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

BÁO CÁO GIỮA KỲ

NHẬP MÔN THỊ GIÁC MÁY TÍNH

Người hướng dẫn: **TS PHẠM VĂN HUY**

Người thực hiện: **HÀ TRỌNG NGUYỄN - 52200148**

CHUNG THÁI KIỆT - 52200141

Lớp **: 22050301**

Khoá **: 26**

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

BÁO CÁO GIỮA KỲ

NHẬP MÔN THỊ GIÁC MÁY TÍNH

Người hướng dẫn: **TS PHẠM VĂN HUY**

Người thực hiện: **HÀ TRỌNG NGUYỄN - 52200148**

CHUNG THÁI KIỆT - 52200141

Lớp **: 22050301**

Khoá **: 26**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024

LỜI CẢM ƠN

Trước tiên, nhóm chúng tôi xin gửi lời cảm ơn chân thành đến Trường Đại học Tôn Đức Thắng và Khoa Công nghệ Thông tin đã tạo điều kiện và cung cấp môi trường học tập, nghiên cứu thuận lợi để chúng tôi có thể hoàn thành bài báo cáo này.

Chúng tôi xin bày tỏ lòng biết ơn sâu sắc đến thầy Phạm Văn Huy đã tận tình hướng dẫn, cung cấp những kiến thức chuyên môn, và luôn hỗ trợ chúng tôi trong suốt quá trình thực hiện bài báo cáo. Sự hướng dẫn và động viên của thầy đã giúp chúng tôi vượt qua những khó khăn và đạt được kết quả như hôm nay.

BÁO CÁO ĐƯỢC HOÀN THÀNH
TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Chúng tôi xin cam đoan đây là sản phẩm báo cáo của riêng của 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 báo cáo 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 báo cáo 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 tháng năm

Tác giả

(ký tên và ghi rõ họ tên)

Hà Trọng Nguyễn

Chung Thái Kiệt

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

Báo cáo này tập trung nghiên cứu và triển khai các mô hình học sâu quan trọng trong lĩnh vực thị giác máy tính, bao gồm CNN, Mask R-CNN, AutoEncoder, GAN, và Vision Transformer. Mục tiêu của báo cáo là cung cấp cái nhìn tổng quan về lý thuyết, triển khai các mô hình thực tế trên các tập dữ liệu tiêu chuẩn, và đánh giá hiệu suất cũng như tính ứng dụng của từng mô hình.

Trong phần lý thuyết, báo cáo đã trình bày chi tiết các kiến thức nền tảng và nguyên lý hoạt động của từng mô hình. CNN được phân tích về cấu trúc và ứng dụng trong phân loại hình ảnh. Mask R-CNN được giới thiệu như một giải pháp toàn diện cho phát hiện và phân đoạn đối tượng. AutoEncoder được nghiên cứu với các biến thể như Denoising AutoEncoder để tái tạo hình ảnh. GAN được sử dụng để tạo hình ảnh giả mạo, trong đó DCGAN là mô hình đặc trưng. Vision Transformer, một kiến trúc hiện đại, được áp dụng trong bài toán phân loại hình ảnh dựa trên cơ chế Self-Attention.

Phần thực nghiệm triển khai các mô hình với dữ liệu tiêu chuẩn để minh họa khả năng ứng dụng của chúng. Các mô hình được đánh giá qua hiệu suất thực tế, thể hiện ưu điểm và hạn chế trong từng bài toán cụ thể. Kết quả cho thấy mỗi mô hình có khả năng xử lý tốt trong các ngữ cảnh khác nhau.

Phần so sánh và đánh giá đã phân tích các mô hình dựa trên các tiêu chí hiệu suất, thời gian huấn luyện, tài nguyên tính toán và khả năng ứng dụng. Báo cáo kết luận rằng mỗi mô hình có thế mạnh riêng tùy thuộc vào bài toán cụ thể. ViT và Mask R-CNN thích hợp cho các bài toán phức tạp, trong khi AutoEncoder và DCGAN là lựa chọn hiệu quả cho các bài toán cụ thể như tái tạo ảnh và tạo dữ liệu giả lập.

MỤC LỤC

DANH MỤC CHỮ VIẾT TẮT

CÁC CHỮ VIẾT TẮT

SOTA: State-of-the-art (tiên tiến nhất)

RPN: Region Proposal Network (Mạng đề xuất vùng)

RoI pooling: Region of Interest pooling (Gộp vùng quan tâm)

RoIAlign: Cải tiến của RoI pooling

CNN: Convolutional Neural Network (Mạng nơ-ron tích chập)

SVM: Support Vector Machine (Máy vectơ hỗ trợ)

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

DANH MỤC HÌNH

DANH MỤC BẢNG

CHƯƠNG 1 - GIỚI THIỆU

1.1 Tổng quan về Computer Vision

Computer Vision, hay thị giác máy tính, là một lĩnh vực nghiên cứu trong trí tuệ nhân tạo và khoa học máy tính tập trung vào việc giúp máy tính hiểu và phân tích hình ảnh hoặc video từ thế giới thực một cách tự động. Mục tiêu của lĩnh vực này là phát triển các thuật toán và mô hình để trích xuất thông tin hữu ích từ dữ liệu thị giác, phục vụ cho việc nhận diện, phân loại và dự đoán.

Các ứng dụng của Computer Vision đã và đang ngày càng mở rộng trong nhiều lĩnh vực như xe tự lái, nhận diện khuôn mặt, xử lý ảnh y tế, kiểm tra chất lượng trong công nghiệp, và giám sát an ninh. Với sự phát triển của các phương pháp học sâu (Deep Learning), Computer Vision đã đạt được những tiến bộ vượt bậc, vượt qua các phương pháp truyền thống và đem lại hiệu quả cao trong nhiều bài toán phức tạp.

1.2 Mục tiêu của báo cáo

Mục tiêu của báo cáo này là nghiên cứu và trình bày các khái niệm lý thuyết cốt lõi cũng như các ứng dụng thực tiễn của 5 chủ đề quan trọng trong lĩnh vực Computer Vision. Các chủ đề được chọn bao gồm: Convolutional Neural Networks (CNNs), R-CNN và Mask R-CNN, AutoEncoder, Generative Adversarial Networks (GAN), và Vision Transformer. Báo cáo sẽ cung cấp cái nhìn tổng quan về lý thuyết của từng mô hình, đi kèm với triển khai minh họa thông qua các ứng dụng cụ thể. Ngoài ra, báo cáo cũng đặt mục tiêu so sánh và đánh giá các mô hình nhằm phân tích ưu, nhược điểm, từ đó đề xuất các hướng phát triển trong tương lai. Thông qua việc thực hiện các mô hình và minh họa demo, báo cáo sẽ tạo nền tảng giúp người học hiểu sâu hơn về bản chất và khả năng ứng dụng của các kỹ thuật trong thị giác máy tính.

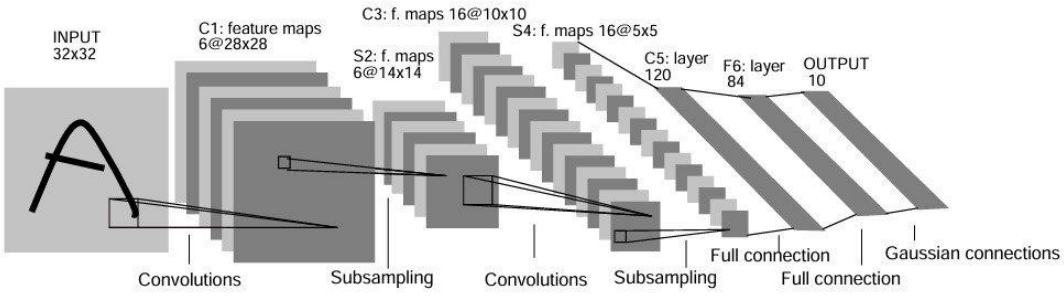
CHƯƠNG 2 - LÝ THUYẾT MÔ HÌNH

2.1 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) là các thuật toán học sâu chuyên biệt được thiết kế để xử lý dữ liệu dạng lưới, chẳng hạn như hình ảnh. Chúng đã cách mạng hóa thị giác máy tính và hiện được sử dụng rộng rãi trong các ứng dụng khác nhau, bao gồm phân loại hình ảnh, phát hiện đối tượng và xử lý ngôn ngữ tự nhiên. Phần này cung cấp một cái nhìn tổng quan toàn diện về CNNs, bao gồm lịch sử, kiến trúc, thành phần, quy trình huấn luyện và các ứng dụng của chúng. Các công thức toán học được bao gồm để cung cấp hiểu biết sâu hơn về các khái niệm cơ bản.

2.1.1 Lịch Sử của CNNs

Nền tảng cho CNNs được đặt ra vào những năm 1980 với sự phát triển của Neocognitron bởi Kunihiko Fukushima, lấy cảm hứng từ vỏ não thị giác của động vật. Tuy nhiên, phải đến năm 1998, CNN thực tế đầu tiên, LeNet-5, mới được Yann LeCun tạo ra để nhận dạng chữ số viết tay. Kiến trúc của LeNet-5, với các tầng convolutional và pooling, đã hình thành nên cơ sở cho các CNNs trong tương lai.



Hình 2.1 Ví dụ Kiến trúc của LeNet-5

Một cột mốc quan trọng trong sự phát triển của CNN là thử thách nhận dạng thị giác quy mô lớn ImageNet (ILSVRC). Vào năm 2012, AlexNet, một CNN sâu hơn và phức tạp hơn, đã vượt trội hơn đáng kể so với các phương pháp truyền thống trong ILSVRC, đánh dấu một bước ngoặt trong thị giác máy tính. Thành công của AlexNet trong ILSVRC đã châm ngòi cho sự gia tăng trong nghiên cứu CNN và dẫn đến sự phát

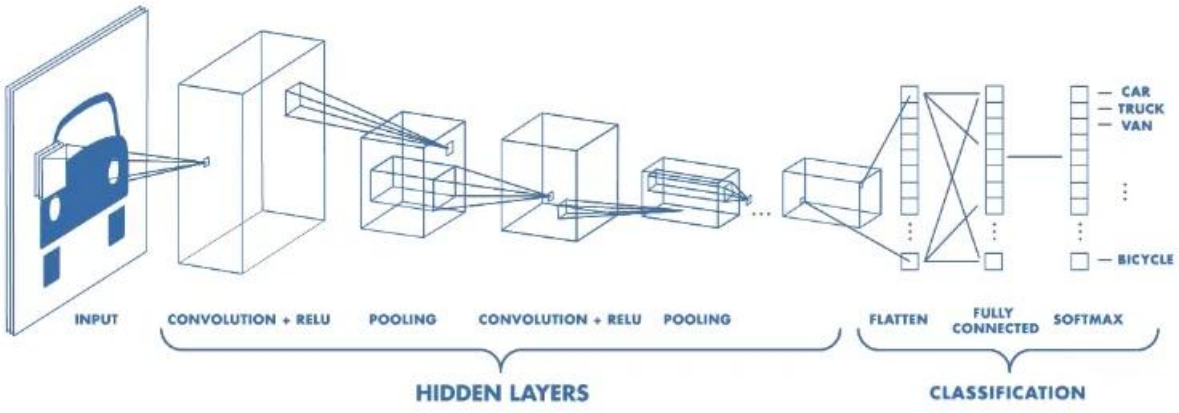
triển của nhiều kiến trúc sáng tạo, chẳng hạn như VGGNet, InceptionNet (GoogLeNet) và ResNet. Mỗi kiến trúc này đều giới thiệu những ý tưởng và cải tiến mới lạ, thúc đẩy ranh giới của khả năng nhận dạng hình ảnh.

2.1.2 Kiến trúc của CNNs

Một CNN điển hình bao gồm ba loại tầng chính:

- **Convolutional layers:** Đây là các khối xây dựng cốt lõi của CNN. Chúng áp dụng một tập hợp các bộ lọc có thể học được cho dữ liệu đầu vào để trích xuất các đặc trưng như cạnh, góc và kết cấu.
- **Pooling layers:** Các tầng này giảm kích thước không gian của feature maps, lấy mẫu dữ liệu xuống trong khi vẫn giữ lại thông tin cần thiết. Điều này làm giảm độ phức tạp tính toán và giúp ngăn ngừa quá trình overfitting.
- **Fully connected layers**: Các tầng này kết nối mọi nơ-ron trong một tầng với mọi nơ-ron trong tầng tiếp theo, thường được sử dụng trong các giai đoạn cuối cùng của CNN để phân loại hoặc hồi quy.

Ngoài các tầng này, CNNs thường bao gồm các hàm kích hoạt và các tầng dropout. Các hàm kích hoạt đưa tính phi tuyến vào mô hình, cho phép nó học các mẫu phức tạp. Các tầng Dropout sẽ hủy kích hoạt ngẫu nhiên các nơ-ron trong quá trình huấn luyện để ngăn ngừa quá trình overfitting.



Hình 2.2 Kiến trúc của CNN

2.1.3 Các thành phần của CNNs

2.1.3.1 Convolutional Layer

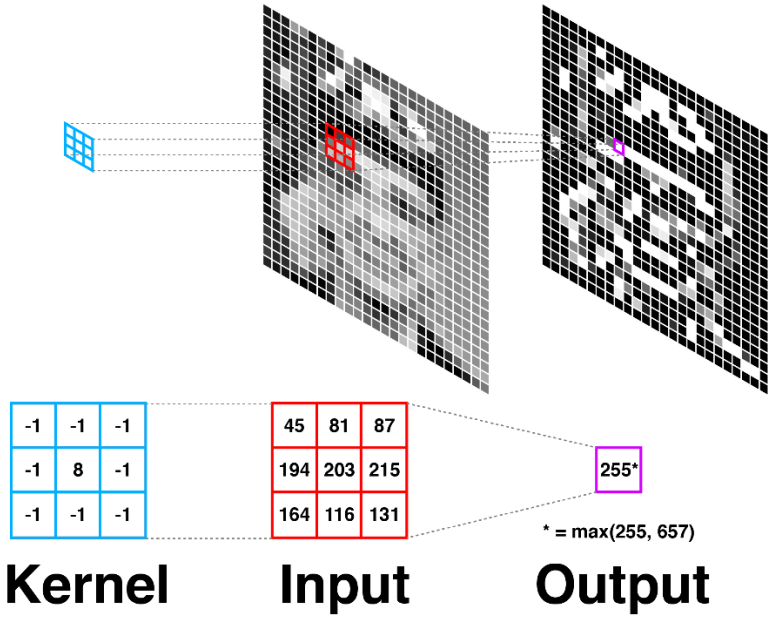
Lớp convolutional thực hiện một phép toán toán học được gọi là phép tích chập. Điều này bao gồm việc trượt một ma trận số nhỏ, được gọi là bộ lọc hoặc kernel, trên dữ liệu đầu vào (ví dụ: hình ảnh). Tại mỗi vị trí, bộ lọc được nhân theo từng phần tử với phần tương ứng của đầu vào và các kết quả được cộng lại. Quá trình này tạo ra feature maps làm nổi bật sự hiện diện của các đặc trưng cụ thể trong đầu vào.

Công thức cho phép tích chập rời rạc 2D là:

$$(f * g)(x_1, x_2) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f(i,j) \cdot g(x_1 - i, x_2 - j)$$

trong đó f là đầu vào và g là bộ lọc.

Để minh họa điều này, hãy tưởng tượng bạn có một hình ảnh và một bộ lọc phát hiện các cạnh dọc. Khi bộ lọc trượt trên hình ảnh, nó tạo ra các giá trị cao ở nơi có các cạnh dọc và các giá trị thấp ở những nơi khác. Điều này tạo ra một bản đồ đặc trưng nhấn mạnh các cạnh dọc trong hình ảnh.



Hình 2.3 Ví dụ Convolutional Layer

Kích thước đầu ra của một lớp convolutional được xác định bởi kích thước đầu vào $(W_1 \times H_1 \times D_1)$, kích thước bộ lọc (F), bước (S) và phần đệm (P):

$$W_2 = \frac{W_1 - F + 2P}{S} + 1$$

$$H_2 = \frac{H_1 - F + 2P}{S} + 1$$

$$D2 = K\ (số\ lượng\ bộ\ lọc)$$

2.1.3.2 Pooling Layer

Các lớp Pooling lấy mẫu xuống các bản đồ đặc trưng bằng cách áp dụng một phép toán pooling, chẳng hạn như max pooling hoặc average pooling, để giảm kích thước không gian của chúng. Max pooling chọn giá trị tối đa trong một cửa sổ pooling, trong khi average pooling tính giá trị trung bình. Điều này làm giảm độ phức tạp tính toán và làm cho mạng ít nhạy cảm hơn với các biến thể nhỏ trong đầu vào.

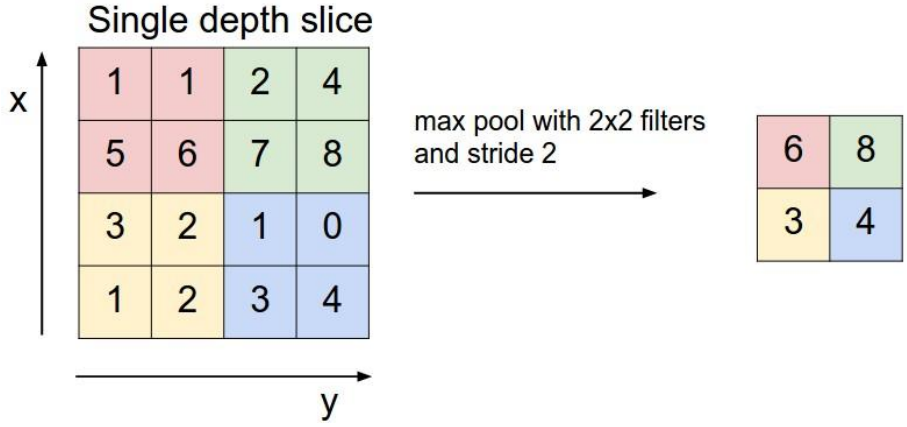
Kích thước đầu ra của một tầng pooling được tính như sau:

$$W_2 = \frac{W_1 - F}{S} + 1$$

$$H_2 = \frac{H_1 - F}{S} + 1$$

$$D_2 = D_1$$

trong đó F là kích thước cửa sổ pooling và S là bước.



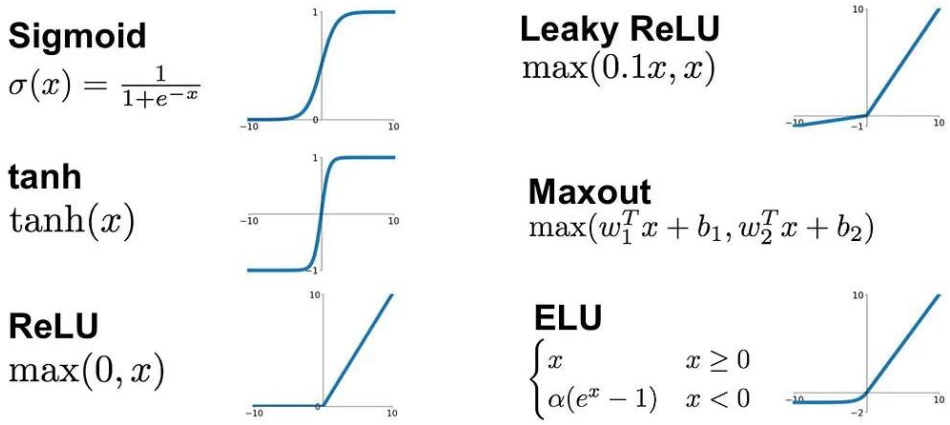
Hình 2.4 Ví dụ Max Pooling

2.1.3.3 Hàm kích hoạt

Các hàm kích hoạt đưa tính phi tuyến vào mạng, cho phép nó học các mẫu và mối quan hệ phức tạp. Không có các hàm kích hoạt, về cơ bản mạng sẽ là một mô hình tuyến tính, hạn chế khả năng học các đặc trưng phức tạp của nó.

