó tối ưu song song. Sự ra đời của kiến trúc Transformer (Vaswani et al., 2017) đã tạo ra bước đột phá lớn. Khác với các mô hình tuần tự, Transformer sử dụng hoàn toàn cơ chế Self-Attention để mô hình hóa mối quan hệ giữa tất cả các từ trong chuỗi cùng một lúc, bất kể khoảng cách. Điều này không chỉ cải thiện khả năng ghi nhớ các phụ thuộc dài hạn mà còn cho phép huấn luyện mô hình nhanh hơn nhờ tính chất song song hóa.

Kiến trúc Transformer bao gồm hai thành phần chính: Encoder và Decoder. Trong đó, Encoder chịu trách nhiệm mã hóa thông tin đầu vào thành các biểu diễn ngữ nghĩa, còn Decoder sử dụng các biểu diễn đó để sinh ra đầu ra. Chính nhờ những ưu điểm vượt trội về hiệu suất và khả năng học ngữ cảnh sâu sắc, Transformer đã nhanh chóng trở thành nền tảng cho hầu hết các mô hình NLP hiện đại.

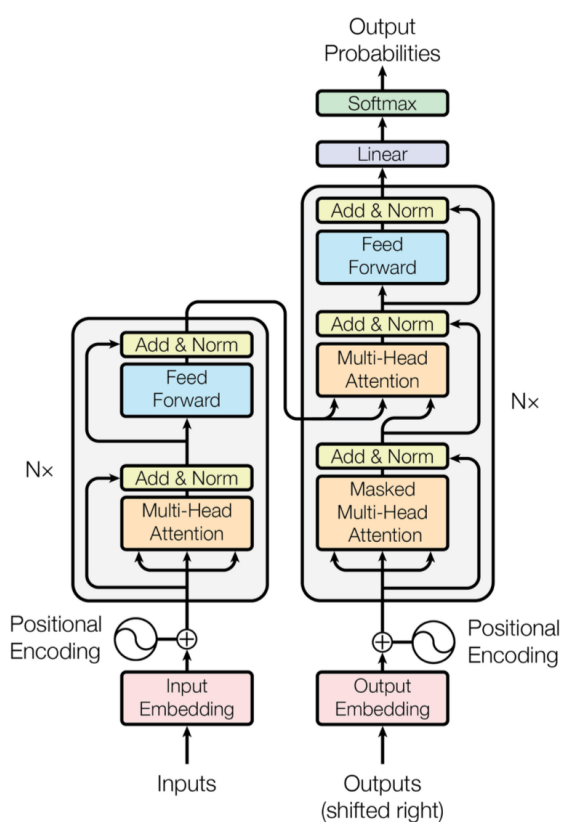


Figure 1: The Transformer - model architecture.

Hình 12: Kiến trúc transformer

2.3.2 BERT: Biểu diễn ngữ nghĩa hai chiều mạnh mẽ

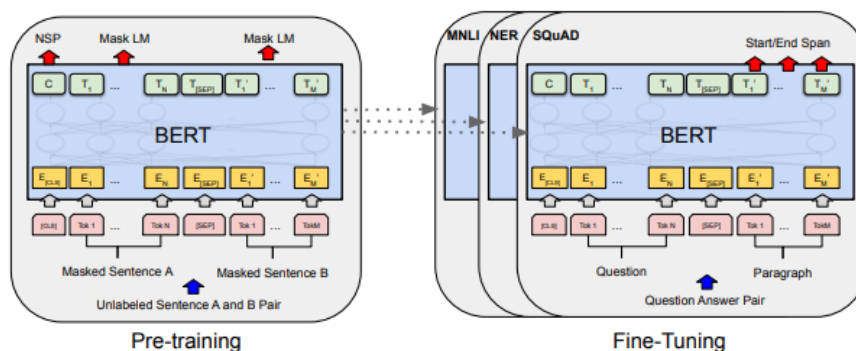
Dựa trên sức mạnh của Transformer Encoder, nhóm nghiên cứu của Google đã đề xuất mô hình BERT (Bidirectional Encoder Representations from Transformers) vào năm 2018. Không giống như các mô hình chỉ học ngữ cảnh từ trái sang phải hoặc phải sang trái trước đó, BERT tận dụng khả năng học hai chiều (bidirectional) cùng lúc, nhờ đó hiểu ngữ cảnh của mỗi từ trong câu một cách toàn diện hơn.

BERT được huấn luyện theo hai nhiệm vụ chính:

- Masked Language Modeling (MLM): Ngẫu nhiên che giấu một số từ trong câu và

yêu cầu mô hình dự đoán lại từ đó, giúp mô hình học được thông tin ngữ cảnh từ cả hai phía.

- Next Sentence Prediction (NSP): Dự đoán mối quan hệ giữa hai câu liên tiếp để học hiểu về mức độ kết nối giữa các câu.



Hình 13: Kiến trúc BERT

Nhờ phương pháp huấn luyện mới lạ này, BERT đã nhanh chóng đạt được kết quả vượt trội trên nhiều bài toán NLP như phân loại văn bản, trả lời câu hỏi, phân tích cảm xúc, v.v.

2.3.3 PhoBERT: BERT dành riêng cho tiếng Việt

Mặc dù BERT gốc rất mạnh mẽ, nhưng nó chủ yếu được huấn luyện trên tập dữ liệu tiếng Anh (BooksCorpus và Wikipedia tiếng Anh). Khi áp dụng cho các ngôn ngữ khác như tiếng Việt, mô hình gặp nhiều hạn chế do sự khác biệt về cú pháp, ngữ nghĩa và cấu trúc ngôn ngữ.

Để khắc phục vấn đề này, nhóm nghiên cứu tại VinAI Research đã phát triển PhoBERT – một mô hình BERT được tiền huấn luyện hoàn toàn trên tập dữ liệu tiếng Việt lớn, bao gồm 20GB văn bản tiếng Việt từ Wikipedia và các nguồn khác. PhoBERT giữ nguyên kiến trúc như BERT-base nhưng thay đổi cách token hóa, sử dụng kỹ thuật Byte-Pair Encoding (BPE) thay cho WordPiece, nhằm tối ưu hóa cho đặc trưng tiếng Việt, nơi từ ngữ thường bao gồm nhiều âm tiết cách nhau bởi dấu cách.

Hiện tại, PhoBERT có hai phiên bản:

- PhoBERT-base (135M tham số)
- PhoBERT-large (370M tham số)

PhoBERT đã đạt được kết quả vượt trội trên nhiều tác vụ xử lý tiếng Việt như phân tích cảm xúc, nhận diện thực thể (NER), phân loại văn bản, và phân tách câu.

Trong khuôn khổ đề tài này, em sử dụng mô hình PhoBERT-base để thực hiện fine-tuning cho bài toán phân tích cảm xúc tiếng Việt, nhằm tận dụng tối đa khả năng biểu diễn ngữ nghĩa ngôn ngữ tự nhiên của mô hình.

2.4 Biểu diễn từ và kỹ thuật Word Embedding

Trong xử lý ngôn ngữ tự nhiên, từ ngữ là đơn vị cơ bản truyền tải ngữ nghĩa của ngôn ngữ. Tuy nhiên, để các mô hình học máy có thể xử lý được văn bản, cần thiết phải chuyển đổi các từ sang dạng số. Đây là bài toán biểu diễn từ (word representation), tức là tìm một cách biểu diễn toán học cho mỗi từ sao cho vừa phù hợp với mô hình tính toán, vừa giữ lại được thông tin ngữ nghĩa của từ.

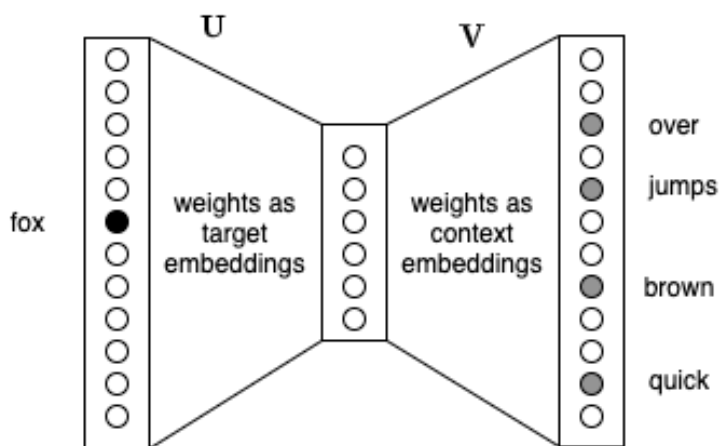
Một trong những phương pháp đơn giản nhất để biểu diễn từ là sử dụng vector one-hot, trong đó mỗi từ được ánh xạ thành một vector nhị phân có độ dài bằng kích thước từ điển, với đúng một phần tử có giá trị bằng 1, còn lại là 0. Mặc dù dễ cài đặt, vector one-hot có nhiều hạn chế, đặc biệt là không thể hiện được mối quan hệ ngữ nghĩa giữa các từ. Cụ thể, độ tương tự (ví dụ: cosine similarity) giữa hai vector one-hot bất kỳ đều bằng 0 nếu chúng khác nhau, do chúng trực giao trong không gian.

Để khắc phục hạn chế trên, các phương pháp embedding từ đã ra đời, trong đó nổi bật là Word2Vec. Đây là kỹ thuật ánh xạ mỗi từ thành một vector thực có chiều cố định (thường là 100–300 chiều), sao cho các từ có ngữ nghĩa gần nhau sẽ có vector gần nhau trong không gian vector. Word2Vec giúp mô hình học được các mối quan hệ ngữ nghĩa và cú pháp giữa các từ, chẳng hạn như: $\text{vec}(\text{"king"}) - \text{vec}(\text{"man"}) + \text{vec}(\text{"woman"}) = \text{vec}(\text{"queen"})$.

Word2Vec là một kỹ thuật biểu diễn từ được huấn luyện thông qua một tác vụ ngữ cảnh đơn giản, bao gồm hai mô hình cơ bản:

- Skip-Gram: Dự đoán từ ngữ cảnh dựa vào từ trung tâm.
Skip-Gram giả định rằng một từ trung tâm có thể sinh ra các từ ngữ cảnh xung quanh nó. Ví dụ, với câu “the man loves his son”, nếu chọn “loves” là từ trung tâm và cửa sổ ngữ cảnh là 2, thì Skip-Gram sẽ cố gắng dự đoán các từ “the”, “man”, “his”, “son” dựa vào “loves”.

