

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN



TIÊU LUẬN CHUYÊN NGÀNH
ĐỀ TÀI
PHÁT HIỆN BẠO LỰC (VIOLENCE DETECTION)
TRONG VIDEO

GVHD: Th.S Quách Đình Hoàng

Sinh viên thực hiện:

Vũ Văn Phước (MSSV: 19133045)

Nguyễn Hoài Nam (MSSV: 19133037)

Tp. Hồ Chí Minh, ngày 10 tháng 9 năm 2022

PHIẾU NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

Họ và tên Sinh viên 1: Vũ Văn Phước MSSV: 19133045
Họ và tên Sinh viên 2: Nguyễn Hoài Nam MSSV: 19133037

Ngành: Kỹ thuật dữ liệu

Tên đề tài: Phát hiện bạo lực (violence detection) trong video

Họ và tên Giảng viên hướng dẫn: Th.S Quách Đình Hoàng

NHẬN XÉT

Về nội dung đề tài khối lượng thực hiện:

.....
.....
.....

1. Ưu điểm:

.....
.....
.....
.....

2. Khuyết điểm

.....
.....
.....
.....

3. Đề nghị cho bảo vệ hay không?

4. Đánh giá loại:

5. Điểm:

Tp. Hồ Chí Minh, ngày 29 tháng 12 năm 2022

Giáo viên hướng dẫn
(Ký & ghi rõ họ tên)

PHIẾU NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN

Họ và tên Sinh viên 1: Vũ Văn Phước MSSV: 19133045

Họ và tên Sinh viên 2: Nguyễn Hoài Nam MSSV: 19133037

Ngành: Kỹ thuật dữ liệu

Tên đề tài: Phát hiện bạo lực (violence detection) trong video

Họ và tên Giảng viên hướng dẫn: TS. Nguyễn Thiên Bảo

NHẬN XÉT

1. Về nội dung đề tài & khối lượng thực hiện:

Lý thuyết:

- Tìm hiểu mô hình mạng convolutional neural network (CNN), recurrent neural network (RNN), long-short term memory (LSTM)
- Phương pháp xây dựng và giải quyết bài toán bạo lực trong video

Thực nghiệm:

- Xây dựng model kết hợp VGG19+LSTM trên tập huấn luyện
- Thực hiện huấn luyện với tập dữ liệu 2000 video gồm 1000 video bạo lực và 1000 video không bạo lực.
- Đánh giá mô hình dựa trên các độ đo: accuracy, precision, recall

Kết quả:

- Mô hình đạt độ chính xác khoảng 90% trên tập dữ liệu thử nghiệm (testing set).
- Độ chính xác cao hơn so với mô hình truyền thống (VGG16+LSTM).
- Xây dựng web app dự đoán bạo lực video.

Links:

- Github: https://github.com/pywind/violence_streamlit
- Drive:
<https://drive.google.com/drive/folders/1N92Pl6v5tBYQtFDvyojMMEIja-kdbR0n>

2. Ưu điểm:

- Triển khai lên web app để người dùng có thể sử dụng
- Tìm kiếm phương pháp giải quyết bài toán
- Tốc độ của nhận diện luôn phản hồi ở mức khoảng từ 1-2 giây

3. Khuyết điểm:

- Chưa bao quát được hết các hành động trong thực tế
- Độ chính xác chỉ mới 90%, chưa thể áp dụng vô thực tế
- Nền tảng triển khai còn đơn giản

4. Đề nghị cho bảo vệ hay không?

5. Đánh giá loại:

Tp. Hồ Chí Minh, ngày 04 tháng 01 năm 2023

Giáo viên hướng dẫn

(Ký & ghi rõ họ tên)

TS. Nguyễn Thiên Bảo

Lời cảm ơn

Lời đầu tiên, nhóm thực hiện xin gửi đến thầy giáo, Th.S Quách Đình Hoàng - giảng viên hướng dẫn lời cảm ơn chân thành và sâu sắc. Nhóm thực hiện xin cảm ơn sự quan tâm và giúp đỡ tận tình của thầy trong suốt quá trình nhóm thực hiện nghiên cứu đề tài. Cảm ơn thầy đã luôn giải đáp những thắc mắc cũng như đưa ra những nhận xét, góp ý giúp nhóm thực hiện cải thiện chất lượng công việc của nhóm. Với sự chỉ bảo của thầy, nhóm chúng em đã có những định hướng tốt trong việc triển khai và thực hiện các yêu cầu trong quá trình làm báo cáo đề tài “Violence detection in video.” Vì khả năng còn hạn chế nên trong quá trình thực hiện báo cáo không tránh khỏi sai sót, kính mong nhận được những ý kiến đóng góp từ thầy để nhóm có thể cải thiện hơn sau này.

Nhóm thực hiện xin chân thành cảm ơn.

Mục lục

Mục lục	6
Danh mục từ viết tắt	8
Danh mục hình ảnh.....	9
Phần 1: Mở đầu.....	10
1.1. Lý do chọn đề tài.....	10
1.2. Mục tiêu của đề tài	10
1.3. Đối tượng và phạm vi nghiên cứu.....	10
1.3.1. Đối tượng nghiên cứu	10
1.3.2. Phạm vi nghiên cứu	10
1.4. Phương pháp nghiên cứu.....	11
1.5. Kết quả dự kiến	11
1.6. Bố cục của báo cáo.....	11
Phần 2: Nội dung	12
Chương 1: Tổng quan về mạng neural.....	12
1.1 Giới thiệu về deep learning.....	12
1.2. Sơ lược về mạng neural sinh học.....	12
1.3. Sơ lược về mạng neural nhân tạo	13
1.4. Mô hình kiến trúc mạng neural.....	13
Chương 2: Convolutional Neural Network (CNN).....	16
2.1. Tổng quan về CNN.....	16
2.2. Convolution operator	16
2.3. ConvNet layers	21
2.4. VGG.....	23
Chương 3: Recurrent Neural Network (RNN).....	25
3.1. Sequence data	25
3.2. Mô hình bài toán RNN	26
3.3. Long-short term memory (LSTM).....	28
3.4. Các lớp RNN sử dụng trong các framework về deep learning.....	30
Chương 4: Các quá trình huấn luyện mô hình	33
4.1. Quá trình Feedforward.....	33
4.2. Quá trình Backpropagation.....	34
4.3. Transfer Learning	35

Chương 5: Nhận diện bạo lực trong video bằng các mô hình neural nhân tạo.....	38
5.1. Tổng quan về bài toán nhận diện bạo lực trong video.....	38
5.2. Phương pháp xử lý dữ liệu.....	39
5.3. Xây dựng giải quyết bài toán.....	41
5.4. Đánh giá mô hình.....	44
5.5. Thảo luận và so sánh.....	48
5.6. Kết luận và công trình tương lai	48
Chương 6: Ứng dụng nhận diện bạo lực trong video bằng Streamlit	50
6.1. Đặt vấn đề	50
6.2. Streamlit.....	50
6.3. Giới thiệu ứng dụng	50
PHẦN 3: Kết luận và thảo luận.....	52
1. Kết quả đạt được	52
2. Hạn chế.....	52
3. Hướng phát triển	53
References	54

Danh mục từ viết tắt

Ký hiệu chữ viết tắt	Ký hiệu đầy đủ	Ý nghĩa (nếu có)
App	Application	Ứng dụng
CNN	Convolutional Neural Network	Mạng neural tích chập
RNN	Recurrent Neural Network	Mạng thần kinh tái tạo
LSTM	Long-short term memory	

Danh mục hình ảnh

Hình 1 : Mô hình neural thần kinh	12
Hình 2 : Mô hình perceptron gồm	13
Hình 3 : Mô hình perceptron đa tầng	14
Hình 4 : Ảnh màu là một tensor ba chiều	16
Hình 5 : Mô phỏng 3D filter cho data nhiều channel.....	17
Hình 6 : Mô tả việc thêm padding có tác dụng giữ lại kích thước	20
Hình 7 : Mô tả bước nhảy của kernel ứng với stride=2	20
Hình 8 : Mô hình Convolution Neural Network	21
Hình 9 : Các loại pooling	22
Hình 10 : Biến đổi ma trận thành vector thông qua fully connected layer	23
Hình 11 : Mạng VGG 16 vs VGG 19	24
Hình 12 : Phân loại các bài toán RNN	25
Hình 13 : Mô hình RNN	26
Hình 14 : Mô hình RNN rút gọn	27
Hình 15 : Kiến trúc của một memory cell.	29
Hình 16 : Module lặp lại bên trong LSTM: x_t và h_t lần lượt là vector đầu vào và kết quả đầu ra vào ô nhớ tại thời điểm t . h_t là giá trị của ô nhớ. ft và ot lần lượt là giá trị của input gate, forget gate và output gate tại thời điểm t . Ct là các giá trị của trạng thái tạm thời của ô nhớ tại thời điểm t	29
Hình 17 : Mô hình RNN sử dụng Dense layer và TimeDistributed	31
Hình 18 : Quá trình feedforward	34
Hình 19 : Quá trình backpropagation	35
Hình 20 : Mô hình VGG 16	36
Hình 21 : Mô hình VGG 16 sử dụng fine tuning	37
Hình 22 : Hình ảnh video bạo lực được tách thành các hình ảnh đơn lẻ.....	39
Hình 23 : Hình ảnh video không bạo lực được tách thành các hình ảnh đơn lẻ.....	39
Hình 24 : Feature extractor sử dụng kiến trúc của mô hình VGG-19	41
Hình 25 : Kiến trúc thành phần LSTM và fully connected layer	42
Hình 26 : Model accuracy ứng với 15 epoch đầu tiên trên tập fold-1 training.....	45
Hình 27 : Model loss ứng với 15 epoch đầu tiên ứng với tập fold-1 training.....	45
Hình 28 : Confusion matrix trên tập dữ liệu testing (kiểm thử) với true label và predicted label	47
Hình 29 : Confusion matrix ứng với tập testing với các độ đo khác nhau	47
Hình 30 : Giao diện tương tác của ứng dụng	51
Hình 31 : Ứng dụng dự đoán bạo lực trả về kết quả cho người dùng	51

Phần 1: Mở đầu

1.1. Lý do chọn đề tài

Các hành vi bạo lực đang hiện hữu, công khai tại rất nhiều nơi và trở thành một mối đe dọa đến an ninh và các vấn đề xã hội. Tham lam cá nhân, thất vọng, tức giận và là những động lực cơ bản của sự gia tăng bạo lực.¹ Các hành vi này được ghi nhận lại thông qua nhiều camera quan sát, từ các thiết bị cá nhân của người qua đường, cảnh sát giao thông. Từ nguồn dữ liệu đó, với sự phát triển của các phương pháp khai thác dữ liệu, tính năng phát hiện bạo lực tự động có khả năng phát hiện nhanh chóng trong trường hợp bạo lực, giảm thiểu chậm trễ sự can thiệp tới các hành vi bạo lực này. [1]

Hướng tiếp cận chủ yếu với các dạng dữ liệu là hình ảnh và video thường là các mô hình học sâu (deep learning). Học sâu đem lại hiệu quả tối ưu và nhanh chóng có thể triển khai lên các thiết bị ngoại vi để hỗ trợ việc phát hiện nhanh chóng các hành vi bạo lực. Vì vậy, nhóm quyết định chọn đề tài “Phát hiện bạo lực trong video” thông qua hướng tiếp cận bằng học sâu (deep learning).

1.2. Mục tiêu của đề tài

Trong đề tài này, chúng tôi tập trung tìm hiểu về các mô hình xử lý dữ liệu dạng video và các mô hình học sâu (deep learning) trong tập dữ liệu Real Life Violence Situations Dataset², từ đó đưa ra mô hình tối ưu và xây dựng ứng dụng triển khai trên nền tảng website.

1.3. Đối tượng và phạm vi nghiên cứu

1.3.1. Đối tượng nghiên cứu

Hành động bạo lực trong các video được nhận biết thông qua các mô hình học sâu (deep learning), thường là một chuỗi hành động trong một khoảng thời gian nhất định.

1.3.2. Phạm vi nghiên cứu

Phạm vi: các mô hình, dữ liệu được tìm hiểu và huấn luyện trên các nền tảng Kaggle.

Phạm vi thời gian: trong khoảng 18 tuần từ 10/8/2022 đến 10/12/2022

Phạm vi nội dung: trong nghiên cứu này, chúng tôi quan tâm chủ yếu tới việc tìm hướng giải quyết bài toán thông qua việc xử lý dữ liệu đầu vào dạng video, sử dụng các mô hình đơn giản và tối ưu cho việc dự đoán bài toán và cuối cùng là đưa mô hình bài toán triển khai lên ứng dụng thực tế.

¹ Kết quả được dựa theo bài báo đã được trích dẫn tại 1. Singh, S., et al., *Video Vision Transformers for Violence Detection*. 2022: p. arXiv: 2209.03561.

² Nguồn dữ liệu: <https://www.kaggle.com/datasets/mohamedmustafa/real-life-violence-situations-dataset>

1.4. Phương pháp nghiên cứu

Phương pháp thu thập số liệu: Tìm kiếm và tổng hợp thông tin, kiến thức, lý thuyết từ các nguồn đã có sẵn thông qua các bài báo, nghiên cứu có sẵn hoặc trên internet.

Phương pháp toán học: Giải thích các mô hình toán học sử dụng bằng sự hiểu biết của sinh viên.

Phương pháp thực nghiệm: Thử nghiệm và tìm hiểu các mô hình thông qua việc huấn luyện và đánh giá thông qua các metric cụ thể.

1.5. Kết quả dự kiến

Kết quả mô hình: Mô hình đạt hiệu suất khoảng 92% khi được dự đoán trên các video được lấy từ các nguồn bên ngoài dữ liệu.

Kết quả ứng dụng: Triển khai mô hình trên nền tảng Internet (web app).

1.6. Bố cục của báo cáo

Chương 1: Tổng quan về deep learning, neural cũng như kiến trúc và mô hình mạng neural

Chương 2: Tổng quan, chức năng và kiến trúc của một convolutional neural network (CNN)

Chương 3: Tổng quan về recurrent neural network (RNN), mô hình và các thuật toán

Chương 4: Các quá trình huấn luyện mô hình

Chương 5: Xây dựng, giải quyết bài toán “nhận diện bạo lực” thông qua video

Chương 6: Triển khai, ứng dụng kết quả của bài toán “nhận diện bạo lực” sử dụng Streamlit

Phần 2: Nội dung

Chương 1: Tổng quan về mạng neural

1.1 Giới thiệu về deep learning

Trước khi bắt đầu tìm hiểu về deep learning, cần biết về machine learning (học máy) là gì, sự khác nhau cơ bản của machine learning và deep learning.

Machine learning là thuật ngữ để chỉ việc dạy cho máy tính cách tự học hỏi, giải quyết và đưa ra kết quả dựa trên dữ liệu vào mà không cần con người lập trình một cách cụ thể.

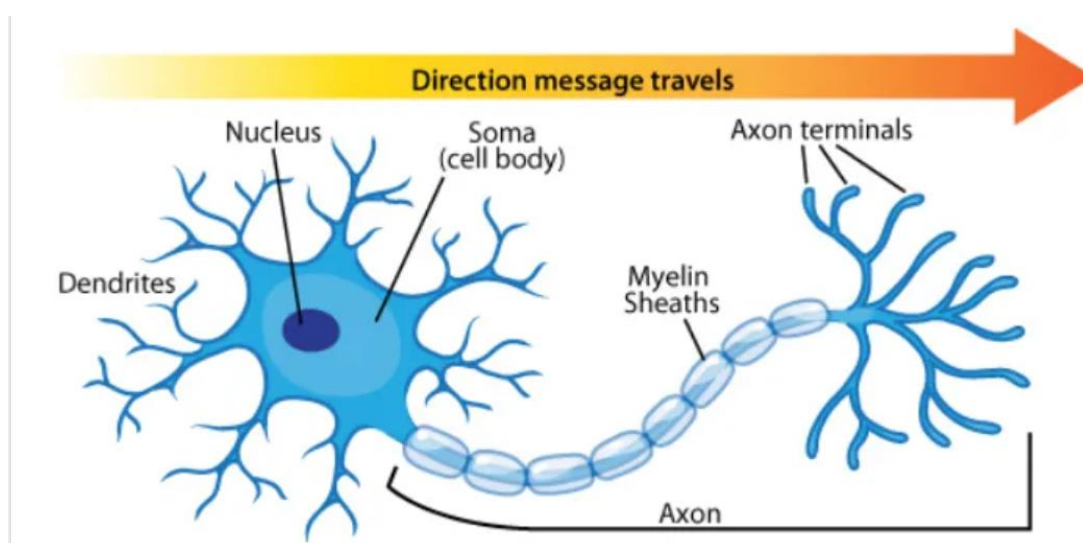
Deep learning (học sâu) là một nhánh sâu hơn của học máy và thiên về mảng trí tuệ nhân tạo hơn. Nó có thể giải quyết các vấn đề sâu hơn mà học máy không giải quyết được.