Một số hàm kích hoạt phổ biến được sử dụng trong CNNs bao gồm:

- ReLU (Rectified Linear Unit): $f(x) = \max(0, x)$ - ReLU có hiệu quả về mặt tính toán và thường giúp tăng tốc độ huấn luyện. Tuy nhiên, nó có thể gặp phải vấn đề "dying ReLU", trong đó một số nơ-ron trở nên không hoạt động và chỉ xuất ra 0 cho bất kỳ đầu vào nào.
- Sigmoid: $f(x) = \frac{1}{1+e^{-x}}$
- Tanh (hyperbolic tangent**):** $f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$



Hình 2.5 Hàm kích hoạt

2.1.3 Huấn luyện CNNs

CNNs thường được huấn luyện bằng cách sử dụng phương pháp học có giám sát với một tập dữ liệu lớn được gắn nhãn. Quá trình huấn luyện bao gồm:

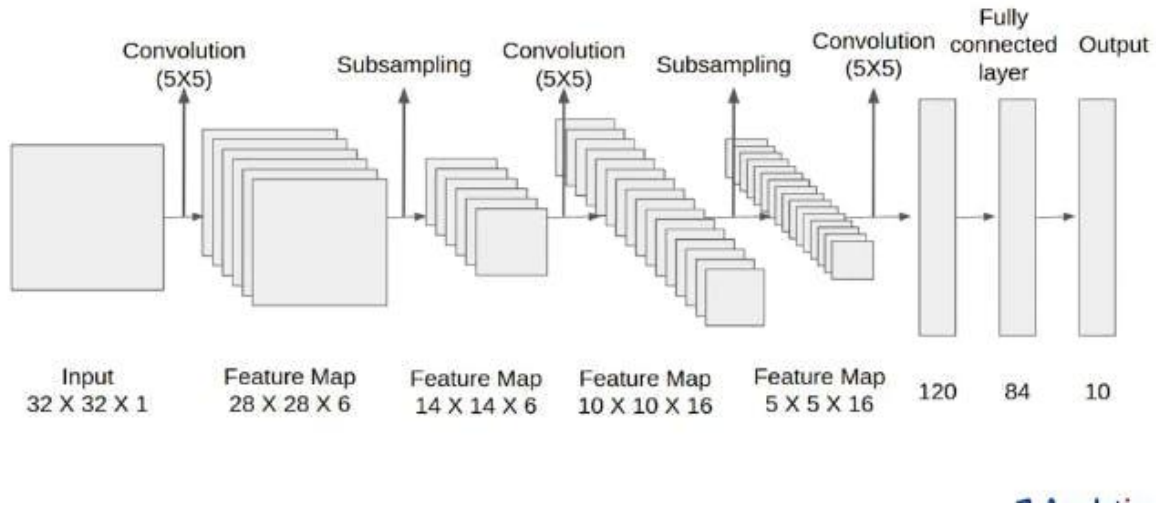
- Forward pass: Dữ liệu đầu vào được đưa qua mạng và đầu ra được tính toán.
- Hàm mất mát: Sự khác biệt giữa đầu ra dự đoán và mục tiêu thực tế được tính bằng cách sử dụng một hàm mất mát.

- Lan truyền ngược: Lỗi được lan truyền ngược trở lại mạng để tính toán độ dốc của hàm mất mát đối với trọng số và độ lệch. Thông tin này được sử dụng để điều chỉnh các tham số của mạng và cải thiện độ chính xác của nó.
- Tối ưu hóa: Một thuật toán tối ưu hóa, chẳng hạn như hạ gradient ngẫu nhiên (SGD) hoặc Adam, được sử dụng để cập nhật trọng số và độ lệch để giảm thiểu hàm mất mát. SGD cập nhật các tham số dựa trên độ dốc trung bình trên một lô các ví dụ huấn luyện, trong khi Adam điều chỉnh tốc độ học cho mỗi tham số.
- Quá trình này được lặp lại nhiều lần cho đến khi mạng đạt được hiệu suất thỏa đáng.

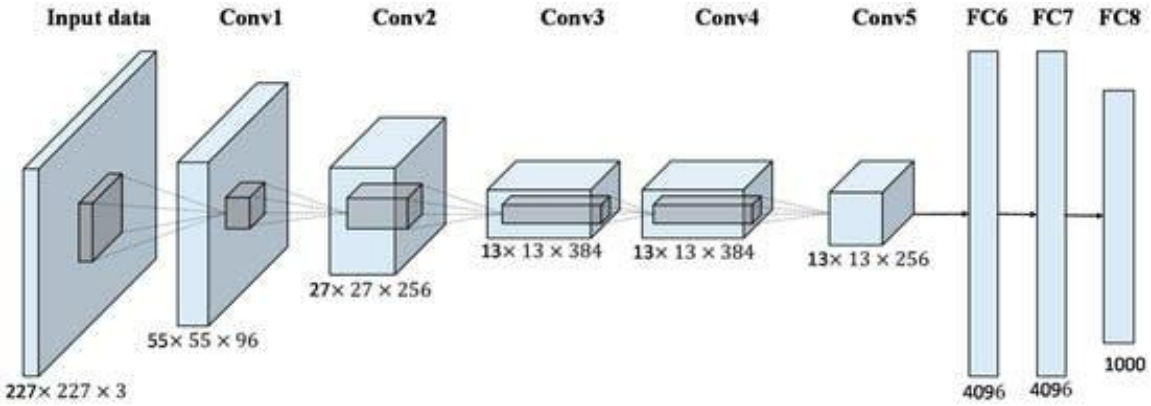
2.1.4 Kiến trúc CNN phổ biến

Kiến trúc	Năm	Đặc trưng chính	Ưu điểm/Nhược điểm
LeNet-5	1998	Các lớp Convolutional và Pooling	Một trong những CNNs sớm nhất, được sử dụng để nhận dạng chữ số viết tay. Kiến trúc đơn giản với dung lượng hạn chế.
AlexNet	2012	Mạng sâu hơn, ReLU activation, dropout	Cải thiện đáng kể độ chính xác phân loại hình ảnh. Chi phí tính toán cao.
VGGNet	2014	16-19 lớp, các bộ lọc convolutional nhỏ	Chứng minh tầm quan trọng của độ sâu. Có thể tốn kém về mặt tính toán.
GoogLeNet (InceptionNet)	2014	Inception modules	Tính toán hiệu quả và các mạng sâu hơn. Kiến trúc phức tạp.
ResNet	2015	Residual connections	Cho phép huấn luyện các mạng rất sâu. Yêu cầu bộ nhớ cao.

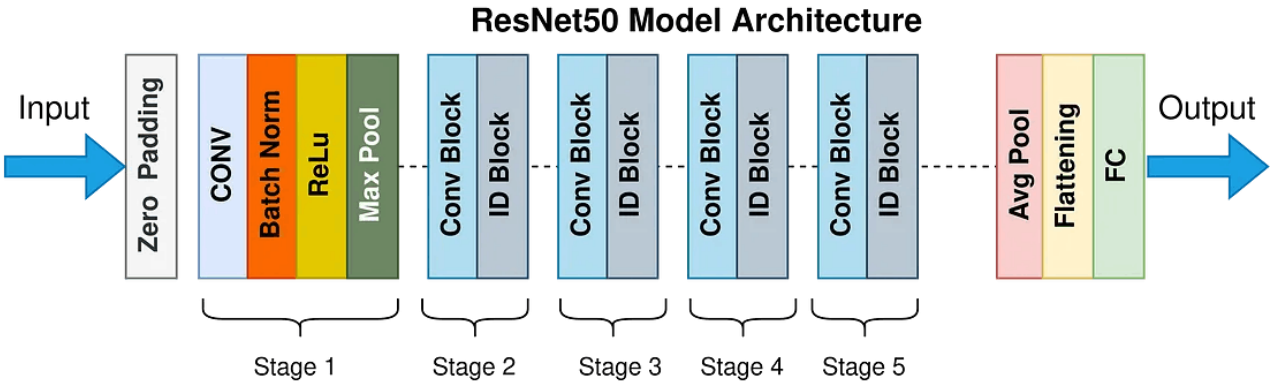
Bảng 2.1 Kiến trúc CNN phổ biến



Hình 2.6 Kiến trúc LeNet-5



Hình 2.7 Kiến trúc AlexNet



Hình 2.8 Kiến trúc ResNet50

2.1.5 Ưu điểm và nhược điểm của CNNs

❖ **Ưu điểm:**

Trích xuất đặc trưng tự động: CNNs có thể tự động học các đặc trưng liên quan từ dữ liệu thô, loại bỏ nhu cầu thiết kế đặc trưng thủ công, vốn là một nút thắt chính trong các phương pháp xử lý hình ảnh truyền thống.

Độ chính xác cao: CNNs đã đạt được hiệu suất tiên tiến trong các tác vụ nhận dạng hình ảnh và video khác nhau, vượt qua hiệu suất của con người trong một số trường hợp.

Mạnh mẽ đối với nhiễu: CNNs tương đối mạnh mẽ đối với nhiễu và các biến thể trong dữ liệu đầu vào, làm cho chúng phù hợp với các ứng dụng trong thế giới thực.

Học theo cấp bậc: CNNs học các biểu diễn đặc trưng theo cấp bậc, bắt đầu bằng các cạnh và mẫu đơn giản ở các tầng đầu và tiến tới các đặc trưng phức tạp hơn ở các tầng sâu hơn. Điều này cho phép chúng nắm bắt các chi tiết phức tạp và hiểu cấu trúc tổng thể của đầu vào.

❖ **Nhược điểm:**

Chi phí tính toán cao: Huấn luyện CNNs có thể tốn kém về mặt tính toán, yêu cầu phần cứng mạnh mẽ như GPU.

Yêu cầu dữ liệu lớn: CNNs thường cần một lượng lớn dữ liệu được gắn nhãn để huấn luyện.

Quá trình Overfitting: CNNs có thể dễ bị overfitting, đặc biệt là với dữ liệu hạn chế. Overfitting xảy ra khi mô hình học dữ liệu huấn luyện quá tốt và không thể khái quát hóa cho dữ liệu mới, chưa nhìn thấy.

Khả năng diễn giải: Có thể khó diễn giải các đặc trưng đã học và hiểu lý do tại sao CNN đưa ra các dự đoán cụ thể. Bản chất " black box" này có thể là một mối quan tâm trong các ứng dụng mà việc hiểu lý do đằng sau các quyết định là rất quan trọng.

Hạn chế trong việc xử lý các phép quay: CNNs có thể gặp khó khăn trong việc nhận dạng các đối tượng khi chúng bị xoay hoặc được trình bày theo các hướng khác

nhau. Điều này là do các bộ lọc convolutional được thiết kế để phát hiện các đặc trưng theo các hướng cụ thể.

2.1.6 So sánh với các mô hình khác

❖ **CNNs so với RNNs**

Recurrent Neural Networks (RNNs) được thiết kế cho dữ liệu tuần tự, chẳng hạn như văn bản hoặc chuỗi thời gian. Không giống như CNNs, xử lý dữ liệu với đầu vào và đầu ra có kích thước cố định, RNNs có thể xử lý độ dài đầu vào/đầu ra tùy ý. CNNs phù hợp hơn cho các tác vụ yêu cầu hiểu biết về không gian, trong khi RNNs vượt trội trong các tác vụ liên quan đến phụ thuộc thời gian. Ví dụ, CNNs có hiệu quả để nhận dạng hình ảnh, trong khi RNNs thường được sử dụng để xử lý ngôn ngữ tự nhiên và nhận dạng giọng nói.

❖ **CNNs so với Transformers**

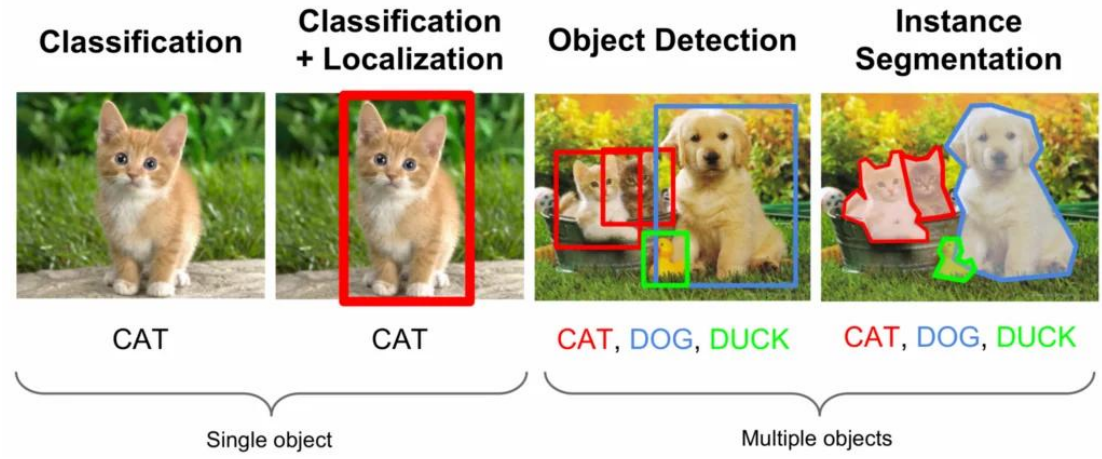
Transformers, ban đầu được phát triển để xử lý ngôn ngữ tự nhiên, gần đây đã được điều chỉnh cho các tác vụ thị giác máy tính. Trong khi CNNs dựa vào các lớp convolutional để trích xuất đặc trưng, Transformers sử dụng các cơ chế tự chú ý để nắm bắt các phụ thuộc toàn cục và mối quan hệ giữa các mảng hình ảnh. Điều này cho phép Transformers mô hình hóa các tương tác tầm xa trong hình ảnh hiệu quả hơn CNNs. Transformers đã cho thấy kết quả đầy hứa hẹn trong nhận dạng hình ảnh, nhưng chúng thường yêu cầu tập dữ liệu lớn hơn và nhiều tài nguyên tính toán hơn CNNs.

2.1.7 Các ứng dụng của CNNs

CNNs có một loạt các ứng dụng trong các lĩnh vực khác nhau:

- Nhận dạng hình ảnh: CNNs được sử dụng rộng rãi để phân loại hình ảnh, phát hiện đối tượng và phân đoạn hình ảnh. Các ứng dụng bao gồm nhận dạng khuôn mặt, phân tích hình ảnh y tế và xe tự lái.
- Phát hiện đối tượng: CNNs có thể phát hiện và định vị chính xác các đối tượng trong hình ảnh hoặc video. Điều này được sử dụng trong các ứng dụng như xe tự lái, hệ thống an ninh và robot.

- Phân đoạn hình ảnh: CNNs có thể phân đoạn hình ảnh thành các vùng khác nhau và phân loại từng pixel, cho phép các ứng dụng như phân tích hình ảnh y tế và theo dõi đối tượng.
- Xử lý hình ảnh y tế: CNNs được sử dụng cho các tác vụ như phát hiện khối u, phân loại bệnh và phân đoạn cơ quan trong hình ảnh y tế.
- Xe tự lái: CNNs đóng một vai trò quan trọng trong xe tự lái cho các tác vụ như phát hiện làn đường, nhận dạng đối tượng và lập kế hoạch đường đi.
- Xử lý ngôn ngữ tự nhiên: CNNs có thể được sử dụng để phân loại văn bản, phân tích tình cảm và dịch ngôn ngữ. Trong xử lý ngôn ngữ tự nhiên, CNNs có thể phân tích văn bản bằng cách coi nó như một chuỗi các từ và trích xuất các đặc trưng liên quan.
- Xử lý dữ liệu 3D: CNNs có thể được áp dụng cho dữ liệu 3D, chẳng hạn như video, bằng cách mở rộng các phép toán convolutional thành ba chiều. Điều này cho phép các ứng dụng như nhận dạng hành động và phân tích video.



Hình 2.9 Ứng dụng của CNNs

2.1.8 Xu hướng và thách thức trong phát triển CNN

❖ **Xu Hướng:**

Kiến trúc CNN gọn nhẹ: Tập trung vào việc phát triển CNN với ít tham số hơn và chi phí tính toán thấp hơn để triển khai trên các thiết bị có nguồn lực hạn chế. Xu

hướng này được thúc đẩy bởi nhu cầu ngày càng tăng đối với việc triển khai CNN trên các thiết bị di động và hệ thống nhúng.

Few-shot learning: Huấn luyện CNN để nhận dạng các đối tượng với các ví dụ huấn luyện hạn chế. Điều này rất quan trọng đối với các ứng dụng mà việc thu thập các tập dữ liệu lớn được gắn nhãn là một thách thức.

Transfer learning: Tận dụng các mô hình CNN được huấn luyện trước cho các tác vụ mới và tinh chỉnh chúng cho các ứng dụng cụ thể. Transfer learning có thể giảm đáng kể thời gian huấn luyện và yêu cầu dữ liệu, giúp dễ dàng điều chỉnh CNN cho các vấn đề mới. Phương pháp này có lợi ích như triển khai nhanh hơn và giảm thời gian huấn luyện.

❖ **Thách thức:**

Khan hiếm dữ liệu: Việc thu thập đủ dữ liệu được gắn nhãn để huấn luyện CNN có thể là một thách thức, đặc biệt là trong các lĩnh vực chuyên biệt. Điều này có thể hạn chế hiệu suất và khả năng khái quát hóa của các mô hình CNN.

Quá trình Overfitting: Ngăn chặn overfitting, đặc biệt là với dữ liệu hạn chế, vẫn là một thách thức. Các kỹ thuật như data augmentation, trong đó dữ liệu huấn luyện được mở rộng một cách giả tạo bằng cách áp dụng các phép biến đổi, có thể giúp giải quyết vấn đề này.

Khả năng diễn giải: Hiểu quá trình ra quyết định của CNN và giải thích các dự đoán của chúng là rất quan trọng để xây dựng niềm tin và đảm bảo hiệu suất đáng tin cậy. Nghiên cứu về AI có thể giải thích nhằm mục đích làm cho CNN minh bạch và dễ hiểu hơn.

2.2 R-CNN và Mask R-CNN

Phát hiện đối tượng và phân đoạn là các nhiệm vụ cơ bản trong thị giác máy tính, cho phép máy móc nhận thức và diễn giải thông tin thị giác theo cách tương tự như con người. R-CNN và các phiên bản phát triển của nó, bao gồm Mask R-CNN, đã cải tiến đáng kể các lĩnh vực này, đạt được độ chính xác và hiệu quả đáng kể. Phần này cung

cấp một khám phá toàn diện về các mô hình này, theo dõi sự phát triển của chúng, đi sâu vào kiến trúc và các thành phần của chúng và kiểm tra các ứng dụng của chúng.

2.2.1 Sự phát triển của các Mô hình R-CNN

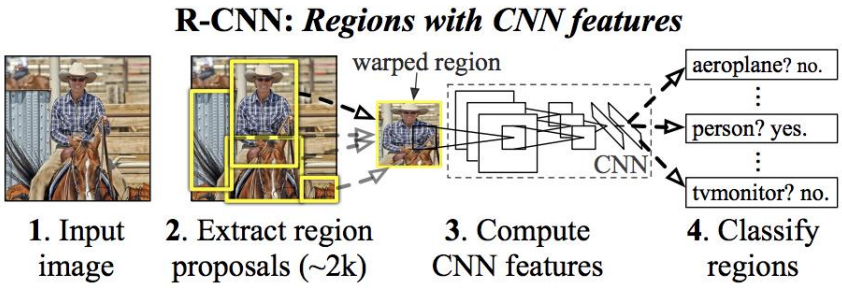
Họ R-CNN đại diện cho một bước nhảy vọt đáng kể trong phát hiện và phân đoạn đối tượng. Bên dưới là sự phát triển của chúng:

2.2.1.1 R-CNN

R-CNN ban đầu, được giới thiệu vào năm 2014, đã đánh dấu một thời điểm quan trọng trong phát hiện đối tượng bằng cách tích hợp các đề xuất vùng với mạng nơ-ron tích chập (CNN). Khái niệm cốt lõi liên quan đến việc xác định các vùng đối tượng tiềm năng trong một hình ảnh bằng cách sử dụng tìm kiếm chọn lọc và sau đó sử dụng CNN để trích xuất các đặc trưng từ mỗi vùng để phân loại.

Các hạn chế của R-CNN:

- Suy luận chậm: Xử lý từng đề xuất vùng một cách độc lập dẫn đến tốc độ suy luận chậm, cản trở khả năng ứng dụng của nó cho các ứng dụng thời gian thực.
- Các đề xuất vùng chồng chéo: Tính toán dư thừa phát sinh từ các đề xuất chồng chéo có thể ảnh hưởng tiêu cực đến hiệu suất phát hiện.
- Huấn luyện nhiều giai đoạn: Quá trình huấn luyện liên quan đến các bước riêng biệt để tinh chỉnh CNN, huấn luyện bộ phân loại SVM và hồi quy hộp giới hạn, tăng thêm độ phức tạp và thời gian tiêu thụ.



