

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



BÁO CÁO THỰC TẬP
THỰC TẬP CƠ SỞ
TÊN ĐỀ TÀI: PHÂN TÍCH CẢM XÚC BẰNG MÔ HÌNH NGÔN NGỮ
BERT

Giảng viên hướng dẫn	: Trần Đình Quế
Họ và tên sinh viên	: Nguyễn Đức Trường
Mã sinh viên	: B22DCCN883
Lớp	: D22CQC�07

Hà Nội – 2025

MỤC LỤC

CHƯƠNG 1: TRÍ TUỆ NHÂN TẠO VÀ CÁC ỨNG DỤNG	5
1. Tổng quan về trí tuệ nhân tạo	5
1.1. Khái niệm trí tuệ nhân tạo	5
1.2. Phân loại trí tuệ nhân tạo	5
1.3. Thành phần cốt lõi của	7
1.4. Lịch sử phát triển trí tuệ nhân tạo	7
2. Các ứng dụng của trí tuệ nhân tạo	8
2.1. AI trong các trò chơi và lĩnh vực giải trí	8
2.2. Xử lý ngôn ngữ tự nhiên	9
2.3. Thị giác máy tính	10
2.4. Hệ thống tự hành	11
2.5. Hệ chuyên gia	11
2.6. Quản lý doanh nghiệp và dữ liệu	12
3. Xu hướng và tương lai của trí tuệ nhân tạo	13
3.1. Xu hướng hiện tại	13
3.2. Tương lai của AI	14
CHƯƠNG 2: CÁC KỸ THUẬT HỌC SÂU	15
1. Mạng nơ-ron tích chập (Convolutional Neural Networks - CNN)	15
1.1. Khái niệm	15
1.2. Cách hoạt động của CNN	15
1.3. Các loại mạng tích chập	17
1.4. Đặc điểm nổi bật của CNN	19
1.5. Ứng dụng điển hình của CNN	19
1.6. Thách thức, hạn chế	19
2. Mạng nơ-ron hồi quy (Recurrent Neural Networks - RNN)	20
2.1. Khái niệm	20
2.3. Các loại mạng Nơ-ron Hồi Tiếp (Recurrent Neural Networks – RNN)	20
2.4. Đặc điểm nổi bật của RNN	22
2.5. Ứng dụng của RNN	22
2.6. Thách thức, hạn chế	23

3.	Mạng bộ nhớ dài hạn ngắn hạn (LSTMs)	23
3.1.	Khái niệm	23
3.2.	Cách hoạt động của LSTMs.....	23
3.3.	Đặc điểm nổi bật của LSTMs	24
3.4.	Ứng dụng.....	24
3.5.	Thách thức, hạn chế	24
3.6.	Ưu điểm của LSTM so với RNN truyền thống.....	24
4.	Mạng học chuyển giao (Transfer Learning).....	25
4.1.	Khái niệm	25
4.2.	Cách hoạt động.....	25
4.3.	Đặc điểm nổi bật	26
4.4.	Ứng dụng với chuyển giao học tập trong Xử lý Ngôn ngữ Tự nhiên (NLP)..	27
4.5.	Thách thức, hạn chế	27
5.	Mạng Transformer	28
5.1.	Khái niệm	28
5.2.	Cách hoạt động.....	28
5.3.	Đặc điểm nổi bật	28
5.4.	Ứng dụng.....	29
6.	Học bán giám sát và tự giám sát (Semi-Supervised & Self-Supervised Learning)	29
6.1.	Khái niệm	29
6.2.	Cách hoạt động của học bán giám sát và tự giám sát	29
6.3.	Ứng dụng.....	31
6.4.	So sánh học bán giám sát và tự giám sát.....	32
CHƯƠNG 3: PHÂN TÍCH CẢM XÚC BẰNG NGÔN NGỮ MÔ HÌNH BERT		33
1.	Tổng quan	33
1.1.	Mục tiêu chính	34
1.2.	Vai trò của huấn luyện mô hình trong toán hệ thống.....	35
1.3.	Thiết kế kiến trúc mô hình ngôn ngữ BERT.....	36
2.	Thu thập và xử lý dữ liệu.....	41
2.1.	Nguồn dữ liệu.....	41
2.2.	Tiền xử lý dữ liệu	42

3.	Quá trình huấn luyện mô hình	44
3.1.	Cách triển khai	44
3.2.	Xây dựng mô hình.....	44
4.	Đánh giá mô hình	47
4.1.	Plot Accuracy	47
4.2.	Plot loss	48
5.	Kiểm tra độ chính xác.....	49
5.1.	Đánh giá độ chính xác trên tập test	49
5.2.	Dự đoán cảm xúc cho một bộ test cụ thể	51
6.	Kết luận	52

CHƯƠNG 1: TRÍ TUỆ NHÂN TẠO VÀ CÁC ỨNG DỤNG

1. Tổng quan về trí tuệ nhân tạo

1.1. Khái niệm trí tuệ nhân tạo

Trí tuệ nhân tạo (AI) được định nghĩa là lĩnh vực nghiên cứu nhằm xây dựng các hệ thống máy tính có khả năng thực hiện các nhiệm vụ thông minh, như nhận diện hình ảnh, xử lý ngôn ngữ tự nhiên, hoặc ra quyết định. Theo giáo trình "Nhập môn trí tuệ nhân tạo" (Tùng Minh Phương, 2015), AI là "khoa học và kỹ thuật tạo ra các máy móc thông minh, đặc biệt là các chương trình máy tính thông minh" (trích từ John McCarthy, 1956). Tuy nhiên, do khái niệm "thông minh" khó xác định chính xác, AI thường được phân loại theo các đặc điểm hành vi và khả năng tư duy.

Russell và Norvig (2010) chia các định nghĩa AI thành bốn nhóm chính:

1. **Hành động như người:** Hệ thống AI mô phỏng hành vi của con người, ví dụ như chatbot trả lời câu hỏi tự nhiên hoặc robot phục vụ trong nhà hàng.
2. **Suy nghĩ như người:** Hệ thống tái tạo quá trình tư duy của con người, chẳng hạn như hệ thống GPS (General Problem Solver) của Newell và Simon (1961), được thiết kế để giải quyết vấn đề theo cách tương tự con người.
3. **Suy nghĩ hợp lý:** Hệ thống sử dụng logic để đưa ra kết luận, như các thuật toán tìm kiếm A* trong điều hướng robot.
4. **Hành động hợp lý:** Hệ thống tối ưu hóa hành vi để đạt được mục tiêu cụ thể, chẳng hạn như xe tự lái điều hướng an toàn trên đường.

Trong thực tiễn, cách tiếp cận "hành động hợp lý" được sử dụng phổ biến nhất vì nó tập trung vào hiệu quả và kết quả, thay vì cố gắng bắt chước hoàn toàn tư duy con người. Điều này đặc biệt quan trọng trong các ứng dụng như quản lý kho, nơi AI cần đưa ra quyết định tối ưu về tồn kho mà không cần suy nghĩ giống con người.

1.2. Phân loại trí tuệ nhân tạo

Dựa trên khả năng và phạm vi ứng dụng, AI được chia thành ba cấp độ chính:

1. **AI hẹp (Narrow AI):** Chỉ thực hiện một nhiệm vụ cụ thể. Ví dụ, hệ thống gợi ý sản phẩm của Amazon, trợ lý ảo Siri, hoặc các công cụ dự đoán nhu cầu trong quản lý kho dựa trên dữ liệu hóa đơn. Đây là loại AI phổ biến nhất hiện nay.
2. **AI tổng quát (General AI):** Có khả năng thực hiện bất kỳ nhiệm vụ trí tuệ nào mà con người có thể làm. Tuy nhiên, General AI vẫn đang trong giai đoạn nghiên cứu và chưa đạt được trong thực tế.
3. **Siêu trí tuệ (Super AI):** Vượt xa trí tuệ con người về mọi mặt. Đây là một khái niệm lý thuyết, gây tranh cãi về tính khả thi và các vấn đề đạo đức liên quan.

So sánh các loại trí tuệ nhân tạo:

Tiêu chí	Narrow AI (AI hẹp)	General AI (AI tổng quát)	Super AI (Siêu trí tuệ)
Khả năng	Chỉ thực hiện nhiệm vụ cụ thể, không linh hoạt ngoài phạm vi lập trình.	Thực hiện bất kỳ nhiệm vụ trí tuệ nào như con người, linh hoạt và tự thích ứng.	Vượt qua trí tuệ con người, tự cải thiện và sáng tạo vượt giới hạn.
Phạm vi ứng dụng	Ứng dụng trong các lĩnh vực cụ thể như nhận diện hình ảnh, chatbot, dự đoán dữ liệu.	Áp dụng cho mọi lĩnh vực, từ khoa học, nghệ thuật đến quản lý.	Giải quyết vấn đề toàn cầu, vượt khả năng con người.
Mức độ phức tạp	Đơn giản, dễ triển khai, tối ưu cho từng nhiệm vụ.	Rất phức tạp, đòi hỏi tích hợp nhiều lĩnh vực như học máy, logic, tâm lý học.	Cực kỳ phức tạp, cần công nghệ mới chưa tồn tại.
Trạng thái hiện tại	Phổ biến, được ứng dụng rộng rãi (Amazon, Tesla, Siri).	Đang nghiên cứu, chưa đạt được (GPT-4 tiến gần nhưng vẫn là Narrow AI).	Chỉ là lý thuyết, chưa có hệ thống thực tế.
Ví dụ	- Chatbot ngân hàng - Gọi ý Netflix - Xe tự lái Tesla	Chưa có ví dụ thực tế, chỉ là mục tiêu nghiên cứu (DeepMind, GPT-4).	Khái niệm khoa học viễn tưởng (HAL 9000, Skynet).
Ưu điểm	Hiệu quả cao, chi phí thấp, dễ triển khai.	Linh hoạt, tiềm năng đột phá khoa học.	Giải quyết vấn đề vượt khả năng con người.
Nhược điểm	Giới hạn phạm vi, phụ thuộc dữ liệu.	Chưa khả thi, rủi ro đạo đức.	Nguy cơ mất kiểm soát, đe dọa nhân loại.

1.3. Thành phần cốt lõi của

AI bao gồm nhiều lĩnh vực chuyên sâu, trong đó các thành phần chính bao gồm:

- **Học máy (Machine Learning):** Cho phép hệ thống học từ dữ liệu mà không cần lập trình chi tiết. Ví dụ, thuật toán k-NN được sử dụng để phân loại email rác.
- **Học sâu (Deep Learning):** Sử dụng mạng nơ-ron sâu để xử lý dữ liệu phức tạp, như nhận diện hình ảnh trong y tế hoặc phân tích ngôn ngữ tự nhiên.
- **Xử lý ngôn ngữ tự nhiên (NLP):** Giúp máy hiểu và tạo ra ngôn ngữ con người, như chatbot hỗ trợ khách hàng hoặc dịch tự động.
- **Thị giác máy tính (Computer Vision):** Cho phép máy phân tích và hiểu hình ảnh hoặc video, như nhận diện khuôn mặt trong an ninh.

Các thành phần này kết hợp với nhau để tạo nên các hệ thống AI mạnh mẽ, có khả năng giải quyết nhiều vấn đề thực tiễn trong các lĩnh vực khác nhau.

1.4. Lịch sử phát triển trí tuệ nhân tạo

Lịch sử của AI trải qua nhiều giai đoạn với các mốc thời gian quan trọng:

- **Thập niên 1950:** Alan Turing đề xuất bài kiểm tra Turing để đánh giá trí thông minh của máy (1950). John McCarthy tổ chức hội nghị Dartmouth (1956), đặt tên cho lĩnh vực "trí tuệ nhân tạo".
- **Thập niên 1980:** Các hệ chuyên gia (Expert Systems) như MYCIN được phát triển, hỗ trợ chẩn đoán y khoa. Tuy nhiên, giai đoạn này gặp khó khăn do hạn chế về phần cứng và dữ liệu.
- **Thập niên 1990-2000:** Học máy bắt đầu nổi lên với các thuật toán như Support Vector Machine (SVM) và Random Forest. Năm 1997, Deep Blue của IBM đánh bại nhà vô địch cờ vua Garry Kasparov, đánh dấu bước tiến lớn trong AI.
- **Thập niên 2010-nay:** Sự bùng nổ của học sâu nhờ dữ liệu lớn, phần cứng mạnh (GPU, TPU), và các mô hình như CNN, RNN, Transformer. Năm 2012, AlexNet chiến thắng cuộc thi ImageNet, mở ra kỷ nguyên của học sâu. Các hệ thống như AlphaGo (2016) và GPT-3 (2020) hay GPT-4 (2024) tiếp tục khẳng định tiềm năng của AI.

Sự phát triển của AI được thúc đẩy bởi ba yếu tố chính:

- Dữ liệu lớn (Big Data): Các cơ sở dữ liệu phân tán như Google Spanner cung cấp lượng dữ liệu khổng lồ để huấn luyện mô hình AI.
- Phần cứng mạnh mẽ: GPU của NVIDIA và TPU của Google tăng tốc độ xử lý các mô hình phức tạp.
- Tiến bộ thuật toán: Các thuật toán như Gradient Descent và Adam cải thiện hiệu quả học sâu.

2. Các ứng dụng của trí tuệ nhân tạo

Các ứng dụng của trí tuệ nhân tạo (AI) rất đa dạng, từ giải trí, y tế, giao thông, đến quản lý doanh nghiệp. AI đã và đang được ứng dụng rộng rãi trong nhiều lĩnh vực, mang lại lợi ích to lớn nhưng cũng đi kèm các thách thức về kỹ thuật, đạo đức và cả xã hội.

2.1. AI trong các trò chơi và lĩnh vực giải trí

AI đã đạt được những bước tiến vượt bậc trong lĩnh vực trò chơi, đặc biệt là các trò chơi chiến lược và trí tuệ như cờ vua và cờ Go. Một trong những cột mốc quan trọng là chiến thắng của Deep Blue (IBM) trước nhà vô địch cờ vua Garry Kasparov vào năm 1997. Deep Blue sử dụng thuật toán tìm kiếm và đánh giá hàng triệu nước đi mỗi giây, thể hiện khả năng xử lý nhanh và chính xác của AI.

Năm 2016, AlphaGo của DeepMind đánh bại Lee Sedol trong cờ Go, một trò chơi có số lượng nước đi lớn hơn nhiều so với cờ vua. AlphaGo kết hợp mạng nơ-ron sâu và học tăng cường để đưa ra các chiến lược sáng tạo, mở ra kỷ nguyên mới cho AI trong các nhiệm vụ phức tạp.

Ngoài ra, AI còn được sử dụng trong trò chơi điện tử để tạo ra nhân vật không phải người chơi (NPC) thông minh, sử dụng các kỹ thuật như cây hành vi và học máy để thích ứng với hành vi người chơi, nâng cao trải nghiệm giải trí.

Lợi ích:

- Tăng tính thách thức và hấp dẫn của trò chơi, thu hút người chơi ở nhiều cấp độ kỹ năng.
- Cải thiện trải nghiệm người dùng thông qua các NPC có hành vi giống con người hơn.
- Hỗ trợ phát triển các công cụ sáng tạo nội dung, như tạo cốt truyện hoặc thiết kế cấp độ trò chơi tự động.
- Góp phần thử nghiệm và phát triển các thuật toán AI tiên tiến, áp dụng ngược vào các lĩnh vực khác như y tế và giao thông.

Thách thức:

- Yêu cầu tài nguyên tính toán lớn, đặc biệt với các trò chơi phức tạp như cờ Go, đòi hỏi phần cứng chuyên dụng (GPU, TPU).
- Thiết kế thuật toán tối ưu để cân bằng giữa hiệu suất và tính thực tế trong thời gian thực.
- Khó duy trì sự cân bằng trong trò chơi khi AI trở nên quá mạnh, làm giảm tính thú vị cho người chơi.
- Chi phí phát triển cao, hạn chế khả năng tiếp cận của các nhà phát triển nhỏ.

2.2. Xử lý ngôn ngữ tự nhiên

Xử lý ngôn ngữ tự nhiên (NLP) cho phép máy hiểu, phân tích và tạo ra ngôn ngữ con người, mở ra nhiều ứng dụng trong giao tiếp và xử lý thông tin. Các ứng dụng nổi bật bao gồm:

- **Nhận dạng tiếng nói:** Các trợ lý ảo như Siri, Google Assistant và Alexa chuyển đổi giọng nói thành văn bản, hỗ trợ người dùng thực hiện tìm kiếm, điều khiển thiết bị thông minh, hoặc quản lý lịch trình. Công nghệ này sử dụng mô hình học sâu như LSTM để xử lý chuỗi âm thanh.
- **Dịch tự động:** Google Translate sử dụng mô hình dịch máy nơ-ron để dịch văn bản giữa hơn 100 ngôn ngữ, đạt độ trôi chảy cao trong các cặp ngôn ngữ phổ biến như Anh-Việt.
- **Hệ thống hỏi đáp:** IBM Watson phân tích khối lượng dữ liệu lớn (như Wikipedia) để trả lời các câu hỏi phức tạp, được ứng dụng trong hỗ trợ khách hàng, nghiên cứu khoa học, và giáo dục.
- **Tóm tắt văn bản:** AI tự động tóm tắt tài liệu dài thành các đoạn ngắn, hỗ trợ báo chí, nghiên cứu, và quản lý thông tin.