Hiện nay, một số các ứng dụng có sử dụng học sâu đáng kể tới như là: xe tự lái, trợ lý ảo...

1.2. Sơ lược về mạng neural sinh học

Nguồn gốc và ý tưởng của mạng neural bắt nguồn từ neural thần kinh hay còn gọi là neural sinh học.

Neural là thành phần cơ bản cấu tạo nên hệ thống thần kinh của con người, các bộ phận chính bao gồm: thân tế bào (soma) chứa nhân tế bào, tiếp đến là sợi trục dài có tác dụng truyền dẫn tín hiệu từ thân tế bào sang các tế bào thần kinh khác và cuối cùng là các sợi phân ra khỏi tế bào được gọi là các sợi nhánh (đuôi gai) có tác dụng tiếp nhận các xung thần kinh khác.



Hình 1: Mô hình neural thần kinh³

³ <https://askabiologist.asu.edu/neuron-anatomy>

Hệ thống thần kinh con người có khoảng 10 triệu các neural, và mỗi neural lại liên kết với những neural khác thông qua các sợi nhánh, tạo nên mạng lưới neural. Một nhánh của mạng được minh họa trên hình số 1.

1.3. Sơ lược về mạng neural nhân tạo

1.3.1. Đặc điểm của mạng neural nhân tạo

Hệ thống mạng neural nhân tạo được xây dựng dựa trên mô hình mạng neural sinh học, trong đó mỗi neural lại ứng với một hàm toán học hay một thuật toán riêng.

Mạng neural nhân tạo là tập hợp một chuỗi các thuật toán khác nhau để xử lý các thông tin đầu vào và đưa ra kết quả cuối cùng một cách tối ưu nhất.

1.3.2. Công dụng của mạng neural nhân tạo

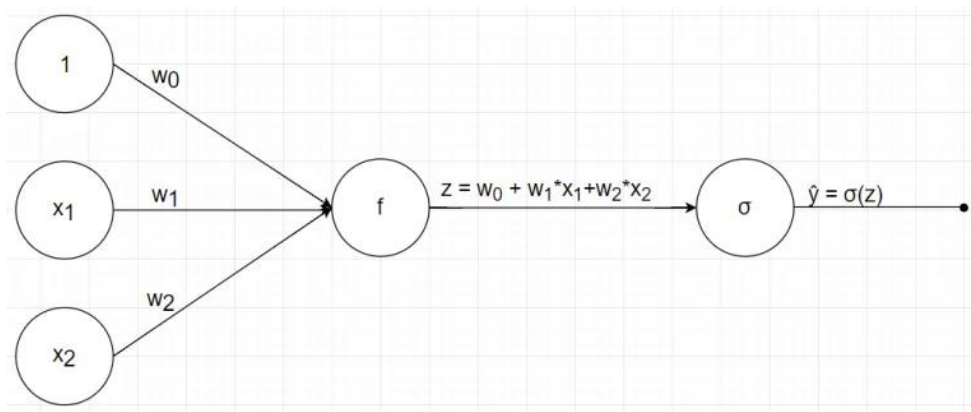
Mạng neural là một mô hình toán học phức tạp được đưa ra để xử lý các thông tin, giải quyết các bài toán, các vấn đề phổ biến trong lĩnh vực học máy, trí tuệ nhân tạo và deep learning (học sâu).

Được áp dụng rộng rãi trong nhiều lĩnh vực, như trong lĩnh vực tài chính, mạng neural nhân tạo hỗ trợ cho quá trình phát triển các quy trình như: các thuật toán giao dịch, các dự báo chuỗi thời gian, phân loại chứng khoán, các mô hình rủi ro tín dụng, phát sinh giá cả...

Ngoài ra, còn được sử dụng khá rộng rãi cho những hoạt động khác như: dự báo thời tiết, tìm kiếm các giải pháp nhằm nghiên cứu tiếp thị, đánh giá rủi ro và phát hiện gian lận...

1.4. Mô hình kiến trúc mạng neural

1.4.1. Mô hình perceptron đơn tầng



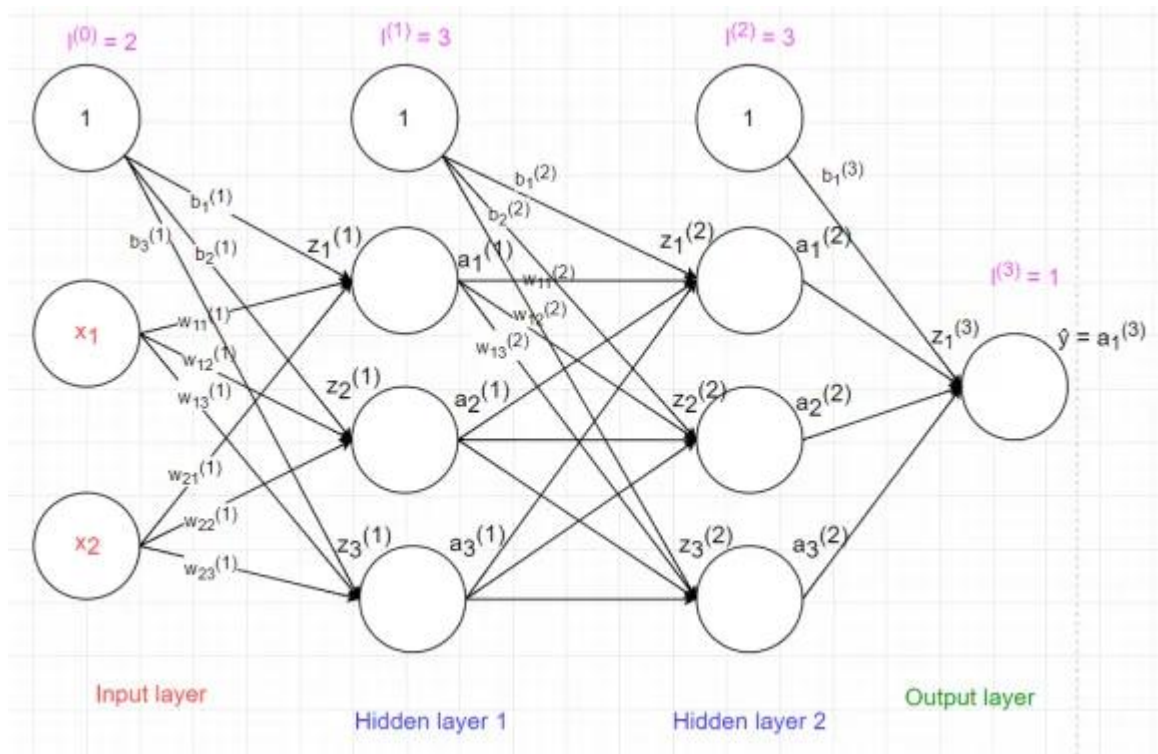
Hình 2: Mô hình perceptron gồm ⁴

Hình số 2 có mô tả về mô hình mạng neural đơn giản nhất, logistics regression hay còn gọi là mô hình perceptron đơn tầng, sở dĩ gọi là mô hình đơn giản nhất vì nó chỉ có duy nhất input layer và output layer.

⁴ <https://nttuan8.com/bai-3-neural-network/>

- Từ các input, tính tổng linear với hệ số bias w_0
- Hàm f là tổng của các node trong input layer nên: $f = 1 * w_0 + w_1 * x_1 + w_2 * x_2$
- Sau đó sử dụng hàm activation function, trong trường hợp này là sigmoid function cho hàm f : $\hat{y} = \sigma(z)$
- Sigmoid function: được sử dụng để chuyển một số thực về khoảng từ (0 - 1), nếu đầu vào là một số âm rất nhỏ thì đầu ra sẽ tiệm cận 0 và đầu vào là một số dương rất lớn thì đầu ra sẽ tiệm cận 1.

1.4.2. Mô hình perceptron đa tầng



Hình 3: Mô hình perceptron đa tầng⁵

Mô hình neural network là sự kết hợp của nhiều tầng perceptron lại với nhau hay còn gọi là perceptron đa tầng. Mỗi mô hình perceptron đa tầng gồm các lớp:

- Lớp đầu tiên là input layer
- Lớp tiếp theo là hidden layer, có thể có nhiều hidden layer trong một mô hình
- Lớp cuối cùng là output layer

Tổng số layer của mô hình được tính theo quy ước: Tổng số layer - 1, bởi vì input layer là không được tính.

⁵ <https://nttuan8.com/bai-3-neural-network/>

Mỗi node trong hidden layer và output layer sẽ kết nối với tất cả các node ở layer trước đó với các hệ số w riêng biệt, đồng thời mỗi node sẽ có một hệ số bias riêng và đều diễn ra các bước tính tổng linear và activation function.

Kí hiệu tổng quát về mô hình perceptron đa tầng:

- Số node trong hidden layer thứ i là $L^{(i)}$

Ma trận $W^{(k)}$ kích thước $L^{(k-1)} * L^{(k)}$ là ma trận hệ số giữa layer $(k-1)$ và layer k , trong đó $w_{ij}^{(k)}$ là hệ số kết nối từ node thứ i của layer $k-1$ đến node thứ j của layer k

Vector $b^{(k)}$ kích thước $L^k * 1$ là hệ số bias của các node trong layer k , trong đó $b_i^{(k)}$ là bias của node thứ i trong layer k

Với mỗi node i trong layer thứ I , thực hiện 2 bước là tính tổng linear và áp dụng activation function

Công thức tổng quát tính linear với layer thứ I :
$$z_i^{(I)} = \sum_{j=1}^{L^{(I-1)}} a_j^{(I-1)} * w_{ij}^{(I)} + b_i^{(I)}$$

Công thức activation function:
$$a_i^{(I)} = \sigma(z_i^{(I)})$$

Vector $Z^{(k)}$ kích thước $(L^{(k)} \times 1)$ là giá trị các node trong layer k sau bước tính tổng linear.

Vector $a^{(k)}$ kích thước $(L^{(k)} \times 1)$ là giá trị của các node trong layer k sau khi áp dụng hàm activation function.

Với ví dụ hình 3 bên trên, mô hình perceptron có 3 layer, gồm input layer (2 node hay $L^{(0)} = 2$), hidden layer1 (3 node), hidden layer2 (3 node) và cuối cùng là output layer (1 node).

Do mỗi node trong hidden layer và output layer đều có bias nên trong input layer và hidden layer cần thêm node 1 để tính bias.

Áp dụng công thức tổng quát linear cho node thứ 2 ở hidden layer 1, có:

$$z_2^{(1)} = x_1 * w_{12} + x_2 * w_{22} + b_2^{(1)}$$

$$a_2^{(1)} = \sigma(z_2^{(1)})$$

Các node khác cũng áp dụng tương tự.

Đây là một dạng lan truyền tiến (feedforward), sẽ được đề cập ở các chương sau. Từ input layer a^0 dẫn đến kết quả $z^{(1)}$ đến $a^{(1)}$ và cứ thế cho tới $a^{(3)}$ và đó cũng chính là output đầu ra mà mô hình đã dự đoán.

Chương 2: Convolutional Neural Network (CNN)

2.1. Tổng quan về CNN

Convolutional Neural Network (CNN) được hiểu là Mạng neural tích chập được sử dụng nhiều trong các mô hình computer vision và các bài toán deep learning hay mạng neural. Một số ứng dụng của CNN liên quan đến computer vision bao gồm phân loại ảnh (image classification), phân vùng hình ảnh (image semantic segmentation), nhận dạng vật thể (object detection) trong hình ảnh hoặc video, phân tích hình ảnh ...

2.2. Convolution operator

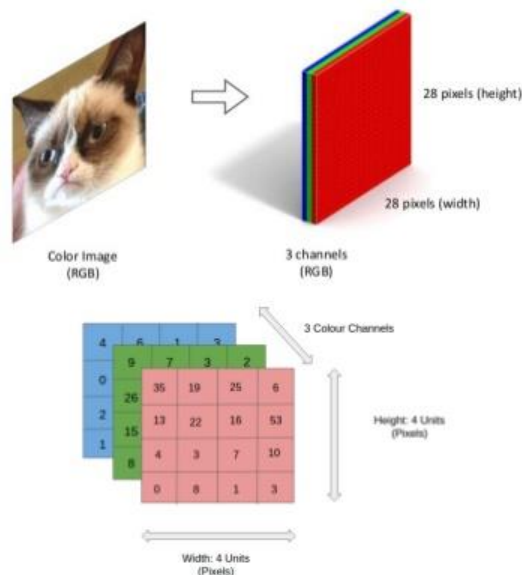
2.2.1. Định nghĩa

Convolution hay tích chập chính là phép *Element-wise multiplication*⁶ giữa ma trận con của tập dữ liệu X với các sliding window (cửa sổ trượt) còn được gọi là kernel, filter hoặc feature detect (W).

Operator là thực hiện hành động trên các Sliding Window lần lượt trên các ma trận con của X có cùng kích thước với W . Kết quả của phép tính này thu được một layer Y có các đặc trưng của X .

Dữ liệu X đầu vào thường là dạng ảnh hoặc tập hợp ảnh (video) là dạng tensor hai chiều có 3 ma trận channel màu red, green, blue chồng lên nhau

color image is 3rd-order tensor



Hình 4: Ảnh màu là một tensor ba chiều⁷

Về mặt ký hiệu của phép convolution: $Y = X * W$ (⁸)

⁶ Hay còn gọi là Hadamard product, theo [https://en.wikipedia.org/wiki/Hadamard_product_\(matrices\)](https://en.wikipedia.org/wiki/Hadamard_product_(matrices))

⁷ Theo nguồn <https://www.slideshare.net/BertonEarnshaw/a-brief-survey-of-tensors>

⁸ Theo nguồn [https://en.wikipedia.org/wiki/Kernel_\(image_processing\)](https://en.wikipedia.org/wiki/Kernel_(image_processing))

Trong đó: Y là tensor (hoặc matrix) kết quả sau khi thực hiện convolution

X là tensor (matrix) đầu vào

W là filter (hoặc kernel) của biểu thức

Với layer thứ l là một convolution layer, thì hình dạng (shape) của layer mà một tensor tuân theo quy luật sau, trích từ [2]: ⁹

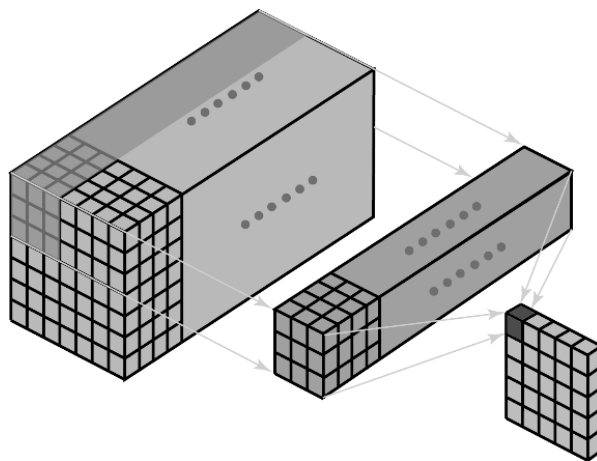
- Input: $n_h^{[l-1]} \times n_w^{[l-1]} \times n_c^{[l-1]}$
- Output: $n_h^{[l]} \times n_w^{[l]} \times n_c^{[l]}$
- Trong đó: $n_w^{[l]} = \left\lfloor \frac{n_w^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} \right\rfloor + 1$; $n_h^{[l]} = \left\lfloor \frac{n_h^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} \right\rfloor + 1$ (*)
- Với: $n_w = width, n_h = height, n_c = color\ of\ channels$
- $f^{[l]} = filter\ size\ in\ square$
- $p^{[l]} = padding$
- $s^{[l]} = stride$

2.2.2. Kernel

Kernel (hay còn được gọi là filter) là một ma trận vuông có kích thước $k \times k$ trong đó $k \in \{x \in \mathbb{N} \mid 2x + 1\}$ là một số lẻ¹⁰

Còn filter là một tensor ba chiều gồm n kernel với số chiều dữ liệu đầu vào. Ví dụ: trong xử lý hình ảnh màu, với hệ màu RGB được biểu diễn thành ba giá trị (red, green, blue) thì được quy đổi tương ứng thành ba channel ứng với 3D của filter.