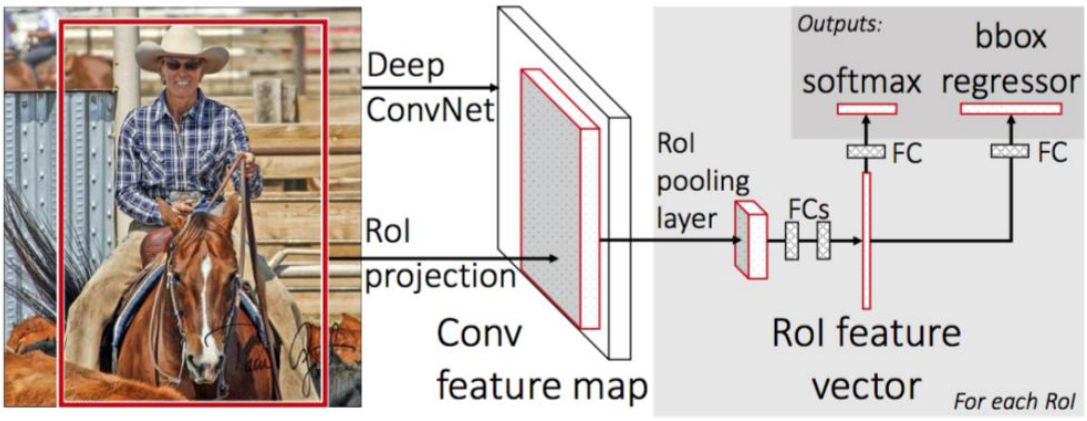
Hình 2.10 Kiến trúc R-CNN

2.2.1.2 Fast R-CNN

Fast R-CNN đã giải quyết các hạn chế về tốc độ của R-CNN bằng cách cho phép tính toán được chia sẻ trên các đề xuất vùng. Thay vì đưa từng đề xuất riêng lẻ vào CNN, Fast R-CNN xử lý toàn bộ hình ảnh bằng CNN một lần. Sau đó, một lớp **Region of Interest (RoI) pooling** (một tầng trích xuất bản đồ đặc trưng có kích thước cố định từ vùng quan tâm trong bản đồ đặc trưng bằng cách chia vùng thành các bin và gộp tối đa trong mỗi bin) trích xuất các đặc trưng cho mỗi đề xuất từ feature maps được chia sẻ. Sự đổi mới này đã tăng tốc đáng kể tốc độ huấn luyện và thử nghiệm đồng thời nâng cao độ chính xác. Tuy nhiên, nó vẫn dựa vào các phương pháp đề xuất vùng bên ngoài, chẳng hạn như Selective Search (một phương pháp thị giác máy tính truyền thống để tạo ra các đề xuất vùng bằng cách nhóm các pixel tương tự dựa trên màu sắc, kết cấu và kích thước), vẫn là một nút thắt cổ chai trong quy trình phát hiện.

Các cải tiến chính:

- Tính toán được chia sẻ: Xử lý toàn bộ hình ảnh bằng CNN chỉ một lần đã giảm đáng kể thời gian tính toán.
- Huấn luyện một giai đoạn: Hàm mất mát đa nhiệm vụ tạo điều kiện huấn luyện đồng thời để phân loại và hồi quy hộp giới hạn, hợp lý hóa quá trình huấn luyện.
- Độ chính xác được cải thiện: Đạt được Độ chính xác trung bình (mAP) cao hơn so với R-CNN.



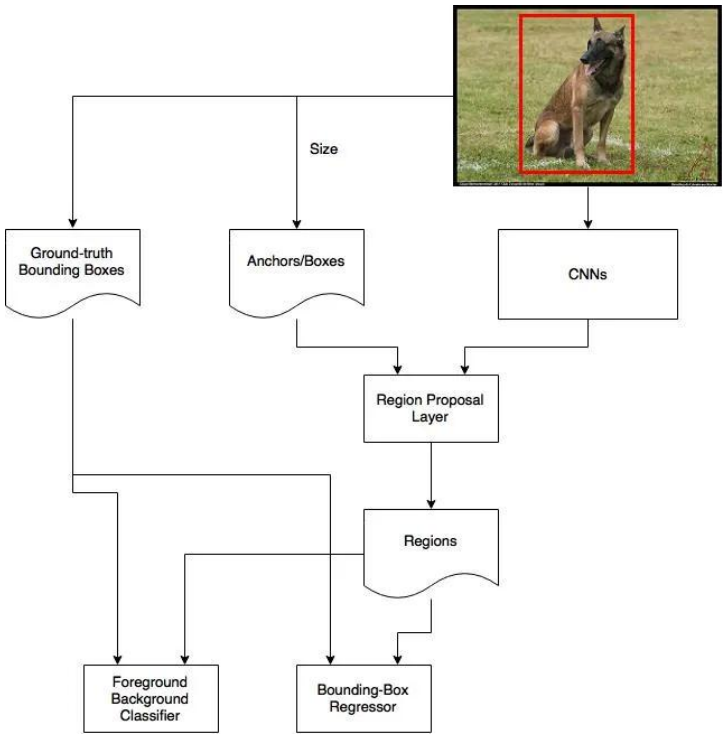
Hình 2.11 Kiến trúc Fast-R-CNN

2.2.1.3 Faster R-CNN

Faster R-CNN đã tăng cường hơn nữa hiệu quả bằng cách giới thiệu **Region Proposal Network (RPN)** để tạo ra các đề xuất vùng. RPN chia sẻ các đặc trưng tích chập với mạng phát hiện, loại bỏ sự phụ thuộc vào các phương pháp đề xuất vùng bên ngoài như tìm kiếm chọn lọc. Điều này dẫn đến khả năng phát hiện đối tượng gần thời gian thực với độ chính xác được nâng cao. Việc tích hợp RPN đã làm cho việc tạo đề xuất vùng nhanh hơn đáng kể so với Tìm kiếm Chọn lọc, một yếu tố quan trọng trong việc đạt được hiệu suất gần thời gian thực.

Các đột phá chính:

- Region Proposal Network (RPN): Một mạng tích chập đầy đủ trượt một cửa sổ nhỏ trên bản đồ đặc trưng và dự đoán điểm đối tượng và bù hộp giới hạn cho các hộp neo ở mỗi vị trí. Tích hợp tạo đề xuất vùng trong mạng, tăng tốc đáng kể quá trình.
- Huấn luyện đầu cuối: Cho phép huấn luyện toàn bộ quy trình phát hiện đối tượng trong một giai đoạn, cải thiện hơn nữa hiệu quả và độ chính xác.



Hình 2.12 Region Proposal Network

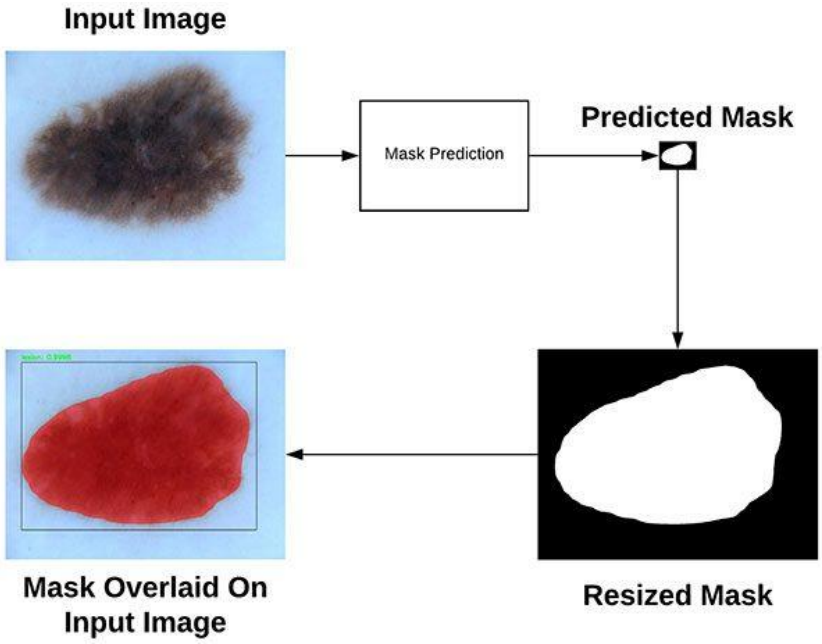
2.2.1.4 Mask R-CNN

Mask R-CNN được mở rộng dựa trên Faster R-CNN để thực hiện phân đoạn cá thể, một nhiệm vụ không chỉ liên quan đến việc phát hiện đối tượng mà còn tạo ra mặt nạ phân đoạn chất lượng cao cho mỗi cá thể. Điều này đạt được bằng cách thêm một nhánh để dự đoán mặt nạ đối tượng song song với nhánh hiện có để nhận dạng hộp giới hạn. Không giống như các mô hình trước đó chủ yếu tập trung vào các hộp giới hạn, Mask R-CNN cung cấp mặt nạ theo pixel, cho phép phân đoạn đối tượng chính xác hơn.

Các đổi mới chính:

- Nhánh dự đoán mặt nạ: Đã thêm một nhánh song song để dự đoán mặt nạ theo pixel cho mỗi cá thể đối tượng.

- RoIAlign: Một phiên bản cải tiến của RoI pooling sử dụng nội suy song tuyến để tránh lỗi lượng tử hóa và bảo toàn thông tin không gian, dẫn đến dự đoán mặt nạ chính xác hơn.
- Tách rời dự đoán mặt nạ và lớp: Dự đoán mặt nạ nhị phân cho mỗi lớp một cách độc lập, cải thiện hiệu suất phân đoạn.



Hình 2.13 Mask R-CNN

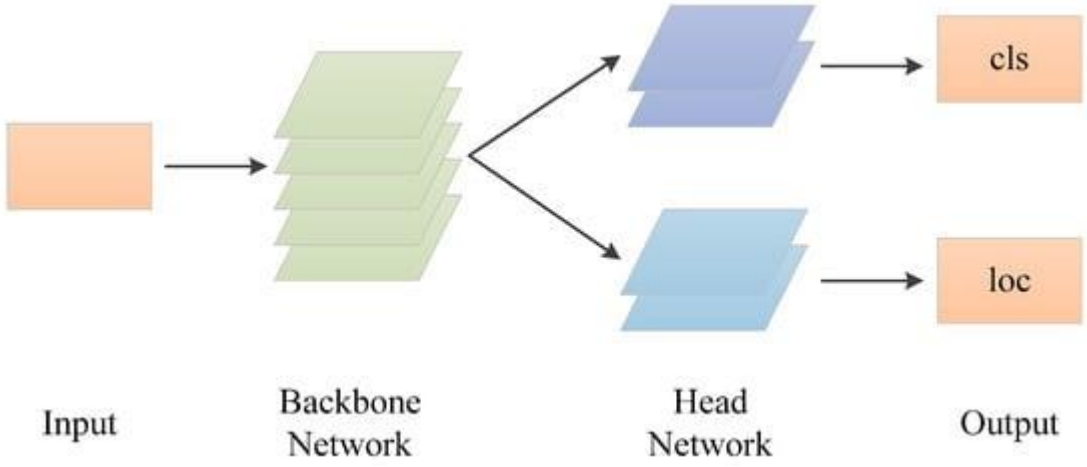
2.2.2 Kiến trúc và các thành phần

2.2.2.1 Kiến trúc mô hình phát hiện đối tượng

Các mô hình phát hiện đối tượng thường bao gồm bốn phần chính:

- Đầu vào: Đầu vào là một hình ảnh chứa các đối tượng cần được phát hiện.
- Backbone Network: Đây là một mạng nơ-ron tích chập (CNN) trích xuất các đặc trưng từ hình ảnh đầu vào. Các mạng xương sống phổ biến bao gồm ResNet, VGG và Inception.

- Head Network: Phần này của mạng lấy các đặc trưng từ mạng xương sống và thực hiện các nhiệm vụ phát hiện đối tượng, chẳng hạn như phân loại và hồi quy hộp giới hạn.
- Đầu ra: Đầu ra bao gồm các hộp giới hạn được dự đoán và nhãn lớp cho các đối tượng được phát hiện.



Hình 2.14 Kiến trúc mô hình phát hiện đối tượng

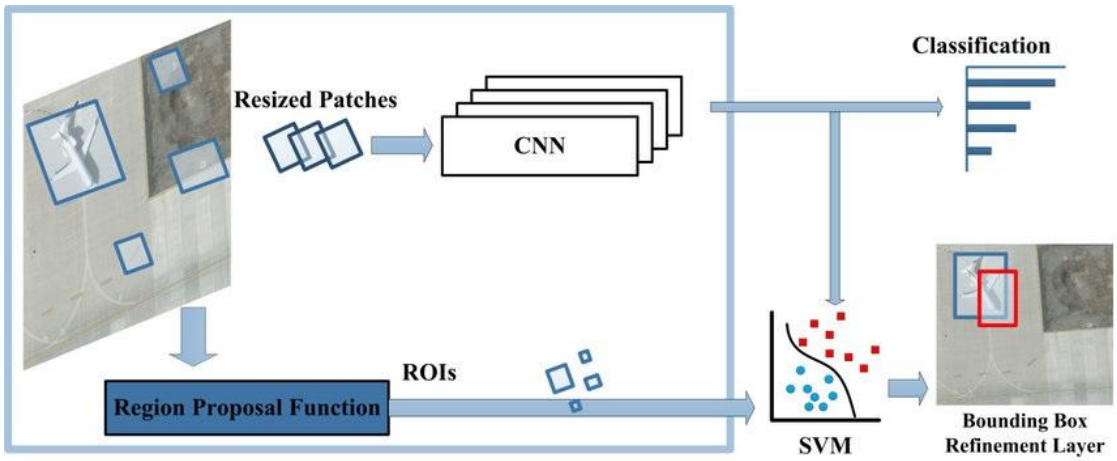
2.2.2.2 R-CNN

Đề xuất vùng: R-CNN sử dụng tìm kiếm chọn lọc để tạo ra một tập hợp các đề xuất vùng, là các vị trí đối tượng tiềm năng trong hình ảnh. Số lượng đề xuất vùng có thể thay đổi tùy thuộc vào biến thể R-CNN cụ thể. Ví dụ: R-CNN ban đầu đã tạo ra khoảng 2000 đề xuất trên mỗi hình ảnh, trong khi Fast R-CNN lấy mẫu nhiều ROI từ cùng một hình ảnh.

Trích xuất đặc trưng: Mỗi đề xuất vùng bị biến dạng thành một kích thước cố định và sau đó được chuyển qua CNN (ví dụ: AlexNet) để trích xuất các đặc trưng.

Phân loại: Một bộ phân loại máy vectơ hỗ trợ (SVM) được huấn luyện cho mỗi lớp đối tượng để phân loại các đặc trưng được trích xuất.

Bounding Box Regression: Mô hình hồi quy tuyến tính tinh chỉnh các hộp giới hạn của các vùng được đề xuất.



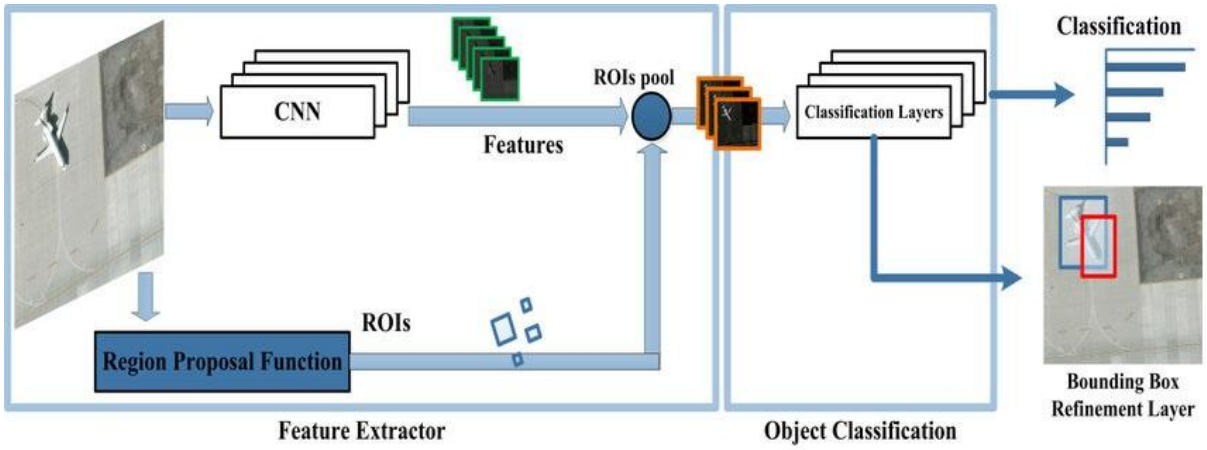
Hình 2.15 Kiến trúc R-CNN

2.2.2.3 Fast R-CNN

Trích xuất đặc trưng: Toàn bộ hình ảnh được xử lý bởi CNN để tạo ra bản đồ đặc trưng.

RoI Pooling: Trích xuất bản đồ đặc trưng có kích thước cố định từ vùng quan tâm (RoI) trong bản đồ đặc trưng.

Phân loại và bounding box regression: Các tầng được kết nối đầy đủ được sử dụng cho cả phân loại và hồi quy hộp giới hạn, với hàm mất mát đa nhiệm vụ tối ưu hóa cả hai đồng thời.



Hình 2.16 Kiến trúc Fast R-CNN

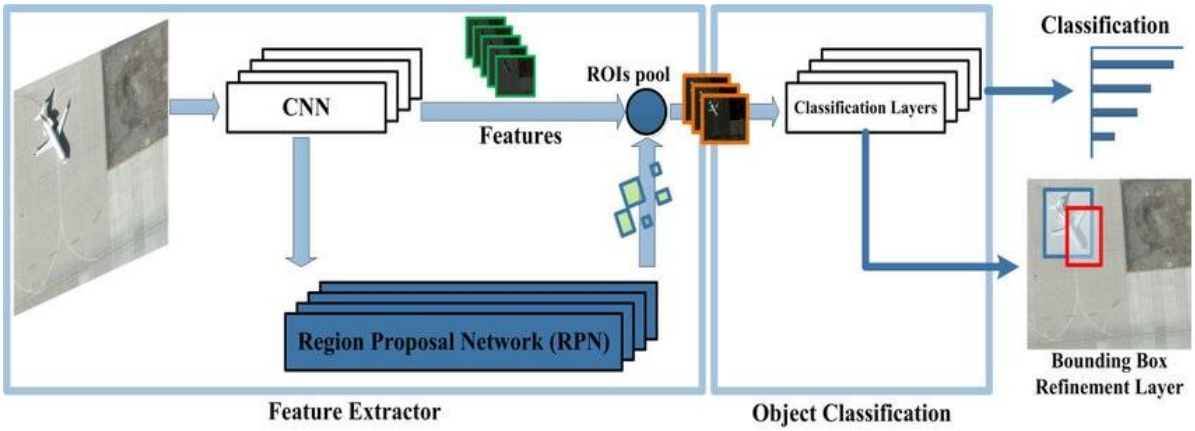
2.2.2.4 Faster R-CNN

Trích xuất đặc trưng: Tương tự như Fast R-CNN, toàn bộ hình ảnh được xử lý bởi CNN để tạo ra bản đồ đặc trưng.

Region Proposal Network (RPN): Tạo các đề xuất vùng trực tiếp từ bản đồ đặc trưng.

RoI Pooling: Trích xuất bản đồ đặc trưng có kích thước cố định từ RoI.

Phân loại và bounding box regression: Tương tự như Fast R-CNN, các tầng được kết nối đầy đủ được sử dụng cho phân loại và hồi quy hộp giới hạn.



Hình 2.17 Kiến trúc Faster R-CNN

2.2.2.5 Mask R-CNN

Trích xuất đặc trưng: Toàn bộ hình ảnh được xử lý bởi CNN để tạo ra bản đồ đặc trưng.

Region Proposal Network (RPN): Region Proposal Network.

RoIAlign: Phiên bản cải tiến của RoI Pooling, sử dụng nội suy song tuyến để tránh lỗi lượng tử hóa.

Phân loại và bounding box regression: Các tầng được kết nối đầy đủ được sử dụng cho phân loại và hồi quy hộp giới hạn.

Mask Prediction: Một nhánh riêng biệt với các tầng tích chập dự đoán mặt nạ nhị phân cho mỗi đề xuất vùng.

2.2.2.6 Các hàm mất mát

Các mô hình R-CNN sử dụng kết hợp các hàm mất mát để tối ưu hóa các nhiệm vụ khác nhau:

- **Classification Loss:** Thông thường, mất mát chéo được sử dụng để đo lường lỗi giữa xác suất lớp được dự đoán và nhãn sự thật mặt đất. Mất mát chéo xử phạt mô hình nhiều hơn đối với các dự đoán không chính xác với độ tin cậy cao.
- **Bounding Box Regression Loss:** Smooth L1 loss thường được sử dụng để đo lường sự khác biệt giữa các predicted bounding box offsets và các round truth offsets. Smooth L1 loss ít nhạy cảm với các ngoại lệ hơn so với L2 loss.
- **Mask Prediction Loss:** Mất mát chéo nhị phân được sử dụng để đo lường lỗi giữa các pixel mặt nạ được dự đoán và các pixel mặt nạ sự thật mặt đất. Hàm mất mát này phù hợp cho các nhiệm vụ phân loại theo pixel.