Lợi ích:

- Cải thiện khả năng truy cập thông tin cho người dùng trên toàn cầu, đặc biệt với dịch tự động và nhận dạng tiếng nói.
- Tăng hiệu quả giao tiếp giữa con người và máy, giảm thời gian xử lý các tác vụ như trả lời khách hàng hoặc tìm kiếm thông tin.
- Hỗ trợ người khuyết tật (như người khiếm thính hoặc khiếm thị) thông qua chuyển đổi giọng nói-văn bản và ngược lại.
- Tự động hóa các quy trình xử lý văn bản, tiết kiệm nhân lực trong các lĩnh vực như pháp lý và báo chí.
- Thúc đẩy giao tiếp đa ngôn ngữ, hỗ trợ hợp tác quốc tế và hội nhập văn hóa.

Thách thức:

- Xử lý ngữ nghĩa phức tạp và sắc thái văn hóa, đặc biệt với các ngôn ngữ ít tài nguyên như tiếng Việt.
- Đảm bảo độ chính xác trong môi trường ồn ào hoặc với giọng nói không chuẩn (như khẩu âm vùng miền).
- Nguy cơ thiên vị trong mô hình ngôn ngữ, dẫn đến các phản hồi không phù hợp hoặc thiếu công bằng.
- Bảo mật dữ liệu người dùng, đặc biệt với các trợ lý ảo lưu trữ thông tin cá nhân.
- Chi phí huấn luyện mô hình ngôn ngữ lớn, đòi hỏi dữ liệu đa dạng và tài nguyên tính toán mạnh.

2.3. Thị giác máy tính

Thị giác máy tính cho phép máy phân tích và hiểu nội dung hình ảnh hoặc video, mang lại các ứng dụng thực tiễn trong nhiều lĩnh vực. Các ví dụ tiêu biểu bao gồm:

- **Nhận dạng đối tượng:** AI nhận diện khuôn mặt, dấu vân tay, hoặc móng mắt, được sử dụng trong an ninh (camera giám sát tại sân bay), xác thực (mở khóa điện thoại), và mạng xã hội (gắn thẻ ảnh trên Facebook).
- **Phân tích hình ảnh y tế:** CheXNet, phát triển bởi Đại học Stanford, sử dụng mạng nơ-ron tích chập (CNN) để phát hiện viêm phổi từ ảnh X-quang, đạt độ chính xác vượt qua bác sĩ chuyên khoa trong một số trường hợp.
- **Nhận dạng chữ in (OCR):** AI nhận dạng văn bản in với độ chính xác 99% theo tiêu chuẩn ISO 1073-1:1976, hỗ trợ số hóa tài liệu và tự động hóa nhập liệu trong ngân hàng, thư viện.

Lợi ích:

- Tăng độ chính xác và tốc độ trong phân tích hình ảnh, vượt qua khả năng của con người trong một số trường hợp.
- Tự động hóa các quy trình như giám sát an ninh, kiểm tra chất lượng sản phẩm, và số hóa tài liệu, tiết kiệm thời gian và chi phí.
- Hỗ trợ chẩn đoán y tế nhanh chóng, đặc biệt ở các khu vực thiếu bác sĩ chuyên môn.
- Cải thiện trải nghiệm người dùng trong các ứng dụng như tìm kiếm hình ảnh hoặc gắn thẻ tự động.
- Tăng cường an toàn thông qua nhận diện đối tượng trong thời gian thực (như phát hiện tội phạm hoặc tai nạn).

Thách thức:

- Hạn chế trong nhận dạng chữ viết tay hoặc đối tượng trong môi trường phức tạp (như ánh sáng yếu, góc nhìn bất lợi).
- Nguy cơ thiên vị trong mô hình nhận diện khuôn mặt, dẫn đến phân biệt đối xử (ví dụ: nhận diện sai người thuộc một số nhóm dân tộc).
- Yêu cầu dữ liệu huấn luyện đa dạng và chất lượng cao để đạt hiệu suất tốt.
- Vấn đề quyền riêng tư, đặc biệt với nhận diện khuôn mặt trong giám sát công cộng.
- Chi phí triển khai cao, bao gồm phần cứng (camera, GPU) và phần mềm.

2.4. Hệ thống tự hành

AI là nền tảng của các hệ thống tự hành, đặc biệt trong lĩnh vực giao thông. Xe tự lái của Waymo và Tesla sử dụng thị giác máy tính, học sâu, và cảm biến (LIDAR, radar) để nhận diện vật thể, lập kế hoạch đường đi, và tránh va chạm. Các thuật toán như CNN và Reinforcement Learning giúp xe xử lý các tình huống giao thông phức tạp. Xe tự lái đạt hiệu suất tương đương con người trong điều kiện giao thông tiêu chuẩn tại Mỹ, nhưng vẫn cần cải thiện ở các môi trường không chuẩn như giao thông Việt Nam với mật độ xe máy cao.

Lợi ích:

- Giảm tai nạn giao thông do lỗi con người, vốn chiếm hơn 90% vụ tai nạn theo thống kê của NHTSA (2020).
- Tăng hiệu quả vận chuyển thông qua tối ưu hóa lộ trình và giảm thời gian chờ.
- Cải thiện trải nghiệm người dùng, đặc biệt cho người khuyết tật hoặc người cao tuổi không thể tự lái xe.
- Hỗ trợ phát triển các dịch vụ vận tải tự động, như taxi robot hoặc giao hàng không người lái.
- Giảm lượng khí thải nhờ tối ưu hóa lái xe và quản lý giao thông đô thị.

Thách thức:

- Yêu cầu cơ sở hạ tầng hiện đại: bản đồ số hóa chính xác và mạng 5G ổn định.
- Giải quyết các vấn đề pháp lý, như trách nhiệm trong trường hợp tai nạn liên quan đến xe tự lái.
- Đảm bảo an toàn trong các điều kiện bất lợi (mưa lớn, sương mù, hoặc đường phố đông đúc).
- Chi phí phát triển và triển khai cao, hạn chế khả năng tiếp cận ở các quốc gia đang phát triển.
- Nguy cơ tấn công mạng, làm gián đoạn hoặc điều khiển sai hệ thống tự hành.

2.5. Hệ chuyên gia

Hệ chuyên gia mô phỏng khả năng ra quyết định của chuyên gia con người, sử dụng cơ sở tri thức và quy tắc suy luận. Các ví dụ nổi bật bao gồm:

- **MYCIN**: Hỗ trợ chẩn đoán nhiễm trùng máu với độ chính xác tương đương bác sĩ, sử dụng 450 quy tắc từ chuyên gia y tế.
- **DENDRAL**: Dự đoán cấu trúc phân tử hữu cơ dựa trên dữ liệu hóa học, hỗ trợ nghiên cứu khoa học.
- **XCON**: Tự động cấu hình máy tính cho công ty DEC, giảm chi phí và thời gian sản xuất.

Lợi ích:

- Hỗ trợ ra quyết định nhanh chóng và chính xác trong các lĩnh vực chuyên môn như y tế, hóa học, và kỹ thuật.
- Giảm phụ thuộc vào chuyên gia con người, đặc biệt ở các khu vực thiếu nhân lực trình độ cao.
- Lưu trữ và tái sử dụng kiến thức chuyên môn, đảm bảo tính nhất quán trong ra quyết định.
- Tăng hiệu quả vận hành trong các quy trình phức tạp, như cấu hình hệ thống hoặc chẩn đoán bệnh.
- Hỗ trợ đào tạo nhân lực mới thông qua việc cung cấp các gợi ý và phân tích chuyên sâu.

Thách thức:

- Phụ thuộc vào chất lượng và độ đầy đủ của cơ sở tri thức, đòi hỏi sự hợp tác chặt chẽ với chuyên gia.
- Khó mở rộng sang các lĩnh vực mới do hạn chế trong việc xây dựng quy tắc cho các vấn đề phức tạp.
- Thiếu khả năng học tập từ dữ liệu mới, khiến hệ thống dễ lỗi thời so với các phương pháp học máy hiện đại.
- Chi phí xây dựng và duy trì cao, đặc biệt khi cần cập nhật cơ sở tri thức thường xuyên.
- Nguy cơ đưa ra quyết định sai nếu dữ liệu đầu vào không chính xác hoặc thiếu thông tin.

2.6. Quản lý doanh nghiệp và dữ liệu

AI cải thiện hiệu quả trong quản lý doanh nghiệp thông qua phân tích dữ liệu và tự động hóa quy trình, đặc biệt trong các lĩnh vực liên quan đến dữ liệu lớn. Các ứng dụng tiêu biểu bao gồm:

- **Quản lý kho:** AI phân tích dữ liệu từ bảng HoaDon để dự đoán nhu cầu sản phẩm, tối ưu hóa tồn kho, và giảm lãng phí. Ví dụ, một cửa hàng văn phòng phẩm sử dụng mô hình học máy để dự báo số lượng bút bi cần nhập dựa trên lịch sử bán hàng, đảm bảo cung ứng kịp thời và giảm chi phí lưu kho.
- **Quản lý công việc:** Các công cụ như Trello tích hợp AI để gợi ý ưu tiên công việc, dự đoán thời gian hoàn thành, và tự động phân loại nhiệm vụ. Ví dụ, AI có thể đề xuất phân công nhiệm vụ dựa trên hiệu suất trước đây của nhân viên, nâng cao hiệu quả quản lý dự án.

- **Phân tích hành vi khách hàng:** AI dự đoán sở thích khách hàng thông qua phân tích dữ liệu giao dịch, hỗ trợ chiến lược tiếp thị cá nhân hóa. Ví dụ, Amazon sử dụng hệ thống gợi ý để đề xuất sản phẩm, tăng tỷ lệ mua hàng.
- **Tối ưu hóa chuỗi cung ứng:** AI dự đoán gián đoạn, quản lý tồn kho đa địa điểm, và chọn tuyến vận chuyển hiệu quả, giảm chi phí và thời gian giao hàng.

3. Xu hướng và tương lai của trí tuệ nhân tạo

3.1. Xu hướng hiện tại

AI đang chứng kiến một số xu hướng quan trọng, phản ánh sự tiến bộ trong công nghệ và nhu cầu thực tiễn của xã hội:

- **Mô hình ngôn ngữ lớn (Large Language Models):** Các mô hình như GPT-4 (OpenAI) và LLaMA (Meta AI) đã cải thiện đáng kể khả năng xử lý ngôn ngữ tự nhiên, từ dịch tự động, tạo nội dung, đến hỗ trợ ra quyết định. Ví dụ, trong quản lý công việc, các công cụ như Trello có thể tích hợp AI để tự động tạo mô tả nhiệm vụ hoặc gợi ý ưu tiên công việc dựa trên phân tích ngôn ngữ. Những mô hình này đòi hỏi tài nguyên tính toán lớn nhưng mang lại hiệu quả cao trong giao tiếp và tự động hóa.
- **AI kết hợp với IoT và 5G:** Sự kết hợp giữa AI, Internet vạn vật (IoT), và mạng 5G cho phép xử lý dữ liệu thời gian thực trong các ứng dụng như nhà thông minh, thành phố thông minh, và chuỗi cung ứng. Trong quản lý kho, AI có thể phân tích dữ liệu từ cảm biến IoT để dự đoán nhu cầu sản phẩm (như bút bi trong bảng HoaDonVe), tối ưu hóa tồn kho và giảm chi phí vận chuyển, nhờ tốc độ truyền dữ liệu của 5G.
- **AI đạo đức và minh bạch (Explainable AI):** Với mối lo ngại về thiên vị và thiếu minh bạch trong các mô hình AI, nghiên cứu về Explainable AI (XAI) đang được thúc đẩy. XAI giúp giải thích cách AI đưa ra quyết định, tăng độ tin cậy trong các lĩnh vực nhạy cảm như y tế và tài chính. Ví dụ, trong chẩn đoán y tế, XAI có thể giải thích lý do một mô hình như CheXNet xác định dấu hiệu viêm phổi từ ảnh X-quang, hỗ trợ bác sĩ ra quyết định chính xác hơn.
- **Tự động hóa thông minh (Intelligent Automation):** AI kết hợp với tự động hóa quy trình robot (RPA) để tối ưu hóa các quy trình kinh doanh. Trong quản lý doanh nghiệp, AI có thể tự động hóa việc phân tích dữ liệu từ cơ sở dữ liệu phân tán (như Google Spanner), phát hiện xu hướng và đề xuất chiến lược, giảm thời gian xử lý và sai sót.

- **AI trong dự báo công nghệ:** Như bạn từng hỏi về dự báo công nghệ, AI được sử dụng để phân tích xu hướng công nghệ (như blockchain, cơ sở dữ liệu phân tán) thông qua dữ liệu từ các nguồn như bài báo khoa học và báo cáo thị trường. Điều này giúp các tổ chức chuẩn bị cho các đổi mới, chẳng hạn tích hợp AI vào quản lý chuỗi cung ứng hoặc bảo mật dữ liệu.

3.2. Tương lai của AI

Tương lai của AI hứa hẹn những bước tiến vượt bậc, nhưng cũng đặt ra các thách thức về kỹ thuật, đạo đức, và xã hội. Dưới đây là các triển vọng chính:

- **Hướng tới General AI:** General AI, có khả năng thực hiện bất kỳ nhiệm vụ trí tuệ nào như con người, là mục tiêu dài hạn của các nhà nghiên cứu. Mặc dù công nghệ hiện tại chưa đạt được, các mô hình như GPT-4 cho thấy tiềm năng tiến gần hơn. General AI có thể thay đổi cách con người làm việc, ví dụ tự động hóa toàn bộ quy trình quản lý kho hoặc thiết kế chiến lược kinh doanh mà không cần can thiệp của con người.
- **AI trong phát triển bền vững:** AI được kỳ vọng đóng vai trò quan trọng trong giải quyết các vấn đề toàn cầu như biến đổi khí hậu, quản lý tài nguyên, và chăm sóc y tế. Ví dụ, AI có thể tối ưu hóa chuỗi cung ứng để giảm lượng khí thải hoặc dự đoán dịch bệnh dựa trên dữ liệu y tế toàn cầu.
- **Tác động đến thị trường lao động:** AI sẽ tự động hóa nhiều công việc thủ công và lặp lại, như nhập liệu hoặc quản lý tồn kho, nhưng đồng thời tạo ra các cơ hội mới trong các lĩnh vực như phát triển AI, phân tích dữ liệu, và quản lý công nghệ. Điều này đòi hỏi đào tạo lại lực lượng lao động để thích ứng với các yêu cầu kỹ năng mới.
- **AI có trách nhiệm:** Các tổ chức như xAI đang tập trung vào phát triển AI minh bạch, công bằng, và an toàn. Trong tương lai, các quy định quốc tế về AI có thể được thiết lập để ngăn chặn lạm dụng, như sử dụng AI trong giám sát không kiểm soát hoặc tạo nội dung giả mạo (deepfake).
- **Tích hợp AI với công nghệ mới:** AI sẽ kết hợp với các công nghệ như blockchain, thực tế ảo (VR), và điện toán lượng tử để tạo ra các ứng dụng đột phá. Ví dụ, blockchain có thể đảm bảo bảo mật dữ liệu trong các hệ thống AI quản lý kho, trong khi điện toán lượng tử tăng tốc độ huấn luyện mô hình học sâu.

CHƯƠNG 2: CÁC KỸ THUẬT HỌC SÂU

Có nhiều loại kỹ thuật học sâu được sử dụng với nhiều cách khác nhau như xử lý dữ liệu trong không gian, hay xử lý dữ liệu theo cách tuần tự (thường dùng với văn bản hoặc chuỗi thời gian). Trong phạm vi nghiên cứu bài tập lớn này, tôi đề cập và nghiên cứu chính với 5 loại kỹ thuật học sâu là Mạng Nơ-ron Tích Chập (CNN), Mạng Nơ-ron Hồi Quy (RNN), Mạng bộ nhớ dài ngắn hạn (LSTMs), Học Chuyển Giao, Mạng Transformer, và Học Bán Giám Sát & Tự Giám Sát.

1. Mạng nơ-ron tích chập (Convolutional Neural Networks - CNN)

1.1. Khái niệm

Mạng Nơ-ron Tích Chập (Convolutional Neural Networks - CNN) là một lớp mạng nơ-ron nhân tạo được thiết kế chuyên biệt để xử lý dữ liệu có cấu trúc lưới, chẳng hạn như hình ảnh (dữ liệu 2D) hoặc chuỗi thời gian (dữ liệu 1D). CNN lấy cảm hứng từ cách hệ thống thị giác của con người hoạt động, đặc biệt là khả năng nhận diện các mẫu cục bộ trong hình ảnh.

CNN sử dụng các phép tích chập (convolution) để trích xuất đặc trưng từ dữ liệu, giúp giảm số lượng tham số và tăng hiệu quả tính toán so với mạng nơ-ron truyền thống (Fully Connected Neural Networks).

1.2. Cách hoạt động của CNN

CNN sử dụng các lớp bộ lọc, được gọi là tích chập, để quét và xử lý dữ liệu đầu vào thành các phần nhỏ hơn (như các phần của hình ảnh). Sau đó, việc gộp các lớp sẽ giảm chiều, cho phép mô hình tập trung vào các tính năng chính.

Một mạng CNN điển hình bao gồm ba loại lớp chính:

- Lớp tích chập (Convolutional Layer)
- Lớp gộp (Pooling Layer)
- Lớp kết nối hoàn toàn (Fully Connected Layer - FC)

Ngoài ra, còn có một số kỹ thuật hỗ trợ như **Dropout** và **Batch Normalization** giúp cải thiện hiệu quả huấn luyện và giảm hiện tượng quá khớp.

a) Lớp tích chập (Convolutional Layer)

Đây là lớp quan trọng nhất trong mạng CNN, nơi mô hình học được các đặc trưng của dữ liệu đầu vào như cạnh, góc, họa tiết, màu sắc, hình dạng,...