Trong mạng neural tích chập (Convolutional neural network), kernel được sử dụng để trích xuất các đặc điểm (features) hữu ích từ dữ liệu (hình ảnh, âm thanh, ...). Kernel di chuyển trên dữ liệu đầu vào, thực hiện phép convolution với vùng của dữ liệu có cùng kích thước với kernel.



Hình 5: Mô phỏng 3D filter cho data nhiều channel

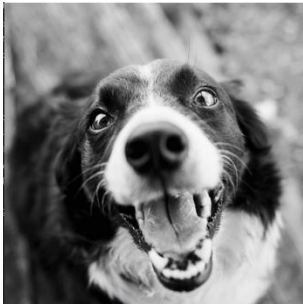


⁹ Theo <https://cs231n.github.io/convolutional-networks/#overview>

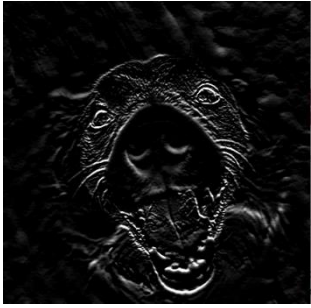


¹⁰ Khi kernel là một ma trận kích thước lẻ, đơn vị chính giữa sẽ là đơn vị mục tiêu để trích xuất đặc trưng. Việc sử dụng kernel chẵn sẽ dẫn đến tình trạng output không thể kiểm soát kích thước và về mặt logic gây ra nhiễu cho output.

Hình 5 biểu thị cách thức mà một kernel 3x3 với tính tích chập trên một đơn vị dữ liệu cùng số n chiều để tạo thành một ma trận trích xuất những đặc điểm của đơn vị dữ đó.

Một số loại kernel thường được sử dụng với một số mục đích khác nhau như: làm mờ ảnh, làm nét ảnh, xác định các vân sáng trên hình ảnh, ...

Ví dụ: Bảng mô tả kết quả do tác giả thực hiện

Mục đích	Kernel	Kết quả
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	 <p><i>Fig 1: ảnh gốc chú chó</i></p>
Edge detection	Laplacian $\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	 <p><i>Fig 2: Ảnh laplacian của chú chó</i></p>
	Sobel $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$	 <p><i>Fig 3: Ảnh sobel của chú chó</i></p>

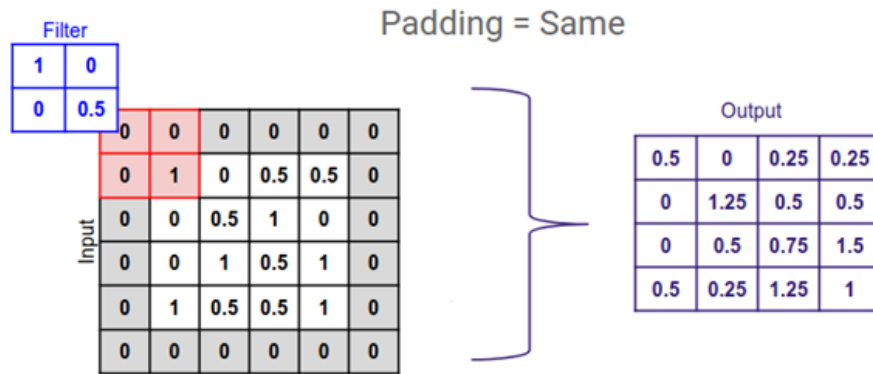
	Prewitt $\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$	 <p><i>Fig 4: Ảnh prewitt của chú chó</i></p>
Sharper	Sharpen $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	 <p><i>Fig 5: Ảnh làm sắc nét</i></p>
Blur	Small blur $\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	 <p><i>Fig 6: Ảnh làm mờ</i></p>

Bảng 1: Kết quả áp dụng của từng loại kernel trên một hình ảnh chú chó, do tác giả thực hiện

2.2.3. Padding

Trong công thức convolution (*), nếu không có padding (p), thì kết quả output Y thu được sẽ có số chiều giảm đi với kích thước của input X. Vì vậy, padding được thêm vào nhằm mục đích giữ nguyên số chiều của output để phù hợp cho các mục đích phân tích.

Cách giải quyết thường là thêm các phần tử ở viền ngoài của dữ liệu input X.



Hình 6: Mô tả việc thêm padding có tác dụng giữ lại kích thước ¹¹

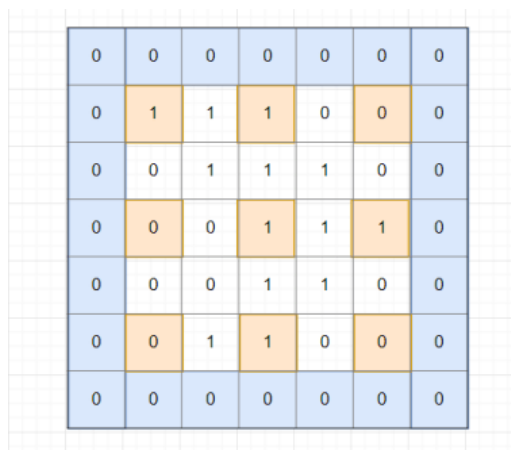
Với việc sử dụng padding = 1 như hình 6, ma trận dữ liệu ban đầu vẫn được giữ nguyên kích thước 4x4 sau khi được qua bước tính tích chập bằng ma trận 2x2.

Việc lựa chọn kích cỡ của padding còn dựa vào kích thước của giá trị W và giá trị đầu vào. Nếu không sử dụng stride, thì padding sẽ được tính: $p = \frac{f - 1}{2}$ ¹² với f = kích thước của kernel [2]

2.2.4. Stride

Stride được điều chỉnh để nén hình ảnh và dữ liệu video. Stride là một tham số của kernel (filter) chuyển động với một số lượng bước nhảy nhất định trên hình ảnh hoặc video theo cả chiều ngang và dọc.

Ví dụ: nếu đặt stride của mạng thần kinh được đặt thành 2, kernel sẽ di chuyển hai pixel hoặc đơn vị, tại một thời điểm.



Hình 7: Mô tả bước nhảy của kernel ứng với stride=2 ¹³

¹¹ Nguồn: <https://ayeshmanthaperera.medium.com/what-is-padding-in-cnns-71b21fb0dd7>

¹² Theo <https://cs231n.github.io/convolutional-networks/#layers> có đề cập

¹³ Nguồn: <https://nttuan8.com/bai-5-gioi-thieu-ve-xu-ly-anh/>

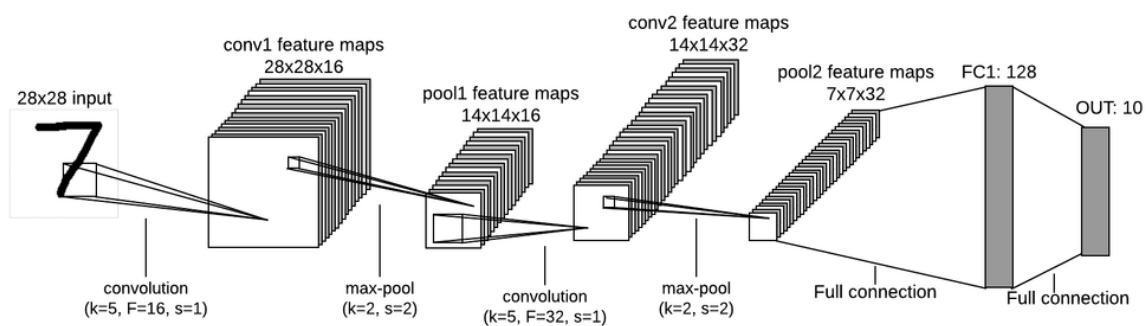
Khi áp dụng $\text{stride}=2$, convolution sẽ được tính như hình 7, cứ các 2 hai điểm dữ liệu theo cả chiều ngang và dọc thì chúng mới bắt đầu được tính convolution.

Nếu giá trị $\text{stride} = 1$ thì tương đương với việc tính convolution trên từng giá trị của ma trận dữ liệu.

Kích thước của kernel ảnh hưởng đến khối lượng đầu ra, do đó stride thường được đặt thành toàn bộ số nguyên, thay vì một phần hoặc thập phân.

2.3. ConvNet layers

Một ConvNet (Convolutional Network) là một chuỗi các layer có nhiệm vụ biến đổi dữ liệu thông qua các hàm biến đổi riêng biệt. Có ba loại layer chính để xây dựng nên ConvNet: convolutional layer, pooling layer, và fully connected layer.



Hình 8: Mô hình Convolution Neural Network ¹⁴

Trong hình minh họa 8, một ví dụ minh họa đơn giản cho một mô hình CNN cho bộ dữ liệu MNIST:

- INPUT [28 x 28] sẽ chứa giá trị của từng pixel của image, ảnh trắng đen theo tập dữ liệu MM
- CONV layer sẽ tính toán output của image theo từng kernel. Kích thước [28x28x16] tương ứng với việc sử dụng 16 kernel.
- POOL layer là bước để down sample để input đơn giản nhưng vẫn giữ nguyên được feature của dữ liệu. Dữ liệu được giảm về [14x14x16]
- CONV layer tiếp theo sử dụng 32 kernel để thu được dữ liệu có kích thước [14x14x32]
- POOL layer sẽ thực hiện hoạt động lấy mẫu xuống dọc theo kích thước, dẫn đến thể tích giảm còn [7x7x32]
- Fully connected layer kéo giãn dữ liệu thành vector có độ dài [1x1x128] sau đó tính toán class scores giữa 10 class của bộ MNIST data.

2.3.1. Convolutional layer

Như tên gọi, trong convolutional layer đóng vai trò quan trọng trong cách thức CNN vận hành. Layer đảm nhiệm việc tính toán convolution nhằm chiết xuất ra những

¹⁴ Nguồn: <https://www.easy-tensorflow.com/tf-tutorials/convolutional-neural-nets-cnns>

feature (tính chất) từ tập dữ liệu. Việc tính toán dựa vào các kernel được tập hợp thành filter, các kernel thường được tập hợp theo một độ sâu nhất định nhằm tối ưu hóa số lượng tham số truyền vào layer tiếp theo.

Theo [3], kích thước của kernel (filter) còn phụ thuộc vào receptive field size (vùng tiếp nhận) so với kích thước của input. Các hyperparameter bao gồm depth, stride và padding.

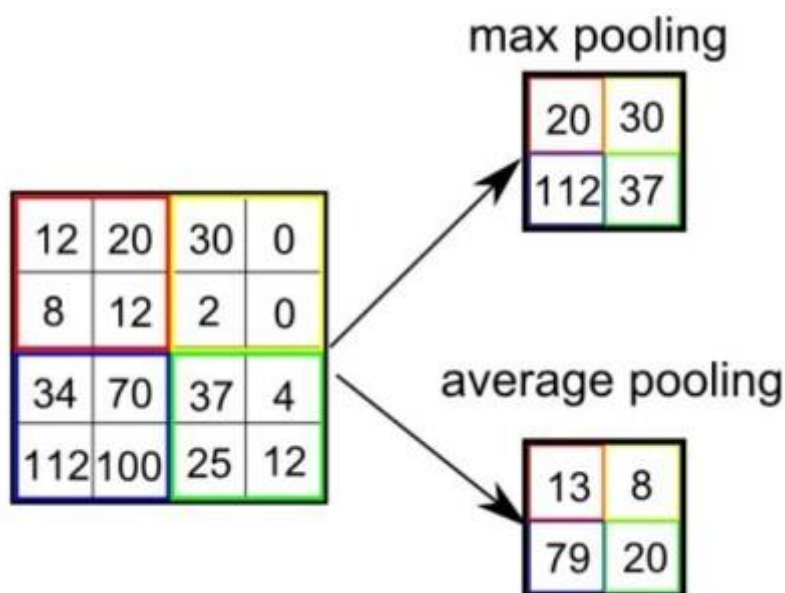
Depth (độ sâu) của output được tạo ra bởi các lớp hidden layer bên trong mạng neural. Giảm siêu tham số này có thể giảm thiểu đáng kể tổng số tế bào thần kinh của mạng, nhưng nó cũng có thể làm giảm đáng kể khả năng nhận dạng mẫu của mô hình.

Tham số stride và padding được sử dụng y nguyên như trên cách tính convolution. Chúng mang ý nghĩa làm giảm thiểu số chiều của dữ liệu hoặc tái tạo giữ nguyên số chiều tương ứng.

2.3.2. Pooling layer

Khác với cách giảm số chiều của dữ liệu thông qua bước nhảy (stride), nó có thể khiến một số thuộc tính quan trọng bị mất mát khi bước nhảy quá lớn. Pooling layer thường được sử dụng giữa các convolutional layer nhằm để giảm kích thước mà vẫn giữ nguyên các thuộc tính quan trọng.

Pooling size là ma trận có kích thước $k \times k$. Với mỗi vùng $k \times k$ trên dữ liệu đầu vào, chúng sẽ được tính toán **maximum** hoặc **average** của dữ liệu rồi tập hợp lại thành một ma trận kết quả. Các thực thi này sẽ áp dụng lên từng channel của dữ liệu.



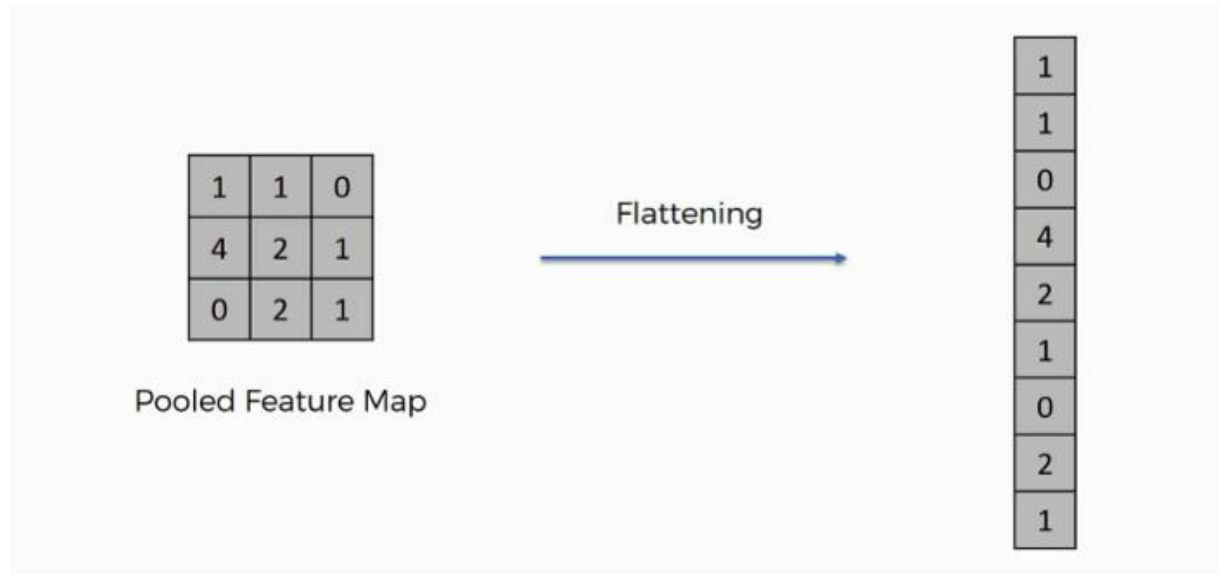
Hình 9: Các loại pooling¹⁵

Trong ví dụ hình 9: pooling size có $k=2$, mô tả ma trận input 4x4 chuyển về ma trận 2x2 thông qua pooling layer.

