

**TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN**



BÀI TIỂU LUẬN GIỮA KỲ

NHẬP MÔN THỊ GIÁC MÁY TÍNH

Người hướng dẫn: **TS PHẠM VĂN HUY**

Người thực hiện: **NGUYỄN HOÀNG KHOA – 52100969**

HUỲNH THỊ TRÀ MY - 52100704

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024

**TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN**



BÀI TIỂU LUẬN GIỮA KỲ

NHẬP MÔN THỊ GIÁC MÁY TÍNH

Người hướng dẫn: **TS PHẠM VĂN HUY**

Người thực hiện: **NGUYỄN HOÀNG KHOA – 52100969**

HUỲNH THỊ TRÀ MY - 52100704

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024

LỜI CẢM ƠN

Chúng em xin gửi lời cảm ơn chân thành đến Trường Đại học Tôn Đức Thắng và Khoa Công nghệ Thông tin đã tạo điều kiện cho Chúng em được học tập và nghiên cứu trong một môi trường chuyên nghiệp, hiện đại với đầy đủ các nguồn tài nguyên phục vụ cho việc học tập và phát triển chuyên môn.

Chúng em đặc biệt bày tỏ lòng biết ơn sâu sắc đến TS. Phạm Văn Huy, người đã tận tình hướng dẫn, cung cấp kiến thức và định hướng cho Chúng em trong suốt quá trình thực hiện bài tiểu luận này. Sự hướng dẫn của thầy không chỉ giúp Chúng em hiểu rõ hơn về các mô hình học sâu trong thị giác máy tính mà còn giúp Chúng em rèn luyện tư duy phân tích, kỹ năng nghiên cứu và khả năng ứng dụng lý thuyết vào thực tiễn. Những góp ý và phản hồi của thầy trong từng giai đoạn nghiên cứu đã giúp Chúng em nhận ra những điểm cần cải thiện, từ đó hoàn thiện bài tiểu luận một cách tốt nhất.

Ngoài ra, Chúng em cũng xin gửi lời cảm ơn đến các giảng viên trong Khoa Công nghệ Thông tin đã truyền đạt những kiến thức nền tảng vững chắc, giúp Chúng em có đủ khả năng tiếp cận với các lĩnh vực chuyên sâu như thị giác máy tính. Các môn học và bài giảng của khoa đã cung cấp cho Chúng em tư duy hệ thống, phương pháp nghiên cứu khoa học và cách tiếp cận các công nghệ hiện đại.

Bài tiểu luận này là kết quả của quá trình học hỏi và nỗ lực không ngừng. Chúng em vô cùng trân trọng những cơ hội mà nhà trường, khoa và thầy cô đã mang lại. Một lần nữa, Chúng em xin chân thành cảm ơn!

ĐỒ ÁN ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là sản phẩm đồ án của riêng tôi / chúng tôi và được sự hướng dẫn của TS Phạm Văn Huy. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình. Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày 22 tháng 12 năm 2024

Tác giả

(ký tên và ghi rõ họ tên)

Nguyễn Hoàng Khoa

Huỳnh Thị Trà My

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của GV hướng dẫn

Tp. Hồ Chí Minh, ngày tháng năm
(kí và ghi họ tên)

Phần đánh giá của GV chấm bài

Tp. Hồ Chí Minh, ngày tháng năm
(kí và ghi họ tên)

TÓM TẮT

Bài tiểu luận này trình bày về các mô hình học sâu quan trọng trong thị giác máy tính, bao gồm Convolutional Neural Networks (CNN), R-CNN, Autoencoder, Vision Transformer (ViT), và Generative Adversarial Networks (GAN). Mỗi mô hình được phân tích về kiến trúc, cơ chế hoạt động, công thức toán học liên quan, cũng như những ứng dụng thực tiễn.

CNN đóng vai trò quan trọng trong việc trích xuất đặc trưng từ hình ảnh, hỗ trợ các tác vụ như phân loại và nhận diện đối tượng. R-CNN tiếp tục phát triển CNN để tối ưu hóa quá trình phát hiện vật thể. Autoencoder được sử dụng trong giảm chiều dữ liệu, tái tạo hình ảnh và phát hiện bất thường. ViT là một phương pháp mới dựa trên Transformer, giúp cải thiện độ chính xác của các bài toán phân loại hình ảnh. Cuối cùng, GAN mở ra một hướng đi mạnh mẽ trong việc tạo dữ liệu tổng hợp, giúp phát triển các công nghệ như deepfake, tăng cường ảnh và mô phỏng dữ liệu thực tế.

Bài viết cũng thảo luận về các thách thức của từng mô hình và so sánh hiệu suất của chúng trong các lĩnh vực ứng dụng khác nhau.

MỤC LỤC

LỜI CẢM ƠN	2
TÓM TẮT	iv
MỤC LỤC.....	1
DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ	4
DANH MỤC HÌNH	4
DANH MỤC BẢNG.....	4
CHƯƠNG I - CONVOLUTIONAL NEURAL NETWORK (CNN).....	5
1.1. Giới thiệu	5
1.2. Kiến trúc của CNN.....	5
1.2.1. Lớp tích chập (Convolutional Layer)	5
1.2.2. Lớp kích hoạt (Activation Layer)	8
1.2.3. Lớp pooling (Pooling Layer)	8
1.2.4. Lớp kết nối đầy đủ (Fully Connected Layer).....	8
1.2.5. Lớp chuẩn hóa (Normalization Layer)	9
1.3. Quy trình hoạt động	9
1.4. Ứng dụng.....	9
1.5. Lợi ích và thách thức.....	10
1.6. Kết luận	10
1.7. Demo	10
1.7.1. Kiến trúc mạng (Model Architecture)	10
1.7.2. Biểu đồ huấn luyện (Training History).....	11
1.7.3. Kết quả đánh giá (Test Evaluation)	12
1.7.4. Dự đoán và phân tích mẫu (Prediction Analysis)	13
CHƯƠNG II - R-CNN	15
2.1. Giới thiệu	15
2.2. Kiến trúc và Cơ chế Hoạt động.....	15

2.3. Các Công thức Toán học trong R-CNN.....	16
2.4. Hạn Chế và Hiệu Năng	17
2.5. Ảnh Hưởng và Mở Rộng	17
2.6. Kết Luận.....	17
2.7. Demo	17
2.7.1. Kết quả bảng dữ liệu	17
2.7.2. Kết quả hình ảnh	18
CHƯƠNG III - AUTO ENCODER	20
3.1. Giới thiệu	20
3.2. Kiến trúc của Auto Encoder.....	20
3.3. Quy trình hoạt động	22
3.4. Ứng dụng.....	23
3.5. Nhược điểm.....	23
3.6. Kết luận	24
3.7. Demo	24
CHƯƠNG IV - VISION TRANSFORMER.....	26
4.1. Giới thiệu	26
4.2. Kiến trúc mô hình	26
4.2.1. Tổng quan mô hình	26
4.2.2. Công thức toán học chính	28
4.2.3. Kết thúc Hybrid.....	28
4.2.4. Định kiến quy nạp.....	28
4.3. Ứng dụng, ưu điểm và nhược điểm	28
4.4. Kết luận	29
4.5. Demo	29
CHƯƠNG V - GAN.....	32
5.1. Giới thiệu	32

5.2. Kiến trúc.....	32
5.2.1. Cấu trúc chính	32
5.2.2. Cơ chế hoạt động	32
5.2.3. Kết quả	32
5.2.4. Huấn luyện GAN	32
5.2.5. Thuật toán huấn luyện.....	33
5.2.6. Các thách thức khi huấn luyện.....	34
5.2.7. Các phiên bản cải tiến của GAN.....	35
5.3. Ưu điểm.....	35
5.4. Nhược điểm.....	36
5.5. Kết luận	36
5.6. Demo	36
5.6.1. Kết quả huấn luyện	36
5.6.2. Kết quả hàm mất mát sau khi huấn luyện GAN	37
CHƯƠNG VI - TÀI LIỆU THAM KHẢO.....	38

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

DANH MỤC HÌNH

Hình 1: Mô hình CNN.....	5
Hình 2: Ma trận Input và Kernel trong CNN	6
Hình 3: Tính tích chập ma trận Input với ma trận Kernel trong CNN.....	6
Hình 4: Ma trận Kernel cho các mục đích thông dụng	7
Hình 5: Kết quả kiến trúc mạng	11
Hình 6: Biểu đồ hàm Loss của mô hình CNN	12
Hình 7: Biểu đồ độ chính xác của mô hình CNN	12
Hình 8: Kết quả độ chính xác của mô hình CNN	13
Hình 9: Kết quả dự đoán của mô hình CNN.....	13
Hình 10: Sơ đồ pipeline xử lý trong mô hình mạng R-CNN.....	15
Hình 11: Kết quả demo RCNN	18
Hình 12: Cấu trúc chung của AutoEncoder	20
Hình 13: Các thành phần bên trong AutoEncoder	21
Hình 14: Demo AutoEncoder.....	24
Hình 15: Tổng quan mô hình ViT.....	27
Hình 16: Demo ViT	30
Hình 17: Quá trình huấn luyện GAN	32
Hình 18: Hàm mất mát của Discriminator (GAN).....	34
Hình 19: Hàm mất mát của Generator (GAN).....	34
Hình 20: Ảnh chữ số viết tay được tạo ra từ mô hình GAN sau khi huấn luyện.....	36
Hình 21: Biểu đồ hàm mất mát của Generator và Discriminator (GAN)	37