❖ Cơ chế hoạt động

- **Bộ lọc (Kernel/Filter)** là một ma trận trọng số nhỏ (thường là 3x3 hoặc 5x5) di chuyển (quét) trên ảnh đầu vào và thực hiện phép tích chập: nhân từng phần tử tương ứng và cộng lại, tạo ra một giá trị duy nhất trong đầu ra gọi là **feature map (bản đồ đặc trưng)**.

- **Trọng số của bộ lọc được học trong quá trình huấn luyện** qua lan truyền ngược (backpropagation).
- **Nhiều bộ lọc khác nhau sẽ học được những đặc trưng khác nhau** của ảnh (ví dụ: bộ lọc này học cạnh ngang, bộ khác học cạnh dọc...).

❖ **Tham số chính**

- **Số lượng bộ lọc:** Mỗi bộ lọc cho ra một feature map. Số lượng bộ lọc càng nhiều thì mạng học càng nhiều đặc trưng khác nhau.
- **Stride (bước nhảy):** Là số bước di chuyển của bộ lọc sau mỗi phép tích chập. Stride lớn → đầu ra nhỏ hơn.
- **Padding:** Để giữ kích thước đầu ra bằng với đầu vào, ta có thể "đệm" thêm các giá trị 0 xung quanh ảnh (zero-padding). Giúp bảo toàn thông tin ở biên ảnh.

❖ **Hàm kích hoạt:** Sau mỗi phép tích chập, đầu ra được đưa qua hàm phi tuyến **ReLU (Rectified Linear Unit):**

$$f(x) = \max(0, x)$$

Điều này giúp mô hình có khả năng học các đặc trưng phi tuyến phức tạp hơn và giảm hiện tượng vanishing gradient.

b) Lớp gộp (Pooling Layer)

Lớp gộp giúp giảm chiều dữ liệu, giảm số lượng tham số và tăng khả năng khái quát của mô hình.

❖ **Cơ chế hoạt động:**

- Bộ lọc không có trọng số, chỉ thực hiện phép toán tổng hợp trên các vùng nhỏ của feature map.
- Kết quả là một phiên bản "nén" của ảnh, nhưng vẫn giữ lại các đặc trưng quan trọng.

❖ **Các loại pooling phổ biến:**

- **Max pooling:** Lấy giá trị lớn nhất trong vùng quét.
- **Average pooling:** Lấy giá trị trung bình trong vùng quét.

Ví dụ: Max pooling 2x2 với stride = 2 sẽ giảm kích thước đầu vào 4 lần.

Lớp gộp giúp mô hình bền vững hơn với các biến đổi nhỏ trong hình ảnh như dịch chuyển, xoay hoặc thay đổi góc nhìn.

c) Lớp kết nối hoàn toàn (Fully Connected Layer - FC)

Là lớp cuối cùng trong CNN, nơi mà các đặc trưng trích xuất từ các lớp trước được “dàn phẳng” và đưa vào mô hình phân loại.

Đặc điểm:

- Tương tự như mạng nơ-ron truyền thống (MLP), mỗi nơ-ron kết nối với toàn bộ đầu vào.
- Thường sử dụng hàm kích hoạt **ReLU** ở các lớp FC trung gian, và **Softmax** ở lớp đầu ra

d) Các Kỹ Thuật Bổ Trợ

- **Dropout**

- Ngẫu nhiên “tắt” một phần nơ-ron trong quá trình huấn luyện với một xác suất (ví dụ 0.5).
- Mục tiêu: giảm hiện tượng **quá khớp (overfitting)** bằng cách làm mô hình không phụ thuộc quá nhiều vào một số nơ-ron cụ thể

- **Batch Normalization**

- Chuẩn hóa đầu ra của từng lớp theo từng batch.
- Lợi ích:
 - Giúp quá trình huấn luyện nhanh hơn.
 - Giảm sự phụ thuộc vào việc chọn khởi tạo trọng số.
 - Tăng độ ổn định và khả năng hội tụ của mạng.

1.3. Các loại mạng tích chập

Từ khi được giới thiệu vào những năm 1980 bởi **Kunihiko Fukushima** với mô hình *Neocognitron*, và sau đó được phát triển mạnh mẽ hơn bởi **Yann LeCun** trong những năm 1990 với *LeNet-5*, mạng tích chập (CNN) đã trải qua nhiều giai đoạn cải tiến đột phá. Những mô hình CNN hiện đại đã giúp giải quyết thành công nhiều bài toán thị giác máy tính phức tạp, đặc biệt là trong các cuộc thi như ImageNet Large Scale Visual Recognition Challenge (ILSVRC).

a) LeNet-5 (1998)

LeNet-5 là một trong những kiến trúc CNN đầu tiên được phát triển bởi Yann LeCun vào năm 1998, chủ yếu phục vụ cho việc nhận dạng chữ viết tay trên tài liệu, như mã bưu chính. Mạng này gồm các lớp tích chập và gộp đan xen, kết thúc bằng một vài lớp kết nối đầy đủ. Dù kiến trúc còn đơn giản, LeNet-5 đặt nền móng cho sự phát triển của các mạng CNN hiện đại.

Đặc điểm chính:

- Gồm 7 lớp (không tính đầu vào), xen kẽ giữa convolution và pooling.
- Sử dụng tanh làm hàm kích hoạt.
- Áp dụng trong nhận dạng chữ viết tay (MNIST).

b) AlexNet (2012)

AlexNet là một trong những mạng CNN nổi tiếng đầu tiên mang lại bước đột phá lớn trong nhận diện hình ảnh. Được phát triển bởi Alex Krizhevsky, Ilya Sutskever, và Geoffrey Hinton vào năm 2012, AlexNet đã giành chiến thắng trong cuộc thi ImageNet ILSVRC.

Đặc điểm chính:

- Gồm 8 lớp học (5 lớp tích chập và 3 lớp fully connected).
- Dùng hàm kích hoạt ReLU thay cho sigmoid.
- Áp dụng Dropout để giảm overfitting.
- Huấn luyện bằng GPU để tăng tốc đáng kể.

c) ZFNet (2013)

ZFNet, được phát triển bởi Matthew Zeiler và Rob Fergus vào năm 2013, là một cải tiến trực tiếp từ AlexNet. Nhóm nghiên cứu đã sử dụng phương pháp trực quan hóa hoạt động bên trong mạng để điều chỉnh kích thước kernel, stride và các thông số khác, từ đó cải thiện độ chính xác trong nhận diện ảnh.

Đặc điểm chính:

- Điều chỉnh kích thước bộ lọc và stride so với AlexNet.
- Trực quan hóa feature map giúp hiểu rõ mạng học gì.
- Đạt thành tích cao hơn AlexNet trên ImageNet.

d) VGGNet (2014)

Được phát triển bởi nhóm Oxford VGG (Visual Geometry Group) vào năm 2014, VGGNet nổi bật với kiến trúc đơn giản, chỉ sử dụng các bộ lọc 3x3 và bước nhảy nhỏ. Sự đồng nhất trong thiết kế giúp mạng dễ dàng được mở rộng về chiều sâu, với các phiên bản như VGG-16 hay VGG-19.

Đặc điểm chính:

Đặc điểm chính:

- Tất cả các bộ lọc đều có kích thước 3x3, stride = 1.
- Dễ mở rộng, phù hợp cho việc tái sử dụng trong các tác vụ khác.
- Đạt hiệu năng cao nhưng yêu cầu tài nguyên lớn.

e) GoogLeNet (Inception) (2014)

Ra mắt năm 2014 bởi Google, GoogLeNet gây ấn tượng với kiến trúc Inception module – một thiết kế cho phép áp dụng nhiều bộ lọc kích thước khác nhau trong cùng một lớp. Điều này giúp mạng học được đặc trưng ở nhiều mức độ mà vẫn giữ được số lượng tham số ở mức thấp.

Đặc điểm chính:

- Sử dụng Inception module kết hợp các filter 1x1, 3x3, 5x5.
- Mạng rất sâu (22 lớp) nhưng ít tham số nhờ 1x1 convolution.
- Dùng trung bình toàn cục (global average pooling) thay cho fully connected cuối cùng.

1.4. Đặc điểm nổi bật của CNN

- **Xử lý hiệu quả dữ liệu không gian:** CNN đặc biệt phù hợp với dữ liệu có cấu trúc không gian như hình ảnh, nhờ khả năng khai thác mối quan hệ cục bộ giữa các điểm ảnh.
- **Chia sẻ trọng số:** Các bộ lọc dùng chung trọng số trên toàn bộ ảnh, giúp giảm đáng kể số lượng tham số và tránh quá khớp.
- **Tự động học đặc trưng:** Không cần thiết kế thủ công đặc trưng, mạng học được các mẫu và đặc điểm quan trọng trực tiếp từ dữ liệu.
- **Hạn chế:** Cần lượng lớn dữ liệu huấn luyện và đòi hỏi tài nguyên tính toán cao (CPU/GPU) để đạt hiệu quả tốt.

1.5. Ứng dụng điển hình của CNN

- **Phân loại hình ảnh:** CNN trong học sâu đặc biệt thành thạo trong phân loại hình ảnh, bao gồm phân loại hình ảnh thành các nhóm đã thiết lập. Họ thành thạo khả năng giải mã hình ảnh cho thấy mèo, chó, ô tô và hoa, điều này chứng tỏ là rất quan trọng khi xử lý các hệ thống thông tin hình ảnh lớn để tổ chức và gắn thẻ.
- **Phát hiện đối tượng (Object Detection):** Thông qua phát hiện vật thể, CNN có khả năng nhận dạng trong khi định vị các mục xác định trong hình ảnh. Phát hiện mục tiêu thông qua CNN cho phép người dùng xác định phương tiện và hình người trong khi xác định vị trí không gian của chúng để triển khai cần xác định mốc vật thể chính xác.
- **Nhận diện khuôn mặt:** Xác định và xác minh danh tính qua hình ảnh khuôn mặt.
- **Chẩn đoán y tế:** Phân tích hình ảnh y khoa như X-quang, MRI để hỗ trợ phát hiện bệnh.

1.6. Thách thức, hạn chế

- **Phụ thuộc vào dữ liệu được gắn nhãn:** CNN cần lượng lớn dữ liệu huấn luyện có nhãn để đạt hiệu quả cao, điều này có thể tốn kém và mất thời gian để thu thập.
- **Chi phí tính toán cao:** Việc xử lý các ảnh có độ phân giải cao hoặc mạng sâu đòi hỏi phần cứng mạnh (GPU/TPU), dẫn đến chi phí huấn luyện và triển khai lớn.

2. Mạng nơ-ron hồi quy (Recurrent Neural Networks - RNN)

2.1. Khái niệm

Mạng Nơ-ron Hồi Quy (Recurrent Neural Networks - RNN) là một loại mạng nơ-ron nhân tạo được thiết kế để xử lý dữ liệu tuần tự, như văn bản, chuỗi thời gian, hoặc âm thanh. RNN nổi bật với khả năng duy trì "bộ nhớ" về các thông tin trước đó thông qua trạng thái ẩn (hidden state), phù hợp cho các bài toán có phụ thuộc thời gian hoặc ngữ cảnh.

2.2. Cách hoạt động của RNN

Mạng nơ-ron hồi quy (Recurrent Neural Network – RNN) hoạt động dựa trên ý tưởng cốt lõi là sử dụng trạng thái ẩn để ghi nhớ thông tin theo thời gian. Trong quá trình xử lý, dữ liệu được đưa qua một lớp mạng duy nhất, nhưng lớp này được lặp đi lặp lại tại mỗi bước thời gian, tạo thành một chu kỳ xử lý.

Tại mỗi thời điểm, trạng thái ẩn của RNN được cập nhật dựa trên đầu vào hiện tại và trạng thái ẩn từ bước trước. Điều này cho phép mạng tích lũy thông tin từ quá khứ và duy trì mối liên hệ giữa các phần tử trong chuỗi dữ liệu. Trạng thái ẩn đóng vai trò như một bộ nhớ tạm, giúp mạng lưu giữ và truyền tải thông tin qua nhiều bước thời gian.

Các tham số như trọng số và độ lệch (bias) được chia sẻ trong toàn bộ quá trình lặp, giúp mô hình duy trì tính nhất quán và giảm số lượng tham số cần huấn luyện. Nhờ vào cơ chế này, RNN có thể phát hiện và học được các mẫu phụ thuộc theo thời gian trong dữ liệu tuần tự như văn bản, âm thanh, hoặc chuỗi thời gian.

Tuy nhiên, do cấu trúc tuần hoàn và lan truyền trạng thái, RNN thường gặp khó khăn khi phải ghi nhớ thông tin trong chuỗi rất dài, vì hiện tượng mất mát hoặc bùng nổ gradient trong quá trình huấn luyện.

2.3. Các loại mạng Nơ-ron Hồi Tiếp (Recurrent Neural Networks – RNN)

Mạng nơ-ron hồi tiếp (RNN) có khả năng xử lý dữ liệu theo chuỗi, nhờ việc duy trì trạng thái ẩn (hidden state) qua từng bước thời gian. Điều này giúp RNN đặc biệt phù hợp với các bài toán tuần tự như dịch máy, nhận dạng giọng nói, phân tích cảm xúc, dự đoán chuỗi thời gian, và nhiều ứng dụng khác trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP) hay dữ liệu âm thanh.

a) Standard RNN (RNN tiêu chuẩn)

Đây là dạng cơ bản nhất của mạng hồi tiếp, trong đó đầu ra tại mỗi bước thời gian được tính dựa trên đầu vào hiện tại và trạng thái ẩn trước đó. Cấu trúc đơn giản của RNN tiêu chuẩn giúp dễ triển khai và trực quan trong cách vận hành. Tuy nhiên, chính sự đơn giản đó cũng là điểm yếu khi mạng phải học các quan hệ phụ thuộc dài hạn – vấn đề vanishing gradient thường khiến việc huấn luyện trở nên kém hiệu quả.

- Phù hợp với chuỗi ngắn, ít quan hệ dài hạn.

- Dễ gặp vấn đề mất mát gradient khi chuỗi quá dài.
- Thường dùng cho các tác vụ đơn giản như dự đoán từ tiếp theo trong câu ngắn.

b) Bidirectional RNN (BRNN)

Bidirectional RNN cải tiến mô hình tiêu chuẩn bằng cách cho phép mạng học thông tin từ cả hai chiều của chuỗi dữ liệu: từ đầu đến cuối và ngược lại. Nhờ đó, mỗi điểm trong chuỗi có thể được dự đoán dựa trên cả quá khứ và tương lai. Điều này đặc biệt hữu ích trong các tác vụ xử lý ngôn ngữ tự nhiên, nơi ngữ nghĩa của một từ thường phụ thuộc vào ngữ cảnh toàn cục.

- Ánh xạ thông tin theo cả hai chiều thời gian.
- Giúp tăng độ chính xác trong phân tích ngữ cảnh.
- Không phù hợp với các ứng dụng thời gian thực do cần toàn bộ chuỗi đầu vào.

c) LSTM (Long Short-Term Memory)

LSTM được phát triển để khắc phục điểm yếu lớn nhất của RNN tiêu chuẩn – hiện tượng vanishing gradient. Kiến trúc LSTM giới thiệu bộ nhớ tế bào (cell state) cùng với ba cổng điều khiển (forget, input, output), cho phép mô hình ghi nhớ hoặc quên thông tin một cách linh hoạt qua thời gian. Nhờ vậy, LSTM có thể học được các quan hệ dài hạn trong chuỗi. Gồm 3 cổng: Forget gate, Input gate, Output gate. LSTM có khả năng học các phụ thuộc dài hạn nhưng yêu cầu nhiều tài nguyên do số lượng tham số lớn.

d) GRU (Gated Recurrent Unit)

GRU là một biến thể đơn giản hơn của LSTM, được thiết kế để tối ưu tốc độ huấn luyện và giảm số lượng tham số. Không giống như LSTM, GRU không có bộ nhớ riêng mà sử dụng trực tiếp hidden state để lưu thông tin. Mặc dù đơn giản hơn, GRU vẫn cho hiệu năng tốt trong nhiều bài toán, đặc biệt là các ứng dụng thời gian thực. GRU chỉ có 2 cổng: Reset gate và Update gate với tốc độ huấn luyện nhanh hơn, ít tham số hơn LSTM nhưng GRU không linh hoạt bằng LSTM khi xử lý các phụ thuộc rất dài.

e) Encoder-Decoder RNN

Encoder-Decoder là một kiến trúc thường gặp trong các bài toán sequence-to-sequence như dịch máy, tóm tắt văn bản, hoặc tạo chú thích ảnh. Mô hình bao gồm hai phần: Encoder sẽ "nén" chuỗi đầu vào thành một vector ngữ cảnh (context vector), sau đó Decoder sử dụng vector này để tạo ra chuỗi đầu ra. Dù hiệu quả trong các bài toán ngắn, mô hình này có thể gặp khó khăn khi xử lý chuỗi đầu vào dài vì toàn bộ thông tin phải được lưu trữ trong một vector cố định.