¹⁵ Nguồn: <https://nttuan8.com/bai-6-convolutional-neural-network/>

2.3.3. Fully connected layer

Sau khi data được xử lý qua nhiều convolutional layer và pooling layer, model đã có được nhiều đặc điểm của hình ảnh. Để chuyển hóa những thông tin này để dự đoán output, cần đưa chúng qua một mạng ANN (mạng neural truyền thống) thông qua fully connected layer.



Hình 10: Biến đổi ma trận thành vector thông qua fully connected layer¹⁶

Tóm lại, thông qua hình 10, một ma trận (tensor) nó sẽ được flatten (kéo giãn) thành các vector để truyền vào mạng ANN nhằm dự đoán ra output.

2.4. VGG

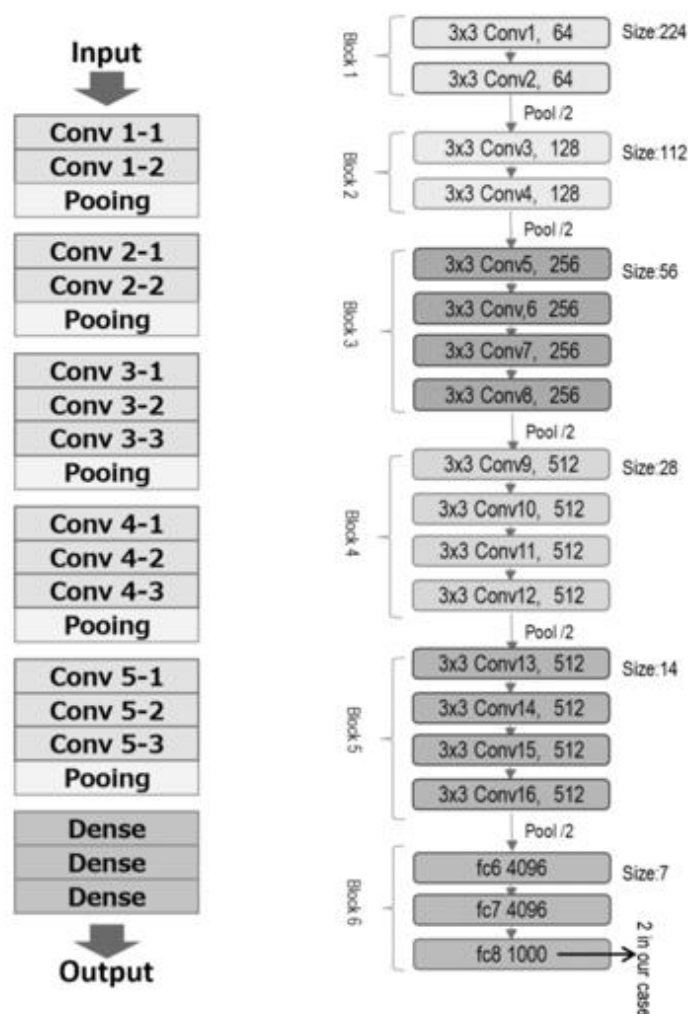
VGG (Visual Geometry Group) là một mô hình mạng CNN do K. Simonyan và A. Zisserman đề xuất trong bài báo “Very Deep Convolutional Networks for Large-Scale Image Recognition” [4]. Kiến trúc này đã đạt được độ chính xác 92,7% và đạt top 5 trên trang ImageNet¹⁷, có hơn 14 triệu hình ảnh thuộc 1000 lớp. Từ "Deep" đề cập đến số lượng các lớp với VGG-16 hoặc VGG-19 bao gồm 16 và 19 lớp convolution (tích chập).

Kiến trúc VGG là cơ sở của các mô hình nhận dạng đối tượng. Hơn nữa, bây giờ nó vẫn là một trong những kiến trúc nhận dạng hình ảnh phổ biến nhất.

So sánh một số mạng VGG thông dụng:

¹⁶ Nguồn: <https://i0.wp.com/nttuan8.com/wp-content/uploads/2019/03/flattern.png>

¹⁷ Theo <https://www.image-net.org/>



Hình 11: Mạng VGG 16 vs VGG 19 ¹⁸

Về cơ bản, mô hình mạng VGG-19 giống với mô hình mạng VGG-16 ngoại trừ việc VGG-19 hơn 3 lớp convolutional so với VGG-16 ở các block 3, block 4 và block 5 như trong hình 21. Bên cạnh đó, cả hai mô hình mạng đều bao gồm 5 block convolution giúp cho việc nhận dạng các sự vật rõ ràng và chính xác hơn. Các lớp convolutional để mô hình học được các thuộc tính của ảnh nên VGG-19 sẽ học được tốt hơn so với VGG-16. Song, do có nhiều lớp hơn nên chiếm nhiều không gian lưu trữ, chi phí tính toán, dẫn đến việc thực hiện chậm hơn và tiêu tốn bộ nhớ GPU hơn so với VGG-16.

¹⁸ Nguồn: <https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918>

Chương 3: Recurrent Neural Network (RNN)

3.1. Sequence data

Dữ liệu dạng sequence là một chuỗi dữ liệu có mối tương quan với nhau, chúng thường được chiết xuất từ những sự kiện xảy ra liên tiếp... VD: video, dữ liệu chứng khoán, biểu đồ thời tiết theo giờ.

Các dữ liệu này có thứ tự. Nếu thay đổi hoặc mất mát sẽ làm ý nghĩa của sự kiện này thay đổi. Một phần dữ liệu nhỏ trong dữ liệu dạng sequence không đại diện cho ý nghĩa của cả tập dữ liệu.

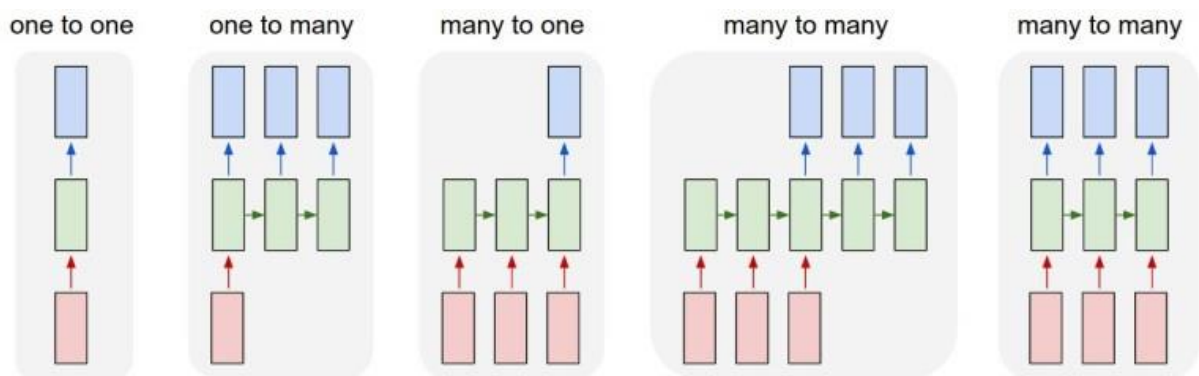
VD: Trong bài toán dự đoán giá chứng khoán bằng giá chứng khoán của những phiên làm việc trước đó. Input là dữ liệu của những lần khám trước đó. Thí dụ i_1 là phiên giao dịch ngày nhất, i_2 là phiên giao dịch của ngày thứ hai, ... Tập hợp của những phiên thành tạo thành chuỗi (i_1, i_2, \dots, i_n) , và nó được gọi là sequence data. RNN sẽ học từ những input và dự đoán giá cổ phiếu của phiên tiếp theo tăng hay giảm.

3.1.1. Recurrent Neural Network

Mạng thần kinh tái tạo (Recurrent Neural Network) – RNN là một mô hình mạng theo chuỗi (neural sequence model)

3.1.2. Phân loại bài toán RNN

Mô hình RNN gồm một số loại chính như sau:



Hình 12: Phân loại các bài toán RNN ¹⁹

- One to one

Mô hình One-to-One cho phép một input duy nhất và một output duy nhất. Cả hai thành phần được cố định. Ứng dụng của nó là Image Classification.

¹⁹ Nguồn: <https://nttuan8.com/bai-13-recurrent-neural-network/>

- One to many

Mô hình One to many cho phép một kích thước input cố định và cung cấp một chuỗi các output dữ liệu. Ứng dụng của nó có thể là Music generation và image captioning.

- Many to one

Mô hình được sử dụng khi một output yêu cầu nhiều đơn vị input hoặc một chuỗi dữ liệu input. Ứng dụng của nó được sử dụng trong Sentiment Analysis

- Many to many

Mô hình có nhiều input đầu vào và cũng có nhiều output. Ứng dụng của nó trong việc dịch ngôn ngữ

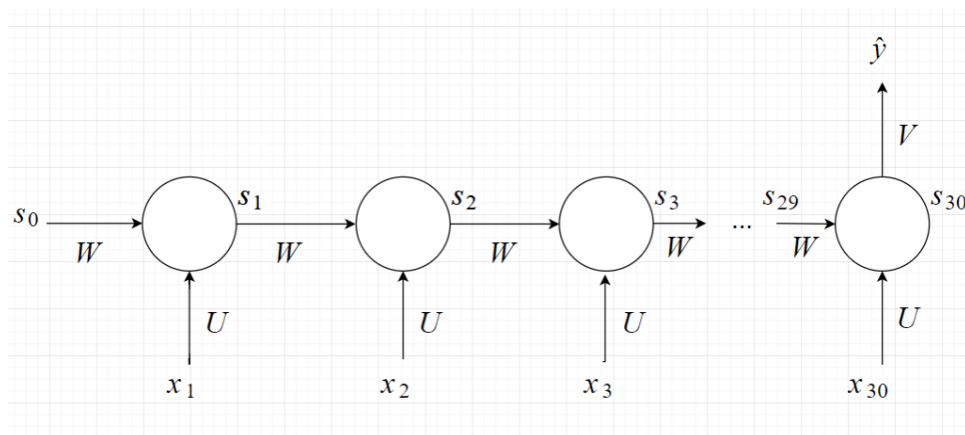
3.2. Mô hình bài toán RNN

3.2.1. Tổng quát

Một mô hình RNN bao gồm hai thành phần chính: Input và output.

Input bao gồm một chuỗi sequence ảnh. Các hình ảnh này đã được xử lý thông qua mô hình CNN (nhằm lấy ra những feature) rồi chuyển thành vector có kích thước $n \times 1$. Mỗi vector thứ i (x_i) tương ứng với hình ảnh thứ i trong mỗi giây.

Output là vector có kích thước $d \times 1$ được softmax function xử lý.



Hình 13: Mô hình RNN²⁰

Mô hình RNN như hình 12 bao gồm nhiều cell, mỗi cell có:

+ Input: vector x_t tại thời điểm t

²⁰ Nguồn: https://nttuan8.com/bai-13-recurrent-neural-network/#Ung_dung_bai_toan_RNN

+ Input s_{t-1} (state của cell trước đó) là state tại thời điểm $t - 1$

+ output là state: $s_t = f(U * x_t + W * s_{t-1})$ [5]

Hàm f là activation function (thường là hàm ReLU, sigmoid, tanh). Init state s_0 là state khởi tạo và thường được đặt là 0

+ Kết quả dự đoán \hat{y} là output của cell cuối cùng, khi đó: $\hat{y} = g(V * s_n)$ Với g là softmax function.

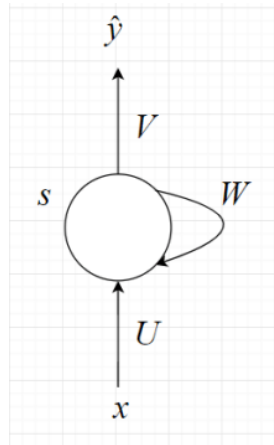
+ U là ma trận có kích thước $m \times n$ và V là ma trận có kích thước $d \times m$ là trọng số của hidden layer và output layer

3.2.2. Loss function (hàm mất mát) và đạo hàm của RNN

Như đã trình bày, output \hat{y} là một softmax function, vậy nên giá trị hàm loss sẽ được tính toán bởi: ²¹

$$L = - \sum_{i=1}^m y_i \cdot \log(\hat{y}_i)$$

Để mô tả tổng quát phương trình, mô hình RNN được rút gọn lại trở thành.



Hình 14: Mô hình RNN rút gọn²²

Với S, V, W, U lần lượt là các tham số được mô tả như hình 13.

Khi đó, có 3 tham số cần tính toán là W, V và U , tương ứng với: $\frac{\partial L}{\partial W}, \frac{\partial L}{\partial V}, \frac{\partial L}{\partial U}$

²¹ https://en.wikipedia.org/wiki/Loss_functions_for_classification#Logistic_loss

²² Nguồn: <https://nttuan8.com/bai-13-recurrent-neural-network/>

Sử dụng Backpropagation Through Time [6]:

Ta có:

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

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial s_n} * \frac{\partial s_n}{\partial W} \quad (*)$$

Mà s_n hay $s_t = f(U * x_t + W * s_{t-1})$

Từ (*), Theo product rule:

$$\frac{\partial s_n}{\partial W} = s_{n-1} + \frac{\partial s_n}{\partial s_{n-1}} * \frac{\partial s_{n-1}}{\partial W} = s_{n-1} + s_{n-2} + \frac{\partial s_{n-1}}{\partial s_{n-2}} * \frac{\partial s_{n-2}}{\partial W} = \sum_{i=2}^n \frac{\partial s_i}{\partial s_{i-1}} * \frac{\partial s_i}{\partial W} \quad (**)$$

Với việc sử dụng s_{i-1} là hệ số khi đạo hàm s_i

Từ (*) và (**)

$$\frac{\partial L}{\partial W} = \sum_{t=0}^n \frac{\partial L}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial s_n} * \left[\prod_{i=t}^{n-1} \frac{\partial s_i}{\partial s_{i-1}} \right] * s_t$$

Ta có thể thấy, việc tính backpropagation theo thời gian nhằm tính ra các weight là công việc tính toán từng state trước thời điểm t . Điều này tạo nên sự phụ thuộc đạo hàm trong hàm Loss và khiến cho việc tính toán đạo hàm trở nên rất phức tạp. Vì vậy, các thuật toán mới sinh ra để tối ưu hóa vấn đề này.

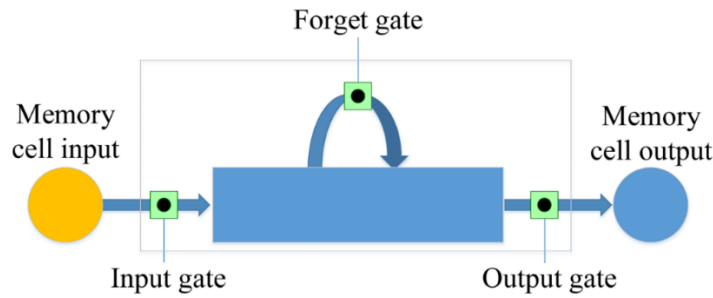
3.3. Long-short term memory (LSTM)

Long-short term memory (LSTM) là một trong nhiều biến thể kiến trúc RNN. Mô hình RNN có đặc điểm là sự phụ thuộc vào nhiều state trước đó dẫn tới hiện tượng vanishing gradient [5]²³. LSTM là một mô hình hiệu quả để chống lại vanishing gradient bằng cách sử dụng các memory cell.