DANH MỤC BẢNG

Bảng 1: Bảng thông tin bounding box RCNN	17
Bảng 2: Bảng kết quả phân loại hình ảnh đầu vào của mô hình ViT	30

CHƯƠNG I - CONVOLUTIONAL NEURAL NETWORK (CNN)

1.1. Giới thiệu

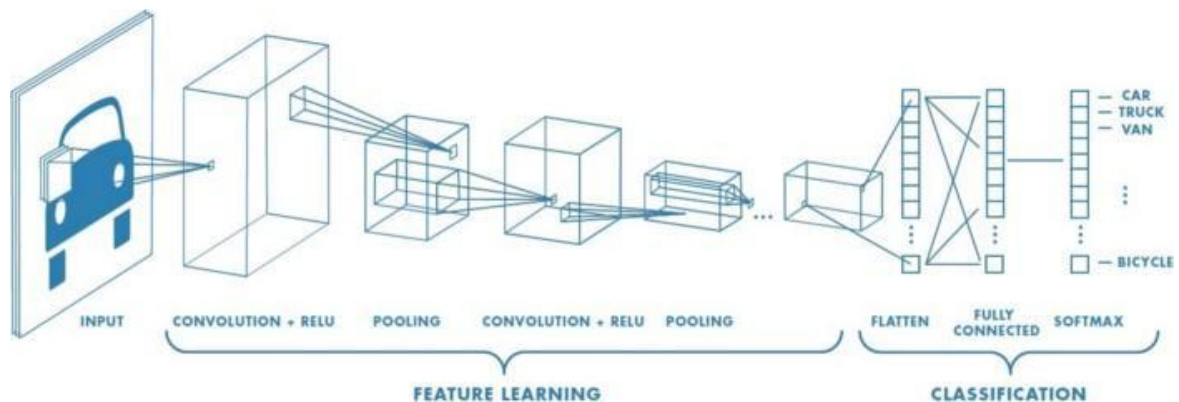
Mạng Nơ-ron Tích chập (CNN) ra đời là một cuộc cách mạng trong lĩnh vực học sâu (deep learning). Được lấy cảm hứng từ cấu trúc của thị giác sinh học, CNN sử dụng các lớp tích chập (convolutional layers) để tự động học các đặc trưng từ dữ liệu đầu vào mà không cần phải thiết kế thủ công.

Trong lĩnh vực thị giác máy tính, CNN có thể khai thác thông tin từ hình ảnh, nhận diện đối tượng trong ảnh, phân đoạn ngữ nghĩa, đến các ứng dụng phức tạp như tạo ảnh và phát hiện bệnh lý.

Việc hiểu rõ cơ chế hoạt động của CNN là bước đầu quan trọng để khai thác tiềm năng của nó trong các ứng dụng thực tế.

1.2. Kiến trúc của CNN

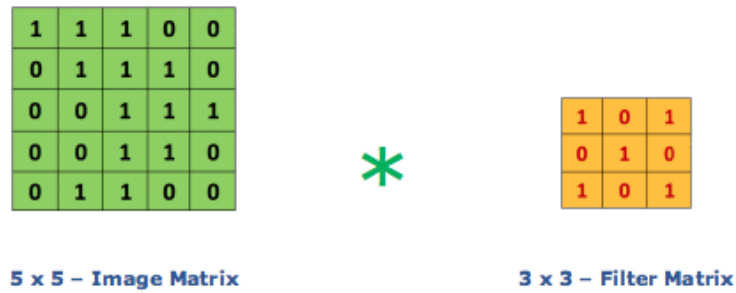
Kiến trúc CNN thường bao gồm các thành phần chính sau:



Hình 1: Mô hình CNN

1.2.1. Lớp tích chập (Convolutional Layer)

Lớp tích chập là nền tảng của CNN, thực hiện phép toán tích chập giữa ma trận đầu vào và một tập kernel (hay filter). Các kernel này hoạt động như những bộ phát hiện đặc trưng, giúp mô hình học được các mẫu như cạnh, góc, hoặc cấu trúc phức tạp hơn trong hình ảnh.



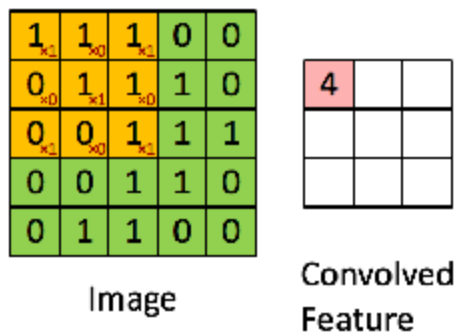
Hình 2: Ma trận Input và Kernel trong CNN

Công thức tích chập cơ bản:

$$Y[i,j] = \sum_{m=1}^M \sum_{n=1}^N X[i+m,j+n] \cdot K[m,n]$$

Trong đó:

- X : Ma trận đầu vào.
- K : Kernel (thường có kích thước là số .
- Y : Ma trận đầu ra (feature map).
- M, N : Kích thước của ma trận Kernel.



Hình 3: Tính tích chập ma trận Input với ma trận Kernel trong CNN



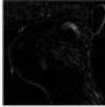




Để dễ hình dung, ma trận Kernel sẽ “trượt” trên ma trận đầu vào và thực hiện phép nhân Convolution (nhân các vị trí tương ứng trên 2 ma trận). Ta sẽ được một ma trận kết quả tạm thời. Tổng các phần tử trên ma trận tạm thời sẽ là giá trị ô trong ma trận kết quả mới

Để đảm bảo đầu ra phù hợp với kích thước mong muốn, việc áp dụng padding và stride được sử dụng. Padding thêm các giá trị 0 xung quanh biên của ma trận đầu vào, trong khi stride xác định bước di chuyển của kernel qua dữ liệu đầu vào:

$$\text{Output size} = \frac{\text{Input size} - \text{Kernel size} + 2 \cdot \text{Padding}}{\text{Stride}} + 1$$

Trong đó:

- Input size: Kích thước ma trận đầu vào.
- Kernel size: Kích thước của kernel.
- Padding: Số lượng pixel được thêm vào viền.
- Stride: Bước di chuyển của kernel.

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Hình 4: Ma trận Kernel cho các mục đích thông dụng

1.2.2. Lớp kích hoạt (Activation Layer)

Sau mỗi phép tích chập, lớp kích hoạt được áp dụng để thêm tính phi tuyến vào mô hình. Hàm ReLU (Rectified Linear Unit) là lựa chọn phổ biến nhất:

$$\text{ReLU}(x) = \max(0, x)$$

ReLU giúp tăng tốc độ học và giảm hiện tượng gradient biến mất (vanishing gradient). Ngoài ReLU, các hàm kích hoạt khác như Leaky ReLU, Sigmoid, hoặc Tanh cũng được sử dụng trong các trường hợp cụ thể.

1.2.3. Lớp pooling (Pooling Layer)

Pooling giảm kích thước của feature map, giữ lại các đặc trưng quan trọng và giảm thiểu số lượng tham số. Hai phương pháp phổ biến là Max Pooling và Average Pooling:

- **Max Pooling:** Lấy giá trị lớn nhất trong mỗi vùng.
- **Average Pooling:** Lấy trung bình các giá trị trong mỗi vùng.

Công thức cho Max Pooling:

$$Y[i, j] = \max_{(m, n) \in \text{Region}} X[m, n]$$

1.2.4. Lớp kết nối đầy đủ (Fully Connected Layer)

Lớp này kết nối tất cả các neuron từ lớp trước với các neuron trong lớp hiện tại, tạo ra một biểu diễn toàn cục của dữ liệu. Biểu thức tổng quát cho đầu ra của lớp fully connected là:

$$Y = W \cdot X + b$$

Trong đó:

- W : Ma trận trọng số.
- X : Đầu vào.
- b : Bias (độ chệch).

1.2.5. Lớp chuẩn hóa (Normalization Layer)

Batch Normalization hoặc Layer Normalization giúp tăng tốc độ huấn luyện và ổn định việc học bằng cách chuẩn hóa các giá trị trong mỗi batch hoặc lớp. Công thức cho Batch Normalization:

$$\hat{x} = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}}$$

$$y = \gamma \cdot \hat{x} + \beta$$

Trong đó:

- x : Giá trị đầu vào.
- μ, σ^2 : Trung bình và phương sai của batch.
- ϵ : Một giá trị rất nhỏ để tránh chia cho 0.
- γ, β : Các tham số học được.