- Áp dụng trong dịch máy, tóm tắt văn bản, tạo chuỗi mới.
- Dễ bị nghẽn thông tin nếu chuỗi đầu vào quá dài.
- Là nền tảng cho các mô hình tiên tiến hơn như Attention và Transformer

2.4. Đặc điểm nổi bật của RNN

RNN đặc biệt phù hợp với các loại dữ liệu có tính tuần tự như văn bản, giọng nói, hoặc chuỗi thời gian. Mạng có khả năng **ghi nhớ thông tin từ các bước trước** nhờ vào trạng thái ẩn, cho phép nắm bắt mối liên hệ theo thứ tự thời gian. Để cải thiện hiệu suất khi xử lý các chuỗi dài, các biến thể như **LSTM (Long Short-Term Memory)** và **GRU (Gated Recurrent Unit)** đã được phát triển, giúp mạng học được các phụ thuộc dài hạn hiệu quả hơn và giảm thiểu hiện tượng mất mát gradient. Tuy nhiên, RNN vẫn tồn tại hạn chế về **hiệu suất tính toán**, đặc biệt khi xử lý chuỗi rất dài. So với các kiến trúc hiện đại như **Transformer**, RNN thường chậm hơn và gặp khó khăn trong việc mô hình hóa các phụ thuộc xa trong chuỗi dữ liệu.

2.5. Ứng dụng của RNN

Trong **xử lý ngôn ngữ tự nhiên (NLP)**, RNN đã đóng vai trò quan trọng trong việc cải thiện hiệu suất của nhiều tác vụ như mô hình hóa ngôn ngữ, dịch máy và phân tích cảm xúc. Với khả năng ghi nhớ ngữ cảnh từ các từ đã xuất hiện trước đó trong chuỗi, RNN giúp hệ thống đưa ra dự đoán chính xác hơn về từ tiếp theo. Đặc biệt, các biến thể như LSTM và GRU đã chứng minh hiệu quả vượt trội trong việc nắm bắt các phụ thuộc dài hạn, góp phần nâng cao độ chính xác trong dịch máy và phân loại cảm xúc.

Trong **dự báo chuỗi thời gian**, RNN là một công cụ mạnh mẽ để xử lý và phân tích các dữ liệu có thứ tự thời gian, chẳng hạn như giá cổ phiếu, nhiệt độ, hay nhu cầu thị trường. Bằng cách học từ các mẫu trong dữ liệu quá khứ, RNN có khả năng dự đoán các giá trị tương lai, làm cho nó trở nên hữu ích trong nhiều ứng dụng kinh tế, tài chính và khí tượng.

Trong lĩnh vực **nhận dạng giọng nói**, RNN đã nâng cao đáng kể hiệu quả của việc chuyển đổi âm thanh thành văn bản. Nhờ khả năng xử lý dữ liệu tuần tự, RNN có thể theo dõi các mối quan hệ theo thời gian trong tín hiệu giọng nói, từ đó cải thiện độ chính xác trong việc nhận dạng và phiên âm lời nói. Điều này đã thúc đẩy sự phát triển của các ứng dụng như trợ lý ảo, dịch vụ nhập liệu bằng giọng nói và các hệ thống giao tiếp tự động.

2.6. Thách thức, hạn chế

Mặc dù RNN có khả năng xử lý dữ liệu tuần tự hiệu quả, nhưng quá trình huấn luyện mạng thường **diễn ra chậm**, đặc biệt khi làm việc với các chuỗi dài và phức tạp. Một trong những vấn đề phổ biến nhất là **hiện tượng biến mất gradient (vanishing gradient)**, khiến mô hình khó học được các mối quan hệ dài hạn trong chuỗi dữ liệu. Điều này gây ra khó khăn trong việc điều chỉnh trọng số qua các bước thời gian xa, làm giảm khả năng ghi nhớ và dự đoán của mạng. Dù các kiến trúc như LSTM và GRU đã được phát triển để giải quyết phần nào vấn đề này, RNN truyền thống vẫn gặp nhiều hạn chế về mặt hiệu suất và khả năng mở rộng.

3. Mạng bộ nhớ dài hạn ngắn hạn (LSTMs)

3.1. Khái niệm

LSTMs (Long Short-Term Memory Networks) là một loại Mạng nơ-ron hồi quy (RNN) được thiết kế để giải quyết vấn đề phụ thuộc lâu dài, cho phép mạng ghi nhớ và sử dụng thông tin từ giai đoạn trước đó trong chuỗi đầu vào.

Nhờ vào cấu trúc gồm các cổng điều khiển (cổng quên, cổng đầu vào và cổng đầu ra), LSTM có khả năng lưu trữ và duy trì thông tin từ các bước trước đó trong chuỗi đầu vào trong khoảng thời gian dài, từ đó cải thiện hiệu suất trong các tác vụ như dịch máy, nhận dạng giọng nói và phân tích chuỗi thời gian.

3.2. Cách hoạt động của LSTMs

LSTM sử dụng **trạng thái ô nhớ (cell state)** và các **cổng** để kiểm soát luồng thông tin qua chuỗi thời gian:

- **Trạng thái ô nhớ:**

Là một "dòng chảy bộ nhớ" chạy xuyên suốt toàn bộ chuỗi, cho phép lưu giữ hoặc điều chỉnh thông tin trong thời gian dài. Nhờ đó, LSTM có thể ghi nhớ các mối quan hệ dài hạn tốt hơn so với RNN thông thường.

- **Các cổng:** Có 3 loại cổng chính, giúp điều chỉnh thông tin nên giữ lại, cập nhật hay loại bỏ:

- **Cổng quên (Forget Gate):** Quyết định phần thông tin nào từ trạng thái trước đó nên bị loại bỏ.
- **Cổng đầu vào (Input Gate):** Xác định thông tin nào từ đầu vào hiện tại sẽ được thêm vào trạng thái ô nhớ.
- **Cổng đầu ra (Output Gate):** Chọn lọc phần thông tin nào từ ô nhớ sẽ được đưa ra làm đầu ra (hidden state) cho bước hiện tại.

Cơ chế này giúp LSTM khắc phục được vấn đề mất dần gradient và xử lý tốt hơn các chuỗi có mối quan hệ phụ thuộc dài.

3.3. Đặc điểm nổi bật của LSTMs

- **Phù hợp với dữ liệu tuần tự:** LSTM được thiết kế lý tưởng để xử lý các chuỗi có thứ tự như chuỗi thời gian, văn bản, âm thanh hoặc dữ liệu cảm biến.
- **Ghi nhớ ngữ cảnh dài hạn:** Nhờ cơ chế cổng và trạng thái ô nhớ, LSTM có thể giữ lại thông tin quan trọng từ các bước trước đó trong chuỗi, giúp mô hình hiểu và tận dụng được ngữ cảnh dài.
- **Giảm thiểu vấn đề gradient biến mất:** So với RNN truyền thống, LSTM khắc phục hiệu quả tình trạng mất mát hoặc suy giảm gradient khi huấn luyện trên chuỗi dài, từ đó giúp mô hình học ổn định hơn.

3.4. Ứng dụng

- **Nhận dạng giọng nói:** LSTM giúp xử lý âm thanh theo thời gian để nhận diện từ ngữ hoặc câu nói, nhờ khả năng ghi nhớ ngữ cảnh và thông tin liên tiếp.
- **Dự báo chuỗi thời gian:** Được sử dụng trong các bài toán như dự đoán giá cổ phiếu, thời tiết hoặc nhu cầu năng lượng, nơi dữ liệu có tính chuỗi và phụ thuộc thời gian.
- **Mô hình hóa ngôn ngữ:** LSTM có thể dự đoán từ tiếp theo trong câu, tạo văn bản hoặc hỗ trợ trong các hệ thống như tự động hoàn thành câu (ví dụ: gợi ý văn bản trong trình nhắn tin).

3.5. Thách thức, hạn chế

Yêu cầu tính toán cao: Việc huấn luyện LSTM đòi hỏi nhiều tài nguyên tính toán, đặc biệt khi xử lý chuỗi dài, như trong bài toán dịch máy với hàng ngàn câu cần phải xử lý.

Cần dữ liệu lớn: Để đạt được kết quả tối ưu, LSTM cần một lượng dữ liệu huấn luyện lớn. Ví dụ, trong nhận dạng giọng nói, nếu không có đủ dữ liệu âm thanh với các nhãn chính xác, mô hình sẽ không thể học tốt và có thể dẫn đến hiệu suất kém.

3.6. Ưu điểm của LSTM so với RNN truyền thống

LSTM (Long Short-Term Memory) được thiết kế để khắc phục các hạn chế cố hữu của RNN truyền thống, đặc biệt là trong việc xử lý các chuỗi dữ liệu dài. Dưới đây là những ưu điểm nổi bật của LSTM:

- **Giải quyết vấn đề vanishing gradient:** Nhờ cấu trúc cổng và bộ nhớ tế bào, LSTM duy trì được tín hiệu gradient ổn định trong quá trình lan truyền ngược, giúp mô hình học được các phụ thuộc dài hạn mà RNN thường không xử lý được.
- **Cơ chế bộ nhớ tế bào thông minh:** Bộ nhớ tế bào (cell state) cho phép thông tin quan trọng được lưu trữ qua nhiều bước thời gian, trong khi các cổng (forget gate, input gate, output gate) điều khiển lượng thông tin được thêm, giữ lại hoặc loại bỏ.

- **Khả năng học phụ thuộc dài hạn tốt:** LSTM có thể ghi nhớ thông tin trong các chuỗi dài, rất phù hợp với các bài toán như dịch máy, phân tích ngữ cảnh, hoặc dự đoán chuỗi thời gian.
- **Ổn định trong huấn luyện:** LSTM thường hội tụ nhanh và ít bị mắc kẹt trong các cực tiểu địa phương (local minima) hơn so với RNN thường.
- **Ứng dụng rộng rãi và hiệu quả cao:** Nhờ khả năng xử lý dữ liệu tuần tự mạnh mẽ, LSTM là nền tảng trong nhiều hệ thống học sâu hiện đại cho các bài toán liên quan đến chuỗi, như nhận diện giọng nói, phân tích văn bản, và tạo nhạc.

4. Mạng học chuyển giao (Transfer Learning)

4.1. Khái niệm

Transfer Learning là quá trình ứng dụng kiến thức từ một nhiệm vụ đã học để giải quyết một nhiệm vụ mới có liên quan. Con người làm điều này một cách tự nhiên — ví dụ, kỹ năng giữ thăng bằng khi đi xe đạp có thể giúp học cách đi xe máy. Trong học máy, thay vì huấn luyện một mô hình từ đầu cho mỗi bài toán, ta có thể sử dụng lại một phần của mô hình đã được huấn luyện trước trên tập dữ liệu lớn, như ImageNet, và tinh chỉnh nó cho nhiệm vụ mới.

Kỹ thuật này đặc biệt hiệu quả khi dữ liệu cho bài toán mới bị hạn chế. Các mô hình như LSTM, BERT hoặc ResNet có thể được dùng lại để tiết kiệm thời gian huấn luyện, cải thiện hiệu suất và giảm nguy cơ quá khớp. Theo Andrew Ng, học chuyển giao sẽ là động lực quan trọng cho sự phát triển của học máy trong tương lai, đặc biệt trong bối cảnh dữ liệu không gán nhãn ngày càng nhiều.

4.2. Cách hoạt động

Trong học chuyển giao, có hai cách tiếp cận phổ biến:

1. **Huấn luyện từ đầu và tái sử dụng mô hình:** Một mô hình được huấn luyện kỹ lưỡng cho nhiệm vụ A có thể được lưu lại và sử dụng làm nền tảng cho nhiệm vụ B. Kiến trúc mô hình, trọng số và các đặc trưng đã học từ nhiệm vụ trước có thể giúp rút ngắn thời gian huấn luyện cho nhiệm vụ mới.
2. **Sử dụng mô hình được huấn luyện sẵn (pre-trained):** Đây là phương pháp phổ biến hiện nay, trong đó các mô hình như VGG, ResNet, hoặc BERT – vốn đã được huấn luyện trên tập dữ liệu lớn (như ImageNet) – được tải xuống và tinh chỉnh cho bài toán cụ thể. Phương pháp này tiết kiệm đáng kể thời gian, tài nguyên và giảm yêu cầu về dữ liệu.

❖ Triển khai học chuyển giao với VGG16

VGG16 là một mạng nơ-ron tích chập nổi tiếng gồm 16 lớp trọng số (13 lớp tích chập và 3 lớp fully-connected), được huấn luyện trên ImageNet với 138 triệu tham số.

Các bước triển khai:

1. **Nhập mô hình VGG16 từ thư viện Keras** (thuộc TensorFlow).
2. **Tải trọng số pre-trained từ ImageNet** và gán vào mô hình.
3. **Đóng băng tất cả các lớp** để không huấn luyện lại chúng.
4. **Thêm các lớp phân loại mới** ở phần đỉnh mô hình (ví dụ: lớp Flatten + Dense + Softmax cho bài toán nhị phân).
 - Flatten() để chuyển đầu ra từ VGG16 thành vector 1 chiều.
 - Một lớp Dense(256, activation='relu') để học các đặc trưng mới.
 - Dropout(0.5) để giảm overfitting.
 - Dense(1, activation='sigmoid') nếu là phân loại nhị phân, hoặc Dense(n_classes, activation='softmax') nếu phân loại đa lớp.
5. **In mô hình tóm tắt** để kiểm tra cấu trúc: chỉ có các lớp mới được thêm là có thể huấn luyện (~50.000 tham số), còn lại (~14,71 triệu) là cố định.

Sau đó, bạn có thể biên dịch mô hình, chọn optimizer, hàm mất mát phù hợp và huấn luyện với hàm *fit()* như thông thường.

❖ Một số lưu ý khi triển khai:

- Dữ liệu đầu vào phải được resize về kích thước phù hợp với VGG16: **224x224 RGB**.
- Phải chuẩn hóa ảnh theo cách mà VGG16 đã được huấn luyện (dùng `preprocess_input()` từ `keras.applications.vgg16`).
- Nếu dữ liệu ít, nên dùng **data augmentation** để tránh overfitting.

4.3. Đặc điểm nổi bật

- Tiết kiệm thời gian và tài nguyên: Sử dụng mô hình đã huấn luyện sẵn giúp giảm đáng kể thời gian và chi phí huấn luyện.
- Yêu cầu ít dữ liệu gán nhãn: Học chuyển giao cho phép huấn luyện hiệu quả ngay cả với lượng dữ liệu ít.
- Hiệu quả với dữ liệu hạn chế: Đặc biệt hữu ích trong các bài toán có ít dữ liệu hoặc dữ liệu khó gán nhãn.

4.4. Ứng dụng với chuyển giao học tập trong Xử lý Ngôn ngữ Tự nhiên (NLP)

Học chuyển giao đã làm thay đổi cách tiếp cận các tác vụ trong NLP bằng cách sử dụng các mô hình ngôn ngữ đã được đào tạo trước như **BERT**, **GPT**, và **RoBERTa**. Những mô hình này được huấn luyện trên lượng lớn dữ liệu văn bản, giúp nắm bắt thông tin ngữ nghĩa và ngữ cảnh, sau đó có thể tinh chỉnh cho các tác vụ cụ thể như phân tích tình cảm, nhận dạng thực thể, dịch máy, và tạo văn bản.

Các Khía Cạnh Chính của Học Chuyển Giao trong NLP:

- **Mô hình ngôn ngữ đã được đào tạo trước:** Các mô hình như BERT, GPT, RoBERTa được huấn luyện trên lượng dữ liệu lớn và có thể được tinh chỉnh cho các tác vụ cụ thể, giảm thiểu sự cần thiết về dữ liệu gán nhãn.
- **Kiến trúc học chuyển giao:** Quy trình bao gồm hai bước chính: đào tạo trước với dữ liệu không nhãn để học các biểu diễn ngôn ngữ chung, sau đó tinh chỉnh trên dữ liệu có nhãn để điều chỉnh cho các tác vụ cụ thể.
- **Ứng dụng:** Học chuyển giao trong NLP đã được áp dụng thành công trong các lĩnh vực như phân tích tình cảm, phân loại văn bản, trả lời câu hỏi, và dịch máy, giúp cải thiện hiệu suất và giảm yêu cầu về dữ liệu.
- **Hướng đi tương lai:** Nghiên cứu tiếp tục cải thiện mô hình, đào tạo, và ứng dụng học chuyển giao trong các ngôn ngữ có ít tài nguyên. Tập trung vào các thách thức như thích ứng miền và cải thiện khả năng tạo ngôn ngữ tự nhiên.

4.5. Thách thức, hạn chế

Học chuyển giao mang lại nhiều lợi ích, nhưng cũng gặp phải một số thách thức quan trọng cần được giải quyết để triển khai thành công. Các thách thức chính bao gồm:

- **Chuyển miền:** Chuyển giao học tập giả định rằng miền nguồn và miền đích có sự tương quan, nhưng thực tế có thể có sự khác biệt lớn giữa chúng. Điều này có thể ảnh hưởng đến hiệu quả của việc chuyển giao kiến thức. Cần chú ý đến sự phân phối dữ liệu và cách biểu diễn tính năng giữa các miền.
- **Lựa chọn nhiệm vụ:** Việc chọn nhiệm vụ nguồn và mục tiêu phù hợp rất quan trọng. Các nhiệm vụ có sự tương đồng về tính năng và mục tiêu sẽ giúp chuyển giao học tập hiệu quả hơn. Nếu nhiệm vụ quá khác biệt, chuyển giao sẽ khó thành công.
- **Chuyển giao tiêu cực:** Đây là hiện tượng khi kiến thức từ miền nguồn cản trở hiệu suất trong miền đích. Nếu tác vụ nguồn quá khác biệt hoặc thông tin không liên quan được chuyển giao, chuyển giao tiêu cực sẽ xảy ra. Cần thận trọng trong việc lựa chọn mô hình và các kỹ thuật tinh chỉnh có thể giảm thiểu vấn đề này.