Về mặt lý thuyết là RNN có thể mang các thông tin từ layer trước đến layer sau. Tuy nhiên thực tế là thông tin chỉ có thể truyền qua một số lượng state nhất định, sau

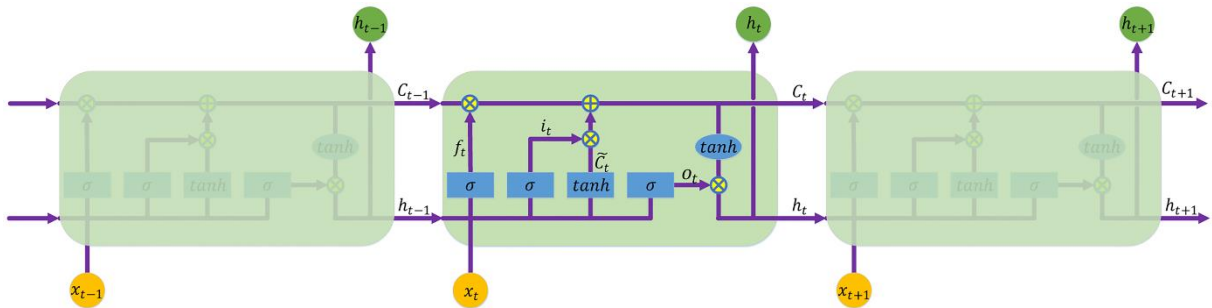
²³ Hiện tượng gradient sẽ nhỏ dần khi đi qua nhiều layer trong quá trình đạo hàm, việc này khiến cho gradient descent không làm thay đổi nhiều các weights của các layer đó, khiến chúng khó hội tụ hoặc thậm chí không thể hội tụ. Điều này dẫn tới kết quả huấn luyện toàn mạng không có kết quả.

đó thì sẽ bị vanishing gradient, hay nói cách khác là model chỉ học được từ các state gần nó tạo nên tính chất short term memory.



Hình 15: Kiến trúc của một memory cell. ²⁴

Một memory cell được cấu thành từ bốn thành phần chính: input gate, output gate, forget gate và self-recurrent neuron. Các gate (cổng) kiểm soát tương tác giữa các memory cell lân cận và chính nó. Input gate có thể điều chỉnh đầu vào của cell. Cổng output gate chính là input gate của một memory cell khác nên nó có thể điều chỉnh trạng thái của cell này. Forget gate có thể chọn ghi nhớ hoặc quên trạng thái trước đó.



Hình 16: Module lặp lại bên trong LSTM: x_t và h_t lần lượt là vector đầu vào và kết quả đầu ra vào ô nhớ tại thời điểm t . h_t là giá trị của ô nhớ nó, f_t và o_t lần lượt là giá trị của input gate, forget gate và output gate tại thời điểm t . C_t là các giá trị của trạng thái tạm thời của ô nhớ tại thời điểm t . ²⁵

Hình số 16 biểu thị trạng thái mạng (network) đầy đủ của một cell, mô tả các thức mỗi giá trị của các cổng (gate) được cập nhật. Trong đó các ký hiệu được trích từ [5]:

- x_t là input vector của memory cell tại thời gian t
- $W_i, W_f, W_c, W_o, U_i, U_f, U_c, U_o, V_o$ là các ma trận trọng số
- b_i, b_f, b_c, b_o là các vector bias
- h_t là giá trị của memory cell tại thời điểm t , h là hidden state

²⁴ Theo: <https://doi.org/10.1371/journal.pone.0180944.g005>

²⁵ Theo: <https://doi.org/10.1371/journal.pone.0180944.g006>

- i_t, \tilde{C}_t là các giá trị của input gate và candidate gate của memory cell tại thời điểm t , được tính toán bởi:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$

$$\tilde{C}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c)^{26}$$

- f_t, C_t là giá trị của forget gate và trạng thái (state) của memory cell tại thời điểm t . Được tính toán bởi:

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

$$C_t = i_t * \tilde{C}_t + f_t * C_{t-1}^{27}$$

- o_t, h_t lần lượt là output gate và giá trị (value) của memory cell tại thời điểm t . Tương ứng với:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + V_o C_t + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

Nhận xét:

\tilde{C}_t tương tự với cách tính s_t trong mô hình RNN. Forget gate c_t sẽ quyết định cần lấy bao nhiêu từ cell state trước đó và input gate sẽ quyết định lấy bao nhiêu từ input của state và hidden layer của layer trước.

Output gate quyết định xem cần lấy bao nhiêu từ cell state để trở thành output của hidden state.

3.4. Các lớp RNN sử dụng trong các framework về deep learning

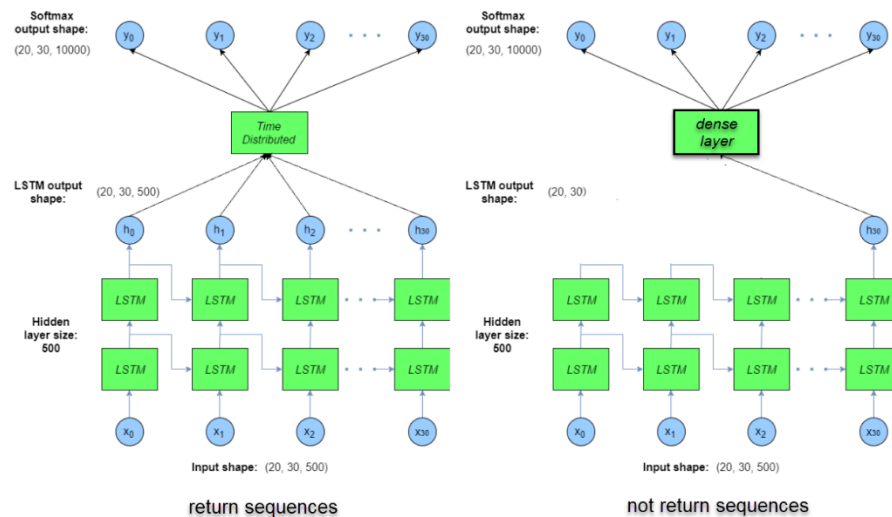
3.4.1. TimeDistributed layer

Lớp TimeDistributed là xác định phân phối của dữ liệu theo thời gian. Về mặt bản chất, TimeDistributed layer không khác gì các dense layer (của fully connected layer), điểm khác biệt ở chỗ Dense layer chỉ lấy vector output tại time step cuối cùng trong LSTM còn TimeDistributed thì lấy hết ở các bước time step khác. Ngoài ra, TimeDistributed layer còn được sử dụng rất nhiều trong các mô hình xử lý video.

²⁶ Theo trang 11/24, i_t và C_t lần lượt là biểu thức thứ (22) và (23)

²⁷ Theo trang 11/24, f_t và C_t là biểu thức thứ (24) và (25)

- Lấy ví dụ mô hình RNN trong bài toán input là chuỗi sequence sử dụng Dense layer thông thường:



Hình 17: Mô hình RNN sử dụng Dense layer và TimeDistributed²⁸

Như hình số 17, các input đầu vào là các vector x_i tại thời điểm i , kết hợp với các hidden layer $(i-1)$ tạo ra các hidden layer (i) , lần lượt cho đến hidden layer cuối và chỉ trả ra output h_{30} .

- Sử dụng TimeDistributed layer:

Hình 17 còn mô tả TimeDistributed layer sẽ nhận dữ liệu đầu vào giống như Dense layer nhưng TimeDistributed layer trả ra một chuỗi các hidden layer từ các time step từ h_0 cho đến h_{30} .

3.4.2. LSTM layer

LSTM layer là lớp tìm hiểu về sự phụ thuộc giữa các bước time step trong cả quá trình thực hiện RNN với dữ liệu kiểu sequence đầu vào. Ví dụ với bài toán input là dữ liệu dạng sequence (video 30s), thì LSTM layer sẽ thực hiện việc học về sự phụ thuộc giữa các bước time step từ h_0 cho đến h_{30} so với dữ liệu đầu vào là video 30 giây.

LSTM layer thực hiện việc tăng sự tương tác giúp cải thiện gradient flow trong cả sequence trong quá trình huấn luyện dữ liệu. Quá trình thực hiện các bước giống

²⁸ Nguồn: <https://phamdinhhkhanh.github.io/2019/12/02/DeepLearningLayer.html>

với mục 3.3 (Long-short term memory), tại output gate sẽ quyết định xem cần lấy bao nhiêu từ cell state để trở thành output của hidden state.

Chương 4: Các quá trình huấn luyện mô hình

4.1. Quá trình Feedforward

Feedforward hay còn gọi là lan truyền thẳng, dữ liệu sẽ đi tuần tự từ đầu vào đến đầu ra thông qua các lớp.

Lấy ví dụ từ mô hình perceptron đa tầng hình 3, sau khi áp dụng công thức tính tổng linear tại node thứ nhất ở hidden layer 1:

$$z_1^{(1)} = x_1 * w_{11} + x_2 * w_{21} + b_1^{(1)}$$

$$a_1^{(1)} = \sigma(z_1^{(1)})$$

Tại node thứ hai:

$$z_2^{(1)} = x_1 * w_{12} + x_2 * w_{22} + b_2^{(1)}$$

$$a_2^{(1)} = \sigma(z_2^{(1)})$$

Tại node thứ ba:

$$z_3^{(1)} = x_1 * w_{13} + x_2 * w_{23} + b_3^{(1)}$$

$$a_3^{(1)} = \sigma(z_3^{(1)})$$

Gọi input layer là x là $a^{(0)}$:

$$Z^{(1)} = \begin{bmatrix} Z_1^{(1)} \\ Z_2^{(1)} \\ Z_3^{(1)} \end{bmatrix} = \begin{bmatrix} a_1^{(0)} * w_{11}^{(1)} + a_2^{(0)} * w_{21}^{(1)} + b_1^{(1)} \\ a_1^{(0)} * w_{12}^{(1)} + a_2^{(0)} * w_{22}^{(1)} + b_2^{(1)} \\ a_1^{(0)} * w_{13}^{(1)} + a_2^{(0)} * w_{23}^{(1)} + b_3^{(1)} \end{bmatrix}$$

$$= (W^{(1)})^T * A^{(0)} + b^{(1)}$$

$$A^{(1)} = \sigma(Z^{(1)})$$

Tương tự :

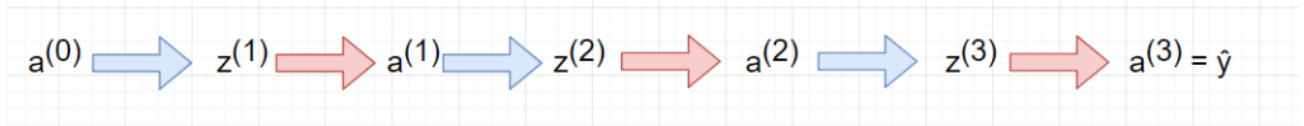
$$Z^{(2)} = (W^{(2)})^T * A^{(1)} + b^{(2)}$$

$$A^{(2)} = \sigma(Z^{(2)})$$

$$Z^{(3)} = (W^{(3)})^T * A^{(2)} + b^{(3)}$$

$$\hat{y} = A^{(3)} = \sigma(Z^{(3)})$$

Quá trình Feedforward diễn ra từ input $a^{(0)}$ đến $a^{(3)}$ và cũng là kết quả dự đoán output:



Hình 18: Quá trình feedforward ²⁹

Quá trình feedforward được mô tả trong hình 18 biểu thị rằng dữ liệu ban đầu đi vào lớp (0) sẽ được tính toán z thông qua các trọng số W và bias b để tính a thông qua activation function và kết thúc cho đến khi thu được output y .

4.2. Quá trình Backpropagation

Trái ngược với quá trình feedforward, còn có quá trình backpropagation, hay còn gọi là lan truyền ngược, giúp tính đạo hàm từ layer cuối cùng đến layer đầu tiên.

Sở dĩ layer cuối cùng được tính trước do nó gần với kết quả dự đoán \hat{y} và hàm loss function.

Với ví dụ ở phần feedforward, đầu tiên tính đạo hàm của hàm Loss theo \hat{Y} :

$$\frac{\partial J}{\partial \hat{Y}}, \text{ trong đó } \hat{Y} = A^{(3)}$$

Tính đạo hàm của hàm Loss J theo $W^{(3)}$ và theo $b^{(3)}$ và theo $A^{(2)}$

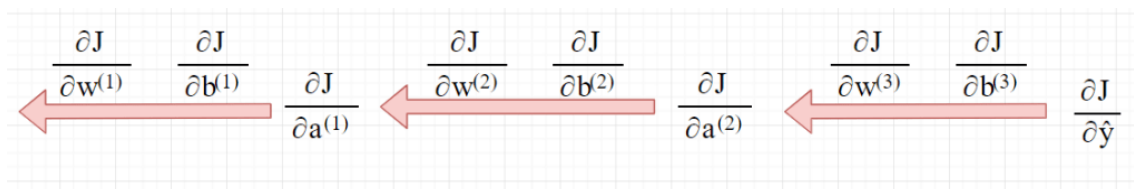
$$\frac{\partial J}{\partial W^{(3)}} = (A^{(2)})^T * \left(\frac{\partial J}{\partial \hat{Y}} \otimes \frac{\partial A^{(3)}}{\partial Z^{(3)}} \right)$$

$$\frac{\partial J}{\partial b^{(3)}} = \left(\sum \left(\frac{\partial J}{\partial \hat{Y}} \otimes \frac{\partial A^{(3)}}{\partial Z^{(3)}} \right) \right)^T$$

$$\frac{\partial J}{\partial A^{(2)}} = \left(\frac{\partial J}{\partial \hat{Y}} \otimes \frac{\partial A^{(3)}}{\partial Z^{(3)}} \right) * (W^{(3)})^T$$

Tiếp tục với layer tiếp theo, cho tới cuối cùng $A^{(0)} = X$ thì kết thúc.

²⁹ Nguồn: <https://nttuan8.com/bai-3-neural-network/>



Hình 19: Quá trình backpropagation ³⁰

Theo hình số 19, khi thu được kết quả từ output \hat{y} , các đạo hàm của J theo trọng W sẽ được tính lần lượt ngược lại theo chiều tính thuận ban đầu để cập nhật các trọng số W .

4.3. Transfer Learning

Kỹ thuật máy học (machine learning) và khai thác dữ liệu (data mining) đã được sử dụng trong nhiều ứng dụng trong thế giới thực. Một giả định (assumption) của phương pháp học máy truyền thống là dữ liệu đào tạo và dữ liệu thử nghiệm được lấy từ cùng một domain (miền giá trị), sao cho không gian đặc trưng đầu vào và đặc điểm phân phối dữ liệu giống nhau. Tuy nhiên, trong một số bài toán học máy trong thế giới thực, giả định này không đúng. Có những trường hợp nơi dữ liệu đào tạo đắt tiền hoặc khó thu thập. Do đó, cần phải tạo ra mô hình có hiệu suất cao được đào tạo với dữ liệu dễ dàng thu được hơn từ các lĩnh vực khác nhau. Phương pháp này được gọi là học chuyển tiếp (transfer learning).[7]

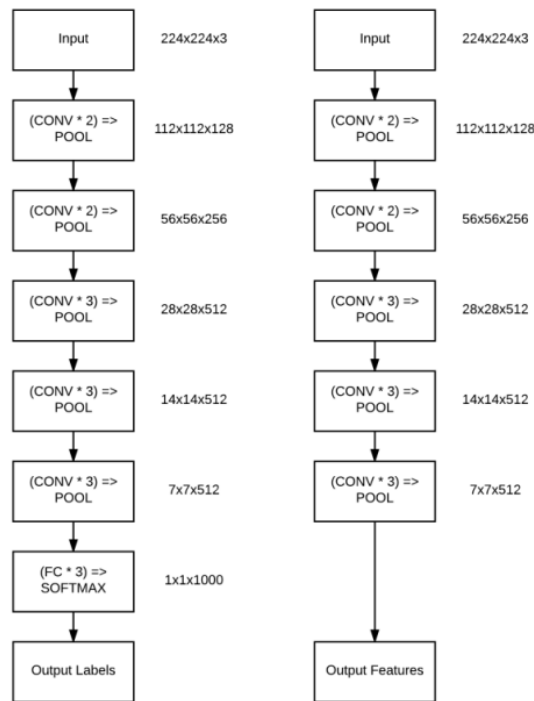
Transfer learning là phương pháp áp dụng các pre-train model từ các bài toán lớn với độ chính xác cao có các yếu tố tương đồng với bài toán hiện tại để huấn luyện mô hình một cách hiệu quả và không tốn quá nhiều thời gian.

Có hai phương pháp Transfer learning: Feature extractor và fine turning.

