TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN

BÀI TIỂU LUẬN
HỌC MÁY (MACHINE LEARNING)

Đề tài:

Tìm hiểu mô hình LSTM và ứng dụng trong bài toán phát hiện hành vi bạo lực.

Tên nhóm: **NHÓM 01**

Nhóm sinh viên thực hiện:

1. Giáp Mạnh Tuyên (Trưởng nhóm)

2. Phạm Thành Long

3. Đặng Tuấn Linh

Giáo viên hướng dẫn: TS. Ngô Hữu Huy

Thái Nguyên, năm 2024

PHÂN CÔNG NHIỆM VỤ

Phân công nhiệm vụ theo tiến độ thực hiện:

Stt	Tên nhiệm vụ	Người thực hiện
1	Cơ sở lý thuyết	Giáp Mạnh Tuyên Phạm Thành Long Đặng Tuấn Linh
2	Tìm hiểu mô hình	Giáp Mạnh Tuyên Phạm Thành Long Đặng Tuấn Linh
3	Tìm hiểu bài toán	Giáp Mạnh Tuyên Đặng Tuấn Linh
4	Thu thập và xử lý dữ liệu	Phạm Thành Long Đặng Tuấn Linh
5	Lựa chọn và xây dựng mô hình	Giáp Mạnh Tuyên Phạm Thành Long
6	Huấn luyện và đánh giá mô hình	Giáp Mạnh Tuyên Phạm Thành Long
7	Tinh chỉnh và cải thiện mô hình	Giáp Mạnh Tuyên Phạm Thành Long
8	Hoàn thiện mô hình	Giáp Mạnh Tuyên Đặng Tuấn Linh
9	Viết báo cáo	Phạm Thành Long Đặng Tuấn Linh

Phân công nhiệm vụ theo thành viên thực hiện:

Stt	Thành Viên	Nhiệm vụ	Chữ ký
1	Phạm Thành Long	- Cơ sở lý thuyết - Tìm hiểu mô hình - Lựa chọn mô hình. - Thu thập và xử lý dữ liệu - Huấn luyện và đánh giá mô hình - Tinh chỉnh và cải thiện mô hình. - Viết báo cáo.	
2	Đặng Tuấn Linh	- Cơ sở lý thuyết - Tìm hiểu mô hình - Tìm hiểu bài toán - Thu thập và xử lý dữ liệu - Hoàn thiện mô hình - Viết báo cáo	
3	Giáp Mạnh Tuyên	- Cơ sở lý thuyết - Tìm hiểu mô hình - Tìm hiểu bài toán - Thu thập và xử lý dữ liệu - Huấn luyện và đánh giá mô hình - Tinh chỉnh và cải thiện mô hình - Hoàn thiện mô hình	

MỤC LỤC

MỞ ĐẦU

Công nghệ không chỉ là một phần của cuộc sống hàng ngày mà còn là minh chứng cho sự tiến bộ vượt bậc của loài người trong việc khám phá và khai thác sức mạnh của trí tuệ nhân tạo. Từ việc nhận diện hình ảnh cho đến dự đoán hành vi phức tạp, AI đã và đang trở thành một phần không thể thiếu trong cuộc sống, đóng vai trò quan trọng trong nhiều lĩnh vực như y tế, tài chính, công nghiệp và giáo dục.

Trong đó, một ứng dụng nổi bật của trí tuệ nhân tạo là khả năng phát hiện và phân tích hành vi bạo lực thông qua thị giác máy tính (computer vision). Thị giác máy tính, một lĩnh vực của AI, giúp máy tính không chỉ "nhìn thấy" mà còn "hiểu" nội dung của hình ảnh và video một cách tương tự như con người. Sự phát triển của thị giác máy tính đã mang đến tiềm năng và cơ hội lớn trong việc ứng dụng AI vào việc giám sát và đảm bảo an toàn xã hội.

Từ những công nghệ đơn giản như nhận diện các chuyển động bất thường, nhận dạng khuôn mặt đến những hệ thống phức tạp hơn như phân tích hành vi và dự đoán nguy cơ, thị giác máy tính đang mở ra những phương thức mới để phát hiện bạo lực và tăng cường an ninh. Sự kết hợp của các công nghệ học máy, xử lý hình ảnh và các phương pháp tiên tiến khác đã thúc đẩy sự phát triển nhanh chóng của lĩnh vực này, tạo nên những bước đột phá trong việc ứng dụng vào các lĩnh vực như an ninh, giáo dục và quản lý công cộng. Trong bối cảnh hiện nay, việc hiểu rõ và phát triển công nghệ phát hiện hành vi bạo lực trở thành một yếu tố quan trọng cho cả các nhà nghiên cứu lẫn các tổ chức, doanh nghiệp có nhu cầu triển khai giải pháp an ninh công nghệ cao.

Trong bài tiểu luận này, chúng em sẽ tìm hiểu về một mô hình học sâu LSTM để giải quyết bài toán phát hiện hành vi bạo lực trong các tình huống khác nhau. Đây là một chủ đề mới mẻ và đầy thách thức, chắc chắn không tránh khỏi những hạn chế và thiếu sót. Chúng em rất mong nhận được sự đóng góp ý kiến từ thầy cô và bạn đọc để bài tiểu luận này hoàn thiện hơn.

Chúng em xin chân thành cảm ơn!

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

1. Trí tuệ nhân tạo (AI)

1.1. Khái niệm

Trong khoa học máy tính, trí tuệ nhân tạo hay AI (tiếng Anh: artificial intelligence), là trí thông minh được thể hiện bằng máy móc, trái ngược với trí thông minh tự nhiên của con người. Thông thường, thuật ngữ "trí tuệ nhân tạo" thường được sử dụng để mô tả các máy móc chủ (hoặc máy tính) có khả năng bắt chước các chức năng "nhận thức" mà con người thường phải liên kết với tâm trí, như "học tập" và "giải quyết vấn đề".

Hình 1.1: Trí tuệ nhân tạo

Khi máy móc ngày càng tăng khả năng, các nhiệm vụ được coi là cần "trí thông minh" thường bị loại bỏ khỏi định nghĩa về AI, một hiện tượng được gọi là hiệu ứng AI. Một câu châm ngôn trong Định lý của Tesler nói rằng "AI là bất cứ điều gì chưa được thực hiện." Ví dụ, nhận dạng ký tự quang học thường bị loại trừ khỏi những thứ được coi là AI, đã trở thành một công nghệ thông thường. Khả năng máy hiện đại thường được phân loại như AI bao gồm thành công hiểu lời nói của con người, cạnh tranh ở mức cao nhất trong trò chơi chiến lược (chẳng hạn như cờ vua), xe hoạt động độc lập, định tuyến thông minh trong mạng phân phối nội dung, và mô phỏng quân sự

1.2. Sự phát triển của công nghệ trí tuệ nhân tạo

Trong bài báo chuyên đề của Alan Turing từ năm 1950, “Máy tính và trí tuệ”, ông đã xem xét vấn đề liệu máy móc có thể suy nghĩ hay không. Trong bài báo này, Turing lần đầu tiên đưa ra thuật ngữ trí tuệ nhân tạo và trình bày nó như một khái niệm

lý thuyết và triết học.

Từ năm 1957 đến năm 1974, sự phát triển của điện toán cho phép máy tính lưu trữ nhiều dữ liệu hơn và xử lý nhanh hơn. Trong giai đoạn này, các nhà khoa học đã phát triển thêm các thuật toán máy học (ML). Sự tiến bộ trong lĩnh vực này đã khiến các cơ quan như Cơ quan Chỉ đạo các Dự án Nghiên cứu Quốc phòng Tiên tiến (DARPA) tạo ra một quỹ cho nghiên cứu AI. Lúc đầu, mục tiêu chính của nghiên cứu này là khám phá xem máy tính có thể phiên âm và dịch ngôn ngữ nói hay không.

Trong suốt những năm 1980, có nguồn tài trợ được tăng cường và các nhà khoa học về bộ công cụ thuật toán mở rộng được sử dụng trong phát triển AI phù hợp. David Rumelhart và John Hopfield đã xuất bản các bài báo về kỹ thuật học sâu, cho thấy máy tính có thể học hỏi từ kinh nghiệm.

Từ năm 1990 đến đầu những năm 2000, các nhà khoa học đã đạt được nhiều mục tiêu cốt lõi của AI, như đánh bại nhà đương kim vô địch cờ vua thế giới. Với nhiều dữ liệu điện toán và khả năng xử lý trong thời đại hiện đại hơn so với những thập kỷ trước, nghiên cứu AI hiện nay trở nên phổ biến và dễ tiếp cận hơn. Nó nhanh chóng phát triển thành trí tuệ chung để phần mềm có thể thực hiện các nhiệm vụ phức tạp. Phần mềm có thể tự tạo, ra quyết định và tự học các nhiệm vụ mà trước đây chỉ giới hạn ở con người.

Trong thế kỷ 21, các kỹ thuật AI đã trải qua sự hồi sinh sau những tiến bộ đồng thời về sức mạnh máy tính, dữ liệu lớn và hiểu biết lý thuyết; và kỹ thuật AI đã trở thành một phần thiết yếu của ngành công nghệ, giúp giải quyết nhiều vấn đề thách thức trong học máy, công nghệ phần mềm và nghiên cứu vận hành.

1.3. Các loại trí tuệ nhân tạo

1.3.1. Phân loại AI dựa trên sự phát triển

a. ANI:

Artificial Narrow Intelligence hay còn gọi là Weak AI là một loại AI được lập trình để thực hiện một hoặc một số lượng giới hạn các công việc, thay vì sở hữu toàn bộ những khả năng nhận thức như não bộ con người. Vì chỉ tập trung vào một công việc nhất định, AI có thể thay thế, thậm chí vượt xa con người ở công việc đó.

Tuy nhiên, vì chỉ được lập trình cho một nhiệm vụ cụ thể, loại AI này chỉ có thể phân tích dữ liệu theo những gì nó đã được "huấn luyện" chứ không thể thực hiện những công việc mà bạn chưa hề dạy cho nó.

Cái tên Weak AI có thể khiến chúng ta nghi ngờ, đánh giá thấp khả năng của loại AI này. Thực chất, cho đến thời điểm này, đây là loại AI duy nhất được tạo ra và sử dụng trong thực tế. Ứng dụng của nó trong thực tế có rất nhiều:

- **Các trợ lý ảo như Siri**: AI phân loại dữ liệu và phản hồi các yêu cầu tìm kiếm rất nhanh.

Hình 1.2: Trợ lý ảo như Siri là một ứng dụng của ANI

- **Tính năng gợi ý**: Nếu bạn từng thắc mắc tại sao Netflix, Spotify lại "hiểu" mình đến vậy, có thể gợi ý cho mình những bộ phim, những bài hát hợp gu mình đến vậy, tất cả là nhờ AI hẹp phân tích những bộ phim, bài hát bạn đã xem trước đó để tìm cho bạn những thứ tương tự.
- **Chatbot**: Chatbot mô phỏng các cuộc hội thoại giữa khách hàng và các nhân viên chăm sóc khách hàng bằng cách trả lời những câu hỏi đơn giản một cách tự động thông qua text hay giọng nói

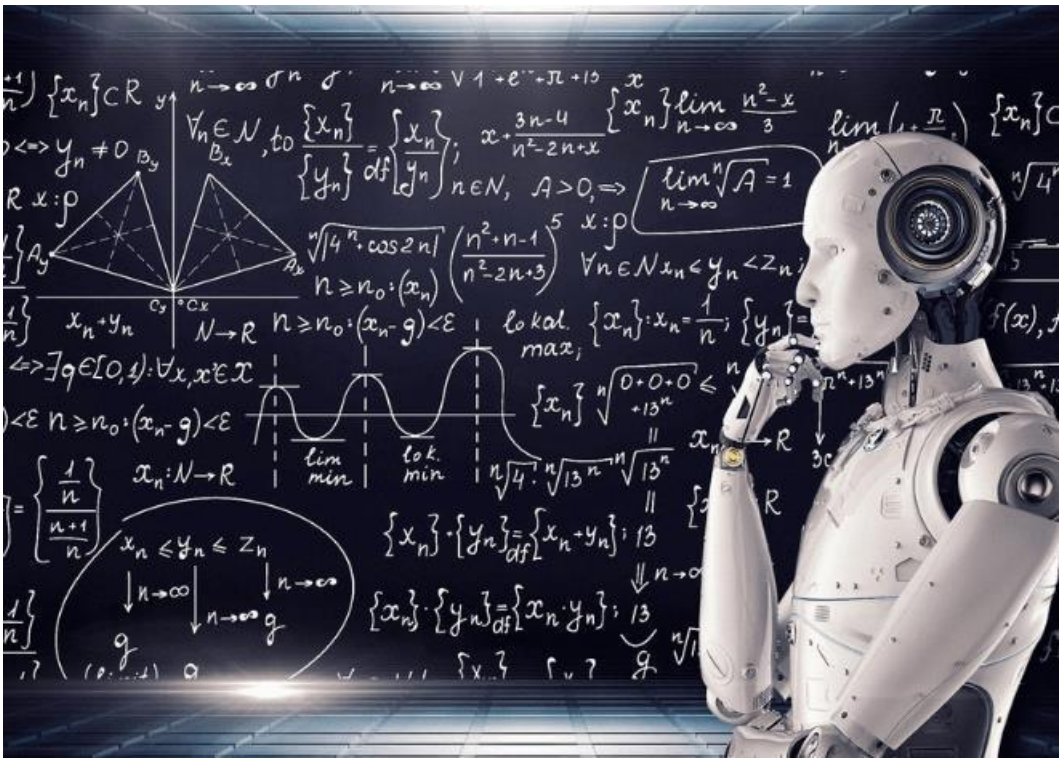
Hình 1.3: Chatbot cũng là một ứng dụng phổ biến của Weak AI

- **Xe tự lái**: Những sản phẩm xe tự lái như Tesla, thuyền hay các robot trong nhà máy đều sử dụng Weak AI. Tuy nhiên, thách thức của chúng là phải được lập trình để có thể tự lái trên những đoạn đường nguy hiểm.

Hình 1.4: Xe tự lái

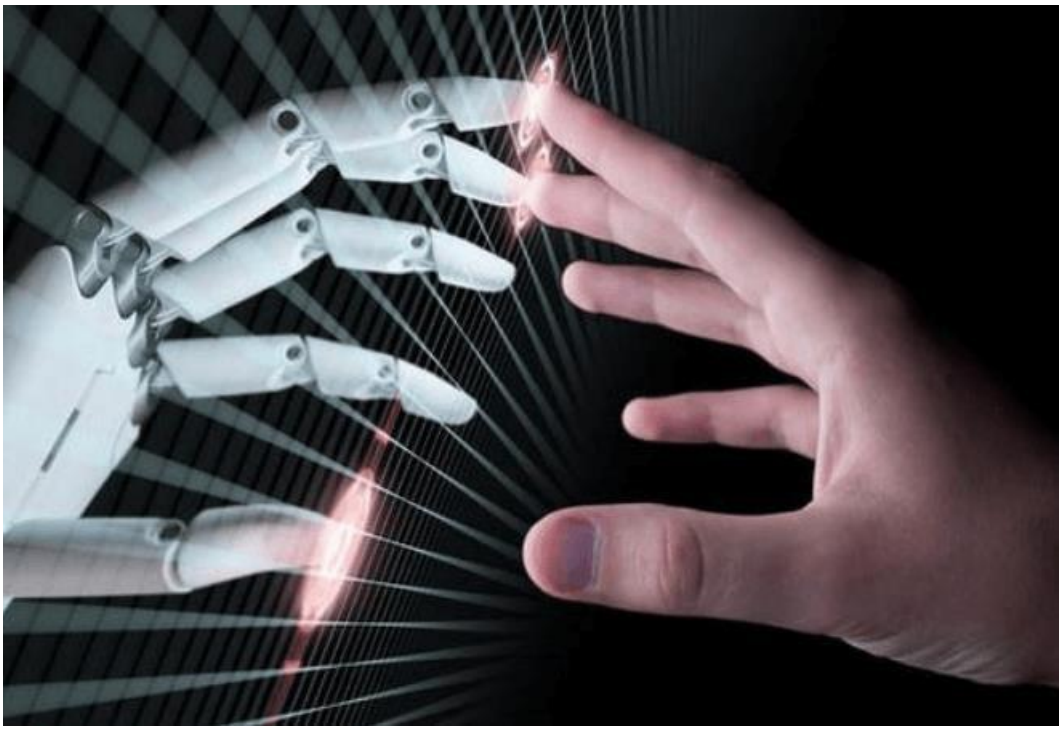
b. AGI

AGI nằm trong Strong AI, là một AI với trí thông minh sánh ngang với con người. Nghĩa là nó có ý thức tự nhận thức để giải quyết các vấn đề, học và lên kế hoạch cho tương lai. Nói khác đi, nó có thể hoàn toàn tự đưa ra các quyết định mà không cần phải được huấn luyện từ trước.



Hình 1.5: AGI

So với ANI, AGI chắc chắn phức tạp hơn nhiều. Và vì vậy, cho đến bây giờ, dù công nghệ phát triển không ngừng, chúng ta vẫn chưa thể tạo ra cỗ máy nào thực sự là AGI. Có thể nói, AGI vẫn là lý thuyết nằm trên những trang giấy. Tuy nhiên, chúng ta đang có những bước tiến và có thể tạo ra thứ được gọi là Partial AGI - AGI một phần.



Hình 1.6: Hiện tại chúng ta chưa đạt đến AGI

c. ASI

Sự phát triển của Trí tuệ nhân tạo có thể sẽ đánh dấu đỉnh cao của nghiên cứu AI, vì cho đến nay AGI sẽ trở thành dạng trí tuệ có khả năng nhất trên trái đất. ASI, ngoài việc tái tạo trí thông minh nhiều mặt của con người, sẽ cực kỳ tốt hơn trong mọi việc họ làm vì có bộ nhớ lớn hơn, xử lý và phân tích dữ liệu nhanh hơn và khả năng ra quyết định. Sự phát triển của AGI và ASI sẽ dẫn đến một kịch bản phổ biến nhất được gọi là điểm kỳ dị. Và trong khi tiềm năng sở hữu những cỗ máy mạnh mẽ như vậy có vẻ hấp dẫn, những cỗ máy này cũng có thể đe dọa sự tồn tại của chúng ta hoặc ít nhất, cách sống của chúng ta.

Tại thời điểm này, thật khó để hình dung tình trạng thế giới của chúng ta khi các loại AI tiên tiến hơn ra đời. Tuy nhiên, rõ ràng là còn một chặng đường dài để đạt được điều đó vì tình trạng phát triển hiện tại của AI so với những gì nó được dự đoán sẽ đi đến đâu vẫn còn ở giai đoạn sơ khai. Đối với những người có cái nhìn tiêu cực về tương lai của AI, điều này có nghĩa là bây giờ còn hơi sớm để lo lắng về điểm kỳ dị và vẫn còn thời gian để đảm bảo an toàn cho AI. Và đối với những người lạc quan về tương lai của AI, việc chúng ta chỉ mới sơ khai về sự phát triển của AI khiến tương lai thậm chí còn thú vị hơn.

1.3.2. Phân loại AI dựa trên sự tương đồng với con người

a. Công nghệ AI phản ứng

Công nghệ AI phản ứng là một loại AI được lập trình để phản ứng với các tình huống cụ thể trong một môi trường nhất định. Công nghệ này có thể giúp chúng ta giải quyết các vấn đề cụ thể và đạt được kết quả nhanh chóng. Công nghệ này được sử dụng rộng rãi trong các hệ thống tự động hóa, robot, hệ thống điều hành và các ứng dụng thương mại điện tử.

b. Công nghệ AI với bộ nhớ hạn chế

Công nghệ AI với bộ nhớ hạn chế là một loại AI được lập trình để lưu trữ và xử lý các thông tin trong một khoảng thời gian ngắn. Công nghệ AI này có thể giúp giảm thiểu thời gian xử lý và tăng cường hiệu suất làm việc. Người ta ứng dụng nó trong các hệ thống nhận dạng giọng nói, xử lý ngôn ngữ tự nhiên và các ứng dụng tương tác giọng nói.

c. Lý thuyết trí tuệ nhân tạo

Đây là một loại AI được lập trình để học hỏi và tự điều chỉnh các mô hình để giải quyết các vấn đề phức tạp. Nó có thể giúp chúng ta giải quyết các vấn đề phức tạp và tăng cường sự sáng tạo. Lý thuyết trí tuệ nhân tạo được sử dụng trong các hệ thống máy học, trích xuất thông tin và các ứng dụng tương tự.

d. Tự nhận thức

Loại AI này được lập trình để tự đánh giá và phát triển chính nó. Công nghệ tự nhận thức có thể giúp chúng ta phát triển các hệ thống AI độc lập và tăng cường khả năng học hỏi. Tự nhận thức được sử dụng rộng rãi trong các hệ thống tự chẩn đoán, tự động vận hành và các ứng dụng tương tự.