Mỗi từ được gán một vector khi đóng vai trò từ trung tâm và một vector khác khi đóng vai trò từ ngữ cảnh. Mô hình sử dụng hàm softmax để tính xác suất có điều kiện sinh ra một từ ngữ cảnh dựa trên từ trung tâm, sau đó tối ưu hàm mất mát bằng phương pháp đạo hàm ngược và gradient descent.

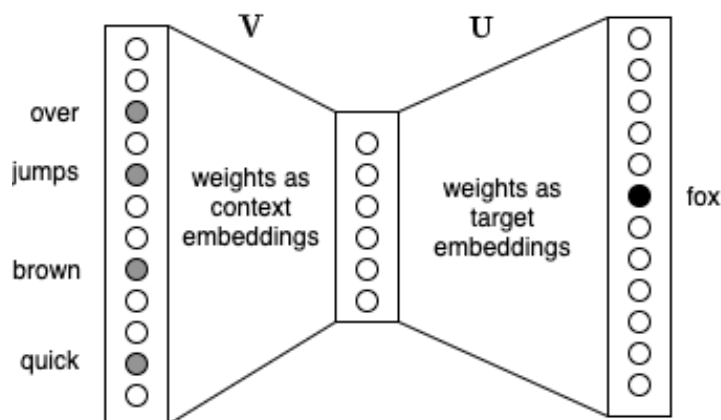


Hình 14: Kiến trúc Skip-gram

- CBOW (Continuous Bag of Words): Dự đoán từ trung tâm dựa trên các từ ngữ cảnh.

Ngược lại với Skip-Gram, CBOW dự đoán từ trung tâm dựa trên các từ ngữ cảnh xung quanh. Với cùng ví dụ trên, CBOW sẽ sử dụng “the”, “man”, “his”, “son” để dự đoán từ “loves”.

CBOW lấy trung bình các vector từ ngữ cảnh và sử dụng softmax để dự đoán từ trung tâm. Quá trình huấn luyện CBOW cũng sử dụng tối ưu hóa gradient và thường có tốc độ huấn luyện nhanh hơn Skip-Gram.



Hình 15: Kiến trúc CBOW

3 Kiến trúc đề xuất và thực nghiệm

Trong khuôn khổ đề tài, em lựa chọn sử dụng một mô hình Word2Vec được huấn luyện sẵn (pre-trained Word2Vec) nhằm mục đích cải thiện hiệu quả mô hình phân loại cảm xúc. Việc sử dụng embedding từ được huấn luyện trước trên tập dữ liệu lớn và đa dạng giúp mô hình có khả năng tiếp cận với các biểu diễn từ giàu ngữ nghĩa, từ đó nâng cao khả năng tổng quát hóa và tăng độ chính xác trên tập dữ liệu huấn luyện cũng như kiểm tra.

So với việc khởi tạo vector từ ngẫu nhiên và huấn luyện cùng mô hình, embedding tiền huấn luyện giúp rút ngắn thời gian hội tụ và hỗ trợ mô hình hiểu rõ hơn mối liên hệ ngữ nghĩa giữa các từ — đặc biệt là các từ mang sắc thái cảm xúc như “không thích”, “rất tốt”, “bình thường”, v.v. Đây là một bước quan trọng trong việc chuẩn bị dữ liệu đầu vào cho các kiến trúc học sâu như CNN, LSTM hoặc hybrid.

Trong quá trình thực nghiệm, embedding này sẽ được tích hợp vào lớp đầu vào của mô hình và có thể lựa chọn cấu hình đóng băng (không huấn luyện lại) hoặc fine-tune (tiếp tục cập nhật) tùy theo đặc điểm của tập dữ liệu và hiệu suất trên tập kiểm tra.

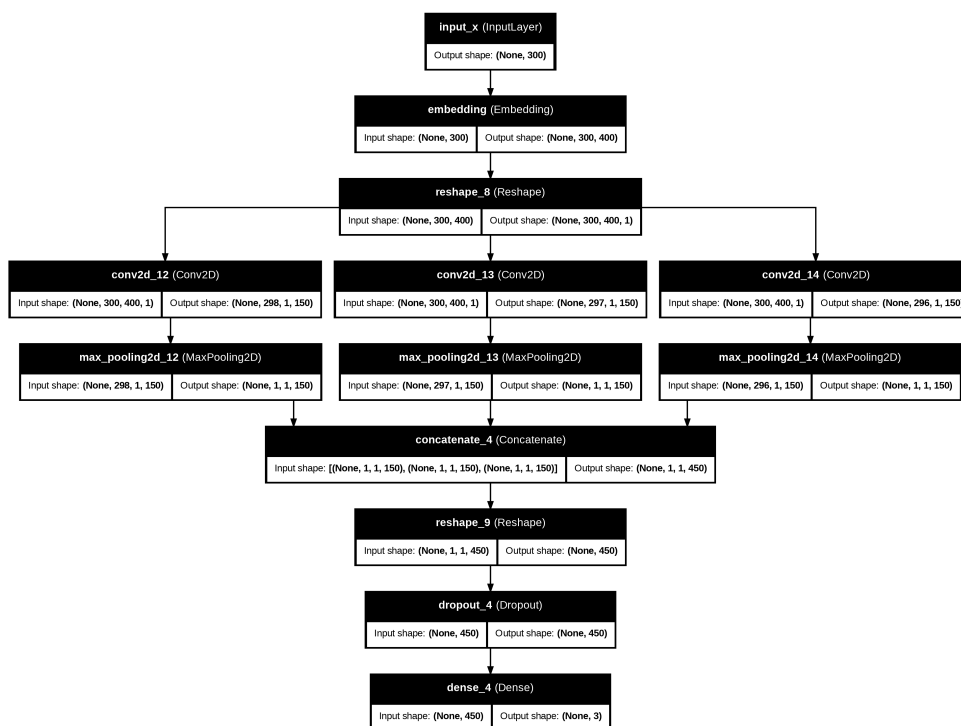
3.1 CNN

Mạng nơ-ron tích chập (CNN) được sử dụng phổ biến trong xử lý ảnh nhưng cũng cho thấy hiệu quả cao trong các bài toán xử lý văn bản, đặc biệt là phân loại văn bản. Khi áp dụng cho văn bản, CNN có khả năng trích xuất các đặc trưng cục bộ bằng cách sử dụng các bộ lọc (filter) trượt trên chuỗi từ. Mỗi filter có thể được hiểu là một công cụ phát hiện các mẫu n-gram (như bigram, trigram) mang tính biểu cảm cao, ví dụ như “rất tốt”, “quá tệ”, “không thích”, giúp mô hình nhận diện cảm xúc mà không cần hiểu toàn bộ ngữ cảnh.

Mô hình TextCNN là một ứng dụng cụ thể của CNN cho bài toán phân loại văn bản. Trong TextCNN, mỗi câu đầu vào được ánh xạ thành ma trận nhúng (embedding), sau đó đi qua nhiều tầng tích chập song song với các kích thước kernel khác nhau (ví dụ 3, 4, 5) để trích xuất các đặc trưng từ các cụm từ có độ dài khác nhau. Sau đó, các đặc trưng được chọn lọc thông qua tầng MaxPooling, gộp lại thành một vector duy nhất, rồi đưa vào tầng Dense để thực hiện phân loại.

TextCNN có ưu điểm nổi bật là đơn giản, huấn luyện nhanh, và hoạt động tốt với văn bản ngắn hoặc trung bình. Khác với các mô hình tuần tự như RNN hoặc LSTM, CNN xử lý toàn bộ câu song song và không phụ thuộc vào thứ tự từ, điều này đặc biệt hữu ích trong các trường hợp mà từ khóa biểu cảm có thể xuất hiện ở bất kỳ vị trí nào trong câu.

Kiến trúc sử dụng như sau:



Hình 16: Kiến trúc CNN đề xuất

Mô hình sử dụng kiến trúc CNN áp dụng cho dữ liệu văn bản theo hướng của TextCNN, được thiết kế để trích xuất đặc trưng cục bộ từ câu đầu vào bằng các tầng tích chập 2D với nhiều kích thước kernel khác nhau.

Câu đầu vào có độ dài cố định là 300 từ, được ánh xạ sang không gian vector 400 chiều thông qua lớp Embedding, tạo ra tensor đầu vào có kích thước (300, 400). Sau đó, tensor này được reshape để thêm chiều kênh, tạo thành (300, 400, 1) phù hợp với Conv2D.

Mô hình sử dụng ba nhánh tích chập song song với kích thước kernel tương ứng với các n-gram: 3, 4 và 5. Mỗi nhánh Conv2D có 150 filters, cho ra các đầu ra có chiều cao lần lượt là 298, 297 và 296 tùy theo kernel size. Sau mỗi tầng tích chập, mô hình áp dụng MaxPooling2D để chọn đặc trưng mạnh nhất từ mỗi nhánh, kết quả thu được là ba tensor có cùng kích thước (1, 1, 150).

Các đặc trưng từ ba nhánh này được gộp lại bằng lớp Concatenate, tạo ra tensor tổng hợp có kích thước (1, 1, 450). Sau đó tensor này được reshape thành vector một chiều (450), đi qua lớp Dropout để giảm overfitting và cuối cùng là Dense với hàm softmax để phân loại thành ba nhãn cảm xúc.