1.3. Quy trình hoạt động

CNN xử lý dữ liệu thông qua một loạt các lớp tích chập, pooling và fully connected, biến đổi dữ liệu từ dạng thô thành các đặc trưng trừu tượng:

1. Tiền xử lý dữ liệu: Dữ liệu được chuẩn hóa để tăng hiệu quả học.
2. Trích xuất đặc trưng: Các lớp tích chập và pooling trích xuất các mẫu trong dữ liệu.
3. Phân loại hoặc dự đoán: Các lớp fully connected và softmax (hoặc sigmoid) thực hiện phân loại hoặc dự đoán kết quả.

1.4. Ứng dụng

CNN đã được ứng dụng rộng rãi trong nhiều lĩnh vực:

Thị giác máy tính:

- + Phân loại hình ảnh (Image Classification).
- + Phát hiện đối tượng (Object Detection).
- + Phân đoạn ảnh (Image Segmentation).
- + Xử lý y tế: Phân tích ảnh X-quang, MRI, và CT scan để chẩn đoán bệnh.

Xử lý video: Theo dõi đối tượng và nhận diện hoạt động.

Các lĩnh vực khác: Phân tích ảnh vệ tinh, nhận diện khuôn mặt, và tạo nội dung (Generative Models).

1.5. Lợi ích và thách thức

Lợi ích

- Tự động học đặc trưng: Không cần thiết kế thủ công.
- Hiệu quả cao: Thích hợp với dữ liệu lớn và phức tạp.
- Tính linh hoạt: Ứng dụng đa dạng.

Thách thức

- Đòi hỏi tài nguyên tính toán lớn: Huấn luyện CNN thường yêu cầu GPU mạnh.
- Quá khớp (Overfitting): Dễ xảy ra khi dữ liệu không đủ lớn.
- Khả năng giải thích hạn chế: CNN hoạt động như một "hộp đen," khó hiểu rõ cách mô hình đưa ra quyết định.

1.6. Kết luận

Mạng Nơ-ron Tích chập đã mở ra những cơ hội mới trong lĩnh vực học sâu, đặc biệt là thị giác máy tính. Với khả năng xử lý mạnh mẽ và linh hoạt, CNN đã và đang tiếp tục định hình các công nghệ hiện đại. Tuy nhiên, để khai thác tối đa tiềm năng của CNN, việc hiểu sâu kiến trúc và xử lý thách thức là điều cần thiết.

1.7.Demo

1.7.1. Kiến trúc mạng (Model Architecture)

- Ý nghĩa: Giới thiệu cấu trúc của CNN (số lớp, loại lớp, kích thước kernel, số lượng filter, hàm kích hoạt).

- Cách lấy:

Sử dụng kết quả từ `model.summary()` trong Notebook.

- Hiển thị kiến trúc của CNN thông qua kết quả của `model.summary()`.

Model: "sequential_3"

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d_6 (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_7 (Conv2D)	(None, 13, 13, 64)	18,496
max_pooling2d_7 (MaxPooling2D)	(None, 6, 6, 64)	0
flatten_3 (Flatten)	(None, 2304)	0
dropout_3 (Dropout)	(None, 2304)	0
dense_3 (Dense)	(None, 10)	23,050

Total params: 42,442 (165.79 KB)
 Trainable params: 42,442 (165.79 KB)
 Non-trainable params: 0 (0.00 B)

Hình 5: Kết quả kiến trúc mạng

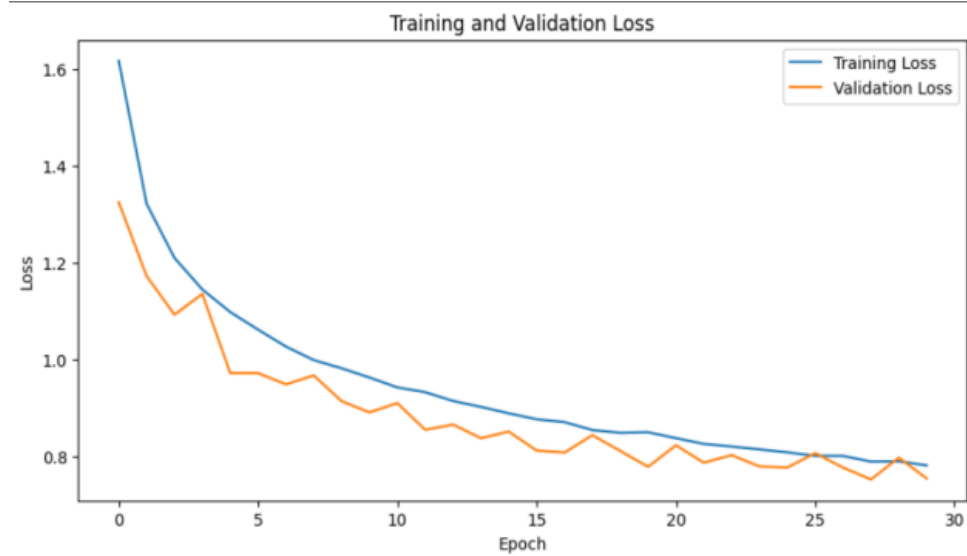
1.7.2. Biểu đồ huấn luyện (Training History)

- Ý nghĩa: Minh họa quá trình học của CNN qua từng epoch.
- Kết quả có thể sử dụng:
 - + Biểu đồ Training Loss và Validation Loss.
 - + Biểu đồ Training Accuracy và Validation Accuracy.
- Cách lấy:

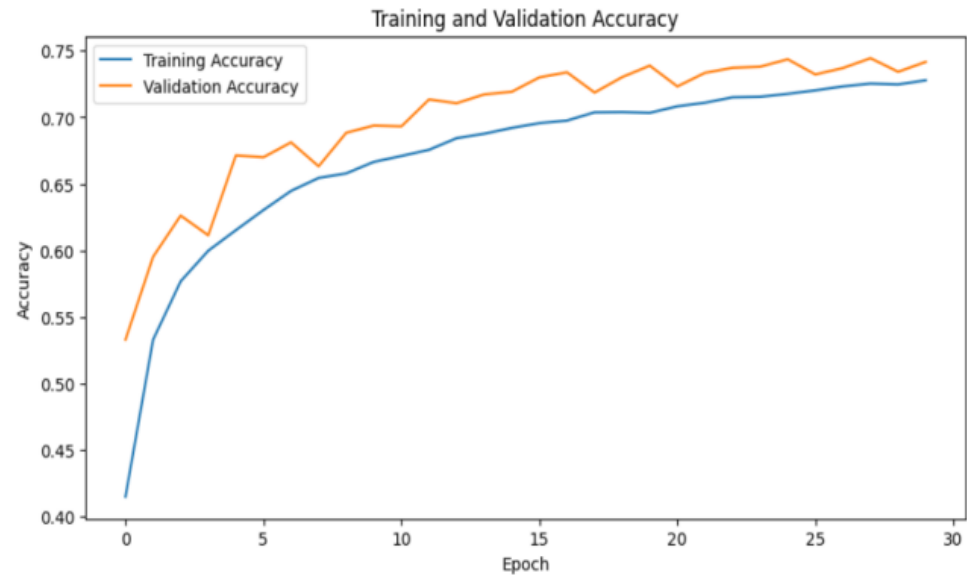
```
plt.plot(history['epoch'], history['accuracy'], label='Training Accuracy')
plt.plot(history['epoch'], history['val_accuracy'], label='Validation Accuracy')
plt.legend()
```

Hai biểu đồ riêng biệt:

- + Loss: Đường cong giảm dần qua các epoch (tốt hơn nếu Validation Loss giảm ổn định).
- + Accuracy: Đường cong tăng dần qua các epoch (tốt hơn nếu Validation Accuracy đạt giá trị cao và ổn định).



Hình 6: Biểu đồ hàm Loss của mô hình CNN



Hình 7: Biểu đồ độ chính xác của mô hình CNN

1.7.3. Kết quả đánh giá (Test Evaluation)

- Ý nghĩa: Đánh giá chất lượng của mô hình sau khi huấn luyện.
- Kết quả có thể sử dụng:
 - + Test Loss: Giá trị hàm mất mát trên tập kiểm tra.
 - + Test Accuracy: Độ chính xác của mô hình trên tập kiểm tra.