1.4. Ưu nhược điểm của trí tuệ nhân tạo

Khả năng xử lý dữ liệu lớn và phức tạp: Trí tuệ nhân tạo có thể xử lý hàng triệu dữ liệu trong thời gian ngắn, giúp chúng ta tìm ra những mối quan hệ và thông tin quan trọng từ các nguồn dữ liệu khác nhau.

Tăng cường hiệu suất làm việc: AI có thể thực hiện các tác vụ tốt hơn và nhanh hơn con người, giúp chúng ta tiết kiệm thời gian và năng lượng trong công việc.

Cải thiện chất lượng cuộc sống: Những ứng dụng của trí tuệ nhân tạo rất đa dạng, từ việc giúp tăng cường an ninh, dự đoán thời tiết, tìm kiếm thông tin, xử lý hình ảnh, đến việc cải thiện các sản phẩm và dịch vụ.

Tuy nhiên, AI chưa phải là công cụ hoàn thiện nhất. Nó vẫn còn tồn tại nhiều khuyết điểm, hạn chế và nhiều thách thức cho các nhà phát triển trong tương tai.

Thiếu tính xác thực và độ tin cậy: AI có thể bị lệch giá hoặc không chính xác nếu không được lập trình đúng cách hoặc không được giám sát chặt chẽ.

Thiếu sáng tạo và khả năng tương tác: AI có thể không thể sáng tạo và tương

tác giống như con người trong một số trường hợp.

Gây ra sự lo lắng về việc thay thế con người: AI có thể thay thế con người trong một số tác vụ, gây ra sự lo lắng về việc mất việc làm và sự ảnh hưởng đến sinh hoạt của con người.

2. Machine Learning cơ bản

2.1. Khái niệm

Máy học là một thuật ngữ đề cập đến các chương trình máy tính có khả năng học hỏi về cách hoàn thành các nhiệm vụ, đồng thời cải thiện hiệu suất theo thời gian.

Học máy là một thành phần quan trọng của lĩnh vực khoa học dữ liệu đang phát triển. Thông qua việc sử dụng phương pháp thống kê, các thuật toán được đào tạo để phân loại hoặc dự đoán và khám phá những thông tin chi tiết trong các dự án khai thác dữ liệu.

Những thông tin chi tiết này hỗ trợ, thúc đẩy việc đưa ra quyết định trong các ứng dụng, công cụ hỗ trợ doanh nghiệp, người dùng. Khi khối lượng dữ liệu tiếp tục mở rộng và phát triển, khả năng dự đoán, phân tích chính xác của máy học sẽ tăng lên.

Do cần có nguồn dữ liệu cực lớn để "học", máy học vẫn cần có sự tham gia của con người trong việc tìm hiểu dữ liệu cơ sở và lựa chọn các kĩ thuật phù hợp để phân tích thông tin, đánh giá mô hình. Đồng thời, trước khi sử dụng, dữ liệu phải được làm sạch, không có sai lệch và không có dữ liệu giả.

Trước đây, các thuật toán máy học chưa được tiếp cận với một lượng lớn dữ liệu đủ lớn để mô hình hóa mối quan hệ giữa các loại dữ liệu. Sự xuất hiện và phát triển của công nghệ Dữ liệu lớn (Big Data) đã cung cấp cho thuật toán machine learning lượng dữ liệu đủ lớn để cải thiện độ chính xác của mô hình và dự đoán.

2.2. Các phương pháp máy học

2.2.1. Máy học có giám sát

Học có giám sát được hiểu là cách sử dụng các tập dữ liệu được gắn nhãn để huấn luyện thuật toán phân loại hoặc dự đoán kết quả một cách chính xác.

Học tập có giám sát giúp các tổ chức giải quyết nhiều vấn đề trong thực tế trên quy mô lớn. Một số phương pháp được sử dụng trong học có giám sát bao gồm mạng nơ-ron, mô hình phân lớp (Naive bayes), hồi quy tuyến tính, hồi quy logistic, rừng ngẫu nhiên (Random forest) và máy hỗ trợ vectơ (SVM - support vector machine).

Thuật toán máy học có giám sát còn được tiếp tục chia nhỏ ra thành hai loại chính:

a. Classification (Phân loại)

Một bài toán được gọi là classification nếu các label của input data được chia

thành một số hữu hạn nhóm.

Ví dụ: Gmail xác định xem một email có phải là spam hay không; các hãng tín dụng xác định xem một khách hàng có khả năng thanh toán nợ hay không ?

b. Regression (Hồi quy)

Nếu label không được chia thành các nhóm mà là một giá trị thực cụ thể. Ví dụ: một căn nhà rộng x m^2, có y phòng ngủ và cách trung tâm thành phố z km sẽ có giá bao nhiêu ?

2.2.2. Máy học không giám sát

Học không giám sát, còn được gọi là học máy không giám sát, sử dụng các thuật toán học máy để phân tích và phân cụm các tập dữ liệu không được gắn nhãn. Các thuật toán này phát hiện ra các mẫu hoặc nhóm dữ liệu ẩn mà không cần sự can thiệp của con người.

Khả năng phát hiện ra những điểm tương đồng và khác biệt trong dữ liệu của phương pháp này khiến nó trở nên lý tưởng cho việc phân tích dữ liệu khám phá, chiến lược bán chéo, phân khúc khách hàng cũng như nhận dạng hình ảnh và mẫu.

Nó cũng được sử dụng để giảm số lượng các tính năng trong một mô hình thông qua quá trình giảm kích thước. Phân tích thành phần chính (PCA -Principal component analysis) và phân tích giá trị đơn lẻ (SVD - Singular value decomposition) là hai cách tiếp cận phổ biến cho nhiệm vụ này.

Các thuật toán khác được sử dụng trong học tập không giám sát bao gồm: mạng nơ-ron, phân cụm k-means và các phương pháp phân cụm theo xác suất.

Các bài toán máy học không giám sát được tiếp tục chia nhỏ thành hai loại:

a. Clustering (phân cụm)

Một bài toán phân nhóm toàn bộ dữ liệu X thành các nhóm nhỏ dựa trên sự liên quan giữa các dữ liệu trong mỗi nhóm.

Ví dụ: phân nhóm khách hàng dựa trên hành vi mua hàng. Điều này cũng giống như việc cho một đứa trẻ rất nhiều mảnh ghép với các hình thù và màu sắc khác nhau, ví dụ tam giác, vuông, tròn với màu xanh và đỏ, sau đó yêu cầu trẻ phân chúng thành từng nhóm. Mặc dù không cho trẻ biết mảnh nào tương ứng với hình nào hoặc màu nào, nhiều khả năng chúng vẫn có thể phân loại các mảnh ghép theo màu hoặc hình dạng.

b. Association

Là bài toán khi chúng ta muốn khám phá ra một quy luật dựa trên nhiều dữ liệu cho trước. Ví dụ: những khách hàng nam mua quần áo thường có xu hướng mua thêm đồng hồ hoặc thắt lưng, dựa vào đó tạo ra một hệ thống gợi ý khách hàng (Recommendation System), thúc đẩy nhu cầu mua sắm.

2.2.3. Học tập bán giám sát

Học tập bán giám sát là sự kết hợp giữa học tập có giám sát và không giám sát. Trong quá trình đào tạo, nó sử dụng một tập dữ liệu có nhãn nhỏ hơn học có giám sát để hướng dẫn phân loại, trích xuất tính năng từ một tập dữ liệu lớn hơn, không được gắn nhãn. Học bán giám sát có thể giải quyết vấn đề trong trường hợp không có đủ dữ liệu được gắn nhãn cho thuật toán học có giám sát.

Một ví dụ điển hình của nhóm này là chỉ có một phần ảnh hoặc văn bản được gán nhãn (ví dụ bức ảnh về người, động vật hoặc các văn bảnkhoa học, chính trị) và phần lớn các bức ảnh/văn bản khác chưa được gán nhãn được thu thập từ internet. Thực tế cho thấy rất nhiều các bài toán Machine Learning thuộc vào nhóm này vì việc thu thập dữ liệu có nhãn tốn rất nhiều thời gian và có chi phí cao. Rất nhiều loại dữ liệu thậm chí cần phải có chuyên gia mới gán nhãn được (ảnh y học chẳng hạn). Ngược lại, dữ liệu chưa có nhãn có thể được thu thập với chi phí thấp từ internet.

2.2.4. Học tăng cường (Reinforcement Learning)

Reinforcement learning là các bài toán giúp cho một hệ thống tự động xác định hành vi dựa trên hoàn cảnh để đạt được lợi ích cao nhất (maximizing the performance). Hiện tại, Reinforcement learning chủ yếu được áp dụng vào Lý Thuyết Trò Chơi (Game Theory), các thuật toán cần xác định nước đi tiếp theo để đạt được điểm số cao nhất.

Ví dụ: Trò chơi cờ vây AlphaGo. Cờ vây được xem là có độ phức tạp cực kỳ cao với tổng số nước đi xấp xỉ 10^{761} so với cờ vua là 10^{120}. Vì vậy thuật toán phải chọn ra một nước đi tối ưu trong số hàng nhiều tỉ lựa chọn. Về cơ bản, AlphaGo bao gồm các thuật toán thuộc cả Supervised learning và Reinforcement learning. Trong phần Supervised learning, dữ liệu từ các ván cờ do con người chơi với nhau được đưa vào để huấn luyện.

Tuy nhiên, mục đích cuối cùng của AlphaGo không phải là chơi như con người mà phải thậm chí thắng cả con người. Vì vậy, sau khi học xong các ván cờ của con người, AlphaGo tự chơi với chính nó với hàng triệu ván chơi để tìm ra các nước đi mới tối ưu hơn. Thuật toán trong phần tự chơi này được xếp vào loại Reinforcement learning.

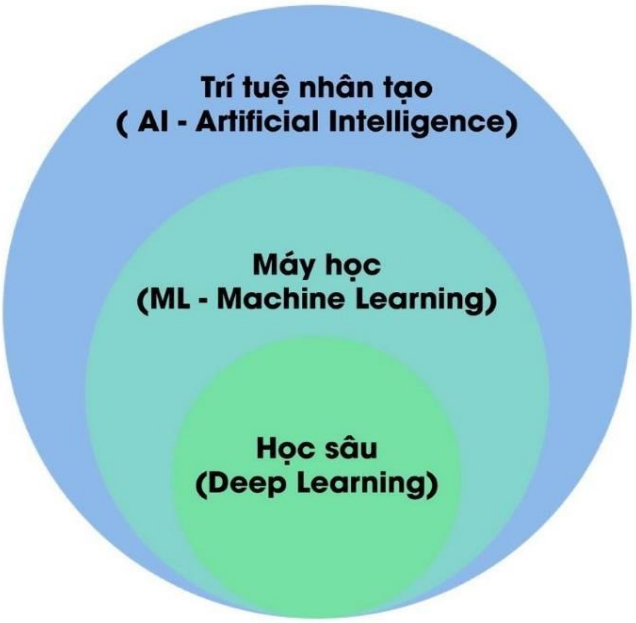
3. Deep Learning cơ bản

3.1. Khái niệm

Deep learning cơ bản là một tập hợp con của học máy, có khác biệt ở một số khía cạnh quan trọng so với công nghệ học máy truyền thống. Học sâu cho phép máy tính giải quyết rất nhiều vấn đề phức tạp bằng cách "học" từ một lượng lớn dữ liệu.

Học sâu thúc đẩy nhiều ứng dụng và dịch vụ trí tuệ nhân tạo (AI - Artificial Intelligent) nhằm nâng cao, cải thiện tính tự động hóa, thực hiện các tác vụ phân tích và vật lý mà không cần sự can thiệp của con người.

Công nghệ học sâu nằm sau các sản phẩm và dịch vụ hàng ngày (chẳng hạn như trợ lý kỹ thuật số, điều khiển TV hỗ trợ giọng nói từ xa, phát hiện gian lận thẻ tín dụng, ô tô tự lái,...).



Hình 1.4: Mối quan hệ giữa học sâu và máy học, trí tuệ nhân tạo

3.2. Phân biệt giữa học sâu và học máy

Các thuật toán học máy tận dụng dữ liệu có cấu trúc, được gắn nhãn để đưa ra dự đoán. Các tính năng cụ thể được xác định từ dữ liệu đầu vào cho mô hình và được tổ chức thành các bảng. Dữ liệu thường trải qua một số xử lý trước để sắp xếp thành một định dạng có cấu trúc.

Đối với học sâu, nó loại bỏ một số bước xử lý trước dữ liệu. Các thuật toán này có thể nhập và xử lý dữ liệu phi cấu trúc, như văn bản và hình ảnh. Đồng thời, chúng tự động hóa việc trích xuất tính năng, do đó, giảm bớt sự tham gia của con người trong một số công đoạn.

Ví dụ, có một bộ ảnh về các vật nuôi khác nhau cần phân loại. Các thuật toán học sâu có thể tự động xác định đặc điểm nào (ví dụ như tai) là quan trọng nhất để phân biệt từng loài động vật. Trong khi đó, để làm được điều này, học máy cần có sự hỗ trợ của con người.

Tiếp theo, thông qua các quá trình giảm độ dốc và lan truyền ngược, thuật toán học sâu tự điều chỉnh sao cho phù hợp, từ đó đưa ra dự đoán và phân biệt các loài động vật với độ chính xác cao.

Các mô hình học máy và học sâu cũng có khả năng thực hiện các kiểu học khác nhau, thường được phân loại là học có giám sát, học không giám sát và học bán giám sát.

4. Linear Regression

Linear regression là một mô hình thống kê đơn giản được sử dụng để dự đoán giá trị của một biến phụ thuộc dựa trên một hoặc nhiều biến độc lập. Nó giả định mối quan hệ tuyến tính giữa các biến độc lập và biến phụ thuộc. Trong mô hình linear regression, chúng ta cố gắng tìm ra một đường thẳng tốt nhất để phù hợp với dữ liệu.

Mô hình linear regression có dạng toán học như sau:

$$y = b_0 + w_1x_1 + w_2x_2 + \ldots + w_nx_n + \varepsilon$$

Trong đó:

- y: là biến phụ thuộc (biến dự đoán).
- $x1, x2, \ldots, xn$ là biến độc lập.
- $b0$: là hệ số chặn.
- $w1, w2, \ldots, wn$: là các hệ số của các biến độc lập.
- ε: là sai số ngẫu nhiên.

Trong linear regression, loss function (hàm mất mát) thường được chọn là Mean Squared Error (MSE), hay còn được gọi là Quadratic Loss. MSE là phổ biến và được sử dụng rộng rãi trong các bài toán hồi quy (regression), bao gồm cả linear regression.

Mục tiêu của linear regression là tìm ra các tham số w và b sao cho MSE là nhỏ nhất, tức là tìm ra đường thẳng (hoặc siêu phẳng trong không gian nhiều chiều) tốt nhất phù hợp với dữ liệu. Điều này thường được thực hiện bằng cách sử dụng phương pháp tối ưu hóa như Gradient Descent để tối thiểu hóa MSE.

Mô hình linear regression thường được sử dụng để dự đoán giá trị số, như dự đoán giá nhà dựa trên diện tích, dự đoán doanh số bán hàng dựa trên quảng cáo, hoặc bất kỳ tình huống nào có mối quan hệ tuyến tính giữa biến đầu vào và biến đầu ra.

5. Logistic Regression

Logistic regression là một mô hình thống kê được sử dụng để dự đoán xác suất của một biến phụ thuộc nhị phân dựa trên một hoặc nhiều biến độc lập. Mặc dù có từ "regression" trong tên, nhưng logistic regression thực ra là một thuật toán phân loại, chứ không phải là một phương pháp hồi quy (regression) như linear regression.

Trong logistic regression, chúng ta sử dụng một hàm sigmoid (hàm logistic) để ánh xạ đầu ra từ một phạm vi liên tục sang một phạm vi giữa 0 và 1, biểu thị xác suất của một biến nhị phân nhất định. Cụ thể, hàm sigmoid được biểu diễn như sau:

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

Trong đó σ là một biểu thức tuyến tính của các biến độc lập:

$$z = \beta_0 + \beta_1x_1 + \beta_2x_2 + \ldots + \beta_nx_n$$

Với:

- ***σ(x)***: là giá trị xác suất được dự đoán, thường là xác suất của lớp tích cực (positive class).
- ***x1, x2, . . . , xn*** là biến độc lập.
- ***β0***: Là hệ số chặn
- ***β1, β2, . . . , βn***: Là các hệ số của biến độc lập

Trong logistic regression, loss function thường được sử dụng là Cross-Entropy Loss, hay còn được gọi là Binary Cross-Entropy Loss, đặc biệt cho bài toán phân loại nhị phân.

Công thức của Binary Cross-Entropy Loss cho logistic regression được biểu diễn như sau:

$$\textbf{Binary CE} = -\frac{1}{N}\sum_{i=1}^{N}(y_i\ log\ log\ (y_{i\_pre}) + (1-y_i)log(1-y_{i\_pre}))$$

Trong đó:

- ***N***: là số lượng mẫu trong tập dữ liệu.
- ***yi***: là giá trị thực tế của mẫu thứ i.
- y_{i_pred}: là giá trị dự đoán của mẫu thứ i.

Mục tiêu của logistic regression là tối ưu hóa Cross-Entropy Loss để tìm ra các tham số tối ưu cho mô hình, thông qua các phương pháp tối ưu hóa như Gradient Descent hoặc các biến thể của nó.

Sau khi huấn luyện mô hình logistic regression, chúng ta sử dụng ngưỡng (threshold) để quyết định xác suất dự đoán là thuộc lớp tích cực (positive class) hay lớp tiêu cực (negative class). Thông thường, ngưỡng này được đặt ở giữa, tức là 0.5, nhưng có thể điều chỉnh để tối ưu hiệu suất của mô hình dựa trên các yêu cầu cụ thể của bài toán.

Logistic regression thường được sử dụng trong các bài toán phân loại nhị phân, như phân loại email spam hoặc dự đoán khả năng một khách hàng sẽ mua một sản phẩm nào đó.

6. Gradient descent

Gradient Descent là một thuật toán tối ưu hóa được sử dụng để cập nhật các tham số của một mô hình máy học như mạng neural, linear regression, logistic regression, và nhiều mô hình khác, dựa trên đạo hàm của một hàm mất mát (loss function) liên quan đến các tham số đó.

Ý tưởng chính của Gradient Descent là điều chỉnh các tham số của mô hình theo hướng ngược lại của đạo hàm của hàm mất mát, với một tỷ lệ học (learning rate) nhất định. Mục tiêu là di chuyển từ vị trí hiện tại trên bề mặt hàm mất mát đến điểm

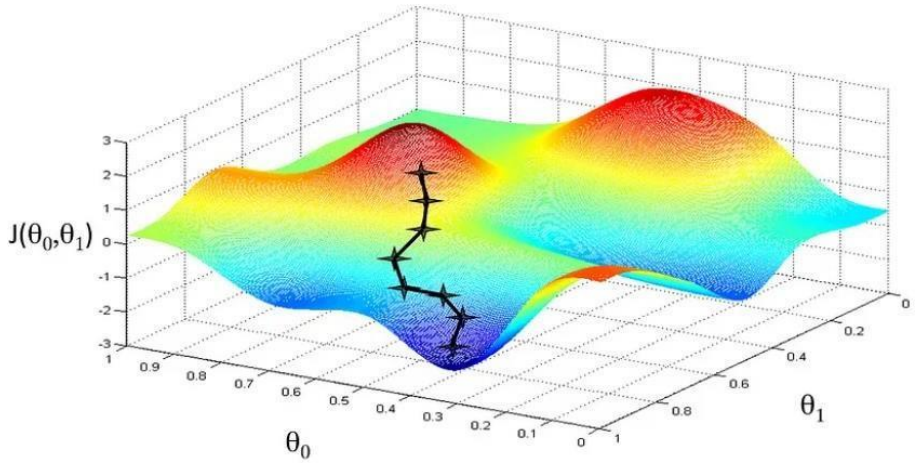
cực tiểu của nó, nơi hàm mất mát đạt được giá trị nhỏ nhất.