Layer (type)	Output Shape	Param #	Connected to
input_x (InputLayer)	(None, 300)	0	-
embedding (Embedding)	(None, 300, 400)	4,000,000	input_x[0][0]
reshape_8 (Reshape)	(None, 300, 400, 1)	0	embedding[0][0]
conv2d_12 (Conv2D)	(None, 298, 1, 150)	180,150	reshape_8[0][0]
conv2d_13 (Conv2D)	(None, 297, 1, 150)	240,150	reshape_8[0][0]
conv2d_14 (Conv2D)	(None, 296, 1, 150)	300,150	reshape_8[0][0]
max_pooling2d_12 (MaxPooling2D)	(None, 1, 1, 150)	0	conv2d_12[0][0]
max_pooling2d_13 (MaxPooling2D)	(None, 1, 1, 150)	0	conv2d_13[0][0]
max_pooling2d_14 (MaxPooling2D)	(None, 1, 1, 150)	0	conv2d_14[0][0]
concatenate_4 (Concatenate)	(None, 1, 1, 450)	0	max_pooling2d_12[0][0... max_pooling2d_13[0][0... max_pooling2d_14[0][0]
reshape_9 (Reshape)	(None, 450)	0	concatenate_4[0][0]
dropout_4 (Dropout)	(None, 450)	0	reshape_9[0][0]
dense_4 (Dense)	(None, 3)	1,353	dropout_4[0][0]

Total params: 4,721,803 (18.01 MB)

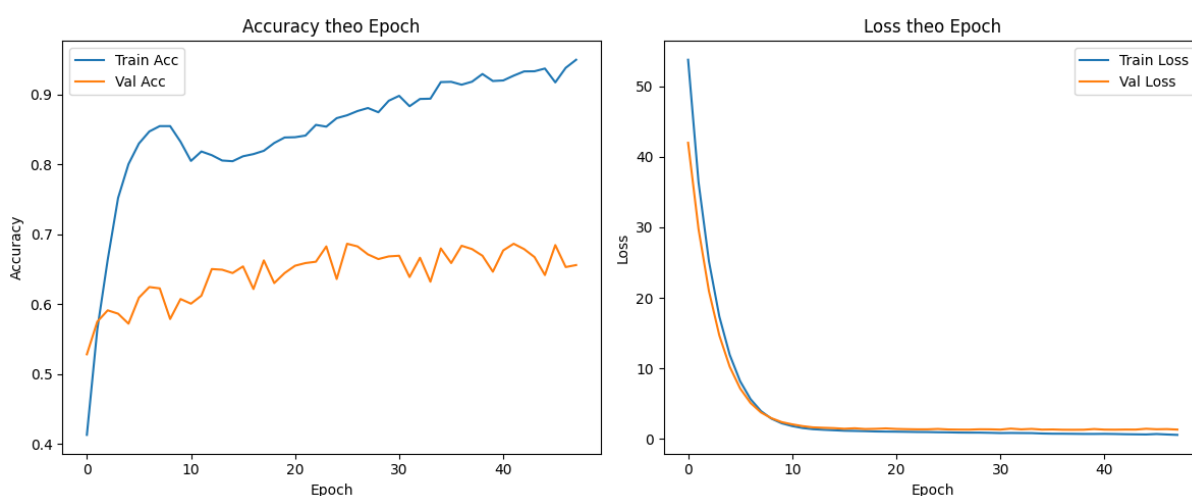
Trainable params: 4,721,803 (18.01 MB)

Non-trainable params: 0 (0.00 B)

None

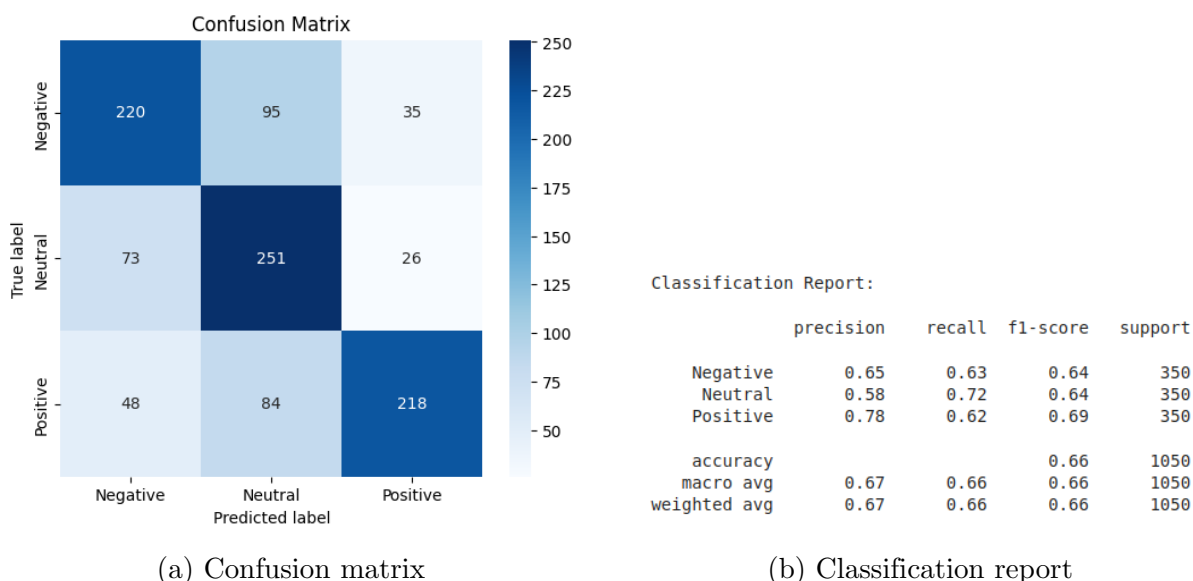
Hình 17: Số lượng tham số của từng lớp

Sau khi huấn luyện, ta thu được các kết quả như sau:



Hình 18: Kết quả huấn luyện mô hình

Đồng thời, kết quả khi sử dụng lên tập test như sau:



Hình 19: Kết quả chạy trên tập test

Một số kết luận rút ra từ các biểu đồ của việc huấn luyện:

- Lớp Neutral có kết quả tốt nhất với 251 mẫu đúng trên tổng số khoảng 350 ($\approx 71.7\%$). Đây là dấu hiệu cho thấy mô hình có khả năng phân biệt rõ các văn bản mang sắc thái trung lập.
- Lớp Negative có 220 mẫu đúng, tuy nhiên số lượng bị nhầm sang Neutral tương đối cao (95 mẫu), cho thấy mô hình gặp khó khăn trong việc phân biệt cảm xúc tiêu cực nhẹ và trung tính.
- Lớp Positive cũng cho kết quả khá tốt với 218 mẫu đúng, tuy nhiên lại bị nhầm nhiều sang Neutral (84 mẫu), cho thấy mô hình vẫn có xu hướng "an toàn", dễ gán nhãn trung tính khi không chắc chắn.

3.2 LSTM

Trong lĩnh vực xử lý ngôn ngữ tự nhiên, các mô hình mạng nơ-ron hồi tiếp (Recurrent Neural Networks – RNN) như LSTM (Long Short-Term Memory) đã được ứng dụng rộng rãi nhờ khả năng ghi nhớ thông tin theo chuỗi thời gian. LSTM đặc biệt hiệu quả trong việc xử lý các câu văn dài, nơi mà ngữ nghĩa của từ hiện tại có thể phụ thuộc vào những từ xuất hiện trước đó. Tuy nhiên, LSTM truyền thống vẫn tồn tại một số nhược điểm quan trọng.

Thứ nhất, LSTM chỉ xử lý theo một chiều thời gian – từ quá khứ đến hiện tại. Điều này dẫn đến việc mô hình không thể khai thác thông tin ngữ cảnh ở phía sau từ hiện tại. Trong nhiều trường hợp, đặc biệt với ngôn ngữ tự nhiên như tiếng Việt hay tiếng Anh, ý nghĩa của một câu không chỉ phụ thuộc vào những từ trước đó, mà còn vào những từ phía sau. Ví dụ, xét câu:

“Dù diễn viên chính diễn xuất chưa thật sự xuất sắc, nhưng bộ phim lại rất cảm động.”

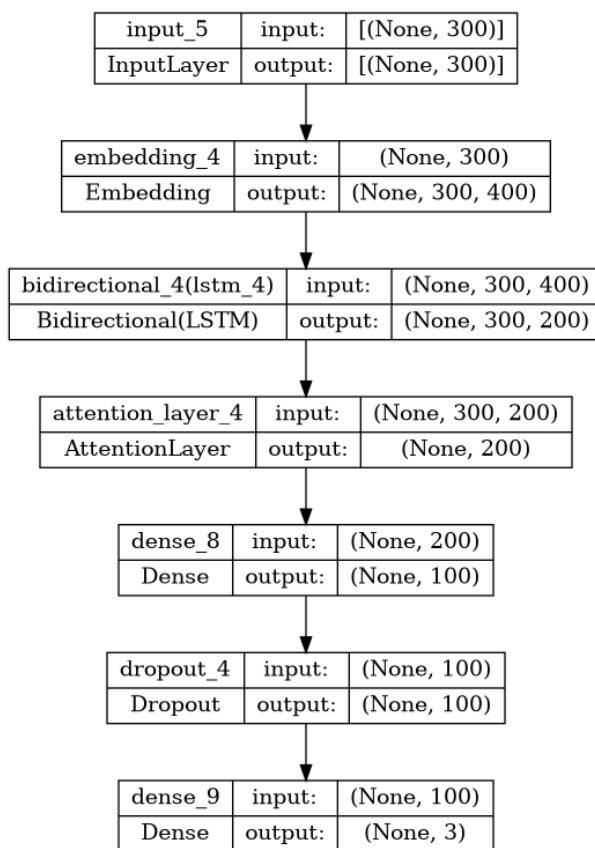
Nếu chỉ đọc đến phần đầu, mô hình có thể vội vàng đánh giá cảm xúc tiêu cực. Tuy nhiên, phần sau “nhưng bộ phim lại rất cảm động” lại mang tính chất đảo ngược cảm xúc. Việc chỉ xử lý một chiều khiến LSTM có thể bỏ sót thông tin mang tính quyết định ở phía sau câu.

Thứ hai, LSTM không có cơ chế nào để tự động xác định từ nào là “quan trọng” hơn các từ khác trong việc ra quyết định. Trong thực tế, không phải tất cả các từ trong câu đều có mức độ ảnh hưởng như nhau. Những từ như “không”, “rất”, “tuyệt vời”, “kinh khủng”, “nhưng”,... thường có vai trò quyết định trong phân tích cảm xúc. Tuy nhiên, LSTM xử lý tuần tự và phân bổ trọng số đều cho tất cả các bước thời gian, dẫn đến nguy cơ “loãng” thông tin quan trọng.