4.3.1. Feature extractor

Feature extractor là quá trình lấy ra các đặc điểm của ảnh thông qua sử dụng ConvNet của pre-trained model, sau đó sẽ được dùng như input của bài toán linear regression hay logistic regression.

³⁰ Nguồn: <https://nttuan8.com/bai-4-backpropagation>



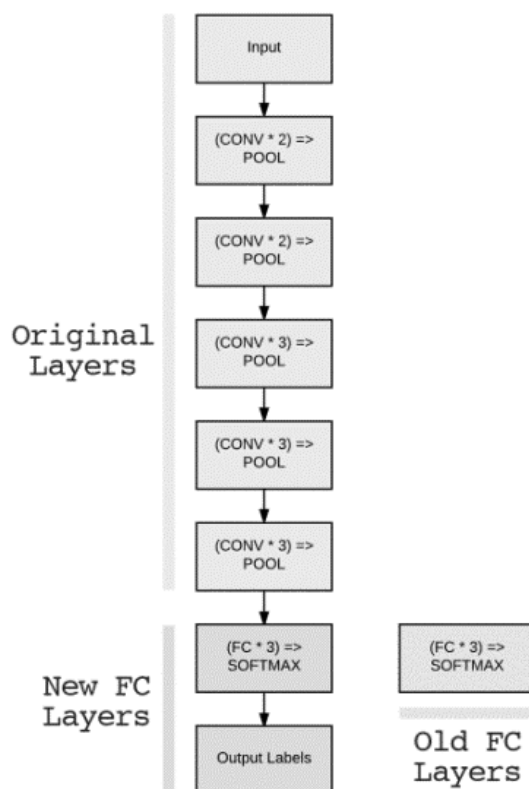
Hình 20: Mô hình VGG 16³¹

Hình 20 phía bên trái là mô hình mạng CNN với mô hình VGG 16, bên phải là mô hình VGG chỉ bao gồm ConvNet, bỏ qua các bước fully connected layer. Feature extractor sẽ lấy “Output Features” để làm input cho các bài toán khác.

4.3.2. Fine tuning

Fine tuning là quá trình lấy ra các đặc điểm của ảnh thông qua sử dụng ConvNet của pre-trained model, sau đó tiếp tục lấy những đặc điểm này làm input cho một CNN hoặc một model mới bằng cách thêm các ConvNet và fully connected layer để học thêm những đặc trưng khác nhằm tối ưu hóa độ chính xác của bài toán.

³¹ Nguồn: <https://nttuan8.com/bai-9-transfer-learning-va-data-augmentation/>



Hình 21: Mô hình VGG 16 sử dụng fine tuning³²

Hình 21 là mô hình VGG 16 sau khi đã bỏ qua fully connected layer cũ và thay thế bằng fully connected layer mới để phù hợp với mô hình mới.

Độ chính xác accuracy khi sử dụng fine tuning sẽ tốt hơn sử dụng feature extractor do fine tuning sử dụng thêm các fully connected layer mới để học các đặc điểm phù hợp với bài toán, vì thế thời gian training khi sử dụng fine tuning sẽ nhiều hơn sử dụng feature extractor.

³² Nguồn: <https://nttuan8.com/bai-9-transfer-learning-va-data-augmentation/>

Chương 5: Nhận diện bạo lực trong video bằng các mô hình neural nhân tạo

5.1. Tổng quan về bài toán nhận diện bạo lực trong video

5.1.1. Tổng quan về tập dữ liệu

Như đã đề cập trong phần “lý do chọn đề tài”, để có một mô hình đạt hiệu quả cao và tương đồng với thực tế, nhóm cần phải tìm một tập dữ liệu được ghi lại từ thực tiễn, độ chân thực cao và không bị trùng lặp. Vì vậy, nhóm đã sử dụng tập dữ liệu trên nền tảng Kaggle³³.

Khi nhóm bắt đầu dự án “Nhận dạng bạo lực từ các video”, nhóm nhận thấy rằng thiếu bộ dữ liệu có sẵn liên quan đến bạo lực giữa các cá nhân có đủ uy tín, vì vậy chúng tôi quyết định lựa chọn một bộ dữ liệu được cải tiến từ bộ dữ liệu đã được nghiên cứu.

Tập dữ liệu từ nguồn [0] [8] mà nhóm sử dụng được bổ sung từ nguồn đã được nghiên cứu [9].

Bộ dữ liệu bao gồm 1000 video bạo lực (violence) và 1000 video không bạo lực (nonviolence) được thu thập từ nguồn YouTube, các video bạo lực trong tập dữ liệu chứa nhiều tình huống đánh nhau thực tế trên đường phố trong một số môi trường và điều kiện cụ thể. Cũng như các video không bạo lực từ tập dữ liệu được thu thập từ nhiều hành động khác nhau của con người như thể thao, ăn uống, đi bộ ... vv.

5.1.2. Tổng quan về phương án đề xuất

Dựa vào cho kiến trúc của phương pháp được thực nghiệm trước đó [9], nhóm nhận thấy mô hình kiến trúc được phân bổ như sau: Đầu tiên, các thao tác tiền xử lý được áp dụng cho các khung (ảnh) video đầu vào. Hai giai đoạn trích xuất đối tượng địa lý liên tiếp được áp dụng; giai đoạn VGG-16 chịu trách nhiệm trích xuất các đặc trưng không gian cho mỗi khung hình và giai đoạn LSTM hoạt động như trình trích xuất các đặc trưng thời gian. Cuối cùng, các đặc trưng được trích xuất được đưa đến các lớp fully connected để phân loại.

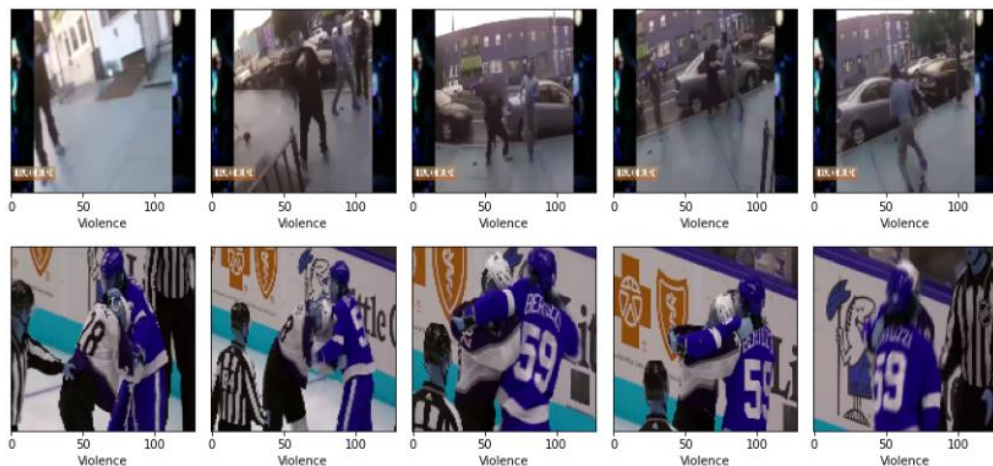
³³ Kaggle, một công ty con của Google. Kaggle cho phép người dùng tìm và xuất bản các tập dữ liệu, khám phá và xây dựng các mô hình trong môi trường khoa học dữ liệu dựa trên web, đồng thời tham gia các cuộc thi để giải quyết các thách thức về khoa học dữ liệu.

Dựa vào kết quả nghiên cứu [9], kết quả thu được là 88.2%³⁴. Nhóm nhận thấy có thể cải tiến sử dụng một pre-train model khác và thiết kế lại giai đoạn LSTM model để phù hợp với mục tiêu ban đầu nhóm đặt ra. Tuy nhiên điều này không đồng nghĩa với việc mô hình thiết kế của nhóm tốt hơn so với kết quả nghiên cứu sẵn có mà nó chỉ phản ánh kết quả phù hợp với yêu cầu với tập dữ liệu này.

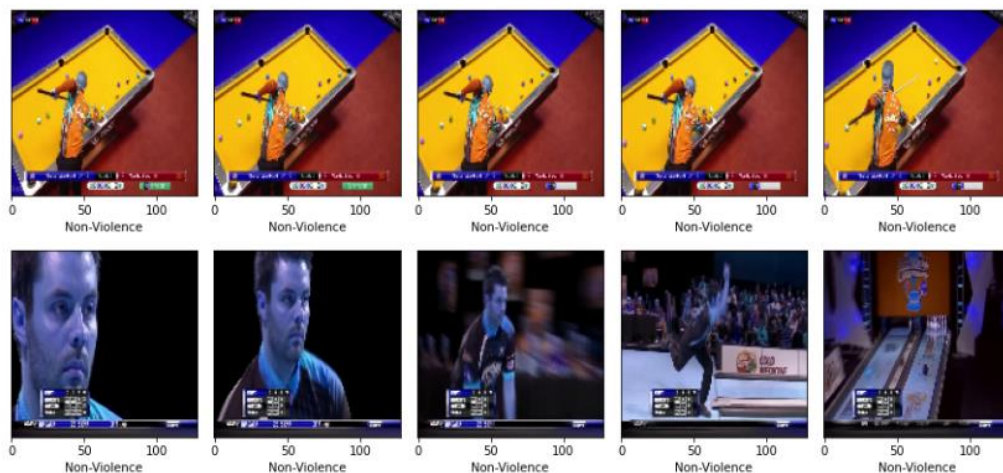
5.2. Phương pháp xử lý dữ liệu

5.2.1. Xử lý video

Vì tập dữ liệu chỉ gồm những video ngắn từ 3-5 giây, tối đa 10 giây với mỗi video. Vì vậy, tập dữ liệu được mô tả bởi bộ (V, F, H, W, C). Với V là số lượng video trên mỗi tập con, F là số lượng frame (ảnh) với mỗi video, H và W lần lượt là chiều dài và chiều rộng của frame thứ I, C là channel ứng với ba màu RGB.



Hình 22: Hình ảnh video bạo lực được tách thành các hình ảnh đơn lẻ



Hình 23: Hình ảnh video không bạo lực được tách thành các hình ảnh đơn lẻ

³⁴ 86.2%, 88.2% and 84.0% trên tập hockey fight, movie và violent-flow datasets

Nhằm để cắt được frame (ảnh) của từng video, nhóm sử dụng OpenCV³⁵ để bắt từng frame (ảnh) và chuyển về dạng RGB, sau đó được resize (điều chỉnh) 128x128x3 như bài báo gốc [9] nhằm phù hợp với mô hình pre-train. Vì mỗi pixel của ảnh sẽ phân bố trong khoảng $[0, 255]$, vì vậy cần điều chỉnh về lại khoảng $[0, 1]$ bằng các lấy phép chia của từng điểm cho 255. Chúng tôi lấy 30 frame cho mỗi video. Như vậy, mỗi tập dữ liệu sẽ được mô tả là một tensor có kích thước $(V, 30, 128, 128, 3)$.

5.2.2. Chuẩn bị dữ liệu đầu vào cho mô hình

Để đảm bảo tính ngẫu nhiên của domain (miền) dữ liệu và các frame trong cùng một video không bị xáo trộn, nhóm lấy toàn bộ đường dẫn của các video trong tập dữ liệu và thu được một tập bao gồm đường dẫn của các video. Sau đó, nhóm xáo trộn ngẫu nhiên các đường dẫn trong tập này. Từ đó, tập đường dẫn mới bao gồm ngẫu nhiên các đường dẫn của các video

Nhằm phục vụ cho quá trình kiểm thử (testing), nhóm chia tập tên đường dẫn thành hai tập là huấn luyện (training) và kiểm thử (testing) với tỉ lệ 8/2, nghĩa là 80% trên tổng số 2000 video cho tập huấn luyện và 20% trên tổng số 2000 video cho kiểm thử. Quá trình này được chia đồng đều trên mỗi phân lớp violence hoặc nonviolence. Kết quả thu được tập tên đường dẫn huấn luyện và tập tên đường dẫn kiểm thử

Sau quá trình xử lý video, nhóm thu được tập dữ liệu có kích thước $(1000, 30, 224, 224, 3)$ ứng với mỗi phân lớp violence và nonviolence. Từ dữ liệu này, để dữ liệu nhỏ gọn và dễ dàng huấn luyện, ta lấy từng điểm dữ liệu chia cho 255, bởi vì mỗi điểm ảnh tương ứng với 8-bit màu (0-255).

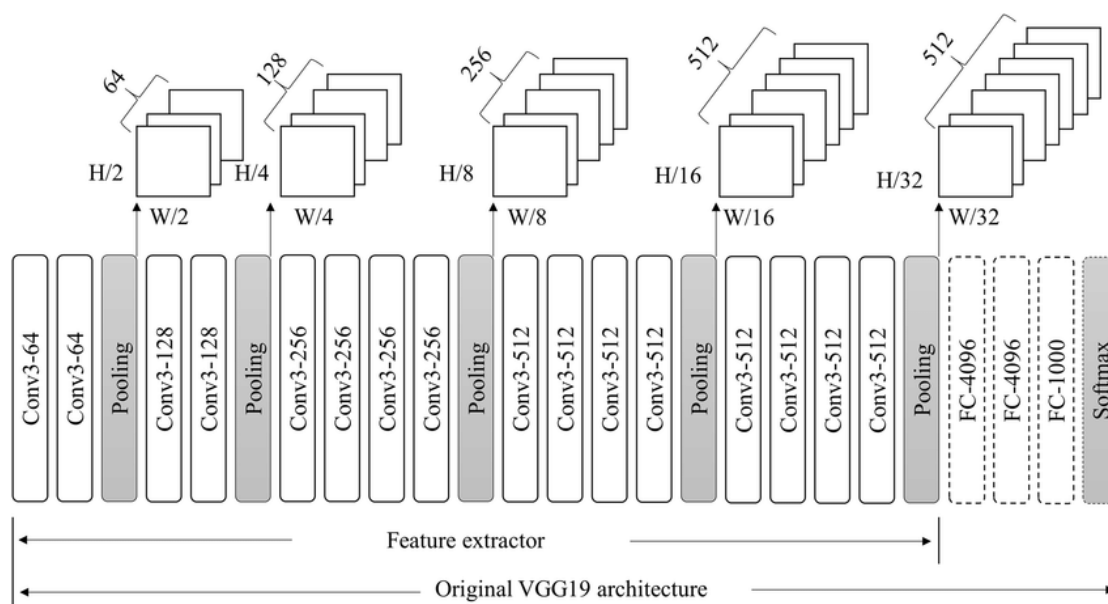
Với tập huấn luyện (training), nhóm sẽ chia thành 3 tập nhỏ hơn tạo thành k-fold³⁶. Với mỗi tập dữ liệu nhỏ, nhóm tách 20% dữ liệu thành tập kiểm định (validation). Tập dữ liệu này giúp cho việc đánh giá mô hình trong quá trình huấn luyện mà không phải sử dụng đến tập kiểm thử (testing).