- **Tính khả dụng của dữ liệu:** Học chuyển giao dựa vào dữ liệu được gắn nhãn trong miền nguồn. Tuy nhiên, trong một số trường hợp, việc thu thập dữ liệu gắn nhãn có thể tốn kém hoặc khan hiếm. Điều này gây khó khăn khi miền đích có ít dữ liệu được gắn nhãn.

5. Mạng Transformer

5.1. Khái niệm

Transformer (2017) là kiến trúc mạng sử dụng cơ chế chú ý (Attention) để xử lý dữ liệu tuần tự, vượt trội hơn RNN trong các tác vụ ngôn ngữ và thị giác máy tính nhờ khả năng học phụ thuộc dài hạn hiệu quả và tính song song cao.

5.2. Cách hoạt động

Trọng tâm của Transformer là cơ chế self-attention, cho phép mỗi phần tử trong chuỗi đánh giá mức độ liên quan của nó với các phần tử còn lại. Điều này giúp mô hình hiểu được ngữ cảnh sâu hơn, từ đó tạo ra biểu diễn tốt hơn cho các tác vụ ngôn ngữ và thị giác.

Kiến trúc Transformer gồm hai phần chính: **Encoder** mã hóa toàn bộ đầu vào thành biểu diễn ngữ nghĩa, còn **Decoder** tạo ra đầu ra từng bước dựa trên thông tin từ encoder và đầu ra trước đó. Mô hình này đặc biệt hiệu quả trong các nhiệm vụ như dịch máy, tóm tắt văn bản và sinh ngôn ngữ.

Từ kiến trúc gốc, nhiều biến thể đã được phát triển. **BERT** chỉ dùng phần encoder, nổi bật trong các tác vụ hiểu ngữ nghĩa như phân tích cảm xúc, phân loại văn bản. **GPT** sử dụng decoder để sinh văn bản mạch lạc. Trong lĩnh vực thị giác, **Vision Transformer (ViT)** chia ảnh thành các phần nhỏ (patch) và xử lý tương tự như chuỗi từ.

Với hiệu quả cao và khả năng xử lý linh hoạt, Transformer đã trở thành nền tảng cho nhiều hệ thống trí tuệ nhân tạo hiện đại.

5.3. Đặc điểm nổi bật

- **Xử lý chuỗi dài hiệu quả:** Nhờ cơ chế tự chú ý (self-attention), Transformer có thể học được các mối quan hệ giữa các phần tử trong chuỗi bất kể khoảng cách, điều mà các mô hình RNN truyền thống thường gặp khó khăn.
- **Hỗ trợ song song hóa:** Không giống như RNN phải xử lý dữ liệu theo thứ tự thời gian, Transformer xử lý toàn bộ chuỗi đầu vào cùng lúc, cho phép khai thác tốt hơn sức mạnh của phần cứng hiện đại như GPU/TPU, từ đó đẩy nhanh quá trình huấn luyện.
- **Khả năng mở rộng cao:** Mô hình có thể được mở rộng với nhiều lớp và tham số hơn để tăng hiệu quả, như đã thấy với các mô hình lớn như BERT, GPT, T5.

- **Tài nguyên tính toán lớn:** Transformer đòi hỏi bộ nhớ và sức mạnh tính toán đáng kể, đặc biệt khi xử lý các chuỗi dài hoặc huấn luyện trên tập dữ liệu lớn.

5.4. Ứng dụng

- **Dịch máy:** Transformer được giới thiệu lần đầu trong lĩnh vực dịch máy với mô hình "Attention is All You Need", và đã vượt qua nhiều phương pháp trước đó nhờ khả năng hiểu và chuyển đổi ngữ cảnh toàn diện giữa các ngôn ngữ
- **Tóm tắt văn bản:** Các mô hình như BART hoặc T5 sử dụng kiến trúc Transformer để tạo ra bản tóm tắt ngắn gọn, súc tích nhưng vẫn giữ được ý chính của văn bản dài.
- **Xử lý hình ảnh:** Vision Transformer (ViT) đã mở rộng kiến trúc Transformer sang lĩnh vực thị giác máy tính, cho phép mô hình học các biểu diễn hình ảnh mạnh mẽ mà không cần mạng nơ-ron tích chập (CNN).
- **Chatbot và tạo văn bản (GPT):** Các mô hình GPT (Generative Pre-trained Transformer) là nền tảng cho nhiều chatbot tiên tiến hiện nay, bao gồm ChatGPT. Chúng có khả năng sinh ngôn ngữ tự nhiên, trả lời câu hỏi, viết sáng tạo, và hỗ trợ trong nhiều tác vụ ngôn ngữ khác.

6. Học bán giám sát và tự giám sát (Semi-Supervised & Self-Supervised Learning)

6.1. Khái niệm

Học bán giám sát là phương pháp học máy kết hợp giữa dữ liệu có nhãn và dữ liệu không nhãn để huấn luyện mô hình. Điều này đặc biệt hữu ích khi dữ liệu có nhãn khan hiếm hoặc tốn kém để thu thập, trong khi dữ liệu không nhãn thì dồi dào.

Học tự giám sát là một hình thức đặc biệt của học không giám sát, trong đó hệ thống tự tạo nhãn từ chính dữ liệu không nhãn. Mô hình học cách dự đoán một phần của dữ liệu từ phần còn lại, qua đó học được các biểu diễn giàu thông tin mà không cần gán nhãn thủ công.

6.2. Cách hoạt động của học bán giám sát và tự giám sát

❖ Học bán giám sát:

- **Bước 1: Huấn luyện mô hình trên dữ liệu có nhãn:** Bắt đầu bằng việc huấn luyện mô hình trên một tập dữ liệu nhỏ có nhãn. Đây là tập dữ liệu có chất lượng cao và mô hình sẽ học các đặc trưng của dữ liệu qua quá trình huấn luyện.

- **Bước 2: Dự đoán nhãn cho dữ liệu không nhãn (Pseudo-labeling):** Sau khi huấn luyện mô hình trên dữ liệu có nhãn, mô hình sẽ được sử dụng để dự đoán nhãn cho dữ liệu không nhãn. Các nhãn này gọi là "pseudo-labels" (nhãn giả). Dù nhãn giả có thể không hoàn hảo, nhưng chúng sẽ giúp mô hình học thêm từ dữ liệu không nhãn.
- **Bước 3: Thêm nhãn giả vào tập huấn luyện:** Các nhãn giả được thêm vào cùng với dữ liệu có nhãn ban đầu để tạo thành một tập huấn luyện lớn hơn. Mô hình tiếp tục học từ tập dữ liệu này, cải thiện dần dần qua các vòng huấn luyện.
- **Bước 4: Lặp lại quá trình:** Quá trình này được lặp đi lặp lại: huấn luyện → dự đoán → tinh chỉnh → huấn luyện lại. Mỗi vòng lặp mô hình sẽ học thêm và cải thiện dựa trên các nhãn giả và dữ liệu có nhãn.

Ví dụ: Giả sử bạn có 100 ảnh có nhãn ban đầu. Sau khi huấn luyện, mô hình dự đoán nhãn cho 1000 ảnh không nhãn. Các nhãn giả này sẽ được sử dụng để huấn luyện mô hình thêm, giúp cải thiện hiệu suất trên tập dữ liệu lớn hơn.

Kỹ thuật bổ sung trong học bán giám sát:

- **Co-training:** Sử dụng nhiều mô hình huấn luyện đồng thời, mỗi mô hình dự đoán nhãn cho dữ liệu không nhãn và bổ sung nhãn giả cho nhau. Điều này giúp tăng cường độ chính xác của nhãn giả và cải thiện hiệu quả học.
- **Self-training:** Một biến thể của học bán giám sát, trong đó mô hình tự huấn luyện lại bằng cách dự đoán nhãn cho dữ liệu không nhãn và sử dụng các nhãn đó cho quá trình huấn luyện tiếp theo.

❖ Học tự giám sát:

- **Bước 1: Mô hình học biểu diễn qua nhiệm vụ giả định (Pretext Task):** Mô hình bắt đầu học thông qua một nhiệm vụ giả định, mà nhiệm vụ này không cần dữ liệu có nhãn. Mục tiêu là giúp mô hình học được các đặc trưng quan trọng từ dữ liệu. Các nhiệm vụ này có thể là:
 - **Masked Language Modeling (MLM):** Một nhiệm vụ phổ biến trong xử lý ngôn ngữ tự nhiên (NLP), như mô hình **BERT**. Trong nhiệm vụ này, một số từ trong câu sẽ bị che đi, và mô hình sẽ cố gắng đoán các từ bị thiếu, từ đó học được các mối quan hệ ngữ nghĩa và ngữ pháp.
 - **Inpainting (hoàn thành ảnh):** Trong trường hợp nhận diện ảnh, mô hình sẽ học cách hoàn thiện một bức ảnh bằng cách khôi phục các phần bị thiếu. Ví dụ: trong **Deep Image Prior**, mô hình học cách tái tạo các phần bị che khuất của hình ảnh.

- **Contrastive Learning:** Mô hình học cách phân biệt giữa các phiên bản biến đổi của cùng một đối tượng và các đối tượng khác nhau. Các phương pháp như **SimCLR** và **MoCo** đã sử dụng kỹ thuật này để học các đặc trưng của hình ảnh mà không cần nhãn. Ví dụ: mô hình sẽ học cách phân biệt một hình ảnh của con mèo với một hình ảnh của con chó, bất kể các biến đổi như góc chụp hoặc độ sáng.

- **Bước 2: Tinh chỉnh cho nhiệm vụ thực tế:** Sau khi học được các đặc trưng tổng quát từ nhiệm vụ giả định, mô hình sẽ được tinh chỉnh (fine-tuned) trên các nhiệm vụ thực tế có nhãn. Việc này giúp mô hình chuyển từ học đặc trưng tổng quát sang học đặc trưng phù hợp với bài toán cụ thể.

Ví dụ: Với một mạng học tự giám sát cho ảnh, mô hình có thể được huấn luyện với nhiệm vụ hoàn thành các phần thiếu trong một bức ảnh. Sau khi học xong, mô hình có thể được sử dụng để phân loại ảnh trong các bài toán như phân loại ảnh động vật hay nhận diện đối tượng.

❖ Một số nhiệm vụ pretext điển hình:

- **Masked Language Modeling:** Che một số từ trong câu và yêu cầu mô hình đoán lại từ bị ẩn (BERT).
- **Inpainting (hoàn thành ảnh):** Che một phần hình ảnh và yêu cầu mô hình khôi phục phần đó.
- **Contrastive Learning:** Học cách phân biệt giữa các phiên bản biến đổi của cùng một đối tượng và đối tượng khác (SimCLR, MoCo).

Kết quả: Mô hình học được các đặc trưng tổng quát, có thể tái sử dụng cho nhiều bài toán khác nhau, kể cả khi dữ liệu có nhãn rất ít.

6.3. Ứng dụng

- **Phân loại văn bản:** Học bán giám sát được áp dụng hiệu quả khi có ít nhãn dữ liệu, ví dụ như trong phân tích cảm xúc từ các bài đánh giá sản phẩm. Dữ liệu không nhãn có thể giúp cải thiện độ chính xác của mô hình trong các tác vụ phân loại văn bản.
- **Nhận diện hình ảnh:** Trong các lĩnh vực như y tế (chẩn đoán bệnh từ ảnh X-quang) hay nông nghiệp (phân loại cây trồng), học bán giám sát có thể giúp tạo ra các mô hình mạnh mẽ dù chỉ có ít dữ liệu nhãn. Điều này giúp tiết kiệm chi phí và thời gian trong việc thu thập dữ liệu nhãn.

- **Mô hình ngôn ngữ lớn:** Các mô hình ngôn ngữ như BERT hay RoBERTa có thể được huấn luyện theo phương pháp tự giám sát với các nhiệm vụ như dự đoán từ bị che (Masked Language Modeling). Sau đó, mô hình này có thể được tinh chỉnh để thực hiện các nhiệm vụ NLP cụ thể như phân loại văn bản hay trả lời câu hỏi.
- **Xử lý âm thanh:** Trong nhận diện giọng nói, đặc biệt khi có ít dữ liệu nhãn, học bán giám sát có thể giúp tăng độ chính xác và cải thiện hiệu suất nhận diện âm thanh, hỗ trợ cho các ứng dụng như trợ lý giọng nói hay hệ thống chuyển đổi giọng nói thành văn bản
- **Tăng cường dữ liệu:** Kết hợp giữa học bán giám sát và học chuyển giao có thể giúp cải thiện hiệu suất mô hình khi làm việc với các tập dữ liệu nhỏ, đặc biệt là khi số lượng dữ liệu nhãn rất hạn chế.

6.4. So sánh học bán giám sát và tự giám sát

Tiêu chí	Học bán giám sát (Semi-supervised Learning)	Học tự giám sát (Self-supervised Learning)
Dữ liệu yêu cầu	Kết hợp dữ liệu có nhãn và không nhãn.	Chỉ sử dụng dữ liệu không nhãn và tự tạo nhãn trong quá trình học.
Phương pháp học	Dùng kỹ thuật như pseudo-labeling (gán nhãn giả) hoặc co-training.	Tạo các nhiệm vụ pretext (ví dụ: Masked Language Modeling, Inpainting) để học biểu diễn.
Ứng dụng phổ biến	Phân loại hình ảnh, nhận diện âm thanh, phân tích văn bản khi ít nhãn.	Mô hình ngôn ngữ như BERT, GPT, SimCLR cho nhận diện hình ảnh và giọng nói.
Yêu cầu dữ liệu có nhãn	Cần một lượng nhỏ dữ liệu có nhãn ban đầu.	Không cần dữ liệu có nhãn, tự tạo nhãn từ dữ liệu không nhãn.
Ví dụ	Phân loại ảnh: Huấn luyện mô hình nhận diện động vật với 1000 ảnh không nhãn và 100 ảnh có nhãn. Sau đó, gán nhãn cho các ảnh không nhãn và tiếp tục huấn luyện.	BERT: Huấn luyện mô hình ngôn ngữ BERT với nhiệm vụ Masked Language Modeling, nơi một số từ trong câu bị che và mô hình phải dự đoán từ bị ẩn.

CHƯƠNG 3: PHÂN TÍCH CẢM XÚC BẰNG NGÔN NGỮ MÔ HÌNH BERT

1. Tổng quan

Trong kỷ nguyên số hóa, nơi dữ liệu văn bản từ mạng xã hội, đánh giá sản phẩm, và các nền tảng trực tuyến bùng nổ, việc hiểu được cảm xúc và ý kiến ẩn sau những dòng chữ đã trở thành một nhu cầu thiết yếu. Nền tảng trực tuyến phát triển cũng kéo theo đó cảm xúc của con người cũng biến đổi theo từng chuyển động công nghệ, và những cảm xúc đó có thể được đo lường và đánh giá. Vậy tại sao phải phân tích cảm xúc? Câu trả lời nằm ở khả năng khai thác thông tin giá trị từ dữ liệu phi cấu trúc, giúp các tổ chức và doanh nghiệp không chỉ nắm bắt được tâm lý, thái độ của người dùng mà còn đưa ra các quyết định chiến lược dựa trên dữ liệu.

Phân tích cảm xúc (Sentiment Analysis) là quá trình xác định và phân loại cảm xúc được thể hiện trong văn bản, thường được biểu thị dưới dạng tích cực, tiêu cực, hoặc trung tính, và đôi khi bao gồm cả các sắc thái phức tạp như mỉa mai, hài hước, hay cảm xúc lẫn lộn. Bằng cách phân tích cảm xúc, các hệ thống có thể trả lời các câu hỏi như: Khách hàng có hài lòng với sản phẩm không? Công chúng phản ứng thế nào với một sự kiện? Hay một thương hiệu đang được nhìn nhận ra sao trên mạng xã hội?

Phân tích cảm xúc không chỉ là một bài toán kỹ thuật mà còn là cầu nối giữa dữ liệu thô và các ứng dụng thực tiễn, từ cải thiện trải nghiệm khách hàng, tối ưu hóa chiến lược tiếp thị, đến giám sát dư luận xã hội. Tuy nhiên, các phương pháp truyền thống như túi từ (Bag of Words), TF-IDF, hay các mô hình học máy đơn giản như SVM và Naive Bayes thường gặp khó khăn trong việc nắm bắt ngữ cảnh và các mối quan hệ phức tạp giữa các từ trong câu. Những hạn chế này đã thúc đẩy sự phát triển của các mô hình học sâu, đặc biệt là các kiến trúc Transformer như BERT (Bidirectional Encoder Representations from Transformers).

Được giới thiệu bởi Google vào năm 2018 trong bài báo “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding” của Devlin et al., BERT đã tạo ra một cuộc cách mạng trong xử lý ngôn ngữ tự nhiên (NLP). Với cơ chế tự chú ý (self-attention), BERT có khả năng hiểu ngữ cảnh hai chiều của văn bản, giúp trích xuất các đặc trưng ngữ nghĩa sâu sắc, từ đó vượt trội trong các bài toán phức tạp như phân tích cảm xúc. BERT được huấn luyện trước (pre-trained) trên lượng dữ liệu khổng lồ từ Wikipedia và BookCorpus thông qua hai tác vụ chính: dự đoán từ bị che (Masked Language Modeling - MLM) và dự đoán câu tiếp theo (Next Sentence Prediction - NSP). Nhờ đó, BERT không chỉ hiểu được cấu trúc ngôn ngữ mà còn nắm bắt các sắc thái cảm xúc tinh vi, khiến nó trở thành lựa chọn lý tưởng cho việc phân tích cảm xúc.