Để khắc phục hai hạn chế kể trên, trong đề tài này, em đề xuất sử dụng mô hình kết hợp giữa Bi-directional LSTM (BiLSTM) và Attention Mechanism.

- BiLSTM là một biến thể mở rộng của LSTM, cho phép mô hình học thông tin theo cả hai chiều thời gian: từ quá khứ đến hiện tại và từ tương lai về quá khứ. Điều này giúp mô hình hiểu được ngữ cảnh toàn vẹn của mỗi từ trong câu, đặc biệt hữu ích khi từ đó chịu ảnh hưởng từ cả những từ đứng trước lẫn sau.
- Attention Mechanism là một kỹ thuật giúp mô hình “tập trung” nhiều hơn vào các từ có tầm quan trọng cao, bằng cách gán trọng số khác nhau cho các từ trong chuỗi đầu vào. Điều này tương tự như cách con người đọc một câu và đặc biệt chú ý đến một vài từ khóa quan trọng hơn phần còn lại.

Tổng thể, sự kết hợp giữa BiLSTM và Attention tạo nên một mô hình mạnh mẽ hơn trong việc phân tích ngữ nghĩa toàn diện và phát hiện trọng tâm cảm xúc trong câu. Đây chính là nền tảng cho kiến trúc đề xuất trong đề tài này. Kiến trúc đề xuất như sau:



Hình 20: Kiến trúc lstm đề xuất

Kiến trúc trên dựa trên sự kết hợp giữa hai thành phần chính: mạng Bi-directional LSTM (BiLSTM) và cơ chế Attention, nhằm cải thiện hiệu quả phân loại cảm xúc trong văn bản tiếng Việt:

1. Lớp nhúng (Embedding Layer)

Dữ liệu văn bản đầu vào được mã hóa thông qua lớp nhúng, sử dụng Word2Vec hoặc embedding tiền huấn luyện phù hợp với ngôn ngữ tiếng Việt. Mỗi từ được ánh xạ thành một vector có chiều cố định, giữ lại các thông tin ngữ nghĩa của từ.

2. BiLSTM Layer

Sau khi được ánh xạ sang không gian vector, chuỗi từ đi qua lớp BiLSTM. Khác với LSTM thông thường chỉ xử lý chuỗi từ theo một chiều (từ trái sang phải), BiLSTM đồng thời học thông tin theo cả hai chiều – từ trái sang phải và từ phải sang trái. Điều này cho phép mô hình hiểu được ngữ cảnh đầy đủ xung quanh mỗi từ, từ đó tăng độ chính xác trong việc diễn giải ý nghĩa câu.

3. Attention Layer

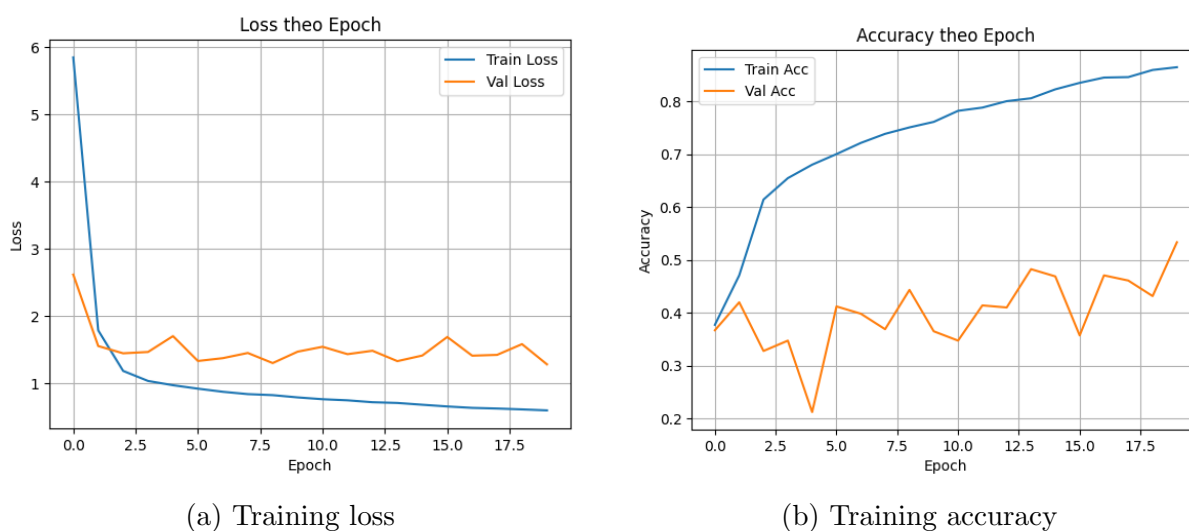
Đầu ra của BiLSTM là một chuỗi các vector biểu diễn từng từ trong câu, đã được tích hợp ngữ cảnh hai chiều. Tuy nhiên, không phải tất cả các vector đều đóng vai trò như nhau. Do đó, mô hình sử dụng thêm Attention Layer – một lớp học trọng số để xác định “mức độ đóng góp” của từng từ vào kết quả phân loại cuối cùng. Cơ chế attention giúp mô hình tập trung nhiều hơn vào những từ mang tính cảm xúc mạnh như “tệ”, “tuyệt vời”, “nhưng”, “không”, v.v.

Cụ thể, attention tạo ra một vector ngữ cảnh có trọng số, bằng cách nhân từng vector đầu ra của BiLSTM với hệ số attention tương ứng, sau đó tổng hợp lại. Kết quả là một vector biểu diễn toàn bộ câu, nhưng đã được làm nổi bật các phần quan trọng.

4. Dense Layer và Output

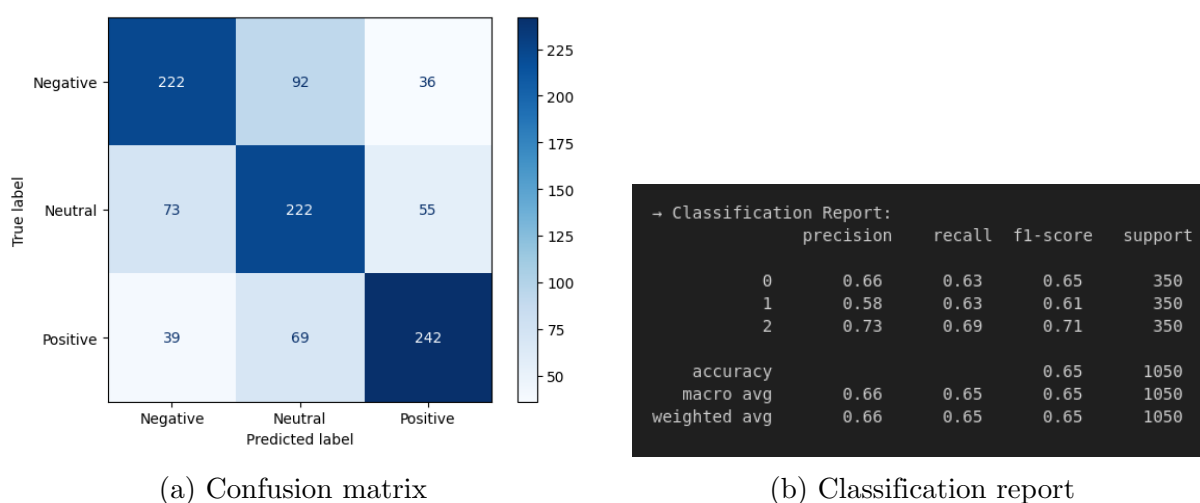
Vector ngữ cảnh từ attention sẽ được đưa qua một hoặc nhiều lớp Dense (mạng fully connected), sau đó sử dụng hàm Softmax để phân loại cảm xúc đầu ra thành các nhãn: tiêu cực, trung tính hoặc tích cực.

Kết quả huấn luyện trên tập dữ liệu cho trước:



Hình 21: Kết quả huấn luyện mô hình

Đồng thời, kết quả khi sử dụng lên tập test như sau:



Hình 22: Kết quả chạy trên tập test

Nhìn chung, mô hình thu được độ chính xác rơi vào khoảng 65,33%. Đồng thời ta có một số nhận xét như sau:

- Lớp Positive là lớp có kết quả tốt nhất, với 242/350 mẫu được phân loại chính xác ($\approx 69.1\%$).
- Lớp Neutral có tỷ lệ đúng là 222/350 mẫu ($\approx 63.4\%$), tuy nhiên có sự nhầm lẫn đáng kể với cả hai lớp còn lại.
- Lớp Negative có kết quả 222/350 đúng ($\approx 63.4\%$), nhưng bị nhầm lẫn nhiều nhất với lớp Neutral (92 mẫu) – cho thấy mô hình khó phân biệt giữa cảm xúc tiêu cực nhẹ và trung tính.

3.3 Hybrid CNN+LSTM

Trong bài toán phân loại cảm xúc văn bản, yêu cầu mô hình không chỉ phát hiện các cụm từ mang tính biểu cảm mà còn phải hiểu được mối liên hệ giữa các từ theo thứ tự trong câu. Do đó, việc kết hợp hai kiến trúc phổ biến là CNN và LSTM được xem là phù hợp để giải quyết đồng thời hai yêu cầu trên.

CNN có khả năng trích xuất đặc trưng cục bộ (local features) thông qua các bộ lọc tích chập. Trong xử lý văn bản, CNN có thể phát hiện các mẫu ngữ nghĩa ngắn như các cụm từ cảm xúc (n-grams), ví dụ như “quá tốt”, “không thích”, “rất dở”, bất kể vị trí của chúng trong câu.

- Tốc độ huấn luyện nhanh do khả năng song song hóa.
- Có khả năng học các mẫu biểu hiện cảm xúc mạnh một cách hiệu quả.
- Tuy nhiên, CNN không xử lý được mối quan hệ theo thứ tự giữa các từ.

LSTM là một dạng mạng nơ-ron hồi tiếp được thiết kế để xử lý chuỗi dữ liệu, có khả năng ghi nhớ thông tin trong dài hạn. Trong xử lý văn bản, LSTM hỗ trợ mô hình học được sự phụ thuộc ngữ nghĩa theo thứ tự từ đầu đến cuối câu.