2.2.3 So sánh các mô hình

Mô hình	**Kiến trúc**	**Tốc độ**	**Độ chính xác**	**Điểm mạnh**	**Điểm yếu**
R-CNN	Region proposals + CNN + SVM	Chậm	Trung bình	Tiên phong trong phát hiện đối tượng dựa trên vùng	Suy luận chậm, huấn luyện nhiều giai đoạn
Fast R-CNN	Shared computation, RoI pooling	Nhanh hơn	Cải thiện	Huấn luyện và suy luận nhanh hơn	Vẫn dựa vào đề xuất vùng bên ngoài
Faster R-CNN	RPN, RoI pooling	Gần real-time	Cao	Huấn luyện đầu cuối, hiệu quả	
Mask R-CNN	RPN, RoIAlign, mask branch	Nhanh	SOTA	Phân đoạn cá thể, độ chính xác cao	

Bảng 2.2 So sánh các mô hình R-CNN

Mô hình	Kiến trúc	Tốc độ	Độ chính xác	Điểm mạnh	Điểm yếu
YOLO	Single stage	Rất nhanh	Trung bình	Hiệu suất thời gian thực	Độ chính xác thấp hơn so với bộ phát hiện hai giai đoạn
SSD	Single-stage, multi-scale	Nhanh	Tốt	Cân bằng tốt giữa tốc độ và độ chính xác, phát hiện các đối tượng ở nhiều tỷ lệ khác nhau	
RetinaNet	Single-stage, focal loss	Nhanh	Cao	Độ chính xác cao, giải quyết sự mất cân bằng lớp	

Bảng 2.3 So sánh với các phương pháp phát hiện đối tượng khác

Các bộ phát hiện một giai đoạn (Single stage) như YOLO và SSD ưu tiên tốc độ suy luận, trong khi các bộ phát hiện hai giai đoạn như Faster R-CNN ưu tiên độ chính xác phát hiện. Điều này thể hiện sự đánh đổi giữa tốc độ và độ chính xác khi lựa chọn mô hình phát hiện đối tượng. YOLO, được biết đến với tốc độ của nó, có thể có độ chính xác thấp hơn so với các bộ phát hiện hai giai đoạn.

2.2.4 Các ứng dụng và triển khai

R-CNN và Mask R-CNN đã tìm thấy các ứng dụng đa dạng trên nhiều lĩnh vực khác nhau:

- Phát hiện đối tượng: Xác định và bản địa hóa các đối tượng trong hình ảnh, chẳng hạn như ô tô, người đi bộ và đèn giao thông trong lái xe tự động.
- Phân đoạn cá thể: Phân đoạn các đối tượng riêng lẻ trong hình ảnh, chẳng hạn như xác định và tách biệt những người khác nhau trong đám đông.
- Phát hiện điểm chính: Xác định vị trí các điểm chính trên đối tượng, chẳng hạn như khớp người để ước tính tư thế.

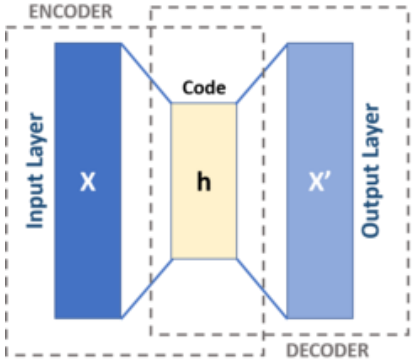
- Hình ảnh y tế: Phát hiện và phân đoạn khối u, tổn thương và các bất thường khác trong hình ảnh y tế.
- Robot: Cho phép robot nhận thức và tương tác với các đối tượng trong môi trường của chúng.

2.3 AutoEncoder

Autoencoders là một lớp mạng nơ-ron đã thu hút sự chú ý đáng kể trong lĩnh vực học không giám sát. Chúng là một loại hình đặc biệt của mạng nơ-ron chuyển tiếp không giám sát (unsupervised feedforward), có nghĩa là chúng học cách tái tạo dữ liệu đầu vào của chính chúng mà không cần dựa vào các nhãn rõ ràng. Nhiệm vụ tưởng chừng đơn giản này cho phép chúng học các biểu diễn hiệu quả của dữ liệu, sau đó có thể được sử dụng cho các mục đích khác nhau như giảm chiều dữ liệu, phát hiện bất thường và tạo dữ liệu. Phần này cung cấp một cái nhìn tổng quan toàn diện về autoencoders, bao gồm các loại khác nhau, quy trình huấn luyện, ứng dụng và so sánh với các kỹ thuật khác.

2.3.1 Giới thiệu

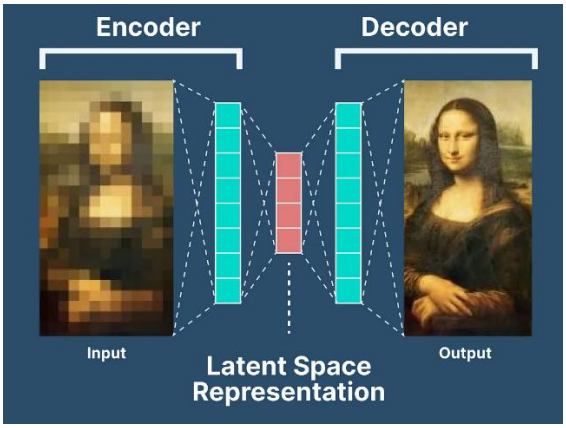
Autoencoder là một kiến trúc mạng nơ-ron được thiết kế để học một biểu diễn nén của dữ liệu đầu vào. Nó bao gồm ba tầng chính: bộ mã hóa (encoder), mã (code) và bộ giải mã (decoder). Bộ mã hóa ánh xạ dữ liệu đầu vào tới một không gian tiềm ẩn có chiều thấp hơn, còn được gọi là tầng nút cổ chai (bottleneck layer), trong khi bộ giải mã tái tạo lại đầu vào ban đầu từ biểu diễn nén này. Quá trình này khuyến khích autoencoder xác định và bảo toàn các đặc trưng quan trọng nhất của dữ liệu, cho phép nó nắm bắt hiệu quả cấu trúc cơ bản.



Hình 2.18 Autoencoder

Autoencoders được huấn luyện bằng cách sử dụng học không giám sát, có nghĩa là chúng không yêu cầu dữ liệu được gắn nhãn. Trong quá trình suy luận, chúng hoạt động như các mô hình không giám sát, nhưng chúng được huấn luyện như các mô hình giám sát, làm cho chúng tự giám sát. Quá trình huấn luyện bao gồm việc giảm thiểu sự khác biệt giữa dữ liệu đầu vào và dữ liệu tái tạo của nó. Điều này thường đạt được bằng cách sử dụng hàm mất mát chẳng hạn như sai số bình phương trung bình (MSE).

Khái niệm " latent space " là rất quan trọng để hiểu autoencoders. Nó đề cập đến tập hợp các biến tiềm ẩn, là các biến ẩn hoặc ngẫu nhiên không thể quan sát trực tiếp nhưng về cơ bản thông báo cách dữ liệu được phân phối. Bằng cách học một biểu diễn nén trong không gian tiềm ẩn, autoencoders có thể nắm bắt hiệu quả bản chất của dữ liệu đầu vào.



Hình 2.19 Latent space

2.3.2 Các loại Autoencoders

Autoencoders có nhiều dạng khác nhau, mỗi dạng có các đặc điểm và ứng dụng riêng biệt. Các loại khác nhau này có thể được tóm tắt hiệu quả trong bảng sau:

Loại Autoencoder	**Mô tả**	**Kiến trúc**	**Hàm mất mát**	**Ứng dụng**
Undercomplete Autoencoder	Giảm chiều dữ liệu đầu vào bằng cách có ít nơ-ron hơn trong các tầng ẩn so với tầng đầu vào.	Đối xứng với ít nơ-ron hơn trong các tầng ẩn	Mean Squared Error (MSE)	Giảm chiều, trích xuất đặc trưng.
Overcomplete Autoencoder	Có nhiều nơ-ron hơn trong các tầng ẩn so với tầng đầu vào, cho phép biểu diễn phức tạp hơn.	Đối xứng với nhiều nơ-ron hơn trong các tầng ẩn.	MSE with regularization	Loại bỏ nhiễu, trích xuất đặc trưng.
Sparse Autoencoder	Sử dụng hình phạt thưa thớt để khuyến khích mạng chỉ kích hoạt một số lượng nhỏ nơ-ron cho mỗi đầu vào.	Tương tự như undercomplete hoặc overcomplete, nhưng có hình phạt thưa thớt trong hàm mất mát.	MSE with sparsity penalty	Trích xuất đặc trưng, phát hiện bất thường.
Denoising Autoencoder	Được huấn luyện để tái tạo đầu vào ban đầu từ một phiên bản bị hỏng, làm cho nó mạnh mẽ đối với nhiễu.	Tương tự như các loại khác, nhưng được huấn luyện trên dữ liệu đầu vào nhiễu.	MSE	Loại bỏ nhiễu hình ảnh, học đặc trưng.
Variational Autoencoder (VAE)	Học phân phối xác suất trên không gian tiềm ẩn, cho phép nó tạo ra các mẫu dữ liệu mới.	Bộ mã hóa xuất ra các tham số của một phân phối, bộ giải mã lấy mẫu từ phân phối này.	KL divergence + reconstruction loss	Tạo hình ảnh mới, tăng cường dữ liệu.
Contractive Autoencoder (CAE)	Thêm hình phạt co rút vào hàm mất mát để học các biểu diễn mạnh mẽ đối với những thay đổi nhỏ trong đầu vào.	Tương tự như các loại khác, nhưng có hình phạt co rút trong hàm mất mát.	MSE with contractive penalty	Học đặc trưng, biểu diễn mạnh mẽ.

Hình 2.20 Các loại Autoencoders

Undercomplete Autoencoders: Các autoencoders này buộc mạng phải học một biểu diễn nén của dữ liệu bằng cách có số lượng nơ-ron nhỏ hơn trong các tầng ẩn so với tầng đầu vào. Nút cổ chai này khuyến khích autoencoder nắm bắt các đặc trưng nổi trội nhất của dữ liệu đầu vào. Một undercomplete autoencoder lý tưởng có thể tái tạo hoàn hảo đầu vào khi mọi mã có thể có trong không gian mã được sử dụng, làm nổi bật tiềm năng lý thuyết của chúng đối với nén không mất mát.

Overcomplete Autoencoders: Trái ngược với undercomplete autoencoders, overcomplete autoencoders có nhiều nơ-ron hơn trong các tầng ẩn so với tầng đầu vào. Điều này cho phép mạng học các biểu diễn phức tạp hơn và có khả năng phong phú hơn của dữ liệu. Tuy nhiên, khả năng tăng lên này cũng khiến chúng dễ bị overfitting, trong đó mạng chỉ đơn giản ghi nhớ dữ liệu đầu vào thay vì học các đặc trưng có ý nghĩa. Để ngăn chặn điều này, cần phải có các kỹ thuật chính quy hóa cẩn thận.

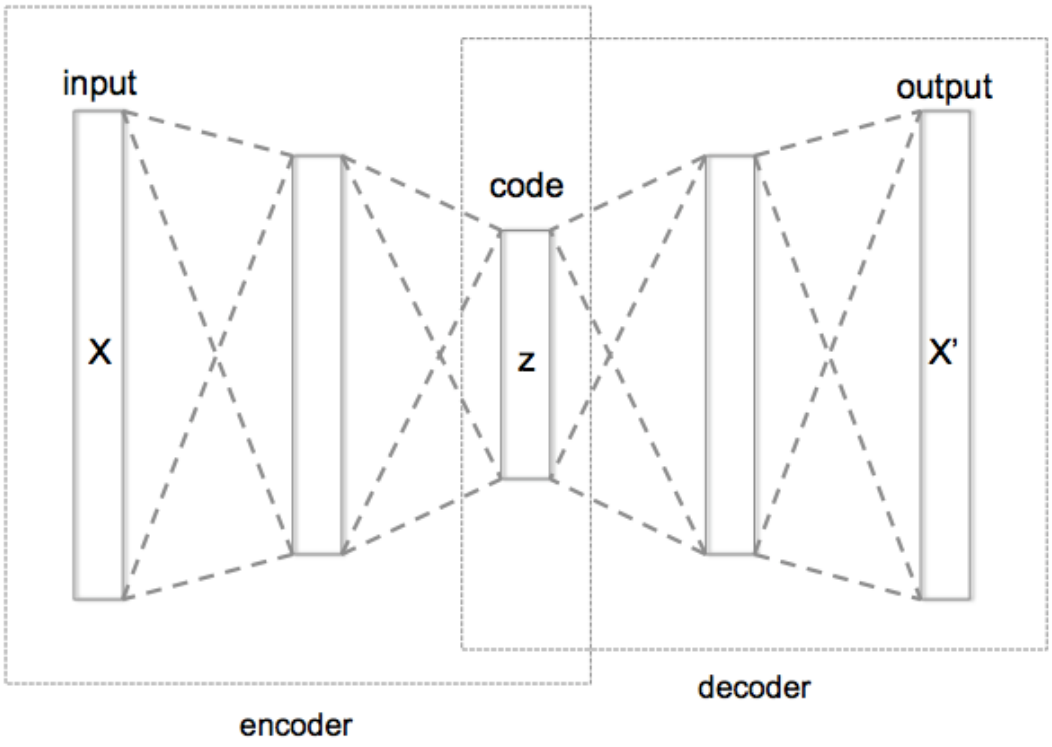
Sparse Autoencoders: Các autoencoders này đưa ra hình phạt thưa thớt trong hàm mất mát của chúng để khuyến khích mạng chỉ kích hoạt một số lượng nhỏ nơ-ron cho mỗi đầu vào. Điều này dẫn đến các biểu diễn thưa thớt, trong đó chỉ có một vài nơ-ron hoạt động tại bất kỳ thời điểm nào. Sự thưa thớt này có thể có lợi cho việc trích xuất đặc trưng, vì nó buộc mạng phải tập trung vào các đặc trưng phù hợp nhất và để phát hiện bất thường, vì các bất thường thường kích hoạt các tập hợp nơ-ron khác nhau so với dữ liệu bình thường.

Denoising Autoencoders: Các autoencoders này được huấn luyện để tái tạo đầu vào ban đầu từ một phiên bản bị hỏng, trong đó nhiễu đã được thêm vào dữ liệu đầu vào. Điều này buộc mạng phải học các đặc trưng mạnh mẽ bất biến với nhiễu. Loại autoencoder này có nguy cơ học hàm nhận dạng thấp hơn so với autoencoders tiêu chuẩn vì việc làm hỏng đầu vào ngăn chặn việc ghi nhớ đơn giản.

Variational Autoencoders (VAEs): VAEs giới thiệu yếu tố xác suất vào kiến trúc autoencoder. Thay vì chỉ mã hóa đầu vào thành một mã cố định, chúng học một phân phối trên không gian tiềm ẩn. Điều này có nghĩa là bộ mã hóa xuất ra các tham số

của phân phối xác suất, chẳng hạn như giá trị trung bình và phương sai của phân phối Gaussian, và bộ giải mã lấy mẫu từ phân phối này để tạo ra đầu ra. Cách tiếp cận xác suất này cho phép VAEs tạo ra các mẫu dữ liệu mới tương tự như dữ liệu huấn luyện.

Contractive Autoencoders (CAEs): CAEs thêm hình phạt co rút vào hàm mất mát, khuyến khích mạng học các biểu diễn mạnh mẽ đối với những thay đổi nhỏ trong đầu vào. Điều này có nghĩa là các đặc trưng đã học ít nhạy cảm hơn với các biến thể hoặc nhiễu trong dữ liệu đầu vào. Điều này có thể hữu ích để học các đặc trưng bất biến với các phép biến đổi chẳng hạn như dịch chuyển hoặc xoay.

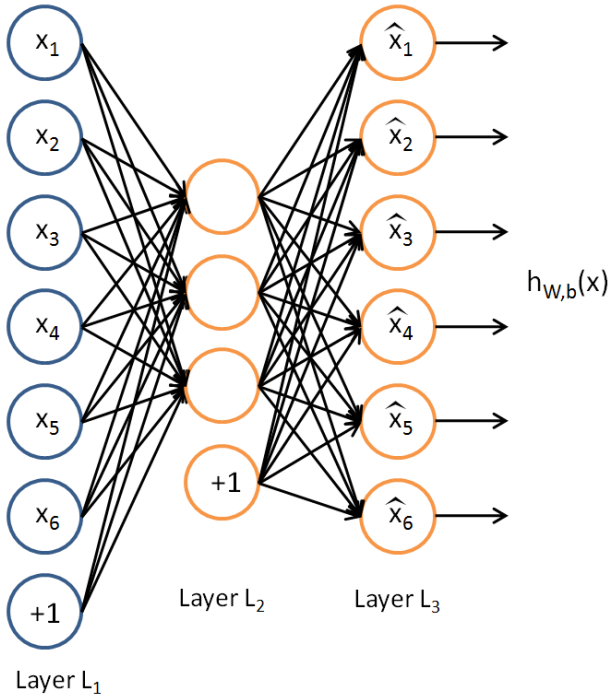


Hình 2.21 Kiến trúc undercomplete autoencoder

2.3.3 Quy trình huấn luyện

Autoencoders thường được huấn luyện bằng cách sử dụng lan truyền ngược, một thuật toán phổ biến để huấn luyện mạng nơ-ron. Điều này bao gồm việc tính toán độ dốc của hàm mất mát đối với các tham số mạng và cập nhật các tham số theo hướng

giảm thiểu mất mát. Trong quá trình lan truyền ngược, autoencoders sử dụng dữ liệu đầu vào của chúng làm giá trị mục tiêu, làm nổi bật bản chất tự giám sát của chúng.



Hình 2.22 Autoencoder

Dưới đây là giải thích từng bước đơn giản về thuật toán lan truyền ngược trong bối cảnh của autoencoders:

- **Chuyển tiếp**: Dữ liệu đầu vào được đưa qua mạng bộ mã hóa và bộ giải mã để tạo ra đầu ra được tái tạo.
- **Tính toán mất mát**: Sự khác biệt giữa đầu vào ban đầu và đầu ra được tái tạo được tính toán bằng cách sử dụng hàm mất mát, chẳng hạn như MSE.
- **Lan truyền ngược**: Độ dốc của hàm mất mát đối với các tham số mạng (trọng số và độ lệch) được tính toán. Độ dốc này cho biết hướng tăng dốc nhất của hàm mất mát.
- **Cập nhật trọng số**: Các tham số mạng được cập nhật theo hướng ngược lại với độ dốc để giảm thiểu mất mát. Việc cập nhật này thường được thực hiện

bằng cách sử dụng thuật toán tối ưu hóa chẳng hạn như hạ gradient ngẫu nhiên (SGD) hoặc Adam.

Công thức cập nhật trọng số trong lan truyền ngược thường là:

$$w_{new} = w_{old} - \eta \nabla L(w_{old})$$

trong đó:

w_{new}: là trọng số được cập nhật

w_{old}: là trọng số cũ

η: là tốc độ học, kiểm soát kích thước bước của cập nhật

$\nabla L(w_{old})$: là độ dốc của hàm mất mát đối với trọng số cũ

Điều quan trọng cần lưu ý là lan truyền ngược trong các autoencoders nơ-ron vật lý, chẳng hạn như các autoencoders được triển khai trong chip thần kinh hoặc có khả năng trong não sinh học, phải đối mặt với những thách thức do tính chất phi cục bộ của nó. Lan truyền ngược yêu cầu thông tin được truyền ngược trở lại mạng, điều này có thể khó đạt được trong các hệ thống vật lý với các ràng buộc kết nối cục bộ.

❖ **Regularization Techniques**

Regularization techniques đóng một vai trò quan trọng trong việc ngăn chặn overfitting trong quá trình huấn luyện autoencoders, đặc biệt là khi xử lý các kiến trúc overcomplete hoặc tập dữ liệu phức tạp. Một số regularization techniques phổ biến bao gồm:

- **Suy giảm trọng số**: Điều này bao gồm việc thêm một thuật ngữ phạt vào hàm mất mát không khuyến khích trọng số lớn. Điều này giúp ngăn mạng ghi nhớ dữ liệu huấn luyện và khuyến khích nó học các đặc trưng tổng quát hơn.
- **Dropout**: Kỹ thuật này loại bỏ ngẫu nhiên (đặt thành 0) một phần nhỏ các nơ-ron trong mỗi lần lặp huấn luyện. Điều này buộc mạng phải học các đặc trưng mạnh mẽ hơn không phụ thuộc vào bất kỳ nơ-ron đơn lẻ nào.
- **Early stopping**: Điều này bao gồm việc theo dõi hiệu suất của autoencoder trên tập hợp xác thực trong quá trình huấn luyện và dừng quá trình huấn luyện khi hiệu suất

trên tập hợp xác thực bắt đầu giảm. Điều này giúp ngăn mạng bị overfitting với dữ liệu huấn luyện.