Có một câu hỏi nữa được đặt ra là: Làm thế nào để tận dụng sức mạnh của BERT để phân loại cảm xúc một cách chính xác và hiệu quả? Tại đây có rất nhiều thông tin và quy trình được xử lý. Quy trình này bao gồm lựa chọn biến thể BERT phù hợp, xử lý dữ liệu văn bản, tinh chỉnh mô hình, và đánh giá hiệu suất, với mục tiêu tạo ra một hệ thống có khả năng ứng dụng thực tế trong các kịch bản đa dạng.

1.1. Mục tiêu chính

Quy trình xây dựng mô hình phân tích cảm xúc bằng BERT được thiết kế để đạt được các mục tiêu cụ thể sau:

- **Mô tả chi tiết quy trình xây dựng mô hình:**
 - Lựa chọn biến thể BERT phù hợp (như BERT-base-uncased, BERT-large, hoặc mBERT cho dữ liệu đa ngôn ngữ) dựa trên yêu cầu về hiệu suất, tài nguyên tính toán, và đặc điểm của tập dữ liệu.
 - Triển khai mô hình sử dụng thư viện Hugging Face Transformers, kết hợp với TensorFlow hoặc PyTorch, tích hợp các lớp phân loại (classification head) để dự đoán cảm xúc.
 - Tinh chỉnh (fine-tuning) mô hình trên tập dữ liệu mục tiêu, tối ưu hóa các tham số để đạt độ chính xác cao, đồng thời đảm bảo mô hình không bị overfitting.
- **Giải thích quy trình xử lý dữ liệu:**
 - Tiền xử lý văn bản: Chuẩn hóa văn bản (loại bỏ ký tự đặc biệt, chuyển đổi chữ thường), mã hóa token bằng tokenizer của BERT, và tạo các định dạng đầu vào như input IDs, attention masks.
 - Phân chia tập dữ liệu thành các phần huấn luyện, kiểm tra, và thử nghiệm để đánh giá khả năng tổng quát hóa của mô hình.
 - Áp dụng kỹ thuật tăng cường dữ liệu văn bản (nếu cần), như thay thế từ đồng nghĩa hoặc dịch ngược (back-translation), để tăng tính đa dạng và cải thiện khả năng tổng quát hóa.
- **Đánh giá hiệu suất mô hình:**
 - Sử dụng các chỉ số như độ chính xác (accuracy), F1-score, precision, recall, và ma trận nhầm lẫn (confusion matrix) để đo lường hiệu quả trên tập kiểm tra.
 - Phân tích khả năng tổng quát hóa trên dữ liệu mới, đặc biệt trong các tình huống thực tế với văn bản đa dạng về cách diễn đạt, từ vựng, hoặc ngữ cảnh.
 - So sánh hiệu suất của BERT với các phương pháp truyền thống để làm rõ ưu điểm của mô hình.

1.2. Vai trò của huấn luyện mô hình trong toán hệ thống

Trong một hệ thống phân tích cảm xúc, mô hình BERT là thành phần cốt lõi, đóng vai trò xử lý và phân loại dữ liệu văn bản để xác định cảm xúc. Việc phân tích cảm xúc là cần thiết vì nó giúp chuyển đổi các dữ liệu văn bản phi cấu trúc thành thông tin có ý nghĩa, từ đó hỗ trợ các quyết định kinh doanh, cải thiện trải nghiệm người dùng, hoặc giám sát xu hướng xã hội. Việc huấn luyện mô hình có các vai trò quan trọng sau:

- **Trích xuất đặc trưng ngữ nghĩa:**
 - BERT sử dụng cơ chế tự chú ý để tạo ra các biểu diễn ngữ nghĩa (contextualized embeddings) dựa trên toàn bộ ngữ cảnh của câu. Điều này cho phép mô hình nhận diện các sắc thái cảm xúc phức tạp, như sự mỉa mai trong câu “Great job, as always!” hoặc cảm xúc tích cực ngấm trong “Not bad at all.”
 - Trong quá trình huấn luyện, mô hình điều chỉnh các biểu diễn này để tập trung vào các đặc trưng liên quan đến cảm xúc, như từ vựng cảm xúc, cấu trúc câu, hoặc ngữ điệu.
- **Tổng quát hóa kiến thức:**
 - Một mô hình BERT được huấn luyện tốt có khả năng xử lý các văn bản mới với cách diễn đạt, ngữ cảnh, hoặc từ vựng khác biệt, đảm bảo hiệu quả trong các ứng dụng thực tế như phân tích bài đăng mạng xã hội hoặc đánh giá sản phẩm.
 - Quá trình tinh chỉnh giúp mô hình thích nghi với tập dữ liệu cụ thể, đồng thời tận dụng kiến thức ngôn ngữ tổng quát từ huấn luyện trước.
- **Tối ưu hiệu suất và triển khai thực tế:**
 - Mô hình cần đạt độ chính xác cao, đồng thời tối ưu về tốc độ suy luận và sử dụng tài nguyên để triển khai trên các hệ thống thời gian thực, như chatbot hoặc hệ thống phân tích phản hồi khách hàng.
 - Các kỹ thuật như giảm kích thước mô hình (model distillation) hoặc sử dụng biến thể nhẹ như DistilBERT giúp đáp ứng yêu cầu triển khai trên thiết bị hạn chế tài nguyên.
- **Ảnh hưởng đến hệ thống tổng thể:** Nếu mô hình không được huấn luyện tốt (overfitting hoặc underfitting), hệ thống có thể đưa ra dự đoán sai lệch, dẫn đến các vấn đề như hiểu sai ý kiến khách hàng hoặc bỏ sót tín hiệu tiêu cực. Một mô hình hiệu quả đảm bảo độ tin cậy và nâng cao giá trị ứng dụng.

1.3. Thiết kế kiến trúc mô hình ngôn ngữ BERT

1.3.1. Lý do chọn BERT

Mô hình BERT (Bidirectional Encoder Representations from Transformers) được chọn làm nền tảng cho bài toán phân tích cảm xúc nhờ các đặc điểm vượt trội trong xử lý ngôn ngữ tự nhiên (NLP):

- **Hiểu ngữ cảnh hai chiều:** BERT sử dụng cơ chế tự chú ý (self-attention) để xem xét toàn bộ ngữ cảnh của câu theo cả hai hướng (trái sang phải và ngược lại). Điều này cho phép mô hình nắm bắt các mối quan hệ phức tạp giữa các từ, ví dụ, nhận diện sự mỉa mai trong câu “Great job, really!” hoặc sự kết hợp cảm xúc trong “The movie was slow but surprisingly touching.”
- **Huấn luyện trước mạnh mẽ:** BERT đã được huấn luyện trước trên lượng dữ liệu khổng lồ (Wikipedia và BookCorpus) với hai tác vụ: dự đoán từ bị che (Masked Language Modeling - MLM) và dự đoán câu tiếp theo (Next Sentence Prediction - NSP). Điều này giúp BERT tạo ra các biểu diễn ngữ nghĩa phong phú, phù hợp cho nhiều bài toán NLP.
- **Hiệu quả khi tinh chỉnh:** Với các trọng số đã huấn luyện trước, BERT chỉ cần tinh chỉnh (fine-tuning) trên tập dữ liệu cụ thể để đạt hiệu suất cao, ngay cả khi dữ liệu huấn luyện hạn chế. Điều này rất quan trọng cho bài toán phân tích cảm xúc, nơi dữ liệu gán nhãn thường không dồi dào.
- **Hỗ trợ công cụ mạnh mẽ:** Thư viện Hugging Face Transformers cung cấp các công cụ dễ sử dụng, từ tokenizer đến mô hình huấn luyện trước, giúp triển khai BERT một cách hiệu quả và linh hoạt.
- **Khả năng tổng quát hóa:** BERT có thể xử lý các văn bản với cách diễn đạt đa dạng, từ tiếng lóng đến ngôn ngữ trang trọng, phù hợp với dữ liệu thực tế như bài đăng mạng xã hội hoặc đánh giá sản phẩm.

1.3.2. Mô tả kiến trúc mô hình ngôn ngữ BERT

Kiến trúc của BERT dựa trên mô hình Transformer, cụ thể là phần mã hóa (encoder) của Transformer gốc được giới thiệu trong bài báo “Attention is All You Need” (Vaswani et al., 2017). BERT bao gồm nhiều lớp Transformer xếp chồng, mỗi lớp thực hiện các phép toán tự chú ý và lan truyền xuôi (feed-forward) để tạo ra các biểu diễn ngữ nghĩa cho văn bản đầu vào. Để áp dụng BERT cho bài toán phân tích cảm xúc, một lớp phân loại (classification head) được thêm vào để dự đoán nhãn cảm xúc (tích cực, tiêu cực, trung tính).

BERT có ba biến thể chính:

- BERT-small: 8 lớp encoder ($L=8$), 512 chiều ẩn ($H=512$), 8 đầu tự chú ý ($A=8$), tổng 15,62 triệu tham số.
- BERT-base: 12 lớp encoder ($L=12$), 768 chiều ẩn ($H=768$), 12 đầu tự chú ý ($A=12$), tổng cộng 110 triệu tham số.
- BERT-large: 24 lớp encoder ($L=24$), 1024 chiều ẩn ($H=1024$), 16 đầu tự chú ý ($A=16$), tổng cộng 340 triệu tham số.

Trong bài toán phân tích cảm xúc, tôi chọn BERT-small-uncased (không phân biệt chữ hoa/thường) vì nó cân bằng giữa hiệu suất và tài nguyên tính toán, phù hợp với các hệ thống có cấu hình trung bình. Kiến trúc BERT bao gồm ba thành phần chính: lớp đầu vào, các lớp Transformer (encoder), và lớp đầu ra.

a) Lớp đầu vào (Input Layer)

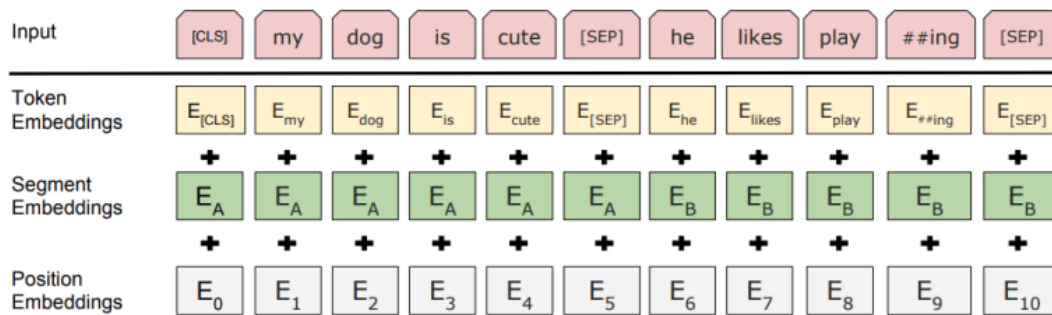
Lớp đầu vào của BERT chuyển đổi văn bản thô thành định dạng mà mô hình có thể xử lý. Quy trình này bao gồm:

- Tokenization:

- Văn bản được mã hóa thành các token bằng **WordPiece tokenizer**, chia từ thành các đơn vị nhỏ hơn (subword units) để xử lý từ vựng hiếm. Ví dụ, từ “playing” có thể được chia thành “play” và “##ing”.
- Từ điển của BERT-base chứa khoảng 30,000 token. Mỗi token được ánh xạ thành một ID duy nhất.
- Hai token đặc biệt được thêm vào:
 - **[CLS]**: Đặt ở đầu chuỗi, đại diện cho toàn bộ câu và được sử dụng cho các tác vụ phân loại (như phân tích cảm xúc).
 - **[SEP]**: Phân tách các câu trong cặp câu hoặc đánh dấu kết thúc chuỗi.

- Nhúng đầu vào (Input Embedding):

- Mỗi token được biểu diễn bằng một vector nhúng, bao gồm ba phần
 - **Token Embedding**: Biểu diễn ý nghĩa của token, lấy từ ma trận nhúng học được.
 - **Segment Embedding**: Phân biệt các câu trong cặp câu (ví dụ, câu A và câu B trong NSP). Trong phân tích cảm xúc, thường chỉ có một câu, nên segment embedding có giá trị giống nhau.
 - **Position Embedding**: Biểu diễn vị trí của token trong chuỗi, vì BERT không có tính tuần tự như RNN. Position embedding được học hoặc sử dụng hàm sin-cos (như trong Transformer gốc).



- Tổng vector nhúng cho mỗi token có chiều 768 (trong BERT-base), được tính bằng cách cộng ba loại nhúng trên.
- **Attention Mask:** Một mảng nhị phân (0 hoặc 1) xác định các token hợp lệ (1) và các token padding (0). Padding được thêm để đảm bảo các chuỗi có độ dài cố định (thường là 128 hoặc 512 token).

b) Các lớp Transformer (Encoder Layers)

Mô hình BERT-small là một chồng gồm **8 lớp Transformer**, mỗi lớp có nhiệm vụ xử lý và biểu diễn thông tin ngữ nghĩa qua hai thành phần chính: **cơ chế tự chú ý đa đầu (Multi-Head Self-Attention)** và **mạng nơ-ron lan truyền xuôi (Feed-Forward Neural Network - FFN)**.

- Multi-Head Self-Attention (Tự chú ý đa đầu)

Cơ chế tự chú ý cho phép mỗi token trong câu "chú ý" đến các token còn lại, giúp mô hình nắm bắt được các mối quan hệ ngữ nghĩa toàn cục.

Ví dụ: Trong câu "*The movie was slow but enjoyable*", mô hình có thể nhận ra mối liên hệ giữa "slow" và "enjoyable".

- BERT-small sử dụng **8 đầu tự chú ý**, mỗi đầu xử lý thông tin trong không gian chiều $d_k = 512/8 = 64$.
- Công thức tính attention như sau:

$$Attention(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)$$

Trong đó: Q, K, V: là các ma trận *query*, *key* và *value* thu được từ vector đầu vào thông qua phép biến đổi tuyến tính.

- Feed-Forward Neural Network (FFN)

Mỗi token sau khi qua khối self-attention sẽ tiếp tục được xử lý qua một mạng nơ-ron hai lớp (áp dụng độc lập cho từng token):

$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

Trong đó:

- W1: ánh xạ từ 512 chiều lên 2048 chiều (gấp 4 lần).
- W2: ánh xạ ngược từ 2048 chiều về 512 chiều.

Hàm kích hoạt ReLU được sử dụng để thêm tính phi tuyến, giúp mô hình học được các đặc trưng phức tạp hơn.

- **Kết nối dư và Chuẩn hóa lớp**

Để ổn định quá trình huấn luyện và duy trì dòng chảy thông tin, mỗi lớp Transformer còn bao gồm:

- **Kết nối dư (Residual Connection):**

$$\text{Output} = x + \text{SubLayer}(x)$$

Trong đó, SubLayer là Self-Attention hoặc FFN.

- **Chuẩn hóa lớp (Layer Normalization):**

$$\text{LayerNorm}(x) = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} \cdot \gamma + \beta$$

Với:

- μ, σ^2 : trung bình và phương sai của đầu vào.
- γ, β : là các tham số có thể học được.

Nhờ cấu trúc lặp lại của các lớp Transformer như vậy, BERT-small có khả năng mô hình hóa thông tin ngữ cảnh hiệu quả, ngay cả với mô hình kích thước nhỏ.

- **Số lớp:** Với 8 lớp Transformer, BERT-small tạo ra các biểu diễn ngữ nghĩa phong phú nhưng nhẹ hơn nhiều so với BERT-base (12 lớp) hoặc BERT-large (24 lớp).

c) Lớp đầu ra (Output Layer)

Để sử dụng BERT-small cho nhiệm vụ phân tích cảm xúc, mô hình được mở rộng thêm một lớp phân loại ở cuối. Quá trình này bao gồm các bước sau:

1. Vector [CLS]

- Token đặc biệt [CLS] được thêm vào đầu mỗi câu đầu vào.
- Sau khi đi qua toàn bộ 8 lớp Transformer, vector biểu diễn của token [CLS] (có kích thước 512 chiều) được chọn làm đại diện cho toàn bộ câu.
- Vector này tổng hợp ngữ nghĩa toàn cục và được sử dụng làm đầu vào cho lớp phân loại.

2. Lớp phân loại

Vector [CLS] được đưa vào một lớp tuyến tính (linear layer) để tạo đầu ra thô (logits):

$$\text{Output} = W \cdot [\text{CLS}] + b$$

- W : ma trận trọng số có kích thước 3×512 , tương ứng với 3 lớp cảm xúc: **tích cực, tiêu cực, trung tính**.
- b : vector bias.