- Thích hợp với các câu có cấu trúc logic rõ ràng hoặc có chứa các biểu thức phủ định, chuyển ý.
- Giải quyết được vấn đề gradient biến mất trong RNN truyền thống.
- Tuy nhiên, quá trình huấn luyện tuần tự nên chậm hơn và kém hiệu quả trong việc phát hiện đặc trưng cục bộ.

LSTM là một dạng mạng nơ-ron hồi tiếp được thiết kế để xử lý chuỗi dữ liệu, có khả năng ghi nhớ thông tin trong dài hạn. Trong xử lý văn bản, LSTM hỗ trợ mô hình học được sự phụ thuộc ngữ nghĩa theo thứ tự từ đầu đến cuối câu.

- Thích hợp với các câu có cấu trúc logic rõ ràng hoặc có chứa các biểu thức phủ định, chuyển ý.
- Giải quyết được vấn đề gradient biến mất trong RNN truyền thống.
- Tuy nhiên, quá trình huấn luyện tuần tự nên chậm hơn và kém hiệu quả trong việc phát hiện đặc trưng cục bộ.

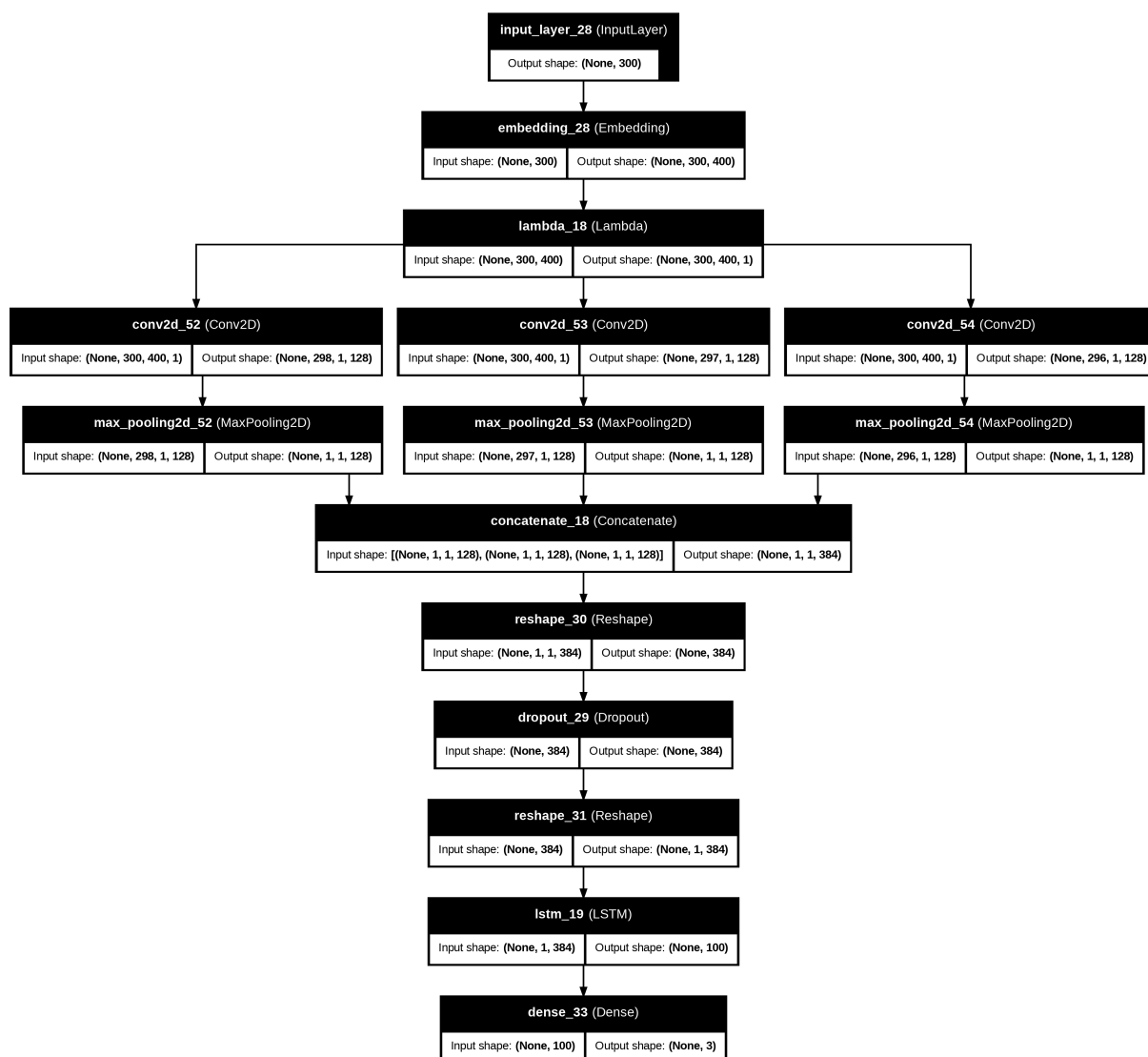
Việc kết hợp hai mô hình theo thứ tự CNN → LSTM cho phép tận dụng ưu điểm của cả hai kiến trúc:

- CNN đóng vai trò trích xuất đặc trưng cục bộ từ câu đầu vào.
- LSTM tiếp nhận các đặc trưng này và học được mối quan hệ theo chuỗi thời gian giữa các đặc trưng đó.

Tổng thể, mô hình kết hợp CNN và LSTM giúp cải thiện khả năng phân loại cảm xúc nhờ:

- Nhận diện các cụm từ biểu cảm quan trọng.
- Hiểu rõ ngữ cảnh và mối liên hệ giữa các thành phần trong câu.

Kiến trúc đề xuất cho việc kết hợp CNN+LSTM như sau:



Hình 23: Kiến trúc hybrid đề xuất

Mô tả kiến trúc mô hình CNN + LSTM:

Kiến trúc mô hình gồm hai thành phần chính: mạng Convolutional 2D (CNN) sử dụng nhiều kích thước kernel và mạng LSTM để học mối quan hệ theo chuỗi.

1. Input & Embedding

- Input Layer: đầu vào là chuỗi văn bản có độ dài cố định là 300 từ.
- Embedding Layer: ánh xạ từ các chỉ số từ vựng thành vector 400 chiều, đầu ra có kích thước (300, 400).

2. Tăng chiều & Tích chập

- Tensor được tăng chiều từ (300, 400) lên (300, 400, 1) để phù hợp với Conv2D.
- Mô hình có 3 nhánh Conv2D song song tương ứng với kernel sizes lần lượt là 3, 4 và 5. Mỗi nhánh có 128 filters.
- Mỗi nhánh sau Conv2D sẽ đi qua MaxPooling2D để rút gọn đặc trưng, đầu ra chuẩn hóa về kích thước (1, 1, 128).

3. Ghép đặc trưng & Reshape

- Các đặc trưng từ 3 nhánh được ghép lại bằng Concatenate, tạo tensor có kích thước (1, 1, 384).
- Sau đó reshape thành vector 1 chiều (384).

4. Dropout & LSTM

- Thực hiện Dropout để giảm overfitting.
- Reshape lại để phù hợp với yêu cầu của LSTM.
- LSTM Layer: có 100 đơn vị ẩn, xử lý chuỗi có chiều dài 1 với đặc trưng 384.

5. Dense Output

- Kết quả từ LSTM được đưa qua lớp Dense 100 đơn vị \rightarrow sau đó lớp Dense(3) với hàm softmax để phân loại thành 3 lớp cảm xúc.

Layer (type)	Output Shape	Param #	Connected to
input_layer_27 (InputLayer)	(None, 300)	0	-
embedding_27 (Embedding)	(None, 300, 400)	3,167,600	input_layer_27[0][0]
lambda_17 (Lambda)	(None, 300, 400, 1)	0	embedding_27[0][0]
conv2d_49 (Conv2D)	(None, 298, 1, 128)	153,728	lambda_17[0][0]
conv2d_50 (Conv2D)	(None, 297, 1, 128)	204,928	lambda_17[0][0]
conv2d_51 (Conv2D)	(None, 296, 1, 128)	256,128	lambda_17[0][0]
max_pooling2d_49 (MaxPooling2D)	(None, 1, 1, 128)	0	conv2d_49[0][0]
max_pooling2d_50 (MaxPooling2D)	(None, 1, 1, 128)	0	conv2d_50[0][0]
max_pooling2d_51 (MaxPooling2D)	(None, 1, 1, 128)	0	conv2d_51[0][0]
concatenate_17 (Concatenate)	(None, 1, 1, 384)	0	max_pooling2d_49[0][0... max_pooling2d_50[0][0... max_pooling2d_51[0][0]
reshape_28 (Reshape)	(None, 384)	0	concatenate_17[0][0]
dropout_28 (Dropout)	(None, 384)	0	reshape_28[0][0]
reshape_29 (Reshape)	(None, 1, 384)	0	dropout_28[0][0]
lstm_18 (LSTM)	(None, 100)	194,000	reshape_29[0][0]
dense_32 (Dense)	(None, 3)	303	lstm_18[0][0]

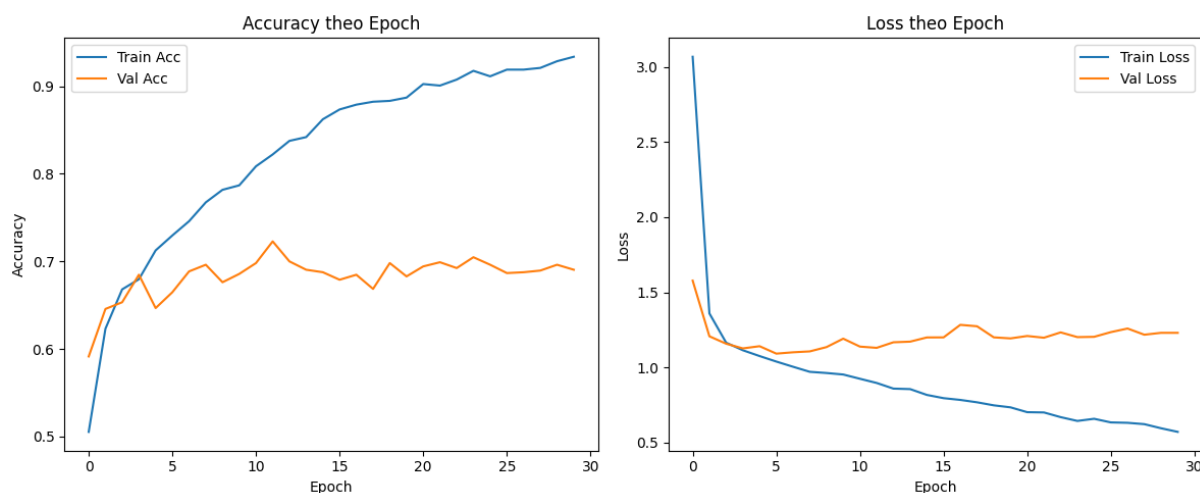
Total params: 3,976,687 (15.17 MB)