Công thức tổng quát của gradient descent:

$$\theta_j := \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j}$$

Trong đó:

- ***θj***: Tham số thứ j của mô hình.
- ***α***: Tỷ lệ học (learning rate).
- ***θj***: Hàm mất mát (loss function).
- $\frac{\partial J(\theta)}{\partial \theta j}$: Đạo hàm của hàm mất mát ***J(θ)*** theo tham số ***θj***

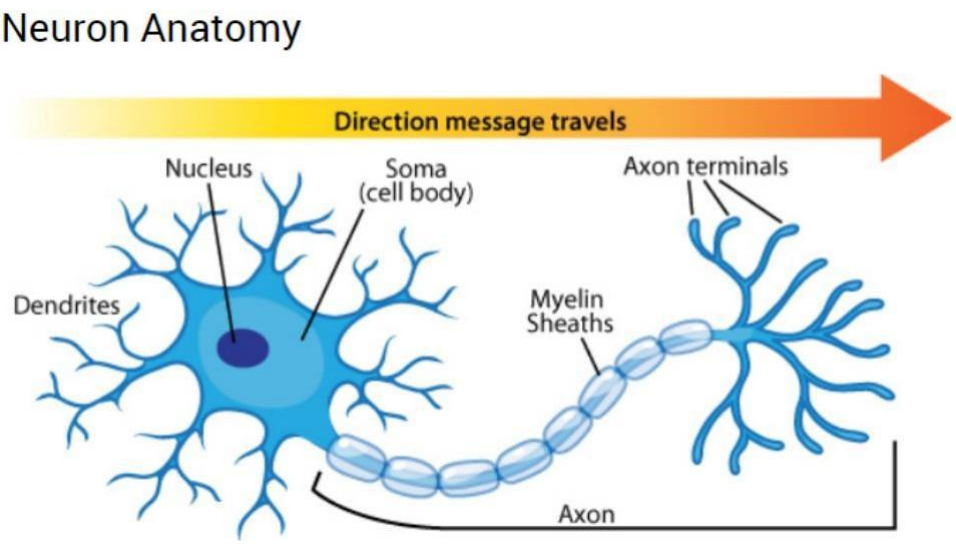


Hình 1.5: Gradient Descent

7. Neural Network

7.1. Neural Network là gì?

Mạng neural là một mô hình tính toán lấy cảm hứng từ cách mà não của con người hoạt động. Nó được sử dụng để giải quyết các vấn đề như phân loại, dự đoán và nhận dạng dữ liệu trong lĩnh vực trí tuệ nhân tạo và học máy. Phương thức này tạo ra một hệ thống thích ứng được máy tính sử dụng để học hỏi từ sai lầm của chúng và liên tục cải thiện.



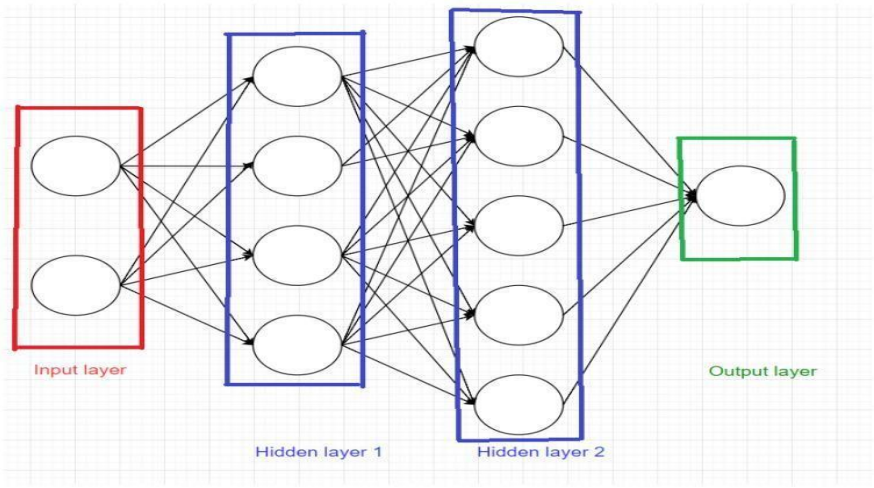
Hình 1.6: Gradient Descent

Neural Network có một số đặc điểm như:

- Mạng lưới nơ-ron nhân tạo hoạt động như nơ-ron trong não bộ con người. Trong đó mỗi nơ-ron là một hàm toán học, có chức năng thu thập và phân loại dữ liệu, thông tin theo cấu trúc chi tiết.
- Neural Network tương đồng với những phương pháp thống kê theo đồ thị đường cong hoặc phân tích hồi quy, bao hàm các nút mạng liên kết với nhau.
- Mỗi nút là một tập hợp tri giác, cấu tạo tương tự hàm hồi quy đa tuyến tính, được sắp xếp liên kết với nhau. Các lớp này sẽ thu thập thông tin, sau đó phân loại và phát tín hiệu đầu ra tương ứng.

Một mạng neural thường bao gồm nhiều "neuron" được tổ chức thành các lớp. Các lớp này bao gồm:

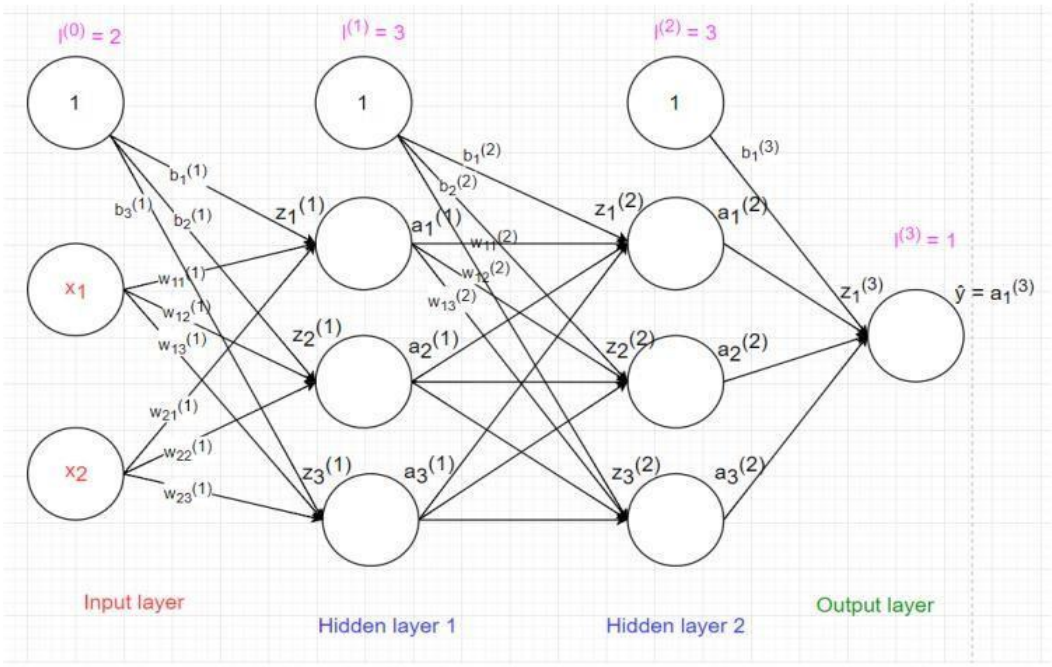
- Lớp nhập khẩu (input layer): Là lớp chứa dữ liệu đầu vào, ví dụ như các đặc trưng của hình ảnh, văn bản hoặc dữ liệu âm thanh.
- Các lớp ẩn (hidden layers): Đây là các lớp nằm giữa lớp đầu vào và lớp đầu ra. Mỗi lớp ẩn bao gồm nhiều neuron và là nơi xử lý thông tin. Số lượng và cấu trúc của các lớp ẩn có thể thay đổi tùy thuộc vào bài toán cụ thể và kiến trúc của mạng.
- Lớp đầu ra (output layer): Là lớp cuối cùng của mạng neural, chịu trách nhiệm cho việc xuất ra kết quả của mô hình.



Hình 1.7: Neural Network

Tổng số lớp của trong mạng neural: số layer – 1 (không tính input layer).

Các neuron trong mạng neural thường có cấu trúc đơn giản, bao gồm các trọng số và hàm kích hoạt. Trọng số được sử dụng để điều chỉnh sự ảnh hưởng của các đầu vào đến neuron, trong khi hàm kích hoạt xác định cách neuron phản ứng dựa trên tổng trọng số của các đầu vào.



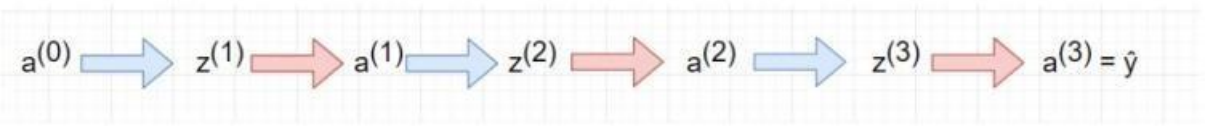
Hình 1.8: Neural Network

7.2. Feedforward

Feedforward là một khái niệm trong mạng neural mô tả việc truyền dữ liệu qua mạng từ lớp đầu vào đến lớp đầu ra mà không có các kết nối phản hồi (feedback) từ lớp đầu ra trở lại lớp đầu vào. Trong quá trình feedforward, dữ liệu được truyền qua

mạng neural theo một hướng duy nhất từ lớp nhập vào đến lớp đầu ra mà không có sự can thiệp từ bên ngoài.

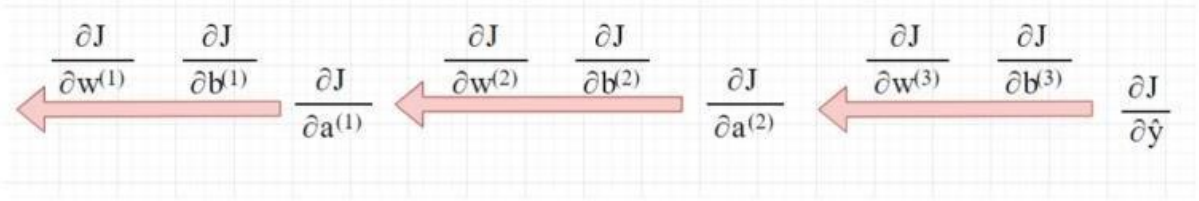
Quá trình feedforward: Dữ liệu được truyền qua mạng từ lớp đầu vào qua các lớp ẩn đến lớp đầu ra. Mỗi neuron trong mạng tính toán tổ hợp tuyến tính của các đầu vào của nó, sau đó áp dụng một hàm kích hoạt để tạo ra đầu ra của neuron. Quá trình này tiếp tục cho đến khi dữ liệu đến lớp đầu ra.



Hình 1.9: Feedforward

7.3. Backpropagation

Backpropagation là một phương pháp quan trọng trong quá trình huấn luyện mạng neural, được sử dụng để tính toán gradient của hàm mất mát theo các trọng số của mạng, từ lớp đầu ra trở lại lớp đầu vào. Quá trình backpropagation giúp mạng neural cập nhật các trọng số sao cho hàm mất mát được giảm thiểu.



Hình 1.10: Backpropagation

Thuật toán trên gồm hai quá trình lan truyền:

Quá trình lan truyền tiến (Forward Pass):

- Dữ liệu đầu vào được truyền qua mạng, từ lớp này sang lớp khác, để tạo ra đầu ra dự đoán.
- Đầu ra dự đoán được so sánh với đầu ra mục tiêu thực tế để tính toán sai số.

Quá trình lan truyền ngược (Backward Pass):

- Sai số được truyền ngược qua mạng để cập nhật các trọng số.
- Đối với mỗi trọng số trong mạng, đạo hàm riêng của sai số theo trọng số đó được tính toán.

7.4. Ứng dụng của Neural Network

Mạng Neuron được áp dụng trong nhiều lĩnh vực khác nhau như: Machine learning – máy học, tài chính, kinh doanh, lập kế hoạch mục tiêu, bảo trì sản phẩm, dự báo thời tiết, nghiên cứu tiếp thị, đánh giá rủi ro, phòng chống gian lận,…

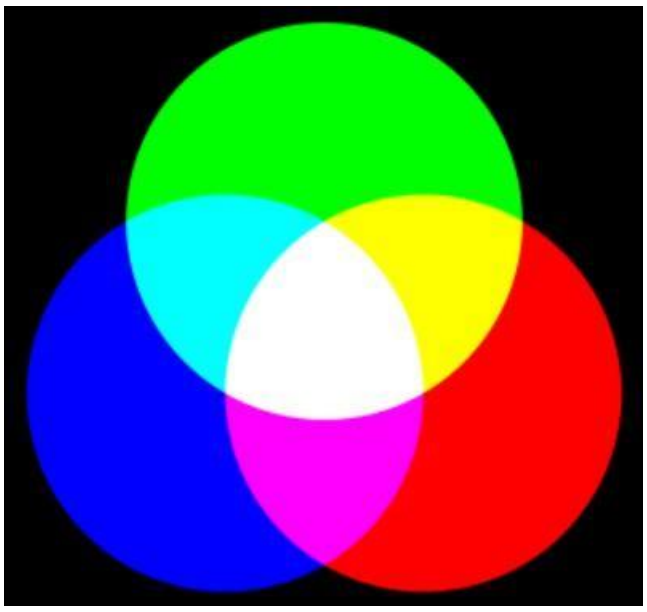
Cụ thể trong lĩnh vực tài chính, Neural Network hỗ trợ các tác vụ như: Thuật toán giao dịch, dự đoán chuỗi thời gian, phân loại thống kê chứng khoán, xây dựng mô hình giảm rủi ro tín dụng, thiết lập các chỉ báo độc quyền hoặc các công cụ kiểm soát giá cả.

Ngoài ra, nơ-ron nhân tạo còn được sử dụng để phân tích giao dịch dựa trên dữ liệu lịch sử. Tuy sự chính xác của các dự báo còn cần thời gian nghiên cứu và kiểm chứng, nhưng không thể phủ nhận vai trò của Neural network hiện nay.

8. Ảnh trong máy tính

8.1. Hệ màu RGB

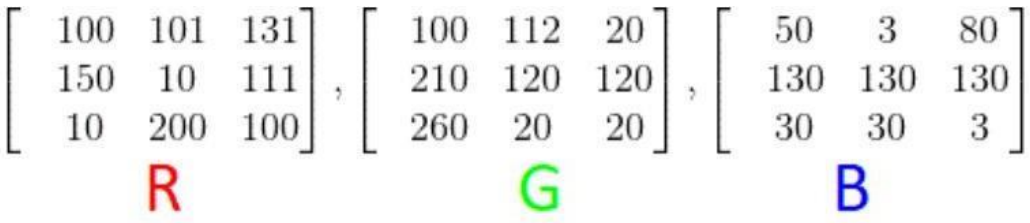
RGB viết tắt của Red (đỏ), Green (xanh lục), Blue (xanh lam), là ba màu chính của ánh sáng khi tách ra từ lăng kính. Khi trộn ba màu trên theo tỉ lệ nhất định có thể tạo thành các màu khác nhau.



Hình 1.11: Hệ màu RGB

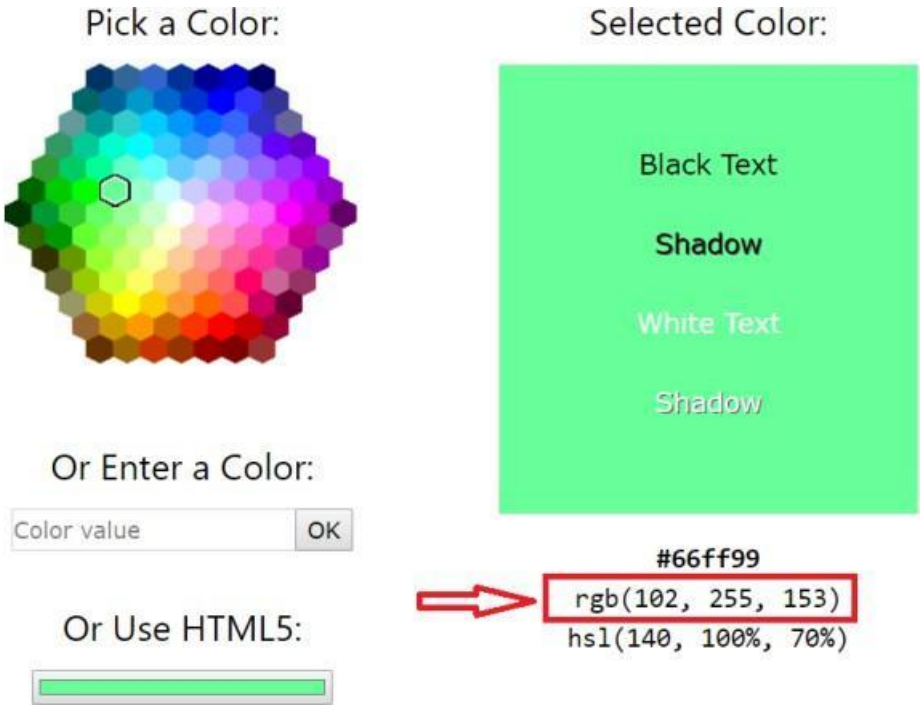
8.2. Ảnh màu

Trong máy tính, ảnh màu là loại ảnh được biểu diễn bằng các pixel mà mỗi pixel có thông tin về màu sắc của nó. Ảnh màu thường được biểu diễn bằng cách sử dụng các kênh màu, thường là các kênh đỏ (Red), xanh lá cây (Green), và xanh dương (Blue), còn được gọi là mô hình màu RGB.



Hình 1.12: Kênh màu

Trong mô hình màu RGB, mỗi pixel được biểu diễn bằng một tổ hợp của các giá trị màu RGB. Mỗi giá trị màu thường được biểu diễn bằng một con số từ 0 đến 255 (8-bit), trong đó 0 thường đại diện cho màu đen và 255 thường đại diện cho màu tương ứng đầy đủ (ví dụ: màu đỏ đầy đủ, màu xanh lá cây đầy đủ, và màu xanh dương đầy đủ).



Hình 1.13: Các cách chọn hệ màu

8.3. Ảnh xám

Mỗi pixel trong ảnh màu được biểu diễn bằng 3 giá trị (r,g,b) còn trong ảnh xám chỉ cần 1 giá trị x để biểu diễn. Nên ảnh xám trong máy tính được biểu diễn với một kênh màu. Mỗi giá trị màu thường được biểu diễn bằng một con số từ 0 đến 255.

$$\begin{bmatrix} 0 & 215 & \dots & 250 \\ 12 & 156 & \dots & 1 \\ \dots & \dots & \dots & \dots \\ 244 & 255 & \dots & 12 \end{bmatrix}$$

Hình 1.14: Biểu diễn màu bằng ma trận số

8.4. Chuyển hệ màu của ảnh

Mỗi pixel trong ảnh màu được biểu diễn bằng 3 giá trị (r,g,b) còn trong ảnh xám chỉ cần 1 giá trị x để biểu diễn.

Khi chuyển từ ảnh màu sang ảnh xám ta có thể dùng công thức:

$$\sigma = r * 0.299 + g * 0.587 + b * 0.114$$

Tuy nhiên khi chuyển ngược lại, bạn chỉ biết giá trị x và cần đi tìm r, g, b nên sẽ không chính xác.

CHƯƠNG 2: TÌM HIỂU VỀ MÔ HÌNH

1. Tổng quan về CNN

1.1. CNN (Convolutional Neural Network) là gì?

Convolutional Neural Network (CNNs – Mạng nơ-ron tích chập) là một trong những mô hình Deep Learning tiên tiến. Nó giúp cho chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao như hiện nay.

CNN được sử dụng nhiều trong các bài toán nhận dạng các object trong ảnh. Để tìm hiểu tại sao thuật toán này được sử dụng rộng rãi cho việc nhận dạng (detection), chúng ta hãy cùng tìm hiểu về thuật toán này.