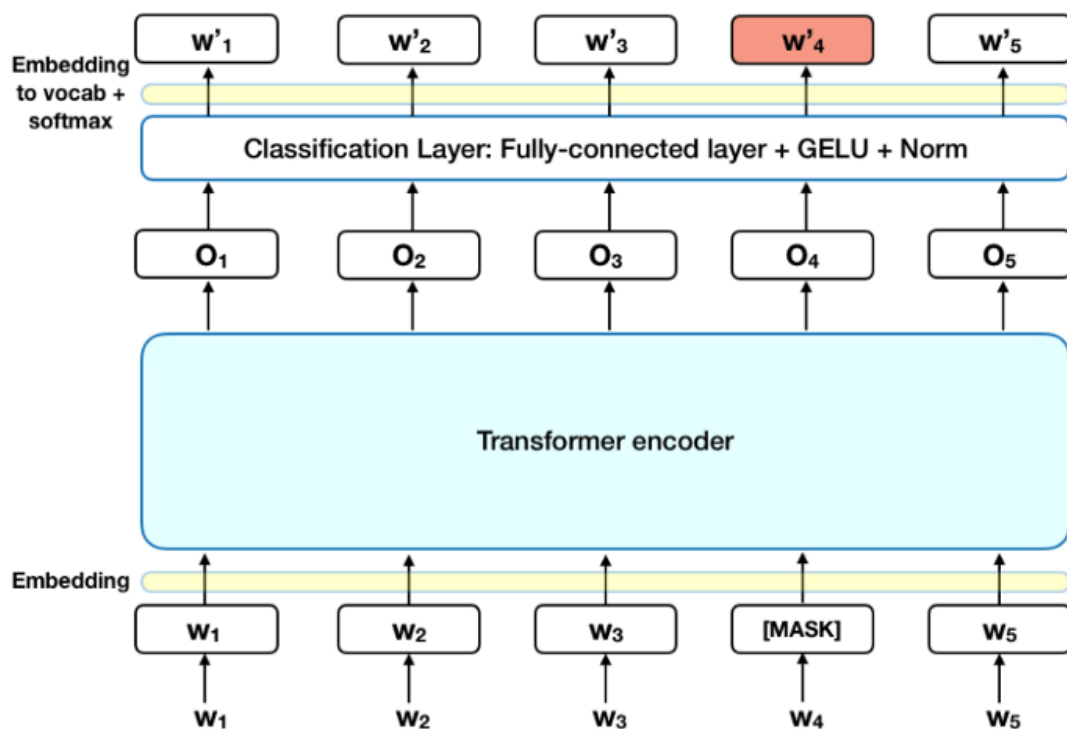
Một lớp **Dropout** (thường với tỷ lệ 0.1) được áp dụng trước lớp tuyến tính để giảm hiện tượng **overfitting** trong quá trình huấn luyện.

3. Hàm kích hoạt Softmax

- Đầu ra từ lớp tuyến tính được đưa qua hàm **Softmax** để chuyển thành xác suất:

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

- Kết quả là một vector xác suất (có 3 giá trị), thể hiện **xác suất câu thuộc mỗi lớp cảm xúc**.



d) Các thông số huấn luyện

- **Learning rate:** $2e-5$ đến $5e-5$, đảm bảo tinh chỉnh ổn định.
- **Batch size:** 16 hoặc 32, phù hợp với GPU 4GB hoặc 8GB nhờ kích thước nhỏ của BERT-small.
- **Epochs:** 3–5 epochs, vì BERT-small hội tụ nhanh khi tinh chỉnh.
- **Optimizer:** AdamW (Adam với điều chỉnh trọng số – weight decay)
 - $\beta_1 = 0.9$
 - $\beta_2 = 0.999$
 - $\epsilon = 1e^{-8}$
- **Loss function: Categorical Crossentropy**
 - Dùng trong bài toán **phân loại nhiều lớp** (positive, negative, neutral).
 - Kết hợp tốt với đầu ra là xác suất từ hàm Softmax.
- **Metric đánh giá:**
 - **Accuracy:** Đo độ chính xác tổng thể.
 - **F1-score:** Cân bằng precision và recall, hữu ích khi dữ liệu không cân bằng.
- **Max sequence length:** 128 token.
- **Dropout:** 0.1. Áp dụng để giảm nguy cơ **overfitting**, đặc biệt trong lớp phân loại.




2. Thu thập và xử lý dữ liệu

2.1. Nguồn dữ liệu

Bài báo "Sentiment Analysis on the Impact of Coronavirus in Social Life Using the BERT Model" (2021) tập trung vào việc phân tích cảm xúc của người dân toàn cầu và Ấn Độ thông qua các bài đăng trên Twitter trong giai đoạn đầu của đại dịch COVID-19.

Thông tin chi tiết về bộ dữ liệu:

- + Nguồn gốc: Dữ liệu được thu thập từ Twitter thông qua các API như Tweepy và Twitter Scraper.
- + Thời gian thu thập: Từ ngày 20 tháng 1 đến ngày 25 tháng 4 năm 2020.
- + Ngôn ngữ: Chỉ bao gồm các tweet bằng tiếng Anh.
- + Nhãn dữ liệu: Mỗi đoạn text đi kèm với chỉ số Id, và các nhãn cảm xúc

 nlp_test.csv	5/12/2025 11:27 PM	Microsoft Excel Co...	1,064 KB
 nlp_train.csv	5/12/2025 11:27 PM	Microsoft Excel Co...	4,342 KB
 nlp_valid.csv	5/12/2025 11:27 PM	Microsoft Excel Co...	516 KB

File **npl_train.csv**: Được sử dụng để **huấn luyện mô hình** — tức là mô hình học các đặc trưng, mẫu (pattern), và mối liên hệ giữa dữ liệu đầu vào (văn bản) và nhãn (cảm xúc). Bao gồm:

- + Id: Chỉ số của đoạn văn bản
- + Text : Nội dung của đoạn văn bản
- + 12 cột tương trưng cho 12 loại cảm xúc: anger, anticipation, disgust, fear, joy, love, optimism, pessimism, sadness, surprise, trust, neutral

id	text	anger	anticipatio	disgust	fear	joy	love	optimism	pessimism	sadness	surprise	trust	neutral
0	He was	1	0	1	0	0	0	0	1	0	0	0	0
1	I'm going	1	1	1	1	0	0	0	1	0	0	0	0
2	By	1	1	1	1	0	0	0	1	0	0	0	0
3	Likewise,	1	0	1	1	0	0	0	1	0	0	0	0
4	People	0	0	0	0	0	0	0	0	0	0	0	1
5	>At	0	1	0	1	0	0	0	1	0	0	0	0
6	I'm not	0	1	0	1	0	0	1	1	0	0	1	0
7	**32	0	0	0	0	0	0	1	0	0	0	0	0
8	That's	1	0	1	0	0	0	1	0	0	0	0	0
9	> #	0	1	0	1	0	0	1	0	0	0	0	0
10	alight	1	1	1	1	0	0	0	1	0	0	0	0
11	Extract:	1	1	1	1	0	0	0	1	1	0	0	0
12	As there	0	1	0	1	0	0	0	0	0	0	1	0
13	>The	0	1	0	1	0	0	0	1	0	0	0	0
14	CYTOKIN	0	1	0	1	0	0	0	1	0	0	0	0
15	là€™m 12 a	1	0	1	0	0	0	0	0	1	0	0	0
16	PARIS:Á I'	1	1	1	1	1	1	1	1	0	0	0	0
17	Excerpt	0	1	0	1	0	0	0	0	0	0	0	0
18	In the	0	0	0	0	0	0	0	0	0	0	0	1
19	The	1	0	1	1	0	0	0	0	0	0	0	0

File **npl_valid.csv**: Dùng để **điều chỉnh siêu tham số** (hyperparameters) như learning rate, batch size, số epoch,...Theo dõi độ chính xác trong quá trình huấn luyện để tránh **overfitting** (quá khớp). Cấu trúc giống file npl_train.csv

File **npl_test.csv**: Dùng để **đánh giá khách quan hiệu suất cuối cùng** của mô hình sau khi đã được huấn luyện và tối ưu. Cấu trúc giống file npl_train.csv

2.2. Tiền xử lý dữ liệu

2.2.1. Xử lý dữ liệu train,valid,test

- Loại bỏ các hàng chứa giá trị thiếu (NaN) trong dữ liệu huấn luyện (train_data).

```
train_sentiment_data= train_data.dropna()
```

- Đặt lại chỉ mục (index) sau khi loại bỏ các hàng có giá trị thiếu, và drop=True đảm bảo không thêm cột chỉ mục cũ vào DataFrame.

```
train_sentiment_data = train_sentiment_data.reset_index(drop=True)
```

- Tương tự với dữ liệu valid(valid_data) và test(test_data)

```
val_sentiment_data = val_data.dropna()
val_sentiment_data = val_sentiment_data.reset_index(drop=True)

test_sentiment_data = test_data.dropna()
test_sentiment_data = test_sentiment_data.reset_index(drop=True)
```

2.2.2. Định nghĩa các nhãn cảm xúc

Định nghĩa các lớp nhãn cảm xúc mà mô hình sẽ phân loại, ví dụ: "anger" (giận dữ), "joy" (vui mừng), v.v.

```
label_class= ['anger', 'anticipation', 'disgust', 'fear', 'joy', 'love',  
              'optimism', 'pessimism', 'sadness', 'surprise', 'trust', 'neutral']
```

2.2.3. Tách dữ liệu đầu vào và nhãn

- Tạo một mảng numpy chứa các văn bản trong dữ liệu huấn luyện từ cột 'text' của train_sentiment_data.

```
input_data_train = train_sentiment_data['text'].values
```

- Tạo một mảng numpy chứa các nhãn cảm xúc (12 nhãn) của dữ liệu huấn luyện từ các cột tương ứng trong train_sentiment_data.

```
label_data_train = train_sentiment_data[label_class].values
```

- Tương tự cho dữ liệu validation và test:

```
input_data_val = val_sentiment_data['text'].values  
label_data_val = val_sentiment_data[label_class].values  
  
input_data_test = test_sentiment_data['text'].values  
label_data_test = test_sentiment_data[label_class].values
```

2.2.4. Kiểm tra kích thước dữ liệu

- Các dòng print() sẽ in ra kích thước của các mảng dữ liệu đầu vào và nhãn của cả ba tập dữ liệu (train, val, test). Đây là bước kiểm tra để đảm bảo dữ liệu đã được chuẩn bị đúng cách.

```
print("Input train shape: ", input_data_train.shape)  
print("Label train shape: ", label_data_train.shape)  
print("Input val shape: ", input_data_val.shape)  
print("Label val shape: ", label_data_val.shape)  
print("Input test shape: ", input_data_test.shape)  
print("Label test shape: ", label_data_test.shape)
```

- Kết quả

```
Input train shape: (1493,)  
Label train shape: (1493, 12)  
Input val shape: (165,)  
Label val shape: (165, 12)  
Input test shape: (374,)  
Label test shape: (374, 12)
```

3. Quá trình huấn luyện mô hình

3.1. Cách triển khai

Để huấn luyện mô hình phân tích cảm xúc, tôi sử dụng framework TensorFlow, Keras, TensorFlow Hub và BERT trong đó:

- **TensorFlow** là một framework mã nguồn mở do Google phát triển, dùng để xây dựng và huấn luyện các mô hình học máy và học sâu. TensorFlow hỗ trợ nhiều nền tảng khác nhau như CPU, GPU và TPU. Trong đề tài, TensorFlow đóng vai trò là nền tảng chính để xây dựng và huấn luyện mô hình phân loại cảm xúc.
- **Keras** là một API cấp cao tích hợp trong TensorFlow, cho phép xây dựng mô hình dễ dàng thông qua các lớp (layers) và cấu trúc mô hình linh hoạt như Functional API hoặc Sequential API. Trong đề tài, tf.keras được sử dụng để:
 - + Khai báo đầu vào văn bản (Input)
 - + Xây dựng các tầng mạng (Dense, Dropout)
 - + Tạo mô hình (Model)
 - + Biên dịch mô hình (compile) với hàm mất mát và thuật toán tối ưu
- **TensorFlow Hub** là một thư viện cho phép sử dụng lại các mô hình học máy đã được huấn luyện sẵn (pre-trained models). Các mô hình này có thể được tải trực tiếp từ internet và tích hợp vào mô hình mới. Trong đề tài, TensorFlow Hub được sử dụng để tải:
 - + **BERT Preprocessing Layer**: Chuẩn hóa và mã hóa văn bản đầu vào theo định dạng yêu cầu của BERT.
 - + **Small BERT Encoder**: Mô hình BERT thu gọn đã được huấn luyện trước, dùng để trích đặc trưng của văn bản.
- **BERT (Bidirectional Encoder Representations from Transformers)**: là mô hình ngôn ngữ tiên tiến do Google phát triển dựa trên kiến trúc Transformer. Mô hình này học biểu diễn ngữ nghĩa của từ và câu bằng cách xem xét ngữ cảnh hai chiều (trái và phải). Trong đề tài, BERT được sử dụng như một bộ trích đặc trưng (feature extractor), không được fine-tune trong quá trình huấn luyện mô hình.

3.2. Xây dựng mô hình

Mô hình được xây dựng bằng cách sử dụng BERT(dạng nhỏ-small BERT) từ TensorFlow_Hub,kết hợp với một mạng MLP (Multilayer Perceptron) ở phần phía sau (fully connected layers) để thực hiện phân loại đa nhãn (multi-label classification).

Hàm build_classifier_model(): Hàm này trả về một mô hình tf.keras.Model đã tích hợp BERT và các lớp Dense để phân loại văn bản đầu vào.

```
def build_classifier_model():
```

1. Input layer (Đầu vào chuỗi văn bản)

#Đầu vào chuỗi văn bản

```
text_input = Input(shape=(), dtype=tf.string, name='text')
```

- Đầu vào là văn bản (dạng chuỗi, không cố định độ dài).
- shape=() nghĩa là đầu vào là một chuỗi duy nhất cho mỗi mẫu.

2. Lớp tiền xử lý BERT

```
preprocessing_layer = hub.KerasLayer('https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3',  
                                     name='preprocessing')  
encoder_inputs = preprocessing_layer(text_input)
```

- Xử lý văn bản: tách token, thêm token đặc biệt ([CLS], [SEP]), padding, tạo attention mask,...
- Chuẩn hóa đầu vào để phù hợp với encoder của BERT.

3. Lớp BERT Encoder

```
encoder = hub.KerasLayer('https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-8_H-512_A-8/2',  
                         trainable=False, name='BERT_encoder')  
outputs = encoder(encoder_inputs)
```

- BERT model pretrained (BERT nhỏ: 8 lớp, 512 chiều ẩn, 8 attention heads).
- trainable=False: mô hình BERT không cập nhật trọng số (đóng băng) → chỉ trích đặc trưng ngôn ngữ.

4. Các lớp Dense phân loại

```
net = outputs['pooled_output']  
net = tf.keras.layers.Dense(400, activation='relu')(net)  
net = tf.keras.layers.Dropout(0.1)(net)  
net = tf.keras.layers.Dense(200, activation='relu')(net)  
net = tf.keras.layers.Dropout(0.1)(net)  
net = tf.keras.layers.Dense(100, activation='relu')(net)  
net = tf.keras.layers.Dropout(0.1)(net)  
net = tf.keras.layers.Dense(50, activation='relu')(net)  
net = tf.keras.layers.Dropout(0.1)(net)  
net = tf.keras.layers.Dense(12, activation='sigmoid', name='classifier')(net)
```

- 4 lớp Dense + Dropout: học các đặc trưng từ đầu ra của BERT.
- Số lượng neuron giảm dần: 400 → 200 → 100 → 50 → 12
- Dropout: ngăn overfitting, mỗi lần "tắt ngẫu nhiên" 10% số neuron
- Lớp cuối cùng: Dense(12, activation='sigmoid'): 12 output → dự đoán xác suất cho 12 nhãn nhị phân độc lập (multi-label classification).

5. Biên dịch chương trình

```
classifier_model = build_classifier_model()
classifier_model.summary()
adam = Adam(learning_rate=0.001)
metric_acc = tf.keras.metrics.BinaryAccuracy(name="binary_accuracy", dtype=None, threshold=0.5)
classifier_model.compile(optimizer=adam, loss='binary_crossentropy', metrics=[metric_acc])
```

- **Optimizer:** Adam, với learning rate = 0.001
- **Loss:** 'binary_crossentropy' phù hợp với multi-label classification (mỗi nhãn độc lập).
- **Metric:** Độ chính xác nhị phân (cho từng nhãn).