Trainable params: 3,976,687 (15.17 MB)

Non-trainable params: 0 (0.00 B)

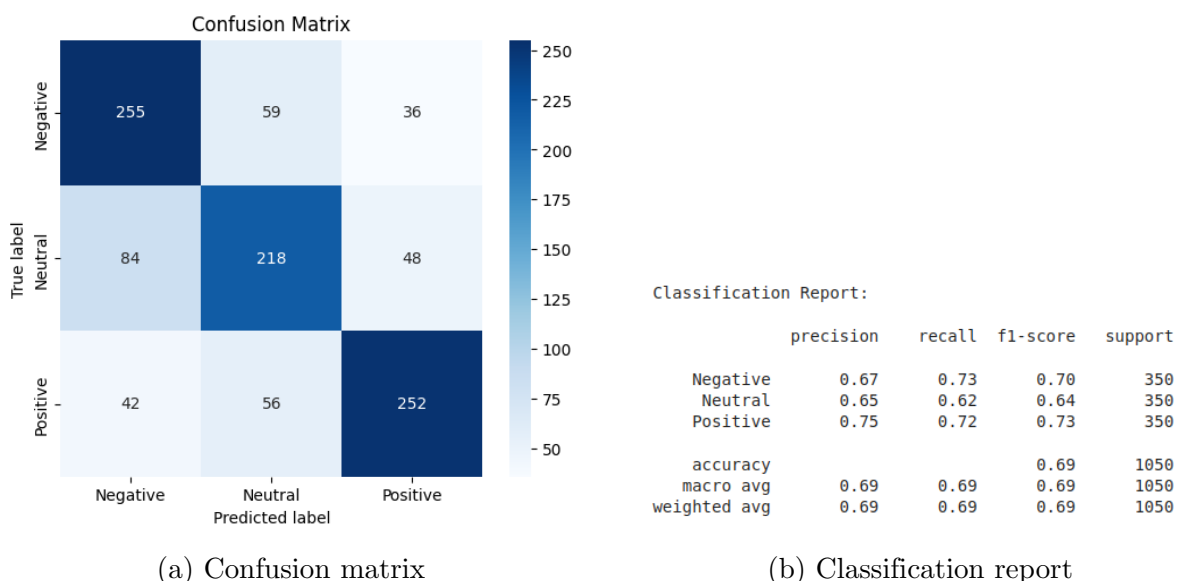
Hình 24: Số lượng tham số của từng lớp

Kết quả huấn luyện trên tập dữ liệu cho trước:



Hình 25: Số lượng tham số của từng lớp

Kết quả sau khi sử dụng model trên tiến hành kiểm thử trên tập test như sau:



Hình 26: Kết quả chạy trên tập test

Một số nhận xét từ kết quả chạy của mô hình:

- Lớp Negative có tỷ lệ phân loại chính xác cao nhất với 255/350 mẫu ($\approx 72.9\%$). Tuy còn bị nhầm sang Neutral (59 mẫu), nhưng vẫn thể hiện độ chính xác tốt trong phát hiện cảm xúc tiêu cực.
- Lớp Positive cũng được mô hình phân loại tương đối chính xác với 252/350 mẫu ($\approx 72\%$), cho thấy mô hình nhận diện tốt các cụm cảm xúc tích cực.
- Lớp Neutral là lớp có nhiều sự nhầm lẫn nhất, đặc biệt bị nhầm sang Negative (84 mẫu), cho thấy mô hình gặp khó khăn trong việc phân biệt cảm xúc trung tính và tiêu cực.

3.4 BERT-based

Sự ra đời của kiến trúc Transformer (Vaswani et al., 2017) đã tạo nên một cuộc cách mạng trong lĩnh vực xử lý ngôn ngữ tự nhiên. Không giống như các kiến trúc tuần tự như RNN hay LSTM, Transformer cho phép xử lý toàn bộ chuỗi dữ liệu cùng lúc thông qua cơ chế self-attention, từ đó nâng cao khả năng nắm bắt mối quan hệ ngữ nghĩa giữa các từ bất kể khoảng cách trong câu.

Dựa trên kiến trúc này, mô hình BERT (Bidirectional Encoder Representations from Transformers) do Google đề xuất vào năm 2018 đã mở rộng khả năng biểu diễn ngữ nghĩa bằng cách huấn luyện hai chiều (bidirectional) và sử dụng kỹ thuật Masked Language Modeling để học ngữ cảnh sâu sắc hơn. BERT nhanh chóng trở thành nền tảng mạnh mẽ cho nhiều bài toán NLP nhờ khả năng hiểu ngữ cảnh linh hoạt và hiệu quả.

Tuy nhiên, do BERT gốc được huấn luyện chủ yếu trên dữ liệu tiếng Anh, khi áp dụng cho tiếng Việt sẽ gặp nhiều hạn chế về ngữ pháp và đặc trưng ngôn ngữ. Nhằm khắc phục vấn đề này, nhóm nghiên cứu tại VinAI Research đã phát triển PhoBERT – một mô hình BERT được tiền huấn luyện hoàn toàn trên tập dữ liệu tiếng Việt với kích thước lớn, kết

hợp cùng phương pháp token hóa Byte-Pair Encoding (BPE) phù hợp hơn với đặc thù tiếng Việt.

Trong phạm vi đề tài này, em sử dụng mô hình PhoBERT-base do VinAI phát triển và tiến hành fine-tune mô hình cho bài toán phân tích cảm xúc tiếng Việt (sentiment analysis), dựa trên tập dữ liệu đã được xử lý sẵn. Quá trình fine-tuning cho phép điều chỉnh các trọng số trong mô hình để phù hợp hơn với đặc trưng của bài toán, từ đó cải thiện độ chính xác trong việc phân loại cảm xúc của văn bản đầu vào.

Các bước thực hiện fine-tune mô hình PhoBERT cho bài toán phân tích cảm xúc:

Để huấn luyện mô hình phân tích cảm xúc tiếng Việt một cách hiệu quả, em lựa chọn sử dụng mô hình PhoBERT-base do VinAI phát triển, với kiến trúc kế thừa từ BERT và được tiền huấn luyện trên tập dữ liệu tiếng Việt quy mô lớn. Quá trình fine-tuning được tiến hành theo các bước cụ thể như sau:

1. Tải và xử lý dữ liệu

Dữ liệu được lấy từ hai tập tin `vlsp_sentiment_train.csv` và `vlsp_sentiment_test.csv`, chứa cặp thông tin Text (nội dung văn bản) và Class (nhãn cảm xúc). Các nhãn ban đầu có giá trị -1, 0, 1, được điều chỉnh thành 0, 1, 2 để phù hợp với yêu cầu đầu vào của mô hình phân loại nhiều lớp. Dữ liệu huấn luyện được chia thành hai phần: 90% cho train và 10% cho validation, đảm bảo có bộ dữ liệu độc lập để đánh giá chất lượng trong quá trình huấn luyện.

2. Tiền xử lý văn bản với PhoBERT Tokenizer

Văn bản tiếng Việt được token hóa bằng AutoTokenizer từ mô hình "vinai/phobert-base", sử dụng kỹ thuật mã hóa BPE (Byte-Pair Encoding). Tokenizer này chuyển văn bản thành `input_ids` và `attention_masks`, là các đầu vào chuẩn hóa mà mô hình PhoBERT yêu cầu.

3. Tạo Dataset và DataLoader

Một lớp Dataset tùy chỉnh được xây dựng để tổ chức dữ liệu theo định dạng cần thiết. Các đối tượng Dataset sau đó được đưa vào DataLoader, cho phép chia dữ liệu thành các batch nhỏ trong lúc huấn luyện, giúp mô hình học hiệu quả hơn và tiết kiệm bộ nhớ.

4. Khởi tạo mô hình PhoBERT cho phân loại

Mô hình AutoModelForSequenceClassification được khởi tạo từ checkpoint "vinai/phobert-base", với số lượng lớp đầu ra (output layer) là 3, tương ứng với ba nhãn cảm xúc. Các trọng số của mô hình sẽ được cập nhật trong suốt quá trình fine-tuning.

5. Cài đặt optimizer, scheduler và hàm mất mát

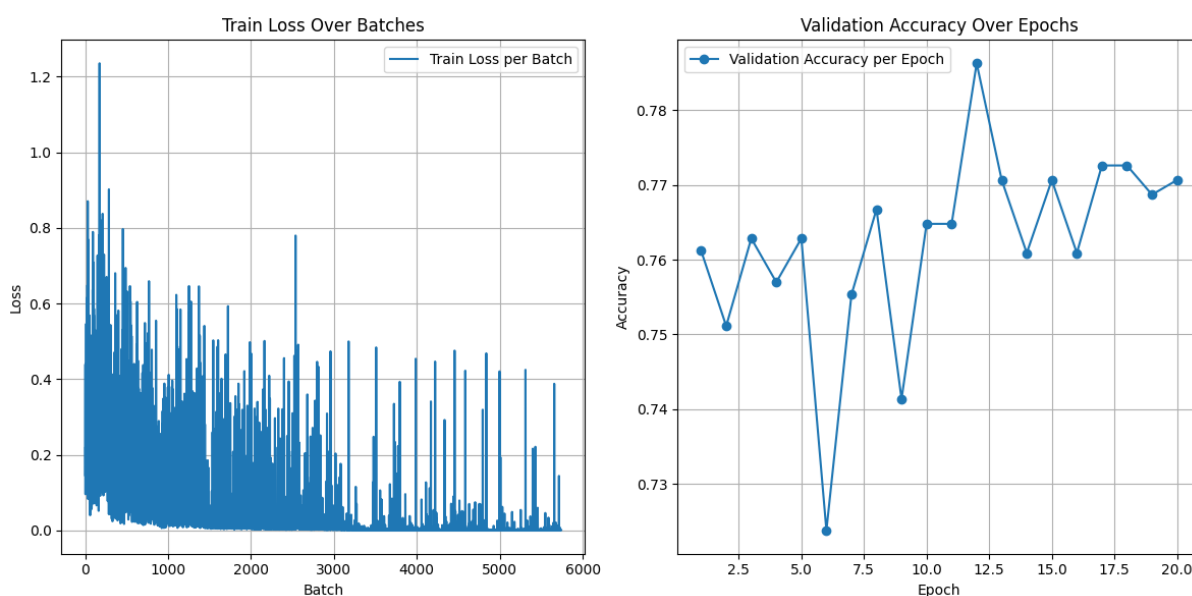
- Thuật toán tối ưu hóa: AdamW với weight decay để tránh overfitting.
- Scheduler học tập: `get_linear_schedule_with_warmup` để điều chỉnh learning rate trong quá trình huấn luyện.
- Hàm mất mát: CrossEntropyLoss, phù hợp cho bài toán phân loại đa lớp.