1.2. Cấu trúc mạng CNN

Mạng CNN là một tập hợp các lớp Convolution chồng lên nhau và sử dụng các hàm nonlinear activation như ReLU và Tanh để kích hoạt các trọng số trong các node. Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo.

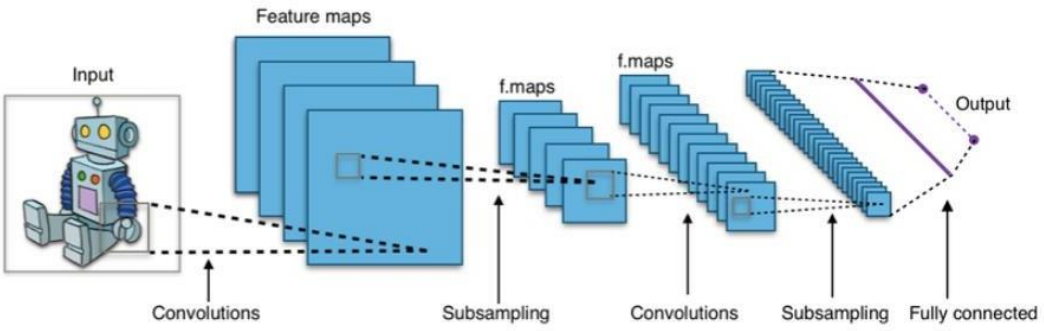
Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo. Trong mô hình mạng truyền thẳng (feedforward neural network) thì mỗi neural đầu vào (input node) cho mỗi neural đầu ra trong các lớp tiếp theo.

Mô hình này gọi là mạng kết nối đầy đủ (fully connected layer) hay mạng toàn vẹn (affine layer). Còn trong mô hình CNNs thì ngược lại. Các layer liên kết được với nhau thông qua cơ chế convolution.

Layer tiếp theo là kết quả convolution từ layer trước đó, nhờ vậy mà ta có được các kết nối cục bộ. Như vậy mỗi neuron ở lớp kế tiếp sinh ra từ kết quả của filter áp đặt lên một vùng ảnh cục bộ của neuron trước đó.

Mỗi một lớp được sử dụng các filter khác nhau thông thường có hàng trăm hàng nghìn filter như vậy và kết hợp kết quả của chúng lại. Ngoài ra có một số layer khác như pooling/subsampling layer dùng để chắt lọc lại các thông tin hữu ích hơn (loại bỏ các thông tin nhiễu).

Trong quá trình huấn luyện mạng (traning) CNN tự động học các giá trị qua các lớp filter dựa vào cách thức mà bạn thực hiện. Ví dụ trong tác vụ phân lớp ảnh, CNNs sẽ cố gắng tìm ra thông số tối ưu cho các filter tương ứng theo thứ tự raw pixel > edges > shapes > facial > high-level features. Layer cuối cùng được dùng để phân lớp ảnh.



Hình 2.1. Cấu trúc mạng CNN

Trong mô hình CNN có 2 khía cạnh cần quan tâm là tính bất biến (Location Invariance) và tính kết hợp (Compositionality). Với cùng một đối tượng, nếu đối tượng này được chiếu theo các góc độ khác nhau (translation, rotation, scaling) thì độ chính xác của thuật toán sẽ bị ảnh hưởng đáng kể.

Pooling layer sẽ cho bạn tính bất biến đối với phép dịch chuyển (translation), phép quay (rotation) và phép co giãn (scaling). Tính kết hợp cục bộ cho ta các cấp độ biểu diễn thông tin từ mức độ thấp đến mức độ cao và trừu tượng hơn thông qua convolution từ các filter.

Đó là lý do tại sao CNNs cho ra mô hình với độ chính xác rất cao.

Cũng giống như cách con người nhận biết các vật thể trong tự nhiên.

Mạng CNN sử dụng 3 ý tưởng cơ bản:

- Các trường tiếp nhận cục bộ (local receptive field).
- Trọng số chia sẻ (shared weights).
- Tổng hợp (pooling).

1.3. Phép tính Convolution

Kí hiệu phép tính convolution ($\otimes$), kí hiệu $\mathbf{Y = X \otimes W}$. Trong đó:

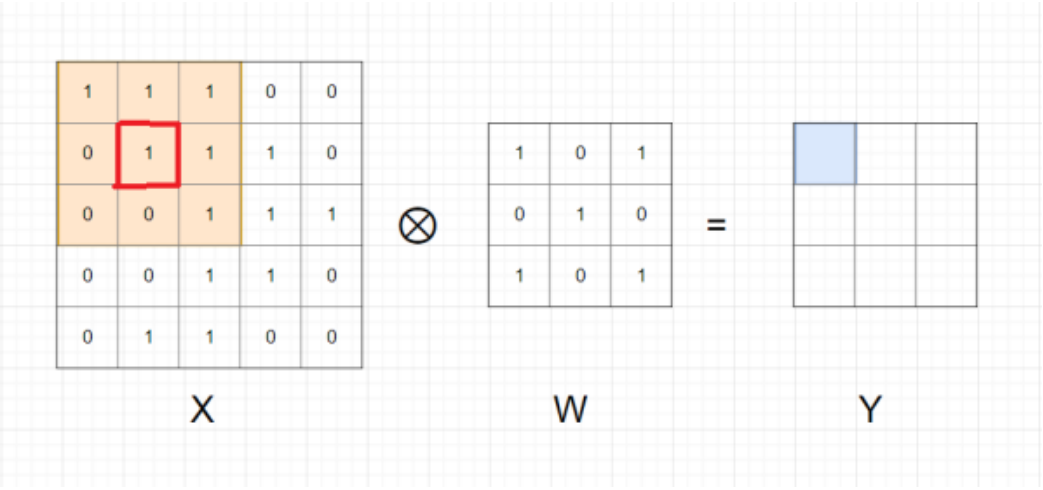
- **X**: là ma trận giá trị điểm ảnh
- **($\otimes$)**: là phép nhân element-wise
- **W**: là một kernel vuông kích thước k*k trong đó k là số lẻ có thể bằng 1,3,5,7,9,…

Phép tính convolution được thực hiện như sau:

Kernel được đặt lên ảnh gốc và di chuyển qua từng vị trí trên ảnh. Tại mỗi vị trí,

kernel được áp dụng lên một vùng của ảnh gốc, và tích các giá trị pixel tương ứng được tính toán. Sau khi tích các giá trị pixel được tính toán từ bước trên được tổng hợp lại để tạo ra giá trị mới cho pixel tại vị trí tương ứng trong ảnh kết quả. Quá trình này được lặp lại cho tất cả các vị trí trong ảnh gốc.

Ví dụ: tính giá trị y_{11}



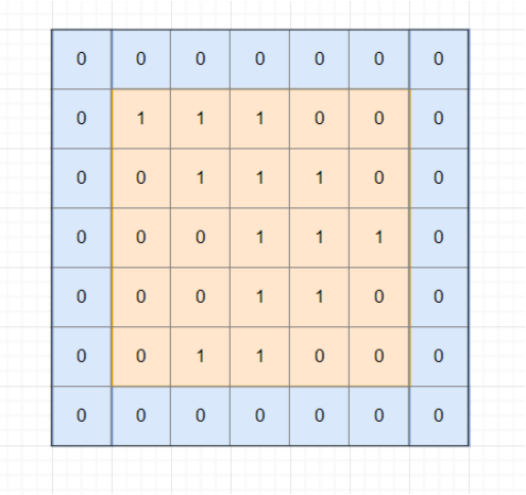
Hình 2.2. Ví dụ về phép tính Convolution

$$y_{11} = w_{11}*x_{11} + w_{12}*x_{12} + w_{13}*x_{13} + w_{21}*x_{21} + w_{22}*x_{22} + w_{23}*x_{23} + w_{31}*x_{31} + w_{32}*x_{32} + w_{33}*x_{33}$$

Mục đích của phép tính convolution: Mục đích của phép tính convolution trên ảnh là làm mờ, làm nét ảnh; xác định các đường;...Mỗi kernel khác nhau thì sẽ phép tính convolution sẽ có ý nghĩa khác nhau.

1.4. Padding

Padding là một kỹ thuật được sử dụng để thay đổi kích thước của đầu vào bằng cách thêm các giá trị (thường là giá trị 0) vào xung quanh biên của ảnh hoặc đầu vào của mạng neural. Mục tiêu chính của việc sử dụng padding là để duy trì kích thước của đầu vào qua các lớp xử lý mà không làm giảm kích thước của đầu ra.



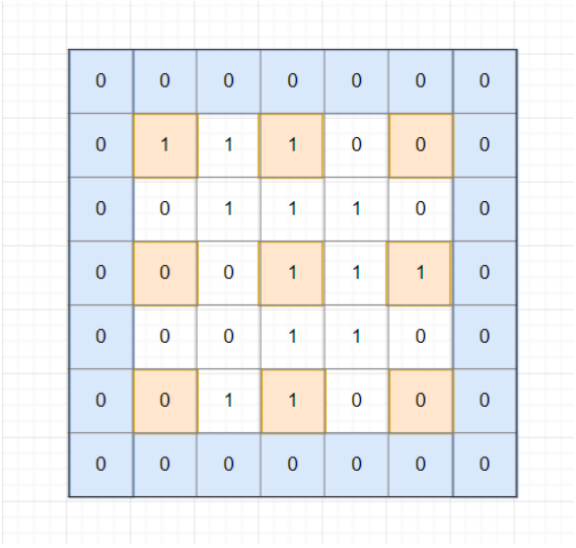
Hình 2.3. Padding

1.5. Stride

Stride định nghĩa bước nhảy của cửa sổ (window) hoặc bộ lọc (filter) khi di chuyển trên đầu vào (input) để thực hiện phép tính convolution hoặc pooling.

Khi áp dụng stride trong phép tính convolution hoặc pooling, cửa sổ hoặc bộ lọc không di chuyển mỗi pixel một cách liên tục, mà sẽ nhảy qua một số pixel cố định được xác định bởi stride. Giá trị stride được xác định trước và thường được chọn sao cho đảm bảo kích thước của đầu ra (output) là phù hợp với yêu cầu của mô hình.

Ví dụ: ta có ma trận X (hình bên dưới) với padding = 1, stride = 2.



Hình 2.4. Stride

Hiểu đơn giản là bắt đầu từ vị trí x11 sau đó nhảy k bước theo chiều dọc và ngang cho đến hết ma trận X.

Kích thước của ma trận Y là 3*3 đã giảm đi đáng kể so với ma trận X.

Công thức tổng quát cho phép tính convolution của ma trận X kích thước m*n với kernel kích thước k * k ,stride = s, padding = p ra ma trận Y kích thước:

$$(\frac{m-k+2p}{s} + 1)*(\frac{n-k+2p}{s} + 1)$$

Trong đó:

- **m**,**n**: là kích thước của đầu vào.
- **k**: là kích thước của bộ lọc.
- **p**: là padding.
- **s**: là stride.

Stride thường dùng để giảm kích thước của ma trận sau phép tính convolution.

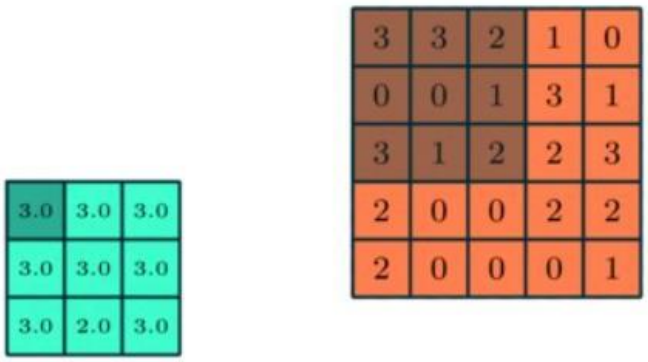
1.6. Pooling layer

Pooling layer thường được dùng giữa các convolutional layer, để giảm kích thước dữ liệu nhưng vẫn giữ được các thuộc tính quan trọng. Việc giảm kích thước dữ liệu giúp giảm các phép tính toán trong model.

Bên cạnh đó, với phép pooling kích thước ảnh giảm, do đó lớp convolution học được các vùng có kích thước lớn hơn. Ví dụ như ảnh kích thước 224*224 qua pooling về 112*112 thì vùng 3*3 ở ảnh 112*112 tương ứng với vùng 6*6 ở ảnh ban đầu. Vì vậy qua các pooling thì kích thước ảnh nhỏ đi và convolutional layer sẽ học được các thuộc tính lớn hơn.

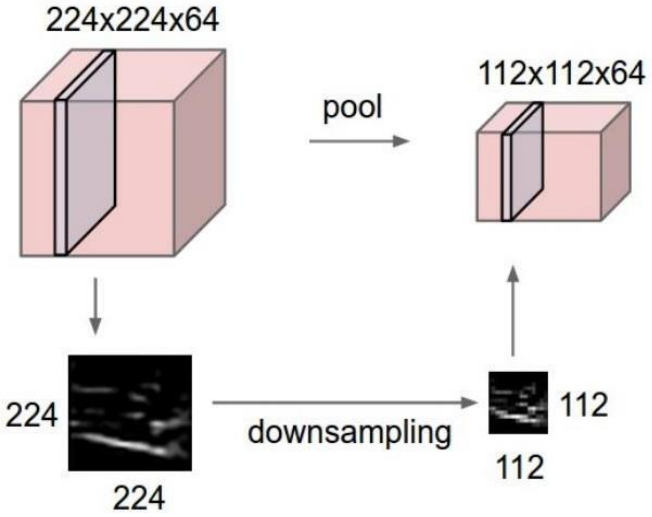
Gọi pooling size kích thước K*K. Input của pooling layer có kích thước H*W*D, ta tách ra làm D ma trận kích thước H*W. Với mỗi ma trận, trên vùng kích thước K*K trên ma trận ta tìm maximum hoặc average của dữ liệu rồi viết vào ma trận kết quả. Quy tắc về stride và padding áp dụng như phép tính convolution trên ảnh.

Ví dụ: max pooling layer với size= (3x3), stride=1, padding=0.



Hình 2.5. Max Pooling

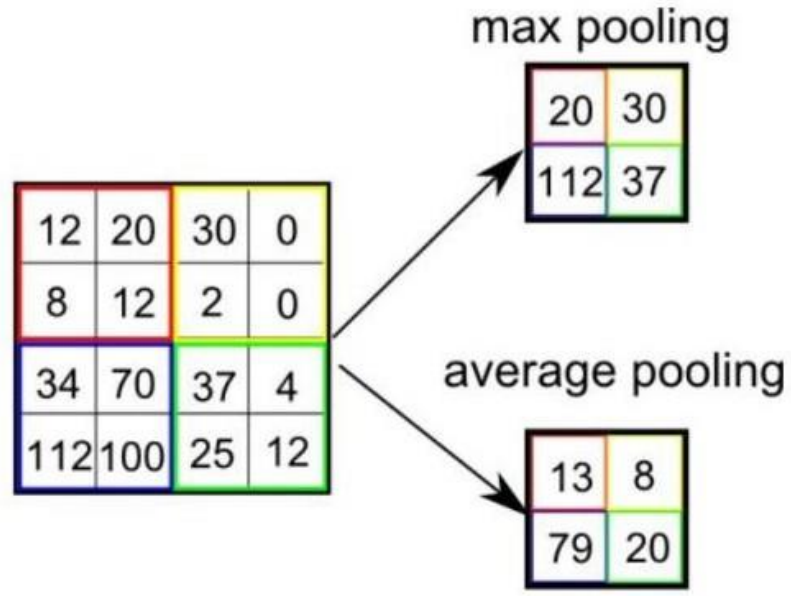
Nhưng hầu hết khi dùng pooling layer thì sẽ dùng size=(2,2), stride=2, padding=0. Khi đó output width và height của dữ liệu giảm đi một nửa, depth thì được giữ nguyên.



Hình 2.6 Pooling

Có 2 loại pooling layer phổ biến là:

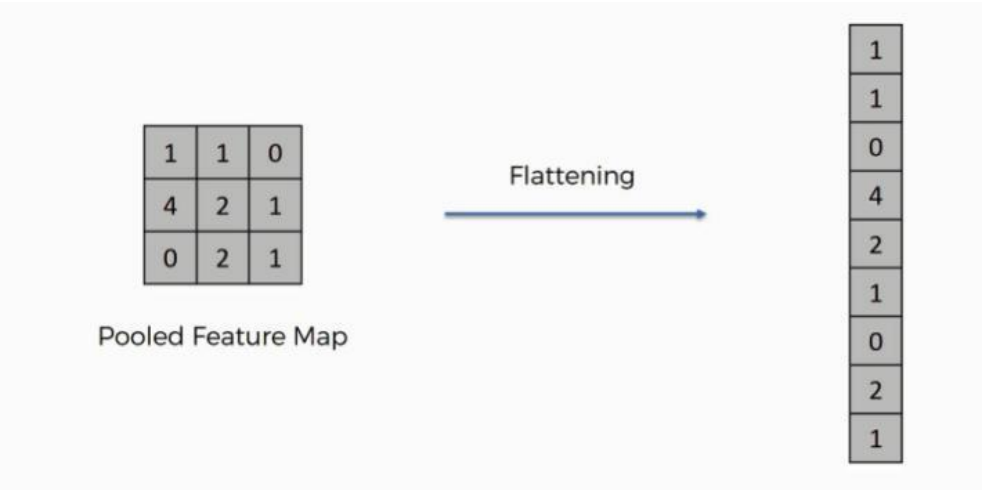
- Max pooling
- Average pooling



Hình 2.7. Max Pooling và Average Pooling

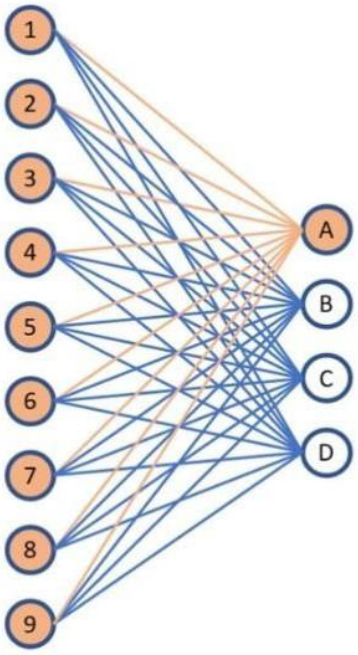
1.7. Fully connected layer

Sau khi ảnh được truyền qua nhiều convolutional layer và pooling layer thì model đã học được tương đối các đặc điểm của ảnh (ví dụ mắt, mũi, khung mặt,...) thì tensor của output của layer cuối cùng, kích thước H*W*D, sẽ được chuyển về 1 vector kích thước (H*W*D, 1).



Hình 2.8. Flattening

Sau đó ta dùng các fully connected layer để kết hợp các đặc điểm của ảnh để ra được output của model.