- Cách lấy:

```
score = model.evaluate(x_test, y_test, verbose=0)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

- Hiển thị:

```
Test loss: 0.7783171534538269
Test accuracy: 0.7387999892234802
```

Hình 8: Kết quả độ chính xác của mô hình CNN

1.7.4. Dự đoán và phân tích mẫu (*Prediction Analysis*)

- Ý nghĩa: Minh họa khả năng phân loại ảnh của mô hình.

- Kết quả có thể sử dụng:

+ Một mẫu ảnh ngẫu nhiên từ tập kiểm tra.

+ Nhãn thực tế của mẫu ảnh (Original Label).

+ Nhãn dự đoán của mẫu ảnh (Predicted Label).

- Cách lấy

```
sample_index = np.random.randint(x_test.shape[0])
sample_image = x_test[sample_index]
prediction = model.predict(np.expand_dims(sample_image, axis=0))
predicted_label = np.argmax(prediction)
original_label = y_test[sample_index][0]
```

- Hiển thị kết quả

```
➡ 1/1 ————— 0s 24ms/step
Original Label: 7
Original Class Name: horse
Predicted Label: 7
Predicted Class Name: horse
```

Hình 9: Kết quả dự đoán của mô hình CNN

CHƯƠNG II - R-CNN

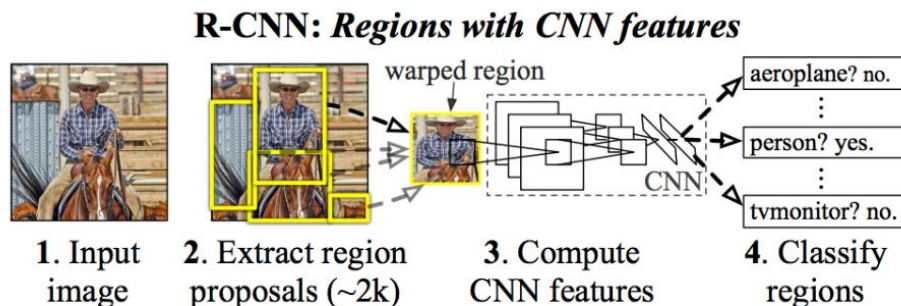
2.1. Giới thiệu

Region-based Convolutional Neural Networks (R-CNN) là một phương pháp tiên phong trong lĩnh vực nhận diện đối tượng (object detection). Được đề xuất bởi Girshick và cộng sự vào năm 2014, R-CNN kết hợp các kỹ thuật truyền thống và hiện đại để phát hiện các đối tượng trong ảnh một cách hiệu quả hơn. Cốt lõi của R-CNN là sử dụng các vùng đề xuất (region proposals) để xác định vị trí các đối tượng tiềm năng và sau đó áp dụng các mạng nơ-ron tích chập (CNN) để phân loại và dự đoán bounding box.

2.2. Kiến trúc và Cơ chế Hoạt động

R-CNN hoạt động thông qua ba bước chính: (1) tạo các vùng đề xuất bằng thuật toán Selective Search, (2) trích xuất đặc trưng từ từng vùng đề xuất thông qua một mạng CNN, và (3) phân loại các vùng này bằng SVM và tinh chỉnh bounding box bằng hồi quy. Phương pháp này kết hợp các kỹ thuật từ thị giác máy tính truyền thống với học sâu, tạo ra một cải tiến lớn so với các phương pháp trước đó.

Một cách tổng quát, đầu vào của R-CNN là một hình ảnh, và đầu ra là danh sách các bounding box và nhãn lớp tương ứng. Thuật toán Selective Search chia nhỏ hình ảnh thành hàng ngàn vùng có khả năng chứa đối tượng, sau đó mỗi vùng được đưa qua một mạng CNN để trích xuất vector đặc trưng. Các vector này sau đó được sử dụng để phân loại và dự đoán.



Hình 10: Sơ đồ pipeline xử lý trong mô hình mạng R-CNN

2.3. Các Công thức Toán học trong R-CNN

a) Trích xuất đặc trưng qua CNN

CNN nhận đầu vào là hình ảnh hoặc vùng đề xuất và trích xuất các đặc trưng thông qua nhiều lớp tích chập. Đầu ra của mạng là một vector đặc trưng:

$$F(x) = CNN(x; \theta)$$

Trong đó:

- x : Hình ảnh đầu vào hoặc vùng đề xuất.
- θ : Các tham số học được của mạng CNN.
- $F(x)$: Vector đặc trưng đầu ra.

b) Phân loại bằng SVM

Sau khi trích xuất đặc trưng, mỗi vector được đưa vào một bộ phân loại SVM để xác định lớp của vùng đó:

$$C = \operatorname{argmax}_i (w_i \cdot F(x) + b_i)$$

Trong đó:

- w_i : Vector trọng số của lớp .
- b_i : Bias của lớp .
- $F(x)$: Vector đặc trưng đầu vào.
- C : Lớp được dự đoán.

c) Hồi quy để tinh chỉnh Bounding Box

R-CNN sử dụng một bộ hồi quy tuyến tính để tinh chỉnh bounding box dựa trên đầu ra của CNN:

$$B' = W \cdot F(x) + b$$

Trong đó:

- B' : Bounding box tinh chỉnh.
- W : Ma trận trọng số của hồi quy.
- b : Bias.
- $F(x)$: Vector đặc trưng của vùng đề xuất.

2.4. Hạn Chế và Hiệu Năng

Mặc dù R-CNN là một bước tiến lớn, nó có những hạn chế đáng kể. Đầu tiên, việc áp dụng mạng CNN cho từng vùng đề xuất khiến mô hình rất chậm và không phù hợp cho ứng dụng thời gian thực. Hơn nữa, việc huấn luyện các thành phần khác nhau (CNN, SVM, hồi quy) đòi hỏi các bước riêng biệt, làm tăng độ phức tạp.

2.5. Ảnh Hưởng và Mở Rộng

R-CNN mở đường cho sự phát triển của các mô hình hiện đại hơn như Fast R-CNN và Faster R-CNN, nơi các bước phân loại và tinh chỉnh bounding box được tối ưu hóa và tích hợp trực tiếp trong một mạng duy nhất. Tuy nhiên, R-CNN vẫn đóng vai trò quan trọng trong lịch sử phát triển của nhận diện đối tượng, minh chứng cho việc kết hợp giữa các kỹ thuật truyền thống và học sâu.

2.6. Kết Luận

R-CNN là một đóng góp quan trọng trong lĩnh vực nhận diện đối tượng, đặt nền móng cho các nghiên cứu và cải tiến sau này. Mặc dù có những hạn chế về hiệu suất, mô hình đã chứng minh tiềm năng của việc sử dụng học sâu trong việc giải quyết các bài toán phức tạp trong thị giác máy tính.

2.7. Demo

2.7.1. Kết quả bảng dữ liệu