6. Huấn luyện mô hình (Fine-tuning) Mô hình được huấn luyện liên tục trong 20 epoch. Trong mỗi epoch:

- Mô hình thực hiện forward, backward và tối ưu trọng số trên từng batch.
 - Các chỉ số như train loss, validation loss, và validation accuracy được ghi nhận.
 - Sau mỗi epoch, mô hình được đánh giá trên tập validation để theo dõi quá trình học.
7. Đánh giá mô hình sau huấn luyện Sau khi hoàn tất 20 epoch, mô hình có kết quả validation tốt nhất sẽ được sử dụng để kiểm thử trên tập test. Các chỉ số đánh giá được tính toán bao gồm:
- Độ chính xác (Accuracy)
 - Báo cáo phân loại (Precision, Recall, F1-score)
 - Ma trận nhầm lẫn (Confusion Matrix)
8. Trực quan hóa quá trình huấn luyện

Quá trình huấn luyện được trực quan hóa bằng biểu đồ biểu diễn train loss theo từng batch. Đồ thị này cho thấy xu hướng giảm đều của hàm mất mát trong suốt quá trình học, từ đó minh chứng khả năng hội tụ và hiệu quả fine-tuning của mô hình PhoBERT cho bài toán sentiment analysis tiếng Việt.

Ta thu được kết quả huấn luyện như hình:



Hình 27: Finetune PhoBERT

Hình 27 mô tả quá trình huấn luyện mô hình PhoBERT cho bài toán phân tích cảm xúc tiếng Việt trong vòng 20 epoch, được thể hiện thông qua hai biểu đồ: Train Loss theo từng batch (trái) và Validation Accuracy theo từng epoch (phải). Các biểu đồ này cho phép đánh giá khả năng hội tụ và tổng quát hóa của mô hình trong suốt quá trình huấn luyện.

- Biểu đồ bên trái: Train Loss Over Batches

Biểu đồ thể hiện giá trị loss trên tập huấn luyện qua từng batch trong toàn bộ quá trình huấn luyện. Có thể thấy rằng:

- Ở giai đoạn đầu, loss dao động mạnh, có lúc vượt quá 1.2 – điều này phản ánh mô hình chưa học được nhiều và còn đang điều chỉnh trọng số ban đầu.
- Khi quá trình huấn luyện tiến triển, loss giảm dần theo chiều hướng rõ rệt. Sau khoảng 1000 batch đầu, loss đã giảm về dưới 0.1 và ổn định hơn.
- Dù vẫn có những đỉnh nhọn nhỏ xuất hiện sau khoảng batch 2000–3000, nhưng nhìn chung mô hình đang hội tụ dần với loss tiến về gần 0.
- Xu hướng giảm này cho thấy mô hình đang học hiệu quả và không gặp hiện tượng overfitting trong giai đoạn huấn luyện.

• Biểu đồ bên phải: Validation Accuracy Over Epochs

Biểu đồ thể hiện độ chính xác (accuracy) trên tập validation theo từng epoch:

- Trong 5 epoch đầu, độ chính xác dao động quanh mức 0.75–0.76, cho thấy mô hình đã đạt hiệu quả tạm ổn từ sớm.
- Từ epoch 6 đến 10, độ chính xác có phần biến động mạnh (giảm xuống dưới 0.73 ở epoch 6), cho thấy mô hình có thể gặp khó khăn với các batch khó hoặc do learning rate chưa ổn định.
- Từ epoch 11 trở đi, mô hình có xu hướng ổn định hơn và đạt độ chính xác cao nhất khoảng 0.786 tại epoch 12.
- Các epoch sau giữ được mức độ chính xác cao và ổn định trong khoảng 0.76–0.78, phản ánh mô hình đã bắt đầu hội tụ và tổng quát hóa tốt trên tập validation.

Sau quá trình huấn luyện mô hình PhoBERT trong 20 epoch, mô hình có hiệu suất tốt nhất trên tập validation (dựa trên độ chính xác cao nhất) đã được lưu lại và sử dụng để đánh giá trên tập kiểm thử. Việc đánh giá được thực hiện dựa trên các chỉ số: độ chính xác (accuracy), precision, recall, f1-score, và hai loại trung bình: macro và weighted.

Epoch 20/20, Training Loss: 0.0038, Validation Accuracy: 0.7706					Evaluation on Test Set using Best Model: Test Accuracy: 0.7686				
Validation Report:					Test Report:				
	precision	recall	f1-score	support		precision	recall	f1-score	support
negative	0.82	0.80	0.81	190	negative	0.80	0.74	0.77	350
neutral	0.71	0.70	0.70	158	neutral	0.70	0.71	0.71	350
positive	0.78	0.81	0.79	162	positive	0.81	0.85	0.83	350
accuracy			0.77	510	accuracy			0.77	1050
macro avg	0.77	0.77	0.77	510	macro avg	0.77	0.77	0.77	1050
weighted avg	0.77	0.77	0.77	510	weighted avg	0.77	0.77	0.77	1050

Hình 28: So sánh kết quả huấn luyện và kiểm thử

Hiệu suất trên tập validation:

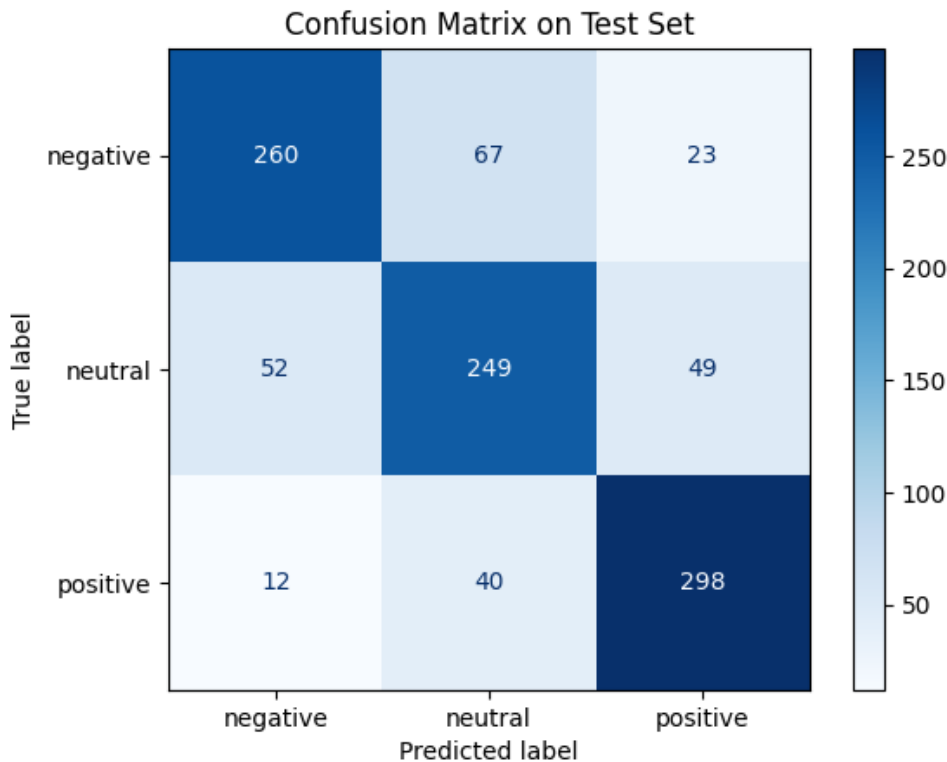
- Độ chính xác (accuracy) trên tập validation đạt 0.7706, cho thấy mô hình học tương đối tốt trên tập huấn luyện.
- Các chỉ số F1-score của từng lớp:
 - negative: 0.81

- neutral: 0.70
- positive: 0.79
- Lớp neutral là lớp có hiệu suất thấp nhất, phản ánh sự khó khăn của mô hình trong việc nhận diện những văn bản không thể hiện rõ cảm xúc.

Hiệu suất trên tập test:

- Độ chính xác trên tập test đạt 0.7686, rất gần với kết quả validation, cho thấy mô hình tổng quát hóa tốt và không bị overfitting.
- Các chỉ số F1-score:
 - negative: 0.77
 - neutral: 0.71
 - positive: 0.83
- Mô hình thể hiện hiệu năng rất tốt với lớp positive và negative, trong khi lớp neutral vẫn là điểm yếu cần cải thiện.

Ngoài ra, ta có thể đánh giá hiệu suất mô hình qua ma trận nhầm lẫn(confusion matrix) thể hiện chi tiết mức độ chính xác và sai lệch của mô hình với từng lớp cảm xúc:



Hình 29: Confusion matrix

Dựa vào hình 29:

- Lớp negative:
 - 260/350 mẫu được phân loại đúng.
 - 67 mẫu nhầm sang neutral và 23 mẫu nhầm sang positive.
 - Mức nhầm lẫn chủ yếu với lớp neutral, cho thấy hai lớp này có thể có ngữ cảnh giao thoa.
- Lớp neutral:
 - 249/350 mẫu được phân loại đúng.
 - Nhầm lẫn gần như chia đều: 52 mẫu bị nhầm sang negative và 49 sang positive.
 - Đây là lớp gây khó khăn nhất cho mô hình do cảm xúc trung lập thường không có đặc trưng ngôn ngữ rõ ràng.
- Lớp positive:
 - 298/350 mẫu được phân loại đúng – cao nhất trong 3 lớp.
 - Chỉ 12 mẫu bị nhầm sang negative và 40 sang neutral.
 - Mô hình có khả năng nhận diện cảm xúc tích cực rất tốt.