6. Bản tóm tắt mô hình đã xây dựng

Model: "model"

Layer (type)	Output Shape	Param #	Connected to
text (InputLayer)	[(None,)]	0	[]
preprocessing (KerasLayer)	{'input_type_ids': (None, 128), 'input_mask': (None, 128), 'input_word_ids': (None, 128)}	0	['text[0][0]']
BERT_encoder (KerasLayer)	{'pooled_output': (None, 512), 'sequence_output': (None, 128, 512), 'encoder_outputs': [(None, 128, 512), (None, 128, 512), (None, 128, 512), (None, 128, 512), (None, 128, 512), (None, 128, 512), (None, 128, 512), (None, 128, 512)], 'default': (None, 512)}	4137318 5	['preprocessing[0][0]', 'preprocessing[0][1]', 'preprocessing[0][2]']
dense (Dense)	(None, 400)	205200	['BERT_encoder[0][9]']
dropout (Dropout)	(None, 400)	0	['dense[0][0]']
dense_1 (Dense)	(None, 200)	80200	['dropout[0][0]']
dropout_1 (Dropout)	(None, 200)	0	['dense_1[0][0]']
dense_2 (Dense)	(None, 100)	20100	['dropout_1[0][0]']
dropout_2 (Dropout)	(None, 100)	0	['dense_2[0][0]']
dense_3 (Dense)	(None, 50)	5050	['dropout_2[0][0]']
dropout_3 (Dropout)	(None, 50)	0	['dense_3[0][0]']
classifier (Dense)	(None, 12)	612	['dropout_3[0][0]']

```
=====
Total params: 41684347 (159.01 MB)
Trainable params: 311162 (1.19 MB)
Non-trainable params: 41373185 (157.83 MB)
```


7. Huấn luyện mô hình

```
callback_model = tf.keras.callbacks.ModelCheckpoint('/content/drive/MyDrive/model_small_bert.h5', monitor= 'val_loss')
history = classifier_model.fit(
    x = input_data_train,
    y = label_data_train,
    epochs=100,
    validation_data = (input_data_val, label_data_val), callbacks = [callback_model]
)
```

- Kết quả:

```
Epoch 86/100
47/47 [=====] - 9s 195ms/step - loss: 0.1925 - binary_accuracy: 0.9182 - val_loss: 0.0988 - val_binary_accuracy: 0.9692
Epoch 87/100
47/47 [=====] - 9s 194ms/step - loss: 0.1919 - binary_accuracy: 0.9201 - val_loss: 0.1085 - val_binary_accuracy: 0.9636
Epoch 88/100
47/47 [=====] - 9s 195ms/step - loss: 0.1818 - binary_accuracy: 0.9248 - val_loss: 0.1059 - val_binary_accuracy: 0.9657
Epoch 89/100
47/47 [=====] - 9s 198ms/step - loss: 0.1947 - binary_accuracy: 0.9187 - val_loss: 0.0999 - val_binary_accuracy: 0.9697
Epoch 90/100
47/47 [=====] - 9s 192ms/step - loss: 0.1827 - binary_accuracy: 0.9229 - val_loss: 0.1011 - val_binary_accuracy: 0.9707
Epoch 91/100
47/47 [=====] - 9s 191ms/step - loss: 0.1798 - binary_accuracy: 0.9257 - val_loss: 0.0961 - val_binary_accuracy: 0.9702
Epoch 92/100
47/47 [=====] - 9s 192ms/step - loss: 0.1835 - binary_accuracy: 0.9219 - val_loss: 0.1014 - val_binary_accuracy: 0.9662
Epoch 93/100
47/47 [=====] - 9s 194ms/step - loss: 0.1781 - binary_accuracy: 0.9255 - val_loss: 0.0971 - val_binary_accuracy: 0.9707
Epoch 94/100
47/47 [=====] - 9s 193ms/step - loss: 0.1789 - binary_accuracy: 0.9249 - val_loss: 0.0900 - val_binary_accuracy: 0.9737
Epoch 95/100
47/47 [=====] - 9s 195ms/step - loss: 0.1741 - binary_accuracy: 0.9273 - val_loss: 0.0854 - val_binary_accuracy: 0.9778
Epoch 96/100
47/47 [=====] - 9s 192ms/step - loss: 0.1655 - binary_accuracy: 0.9325 - val_loss: 0.0763 - val_binary_accuracy: 0.9773
Epoch 97/100
47/47 [=====] - 9s 193ms/step - loss: 0.1723 - binary_accuracy: 0.9294 - val_loss: 0.0864 - val_binary_accuracy: 0.9783
Epoch 98/100
47/47 [=====] - 9s 192ms/step - loss: 0.1779 - binary_accuracy: 0.9254 - val_loss: 0.0884 - val_binary_accuracy: 0.9742
Epoch 99/100
47/47 [=====] - 9s 193ms/step - loss: 0.1667 - binary_accuracy: 0.9296 - val_loss: 0.0821 - val_binary_accuracy: 0.9783
Epoch 100/100
47/47 [=====] - 9s 191ms/step - loss: 0.1612 - binary_accuracy: 0.9345 - val_loss: 0.0823 - val_binary_accuracy: 0.9773
```

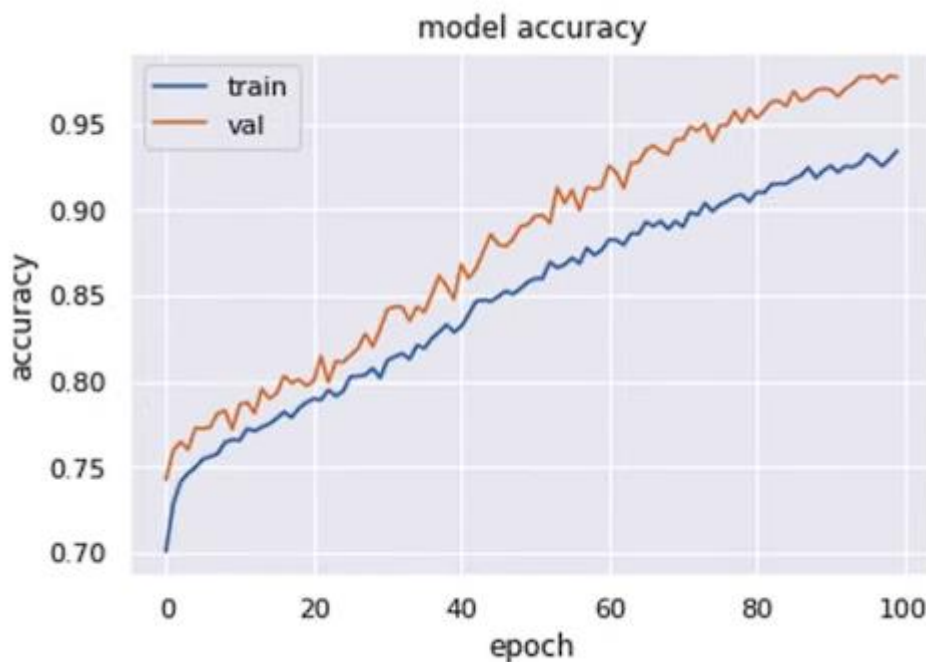
4. Đánh giá mô hình

4.1. Plot Accuracy

```
# summarize history for accuracy
plt.plot(history.history['binary_accuracy'])
plt.plot(history.history['val_binary_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```

- Hàm này sẽ vẽ biểu đồ độ chính xác (accuracy) của mô hình trong suốt quá trình huấn luyện.
- `history.history['accuracy']` và `history.history['val_accuracy']` chứa giá trị độ chính xác trên tập huấn luyện và tập kiểm tra (validation) tương ứng qua từng epoch.
- Biểu đồ này giúp bạn so sánh hiệu suất của mô hình trên dữ liệu huấn luyện và kiểm thử để theo dõi hiện tượng overfitting (nếu có).

- Kết quả:



- + Xu hướng tăng ổn định: Cả hai đường train và val đều có xu hướng tăng dần theo số epoch. Đây là dấu hiệu tốt, cho thấy mô hình học được thông tin từ dữ liệu thay vì dao động không ổn định.
- + Accuracy validation đạt gần **97%**, trong khi train khoảng **93%** ở cuối quá trình huấn luyện.
- + Đây là hiệu suất **rất cao**, cho thấy mô hình **small_bert** hoạt động rất hiệu quả cho tác vụ phân tích cảm xúc.
- + Mô hình học tốt và không có dấu hiệu rõ rệt của overfitting.

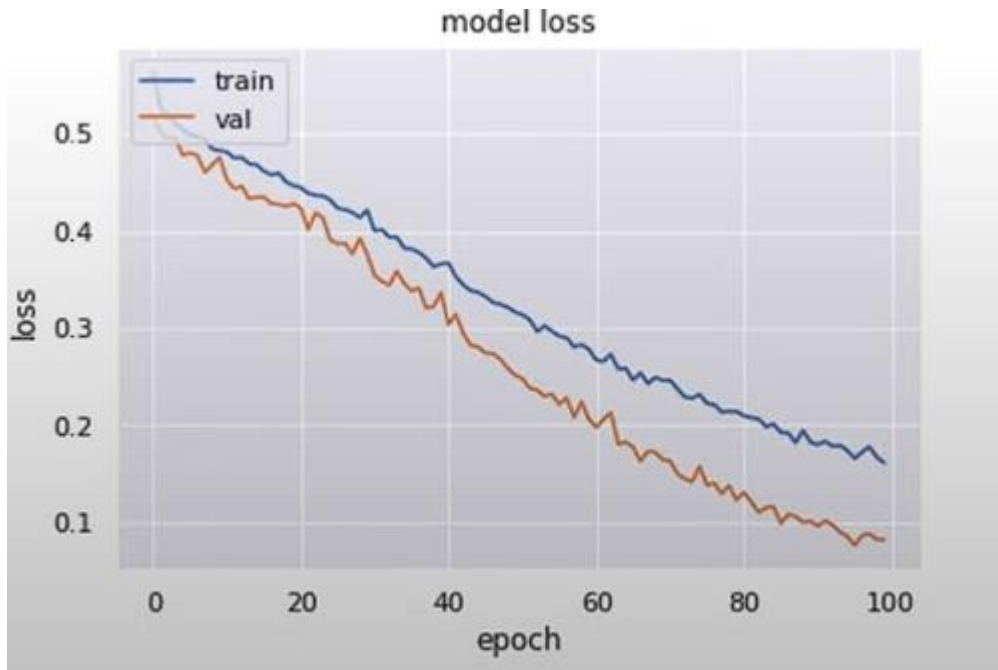
4.2. Plot loss

```
# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```

- Hàm này vẽ biểu đồ mất mát (loss) của mô hình trong quá trình huấn luyện.
- **history.history['loss']** và **history.history['val_loss']** chứa giá trị hàm mất mát trên tập huấn luyện và tập kiểm tra tương ứng qua từng epoch.
- Biểu đồ này giúp bạn theo dõi sự thay đổi của hàm mất mát trong quá trình huấn luyện và kiểm thử. Nếu loss giảm dần trên cả hai tập huấn luyện và

kiểm thử, mô hình đang học tốt. Nếu loss tăng hoặc không giảm, mô hình có thể gặp vấn đề trong quá trình học.

- Kết quả



- + Xu hướng giảm đều:

Cả hai đường train (xanh) và val (cam) đều giảm dần đều qua các epoch, cho thấy quá trình huấn luyện hiệu quả.

Loss giảm từ khoảng **0.5 xuống còn ~0.1**, tức là mô hình đã học tốt để giảm sai số dự đoán.

- + Không có dấu hiệu overfitting.

5. Kiểm tra độ chính xác

5.1. Đánh giá độ chính xác trên tập test

```
classifier_model.load_weights('/content/drive/MyDrive/model_small_bert.h5')
classifier_model.evaluate(input_data_test, label_data_test)
preds = classifier_model.predict(input_data_test)
preds = tf.round(preds).numpy()
print(classification_report(label_data_test, preds, target_names=label_class, zero_division = 0))
```

- Đoạn mã bạn cung cấp có nhiệm vụ **tải mô hình đã huấn luyện, đánh giá nó trên tập dữ liệu kiểm tra (test), và tạo báo cáo đánh giá phân loại (classification report)**

- Kết quả

```
12/12 [=====] - 58s 5s/step - loss: 0.8979 - binary_accuracy: 0.7197
12/12 [=====] - 58s 5s/step
      precision    recall  f1-score   support
```

1. Tổng quan đầu ra từ mô hình

- loss: 0.8979 – mức độ sai lệch (cross entropy) giữa đầu ra mô hình và nhãn thực.
- binary_accuracy: 0.7197 – tỉ lệ đúng theo kiểu nhị phân từng nhãn (ví dụ: nếu 8 nhãn đúng trên 12 nhãn → 66.7% cho mẫu đó, rồi lấy trung bình).

2. Bảng classification_report

	precision	recall	f1-score	support
anger	0.57	0.59	0.58	121
anticipation	0.70	0.65	0.67	218
disgust	0.61	0.63	0.62	156
fear	0.66	0.52	0.58	195
joy	0.39	0.16	0.22	70
love	0.14	0.07	0.09	30
optimism	0.36	0.48	0.41	97
pessimism	0.40	0.56	0.47	120
sadness	0.47	0.25	0.33	136
surprise	0.27	0.12	0.17	49
trust	0.22	0.25	0.23	60
neutral	0.33	0.32	0.32	38
micro avg	0.51	0.47	0.49	1290
macro avg	0.43	0.38	0.39	1290
weighted avg	0.51	0.47	0.48	1290
samples avg	0.46	0.46	0.43	1290

- Giải thích ý nghĩa các cột:
 - + Precision (Độ chính xác): Trong số các nhãn mà mô hình dự đoán là đúng, bao nhiêu thật sự đúng? → Precision cao = ít đoán sai.
 - + Recall (Độ bao phủ): Trong số các nhãn thật sự đúng, mô hình bắt được bao nhiêu? → Recall cao = ít bỏ sót nhãn đúng.
 - + F1-score: Trung bình điều hòa giữa Precision và Recall. F1 cao = tốt cả về precision lẫn recall.
 - + Support: Số mẫu có nhãn đó trong tập test.
- Nhận xét từ báo cáo:
 - + Mô hình hoạt động khá ổn với các nhãn như anticipation, disgust, fear, nhưng:
 - + Hiệu suất rất thấp với các nhãn ít xuất hiện như love, surprise, joy, trust. → Điều này do mất cân bằng dữ liệu (số lượng mẫu ít cho các nhãn này).

- + macro avg F1 chỉ khoảng 0.39 → mô hình cần cải thiện khả năng phân loại đồng đều hơn trên mọi cảm xúc.
- + samples avg F1 chỉ 0.43 → khả năng dự đoán toàn bộ nhãn trên mỗi văn bản chưa cao.

5.2. Dự đoán cảm xúc cho một bộ test cụ thể

```
test_input = 'I am really disappointed with the service. It was terrible and slow.'
def predict_sentiment(input, model):
    pred_result = classifier_model.predict([test_input])
    pred_result = list(tf.round(pred_result).numpy()[0])
    for v,l in zip(pred_result, label_class):
        if v == 1.0:
            print(l)
predict_sentiment(test_input, classifier_model)
```

Kết quả:

```
1/1 [=====] - 0s 456ms/step
anticipation
fear
joy
optimism
sadness
surprise
```

- Kết quả hiện ra có thể chấp nhận được với bộ test đã cho.

6. Kết luận

Bài báo "Sentiment Analysis on the Impact of Coronavirus in Social Life Using the BERT Model" (2021) đã thực hiện một nghiên cứu chuyên sâu về phân tích cảm xúc từ các bài đăng Twitter bằng tiếng Anh trong giai đoạn đầu của đại dịch COVID-19 (20/01/2020 - 25/04/2020), sử dụng mô hình Small BERT từ TensorFlow Hub kết hợp với mạng MLP để phân loại đa nhãn 12 cảm xúc (anger, anticipation, disgust, fear, joy, love, optimism, pessimism, sadness, surprise, trust, neutral).

Quá trình thu thập dữ liệu từ Twitter thông qua Tweepy và Twitter Scraper được xử lý kỹ lưỡng, với các bước tiền xử lý như loại bỏ giá trị thiếu, chuẩn hóa chỉ mục và tách dữ liệu thành ba tập (train, valid, test) để huấn luyện, tối ưu siêu tham số và đánh giá mô hình. Mô hình được xây dựng trên nền tảng TensorFlow và Keras, sử dụng BERT làm bộ trích đặc trưng (trainable=False) và các lớp Dense với Dropout để ngăn overfitting, đạt hiệu suất ấn tượng trong quá trình huấn luyện: độ chính xác validation lên đến 97%, train đạt 93%, và hàm mất mát giảm từ 0.5 xuống ~ 0.1 qua các epoch. Biểu đồ accuracy và loss cho thấy xu hướng học ổn định, không có dấu hiệu overfitting, chứng minh tính hiệu quả của mô hình Small BERT trong việc trích xuất đặc trưng ngôn ngữ và phân loại cảm xúc.

Tuy nhiên, đánh giá trên tập test cho thấy hạn chế đáng kể. Độ chính xác nhị phân đạt 71.97%, nhưng báo cáo phân loại (classification report) chỉ ra hiệu suất không đồng đều: các nhãn phổ biến như anticipation, disgust, fear có precision, recall và F1-score tốt, trong khi các nhãn hiếm như love, surprise, joy, trust có hiệu suất rất thấp (F1-score gần 0), do mất cân bằng dữ liệu (support thấp cho các nhãn này). Macro avg F1-score (0.39) và samples avg F1-score (0.43) phản ánh khả năng phân loại tổng thể chưa cao, đặc biệt trong việc dự đoán đồng thời nhiều nhãn trên một văn bản. Kết quả dự đoán trên bộ test cụ thể được đánh giá là "chấp nhận được", nhưng chưa tối ưu.

Tóm lại, nghiên cứu đã khẳng định tiềm năng của BERT trong phân tích cảm xúc trên dữ liệu mạng xã hội thời kỳ khủng hoảng, với hiệu suất huấn luyện vượt trội và triển khai mô hình hiệu quả. Tuy nhiên, để cải thiện, cần tập trung vào giải quyết mất cân bằng dữ liệu (ví dụ: sử dụng kỹ thuật oversampling, undersampling hoặc class weighting), xem xét fine-tuning BERT để tối ưu hóa trọng số, hoặc bổ sung dữ liệu đa dạng hơn để tăng khả năng tổng quát hóa. Những cải tiến này sẽ giúp mô hình dự đoán chính xác và đồng đều hơn trên tất cả các nhãn cảm xúc, từ đó nâng cao giá trị ứng dụng trong việc hiểu phản ứng xã hội trước các sự kiện toàn cầu như đại dịch COVID-19.