Bảng dưới đây hiển thị thông tin về bounding box được dự đoán trên ảnh đầu vào bởi mô hình Faster RCNN:

image_id	width	height	source	x	y	w	h
b6ab77fd7	1024	1024	usask_1	834	222	56	36
b6ab77fd7	1024	1024	usask_1	226	548	130	58
b6ab77fd7	1024	1024	usask_1	377	504	74	160
b6ab77fd7	1024	1024	usask_1	834	95	109	107
b6ab77fd7	1024	1024	usask_1	26	144	124	117

Bảng 1: Bảng thông tin bounding box RCNN

Một số điểm quan sát được:

- Bảng chứa thông tin về ảnh đầu vào, bao gồm kích thước và tọa độ bounding box của các đối tượng được phát hiện.

- Các tọa độ (x, y) đánh dấu góc trên bên trái của bounding box, trong khi (w, h) biểu thị chiều rộng và chiều cao.

- Bounding box có kích thước khác nhau, cho thấy mô hình có thể phát hiện nhiều vật thể với hình dạng đa dạng.

2.7.2. Kết quả hình ảnh

Hình dưới đây hiển thị các bounding box được vẽ trên ảnh đầu vào dựa trên kết quả dự đoán của mô hình:



Hình 11: Kết quả demo RCNN

Một số quan sát quan trọng:

- Các bounding box được xác định rõ ràng, thể hiện vị trí của đối tượng trong ảnh.
- Mô hình có thể phát hiện nhiều vật thể trong cùng một ảnh và khoanh vùng chính xác.

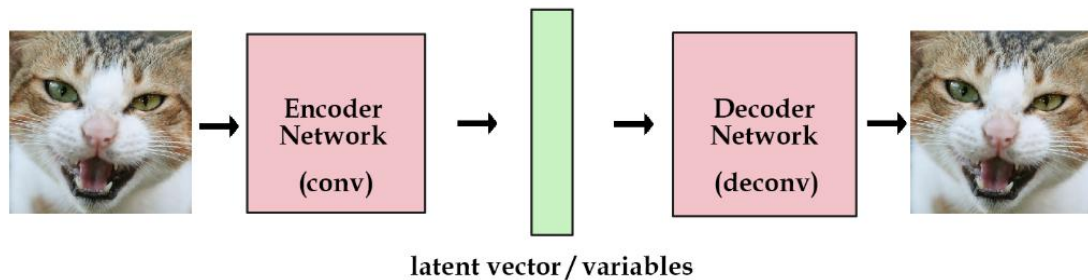
- Kích thước và vị trí bounding box phản ánh độ chính xác của mô hình trong việc nhận diện.

Kết quả trên cho thấy mô hình Faster RCNN hoạt động hiệu quả trong việc phát hiện đối tượng, tuy nhiên vẫn có thể cải thiện thêm để tăng độ chính xác và giảm nhiễu.

CHƯƠNG III - AUTO ENCODER

3.1. Giới thiệu

Bộ mã hóa tự động là một mạng nơ-ron có thể học cách nén và tái tạo lại dữ liệu đầu vào, chẳng hạn như hình ảnh, bằng cách sử dụng một lớp nơ-ron ẩn (hidden layer). Mô hình bộ mã hóa tự động bao gồm hai phần: **Encoder (bộ mã hoá)** và **Decoder (bộ giải mã)**.



Hình 12: Cấu trúc chung của AutoEncoder

Trong Autoencoder, cả **Encoder** và **Decoder** đều được tạo thành từ sự kết hợp của các lớp NN (Neural Networks), giúp giảm kích thước của hình ảnh đầu vào bằng cách tái tạo nó. Trong trường hợp của CNN Autoencoder, các lớp này là các lớp CNN (Convolutional, Max Pool, Flattening, v.v.) trong khi trong trường hợp của RNN/LSTM, các lớp tương ứng của chúng được sử dụng

3.2. Kiến trúc của Auto Encoder

Các mô hình bộ mã hóa tự động thường được sử dụng cho các tác vụ xử lý hình ảnh trong thị giác máy tính. Trong trường hợp sử dụng này, đầu vào là hình ảnh và đầu ra là hình ảnh được tái tạo. Mô hình học cách mã hóa hình ảnh thành một biểu diễn nén. Sau đó, mô hình giải mã biểu diễn này để tạo ra hình ảnh mới gần nhất có thể với đầu vào ban đầu.

Encoder:

Encoder nhận dữ liệu đầu vào (ví dụ: hình ảnh hoặc vector) và nén nó thành một vector đặc trưng tiềm ẩn có kích thước nhỏ hơn đầu vào ban đầu.

Công thức tổng quát:

$$z = f(x; \theta_e)$$

Trong đó:

- z : Biểu diễn tiềm ẩn (latent vector).
- $f()$: Hàm ánh xạ (thường là mạng nơ-ron).
- θ_e : Các tham số của encoder.

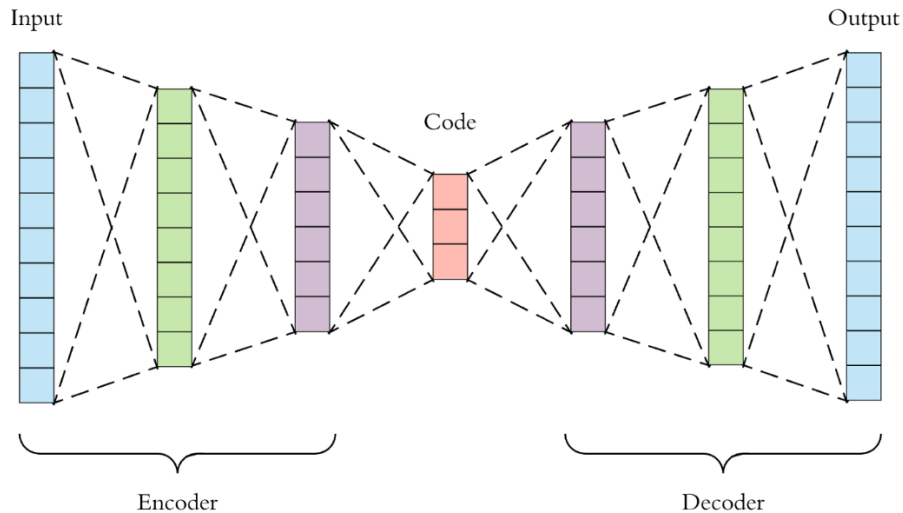
Decoder:

Decoder nhận vector z và tái tạo lại dữ liệu đầu vào x' , sao cho $x' \approx x$.

Công thức tổng quát:

$$x' = g(z; \theta_d)$$

- x' : Dữ liệu tái tạo.
- $g()$: Hàm ánh xạ (thường là mạng nơ-ron).
- θ_d : Các tham số của decoder.



Hình 13: Các thành phần bên trong AutoEncoder

Thành phần **Code** là một dạng *nút thắt cổ chai*, là phần nhỏ nhất và quan trọng nhất của mạng nơ-ron này. Nó tồn tại để hạn chế luồng thông tin đến bộ giải mã từ bộ mã hóa, do đó, chỉ cho phép thông tin quan trọng nhất đi qua. Code giúp chúng ta hình

thành biểu diễn kiến thức của đầu vào, ngăn quá khớp dữ liệu (overfitting). Thành phần Code càng nhỏ, nguy cơ quá khớp càng thấp. Lớp này thường được triển khai dưới dạng lớp tuyến tính hoặc dưới dạng tensor nếu chúng ta sử dụng phép tích chập.

Các **hàm ánh xạ** (mapping functions) ở các tầng nơ-ron thường sử dụng các hàm kích hoạt (activation functions). Các hàm kích hoạt là thành phần quan trọng trong mạng nơ-ron để giới thiệu tính phi tuyến, giúp mạng có khả năng học và biểu diễn các mối quan hệ phức tạp trong dữ liệu.

Các hàm ánh xạ (hàm kích hoạt) phổ biến:

- Sigmoid: Ánh xạ giá trị vào khoảng $(0, 1)$, giúp mô hình phù hợp với dữ liệu xác suất.
- Tanh: Tương tự Sigmoid nhưng đầu ra nằm trong khoảng $(-1, 1)$, thường dùng trong biểu diễn tiềm ẩn.
- ReLU/LeakyReLU: Hiệu quả trong xử lý gradient và thường dùng trong encoder và decoder.
- Softmax: Chuyển đổi giá trị thành xác suất, dùng ở lớp đầu ra khi phân loại.

3.3. Quy trình hoạt động

Hình ảnh đầu vào: Bộ mã hóa tự động lấy hình ảnh làm đầu vào, thường được biểu diễn dưới dạng ma trận các giá trị pixel. Hình ảnh đầu vào có thể có kích thước bất kỳ, nhưng nó thường được chuẩn hóa để cải thiện hiệu suất của bộ mã hóa tự động.