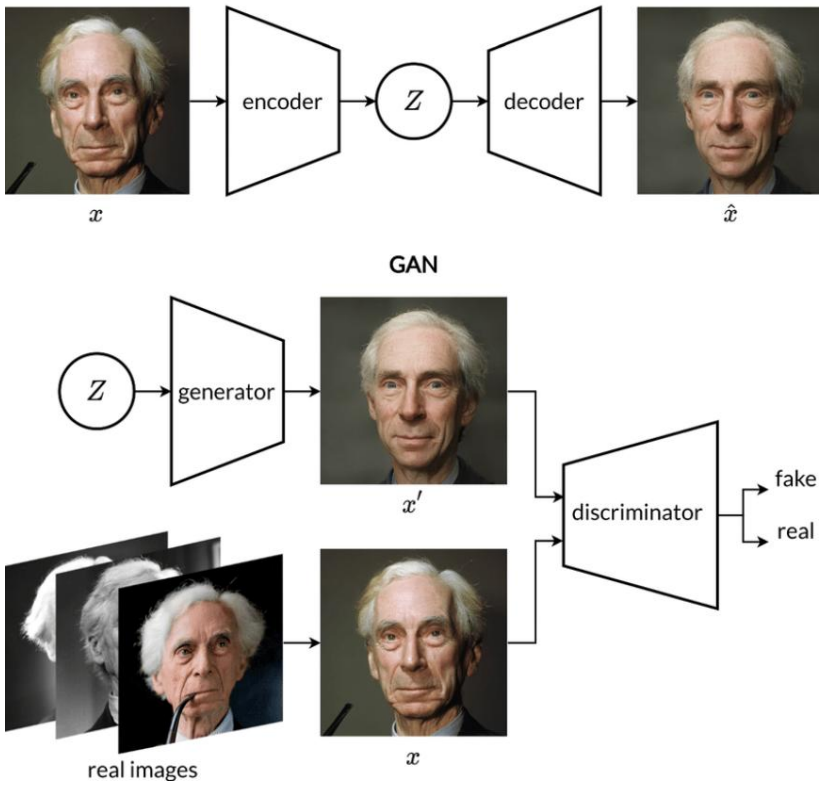
2.3.4 So sánh với các phương pháp khác

Autoencoders thường được so sánh với các kỹ thuật giảm chiều khác như Principal Component Analysis (PCA) và t-distributed Stochastic Neighbor Embedding (t-SNE). Trong khi PCA là một kỹ thuật tuyến tính tìm các thành phần chính của dữ liệu với cơ sở trực giao, autoencoders có thể mô hình hóa các hàm phi tuyến tính phức tạp, làm cho chúng phù hợp hơn với các tập dữ liệu phức tạp có cấu trúc phi tuyến tính. PCA có thể được coi là một trường hợp đặc biệt của autoencoder một tầng với hàm kích hoạt tuyến tính. Tuy nhiên, do số lượng tham số lớn, autoencoders dễ bị overfitting, mặc dù chính quy hóa và thiết kế cẩn thận có thể giảm thiểu vấn đề này.

t-SNE là một kỹ thuật phi tuyến tính khác tập trung vào việc bảo toàn cấu trúc lân cận cục bộ của dữ liệu, làm cho nó hữu ích cho việc hình dung. Tuy nhiên, không giống như autoencoders, t-SNE không học ánh xạ tham số có thể được sử dụng để mã hóa các điểm dữ liệu mới. Autoencoders và t-SNE có thể được sử dụng cùng nhau, trong đó autoencoder trước tiên giảm chiều dữ liệu và sau đó t-SNE được áp dụng để giảm hơn nữa dữ liệu xuống hai hoặc ba chiều để hình dung, tận dụng điểm mạnh của cả hai kỹ thuật.

Autoencoders cũng được so sánh với các mô hình tạo sinh như Generative Adversarial Networks (GANs). Trong khi cả hai đều có thể tạo ra các mẫu dữ liệu mới, chúng khác nhau về cách tiếp cận và cách chúng học không gian tiềm ẩn. Autoencoders học một cách rõ ràng một biểu diễn nén của dữ liệu trong không gian tiềm ẩn và sau đó sử dụng nó để tạo ra các mẫu mới, trong khi GANs học một cách ngầm định một không gian tiềm ẩn được sử dụng để tạo ra các mẫu dữ liệu mới nhưng không thể diễn giải trực tiếp. GANs thường được biết là khó huấn luyện hơn autoencoders, vì chúng liên quan đến min-max game giữa generator và discriminator,

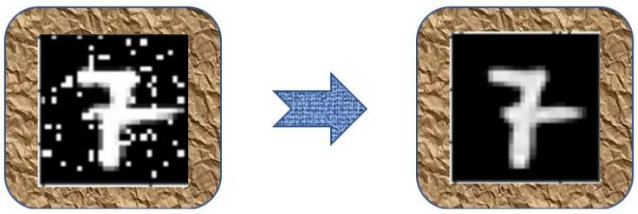
và chúng dễ gặp phải các vấn đề như sập chế độ hơn, trong đó generator tạo ra các loại đầu ra hạn chế.



Hình 2.23 So sáng Autoencoder vs GAN

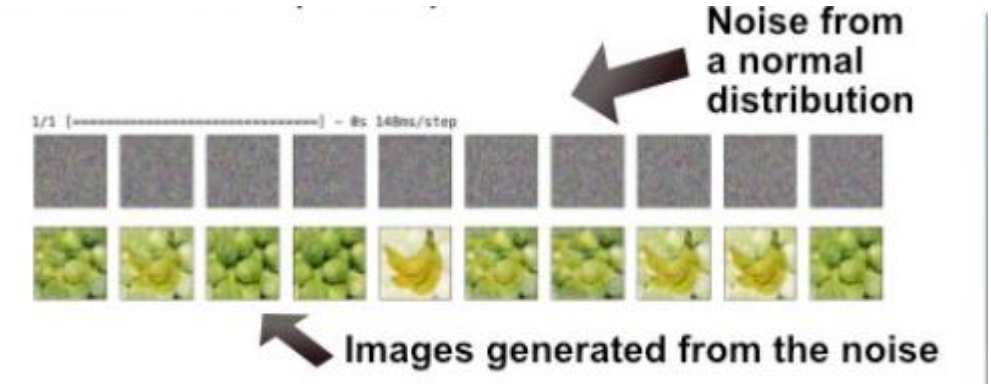
2.3.5 Ứng dụng của Autoencoders

Loại bỏ nhiễu hình ảnh: Denoising autoencoders có thể được sử dụng để loại bỏ nhiễu khỏi hình ảnh bằng cách huấn luyện chúng để tái tạo hình ảnh sạch từ đầu vào nhiễu. Điều này đạt được bằng cách làm hỏng hình ảnh đầu vào bằng nhiễu và sau đó huấn luyện autoencoder để tái tạo lại hình ảnh gốc, sạch sẽ.



Hình 2.24 Loại bỏ nhiễu

Tạo hình ảnh mới: Variational autoencoders có thể được sử dụng để tạo ra các hình ảnh mới tương tự như dữ liệu huấn luyện bằng cách lấy mẫu từ phân phối không gian tiềm ẩn đã học. Bằng cách thao tác các giá trị trong không gian tiềm ẩn, chúng ta có thể tạo ra các hình ảnh mới với các biến thể trong các đặc trưng được nắm bắt bởi autoencoder.



Hình 2.25 Tạo hình ảnh mới

Phát hiện bất thường: Autoencoders có thể được sử dụng để phát hiện bất thường bằng cách huấn luyện chúng trên dữ liệu bình thường và sau đó xác định các điểm dữ liệu có lỗi tái tạo cao. Các bất thường, khác với dữ liệu bình thường, sẽ dẫn đến lỗi tái tạo cao hơn vì autoencoder, được huấn luyện trên dữ liệu bình thường, sẽ không thể tái tạo chúng một cách chính xác.

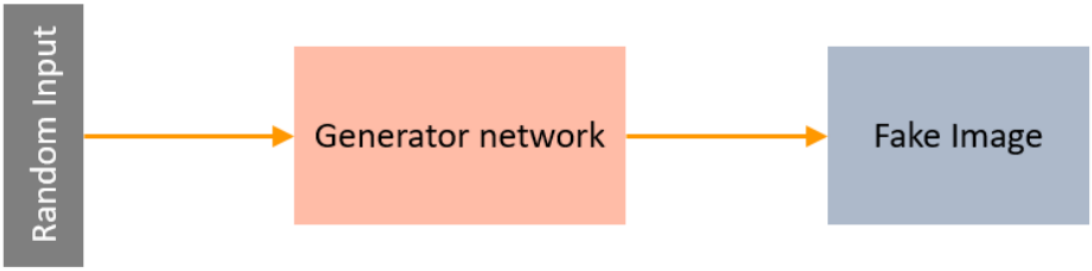
Giảm chiều: Autoencoders, đặc biệt là undercomplete autoencoders, có thể được sử dụng để giảm chiều bằng cách học một biểu diễn nén của dữ liệu trong không gian tiềm ẩn. Điều này có thể hữu ích cho các nhiệm vụ khác nhau như hình dung, trích xuất đặc trưng và giảm chi phí tính toán của các thuật toán học máy tiếp theo.

Xử lý ngôn ngữ tự nhiên: Autoencoders có thể được sử dụng cho các nhiệm vụ NLP khác nhau như tóm tắt văn bản, dịch máy và phân tích tình cảm. Chúng có thể học các biểu diễn nén của dữ liệu văn bản, nắm bắt các mối quan hệ ngữ nghĩa giữa các từ và tạo văn bản mới có nghĩa tương tự.

Điều quan trọng là autoencoders còn được gọi là self-encoders và chúng có bản chất mất mát. Điều này có nghĩa là đầu ra sẽ là một phiên bản suy giảm của đầu vào, mặc dù mục tiêu là giảm thiểu sự suy giảm này.

2.4 Generative Adversarial Network (GAN)

Generative Adversarial Networks (GANs) đã nổi lên như một lớp mô hình học sâu mạnh mẽ với khả năng tạo ra các mẫu dữ liệu chân thực và đa dạng. Được giới thiệu bởi Ian Goodfellow và cộng sự vào năm 2014, GANs đã cách mạng hóa lĩnh vực mô hình hóa tạo sinh, tác động đến các lĩnh vực khác nhau như thị giác máy tính, xử lý ngôn ngữ tự nhiên và khám phá thuốc. Báo cáo này cung cấp một cái nhìn tổng quan toàn diện về GANs, đi sâu vào kiến trúc, quy trình huấn luyện, các biến thể, thách thức và ứng dụng của chúng.



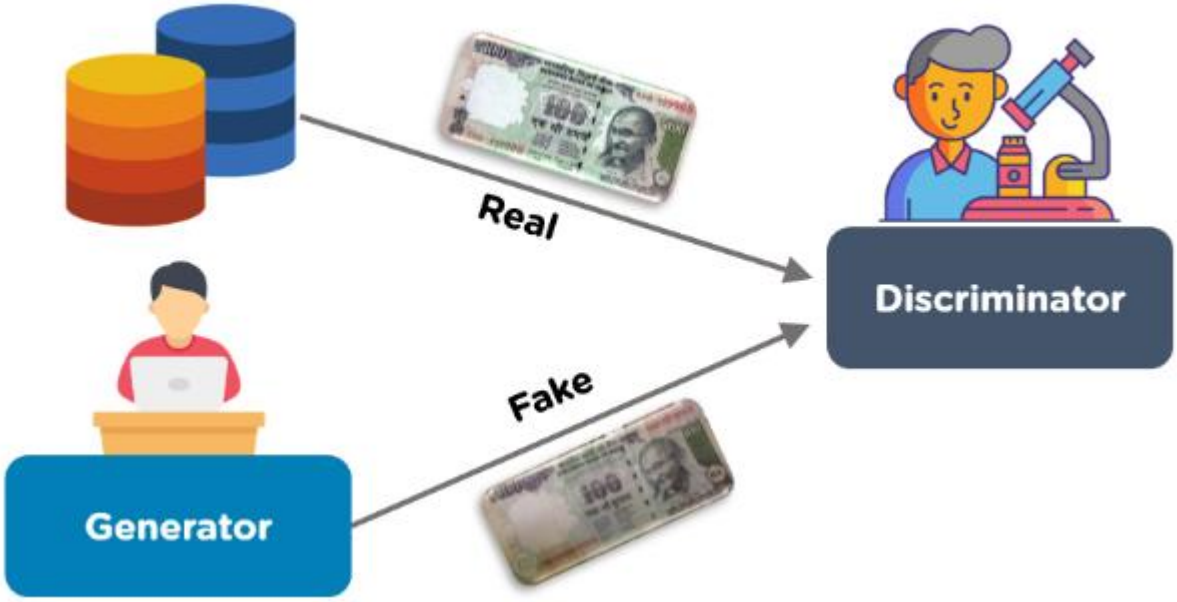
Hình 2.26 Mục tiêu của GANs

2.4.1 Cách hoạt động của GANs

GANs bao gồm hai mạng nơ-ron, **generator** và **discriminator**, tham gia vào một competitive game, một khái niệm được gọi là two-player minimax game. Mục tiêu của generator là tạo ra các mẫu dữ liệu tổng hợp gần giống với dữ liệu thực, trong khi discriminator tập trung vào việc phân biệt giữa các mẫu được tạo này và các mẫu thực tế từ tập dữ liệu huấn luyện. Quá trình đối nghịch này buộc cả hai mạng phải cải thiện hiệu suất của chúng theo cách lặp đi lặp lại.

Generator lấy nhiễu ngẫu nhiên làm đầu vào và áp dụng một loạt các phép biến đổi để tạo ra một mẫu dữ liệu tổng hợp. Mặt khác, discriminator nhận cả các mẫu thực

từ dữ liệu huấn luyện và các mẫu giả do generator tạo ra. Nhiệm vụ của nó là phân tích các đầu vào này và đưa ra xác suất cho biết khả năng đầu vào là thật hay giả.



Hình 2.27 Discriminator

2.4.2 Huấn luyện GAN

Huấn luyện GAN có thể được chia thành các bước sau:

1. Xác định vấn đề: Xác định loại dữ liệu bạn muốn tạo (ví dụ: hình ảnh, văn bản, âm nhạc).
2. Chọn kiến trúc GAN: Chọn kiến trúc GAN phù hợp dựa trên vấn đề và dữ liệu (ví dụ: DCGAN cho hình ảnh, SeqGAN cho văn bản).
3. Huấn luyện discriminator trên dữ liệu thực: Cung cấp cho discriminator các mẫu dữ liệu thực và huấn luyện nó để phân loại chúng là thật.
4. Tạo đầu vào giả cho generator: Sử dụng generator để tạo các mẫu dữ liệu giả từ nhiễu ngẫu nhiên.
5. Huấn luyện discriminator trên dữ liệu giả: Cung cấp cho discriminator các mẫu dữ liệu giả và huấn luyện nó để phân loại chúng là giả.

6. Huấn luyện generator với đầu ra của discriminator: Sử dụng đầu ra của discriminator để hướng dẫn generator cải thiện chất lượng của các mẫu giả của nó.
7. Lặp lại bước 3-6: Lặp lại quy trình huấn luyện cho đến khi generator tạo ra dữ liệu thực tế có thể đánh lừa discriminator.

2.4.3 Công thức toán học

Nền tảng của GANs nằm trong một min-max game giữa generator (G) và discriminator (D). Hàm mục tiêu của trò chơi này được định nghĩa là:

$$V(G,D) = E_{x\sim p_{data}}\big[\log\big(D(x)\big)\big] + E_{z\sim p_z}[\log\,(1 - D\big(G(x)\big))]$$

trong đó:

x: biểu thị một mẫu dữ liệu thực.

z: biểu thị một nhiễu ngẫu nhiên đầu vào cho generator.

p_{data}: là phân phối của dữ liệu thực.

p_z: là phân phối của nhiễu đầu vào.

$D(x)$: là đầu ra của discriminator cho một mẫu thực x, biểu thị xác suất x là thật.

$G(z)$: là đầu ra của generator cho một nhiễu đầu vào z.

$D(G(z))$: là đầu ra của discriminator cho một mẫu được tạo G(z), biểu thị xác suất $G(z)$ là thật.

Discriminator mục đích tối đa hóa hàm giá trị này. Nói cách khác, nó cố gắng gán xác suất cao cho các mẫu dữ liệu thực (D(x)) và xác suất thấp cho các mẫu được tạo (D(G(z))). Điều này có thể được hiểu bằng cách xem xét hai phần của phương trình riêng biệt:

- $E_{x\sim p_{data}}[log(D(x))]$: Thuật ngữ này khuyến khích discriminator gán xác suất cao cho các mẫu dữ liệu thực. Xác suất D(x) càng cao thì giá trị của log(D(x)) càng lớn, và do đó giá trị của thuật ngữ này càng cao.

- $E_{z\sim p_z}[log(1-D(G(z)))]$: Thuật ngữ này khuyến khích discriminator gán xác suất thấp cho các mẫu được tạo. Xác suất D(G(z)) càng thấp thì giá trị của (1−D(G(z))) càng lớn, và do đó giá trị của log(1−D(G(z))) càng cao.

Mặt khác, generator nhằm mục đích giảm thiểu hàm giá trị. Điều này có nghĩa là nó cố gắng tạo ra các mẫu mà discriminator sẽ phân loại là thật với xác suất cao. Discriminator tối ưu, được ký hiệu là $D^*(x)$, đạt được khi:

$$D^*(x) = \frac{p_{data}(x)}{p_{data(X)}+p_g(x)}$$

trong đó p_g là phân phối của dữ liệu được tạo. Công thức này chỉ ra rằng discriminator tối ưu gán xác suất cho một mẫu tỷ lệ với tỷ lệ xác suất của mẫu đó đến từ phân phối dữ liệu thực so với xác suất của nó đến từ phân phối dữ liệu thực hoặc được tạo.

Hàm mất mát của generator có thể được biểu thị là:

$$J_G = -\frac{m}{1}\sum_{i=1}^{m} logD(G(z_i))$$

trong đó m là số lượng mẫu. Hàm mất mát này khuyến khích generator tạo ra các mẫu mà discriminator phân loại là thật với xác suất cao.

Hàm mất mát của discriminator là:

$$J_D = -\frac{m}{1}\sum_{i=1}^{m}[logD(x_i) + log(1-D(G(z_i)))]$$

Hàm mất mát này khuyến khích discriminator phân loại chính xác các mẫu thực là thật (bằng cách tối đa hóa $logD(x_i)$) và các mẫu được tạo là giả (bằng cách tối đa hóa $log(1-D(G(z_i)))$). Các hàm mất mát này được sử dụng để cập nhật các tham số của generator và discriminator trong quá trình huấn luyện.

Ngoài các hàm kích hoạt thường được sử dụng như ReLU và Leaky ReLU, một hàm kích hoạt gần đây được gọi là "swish" đã được đề xuất. Công thức của nó là

$f(x) = x \cdot sigmoid(x)$. Swish là một hàm trơn, đơn điệu, linh hoạt hơn để kích hoạt/hủy kích hoạt và được cho là hoạt động tốt hơn trong các kiến trúc sâu.

2.4.4 Các loại GANs

Loại GAN	Kiến trúc	Ưu điểm	Nhược điểm
DCGAN	Convolutional neural networks	Ổn định tăng lên, biểu diễn hình ảnh tốt hơn	Có thể khó huấn luyện trên các tập dữ liệu phức tạp
cGAN	Conditional input	Kiểm soát đầu ra được tạo	Yêu cầu dữ liệu được gắn nhãn
CycleGAN	Two generators and two discriminators	Image-to-image translation without paired data	Có thể bị sụp đổ chế độ
StyleGAN	Style-based generator	Hình ảnh độ phân giải cao, kiểm soát kiểu chi tiết	Tốn kém về mặt tính toán
BigGAN	Scaled-up architecture	Hình ảnh có độ trung thực cao, chất lượng và sự đa dạng được cải thiện	Yêu cầu tài nguyên tính toán đáng kể

Bảng 2.4 Các loại GANs

Kể từ khi GAN ban đầu được giới thiệu, nhiều biến thể đã được đề xuất để giải quyết các hạn chế và cải thiện hiệu suất. Một số loại GANs đáng chú ý bao gồm:

Deep Convolutional GAN (DCGAN): Biến thể này sử dụng các mạng nơ-ron tích chập cho cả generator và discriminator, làm cho nó phù hợp với các nhiệm vụ tạo hình ảnh. DCGANs tận dụng các lớp tích chập, chuẩn hóa theo batch và kích hoạt leaky ReLU trong discriminator, trong khi generator sử dụng các lớp chuyển vị tích chập, chuẩn hóa theo batch và kích hoạt ReLU. Lựa chọn kiến trúc này đã được chứng minh là cải thiện sự ổn định của việc học và chất lượng của hình ảnh được tạo.

Conditional GAN (cGAN): cGANs cho phép tạo các mẫu dữ liệu có điều kiện dựa trên các thuộc tính hoặc nhãn cụ thể. Điều này cho phép kiểm soát đầu ra được tạo, chẳng hạn như tạo hình ảnh của các danh mục cụ thể hoặc với các tính năng mong muốn. Bằng cách cung cấp cho cả generator và discriminator thông tin bổ sung, cGANs có thể hướng dẫn quá trình tạo đến các kết quả cụ thể.