Hình 2.9. Fully Connected Layer

2. Mạng hồi quy

2.1. RNN

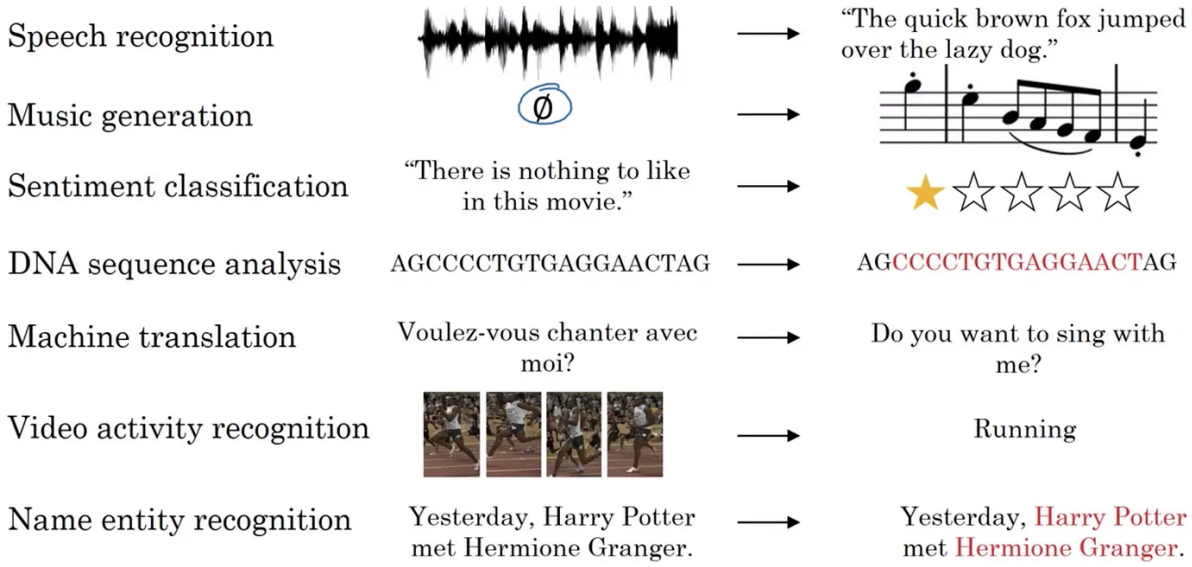
Bài toán: Cần phân loại hành động của người trong video, input là video 30s, output là phân loại hành động, ví dụ: đứng, ngồi, chạy, đánh nhau, bắn súng,... Khi xử lý video ta hay gặp khái niệm FPS (frame per second) tức là bao nhiêu frame (ảnh) mỗi giây. Ví dụ 1 FPS với video 30s tức là lấy ra từ video 30 ảnh, mỗi giây một ảnh để xử lý. Ta dùng 1 FPS cho video input ở bài toán trên, tức là lấy ra 30 ảnh từ video, ảnh 1 ở giây 1, ảnh 2 ở giây 2,... ảnh 30 ở giây 30. Bây giờ input là 30 ảnh: ảnh 1, ảnh 2,... ảnh 30 và output là phân loại hành động. Nhận xét: Các ảnh có thứ tự: ảnh 1 xảy ra trước ảnh 2, ảnh 2 xảy ra trước ảnh 3,... Nếu ta đảo lộn các ảnh thì có thể thay đổi nội dung của video. Ví dụ: nội dung video là cảnh bắn nhau, thứ tự đúng là A bắn trúng người B và B chết, nếu ta đảo thứ tự ảnh thành người B chết xong A mới bắn thì rõ ràng bây giờ A không phải là kẻ giết người => nội dung video bị thay đổi. Ta có thể dùng CNN để phân loại 1 ảnh trong 30 ảnh trên, nhưng rõ ràng là 1 ảnh không thể mô tả được nội dung của cả video. Ví dụ: Cảnh người cướp điện thoại, nếu ta chỉ dùng 1 ảnh là người đấy cầm điện thoại lúc cướp xong thì ta không thể biết được cả hành động cướp.

Do đó cần một mô hình mới có thể giải quyết được bài toán với input là sequence (chuỗi ảnh 1->30). Từ nhu cầu đó Recurrent Neural Network (RNN) hay mạng nơ ron hồi quy ra đời.

2.1.1. Dữ liệu dạng sequence

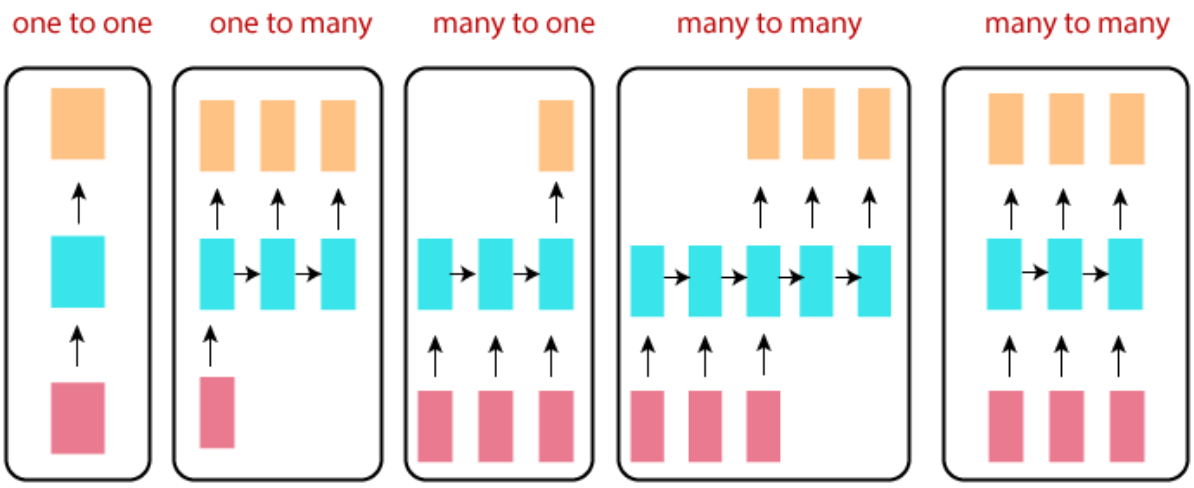
Dữ liệu có thứ tự như các ảnh tách từ video ở trên được gọi là sequence, time-series data. Trong bài toán dự đoán đột quỵ tim cho bệnh nhân bằng các dữ liệu tim mạch khám trước đó. Input là dữ liệu của những lần khám trước đó, ví dụ i1 là lần khám tháng 1, i2 là lần khám tháng 2,... i8 là lần khám tháng 8. (i1,i2,..i8) được gọi là sequence data. RNN sẽ học từ input và dự đoán xem bệnh nhân có bị đột quy tim hay không.

Ví dụ khác là trong bài toán dịch tự động với input là 1 câu, ví dụ "tôi yêu Việt Nam" thì vị trí các từ và sự xắp xếp cực kì quan trọng đến nghĩa của câu và dữ liệu input các từ [’tôi’, ’yêu’, ’việt’, ’nam’] được gọi là sequence data. Trong bài toán xử lý ngôn ngữ (NLP) thì không thể xử lý cả câu được và người ta tách ra từng từ (chữ) làm input, giống như trong video người ta tách ra các ảnh (frame) làm input.



Hình 2.10. Dữ liệu dạng sequence.

2.1.2. Phân loại bài toán RNN



Hình 2.11. Các mô hình bài toán RNN.

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

2.1.3. Ứng dụng bài toán RNN

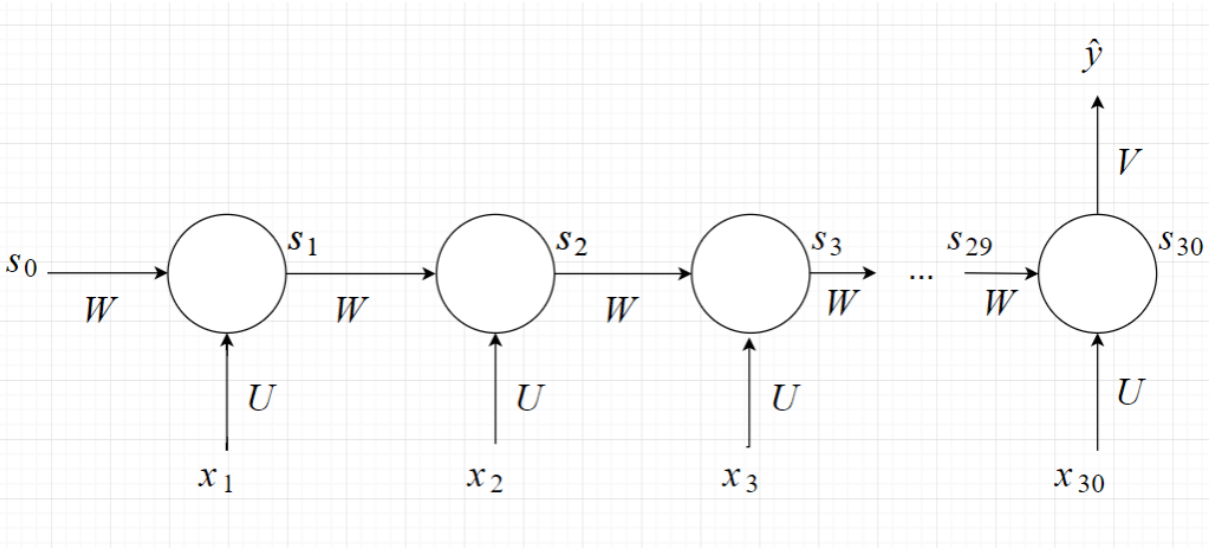
Về cơ bản nếu bạn thấy sequence data hay time-series data và bạn muốn áp dụng deep learning thì bạn nghĩ ngay đến RNN. Dưới đây là một số ứng dụng của RNN:

- Speech to text: Chuyển giọng nói sang text.
- Sentiment classification: Phân loại bình luận của người dùng, tích cực hay tiêu cực.
- Machine translation: Bài toán dịch tự động giữa các ngôn ngữ.

- Video recognition: Nhận diện hành động trong video.
- Heart attack: Dự đoán đột quỵ tim.

2.1.4. Mô hình bài toán RNN

Với input là dạng sequence data (đoạn văn bản) và output có dạng là vector có kích thước d*1 (d là các mức độ cảm xúc cần phân loại), softmax function được sử dụng.

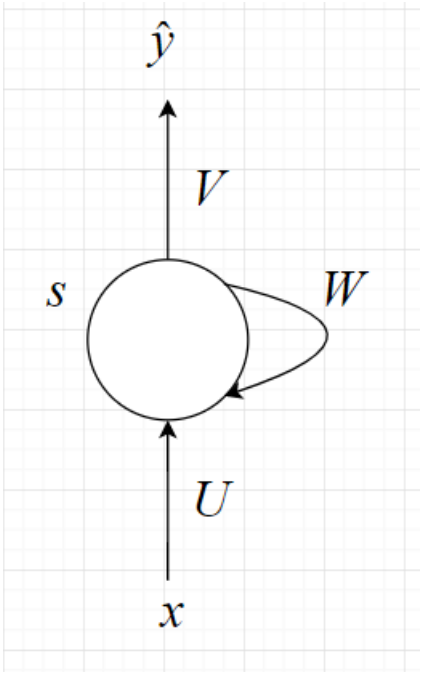


Hình 2.12. Mô hình RNN cho bài toán.

Ta có:

- Mô hình có n input (số lượng từ) và 1 output, các input được cho vào model đúng với thứ tự từ trong câu (sentence): x1, x2,...xn.
- Mỗi hình tròn được gọi là 1 state, state t có input là xt và st−1 (output của state trước); output là st = f(U ∗ xt +W ∗ st−1). f là activation function thường là Tanh hoặc ReLU.
- Có thể thấy st mang cả thông tin từ state trước (st−1) và input của state hiện tại => st giống như memory nhớ các đặc điểm của các input từ x1 đến xt s0 được thêm vào chỉ cho chuẩn công thức nên thường được gán bằng 0 hoặc giá trị ngẫu nhiên. Có thể hiểu là ban đầu chưa có dữ liệu gì để học thì memory rỗng.
- Do ta chỉ có 1 output, nên sẽ được đặt ở state cuối cùng, khi đó s30 học được thông tin từ tất cả các input. yˆ = g(V ∗ s30). g là activation function, trong bài này là bài toán phân loại nên sẽ dùng softmax.
- Ta thấy là ở mỗi state các hệ số W, U là giống nhau nên model có thể được

viết lại thành:



Hình 2.13. Mô hình RNN rút gọn.

Tóm lại:

- xi là vector có kích thước n*1, si là vector có kích thước m*1, yi là vector có kích thước d*1. U là ma trận có kích thước m*n, W là ma trận có kích thước m*m và V là ma trận có kích thước d*m.
- s0 = 0,st = f(U ∗ xt +W ∗ st−1) với t >= 1
- yˆ = g(V ∗ s30)

2.1.5. Backpropagation through time

Có 3 tham số ta cần phải tìm là W, U, V. Để thực hiện gradient descent, ta cần tính: $\frac{\partial L}{\partial W}$, $\frac{\partial L}{\partial U}$, $\frac{\partial L}{\partial V}$.

Tính đạo hàm với V khá đơn giản:

$$\frac{\partial L}{\partial V} = \frac{\partial L}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial V}$$

Tuy nhiên với U và V thì lại khác:

$$\frac{\partial L}{\partial V} = \frac{\partial L}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial s_{30}} * \frac{\partial s_{30}}{\partial W}$$

Do $s_{30} = f(W * s_{29} + U * s_{30})$ có s_{29}phụ thuộc vào W. Nên áp dụng công

thức: (f(x) ∗ g(x))’ = f’(x) ∗ g(x) + f(x) ∗ g’(x). Ta có:

$$\frac{\partial s_{30}}{\partial W} = \frac{\partial s'_{30}}{\partial W} + \frac{\partial s_{30}}{\partial s_{29}} * \frac{\partial s_{29}}{\partial W}$$

Trong đó:

$\frac{\partial s'_{30}}{\partial W}$ là đạo hàm của s_{30} với W khi coi s_{29} là constant với W. Áp dụng chain rule:

$$\frac{\partial L}{\partial W} = \sum_{i=0}^{30} \frac{\partial L}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial s_{30}} * \frac{\partial s_{30}}{\partial s_i} * \frac{\partial s'_i}{\partial s_{30}}$$

Trong đó:

2.2. LSTM

2.2.1. LSTM (long-short term memory)

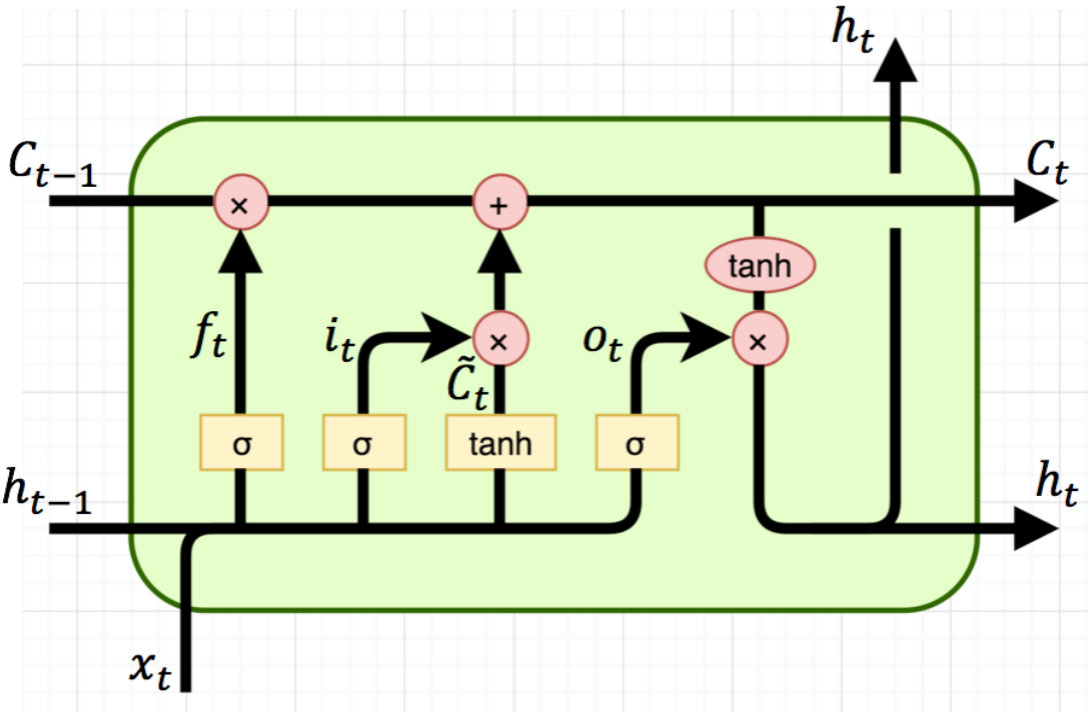
a. Long-short term memory (LSTM)

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

Như vậy về lý thuyết là RNN có thể mang thông tin từ các layer trước đến các layer sau, nhưng thực tế là thông tin chỉ mang được qua một số lượng state nhất định, sau đó thì sẽ bị vanishing gradient, hay nói cách khác là model chỉ học được từ các state gần nó => short term memory. Cùng thử lấy ví dụ về short term memory nhé. Bài toán là dự đoán từ tiếp theo trong đoạn văn. Đoạn đầu tiên "Mặt trời mọc ở hướng ...", ta có thể chỉ sử dụng các từ trước trong câu để đoán là đông. Tuy nhiên, với đoạn, "Tôi là người Việt Nam. Tôi đang sống ở nước ngoài. Tôi có thể nói trôi chảy tiếng ..." thì rõ ràng là chỉ sử dụng từ trong câu đấy hoặc câu trước là không thể dự đoán được từ cần điền là Việt. Ta cần các thông tin từ state ở trước đó rất xa => cần long term memory điều mà RNN không làm được => Cần một mô hình mới để giải quyết vấn đề này => Long shortterm memory (LSTM) ra đời.

Ở state thứ t của mô hình LSTM:

- Output: ct ,ht , ta gọi c là cell state, h là hidden state.
- Input: ct−1,ht−1, xt . Trong đó xt là input ở state thứ t của model. ct−1,ht−1 là output của layer trước. h đóng vai trò khá giống như s ở RNN, trong khi c là điểm mới của LSTM.



Hình 2.14. Mô hình LSTM.

Cách đọc biểu đồ trên: bạn nhìn thấy kí hiệu σ, tanh ý là bước đấy dùng sigma, tanh activation function. Phép nhân ở đây là element-wise multiplication, phép cộng là cộng ma trận.

ft ,it ,ot tương ứng với forget gate, input gate và output gate.

- Forget gate: ft = σ(Uf ∗ xt +Wf ∗ ht−1 +bf)
- Input gate: it = σ(Ui ∗ xt +Wi ∗ ht−1 +bi)
- Output gate: ot = σ(Uo ∗ xt +Wo ∗ ht−1 +bo)

Nhận xét: 0 < ft ,it ,ot < 1; bf ,bi ,bo là các hệ số bias; hệ số W, U giống như trong bài RNN.

- c˜t = tanh(Uc ∗ xt +Wc ∗ ht−1 +bc), bước này giống hệt như tính st trong RNN.
- ct = ft ∗ ct−1 + it ∗ c˜t , forget gate quyết định xem cần lấy bao nhiêu từ cell state trước và input gate sẽ quyết định lấy bao nhiêu từ input của state và hidden layer của layer trước.
- ht = ot ∗tanh(ct), output gate quyết định xem cần lấy bao nhiêu từ cell state để trở thành output của hidden state. Ngoài ra ht cũng được dùng để tính ra output yt cho state t.

Nhận xét: ht , c˜t khá giống với RNN, nên model có short term memory. Trong

khi đó ct giống như một băng chuyền ở trên mô hình RNN vậy, thông tin nào cần quan trọng và dùng ở sau sẽ được gửi vào và dùng khi cần => có thể mang thông tin từ đi xa=> long term memory. Do đó mô hình LSTM có cả short term memory và long term memory.

b. LSTM chống vanishing gradient

2.2.2. Bidirectional LSTM

Bi-LSTM (Bidirectional Long Short-Term Memory) là một loại mạng nơ-ron hồi quy (RNN) xử lý dữ liệu tuần tự theo cả hướng thuận và hướng ngược. Nó kết hợp LSTM với xử lý song hướng, cho phép mô hình nắm bắt cả bối cảnh 'quá khứ' và 'tương lai' của chuỗi đầu vào.

Để hiểu về Bi-LSTM, chúng ta hãy phân tích các thành phần và chức năng của nó:

- Xử lý song hướng: Không giống như RNN truyền thống chỉ xử lý chuỗi đầu vào theo một hướng (hoặc tiến hoặc lùi), Bi-LSTM xử lý chuỗi theo cả hai hướng cùng lúc. Nó bao gồm hai lớp LSTM: một lớp xử lý chuỗi theo hướng tiến và lớp còn lại theo hướng lùi. Mỗi lớp duy trì trạng thái ẩn và ô nhớ riêng.
- Forward Pass: Trong quá trình forward pass, chuỗi đầu vào được đưa vào lớp LSTM forward từ bước thời gian đầu tiên đến bước cuối cùng. Tại mỗi bước thời gian, LSTM forward tính toán trạng thái ẩn của nó và cập nhật ô nhớ dựa trên đầu vào hiện tại và trạng thái ẩn và ô nhớ trước đó.
- Backward Pass: Đồng thời, chuỗi đầu vào cũng được đưa vào lớp LSTM ngược theo thứ tự ngược lại, từ bước thời gian cuối cùng đến bước đầu tiên. Tương tự như forward pass, LSTM ngược tính toán trạng thái ẩn của nó và cập nhật ô nhớ dựa trên đầu vào hiện tại và trạng thái ẩn và ô nhớ trước đó.
- Kết hợp các trạng thái tiến và lùi: Khi các lần chuyển tiếp và lùi hoàn tất, các trạng thái ẩn từ cả hai lớp LSTM được kết hợp tại mỗi bước thời gian. Sự kết hợp này có thể đơn giản như nối các trạng thái ẩn hoặc áp dụng một số phép biến đổi khác.