³⁵ OpenCV (Open-Source Computer Vision Library) là một thư viện các chức năng lập trình chủ yếu nhằm vào thị giác máy tính thời gian thực

³⁶ Mục đích của điều này để phù hợp với giới hạn phân cứng

5.3. Xây dựng giải quyết bài toán

5.3.1. Xác định mô hình mạng



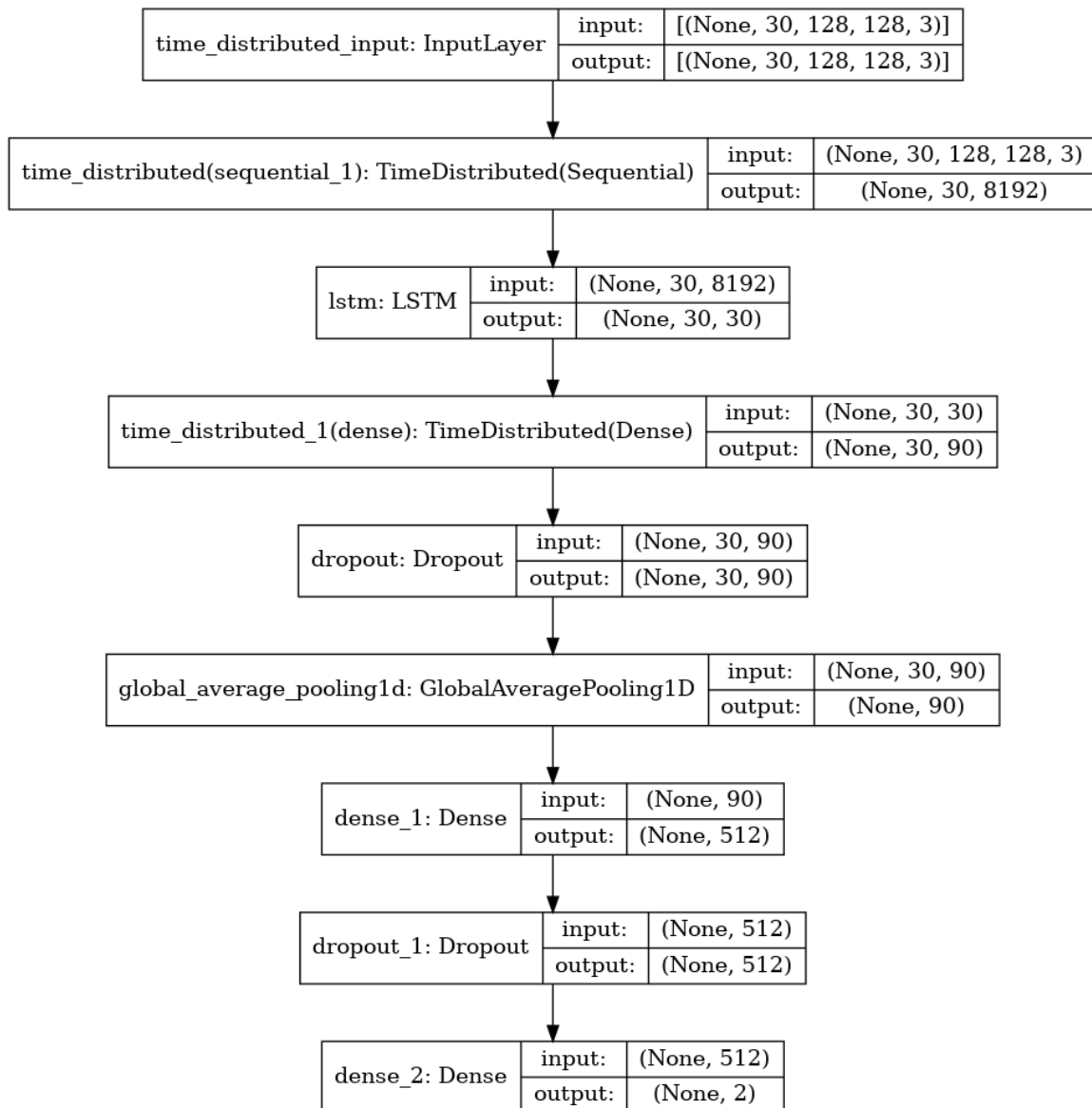
Hình 24: Feature extractor sử dụng kiến trúc của mô hình VGG-19³⁷

Như đã đề cập ở phần trước, mô hình nhóm sử dụng sẽ bao gồm ba thành phần chính đó là: Sử dụng mô hình VGG-19 theo mô tả trên hình 24 như một lớp feature extractor. Mô hình VGG-19 bao gồm 5 block convolution nhằm chiết xuất đặc điểm của dữ liệu, xen vào giữa chúng là các lớp pooling dùng để thu nhỏ số chiều của dữ liệu lại. Mô hình nhóm sử dụng sẽ loại bỏ đi lớp fully connected và lớp softmax và trở thành đầu vào cho thành phần thứ hai.

Thành phần thứ hai của mô hình bao gồm các thành phần của mô hình LSTM: TimeDistributed và LSTM. Lớp TimeDistributed thường được sử dụng trong các mô hình xử lý video. Nó xác định phân phối của dữ liệu theo thời gian, nó sẽ lấy hết toàn bộ thông tin (các kích thước và channel sẽ được gộp lại) của các time step (ở đây nhóm quy định là 30) và đưa vào lớp LSTM. Tại lớp LSTM này, mô hình sẽ học dữ liệu sequence có từ data. Sau đó, cần một lớp TimeDistributed tiếp theo nhằm lấy thông tin từ lớp LSTM để thu được về dạng sequence, sau đó mới có thể đưa vào lớp fully connected.

³⁷ Nguồn: <https://www.researchgate.net/profile/Bong-Chul-Kim-2>

Thành phần thứ ba của mô hình gồm lớp fully connected: gồm các lớp dense thông thường kết nối với nhau như mạng ANN truyền thống. Thêm vào đó, các lớp dropout giúp giảm bớt việc kết nối quá chặt chẽ trong mạng (dễ dẫn tới overfitting) và giảm thời gian huấn luyện của mô hình. Số chiều của mô hình sẽ được điều chỉnh trong suốt thành phần này cho đến khi thu được đầu ra. Output (đầu ra) bao gồm 2 phân lớp chính: violence (1) và nonviolence (0) tại lớp dense cuối cùng.



Hình 25: Kiến trúc thành phần LSTM và fully connected layer

5.3.2. Mô tả quy trình huấn luyện mạng

5.3.2.1. Các helper function

- Learning Rate Scheduler

+ Vào đầu mỗi epoch, nó sẽ cập nhật lại giá trị learning rate được truyền vào, với epoch hiện tại và chỉ số learning rate hiện tại và áp dụng chỉ số learning rate mới cho hàm tối ưu hóa.

- Early stopping

+ Nếu coi mục tiêu của training là giảm thiểu tối đa giá trị mất mát (loss). Hàm `model.fit()` sẽ lặp lại để kiểm tra tại mỗi khi kết thúc một epoch liệu loss có còn giảm không, xem xét các giá trị `min_delta` (sự cải thiện tối thiểu), `patience` (số epoch không có sự cải thiện) nếu có. Khi loss không còn giảm nữa thì model sẽ trở thành “True” và kết thúc training.

- Reduce Learning Rate on Plateau (`ReduceLROnPlateau`)

+ Model kiểm tra và nếu không có sự cải thiện sẽ xem đó là một “patience” của epoch, sau đó giá trị learning rate sẽ giảm xuống.

- Model Checkpoint: Được sử dụng trong việc training kết hợp `model.fit` để lưu model hoặc các trọng số trong các khoảng thời gian.

5.3.2.2. Một số hyper parameter

- Adam optimizer

+ Là phương pháp giảm dần độ dốc của gradient descent dựa trên các ước lượng tương ứng của first-order và second order.

- Batch size

+ Là số lượng mẫu dữ liệu trong một lần huấn luyện. Ví dụ, trong bài toán phân loại chó mèo, chọn `batch size = 32`, nghĩa là ta sẽ tính gradient cho 32 bức ảnh rồi mới cập các trọng số của mạng một lần.

5.3.2.3. Điều chỉnh siêu tham số cho mô hình

Mô hình được huấn luyện dựa trên dữ liệu đã được xử lý và được mô tả trong phần 2.2. Để chuẩn bị cho quá trình huấn luyện, các helper function được sắp xếp theo thứ tự [`learning rate scheduler`, `model checkpoints`, `early stopping`, `Reduce LR On Plateau`].

Các hyperparameter nhóm chọn lựa bao gồm: `patience = 5`, `start learning rate = 0.00001`, `min learning rate = 0.00001`, `max learning rate = 0.00005` và `batch size = 16`. Lý do chọn những chỉ số này bởi vì: với `patience`, mô hình khi huấn luyện thường sẽ xảy ra hiện tượng chững lại (loss không đổi), nhưng khi nếu mô hình không giảm loss,

điều này có nghĩa mô hình đã hội tụ. Learning rate được đặt ở mức thấp và ít thay đổi giúp cho mô hình không gặp phải tình trạng nhảy cóc. Batch size với mức 16 để phù hợp với phần cứng.

Hyper-parameter	Giá trị
Số lượng phân lớp	2
Batch size	16
Epoch	100
Learning rate	[0.00001, 0.00005]
Patience	5

Bảng 2: Các hyper-parameter của mô hình đề xuất

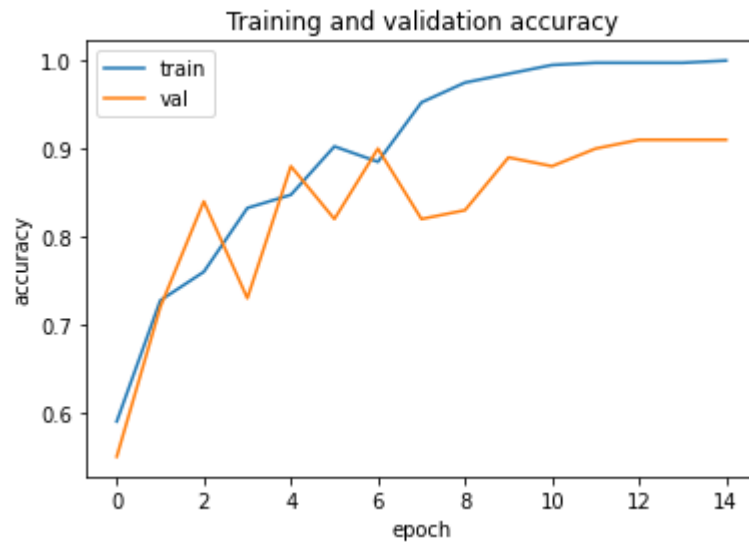
5.4. Đánh giá mô hình

5.4.1. Learning history

Một trong những phương pháp đánh giá mô hình đào tạo tất cả các mô hình học sâu là training history (lịch sử huấn luyện). Nó ghi lại các số liệu đào tạo cho mỗi epoch. Điều này bao gồm mất mát, độ chính xác trên tập huấn luyện và mất mát, độ chính xác cho tập dữ liệu xác thực nếu một tập dữ liệu được đặt.

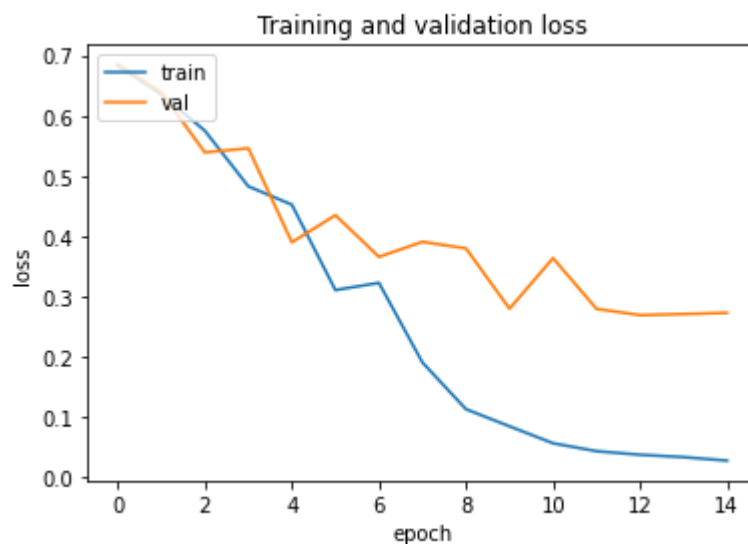
Trước hết, vì lý do cần phải chia dataset thành các phần và giữ lại các weight qua mỗi lần huấn luyện, nhóm chỉ lưu trữ lại lần training history đầu tiên và đánh giá mô hình dựa trên epoch tốt nhất

Các biểu đồ được mô tả lại quá trình huấn luyện dưới đây. Lịch sử của tập dữ liệu xác thực (validation) và huấn luyện (training) được biểu diễn đồng thời nhằm so sánh sự khác biệt.



Hình 26: Model accuracy ứng với 15 epoch đầu tiên trên tập fold-1 training

Từ biểu đồ về độ chính xác (accuracy), mô hình cho thấy rằng mô hình đang có xu hướng về độ chính xác đang cải thiện dần trên cả hai tập dữ liệu. Tuy nhiên, mô hình chỉ mới trong khoảng 10 epoch đã dừng dựa trên sự can thiệp của early stopping chứng tỏ mô hình đã bão hòa dưới tập đầu tiên trong k-fold. Nếu có thể bổ sung thêm dữ liệu hoặc huấn luyện đồng thời, mô hình có thể cải thiện.



Hình 27: Model loss ứng với 15 epoch đầu tiên ứng với tập fold-1 training

Từ biểu đồ loss, mô hình có sự chênh lệch khá rõ rệt giữa loss trên tập train (huấn luyện) với loss trên tập validation (kiểm định). Dựa trên một số nhận định của nhóm, có thể do dữ liệu chưa đủ (do giới hạn trên thiết bị phần cứng) dẫn đến việc loss còn tương đối cao trên tập fold đầu tiên. Điều này có thể khắc phục qua những tập fold tiếp theo dựa trên các trọng số đã được lưu lại.

Trải qua ba lần huấn luyện, dựa trên kết quả của epoch tốt nhất, nhóm thu lại được như sau:

Tập dữ liệu	Loss	Accuracy
Huấn luyện (training set)	0.1467	0.9542
Xác thực (validation)	0.1531	0.95

Bảng 3: Kết quả huấn luyện với epoch tốt nhất

Sau nhiều epoch, độ chính xác tối đa mà mô hình có thể đạt được trên tập huấn luyện là 95.42%, độ chính xác trên tập xác thực là 95%. Learning curve của mô hình trên tập k-fold đầu tiên được mô tả như trên hai hình bên trên.

5.4.2. Confusion matrix

Các phép đo chính xác như Precision, Recall và F1 cũng được ước tính và mô tả để đánh giá thêm về mô hình trên tập kiểm thử (testing set).

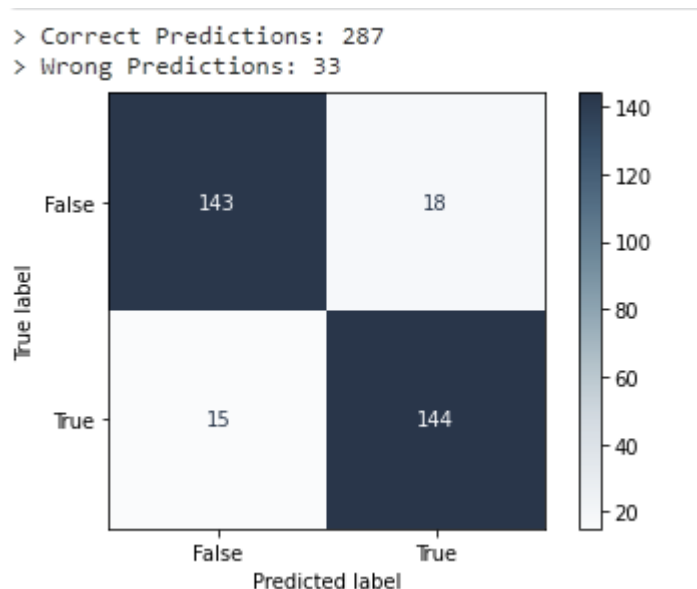
Những tham số này cho biết nhiều thông tin về bộ phân loại hoạt động tốt như thế nào thay vì chỉ nhìn vào độ chính xác tổng thể. Các thông số precision, sensitivity (recall) và F1-score được tính thông qua những công thức sau:

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

$$F1\ score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Dựa trên các công thức tính toán đó, nhóm tính toán dự trên 200 video trên mỗi phân lớp trong tập huấn luyện. Kết quả thu lại thông qua confusion matrix và các chỉ số Precision, Recall, F1 score.



Hình 28: Confusion matrix trên tập dữ liệu testing (kiểm thử) với true label và predicted label

Hình số 28 mô tả confusion matrix (ma trận nhầm lẫn) trên tập dữ liệu thử nghiệm với hai label là true label và predicted label. Bên cạnh đó, trong mỗi label có hai phân lớp tương ứng: False – Nonviolence và True – Violence. Confusion matrix cung cấp một số thông tin sơ bộ như sau: tổng số dự đoán đúng (287), số dự đoán sai (33). Bốn góc phần tư của ma trận tương ứng với: True negative (Góc phần tư trên cùng bên trái) = 143; True Negative (Góc phần tư trên cùng bên phải) = 18; False Positive (Góc phần tư dưới cùng bên trái) = 15; Tích cực thực sự (Góc phần tư dưới cùng bên phải) = 144.