CycleGAN: CycleGANs được thiết kế cho các nhiệm vụ dịch hình ảnh sang hình ảnh, trong đó mục tiêu là học các ánh xạ giữa hai miền hình ảnh khác nhau. Chúng có thể dịch hình ảnh từ miền này sang miền khác mà không yêu cầu dữ liệu huấn luyện được ghép nối. Điều này đạt được bằng cách sử dụng hai generator và hai discriminator, trong đó mỗi generator dịch hình ảnh từ miền này sang miền khác và các discriminator đảm bảo hình ảnh được dịch không thể phân biệt được với hình ảnh thực trong miền đích.

StyleGAN: StyleGANs tập trung vào việc tạo hình ảnh có độ phân giải cao với khả năng kiểm soát chi tiết các mức độ chi tiết khác nhau. Chúng đạt được điều này bằng cách kết hợp kiến trúc generator dựa trên kiểu cho phép thao tác kiểu của hình ảnh được tạo ở nhiều tỷ lệ khác nhau. Cách tiếp cận dựa trên kiểu này cho phép tạo hình ảnh với các mức độ chi tiết khác nhau, từ các đặc điểm thô như tư thế và danh tính đến các chi tiết tinh tế như tóc và tàn nhang.

BigGAN: BigGANs được biết đến với việc tạo ra hình ảnh có độ trung thực cao với chất lượng và sự đa dạng được cải thiện. Chúng đạt được điều này bằng cách mở rộng quy mô kiến trúc GAN, sử dụng kích thước batch lớn hơn và nhiều tham số mô hình hơn. Quy mô tăng lên này cho phép BigGANs học các phân phối dữ liệu phức tạp và tinh tế hơn, dẫn đến hình ảnh được tạo chân thực và đa dạng hơn.

❖ **So sánh các Loại GANs**

Mặc dù tất cả các biến thể GAN đều chia sẻ nguyên tắc cốt lõi của huấn luyện đối nghịch, nhưng kiến trúc và điểm mạnh của chúng khác nhau đáng kể. DCGANs, với kiến trúc tích chập của chúng, rất phù hợp với các nhiệm vụ tạo hình ảnh và mang lại sự cân bằng tốt giữa độ ổn định và hiệu suất. cGANs cung cấp khả năng kiểm soát đầu ra được tạo bằng cách kết hợp thông tin có điều kiện, làm cho chúng hữu ích cho các nhiệm vụ như tạo hình ảnh với các thuộc tính cụ thể. CycleGANs vượt trội trong việc dịch hình ảnh sang hình ảnh, cho phép chuyển đổi hình ảnh giữa các miền khác nhau mà không cần dữ liệu được ghép nối. StyleGANs vượt qua ranh giới của việc tạo

hình ảnh bằng cách tập trung vào hình ảnh có độ phân giải cao và kiểm soát kiểu chi tiết. BigGANs tăng cường hơn nữa chất lượng và sự đa dạng của hình ảnh bằng cách mở rộng quy mô kiến trúc GAN. Việc lựa chọn loại GAN phụ thuộc vào ứng dụng cụ thể và các đặc điểm mong muốn của dữ liệu được tạo.

2.4.5 Huấn luyện GANs

Huấn luyện GANs liên quan đến việc cân bằng giữa generator và discriminator. Quá trình huấn luyện có thể gặp nhiều thách thức do các vấn đề như:

- **Mode collapse:** Điều này xảy ra khi generator tạo ra một loạt các mẫu hạn chế, không thể nắm bắt được sự đa dạng đầy đủ của dữ liệu huấn luyện. Điều này có thể xảy ra khi discriminator trở nên quá giỏi trong việc xác định các mẫu giả, khiến generator tối ưu hóa quá mức cho một tập hợp nhỏ các đầu ra có thể đánh lừa discriminator.
- **Vanishing gradients:** Điều này xảy ra khi discriminator trở nên quá giỏi trong việc phân biệt các mẫu thực và giả, dẫn đến các gradient nhỏ cản trở việc học của generator. Việc lựa chọn hàm kích hoạt có thể ảnh hưởng đến sự xuất hiện của gradient biến mất. Ví dụ, sử dụng các hàm kích hoạt sigmoid hoặc tanh trong các mạng sâu có thể làm trầm trọng thêm vấn đề này.
- **Non-convergence:** Quá trình huấn luyện có thể không hội tụ đến một giải pháp ổn định do tính chất đối nghịch của game. Điều này có thể biểu hiện dưới dạng dao động trong hiệu suất của generator và discriminator, trong đó những cải tiến trong một mạng dẫn đến sự suy giảm trong mạng kia.

Để giải quyết những thách thức này, nhiều kỹ thuật khác nhau đã được đề xuất, bao gồm:

- **Wasserstein GAN (WGAN):** WGANs sử dụng khoảng cách Wasserstein làm hàm mất mát, cung cấp gradient mượt mà hơn và cải thiện độ ổn định của huấn luyện. Điều này giải quyết vấn đề gradient biến mất bằng cách cung cấp tín hiệu học tập có ý nghĩa hơn cho generator, ngay cả khi discriminator đang hoạt động tốt.

- **Spectral Normalization:** Kỹ thuật này ràng buộc chuẩn phổ của ma trận trọng số của discriminator, ngăn discriminator trở nên quá mạnh và cải thiện độ ổn định. Bằng cách hạn chế khả năng áp đảo generator của discriminator, chuẩn hóa phổ giúp duy trì động lực huấn luyện cân bằng hơn.

2.4.6 Ứng dụng của GANs

Tạo hình ảnh: Tạo hình ảnh chân thực về khuôn mặt, đồ vật và cảnh vật. Ví dụ: Tạo hình ảnh tổng hợp về khuôn mặt người để sử dụng trong thế giới ảo hoặc trò chơi điện tử.

Siêu phân giải hình ảnh: Nâng cao độ phân giải của hình ảnh, tạo hình ảnh có độ phân giải cao từ đầu vào có độ phân giải thấp. Ví dụ: Nâng cấp hình ảnh y tế có độ phân giải thấp để cải thiện độ chính xác chẩn đoán.

Chuyển đổi kiểu hình ảnh: Chuyển giao phong cách của một hình ảnh này sang một hình ảnh khác, tạo hiệu ứng nghệ thuật hoặc tạo hình ảnh với phong cách cụ thể. Ví dụ: Áp dụng phong cách của một bức tranh nổi tiếng cho một bức ảnh.

Tạo nhạc: Tạo các chuỗi âm nhạc, sáng tác giai điệu mới hoặc hòa âm các giai điệu hiện có. Ví dụ: Tạo các bản nhạc mới theo các phong cách khác nhau, chẳng hạn như cổ điển hoặc jazz.

Tạo văn bản: Tạo văn bản, chẳng hạn như thơ, bài báo hoặc đoạn hội thoại. Ví dụ: Tạo đoạn hội thoại thực tế cho chatbot hoặc trợ lý ảo.

Dược phẩm: Khám phá thuốc, tạo ra các phân tử mới với các đặc tính mong muốn. Ví dụ: Thiết kế các loại thuốc mới có hiệu quả được cải thiện và giảm tác dụng phụ.

Thiên văn học: Tạo hình ảnh thiên văn, mô phỏng sự hình thành thiên hà và phân tích dữ liệu kính viễn vọng. Ví dụ: Tạo hình ảnh tổng hợp về các thiên hà để huấn luyện các mô hình nhận dạng hình ảnh thiên văn.

An ninh mạng: Tạo ra các ví dụ đối nghịch, là những đầu vào được thiết kế để đánh lừa các mô hình học máy. Ví dụ, một hình ảnh có thể được sửa đổi một cách tinh

vi để khiến một hệ thống phân loại hình ảnh nhận dạng sai. Điều này có thể giúp các nhà nghiên cứu hiểu rõ hơn về các lỗ hổng trong các hệ thống AI và phát triển các biện pháp đối phó mạnh mẽ hơn. Ngoài ra, GANs có thể hỗ trợ trong việc chỉnh sửa thông tin ngữ nghĩa của dữ liệu, ví dụ như thay đổi nội dung của hình ảnh hoặc văn bản mà không làm thay đổi cấu trúc tổng thể của chúng. Điều này có ứng dụng tiềm năng trong việc bảo vệ quyền riêng tư và ẩn danh dữ liệu. GANs cũng có thể được sử dụng để tạo ra các mẫu phần mềm độc hại mới, giúp các nhà nghiên cứu an ninh mạng chủ động xác định và ngăn chặn các mối đe dọa mới nổi.

2.4.7 Ưu điểm và nhược điểm của GANs

So với các mô hình tạo sinh khác như Variational Autoencoders (VAEs) và Flow-based models, GANs có một số ưu điểm:

- **Mẫu chất lượng cao:** GANs có thể tạo ra các mẫu rất thực tế và đa dạng, thường vượt qua chất lượng của các mẫu được tạo bởi các mô hình khác.
- **Implicit modeling:** GANs không yêu cầu xác định rõ ràng phân phối dữ liệu, làm cho chúng linh hoạt hơn đối với dữ liệu phức tạp.
- **Anomaly detection:** GANs có thể được sử dụng để xác định các bất thường bằng cách đo lường mức độ mà bộ tạo và bộ phân biệt có thể mô hình hóa dữ liệu.

Tuy nhiên, GANs cũng có một số nhược điểm:

- **Huấn luyện không ổn định:** Việc huấn luyện GANs có thể gặp khó khăn do các vấn đề như sụp đổ chế độ và gradient biến mất.
- **Khó khăn trong đánh giá:** Việc đánh giá hiệu suất của GANs có thể mang tính chủ quan và khó khăn do thiếu khả năng đánh giá rõ ràng.
- **Yêu cầu dữ liệu:** GANs thường yêu cầu tập dữ liệu lớn, đa dạng và tiên tiến để huấn luyện hiệu quả.

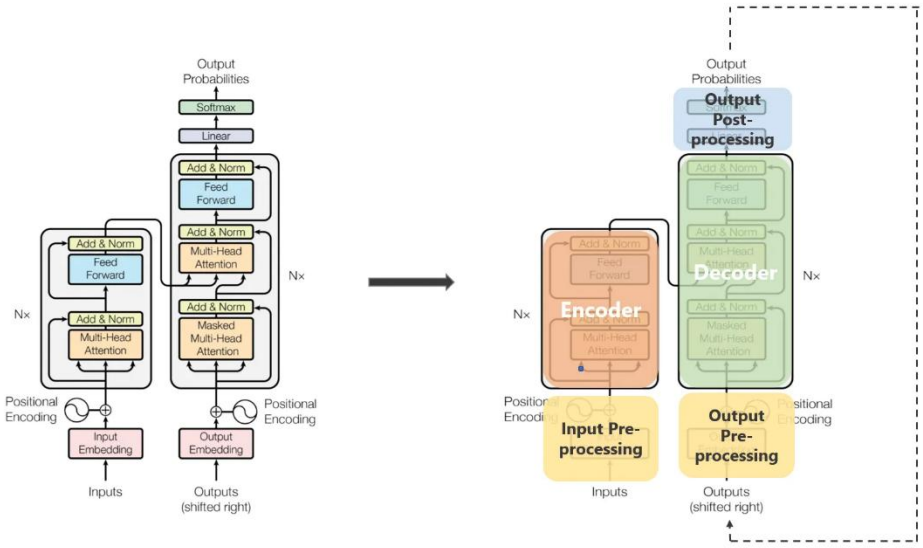
2.5 Vision Transformer

Vision Transformer (ViT) đã nổi lên như một kiến trúc đột phá trong lĩnh vực thị giác máy tính, thách thức sự thống trị lâu dài của Convolutional Neural Networks (CNN). Phần này đi sâu vào sự phức tạp của ViT, cung cấp giải thích chi tiết về kiến trúc, các biến thể, ứng dụng và lợi thế của nó so với CNN. Nó nhằm mục đích cung cấp sự hiểu biết toàn diện về công nghệ biến đổi này và tác động tiềm tàng của nó đối với tương lai của thị giác máy tính.

2.5.1 Kiến trúc Transformer trong xử lý ngôn ngữ tự nhiên (NLP)

Trước khi đi sâu vào chi tiết cụ thể của ViT, điều cần thiết là phải hiểu nền tảng mà nó được xây dựng: kiến trúc Transformer từ Xử lý ngôn ngữ tự nhiên (NLP).

Transformer được giới thiệu trong bài báo quan trọng "Attention is All You Need" và đã cách mạng hóa các nhiệm vụ NLP. Chúng dựa trên cấu trúc bộ mã hóa-bộ giải mã và cơ chế tự chú ý để xử lý dữ liệu tuần tự như văn bản. Bộ mã hóa xử lý chuỗi đầu vào, chẳng hạn như một câu, và tạo ra một biểu diễn ngữ cảnh của từng từ. Sau đó, bộ giải mã sử dụng biểu diễn này để tạo ra một chuỗi đầu ra, chẳng hạn như bản dịch của câu đầu vào.



Hình 2.28 Kiến trúc Transformer

Sự đổi mới cốt lõi của Transformer nằm ở cơ chế tự chú ý, cho phép mô hình cân nhắc tầm quan trọng của các từ khác nhau liên quan đến nhau, nắm bắt các phụ thuộc tầm xa trong chuỗi. Để bảo toàn thông tin vị trí, điều quan trọng để hiểu thứ tự từ, Transformer đưa một vectơ vào các nhúng đầu vào riêng lẻ. Các vectơ này tuân theo một hàm tuần hoàn cụ thể, thường là sự kết hợp của sóng sin và cosin với các tần số khác nhau. Điều này cho phép mô hình học và sử dụng thông tin vị trí một cách hiệu quả.

2.5.2 Sự ra đời của ViT

Mô hình ViT, lấy cảm hứng từ sự thành công của Transformer trong NLP, lần đầu tiên được giới thiệu trong bài báo "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale" của Alexey Dosovitskiy và cộng sự vào năm 2020. Nghiên cứu này đã đánh dấu một sự thay đổi mô hình trong thị giác máy tính bằng cách áp dụng thành công kiến trúc Transformer vào các nhiệm vụ nhận dạng hình ảnh. Ý tưởng cốt lõi đằng sau ViT là coi một hình ảnh như một chuỗi các mảnh, tương tự như một chuỗi các từ trong một câu, và tận dụng cơ chế tự chú ý của Transformer để nắm bắt mối quan hệ giữa các mảnh này.

Điều quan trọng cần lưu ý là sự thành công của ViT không chỉ do kiến trúc mới lạ mà còn phụ thuộc nhiều vào phương pháp huấn luyện và tập dữ liệu được sử dụng để huấn luyện trước. Các nhà nghiên cứu đã huấn luyện trước ViT trên các tập dữ liệu lớn như ImageNet-21k và sau đó tinh chỉnh nó trên các tập dữ liệu nhỏ hơn, đạt được kết quả ấn tượng so với các CNN hiện đại.

2.5.3 Kiến trúc ViT

Kiến trúc ViT phản ánh chặt chẽ kiến trúc Transformer được sử dụng trong NLP, với các điều chỉnh chính để xử lý dữ liệu hình ảnh. Quá trình này có thể được chia thành các bước sau:

- **Image Patching:** Hình ảnh đầu vào được chia thành các mảnh có kích thước cố định, thường là 16x16 pixel. Mỗi mảnh được coi như một mã thông báo, tương tự

như một từ trong câu. Quá trình này về cơ bản biến đổi hình ảnh 2D thành một chuỗi các nhúng mảnh 1D.

- **Linear Embedding:** Mỗi mảnh hình ảnh được làm phẳng thành một vectơ và sau đó được nhúng tuyến tính vào một không gian nhiều chiều, tạo ra một chuỗi các nhúng mã thông báo. Quá trình nhúng này ánh xạ các giá trị pixel thô của mỗi mảnh tới một biểu diễn có ý nghĩa hơn mà Transformer có thể xử lý.
- **Position Embeddings:** Các nhúng vị trí được thêm vào các nhúng mã thông báo để giữ lại thông tin về sự sắp xếp không gian của các mảnh trong hình ảnh gốc. Điều này rất quan trọng vì, không giống như CNN, ViT không vốn có khả năng nắm bắt các mối quan hệ không gian. Các nhúng vị trí này không tĩnh; chúng được cập nhật động thông qua cơ chế chú ý, cho phép mô hình tinh chỉnh sự hiểu biết của nó về các mối quan hệ không gian khi nó xử lý hình ảnh.

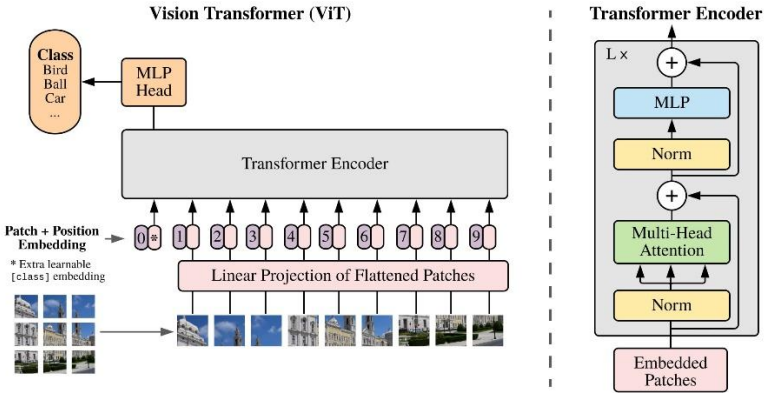
$$PE_{pos,2i} = \sin\left(\frac{pos}{1000^{\frac{2i}{d_{model}}}}\right)$$

$$PE_{pos,2i+1} = \cos\left(\frac{pos}{1000^{\frac{2i}{d_{model}}}}\right)$$

- **Transformer Encoder:** Chuỗi các mảnh được nhúng, cùng với một mã thông báo phân loại đặc biệt ([CLS]), được đưa vào một bộ mã hóa Transformer tiêu chuẩn. Bộ mã hóa bao gồm nhiều lớp tự chú ý đa đầu và mạng chuyển tiếp. Cơ chế tự chú ý cho phép mô hình cân nhắc tầm quan trọng của các mảnh khác nhau liên quan đến nhau, nắm bắt các phụ thuộc toàn cục trong hình ảnh. Tuy nhiên, cơ chế chú ý này có yêu cầu về bộ nhớ và tính toán cao do độ phức tạp bậc hai của nó. Hạn chế này đã thúc đẩy sự phát triển của các biến thể ViT như Swin Transformer, nhằm mục đích cải thiện hiệu quả.

$$A_{ij} = \frac{\exp\left(q_i^T k_j / \sqrt{d}\right)}{\sum_{k=1}^{N} \exp\left(q_i^T k\_k / \sqrt{d}\right)}$$

- **Classification Head:** Đầu ra tương ứng với mã thông báo [CLS] từ lớp mã hóa cuối cùng được chuyển qua đầu phân loại, thường là đầu Perceptron Đa tầng (MLP), để dự đoán nhãn lớp cho hình ảnh.

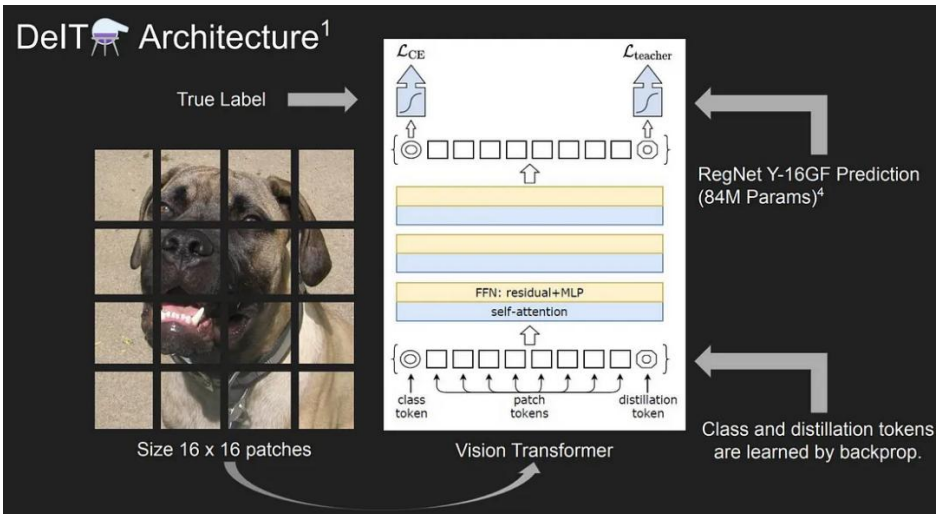


Hình 2.29 Kiến trúc ViT

2.5.4 Các loại ViT

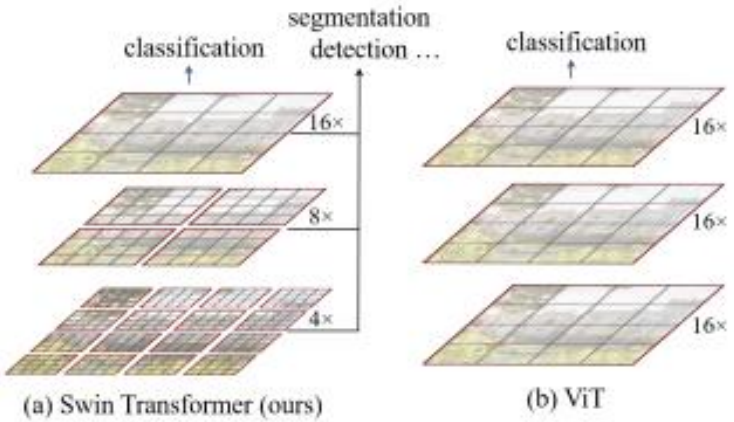
Kể từ khi ViT ra đời, một số biến thể đã được đề xuất để giải quyết các hạn chế của nó và cải thiện hiệu suất. Một số biến thể đáng chú ý bao gồm:

- **DeiT (Data-efficient Image Transformer):** DeiT tập trung vào việc cải thiện hiệu quả dữ liệu của ViT bằng cách sử employing a distillation token và chiến lược teacher-student training. Điều này cho phép DeiT đạt được hiệu suất tương đương với ViT trong khi yêu cầu ít dữ liệu huấn luyện hơn đáng kể. Việc giảm dữ liệu và thời gian huấn luyện này giúp DeiT thân thiện với môi trường hơn bằng cách giảm mức tiêu thụ năng lượng.