Lợi ích của Bi-LSTM là nó không chỉ nắm bắt được bối cảnh trước một bước thời gian cụ thể (như trong RNN truyền thống) mà còn nắm bắt được bối cảnh sau đó. Bằng cách xem xét cả thông tin trong quá khứ và tương lai, Bi-LSTM có thể nắm bắt được các phụ thuộc phong phú hơn trong chuỗi đầu vào.

Ưu điểm:

- Ghi lại các phụ thuộc dài hạn: LSTM song hướng có thể ghi lại các phụ thuộc dài hạn trong dữ liệu tuần tự bằng cách xử lý các chuỗi đầu vào theo cả hướng thuận và hướng ngược. Điều này làm cho chúng phù hợp với các tác vụ đòi hỏi mô hình hóa ngữ cảnh trong thời gian dài, chẳng hạn như nhận dạng giọng nói hoặc xử lý ngôn ngữ tự nhiên.
- Hiệu suất được cải thiện: LSTM song hướng thường hoạt động tốt hơn LSTM truyền thống trong các tác vụ như nhận dạng giọng nói, dịch máy và phân tích tình cảm.
- Kiến trúc linh hoạt: Kiến trúc của LSTM song hướng rất linh hoạt và có thể tùy chỉnh bằng cách thêm các lớp bổ sung, chẳng hạn như lớp tích chập hoặc lớp chú ý, để cải thiện hiệu suất.
- Nhược điểm:
- Chi phí tính toán cao: LSTM song hướng có thể tốn kém về mặt tính toán do cần phải xử lý chuỗi đầu vào theo cả hai hướng. Điều này có thể khiến chúng không thực tế khi sử dụng trong môi trường hạn chế về tài nguyên.
- Yêu cầu lượng dữ liệu lớn: LSTM song hướng yêu cầu lượng dữ liệu đào tạo lớn để học các biểu diễn có ý nghĩa của chuỗi đầu vào. Nếu không có đủ dữ liệu đào tạo, mô hình có thể quá phù hợp với tập đào tạo hoặc không thể khái quát hóa thành dữ liệu mới.
- Khó diễn giải: LSTM song hướng thường được coi là "hộp đen", khiến việc diễn giải cách mô hình đưa ra dự đoán trở nên khó khăn. Điều này có thể gây ra vấn đề trong các ứng dụng mà khả năng diễn giải là quan trọng, chẳng hạn như chẩn đoán y tế hoặc phân tích tài chính.

Nhìn chung, quyết định sử dụng LSTM hai chiều nên phụ thuộc vào nhiệm vụ cụ thể và các nguồn lực sẵn có, cũng như sự đánh đổi giữa hiệu suất và khả năng diễn giải.

3. ResNet 50

3.1. Giới Thiệu ResNet50

ResNet (Residual Network) được giới thiệu đến công chúng vào năm 2015 và thậm chí đã giành được vị trí thứ 1 trong cuộc thi ILSVRC 2015 với tỉ lệ lỗi top 5 chỉ 3.57%. Không những thế nó còn đứng vị trí đầu tiên trong cuộc thi ILSVRC and COCO 2015 với ImageNet Detection, ImageNet localization, Coco detection và Coco segmentation.Hiện tại thì có rất nhiều biến thể của kiến trúc ResNet với số lớp khác

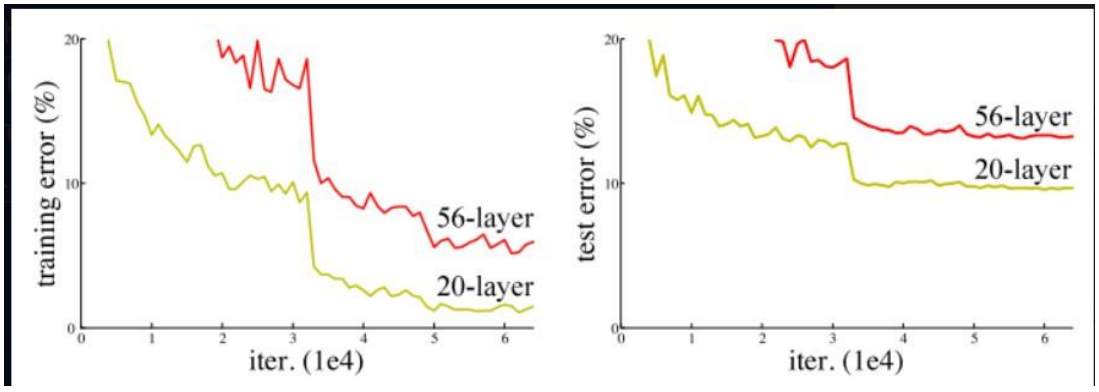
nhau như ResNet-18, ResNet-34, ResNet-50, ResNet-101, ResNet-152,...Với tên là ResNet theo sau là một số chỉ kiến trúc ResNet với số lớp nhất định.

3.2. Tại sao lại xuất hiện mạng ResNet50

Mạng ResNet (R) là một mạng CNN được thiết kế để làm việc với hàng trăm lớp. Một vấn đề xảy ra khi xây dựng mạng CNN với nhiều lớp chập sẽ xảy ra hiện tượng Vanishing Gradient dẫn tới quá trình học tập không tốt.

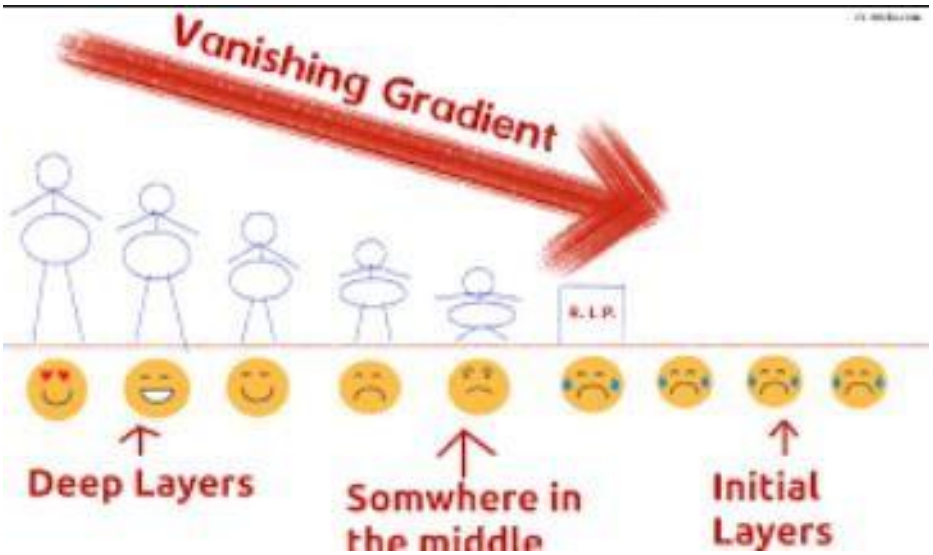
- **Vanishing Gradient**

Backpropagation Algorithm là một kỹ thuật thường được sử dụng trong quá trình tranining. Ý tưởng chung của thuật toán lá sẽ đi từ output layer đến input layer và tính toán gradient của cost function tương ứng cho từng parameter (weight) của mạng. Gradient Descent sau đó được sử dụng để cập nhật các parameter đó.



Hình 2.15. Vanishing Gradient

Toàn bộ quá trình trên sẽ được lặp đi lặp lại cho tới khi mà các parameter của network được hội tụ. Thông thường chúng ta sẽ có một hyperparametr (số Epoch - số lần mà tranining set được duyệt qua một lần và weights được cập nhật) định nghĩa cho số lượng vòng lặp để thực hiện quá trình này. Nếu số lượng vòng lặp quá nhỏ thì ta gặp phải trường hợp mạng có thể sẽ không cho ra kết quả tốt và ngược lại thời gian tranining sẽ lâu nếu số lượng vòng lặp quá lớn.

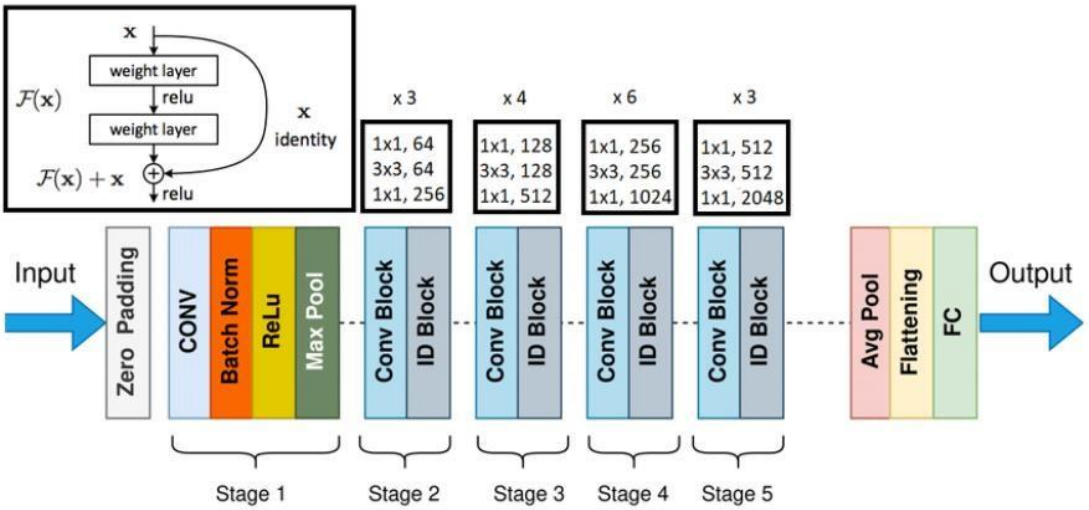


Hình 2.16. Vanishing Gradient

Tuy nhiên, trong thực tế Gradients thường sẽ có giá trị nhỏ dần khi đi xuống các layer thấp hơn. Dẫn đến kết quả là các cập nhật thực hiện bởi Gradients Descent không làm thay đổi nhiều weights của các layer đó và làm chúng không thể hội tụ và mạng sẽ không thu được kết quả tốt. Hiện tượng như vậy gọi là Vanishing Gradients.

3.3. Kiến trúc của mạng ResNet50

ResNet-50 bao gồm 50 lớp được chia thành 5 khối, mỗi khối chứa một tập hợp các khối còn lại. Các khối còn lại cho phép lưu giữ thông tin từ các lớp trước đó, giúp mạng học cách biểu diễn dữ liệu đầu vào tốt hơn.

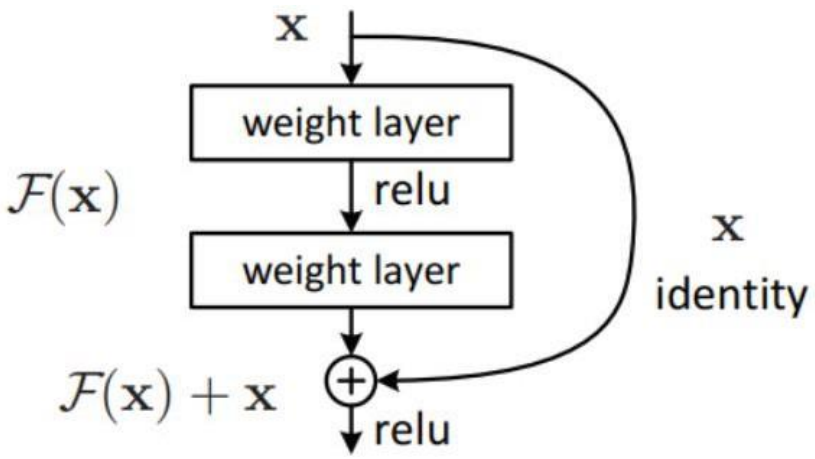


Hình 2.17. Kiến trúc mạng ResNet50

3.3.1. Residual block

Residual Block là một thành phần quan trọng trong mạng nơ-ron sâu, đặc biệt là trong mạng ResNet (Residual Networks). Được giới thiệu bởi He et al. vào năm 2015, ResNet đã giúp cải thiện hiệu suất của các mô hình sâu bằng cách giảm thiểu hiện tượng vanishing gradient và cho phép xây dựng các mạng nơ-ron với hàng trăm lớp mà vẫn có thể huấn luyện được.

Residual Block được thiết kế để học sự khác biệt (residual) giữa đầu ra dự đoán và đầu vào của một chuỗi lớp. Cơ chế này giúp mô hình tập trung vào việc học những điều mới thay vì học lại toàn bộ thông tin từ đầu vào. Điều này có lợi ích lớn trong việc huấn luyện các mạng nơ-ron sâu bởi vì nó giảm thiểu hiện tượng vanishing gradient và cho phép thông tin lan truyền qua các lớp một cách hiệu quả hơn.



Hình 2.18. Residual Block

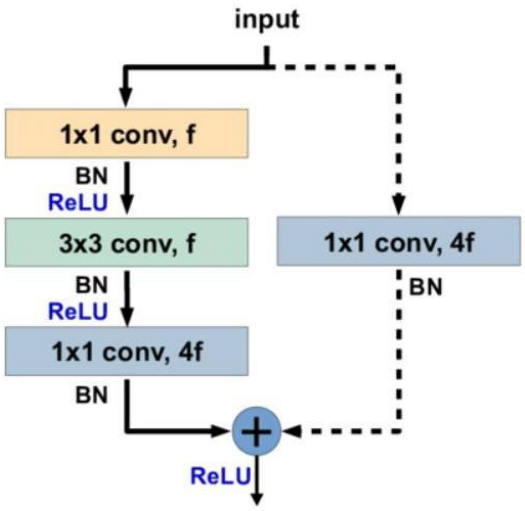
Cấu trúc của một Residual Block bao gồm:

- Main path (đường chính): Một chuỗi các lớp convolutional, normalization và activation function được sử dụng để biến đổi dữ liệu đầu vào.
- Shortcut Connection (Kết nối tắt): Một kết nối tắt trực tiếp từ đầu vào của block tới đầu ra của nó, mà không qua bất kỳ biến đổi nào.
- Kết hợp: Đầu ra của Main Path và đầu vào của Residual Block được cộng lại với nhau để tạo ra đầu ra cuối cùng của Residual Block.

Residual Block giúp cải thiện khả năng học của mạng nơ-ron sâu bằng cách tập trung vào việc học sự khác biệt giữa đầu ra dự đoán và đầu vào, thay vì cố gắng học lại toàn bộ thông tin. Điều này làm cho mạng có khả năng học được các biểu diễn phức tạp của dữ liệu một cách hiệu quả hơn và tránh được hiện tượng vanishing gradient.

3.3.2. Convolutional block

Trong ResNet-50, Convolutional Block thường được sử dụng trong các residual block ở đầu của mạng hoặc khi kích thước của feature map cần thay đổi (chẳng hạn, khi sử dụng stride > 1). Dưới đây là cấu trúc của Convolutional Block trong ResNet-50:



Hình 2.19. Convolutional Block

Cấu trúc của mỗi Convolutional Block trong ResNet-50 bao gồm ba lớp convolutional:

Convolutional Layer với kernel size 1x1:

- Sử dụng để giảm số lượng kênh và kích thước của feature map.
- Thường được sử dụng để giảm chiều dữ liệu và trích xuất các đặc trưng cấp thấp.

Convolutional Layer với kernel size 3x3:

- Là lớp chính trong Convolutional Block.
- Sử dụng stride 1 để duy trì kích thước của feature map.

- Kernel size 3x3 giúp mạng học các đặc trưng phức tạp hơn.

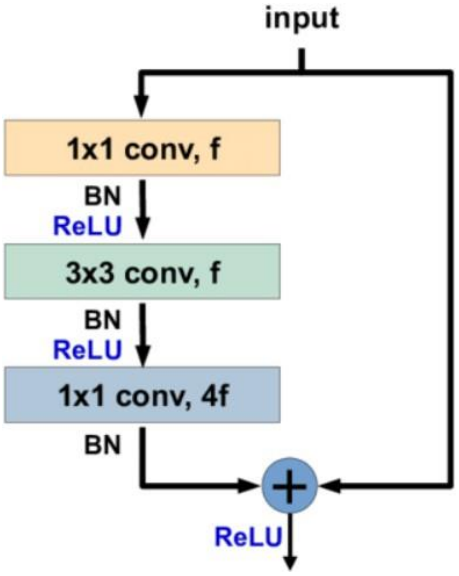
Convolutional Layer với kernel size 1x1:

- Sử dụng để tăng số lượng kênh và kích thước của feature map trở lại.
- Tạo ra kết quả cuối cùng của Convolutional Block với số lượng kênh mong muốn.

Cuối cùng, kết nối tắt (shortcut connection) trong Convolutional Block được thực hiện thông qua một convolutional layer với kernel size 1x1 để điều chỉnh số lượng kênh hoặc kích thước, đảm bảo rằng output của block có cùng kích thước với input để có thể được cộng lại với nhau một cách phù hợp.

3.3.3. Indentity block

Identity Block là một loại khối cơ bản được sử dụng để xây dựng mạng. Nó giữ nguyên kích thước và số lượng kênh của feature map, trong khi vẫn cho phép mạng học các biểu diễn phức tạp của dữ liệu.



Hình 2.15. Indentity Block

Main Path (Đường chính):

- Gồm một chuỗi các lớp convolutional để xử lý dữ liệu đầu vào.
- Mỗi Convolutional layer thường đi kèm với Batch Normalization và

Activation function (thường là ReLU).

Shortcut Connection (Kết nối tắt):

- Kết nối tắt trực tiếp từ đầu vào của block tới đầu ra của block mà không qua bất kỳ biến đổi nào.
- Kết nối tắt giúp thông tin ban đầu được truyền qua mà không bị thay đổi, giúp tránh được hiện tượng vanishing gradient và mất mát thông tin.

Kết hợp:

- Kết quả đầu ra của Main Path được cộng với đầu vào của block thông qua kết nối tắt.
- Cộng hai đầu vào này tạo ra đầu ra cuối cùng của Identity Block.

Identity Block trong ResNet-50 giúp đảm bảo rằng thông tin ban đầu không bị thay đổi nhiều qua quá trình lan truyền ngược, và đồng thời cho phép mạng học các biểu diễn phức tạp của dữ liệu thông qua các lớp convolutional

CHƯƠNG III: ÁP DỤNG CNN VÀ LTSM VÀO BÀI TOÁN PHÁT HIỆN HÀNH VI BẠO LỰC

1. Mô tả bài toán

Bài toán phát hiện hành vi bạo lực là quá trình xác định và phân loại các hành vi bạo lực từ chuỗi dữ liệu hình ảnh hoặc video đầu vào. Mục tiêu của bài toán này là từ dữ liệu đầu vào, phát hiện và nhận diện chính xác các hành vi có dấu hiệu bạo lực để hỗ trợ trong giám sát an ninh và các ứng dụng liên quan.

- **Đầu vào**: Dữ liệu hình ảnh hoặc video, thường là các cảnh quay chứa hành vi của nhiều người trong không gian công cộng, nơi cần xác định xem có hành vi bạo lực nào đang diễn ra hay không.
- **Đầu ra**: Phát hiện hành vi bạo lực dưới dạng nhãn hoặc cảnh báo cho các hành vi nghi ngờ có tính chất bạo lực (ví dụ: xô xát, đánh đấm, ẩu đả).