Từ ma trận trên, nhóm còn rút ra được một số độ đo precision, recall, f1-score và accuracy tương ứng với từng phân lớp.

	precision	recall	f1-score	support
NonViolence	0.91	0.89	0.90	161
Violence	0.89	0.91	0.90	159
accuracy			0.90	320
macro avg	0.90	0.90	0.90	320
weighted avg	0.90	0.90	0.90	320

Hình 29: Confusion matrix ứng với tập testing với các độ đo khác nhau

Kết quả thu được trong hình 29 biểu diễn các thông số của mô hình:

- Với từng phân lớp độ đều có f1-score là 0.90
- Độ chính xác accuracy là 0.9, bên cạnh đó macro và weighted avg đều là 0.90.

Ngoài ra trên tập kiểm thử, nhóm còn thu lại được loss của mô hình là 0.2821.

5.5. Thảo luận và so sánh

Phần tiểu mục này của nhóm đưa ra một phân tích so sánh giữa nghiên cứu hiện tại và các phương pháp tiên tiến nhất (state of the art - SOTA) trước đây trong việc phát hiện bạo lực từ video. Theo như SOTA của bài toán action recognition on real life violence [9], độ chính xác đạt được từ cả các mô hình cũng như các phương pháp tiếp cận dựa trên học sâu đã được minh họa trong phần so sánh bằng bảng bên dưới.

No.	Model	Accuracy	Paper	Year
1	DeVTr	96.25%	Data Efficient Video Transformer for Violence Detection ³⁸	2021
2	Temporal Fusion CNN + LSTM	91%	A Temporal Fusion Approach ³⁹	2021
3	VGG19+LSTM	90%	–	2022
4	CNN+LSTM	88.8%	Violence Recognition from Videos using Deep Learning Techniques ⁴⁰	2019

Bảng 4: So sánh độ chính xác từ các mô hình đã được công nhận và SOTA của bài toán

Từ bảng kết quả so sánh có thể thấy, mô hình có độ chính xác thấp hơn khá nhiều so với SOTA (một mô hình tương đối phức tạp) và thấp hơn một mô hình kết hợp CNN+LSTM khác là Temporal Fusion CNN + LSTM. Tuy nhiên mô hình đã có thể có độ chính xác cao hơn mô hình tiền nhiệm trước đó sử dụng VGG16 (CNN+LSTM).

Để một mô hình có thể triển khai lên thực tế phải đảm bảo độ chính xác (accuracy) > 95%, vì vậy mô hình của nhóm đã chưa đủ tiệm cận mức này để đảm khả năng triển khai.

5.6. Kết luận và công trình tương lai

Phát hiện bạo lực là rất quan trọng đối với nhiều ứng dụng, nhưng tính chủ quan của nó khiến mô hình chung khó chính xác. Hiện tại, bài toán về nhận dạng bạo lực trong video đã có nhiều mô hình từ phức tạp đến đơn giản với độ chính xác cao. Mô

³⁸ <https://paperswithcode.com/paper/data-efficient-video-transformer-for-violence>

³⁹ <https://paperswithcode.com/paper/a-temporal-fusion-approach-for-video>

⁴⁰ <https://paperswithcode.com/paper/violence-recognition-from-videos-using-deep>

hình nhóm đề xuất và thử nghiệm cũng là một trong số đó. Các mô hình đều có các đặc điểm chung bắt nguồn từ mô hình học sâu (deep learning) và cách tiếp cận giống nhau. Đầu tiên, các video hay clip sẽ được tách thành nhiều frame (khung ảnh) không trùng nhau và sau đó ứng dụng các phép toán image augmentation để nâng cao hiệu quả hình ảnh.

Với mô hình hiện tại, độ chính xác chưa thể so sánh so với các mô hình state of the art (SOTA) khác vì sự đơn giản và theo phương pháp dựa trên CNN. Nhược điểm tiềm ẩn của mô hình của nhóm đề xuất là chưa thể phát hiện các hành vi bạo lực theo đám đông hoặc những hình thức bạo lực vượt ra khỏi phạm vi của bộ dữ liệu, nó cần một lượng lớn dữ liệu với điều kiện hoàn cảnh cụ thể để mô hình đào tạo có thể tốt hơn.

Chương 6: Ứng dụng nhận diện bạo lực trong video bằng Streamlit

6.1. Đặt vấn đề

Thay vì chỉ hoàn thiện mô hình trên thực nghiệm, để mô hình có thể tiến tới thực tiễn, nhóm đã triển khai ứng dụng web app để thuận tiện cho người dùng dễ dàng sử dụng hay kiểm tra độ chính xác của mô hình. Để triển khai một cách nhanh chóng, hiệu quả, phù hợp với mục tiêu đặt ra, nhóm đã sử dụng Streamlit để tiến hành.

6.2. Streamlit

Streamlit là một thư viện mã nguồn mở (open source), hỗ trợ việc triển khai và tùy chỉnh các web app một cách dễ dàng trong thời gian nhanh chóng, thường dùng cho học máy (machine learning) và khoa học dữ liệu (data science).

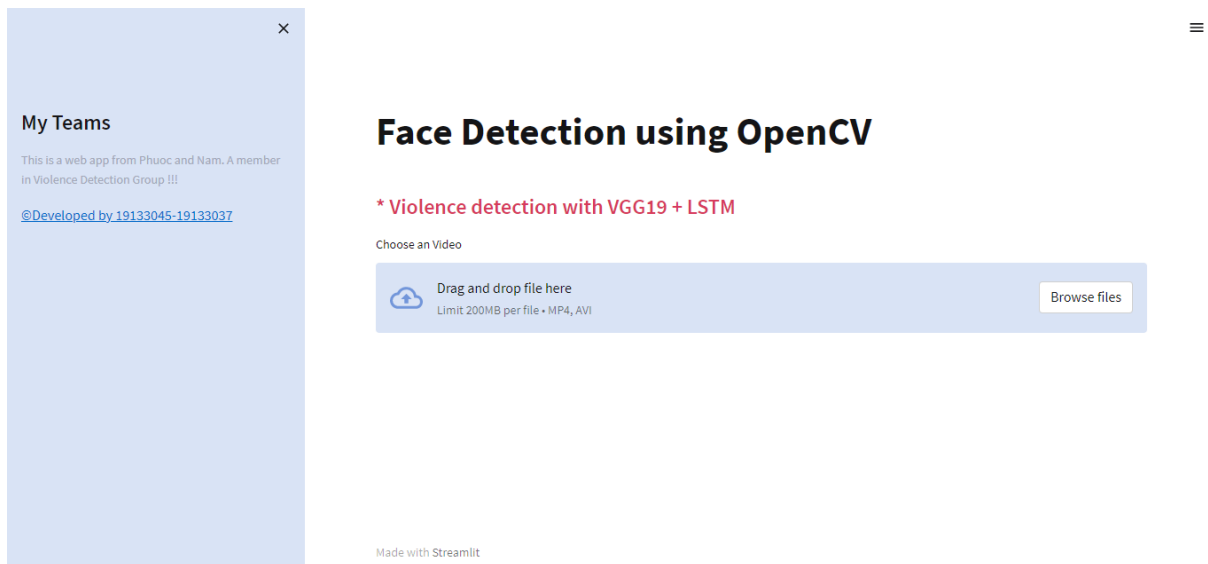
Lợi ích của việc sử dụng Streamlit:

- Hỗ trợ triển khai, điều chỉnh web app dễ dàng trong thời gian ngắn
- Demo dễ dàng các dự án nhỏ mà không cần custom (chỉnh sửa) quá nhiều, thích hợp cho học máy và khoa học dữ liệu.

Hình thức triển khai và cấu hình: Streamlit cho phép cấu hình các thành phần xuất hiện trên web app. Các thành phần phổ biến là title của web, các header, các biểu đồ, ... Ngoài ra, Streamlit còn có một số tính năng tương tác như checkbox, select box.

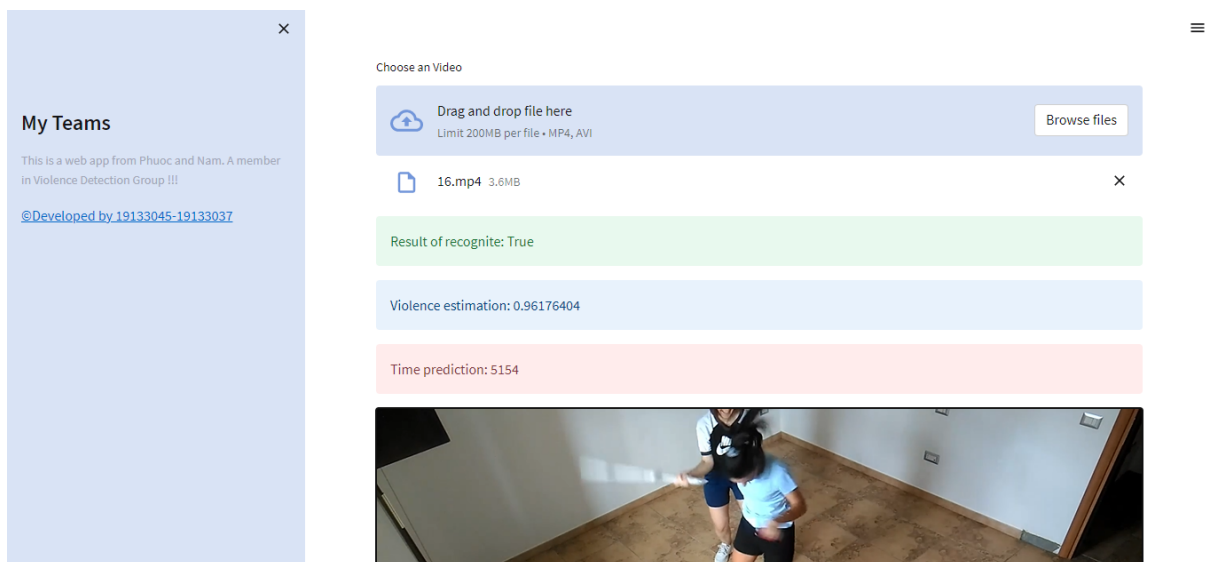
6.3. Giới thiệu ứng dụng

Ứng dụng của nhóm được triển khai với giao diện của Streamlit trên nền tảng Streamlit cloud. Người sử dụng có thể truy cập vào <https://pywind-violence-streamlit-main-9vavv1.streamlit.app/> để có thể sử dụng khả năng dự đoán của mô hình.



Hình 30: Giao diện tương tác của ứng dụng

Với giao diện trong hình 30, người dùng sẽ có tính năng upload video hoặc clip cần dự đoán. Sau đó, ứng dụng sẽ tự tính toán và trả lại kết quả cho người dùng.



Hình 31: Ứng dụng dự đoán bạo lực trả về kết quả cho người dùng

Dựa trên video hoặc clip người dùng đăng tải lên trên ứng dụng, bằng việc sử dụng mô hình, ứng dụng trả về ba thông số. Đầu tiên là kết quả có bạo lực trong video bằng “Result of recognize” là True hoặc False. Thứ hai là “violence estimation” sẽ là tỉ lệ xác suất dự đoán của mô hình trên dữ liệu đăng tải. Thứ ba là thời gian dự đoán trên video.

Với ứng dụng đơn giản này, nhóm hi vọng có thể áp dụng những mô hình học sâu vào thực tiễn và giúp ích cho cộng đồng.

PHẦN 3: Kết luận và thảo luận

1. Kết quả đạt được

Dựa trên kết quả đã nghiên cứu, nhóm đã thu được kết quả có sự kết hợp của một mô hình CNN là VGG19 và một mô hình LSTM. Mô hình VGG19 được sử dụng như là một feature extractor để chiết tách các đặc điểm của nhiều hình ảnh trong một video. Mô hình LSTM sử dụng để phát hiện chuỗi hành động trong các hình ảnh đó nhằm xác định hành vi bạo lực trong video.

Từ mô hình đã đề xuất, mô hình đạt độ chính xác khoảng 90% trên tập dữ liệu được sử dụng và mô tả - 20% trên tổng số dữ liệu ban đầu. Số liệu này thu được từ confusion matrix và tính toán thành các thông số như precision, recall và f1-score.

Những điểm cải tiến của đề tài so với nghiên cứu trước đó bao gồm: cải thiện độ chính xác của mô hình CNN+LSTM, giảm bớt độ phức tạp của lớp fully connected layer (từ 3 giảm về 2) so với mô hình năm 2019⁴¹ giúp cho thời gian huấn luyện và dự đoán giảm đáng kể.

Từ đó, nhóm lưu lại các hệ số của mô hình và triển khai trên nền tảng Streamlit cloud. Đây là một ứng dụng web có thể dự đoán trực tuyến dựa trên video mà người dùng đăng tải lên. Tốc độ của ứng dụng luôn phản hồi ở mức khoảng từ 1-2 giây.

2. Hạn chế

Tuy nhiên, bài nghiên cứu chỉ giới hạn trên một tập dữ liệu bạo lực về chủ đề nhất định nên không thể bao quát toàn bộ trường hợp xảy ra trong thực tế. Các hạn chế về mặt lựa chọn mô hình và các siêu tham số có thể là nguyên nhân dẫn đến mô hình chưa thể đạt ngưỡng mong muốn và triển khai thực tiễn, mô hình cần ít nhất đạt ngưỡng 95-97% để có thể đạt chỉ tiêu.

Với ứng dụng được nhóm thực hiện, nhóm nhận thấy ứng dụng còn tương đối đơn giản. Ứng dụng này chỉ có thể mang tính chất tham khảo, sử dụng cho mục đích dự đoán và nghiên cứu, chưa đủ khả năng ứng dụng thực tế

⁴¹ <https://paperswithcode.com/paper/violence-recognition-from-videos-using-deep>

3. Hướng phát triển

Nghiên cứu này có thể sử dụng như một phần của tài liệu tham khảo để có thể tái tạo một hệ thống dự đoán bạo lực theo thời gian thực. Việc triển khai các mô hình lên trên các hệ thống sensor như camera có thể giúp cho bài toán được ứng dụng vào thực tiễn.

Nếu có thể đưa mô hình lên hệ thống camera và triển khai lưu trữ lại kết quả có thể giúp cho quá trình phân tích dữ liệu và nghiên cứu. Nếu có thời gian và tài nguyên để tiếp tục nghiên cứu, nhóm dự kiến nghiên cứu thử nghiệm và so sánh nhiều mô hình hoặc những mô hình SOTA hiện có nhằm tìm ra mô hình có độ chính xác tốt nhất. Bên cạnh đó, nhóm muốn xây dựng một hệ thống phát hiện bạo lực được tích hợp (A fully integrated) để phát triển thành khóa luận tốt nghiệp.

References

1. Singh, S., et al., *Video Vision Transformers for Violence Detection*. 2022: p. arXiv: 2209.03561.
2. Fei-Fei Li, J.W., Jiajun Wu. *CS231n Convolutional Neural Networks for Visual Recognition*. CS231n: Deep Learning for Computer Vision 2022; Available from: <https://cs231n.github.io/>.
3. O'Shea, K. and R. Nash, *An introduction to convolutional neural networks*. arXiv preprint arXiv:1511.08458, 2015.
4. Simonyan, K. and A.J.a.p.a. Zisserman, *Very deep convolutional networks for large-scale image recognition*. 2014.
5. Bao, W., J. Yue, and Y.J.P.o. Rao, *A deep learning framework for financial time series using stacked autoencoders and long-short term memory*. 2017. **12**(7): p. e0180944.
6. Werbos, P.J., *Backpropagation through time: what it does and how to do it*. Proceedings of the IEEE, 1990. **78**(10): p. 1550-1560.
7. Weiss, K., T.M. Khoshgoftaar, and D.J.J.o.B.d. Wang, *A survey of transfer learning*. 2016. **3**(1): p. 1-40.
8. Elesawy, M., *Real Life Violence Situations Dataset*. 2019, Mohamad Hussein (Editor), Mina Abd El Massih (Editor).
9. Soliman, M.M., et al. *Violence recognition from videos using deep learning techniques*. in *2019 Ninth International Conference on Intelligent Computing and Information Systems (ICICIS)*. 2019. IEEE.