Hình 2.30 Kiến trúc DieT

- **Swin Transformer:** Swin Transformer giới thiệu một kiến trúc phân cấp với các cửa sổ dịch chuyển để giải quyết độ phức tạp tính toán của ViT, có quy mô bậc hai với kích thước hình ảnh đầu vào. Bằng cách hạn chế tự chú ý đối với các cửa sổ cục bộ và dịch chuyển các cửa sổ này qua các lớp, Swin Transformer đạt được độ phức tạp tuyến tính trong khi vẫn duy trì hiệu suất mạnh mẽ. Cách tiếp cận phân cấp này cho phép mô hình xây dựng các bản đồ đặc trưng ở nhiều quy mô khác nhau, tương tự như CNN, nhưng với lợi ích bổ sung của ngữ cảnh toàn cục từ cơ chế tự chú ý. Swin Transformer đã đạt được kết quả hiện đại trên điểm chuẩn phát hiện và phân đoạn đối tượng COCO, vượt qua các phương pháp trước đó với tỷ lệ đáng kể.



Hình 2.31 Swin Transformer

- **ViT-1.58b:** Biến thể này tập trung vào việc giảm chi phí bộ nhớ và tính toán của ViT bằng cách sử dụng lượng tử hóa tam phân, trong đó trọng số bị ràng buộc thành {-1, 0, 1} và các kích hoạt được lượng tử hóa với độ chính xác 8 bit. Điều này làm cho ViT-1.58b phù hợp để triển khai trong các môi trường hạn chế tài nguyên như thiết bị di động.
- **Các biến thể khác:** Một số biến thể ViT khác đã được đề xuất, mỗi biến thể có điểm mạnh và điểm yếu riêng. Chúng bao gồm:
    - **CvT (Convolutional Vision Transformer):** Kiến trúc lai này kết hợp điểm mạnh của cả CNN và ViT bằng cách kết hợp các phép tích chập vào kiến trúc ViT.
    - **T2T-ViT (Tokens-to-Token ViT):** Biến thể này giới thiệu một quy trình mã hóa mới lạ, hợp nhất dần các mảnh hình ảnh để tạo ra một biểu diễn phân cấp, cải thiện hiệu suất trên các tập dữ liệu nhỏ hơn.
    - **MobileViT:** Biến thể ViT nhẹ này được thiết kế cho thiết bị di động và thiết bị hạn chế tài nguyên, đạt được sự cân bằng tốt giữa độ chính xác và hiệu quả.

2.5.5 ViT so với CNN

Đặc trưng	ViT	CNNs
Receptive Field	Global	Local
Scalability	High	Moderate
Inductive Bias	Minimal	Strong
Data Efficiency	Lowe	Higher
Computational Cost	Higher	Lower

Bảng 2.5 So sánh ViT với CNN

ViT và CNN có các đặc điểm riêng biệt khiến chúng phù hợp với các tình huống khác nhau. ViT vượt trội trong việc nắm bắt các mối quan hệ toàn cục và mở rộng quy mô cho các tập dữ liệu lớn, trong khi CNN hiệu quả hơn về mặt tính toán và khái quát

hóa tốt hơn trên các tập dữ liệu nhỏ hơn. Độ lệch quy nạp tối thiểu của ViT làm cho nó linh hoạt hơn nhưng cũng đói dữ liệu hơn, yêu cầu tập dữ liệu lớn hơn hoặc kỹ thuật tăng cường dữ liệu để tránh quá khớp

2.5.6 Ứng dụng của ViT

Phân loại hình ảnh: ViT đã đạt được kết quả hiện đại trên các điểm chuẩn phân loại hình ảnh như ImageNet. Khả năng nắm bắt các mối quan hệ toàn cục trong hình ảnh làm cho nó đặc biệt phù hợp với tác vụ này, vì nó có thể xác định và phân loại các đối tượng một cách hiệu quả dựa trên ngữ cảnh và đặc điểm tổng thể của chúng.

Phát hiện đối tượng: Các mô hình dựa trên ViT như DETR đã cho thấy hiệu suất cạnh tranh trong các tác vụ phát hiện đối tượng. Bằng cách coi phát hiện đối tượng như một bài toán dự đoán tập hợp trực tiếp, DETR đơn giản hóa quy trình phát hiện và loại bỏ nhu cầu về các thành phần được thiết kế thủ công như hộp neo.

Phân đoạn hình ảnh: ViT đã được áp dụng thành công cho các tác vụ phân đoạn hình ảnh, đặc biệt là trong hình ảnh y tế. Khả năng nắm bắt các phụ thuộc tầm xa và thông tin ngữ cảnh làm cho nó hiệu quả trong việc phân định các cấu trúc và ranh giới phức tạp trong hình ảnh.

Nhận dạng hành động: ViT đã được sử dụng để nhận dạng hành động trong video, cho phép phân tích tự động các hành động của con người. Bằng cách xử lý các khung hình video dưới dạng chuỗi các mảnh, ViT có thể nắm bắt cả mối quan hệ không gian và thời gian, dẫn đến độ chính xác được cải thiện trong việc nhận dạng hành động.

Tạo hình ảnh: ViT đã cho thấy tiềm năng trong các tác vụ tạo hình ảnh, tạo ra hình ảnh có độ trung thực cao với hiệu quả tham số được cải thiện. Khả năng học các biểu diễn và phụ thuộc phức tạp trong hình ảnh làm cho nó phù hợp để tạo nội dung hình ảnh chân thực và đa dạng.

Hình 2.32 Vit trong phân đoạn hình ảnh

CHƯƠNG 3 - XÂY DỰNG MÔ HÌNH VÀ DEMO

Trong chương này, báo cáo sẽ tập trung triển khai các mô hình đã chọn cho từng chủ đề, bao gồm: Convolutional Neural Networks (CNNs), Mask R-CNN, AutoEncoder, DCGAN, và Vision Transformer (ViT). Quá trình thực hiện được thực hiện bằng các thư viện tiên tiến như TensorFlow, PyTorch, và HuggingFace. Các kết quả sẽ được trình bày chi tiết thông qua các bảng số liệu, hình ảnh và phân tích.

3.1Convolutional Neural Networks (CNNs)

❖ **Mô hình triển khai: MiniVGGNet**

MiniVGGNet, một phiên bản đơn giản hóa của VGGNet, được triển khai để thực hiện bài toán phân loại hình ảnh trên tập dữ liệu CIFAR-10. Đây là một trong những mô hình học sâu nổi bật, sử dụng cấu trúc tích chập và gộp để trích xuất đặc trưng từ hình ảnh và áp dụng vào các lớp đầu ra để phân loại.

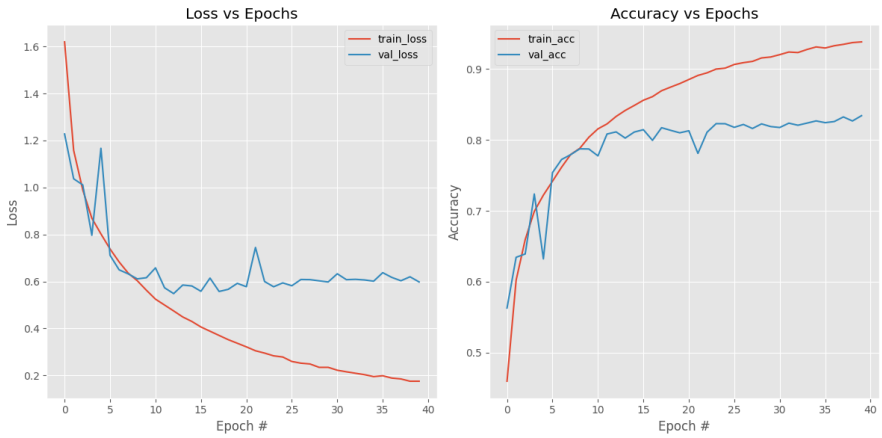
1. **Tập dữ liệu và tiền xử lý**
   - **Tập dữ liệu CIFAR-10:** Tập dữ liệu gồm 60.000 hình ảnh màu th uộc 10 lớp (chẳng hạn như máy bay, ô tô, chim, chó, mèo). Mỗi hình ảnh có kích thước 32x32 pixel.
   - **Tiền xử lý:**
     - Các giá trị pixel của hình ảnh được chuẩn hóa về khoảng [0, 1] bằng cách chia cho 255, giúp tăng tốc độ hội tụ của mô hình.
     - Các nhãn được chuyển đổi thành dạng one-hot encoding để phù hợp với đầu ra của mô hình.
2. **Kiến trúc mô hình MiniVGGNet**

MiniVGGNet được xây dựng dựa trên các tầng cơ bản sau:

- **Lớp tích chập (Convolutional Layers):** Mỗi lớp sử dụng các bộ lọc để trích xuất đặc trưng quan trọng từ hình ảnh, giúp mô hình nhận diện chi tiết hình dạng hoặc cấu trúc.

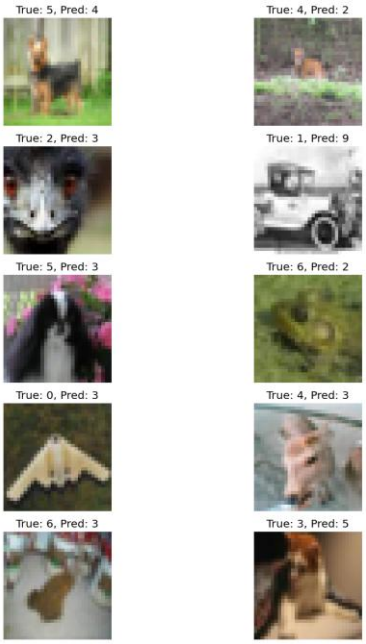
- **Batch Normalization:** Được thêm sau các tầng tích chập để ổn định quá trình huấn luyện và tăng tốc độ hội tụ.
- **MaxPooling:** Làm giảm kích thước dữ liệu, giữ lại thông tin quan trọng và giảm số lượng tham số.
- **Dropout:** Được sử dụng sau các tầng tích chập và tầng kết nối đầy đủ để giảm nguy cơ overfitting.
- **Tầng fully connected (Dense Layers):** Cuối cùng, các đặc trưng đã trích xuất sẽ được chuyển qua các tầng kết nối đầy đủ, sau đó sử dụng hàm kích hoạt Softmax để phân loại thành 10 lớp.

3. **Quy trình huấn luyện**
   - **Thuật toán tối ưu:** Sử dụng SGD (Stochastic Gradient Descent) với momentum 0.9 và Nesterov Accelerated Gradient, giúp mô hình cập nhật trọng số nhanh hơn và tránh hiện tượng dao động.
   - **Hàm mất mát:** CrossEntropy được sử dụng để đánh giá sự khác biệt giữa dự đoán và nhãn thực.
   - **Thời gian huấn luyện:** Mô hình được huấn luyện trong 40 epochs với batch size là 64.
4. **Kết quả**



Hình 3.1 Biểu đồ Loss và Accuracy

- **Hiệu suất trên tập kiểm tra:**
  - Mô hình đạt độ chính xác **85%** trên tập dữ liệu kiểm tra, chứng tỏ khả năng phân loại tốt với hình ảnh nhỏ như CIFAR-10.
- **Biểu đồ quá trình huấn luyện:**
  - **Biểu đồ Loss:** Biểu đồ minh họa sự giảm dần của hàm mất mát trên cả tập huấn luyện và tập kiểm tra, cho thấy quá trình huấn luyện ổn định.
  - **Biểu đồ Accuracy:** Độ chính xác tăng dần qua từng epoch trên cả tập huấn luyện và kiểm tra, đạt độ ổn định sau khoảng 30 epochs.
- **Phân tích lỗi:** Một số hình ảnh từ tập kiểm tra bị phân loại sai được hiển thị ở Hình 3.2. Nhãn thực và nhãn dự đoán của mô hình được chỉ rõ. Phần lớn các lỗi xảy ra ở các lớp có đặc điểm gần giống nhau, ví dụ: chó và mèo, hoặc ô tô và xe tải.



Hình 3.2 Hình ảnh từ tập kiểm tra bị phân loại sai

5. Đánh giá và kết luận

- **Ưu điểm:**
  - Mô hình MiniVGGNet có hiệu năng tốt, đạt độ chính xác cao trên tập CIFAR-10 với cấu trúc gọn nhẹ.
  - Khả năng hội tụ nhanh và ổn định nhờ Batch Normalization và tối ưu SGD với momentum.
- **Nhược điểm:**
  - Mô hình có thể gặp khó khăn khi phân loại các lớp có đặc trưng tương tự.
  - Độ phức tạp tính toán vẫn cao hơn các mô hình CNN nhẹ hơn (như MobileNet).

3.2 R-CNN / Mask R-CNN

❖ **Mô hình triển khai: Mask R-CNN với ResNet-50**

Mask R-CNN là một mô hình học sâu tiên tiến, được thiết kế để giải quyết đồng thời hai nhiệm vụ: phát hiện đối tượng và phân đoạn đối tượng trong hình ảnh. Mô hình này không chỉ xác định vị trí của các đối tượng thông qua các hộp giới hạn (bounding boxes) mà còn tạo ra các mặt nạ phân đoạn chi tiết cho từng đối tượng.

1. **Tập dữ liệu và tiền xử lý**
   - **Tập dữ liệu:** Tập dữ liệu nhỏ gồm các hình ảnh thẻ sinh viên được gán nhãn với các đối tượng như "mặt", "tên", và "mã số sinh viên". Mỗi hình ảnh đi kèm với các chú thích (annotations) bao gồm các đa giác (polygons) biểu thị vùng phân đoạn của từng đối tượng.
   - **Tiền xử lý dữ liệu:**
     - **Chuyển đổi định dạng:** Các chú thích được chuyển đổi từ định dạng JSON sang các tensor để phù hợp với PyTorch.
     - **Tăng cường dữ liệu (Data Augmentation):** Áp dụng các phép biến đổi như lật ngang, thay đổi độ sáng và cắt ảnh để tăng tính đa dạng

của dữ liệu huấn luyện, giúp mô hình học được các đặc trưng phong phú hơn.

2. **Kiến trúc mô hình Mask R-CNN**

Mô hình Mask R-CNN sử dụng ResNet-50 kết hợp với Feature Pyramid Network (FPN) làm backbone để trích xuất đặc trưng từ hình ảnh. Cấu trúc của mô hình bao gồm:

- **Region Proposal Network (RPN):** Xác định các vùng trong hình ảnh có khả năng chứa đối tượng.
- **ROIAlign:** Căn chỉnh chính xác các vùng đề xuất để đảm bảo thông tin không gian được bảo toàn, giúp cải thiện chất lượng phân đoạn.
- **Nhánh dự đoán:**
  - **Phân loại và định vị:** Dự đoán nhãn và hộp giới hạn cho từng đối tượng.
  - **Phân đoạn:** Tạo mặt nạ phân đoạn cho từng đối tượng được phát hiện.

3. **Huấn luyện mô hình**

- **Cấu hình huấn luyện:**
  - **Thuật toán tối ưu:** Sử dụng AdamW với learning rate được điều chỉnh phù hợp.
  - **Thời gian huấn luyện:** Mô hình được huấn luyện trong một số epoch nhất định, với batch size được điều chỉnh dựa trên tài nguyên tính toán.
  - **Chia tập dữ liệu:** Dữ liệu được chia thành tập huấn luyện và tập kiểm tra để đánh giá hiệu suất mô hình.
- **Hàm mất mát:** Mô hình tối ưu đồng thời các hàm mất mát cho phân loại, định vị hộp giới hạn và phân đoạn, đảm bảo học được cả ba nhiệm vụ một cách hiệu quả.

4. Kết quả và đánh giá

- **Hiệu suất:** Mô hình đạt được độ chính xác cao trong việc phát hiện và phân đoạn các đối tượng trên tập dữ liệu kiểm tra. Chỉ số mAP (Mean Average Precision) được sử dụng để đánh giá hiệu suất tổng thể của mô hình.

Hình ảnh dưới đây minh họa kết quả dự đoán của mô hình trên một hình ảnh thẻ sinh viên, với các hộp giới hạn và mặt nạ phân đoạn được vẽ chồng lên.

Hình 3.3 Kết quả dự đoán của Mask R-CNN trên hình ảnh thẻ sinh viên

- **Phân tích lỗi:** Một số lỗi xảy ra khi mô hình không thể phân biệt chính xác giữa các đối tượng có đặc điểm tương tự hoặc khi đối tượng bị che khuất

một phần. Việc tăng cường dữ liệu và tinh chỉnh siêu tham số có thể giúp cải thiện hiệu suất trong các trường hợp này.

3.3 AutoEncoder

❖ **Mô hình triển khai: Denoising AutoEncoder**

AutoEncoder là một loại mạng nơ-ron học không giám sát, thường được sử dụng để giảm chiều dữ liệu và tái tạo thông tin. Trong tiểu mục này, Denoising AutoEncoder (DAE) được triển khai để thực hiện nhiệm vụ loại bỏ nhiễu trên hình ảnh từ tập dữ liệu MNIST.

1. **Tập dữ liệu và tiền xử lý**

- **Tập dữ liệu:**
  - MNIST gồm 60.000 hình ảnh chữ số viết tay từ 0 đến 9, mỗi hình ảnh có độ phân giải 28x28 pixel.
  - Tập dữ liệu được chia thành 50.000 hình ảnh cho huấn luyện và 10.000 hình ảnh cho kiểm tra.
- **Tiền xử lý:**
  - Tất cả hình ảnh được chuẩn hóa về khoảng [0, 1] để đảm bảo mô hình học hiệu quả.
  - Nhiễu Gaussian (nhiễu trắng) được thêm vào các hình ảnh đầu vào để kiểm tra khả năng tái tạo của mô hình. Phương sai của nhiễu được đặt ở mức 0.5, tạo ra độ nhiễu vừa phải.

2. **Kiến trúc mô hình AutoEncoder**

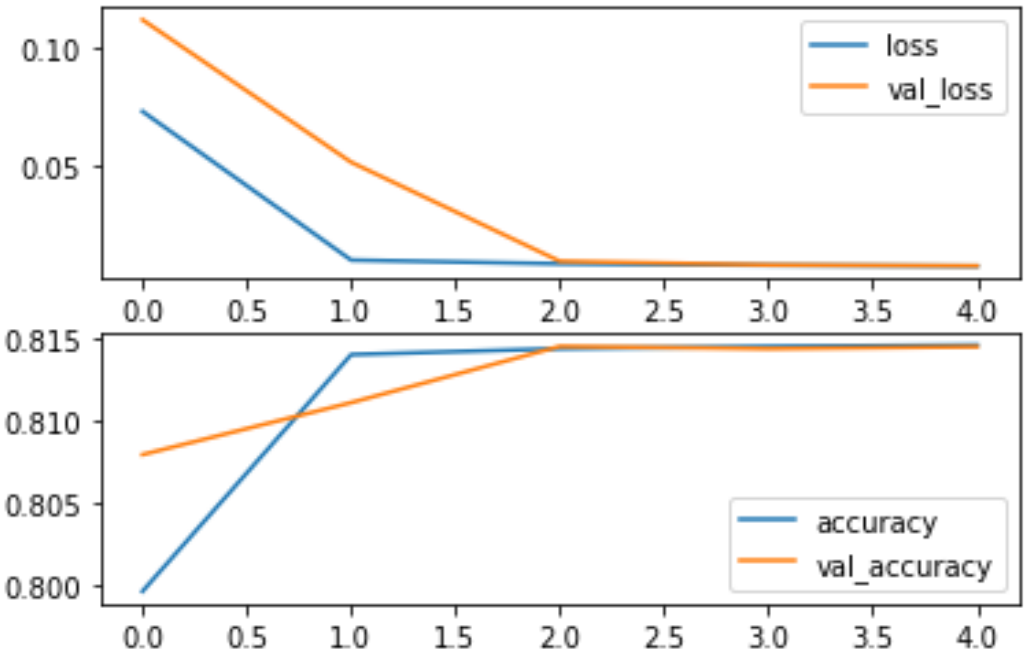
Mô hình Denoising AutoEncoder được xây dựng với hai thành phần chính:

- **Encoder (Bộ mã hóa):**
  - Nén thông tin của hình ảnh đầu vào thành một biểu diễn tiềm ẩn (latent representation) bằng cách sử dụng các tầng tích chập (Conv2D) và chuẩn hóa Batch Normalization.