Quy trình của bài toán phát hiện hành vi bạo lực có thể được mô tả như sau:

- **Thu thập dữ liệu**: Thu thập dữ liệu video từ các nguồn công cộng, bao gồm cả các tình huống có và không có hành vi bạo lực. Dữ liệu này cần đa dạng về điều kiện ánh sáng, góc quay, số lượng người trong khung hình, và các tình huống khác nhau để tăng độ chính xác của mô hình.
- **Tiền xử lý dữ liệu**: Chuẩn hóa và làm sạch dữ liệu video để loại bỏ nhiễu. Các bước tiền xử lý thường bao gồm cắt khung hình để tập trung vào các hành vi của đối tượng, điều chỉnh độ sáng, và chuyển đổi video thành dạng khung hình liên tiếp để trích xuất đặc trưng.
- **Sử dụng các mô hình học sâu**: Sử dụng các kiến trúc mạng học sâu như CNN (Convolutional Neural Network) kết hợp với LSTM (Long Short-Term Memory) để trích xuất đặc trưng từ chuỗi khung hình và phân tích động thái của đối tượng. Các mô hình này giúp nhận diện các dấu hiệu bất thường trong hành vi của người dùng.
- **Phân tích và dự đoán**: Sau khi trích xuất đặc trưng, các mô hình như Softmax hoặc các lớp phân loại khác có thể được sử dụng để dự đoán và phân loại xem có xảy ra hành vi bạo lực hay không, dựa trên các đặc trưng đã trích xuất từ chuỗi hành động trong video.
- **Kiểm tra và đánh giá mô hình**: Đánh giá hiệu suất của mô hình trên tập kiểm tra bằng cách sử dụng các chỉ số như độ chính xác, độ nhạy, độ đặc hiệu và F1-score để đảm bảo mô hình có thể phát hiện đúng các hành vi bạo lực trong nhiều tình huống khác nhau.

Bài toán phát hiện hành vi bạo lực có nhiều ứng dụng trong các lĩnh vực như:

- **An ninh và giám sát**: Giúp các cơ quan an ninh nhanh chóng phát hiện và phản ứng với các tình huống bạo lực xảy ra trong khu vực công cộng như sân bay, nhà ga, công viên hoặc khu phố.

- **Giám sát trong trường học**: Ứng dụng công nghệ vào giám sát, hỗ trợ phát hiện các hành vi xung đột hoặc bạo lực học đường để can thiệp kịp thời.
- **Các sự kiện đông người**: Phát hiện hành vi bạo lực trong các sự kiện lớn như lễ hội, buổi hòa nhạc, hoặc trận đấu thể thao, nơi khả năng xảy ra xung đột cao.
- **Nghiên cứu hành vi**: Giúp các nhà nghiên cứu và chuyên gia trong lĩnh vực xã hội học phân tích các mẫu hành vi bạo lực, từ đó đưa ra các biện pháp phòng ngừa hiệu quả hơn.

Bài toán phát hiện hành vi bạo lực là một lĩnh vực mới với tiềm năng ứng dụng cao, đặc biệt trong bối cảnh ngày càng cần thiết các hệ thống giám sát thông minh và tự động.

1.1. Đặt vấn đề

Phát hiện hành vi bạo lực là một lĩnh vực quan trọng trong việc xây dựng các hệ thống giám sát thông minh nhằm phát hiện sớm các tình huống có khả năng gây nguy hiểm. Với sự tiến bộ vượt bậc của các công nghệ như máy học và học sâu, việc nhận diện và phân tích hành vi từ hình ảnh và video đã trở nên khả thi hơn bao giờ hết. Thông qua việc phát hiện hành vi bạo lực, chúng ta có thể tăng cường an ninh trong nhiều môi trường như khu vực công cộng, trường học, và các sự kiện lớn.

Việc áp dụng công nghệ phát hiện hành vi bạo lực vào thực tế mang lại nhiều lợi ích quan trọng, bao gồm:

- **Phát hiện sớm và can thiệp kịp thời**: Hệ thống có thể giúp nhận diện các tình huống xung đột ngay từ giai đoạn đầu, cho phép can thiệp kịp thời để ngăn chặn bạo lực leo thang.
- **Tăng cường an ninh công cộng**: Trong các khu vực đông người như sân bay, nhà ga, hay các sự kiện thể thao, phát hiện hành vi bạo lực có thể giúp giám sát an ninh một cách tự động, giảm thiểu các nguy cơ đối với cộng đồng.
- **Hỗ trợ trong giáo dục và môi trường học đường**: Phát hiện bạo lực trong trường học giúp ngăn chặn và xử lý các tình huống bạo lực học đường, tạo ra môi trường an toàn hơn cho học sinh và giáo viên.

Trong bài toán phát hiện hành vi bạo lực, có một số thách thức quan trọng mà cần phải đối mặt:

- **Độ chính xác trong nhận diện hành vi**: Việc phân biệt các hành vi bạo lực và không bạo lực là phức tạp, đặc biệt là khi các hành vi này có thể giống nhau (chẳng hạn như chơi đùa và xô xát). Điều này đòi hỏi mô hình phải có khả năng phân tích chính xác các chi tiết nhỏ trong hành động.
- **Thiếu dữ liệu chất lượng cao**: Dữ liệu video về hành vi bạo lực thường khó thu thập và gán nhãn do tính chất nhạy cảm và các vấn đề liên quan đến quyền riêng tư. Các video bạo lực có thể không nhiều và thường kèm theo các yếu tố gây nhiễu, ảnh hưởng đến chất lượng huấn luyện mô hình.

- **Biến động ánh sáng và góc quay**: Các video giám sát thường có chất lượng thấp, ánh sáng không đồng đều, và góc quay không cố định, gây khó khăn cho mô hình trong việc nhận diện chính xác hành vi bạo lực.
- **Phân tích chuỗi hành động liên tục**: Bạo lực là một dạng hành vi phức tạp, thường bao gồm chuỗi các hành động xảy ra liên tiếp. Mô hình cần có khả năng phân tích những chuỗi hành động này để đưa ra dự đoán chính xác, điều này đòi hỏi khả năng xử lý dữ liệu tuần tự và hiểu rõ ngữ cảnh.
- **Đa dạng văn hóa và bối cảnh**: Định nghĩa về hành vi bạo lực có thể khác nhau giữa các nền văn hóa và bối cảnh xã hội, điều này tạo ra thách thức trong việc xây dựng một mô hình có khả năng tổng quát và phù hợp với nhiều môi trường khác nhau.

1.2. Mục tiêu

Mục tiêu chính trong bài toán phát hiện hành vi bạo lực là xây dựng một hệ thống có khả năng phát hiện và phân loại các hành vi bạo lực từ dữ liệu video hoặc hình ảnh, hỗ trợ giám sát và can thiệp sớm trong các tình huống nguy hiểm. Để đạt được mục tiêu này, cần thực hiện các mục tiêu cụ thể sau:

- **Phát hiện chính xác**: Hệ thống cần phân biệt rõ ràng giữa các hành vi bạo lực và phi bạo lực với độ chính xác cao, đáp ứng yêu cầu của các ứng dụng như giám sát an ninh, bảo vệ công cộng, và xử lý hành vi bạo lực trong trường học.
- **Xử lý ngữ cảnh phức tạp**: Hệ thống cần có khả năng nhận diện hành vi trong các tình huống phức tạp và đa dạng, như trong các không gian công cộng, nơi có nhiều người, hoặc trong điều kiện ánh sáng và góc nhìn thay đổi liên tục.
- **Độ tin cậy và ổn định**: Hệ thống cần đảm bảo hoạt động ổn định và đáng tin cậy trong nhiều điều kiện khác nhau, bao gồm những yếu tố như chất lượng video kém, các góc quay khác nhau, và sự nhiễu động từ các hành động khác.
- **Hiệu suất tính toán**: Đặc biệt với các ứng dụng giám sát thời gian thực, hệ thống cần xử lý dữ liệu nhanh chóng và hiệu quả để đưa ra cảnh báo sớm, giúp can thiệp kịp thời khi phát hiện hành vi bạo lực.
- **Phản hồi nhanh**: Hệ thống cần cung cấp kết quả phát hiện một cách nhanh chóng nhằm đáp ứng yêu cầu thời gian thực trong các ứng dụng giám sát an ninh, giúp tăng cường khả năng bảo vệ và giảm thiểu nguy cơ.
- **Tính linh hoạt và mở rộng**: Hệ thống cần dễ dàng mở rộng, cho phép phát hiện thêm nhiều dạng hành vi bạo lực khác hoặc nâng cao hiệu suất khi có thêm dữ liệu mới.
- **Khả năng tích hợp**: Hệ thống cần có khả năng tích hợp với các thiết bị giám sát, hệ thống an ninh và các nền tảng khác, đảm bảo khả năng triển khai linh hoạt trên nhiều thiết bị, bao gồm cả camera an ninh và hệ thống phân tích video.

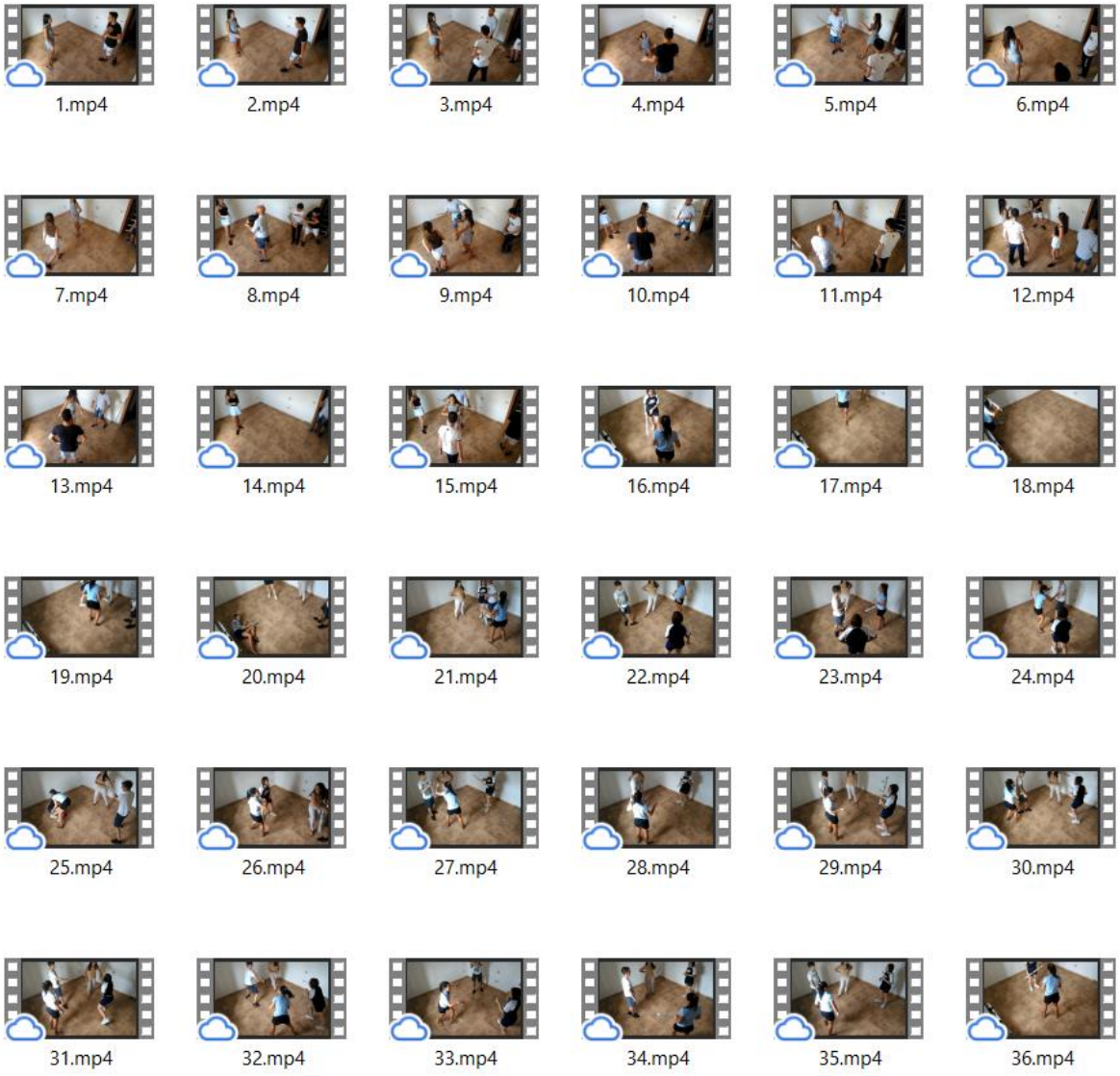
- **Bảo mật và quyền riêng tư**: Đặc biệt quan trọng trong các ứng dụng giám sát, hệ thống cần tuân thủ các quy định bảo mật và bảo vệ quyền riêng tư của cá nhân, đảm bảo việc sử dụng dữ liệu hình ảnh hoặc video được thực hiện một cách an toàn và tôn trọng quyền cá nhân.

2. Bộ dữ liệu

Nhóm đã thu thập tập dữ liệu về các hành động bạo lực với không bạo lực của nhóm Vaidehi Pawar trên Kaggle. Trong đó, bao gồm 2 nhóm hành động chính khác nhau như sau: 'violence' và 'non-violence' với bao gồm tổng 230 video, mỗi video đều có tốc độ là 30 FPS và được chia cụ thể dữ liệu cho các hành động như dưới:

Hành động	**Violence**	**Female**
Số lượng	110	120

Bảng 1: Bảng dữ liệu các hành động trong tập dữ liệu



Hình 3.1: Hình ảnh ví dụ về tập dữ liệu

3. Tiền xử lý dữ liệu

3.1. Trích xuất khung hình

Chương trình dưới tạo dataset từ các video trong tập dữ liệu, mỗi video thuộc một lớp cụ thể. Đầu tiên, các danh sách features, labels, và video_files_paths được khởi tạo để lần lượt lưu trữ các frame trích xuất từ video, nhãn của từng video (dựa vào lớp), và đường dẫn đầy đủ của video. Sau đó duyệt qua từng lớp trong danh sách CLASSES_LIST và lấy danh sách các video trong thư mục của lớp đó. Với mỗi video, chương trình sẽ xác định đường dẫn đầy đủ và sử dụng hàm frames_extraction

(video_file_path) để trích xuất các frame. Các frame này, nhãn của lớp (class_index), và đường dẫn video được thêm vào các danh sách tương ứng. Cuối cùng, hàm trả về ba danh sách features, labels, và video_files_paths, dùng để tạo dataset phục vụ huấn luyện mô hình phát hiện hành vi bạo lực.

```python
def frames_extraction(video_path):
    frames_list = []

    # Read the Video File
    video_reader = cv2.VideoCapture(video_path)

    # Get the total number of frames in the video.
    video_frames_count = int(video_reader.get(cv2.CAP_PROP_FRAME_COUNT))

    # Iterate through the Video Frames.
    while True:
        # Reading the frame from the video.
        success, frame = video_reader.read()

        if not success:
            break

        # Resize the Frame to fixed height and width.
        resized_frame = cv2.resize(frame, (IMAGE_HEIGHT, IMAGE_WIDTH))

        # Normalize the resized frame
        normalized_frame = resized_frame / 255

        # Append the normalized frame into the frames list
        frames_list.append(normalized_frame)

    video_reader.release()
    return frames_list
```

Hình 3.2: Chương trình trích xuất khung hình từ video

Kết quả thu được sau khi thực hiện chương trình là một tập dữ liệu gồm các ảnh được trích xuất từ video, mỗi video sẽ trích xuất với lượng frame bằng với SEQUENCE_LENGTH cụ thể là 30. Tức là nếu 1 video gồm 5 giây và tốc độ là 30 FPS thì cứ 5 frames thì sẽ lưu lấy 1 frame. Bảng dưới sẽ thống kê tổng tất cả frame đã thu được từ các video:

Hành động	**Violence**	**Female**
Số lượng	3300	3600

Bảng 2: Bảng dữ liệu các hành động trong tập dữ liệu

3.2. Gán nhãn dữ liệu

Chương trình dưới đây dùng để tạo dataset cho bài toán phát hiện hành vi bạo

lực từ video, trong đó mỗi video được gán nhãn theo hai lớp: "violence" (bạo lực) và "nonviolence" (không bạo lực). Đầu tiên, ba danh sách features, labels, và video_files_paths được khởi tạo để lần lượt lưu trữ các frame được trích xuất từ video, nhãn của từng video, và đường dẫn đầy đủ của video. Chương trình duyệt qua từng lớp "violence" và "non-violence" từ danh sách CLASSES_LIST, trong đó class_index đại diện cho lớp (0 cho "non-violence" và 1 cho "violence"). Với mỗi video trong thư mục của lớp, chương trình xác định đường dẫn đầy đủ và sử dụng hàm frames_extraction(video_file_path) để trích xuất các frame từ video. Các frame, nhãn của lớp, và đường dẫn video được thêm vào các danh sách tương ứng. Kết quả thu được sau khi chạy chương trình là ba danh sách features, labels, và video_files_paths để xây dựng dataset phục vụ huấn luyện mô hình phát hiện hành vi bạo lực.

```
[ ]  def create_dataset():
         features = []
         labels = []
         video_files_paths = []

         # Iterating through all the classes.
         for class_index, class_name in enumerate(CLASSES_LIST):
             print(f'Extracting Data of Class: {class_name}')

             # Get the list of video files present in the specific class name directory.
             files_list = os.listdir(os.path.join(DATASET_DIR, class_name))

             # Iterate through all the files present in the files list.
             for file_name in files_list:
                 # Get the complete video path.
                 video_file_path = os.path.join(DATASET_DIR, class_name, file_name)

                 # Extract all frames of the video file.
                 frames = frames_extraction(video_file_path)

                 # Append the data to their respective lists.
                 features.append(frames)
                 labels.append(class_index)
                 video_files_paths.append(video_file_path)

         return features, labels, video_files_paths
```

Hình 3.3: Chương trình gán nhãn cho tập dữ liệu

3.3. Chuyển đổi nhãn và chia tập dữ liệu

Đầu tiên, chương trình chuyển đổi các nhãn labels thành dạng one-hot bằng cách sử dụng hàm to_categorical(labels). Phương pháp one-hot encoding giúp biểu diễn nhãn dưới dạng vector nhị phân, trong đó mỗi nhãn được đại diện bằng một vector có độ dài bằng số lớp và chỉ có một giá trị "1" tại vị trí tương ứng với nhãn của nó, các vị trí khác là "0". Điều này giúp mô hình dễ dàng xử lý các lớp đầu ra.

```
[ ]  # convert labels into one-hot-encoded vectors
     one_hot_encoded_labels = to_categorical(labels)
```

Hình 3.4: Chương trình thực hiện việc chuyển đổi nhãn

Sau khi chuyển đổi nhãn, chương trình sử dụng train_test_split để chia dữ liệu thành hai phần: 80% cho huấn luyện (features_train, labels_train) và 20% cho kiểm tra (features_test, labels_test). Tham số shuffle=True giúp xáo trộn dữ liệu trước khi chia để đảm bảo tính ngẫu nhiên và tránh trường hợp dữ liệu bị phân lớp, còn random_state=42 đảm bảo tính nhất quán khi chạy lại. Việc chia dữ liệu này giúp mô hình có tập dữ liệu để huấn luyện và một tập khác để kiểm tra, đánh giá hiệu suất.

```
[ ]  # Split the Data into Train ( 80% ) and Test Set ( 20% ).
     features_train, features_test, labels_train, labels_test = train_test_split(features, one_hot_encoded_labels, test_size = 0.2,
                                                                                  shuffle = True, random_state = 42)
```

Hình 3.5: Chương trình thực hiện chia tập dữ liệu

4. Tinh chỉnh, xây dựng và huấn luyện, đánh giá mô hình

4.1. Tinh chỉnh chương trình

Chương trình dưới sử dụng mô hình ResNet50, một mạng nơ-ron tích chập sâu nổi tiếng, để làm nền tảng cho bài toán phân loại hoặc phát hiện hành vi trong video. Đầu tiên, mô hình ResNet50 được tải từ keras.applications, với tham số include_top=False nhằm loại bỏ các lớp đầu ra ban đầu của mô hình (phần trên cùng), cho phép chúng ta thêm các lớp tùy chỉnh cho bài toán cụ thể. Tham số weights = "imagenet" cho phép sử dụng các trọng số đã được huấn luyện trước trên bộ dữ liệu ImageNet, giúp mô hình bắt đầu với các đặc trưng mạnh mẽ hơn thay vì huấn luyện từ đầu.

```
[ ]  from keras.applications import ResNet50

     resnet = ResNet50(include_top=False , weights="imagenet", input_shape=(IMAGE_HEIGHT,IMAGE_WIDTH,3))

     #Fine-Tuning to make the last 40 layer trainable
     resnet.trainable = True
```

Hình 3.6: Chương trình tinh chỉnh mô hình ResNet50

Kích thước đầu vào của mô hình được đặt là (64, 64, 3), tương ứng với chiều cao và chiều rộng hình ảnh là 64 pixel và 3 kênh màu (RGB). Sau đó, dòng resnet.trainable = True cho phép tinh chỉnh (fine-tuning) toàn bộ mô hình, bao gồm cả 40 lớp cuối cùng, nhằm tối ưu hóa cho bài toán cụ thể. Việc tinh chỉnh này giúp mô hình học thêm đặc trưng từ dữ liệu của bài toán hiện tại mà vẫn giữ được các đặc trưng chung từ ImageNet, cải thiện hiệu quả trong bài toán phân loại hành vi.

```
model = Sequential()

#Specifying Input to match features shape
model.add(Input(shape = (SEQUENCE_LENGTH, IMAGE_HEIGHT, IMAGE_WIDTH, 3)))

# Passing mobilenet in the TimeDistributed layer to handle the sequence
model.add(TimeDistributed(resnet))

model.add(Dropout(0.25))

model.add(TimeDistributed(Flatten()))

lstm_fw = LSTM(units=32)
lstm_bw = LSTM(units=32, go_backwards = True)

model.add(Bidirectional(lstm_fw, backward_layer = lstm_bw))

model.add(Dropout(0.25))

model.add(Dense(256,activation='relu'))
model.add(Dropout(0.25))

model.add(Dense(128,activation='relu'))
model.add(Dropout(0.25))

model.add(Dense(64,activation='relu'))
model.add(Dropout(0.25))

model.add(Dense(32,activation='relu'))
model.add(Dropout(0.25))


model.add(Dense(2, activation = 'softmax'))
```

Hình 3.7: Chương trình tinh chỉnh mô hình huấn luyện với tập dữ liệu giới tính

Tiếp đến, chương trình trên định nghĩa một hàm create_model() để xây dựng mô hình học sâu (Deep Learning) cho bài toán phát hiện hành vi, sử dụng Keras. Mô hình được thiết kế để xử lý chuỗi các khung hình, tận dụng mạng tích chập (CNN) ResNet50 để trích xuất đặc trưng từ từng khung hình và mạng LSTM hai chiều (Bidirectional LSTM) để học các mối quan hệ thời gian giữa các khung hình trong chuỗi.