Mã hóa: Bộ mã hóa tự động nén hình ảnh đầu vào thành biểu diễn có chiều thấp hơn, được gọi là không gian tiềm ẩn, bằng cách sử dụng bộ mã hóa. Bộ mã hóa là một chuỗi các lớp tích chập trích xuất các mức đặc trưng khác nhau từ hình ảnh đầu vào. Mỗi lớp áp dụng một bộ lọc cho hình ảnh đầu vào và xuất ra một bản đồ đặc trưng làm nổi bật các mẫu và cấu trúc cụ thể trong hình ảnh.

Biểu diễn tiềm ẩn: Đầu ra của bộ mã hóa là biểu diễn nén của hình ảnh đầu vào trong không gian tiềm ẩn. Biểu diễn tiềm ẩn này nắm bắt các tính năng quan trọng nhất của hình ảnh đầu vào và thường là biểu diễn theo chiều nhỏ hơn của hình ảnh đầu vào.

Giải mã: Bộ mã hóa tự động tái tạo lại hình ảnh đầu vào từ biểu diễn tiềm ẩn bằng bộ giải mã. Bộ giải mã là một tập hợp nhiều lớp giải mã, tăng dần kích thước của bản đồ đặc trưng cho đến khi đầu ra cuối cùng có cùng kích thước với hình ảnh đầu vào. Mỗi lớp áp dụng một bộ bộ lọc để lấy mẫu bản đồ đặc trưng, tạo ra hình ảnh được tái tạo.

Hình ảnh đầu ra: Đầu ra của bộ giải mã là hình ảnh được tái tạo tương tự như hình ảnh đầu vào. Tuy nhiên, hình ảnh được tái tạo có thể không giống với hình ảnh đầu vào do bộ mã hóa tự động đã học cách nắm bắt các đặc điểm quan trọng nhất của hình ảnh đầu vào trong biểu diễn tiềm ẩn.

3.4. Ứng dụng

- Thị giác máy tính:
 - + Khử nhiễu hình ảnh
 - + Nén hình ảnh
 - + Truy xuất hình ảnh
 - + Tạo hình ảnh
- Y tế: Cải thiện hình ảnh MRI bằng loại bỏ nhiễu và thành phần giả
- Lĩnh vực khác: Nhận dạng khuôn mặt, phát hiện điểm bất thường, phát hiện đặc điểm, bảo đảm chất lượng các ứng dụng sản xuất công nghiệp

3.5. Nhược điểm

Quá khớp (Overfitting): Autoencoder có thể chỉ học cách sao chép dữ liệu đầu vào thay vì trừu tượng hóa các biểu diễn quan trọng.

Hiệu quả phụ thuộc vào thiết kế: Cần lựa chọn cẩn thận số lượng tầng, số lượng nơ-ron, và các hàm kích hoạt để đạt được hiệu quả tốt nhất.

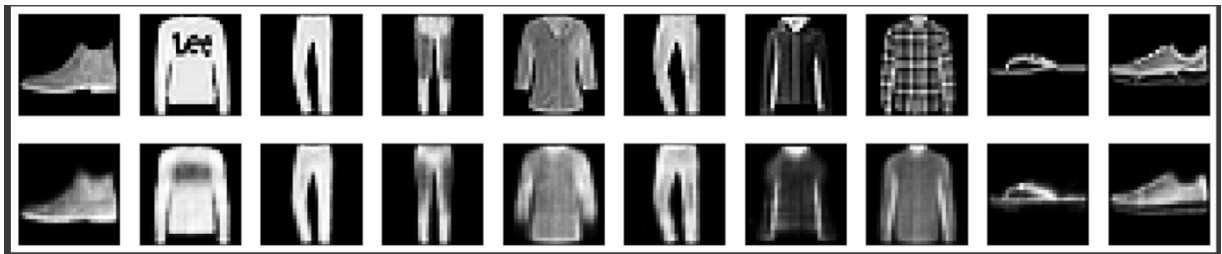
Không đảm bảo ý nghĩa biểu diễn: Không phải lúc nào biểu diễn tiềm ẩn cũng dễ giải thích hoặc có ý nghĩa trực quan.

Khó huấn luyện trên dữ liệu lớn: Autoencoder đòi hỏi lượng lớn tài nguyên tính toán, đặc biệt với các dữ liệu phức tạp như hình ảnh độ phân giải cao.

3.6. Kết luận

Autoencoder là một công cụ mạnh mẽ trong học sâu, đặc biệt trong các bài toán giảm chiều dữ liệu, phát hiện bất thường và học biểu diễn tiềm ẩn. Tuy nhiên, hiệu quả của chúng phụ thuộc vào thiết kế mô hình và dữ liệu. Các mở rộng như VAE, Denoising Autoencoder, và CAE đã cải thiện tính ứng dụng của autoencoder trong nhiều lĩnh vực. Với sự phát triển của AI, autoencoder vẫn tiếp tục là một công cụ quan trọng, mở ra các cơ hội mới trong học không giám sát và học biểu diễn.

3.7. Demo



Hình 14: Demo AutoEncoder

Giải thích hình ảnh minh họa:

Hàng trên là hình ảnh gốc từ tập kiểm tra, hàng dưới là hình ảnh được tái tạo bởi mô hình Autoencoder.

Mô hình Autoencoder được huấn luyện trên tập dữ liệu Fashion-MNIST, gồm các hình ảnh đen trắng 28x28 pixel của các loại quần áo, giày dép. Kết quả như sau:

Kết quả tái tạo:

Hình ảnh tái tạo giữ được cấu trúc tổng quát của hình ảnh gốc, các đối tượng như giày, áo, và quần được tái hiện rõ ràng.

Tuy nhiên:

- + Hình ảnh tái tạo có xu hướng bị mờ nhẹ, đặc biệt ở các đối tượng phức tạp (như áo có hoa văn hoặc cấu trúc nhiều chi tiết).
- + Các hình ảnh đơn giản như giày hoặc áo phông được tái tạo tốt hơn.

Hiệu suất mô hình:

- + Mô hình được huấn luyện trong 30 epoch với giá trị loss và val_loss giảm đều, không có dấu hiệu overfitting.
- + Giá trị loss giảm từ 0.0636 (epoch 1) xuống 0.0103 (epoch 30), cho thấy mô hình học hiệu quả và hội tụ tốt.

CHƯƠNG IV - VISION TRANSFORMER

4.1. Giới thiệu

Tại CHƯƠNG I - ta đã tìm hiểu về CNN như một giải pháp tiên tiến được sử dụng rộng rãi trong Thị giác máy tính. Đến năm 2020-2022, Vision Transformer (ViT) đã được giới thiệu và dần nổi lên thành một giải pháp mới mang tính cạnh tranh với mô hình CNN. Mô hình Vision Transformer (ViT) được giới thiệu trong một nghiên cứu tại hội nghị ICLR 2021. Bài báo này với tiêu đề "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale" được thực hiện bởi Neil Houlsby, Alexey Dosovitskiy cùng nhóm nghiên cứu từ Google Research Brain Team.

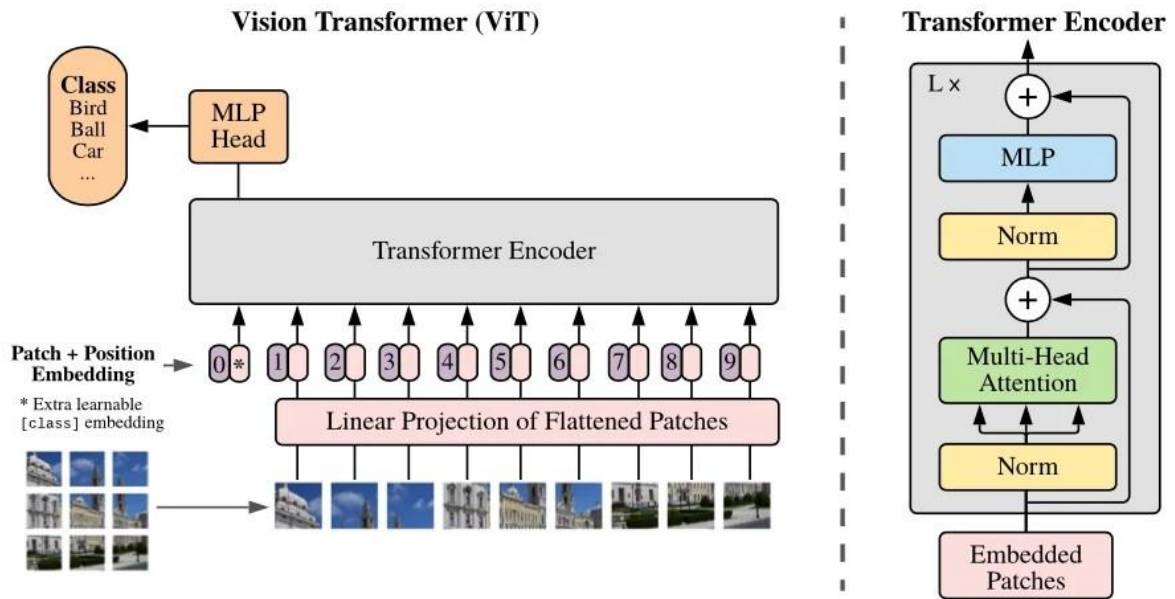
Vision Transformer (ViT) là một mô hình học sâu áp dụng kiến trúc Transformer từ NLP vào lĩnh vực thị giác máy tính, với mục tiêu xử lý hình ảnh như chuỗi các patch thay vì các pixel.