Tổng thể, mô hình PhoBERT fine-tuned đã đạt hiệu suất ổn định, chính xác cao với các lớp cảm xúc rõ ràng (negative, positive), và chỉ gặp khó khăn chủ yếu với lớp trung lập (neutral). Đây là kết quả đáng khích lệ, đặc biệt với ngôn ngữ tiếng Việt, và tạo nền tảng vững chắc cho việc phát triển các ứng dụng phân tích cảm xúc trong thực tế.

Bảng 1: Một số ví dụ mô hình PhoBERT dự đoán sai trên tập test

Văn bản	Nhãn thật	Dự đoán
Con miband của e đang chạy ngon lành thì tự nhiên h nó ko pair đc nữa -_- pair lại thì hiện thông báo "...incorrect PIN or passkey", cấm sạc thì chỉ chớp 1 đèn (ko phải hết pin), tắt máy, tắt bluetooth các kiểu rồi mà vẫn vậy -_- có bác nào bị vậy ko ???	neutral	negative
Gia d 5tr hot ljen,hh	neutral	positive
Nhìn cũng được nhưng nhỡ nó rơi cái thì toi	negative	neutral
Mẹ mình từng sang Đài Loan và có mua 1 cái iphone6 plus bị i như vậy đã được đổi máy mới và đang chờ người quen mang từ bên đó về	neutral	negative
hix tự dưng nó bị như thế, con này sống tốt mà em ngồi trong lớp cũng khá là kín, sóng 3G vẫn căng đét, người ta gọi đến với mình gọi đi vẫn nghe gọi bình thường nhưng mỗi nhắn tin là toàn messenger not sent, hiện dấu chấm than, đã reset lại máy vẫn thế	negative	neutral

4 Đánh giá

Từ các phương pháp đã đề xuất, ta tóm tắt kết quả chạy qua bảng sau:

Model	Params	Train Acc	Test Acc	Epochs
CNN(proposed)	4.7M	95.39%	65.62%	48
LSTM(proposed)	3.6M	86.47%	65.33%	18
Hybrid CNN+LSTM(proposed)	3.97M	97.25%	69.04%	30
BERT-based	135M	77.06%	76.86%	20

Bảng 2: So sánh các mô hình phân loại cảm xúc

Nhận xét và so sánh giữa các mô hình:

Từ bảng 2, có thể rút ra một số nhận định quan trọng về hiệu suất của các mô hình phân loại cảm xúc như sau:

- Mô hình CNN và Hybrid CNN+LSTM đạt độ chính xác huấn luyện rất cao (trên 95%), đặc biệt là hybrid đạt tới 97.25%, cho thấy khả năng ghi nhớ rất tốt các đặc trưng trong tập huấn luyện. Tuy nhiên, khoảng cách giữa accuracy trên tập huấn luyện và tập kiểm thử tương đối lớn, cho thấy các mô hình này có thể bị overfitting – học quá sát dữ liệu huấn luyện nhưng chưa tổng quát tốt sang dữ liệu mới.
- Mô hình LSTM thuần túy có số tham số thấp nhất (3.6 triệu) và cho kết quả tương đối ổn định (Train Acc = 86.47%, Test Acc = 65.33%). Tuy nhiên, hiệu suất test của nó vẫn thua đáng kể so với các mô hình dựa trên Transformer.
- Mô hình PhoBERT (pretrained) vượt trội hơn tất cả các mô hình truyền thống cả về độ chính xác và tính ổn định. Mặc dù không được huấn luyện từ đầu, mô hình này đạt được Test Accuracy = 76.86% chỉ sau 20 epoch, cao hơn đáng kể so với các mô hình còn lại. Điều này cho thấy sức mạnh của các mô hình ngôn ngữ lớn đã được tiền huấn luyện trên tập dữ liệu quy mô lớn. PhoBERT còn có ưu điểm lớn trong việc học được ngữ cảnh sâu và mối quan hệ giữa các từ tiếng Việt – điều mà các mô hình CNN/LSTM không thể khai thác hết được.
- Xét về số lượng tham số, PhoBERT có quy mô lớn nhất (135 triệu tham số), lớn gấp hàng chục lần so với các mô hình truyền thống. Điều này đồng nghĩa với chi phí tính toán và bộ nhớ cao hơn, nhưng đổi lại là hiệu năng vượt trội.

5 Kết luận

5.1 Tổng kết

Trong đề tài này, em đã tiến hành nghiên cứu và thực nghiệm bài toán phân tích cảm xúc văn bản tiếng Việt bằng cách áp dụng mô hình ngôn ngữ hiện đại PhoBERT-base. Bên cạnh đó, các mô hình truyền thống như CNN, LSTM, và Hybrid CNN+LSTM cũng được xây dựng và so sánh để đánh giá sự khác biệt về hiệu quả giữa phương pháp học sâu truyền thống và mô hình Transformer tiền huấn luyện.

Quá trình thực hiện gồm nhiều giai đoạn liên tiếp:

- Tiền xử lý dữ liệu, chuẩn hóa nhãn và phân chia tập train-validation-test.
- Chuẩn bị pipeline huấn luyện chuyên biệt cho từng loại mô hình.
- Tiến hành fine-tuning mô hình PhoBERT với chiến lược tối ưu học sâu qua 20 epoch, đồng thời theo dõi hiệu suất trên tập validation để lưu lại mô hình tốt nhất.
- Đánh giá tổng thể trên tập test, phân tích chi tiết các lỗi dự đoán thông qua báo cáo classification và ma trận nhầm lẫn.

Kết quả đạt được cho thấy:

- Mô hình PhoBERT-base sau khi fine-tune đã đạt độ chính xác 76.86% trên tập test, vượt trội so với các mô hình CNN, LSTM và Hybrid CNN+LSTM truyền thống vốn chỉ đạt mức khoảng 65% – 69%.
- Mô hình truyền thống có ưu điểm về kích thước nhẹ và tốc độ huấn luyện nhanh, tuy nhiên khả năng biểu diễn ngữ nghĩa và học phụ thuộc dài hạn còn hạn chế.
- Ngược lại, PhoBERT cho thấy khả năng nắm bắt tốt ngữ cảnh sâu rộng trong câu, đồng thời có sự ổn định cao giữa kết quả tập validation và test, chứng tỏ khả năng tổng quát hóa tốt.

Tuy nhiên, cũng cần lưu ý rằng:

- Việc phân loại văn bản thuộc lớp neutral (trung lập) vẫn còn gặp khó khăn. Điều này xuất phát từ tính chất mơ hồ của lớp neutral trong thực tế, khi ranh giới giữa neutral và các cảm xúc tích cực hoặc tiêu cực là không rõ ràng.
- PhoBERT, mặc dù mạnh mẽ, yêu cầu tài nguyên tính toán lớn hơn đáng kể so với các mô hình nhỏ, đòi hỏi sự cân nhắc khi triển khai thực tế trên các hệ thống có tài nguyên hạn chế.

5.2 Định hướng phát triển trong tương lai

Dựa trên kết quả đạt được, đề tài có thể tiếp tục mở rộng và hoàn thiện theo các hướng sau:

- Tối ưu hóa mô hình: Áp dụng các kỹ thuật giảm tải mô hình như Adapter Layers hoặc Knowledge Distillation, giúp triển khai PhoBERT trên các thiết bị tài nguyên hạn chế (như điện thoại di động hoặc IoT).
- Tăng cường dữ liệu: Thu thập thêm dữ liệu, đặc biệt tập trung vào việc mở rộng tập văn bản neutral, đồng thời áp dụng kỹ thuật data augmentation để làm phong phú tập huấn luyện.
- Kết hợp đa mô hình: Khảo sát phương pháp ensemble (kết hợp nhiều mô hình) hoặc stacking để tận dụng ưu điểm của các mô hình khác nhau, nhằm nâng cao hiệu suất tổng thể.
- Ứng dụng thực tế: Tích hợp mô hình vào các hệ thống chatbot, phân tích phản hồi khách hàng, hoặc các hệ thống hỗ trợ chăm sóc khách hàng tự động, để kiểm nghiệm hiệu quả ứng dụng thực tiễn.

TÀI LIỆU THAM KHẢO

Tài liệu

- [1] Jason Baldridge, Jakob Bauer, Mukul Bhutani, Nicole Brichtova, Andrew Bunner, Kelvin Chan, Yichang Chen, Sander Dieleman, Yuqing Du, Zach Eaton-Rosen, et al. Imagen 3. *arXiv preprint arXiv:2408.07009*, 2024.
- [2] Stanislav Frolov, Tobias Hinz, Federico Raue, Jörn Hees, and Andreas Dengel. Adversarial text-to-image synthesis: A review. *Neural Networks*, 144:187–209, 2021.
- [3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.
- [4] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.
- [5] Calvin Luo. Understanding diffusion models: A unified perspective. *arXiv preprint arXiv:2208.11970*, 2022.
- [6] Ron Mokady, Amir Hertz, and Amit H. Bermano. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.
- [7] Alex Nichol and Prafulla Dhariwal. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.
- [8] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.
- [9] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022.
- [10] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.
- [11] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.
- [12] Chenshuang Zhang, Chaoning Zhang, Mengchun Zhang, and In So Kweon. Text-to-image diffusion models in generative ai: A survey. *arXiv preprint arXiv:2303.07909*, 2023.

- [13] Rui Zhou, Cong Jiang, and Qingyang Xu. A survey on generative adversarial network-based text-to-image synthesis. *Journal of Mechanical, Electrical & Information Engineering*, page Available online 29 April 2021, April 2021. Received 26 November 2020, Revised 13 April 2021, Accepted 16 April 2021.