- **Lớp đầu vào**: Mô hình bắt đầu với một lớp Input, định dạng đầu vào là (SEQUENCE_LENGTH, IMAGE_HEIGHT, IMAGE_WIDTH, 3), với SEQUENCE_LENGTH là số lượng khung hình trong một chuỗi và trong chương trình đang set là 30. IMAGE_HEIGHT và IMAGE_WIDTH lần lượt là chiều cao và chiều rộng của khung hình đầu vào cụ thể là 64x64, và 3 là số kênh màu (RGB).
- **Trích xuất đặc trưng với ResNet50**: Mạng ResNet50 được đặt bên trong lớp TimeDistributed, giúp áp dụng mô hình CNN này lên từng khung hình riêng lẻ

trong chuỗi. Việc sử dụng ResNet50 giúp trích xuất các đặc trưng từ từng khung hình, thay vì phải huấn luyện lại từ đầu.

- **Lớp LSTM hai chiều**: Sau khi trích xuất đặc trưng, dữ liệu được đưa qua LSTM hai chiều (Bidirectional LSTM) để học các mối quan hệ thời gian. LSTM này có 32 đơn vị ẩn (hidden units) cho mỗi chiều, và go_backwards=True được đặt cho LSTM ngược (backward LSTM layer), cho phép mô hình học cả các mối quan hệ từ trước ra sau và từ sau ra trước trong chuỗi.
- **Các Fully-Connected Layer và Dropout**: Sau LSTM, mô hình có nhiều lớp Dense với các kích thước lần lượt là 256, 128, 64, và 32 nơ-ron, mỗi lớp đi kèm với Dropout(0.25) để giảm thiểu hiện tượng overfitting.
- **Lớp đầu ra**: Lớp cuối cùng là lớp Dense với 2 nơ-ron và hàm kích hoạt softmax, nhằm phân loại thành 2 lớp (có thể là hành vi bạo lực và không bạo lực).

4.2. Huấn luyện mô hình

4.2.1. Callback

Trước khi huấn luyện mô hình, nhóm đã đặt quá trình callback trong quá trình huấn luyện mô hình, nhằm theo dõi và điều chỉnh việc học của mô hình một cách hiệu quả hơn.

```
from tensorflow.keras.optimizers import Adam
# Create Early Stopping Callback to monitor the accuracy
early_stopping_callback = EarlyStopping(monitor = 'val_accuracy', patience = 10, restore_best_weights = True)

# Create ReduceLROnPlateau Callback to reduce overfitting by decreasing learning
reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss',
                                                 factor=0.3,
                                                 patience=5,
                                                 min_lr=0.000005,
                                                 verbose=1)
```

Hình 3.8: Chương trình setup callback cho quá trình huấn luyện

Chương trình này tạo ra hai callback cho quá trình huấn luyện mô hình để cải thiện hiệu suất và tránh overfitting. Callback đầu tiên là EarlyStopping, được sử dụng để dừng sớm quá trình huấn luyện nếu độ chính xác (accuracy) trên tập validation không cải thiện sau một số vòng lặp nhất định. Ở đây, patience được đặt là 10, nghĩa là nếu sau 10 epoch liên tiếp mà độ chính xác không tăng, quá trình huấn luyện sẽ dừng và mô hình sẽ khôi phục về trọng số tốt nhất đã đạt được trước đó (restore_best_weights=True), giúp tránh tình trạng overfitting.

Callback thứ hai là ReduceLROnPlateau, giúp giảm tốc độ học (learning rate) khi mô hình không còn cải thiện về lỗi trên tập validation (val_loss). Nếu val_loss không giảm trong 5 epoch liên tiếp (patience=5), thì ReduceLROnPlateau sẽ giảm learning rate bằng cách nhân với một hệ số (factor=0.3). Learning rate có thể giảm

xuống mức tối thiểu là 0.000005. Điều này giúp mô hình dễ dàng hội tụ khi đạt đến các vùng tối ưu tốt hơn mà không bị mắc kẹt ở các vùng học kém.

4.2.2. Huấn luyện mô hình

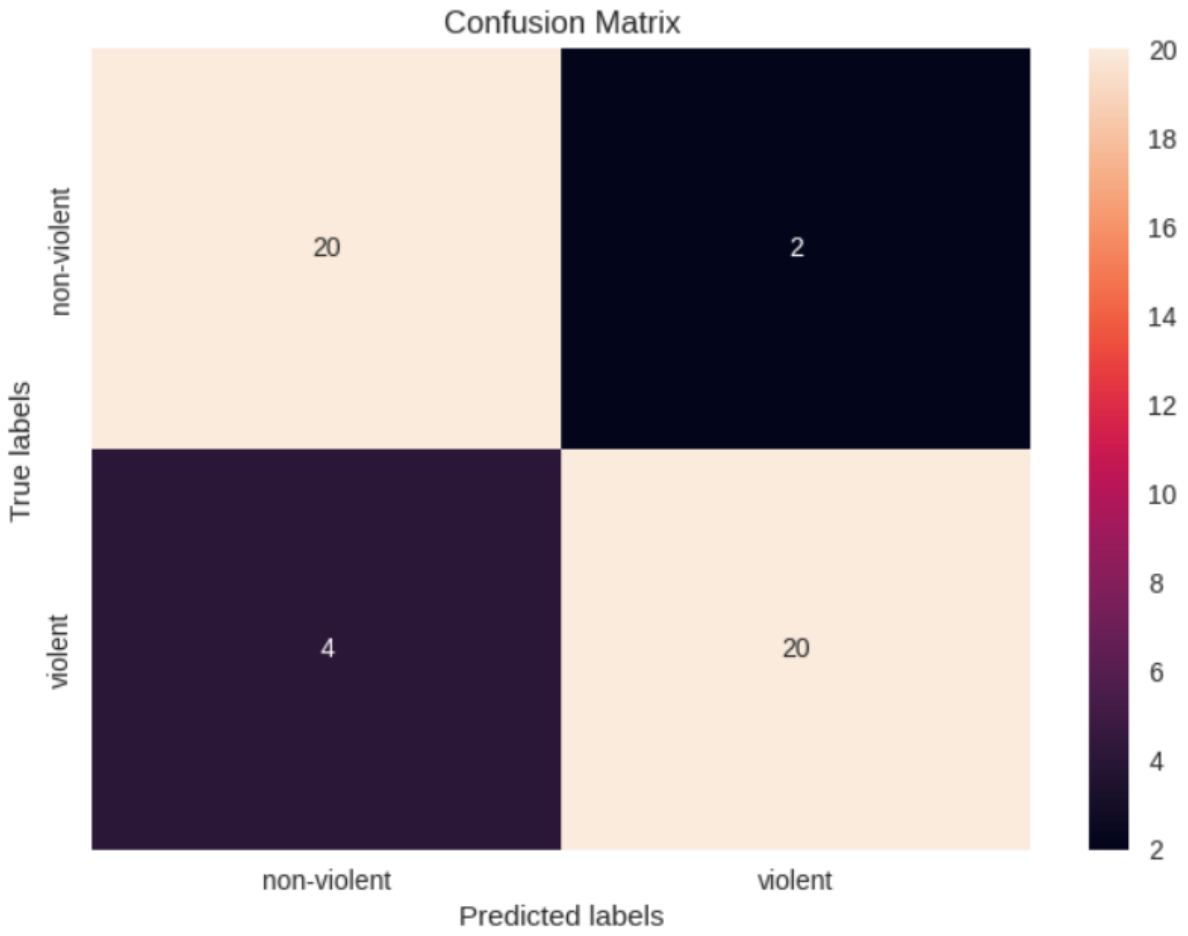
```
# Fitting the model
history = resnetLSTM.fit(x = features_train,
                         y = labels_train,
                         epochs = 50,
                         batch_size = 8,
                         shuffle = True,
                         validation_split = 0.2,
                         callbacks = [early_stopping_callback,reduce_lr])
```

Hình 3.9: Huấn luyện cho mô hình phát hiện hành vi bạo lực

Trong bước huấn luyện này, chương trình sử dụng mô hình resnetLSTM vừa được xây dựng để thực hiện huấn luyện trên dữ liệu features_train với các nhãn tương ứng là labels_train. Quá trình huấn luyện diễn ra trong tối đa 50 epoch, với mỗi batch gồm 8 mẫu, và dữ liệu được xáo trộn (shuffle=True) để tăng tính ngẫu nhiên và cải thiện hiệu quả huấn luyện. Một phần 20% của dữ liệu huấn luyện được tách ra làm tập validation (validation_split=0.2) để theo dõi và đánh giá hiệu suất mô hình trên dữ liệu không nhìn thấy trong mỗi epoch. Hai callback, early_stopping_callback và reduce_lr, được sử dụng trong quá trình này.

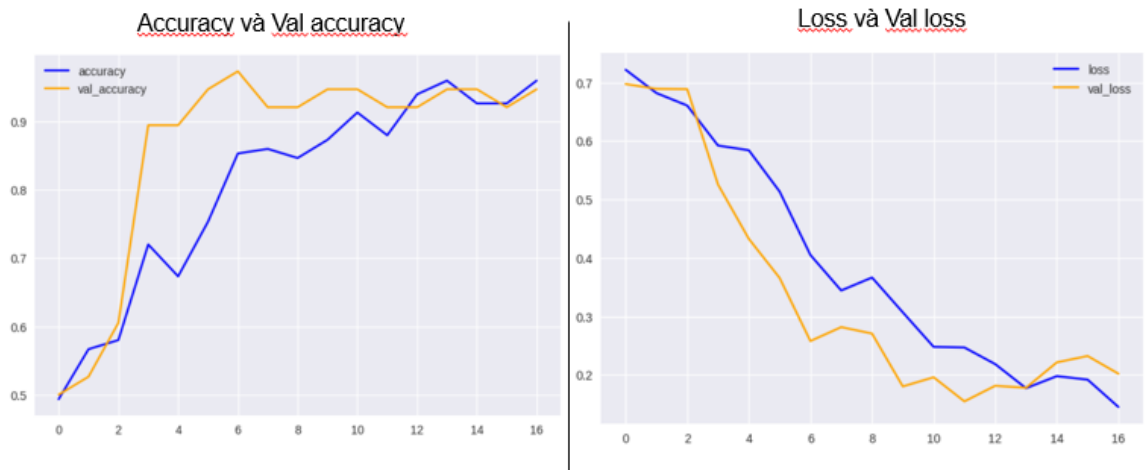
4.3. Đánh giá mô hình

Bài toán phát hiện hành vi bạo lực là một bài toán phân lớp hai nhãn. Để đánh giá hiệu quả của các mô hình trong bài toán nhận diện hành vi bạo lực này, chúng em sử dụng các chỉ số đánh giá như Accuracy và Loss. Giả sử gọi nhãn lớp của các hành vi được xác nhận là "Violent" là lớp một và "Non-Violent" là lớp hai. Khi mô hình phân lớp tiến hành phân loại các tình huống, sẽ xảy ra nhiều trường hợp khác nhau trong việc phân loại, bao gồm cả việc mô hình dự đoán đúng và sai các hành vi. Qua đó, chúng em có thể phân tích chi tiết các chỉ số để đánh giá hiệu suất của mô hình và cải thiện độ chính xác trong việc nhận diện hành vi bạo lực.



Hình 3.5. Đánh giá mô hình

- Ô [1, 1]: mẫu được phân diện là Non-Violent chính xác (thực tế là Non-Violent).
- Ô [1, 2]: mẫu được phân diện là Violent chính xác (thực tế là Non-Violent).
- Ô [2, 2]: mẫu được phân loại Violent chính xác (thực tế là Violent).
- Ô [2, 1]: mẫu được phân loại Violent chính xác (thực tế là Non-Violent).



Hình 3.13: Quá trình train với model ResNet50 + LSTM

Từ hình ảnh ở trên cho thấy độ lỗi của mô hình tiến dần về 0, điều này thể hiện rằng mô hình đang học cách dự đoán đúng kết quả và hiệu suất của nó đang được cải thiện. Đây là một dấu hiệu tích cực và phản ánh sự tiến bộ của mô hình trong việc học từ dữ liệu huấn luyện và đánh giá độ chính xác của nó trên dữ liệu xác thực.

Ta cũng có thể thấy hiệu suất của mô hình trên tập huấn luyện và tập đánh giá đang tiến gần tới 1. Đặc biệt hiệu suất những epoch gần cuối trên tập huấn luyện và tập đánh giá có hiệu suất gần bằng nhau điều này có nghĩa mô hình đang phát triển theo hướng tích cực và tránh được hiện tượng overfitting.

Về đánh giá kết quả lỗi của mô hình sử dụng kiến trúc mô hình trên tập huấn luyện và tập xác thực trong các hình ảnh ở trên, chúng ta thấy các hàm đo độ chính xác và hàm mất mát có độ ổn định. Điều này cho thấy khả năng tổng quát hóa của mô hình khá tốt.

5. Kết quả thực nghiệm

Dưới đây là một số hình ảnh kết quả nhóm thực hiện dự đoán bằng dữ liệu cá nhân:

Hình 3.15: Kết quả thực nghiệm với các khung hình có dấu hiệu bạo lực

Hình 3.16: Kết quả thực nghiệm với các khung hình không có dấu hiệu bạo lực

6. Khó khăn gặp phải

Trong quá trình thực hiện và đánh giá bài toán phát hiện hành vi bạo lực, nhóm đã gặp phải nhiều khó khăn và hạn chế đáng kể liên quan đến chất lượng và tính sẵn có của dữ liệu. Trước hết, do hành vi bạo lực là một hiện tượng phức tạp và nhạy cảm, việc thu thập dữ liệu phù hợp gặp nhiều trở ngại. Các hành vi bạo lực thường xảy ra ở những nơi ít người qua lại và không có sự hiện diện của camera an ninh giám sát, dẫn đến số lượng dữ liệu có sẵn để sử dụng vô cùng hạn chế. Những dữ liệu có được thường không đủ phong phú để bao quát nhiều trường hợp khác nhau, gây khó khăn

trong việc xây dựng một mô hình có khả năng phát hiện bạo lực với độ chính xác cao.

Bên cạnh đó, hành vi bạo lực có thể diễn ra dưới nhiều hình thức và trong các bối cảnh khác nhau như đường phố, trường học, công viên, hoặc nhà riêng, từ các hành vi bạo lực nhẹ đến nghiêm trọng. Tuy nhiên, việc thu thập đủ dữ liệu để bao quát tất cả các tình huống và mức độ này là rất khó khăn. Điều này dẫn đến mô hình dễ bị nhầm lẫn khi gặp các tình huống mới không có trong tập huấn luyện. Ngoài ra, việc thiếu dữ liệu nhãn chính xác cũng là một thách thức lớn. Gán nhãn hành vi bạo lực đòi hỏi sự chính xác và có thể cần sự đánh giá của chuyên gia, vì các hành vi này có thể dễ bị nhầm với các hành vi không bạo lực, như chơi đùa hoặc tập luyện thể thao. Việc này không chỉ tốn thời gian và chi phí mà còn khó đảm bảo chất lượng dữ liệu đầu vào cho mô hình.

KẾT LUẬN

ResNet50 là một mô hình mạng nơ-ron sâu tiên tiến, được sử dụng rộng rãi trong việc phát hiện hành vi bạo lực từ video. Với kiến trúc đặc biệt, ResNet50 sử dụng các khối residual giúp cải thiện khả năng học tập của mô hình, cho phép nó nhận diện và phân tích các hành động phức tạp trong video một cách hiệu quả. Mô hình này có khả năng trích xuất các đặc trưng không gian từ các khung hình, giúp xác định các hành vi bạo lực dựa trên sự tương tác giữa các đối tượng trong môi trường.

Khi kết hợp ResNet50 với các mạng LSTM (Long Short-Term Memory), chúng ta có thể phân tích thông tin không gian và thời gian từ video. ResNet50 chịu trách nhiệm trích xuất đặc trưng từ các khung hình riêng lẻ, trong khi LSTM xử lý chuỗi dữ liệu để nhận diện các mẫu hành vi bạo lực diễn ra theo thời gian. Sự kết hợp này giúp mô hình nắm bắt được các biến đổi hành động và động lực giữa các đối tượng, từ đó đưa ra những dự đoán chính xác hơn về hành vi. Kết quả cho thấy mô hình ResNet50 + LSTM đạt độ chính xác tới 97%, điều này cho thấy khả năng vượt trội trong việc phát hiện hành vi bạo lực.

Tuy nhiên, việc sử dụng ResNet50 và LSTM cũng đối mặt với một số thách thức. Mô hình có thể gặp khó khăn khi phân tích trong các tình huống ánh sáng yếu hoặc khi có nhiều đối tượng xuất hiện trong khung hình cùng lúc, làm cho việc xác định hành vi bạo lực trở nên phức tạp. Hơn nữa, LSTM yêu cầu một lượng lớn dữ liệu huấn luyện để đạt được hiệu suất tốt, và việc thu thập và gán nhãn dữ liệu cho các hành vi bạo lực có thể tốn kém và khó khăn. Ngoài ra, thời gian xử lý của mô hình có thể lâu hơn so với các phương pháp truyền thống, đặc biệt là trong các ứng dụng yêu cầu phản hồi thời gian thực.

Mặc dù có những thách thức, việc áp dụng ResNet50 và LSTM trong phát hiện hành vi bạo lực hứa hẹn mang lại nhiều ứng dụng hữu ích trong an ninh công cộng, giám sát và quản lý rủi ro, giúp tăng cường an toàn cho cộng đồng và nâng cao nhận thức về các tình huống bạo lực.

TÀI LIỆU THAM KHẢO

1. Giáo trình Thị Giác Máy Tính-Phùng Thế Huân

2. Giáo trình Deep Learning cơ bản-Ngô Hữu Huy

3. https://viso.ai/deep-learning/vgg-very-deep-convolutional- networks/

4. https://viblo.asia/p/gioi-thieu-mang-resnet-vyDZOa7R5wj

5. https://datagen.tech/guides/computer-vision/vgg16/

6. https://viso.ai/deep-learning/resnet-residual-neural-network/

7. https://medium.com/@anishnama20/understanding-bidirectional-lstm-for-sequential-data-processing-b83d6283befc

8. https://viblo.asia/p/recurrent-neural-network-tu-rnn-den-lstm-gGJ597z1ZX2

9. https://www.researchgate.net/publication/351840108_Image_Caption_Generation_Methodologies