Một hình ảnh được chia thành các patch nhỏ có kích thước cố định (ví dụ: 16x16), và mỗi patch được ánh xạ thành vector thông qua lớp nhúng (embedding). Các vector này sau đó được đưa vào bộ mã hóa Transformer chuẩn, nơi khả năng tự chú ý (self-attention) được sử dụng để trích xuất các đặc trưng toàn cục và cục bộ. Để thực hiện phân loại, một "mã thông báo phân loại" (classification token) có thể học được được thêm vào chuỗi patch embedding để biểu diễn toàn bộ hình ảnh, cho phép mô hình dự đoán đầu ra cuối cùng.

4.2. Kiến trúc mô hình

4.2.1. Tổng quan mô hình

ViT chuyển đổi ảnh 2D thành một chuỗi các patch embeddings để đưa vào Transformer, mô hình vốn xử lý chuỗi 1D. Các bước chính bao gồm:



Hình 15: Tổng quan mô hình ViT

a) Chuyển đổi ảnh thành patch:

Hình ảnh $x \in \mathbb{R}^{H \times W \times C}$ được chia thành $N = HW/P^2$ patch kích thước $P \times P$.

Mỗi patch được làm phẳng và ánh xạ sang không gian ẩn có kích thước D bằng một phép chiếu tuyến tính học được.

b) Embedding và thông tin vị trí:

Một vector học được (tương tự [class] token trong BERT) được thêm vào để đại diện cho toàn bộ ảnh.

Embedding vị trí được thêm vào để giữ thông tin thứ tự không gian, sử dụng embedding 1D thay vì 2D phức tạp hơn.

c) Encoder Transformer:

Gồm các lớp luân phiên giữa multi-headed self-attention (MSA) và MLP.

LayerNorm (LN) được áp dụng trước mỗi khối và sử dụng kết nối tắt (residual connection).

d) Head phân loại:

Trong giai đoạn tiền huấn luyện, sử dụng một MLP với một tầng ẩn.

Trong giai đoạn tinh chỉnh, sử dụng một tầng tuyến tính đơn giản.

4.2.2. Công thức toán học chính

Embedding vị trí và patch:

$$z_0 = [x_{class}; x_p^1 E; x_p^2 E; \dots; x_p^N E] + E_{pos}, E \in \mathbb{R}^{(P^2 \cdot C) \times D}, E_{pos} \in \mathbb{R}^{(N+1) \times D}$$

Self-attention và MLP:

$$z'_1 = \text{MSA}(\text{LN}(z_{1-1})) + z_{1-1}, l = 1 \dots L$$

$$z_l = \text{MLP}(\text{LN}(z'_1)) + z'_1, l = 1 \dots L$$

Biểu diễn cuối cùng của ảnh

$$y = \text{LN}(z_L^0)$$

4.2.3. Kết thúc Hybrid

ViT có thể kết hợp với CNN để tận dụng đặc trưng từ bản đồ đặc trưng (feature map) của CNN, thay vì các patch ảnh thô.

Trong trường hợp đặc biệt, kích thước patch là 1×1 , giúp chuyển đổi các chiều không gian của feature map thành chuỗi đầu vào.

4.2.4. Định kiến quy nạp

So với CNN:

CNN tận dụng tốt các đặc điểm như tính địa phương, cấu trúc lân cận 2D, và tính bất biến dịch chuyển (translation equivariance) trong từng lớp.

ViT có rất ít định kiến quy nạp liên quan đến hình ảnh, đòi hỏi mô hình phải học mọi mối quan hệ không gian từ đầu.

4.3. Ứng dụng, ưu điểm và nhược điểm

Ứng dụng của ViT:

- Phân loại hình ảnh.

- Phát hiện và phân đoạn đối tượng.
- Nâng cao chất lượng hình ảnh.

Ưu điểm

Nhiều ứng dụng quan trọng như nhận diện hình ảnh, phân đoạn ảnh, phát hiện đối tượng, và chuyển đổi hình ảnh. Nhờ cơ chế Self-Attention, ViT có khả năng học các đặc trưng toàn cục tốt hơn so với CNN, giúp mô hình tổng quát hóa hiệu quả và đạt hiệu suất cao trong các tác vụ phức tạp khi được huấn luyện trên tập dữ liệu lớn. Kiến trúc của ViT cũng dễ dàng mở rộng và kết hợp với các mô hình khác, mang lại tính linh hoạt và đa năng.

Nhược điểm

ViT đòi hỏi lượng dữ liệu lớn để huấn luyện, tiêu tốn nhiều tài nguyên tính toán và có thể gặp khó khăn khi làm việc với các tập dữ liệu nhỏ hoặc hạn chế về phân cứng. Việc triển khai mô hình này cũng phụ thuộc nhiều vào pretraining trên các tập dữ liệu lớn như ImageNet. Dù vậy, với khả năng học sâu và tiềm năng phát triển, ViT đang trở thành lựa chọn hàng đầu trong các ứng dụng thị giác máy tính tiên tiến.

4.4. Kết luận

Tóm lại, ViT là một cách tiếp cận sáng tạo để áp dụng Transformer trong thị giác máy tính, với kiến trúc linh hoạt nhưng yêu cầu dữ liệu lớn để đạt hiệu quả cao.

4.5. Demo

Hình dưới đây là một ảnh được sử dụng làm đầu vào cho mô hình Vision Transformer.



Hình 16: Demo ViT

Bảng dưới đây hiển thị kết quả phân loại của ViT trên ảnh đầu vào:

Image	Pre	Confidence score
vit_input	“Mèo”	98.7%

Bảng 2: Bảng kết quả phân loại hình ảnh đầu vào của mô hình ViT

Một số điểm quan sát được:

- Mô hình ViT có thể nhận diện chính xác đối tượng trong ảnh.
- Độ tin cậy của kết quả khá cao, cho thấy ViT hoạt động tốt trên ảnh thử nghiệm.
- ViT không yêu cầu trích xuất đặc trưng bằng CNN, giúp mô hình có khả năng tổng quát tốt hơn trên nhiều loại dữ liệu hình ảnh.

Dưới đây là đoạn mã Python ngắn để thực hiện phân loại hình ảnh bằng mô hình ViT:

```
from transformers import ViTFeatureExtractor, ViTForImageClassification
from PIL import Image
import torch
```

Tải mô hình và bộ tiền xử lý

```
def load_model():
    model = ViTForImageClassification.from_pretrained("google/vit-base-patch16-224")
    processor = ViTFeatureExtractor.from_pretrained("google/vit-base-patch16-224")
    return model, processor
```

Dự đoán hình ảnh

```
def classify_image(image_path, model, processor):
    image = Image.open(image_path).convert("RGB")
    inputs = processor(images=image, return_tensors="pt")
    with torch.no_grad():
        outputs = model(**inputs)
    predicted_class = outputs.logits.argmax(-1).item()
    return model.config.id2label[predicted_class]
```

Chạy thử nghiệm

```
if __name__ == "__main__":
    model, processor = load_model()
    image_path = "vit_input.png" # Đường dẫn đến ảnh
    label = classify_image(image_path, model, processor)
    print(f'Kết quả phân loại: {label}')
```

CHƯƠNG V - GAN

5.1. Giới thiệu

GANs (Generative Adversarial Networks) là một mô hình học sâu được thiết kế để tạo ra dữ liệu mới, giống với dữ liệu thật trong một tập huấn luyện. Được đề xuất bởi Ian Goodfellow vào năm 2014, GANs sử dụng hai mạng thần kinh đối kháng để mô hình hóa và tái tạo phân phối dữ liệu, với ứng dụng rộng rãi trong các lĩnh vực như xử lý ảnh, âm thanh, và tổng hợp nội dung sáng tạo.

5.2. Kiến trúc

5.2.1. Cấu trúc chính

Gồm 2 thành phần chính:

- Generator: tạo hình ảnh giả ngày càng giống ảnh thật.
- Discriminator: phân biệt ảnh giả và ảnh thật

5.2.2. Cơ chế hoạt động

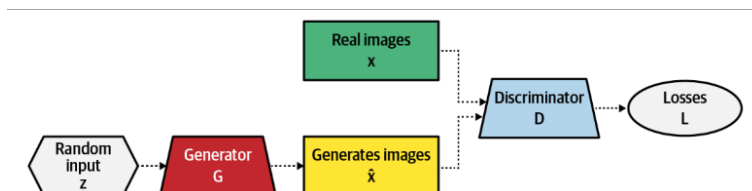
- Hai mạng cạnh tranh với nhau, giúp cả hai ngày càng cải thiện.
- Quá trình này diễn ra đến khi đạt Nash equilibrium, khi cả hai mạng không thể chiếm ưu thế hoàn toàn.

5.2.3. Kết quả

Khi Generator tạo ảnh giả chân thực đến mức Discriminator không thể phân biệt được, nó chỉ đoán ngẫu nhiên giữa ảnh thật và giả.

5.2.4. Huấn luyện GAN

Quá trình luân phiên huấn luyện giữa hai mạng cho đến khi hội tụ.



Hình 17: Quá trình huấn luyện GAN

Giải thích sơ đồ huấn luyện GAN trong Hình 17:

Random Input (z): Đầu vào ngẫu nhiên (vector nhiễu) từ phân phối ngẫu nhiên như Gaussian hoặc Uniform, đưa vào Generator.

Generator (G): Mạng thần kinh tạo ảnh giả từ z, học cách tạo ảnh giống ảnh thật x.

Real Images (x): Tập dữ liệu ảnh thật dùng để so sánh với ảnh giả.

Discriminator (D): Mạng thần kinh phân biệt ảnh thật và giả, đưa ra xác suất ảnh là thật hay giả.

Losses (L): Hàm mất mát tối ưu hóa cả hai mạng:

Generator: Lừa Discriminator tin ảnh giả là thật.

Discriminator: Phân biệt chính xác ảnh thật và giả.

Quy trình hoạt động: Generator nhận đầu vào ngẫu nhiên z, tạo ảnh giả x'. Cả ảnh thật x và ảnh giả x' được đưa vào Discriminator để đánh giá. Kết quả phân biệt được dùng để tính hàm mất mát L, từ đó cập nhật tham số của cả hai mạng. Quá trình lặp lại cho đến khi Generator tạo ra ảnh giả mà Discriminator không thể phân biệt được.

5.2.5. Thuật toán huấn luyện

Pseudocode (Minibatch Stochastic Gradient Descent)

a) Tổng quan:

Thuật toán sử dụng Minibatch Stochastic Gradient Descent (SGD) để huấn luyện một GAN cơ bản (vanilla GAN).

Gồm 2 bước chính:

- Huấn luyện Discriminator (D).
- Huấn luyện Generator (G).

b) Quy trình huấn luyện D:

Lặp lại k bước (k là siêu tham số):

- Lấy minibatch gồm m mẫu ngẫu nhiên z từ phân phối nhiễu.

- Lấy minibatch gồm m mẫu từ dữ liệu thật.
- Cập nhật trọng số của D bằng cách tối đa hóa hàm:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m \left[\log D \left(\mathbf{x}^{(i)} \right) + \log \left(1 - D \left(G \left(\mathbf{z}^{(i)} \right) \right) \right) \right].$$

Hình 18: Hàm mất mát của Discriminator (GAN)

c) Quy trình huấn luyện G :

Lặp lại một lần:

- + Lấy minibatch gồm m mẫu nhiễu \mathbf{z} .
- + Cập nhật trọng số của G bằng cách tối thiểu hóa hàm:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log \left(1 - D \left(G \left(\mathbf{z}^{(i)} \right) \right) \right).$$

Hình 19: Hàm mất mát của Generator (GAN)

d) Cơ chế:

- + D : Học cách phân biệt giữa dữ liệu thật và dữ liệu giả.
- + G : Học cách tạo ra dữ liệu giả sao cho khó phân biệt với dữ liệu thật.

5.2.6. Các thách thức khi huấn luyện

a) Mất cân bằng giữa D và G

Nếu D quá mạnh, G sẽ không thể học cách tạo ra dữ liệu giả.

Nếu G quá mạnh, D sẽ không thể phân biệt, dẫn đến hội tụ không ổn định.

b) Mode Collapse

G chỉ tạo ra một vài mẫu dữ liệu giống nhau, mất đi sự đa dạng.

Nguyên nhân: G phát hiện ra một kiểu đầu ra hiệu quả trong việc đánh lừa D .

c) Nhạy cảm với siêu số

GAN dễ bị ảnh hưởng bởi việc chọn siêu tham số (learning rate, kích thước minibatch, v.v.).

5.2.7. Các phiên bản cải tiến của GAN

Convolutional GANs: DCGAN cải thiện hiệu suất xử lý ảnh bằng cách sử dụng convolutional layers thay vì dense layers.

WGAN & WGAN-GP: Giải quyết vanishing gradient và đảm bảo huấn luyện ổn định bằng cách tối ưu hóa khoảng cách Wasserstein và áp dụng gradient penalty thay vì weight clipping.

Khắc phục Mode Collapse:

Minibatch Discrimination, Unrolled GAN, PacGAN: Đảm bảo tính đa dạng của dữ liệu sinh.

Progressive Growing GAN: Huấn luyện dần từ độ phân giải thấp đến cao để sinh ảnh chất lượng cao.

Conditional GAN (cGAN): Sinh dữ liệu có điều kiện dựa trên nhãn hoặc thông tin.

Self-Attention GAN (SAGAN): Sử dụng self-attention để cải thiện chất lượng ảnh lớn.

StyleGAN: Tạo ảnh siêu thực và kiểm soát chi tiết trong ảnh.

GAN kết hợp khác: CycleGAN, Pix2Pix, InfoGAN mở rộng ứng dụng cho nhiều miền dữ liệu khác nhau.

5.3. Ưu điểm

Tạo dữ liệu chất lượng cao và sáng tạo (ảnh, âm thanh, văn bản).

Linh hoạt, ứng dụng đa dạng (chuyển đổi miền, tăng độ phân giải, sinh dữ liệu có kiểm soát).

Không cần dữ liệu ghép cặp trong nhiều biến thể.

Mô hình hóa phân phối dữ liệu phức tạp.

Tiềm năng sáng tạo, không chỉ sao chép dữ liệu.

5.4. Nhược điểm

Khó huấn luyện, dễ gặp vanishing gradient hoặc mode collapse.

Đòi hỏi nhiều dữ liệu và tài nguyên tính toán.

Khó đánh giá chất lượng đầu ra và hội tụ.

Nhạy cảm với siêu tham số.

Hạn chế trong các bài toán ngoài dữ liệu hình ảnh hoặc âm thanh.

Dễ đạt trạng thái cân bằng không tối ưu.

5.5. Kết luận

GANs là một trong những bước tiến quan trọng trong lĩnh vực học sâu, mang lại khả năng mạnh mẽ trong việc mô phỏng và sáng tạo dữ liệu mới. Với ứng dụng rộng rãi trong xử lý ảnh, âm thanh, và văn bản, GANs đã chứng minh tiềm năng vượt trội trong việc giải quyết các bài toán phức tạp. Tuy nhiên, để khai thác tối đa lợi ích từ GANs, cần khắc phục những thách thức về huấn luyện và tối ưu hóa. Với sự phát triển liên tục, GANs hứa hẹn mở ra nhiều hướng đi mới, từ nghệ thuật số đến khoa học dữ liệu, mang lại giá trị thực tiễn và sáng tạo vượt bậc.

5.6.Demo

5.6.1.Kết quả huấn luyện



Hình 20: Ảnh chữ số viết tay được tạo ra từ mô hình GAN sau khi huấn luyện

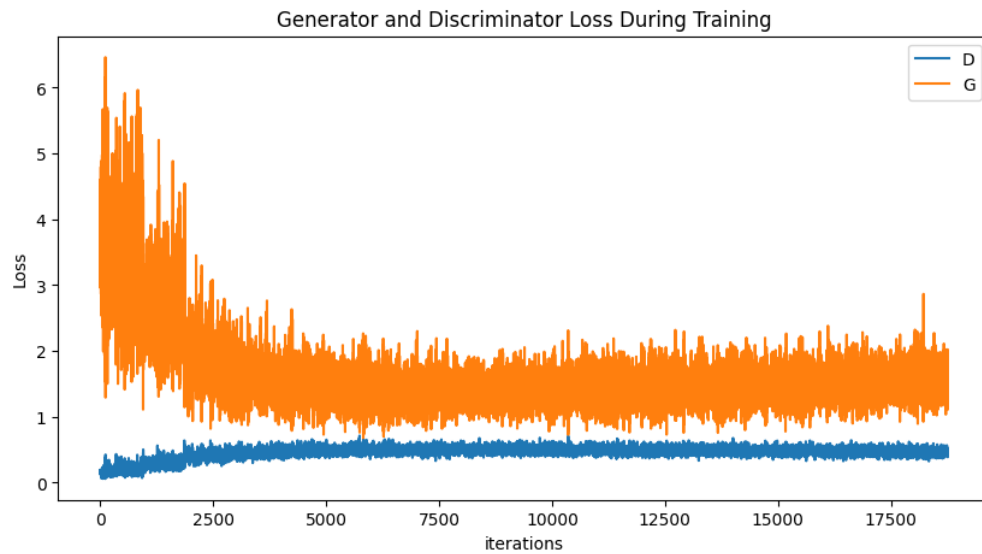
Hình trên hiển thị một tập hợp các số viết tay được tạo ra bởi Generator sau khi huấn luyện. Một số quan sát quan trọng:

Các số 3, 5, 6, 7, 9 có hình dáng rõ ràng và dễ nhận diện. Các số 3, 5, 6, 7, 9 có hình dáng rõ ràng và dễ nhận diện.

Một số số như 0, 1 có thể bị biến dạng nhẹ, nhưng vẫn mang đặc điểm của số viết tay thật.

Một số mẫu ảnh có hiện tượng mờ hoặc nhiễu nhẹ, cho thấy mô hình vẫn có thể cải thiện thêm.

5.6.2. Kết quả hàm mất mát sau khi huấn luyện GAN



Hình 21: Biểu đồ hàm mất mát của Generator và Discriminator (GAN)

Biểu đồ trên hiển thị quá trình thay đổi giá trị mất mát (loss) của Generator (G) và Discriminator (D) trong suốt quá trình huấn luyện. Một số điểm quan sát được:

Ban đầu, loss của Generator rất cao nhưng giảm dần theo thời gian, cho thấy mô hình dần học được cách tạo ảnh chất lượng hơn.

Loss của Discriminator dao động ổn định quanh một giá trị nhất định, phản ánh sự cân bằng giữa hai mô hình.

Sự dao động nhẹ ở cuối quá trình huấn luyện có thể do Generator đang cố gắng tạo ra ảnh phức tạp hơn.

CHƯƠNG VI - TÀI LIỆU THAM KHẢO

Tiếng Anh

- [1]. Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. “ImageNet Classification with Deep Convolutional Neural Networks.” Advances in Neural Information Processing Systems (NeurIPS), 2012.
- [2]. Alexey Dosovitskiy et al. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.” International Conference on Learning Representations (ICLR), 2021.
- [3]. Ian Goodfellow et al. “Generative Adversarial Nets.” Advances in Neural Information Processing Systems (NeurIPS), 2014.
- [4]. Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT Press, 2016.
- [5]. Ross Girshick et al. “Rich feature hierarchies for accurate object detection and semantic segmentation.” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014.