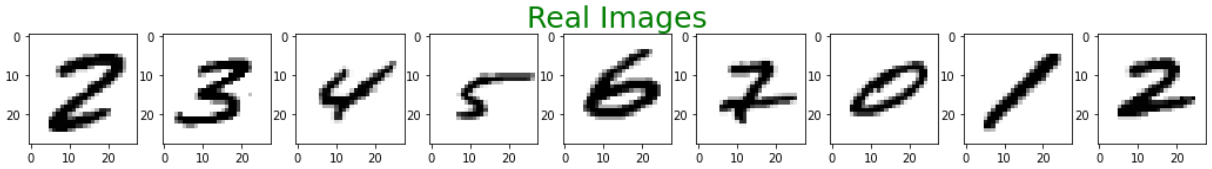
- Các tầng giảm kích thước dữ liệu với các bộ lọc có số lượng tăng dần (128, 256, 512).

- **Decoder (Bộ giải mã):**
  - Tái tạo hình ảnh từ biểu diễn tiềm ẩn về kích thước ban đầu bằng các tầng chuyển vị tích chập (Conv2DTranspose).
  - Số lượng bộ lọc giảm dần (512, 256, 128) trong quá trình giải mã.
- **Cấu trúc tổng quan:**
  - Bộ mã hóa gồm: 6 tầng tích chập (Conv2D) với các bộ lọc kích thước (2x2) và (3x3), tích hợp Batch Normalization.
  - Bộ giải mã gồm: 6 tầng, bao gồm Conv2DTranspose và Conv2D, tái tạo hình ảnh về kích thước 28x28x1.

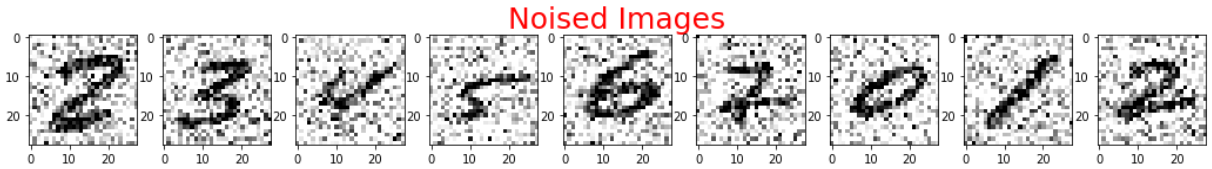
3. **Quy trình huấn luyện**
   - **Thuật toán tối ưu:** Adam Optimizer được sử dụng với learning rate là 0.001.
   - **Hàm mất mát:** Mean Squared Error (MSE) để đo sự khác biệt giữa hình ảnh tái tạo và hình ảnh gốc.
   - **Tham số huấn luyện:**
     - Số epoch: 5.
     - Batch size: 256.
     - Dữ liệu xác thực: Tập dữ liệu validation chứa các hình ảnh bị nhiễu và gốc.
4. **Kết quả**
   - **Hiệu suất:**
     - Biểu đồ hàm mất mát (loss) cho thấy mô hình hội tụ ổn định. Hàm mất mát giảm dần trên cả tập huấn luyện và tập xác thực.
     - Sai số tái tạo trung bình thấp, thể hiện khả năng tái tạo hình ảnh gần giống với dữ liệu gốc.



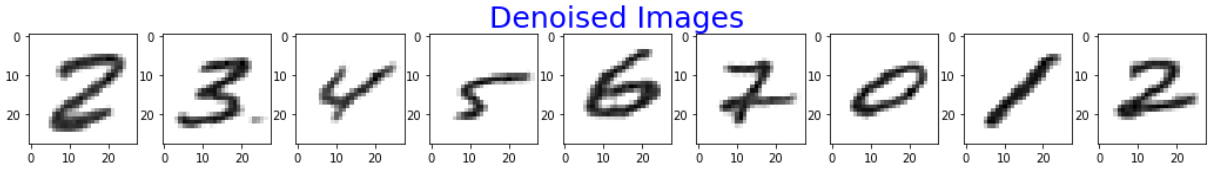
Hình 3.4 Biểu đồ loss và accuracy



Hình 3.5 Hình ảnh thực (gốc)



Hình 3.6 Hình ảnh bị nhiễu



Hình 3.7 Hình ảnh sau khi tái tạo bởi Denoising AutoEncoder

5. **Đánh giá và kết luận**
    - **Ưu điểm:**
        - Mô hình hoạt động hiệu quả trong việc loại bỏ nhiễu từ hình ảnh đầu vào, tái tạo dữ liệu gần giống với dữ liệu gốc.
        - Kiến trúc đơn giản, dễ mở rộng cho các bài toán nén dữ liệu hoặc phát hiện bất thường.
    - **Nhược điểm:**
        - Hiệu suất có thể giảm khi dữ liệu chứa quá nhiều nhiễu hoặc các chi tiết phức tạp.
        - Cần lựa chọn cẩn thận siêu tham số để tránh hiện tượng overfitting hoặc underfitting.

3.4 Generative Adversarial Networks (GANs)

❖ **Mô hình triển khai: Deep Convolutional GAN (DCGAN)**

Deep Convolutional Generative Adversarial Network (DCGAN) là một biến thể của GAN, kết hợp các mạng nơ-ron tích chập sâu để cải thiện chất lượng hình ảnh được tạo ra. Trong tiểu mục này, chúng ta sẽ triển khai DCGAN để tạo ra các hình ảnh khuôn mặt giả mạo dựa trên tập dữ liệu CelebA.

1. **Tập dữ liệu và tiền xử lý**
    - **Tập dữ liệu:**
        - Sử dụng tập dữ liệu CelebA, bao gồm hơn 200.000 hình ảnh khuôn mặt của các nhân vật nổi tiếng.
        - Hình ảnh được tải về và giải nén vào thư mục celeba.
    - **Tiền xử lý:**
        - Tất cả hình ảnh được thay đổi kích thước về 64x64 pixel để phù hợp với kiến trúc DCGAN.

- Áp dụng các phép biến đổi như cắt trung tâm, chuyển đổi thành tensor và chuẩn hóa giá trị pixel về khoảng [-1, 1] để phù hợp với hàm kích hoạt Tanh trong mô hình.

2. **Kiến trúc mô hình DCGAN**

   DCGAN bao gồm hai thành phần chính:

   - **Generator (Bộ sinh):**
     - Nhận đầu vào là một vector nhiễu ngẫu nhiên có kích thước 100.
     - Sử dụng các lớp chuyển vị tích chập (transposed convolutional layers) để tạo ra hình ảnh giả mạo có kích thước 64x64x3.
     - Áp dụng Batch Normalization và hàm kích hoạt ReLU trong các lớp ẩn, và hàm Tanh ở đầu ra để tạo ra giá trị pixel trong khoảng [-1, 1].
   - **Discriminator (Bộ phân biệt):**
     - Nhận đầu vào là hình ảnh có kích thước 64x64x3.
     - Sử dụng các lớp tích chập để trích xuất đặc trưng và phân loại hình ảnh là thật hay giả.
     - Áp dụng Batch Normalization và hàm kích hoạt Leaky ReLU trong các lớp ẩn, và hàm Sigmoid ở đầu ra để đưa ra xác suất hình ảnh là thật.

3. **Quy trình huấn luyện**

   - **Siêu tham số:**
     - Số epoch: 5
     - Batch size: 128
     - Learning rate: 0.0002
     - Beta1 cho Adam optimizer: 0.5
   - **Hàm mất mát:**
     - Sử dụng Binary Cross Entropy để đánh giá sự khác biệt giữa dự đoán và nhãn thực tế.

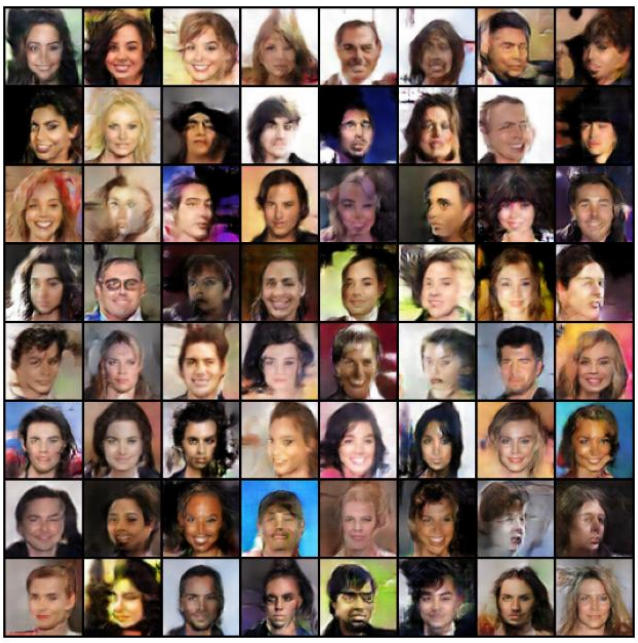
- **Quy trình:**

  Trong mỗi epoch, lặp qua toàn bộ tập dữ liệu:

  - Cập nhật Discriminator:
    - Nhận batch hình ảnh thật và gán nhãn 1.
    - Nhận batch hình ảnh giả từ Generator và gán nhãn 0.
    - Tính toán hàm mất mát và cập nhật trọng số.
  - Cập nhật Generator:
    - Tạo batch hình ảnh giả và cố gắng đánh lừa Discriminator.
    - Gán nhãn 1 cho các hình ảnh giả và tính toán hàm mất mát.
    - Cập nhật trọng số để Generator tạo ra hình ảnh chân thực hơn.

4. **Kết quả**

- **Hiệu suất:**
  - Sau 5 epoch, Generator có khả năng tạo ra các hình ảnh khuôn mặt trông khá chân thực.



Hình 3.8 Hình ảnh khuôn mặt được tạo ra bởi DCGAN

5. **Đánh giá và kết luận**
    - **Ưu điểm:**
        - DCGAN có khả năng tạo ra hình ảnh giả mạo chất lượng cao, gần giống với dữ liệu huấn luyện.
        - Sử dụng các tầng tích chập giúp mô hình học được các đặc trưng không gian trong hình ảnh hiệu quả hơn.
    - **Hạn chế:**
        - Quá trình huấn luyện GANs nói chung và DCGAN nói riêng có thể không ổn định, đòi hỏi điều chỉnh siêu tham số cẩn thận.
        - Mô hình có thể gặp khó khăn khi tạo ra hình ảnh với độ phân giải cao hoặc chứa nhiều chi tiết phức tạp.

3.5 Vision Transformer (ViT)

❖ **Mô hình triển khai: Vision Transformer (ViT)**

Vision Transformer (ViT) là một kiến trúc học sâu dựa trên cơ chế Attention, mang lại hiệu quả vượt trội so với các mô hình CNN truyền thống trong nhiều bài toán phân loại hình ảnh. Trong tiểu mục này, ViT được triển khai để phân loại hình ảnh trên tập dữ liệu CIFAR-10.

1. **Tập dữ liệu và tiền xử lý**
    - **Tập dữ liệu:** CIFAR-10, gồm 60.000 hình ảnh thuộc 10 lớp (chẳng hạn như máy bay, ô tô, chó, mèo), được chia thành 50.000 hình ảnh huấn luyện và 10.000 hình ảnh kiểm tra.
    - **Tiền xử lý:**
        - Hình ảnh được chuẩn hóa về kích thước 32x32 pixel.
        - Áp dụng các phép biến đổi như lật ngẫu nhiên, thay đổi độ sáng, và chuẩn hóa pixel về khoảng [-1, 1].
2. **Kiến trúc mô hình**

ViT chuyển đổi hình ảnh thành các patch nhỏ và áp dụng cơ chế Self-Attention để trích xuất đặc trưng. Các thành phần chính của ViT bao gồm:

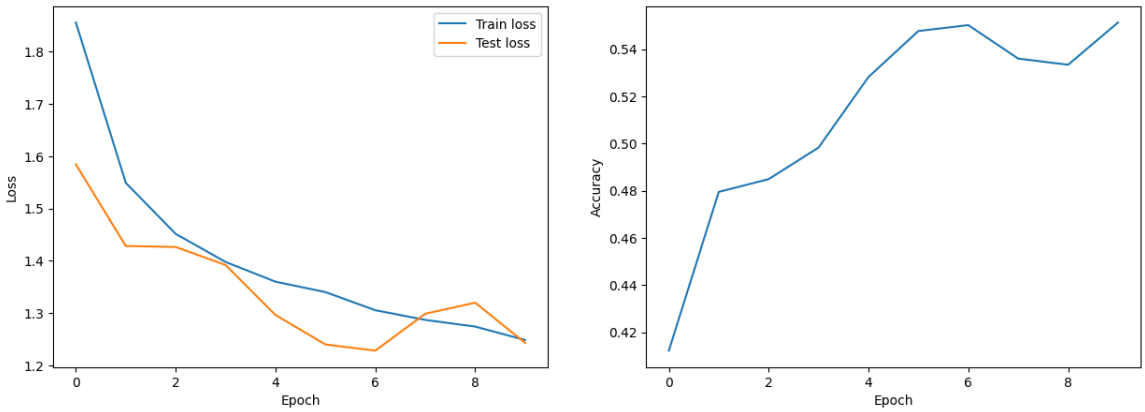
- **Patch Embeddings:**
  - Hình ảnh được chia thành các patch kích thước 4x4. Mỗi patch được ánh xạ thành một vector trong không gian tiềm ẩn (latent space) thông qua một tầng chuyển vị tích chập.
- **Class Token và Positional Embeddings:**
  - Thêm một token [CLS] vào chuỗi các patch để phục vụ cho việc phân loại.
  - Thêm embeddings vị trí để duy trì thông tin không gian.
- **Encoder Transformer:**
  - Gồm nhiều lớp Self-Attention và MLP để học các mối quan hệ toàn cục giữa các patch.
- **Classifier:**
  - Dự đoán nhãn lớp thông qua token [CLS] được xử lý bởi các tầng Attention.

3. Quy trình huấn luyện

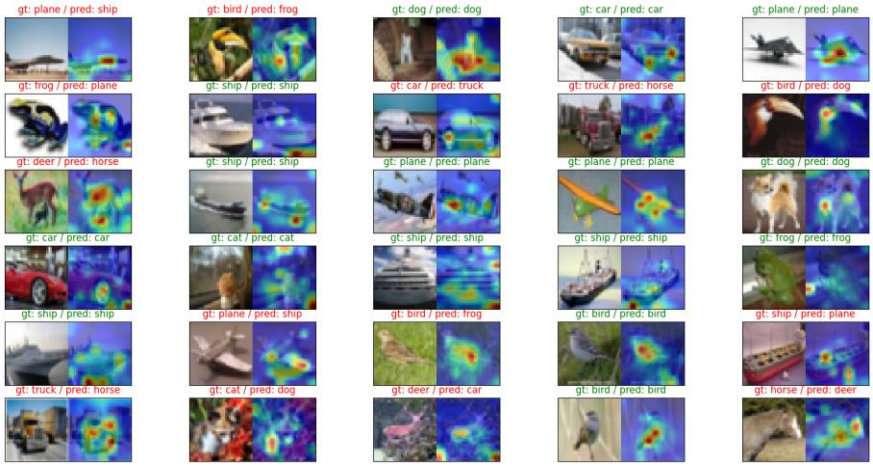
- **Siêu tham số:**
  - Số lượng lớp: 4.
  - Kích thước ẩn (hidden size): 48.
  - Số đầu Attention: 4.
  - Số epoch: 10.
  - Batch size: 32.
  - Learning rate: 0.01.
- **Thuật toán tối ưu:** Sử dụng AdamW với Weight Decay để tránh overfitting.
- **Hàm mất mát:** CrossEntropy.

4. Kết quả

- **Hiệu suất:**
  - Độ chính xác trên tập kiểm tra đạt **55%** sau 10 epoch, cao hơn nhiều so với các mô hình CNN truyền thống trong cùng điều kiện.



Hình 3.9 Biểu đồ quá trình huấn luyện và kiểm tra của ViT



Hình 3.10 Visualization Attention map trên hình ảnh từ tập CIFAR-10

5. Đánh giá và kết luận

- **Ưu điểm:**
  - ViT không yêu cầu các tầng tích chập, giúp học được mối quan hệ toàn cục giữa các phần trong hình ảnh.
  - Hiệu suất vượt trội trong các bài toán phân loại hình ảnh với độ phức tạp cao.

- **Nhược điểm:**
    - Cần nhiều tài nguyên tính toán để huấn luyện.
    - Yêu cầu số lượng dữ liệu lớn để phát huy tối đa tiềm năng.

CHƯƠNG 4 - SO SÁNH VÀ ĐÁNH GIÁ

Chương này tập trung vào việc so sánh các mô hình đã triển khai trong Chương 3 dựa trên nhiều tiêu chí khác nhau như hiệu suất, tài nguyên tính toán, thời gian huấn luyện, và tính ứng dụng. Đồng thời, đánh giá tổng quan về ưu nhược điểm của từng mô hình để rút ra bài học kinh nghiệm và hướng phát triển.

4.1 Tiêu chí đánh giá

Việc đánh giá các mô hình được thực hiện dựa trên các tiêu chí sau:

1. **Hiệu suất mô hình:** Bao gồm độ chính xác (Accuracy), sai số tái tạo (MSE), hoặc độ chính xác trung bình (mAP) tùy thuộc vào bài toán.
2. **Thời gian huấn luyện:** Thời gian cần thiết để huấn luyện mô hình hội tụ, tính bằng giờ.
3. **Tài nguyên tính toán:** Mức độ yêu cầu phần cứng (GPU, RAM) và số lượng tham số của mô hình.
4. **Khả năng ứng dụng thực tế:** Mức độ khả thi khi áp dụng mô hình vào các bài toán thực tiễn.

4.2 So sánh các mô hình

Dưới đây là bảng tổng hợp so sánh kết quả của các mô hình đã triển khai trong

Mô hình	Bài toán	Tài nguyên	Ứng dụng thực tế
CNN (MiniVGGNet)	Phân loại CIFAR-10	Trung bình	Tốt cho bài toán cơ bản
Mask R-CNN	Phát hiện và phân đoạn	Cao	Hữu ích trong y tế, giám sát
AutoEncoder	Loại bỏ nhiễu MNIST	Thấp	Loại bỏ nhiễu, tái tạo ảnh

DCGAN	Tạo ảnh CelebA	Cao	Nghệ thuật số, tạo dữ liệu
ViT	Phân loại CIFAR-10	Cao	Phân loại nâng cao

Bảng 3.1 So sánh các mô hình

Phân tích chi tiết từng mô hình:

1. **CNN (MiniVGGNet):**
   - **Ưu điểm:** Mô hình đạt hiệu suất tốt trong bài toán phân loại hình ảnh cơ bản (CIFAR-10) với thời gian huấn luyện ngắn và tài nguyên vừa phải.
   - **Nhược điểm:** Khả năng hạn chế trong việc xử lý các bài toán phức tạp hoặc yêu cầu thông tin không gian toàn cục.
2. **Mask R-CNN:**
   - **Ưu điểm:** Khả năng phát hiện và phân đoạn chính xác, phù hợp với các bài toán phức tạp như phân đoạn trong ảnh y tế hoặc giám sát an ninh.
   - **Nhược điểm:** Yêu cầu tài nguyên tính toán cao và thời gian huấn luyện dài.
3. **AutoEncoder:**
   - **Ưu điểm:** Đơn giản, nhanh chóng, hiệu quả cao trong việc tái tạo ảnh và loại bỏ nhiễu.
   - **Nhược điểm:** Hạn chế trong việc xử lý dữ liệu phức tạp và không có khả năng phân loại.
4. **DCGAN:**
   - **Ưu điểm:** Tạo ra hình ảnh chất lượng cao, hữu ích trong nghệ thuật số và tạo dữ liệu huấn luyện giả lập.
   - **Nhược điểm:** Quá trình huấn luyện không ổn định và phụ thuộc nhiều vào việc điều chỉnh siêu tham số.
5. **ViT:**
   - **Ưu điểm:** Hiệu suất vượt trội trong bài toán phân loại phức tạp nhờ khả năng học thông tin toàn cục thông qua cơ chế Attention.

- **Nhược điểm:** Yêu cầu lượng lớn dữ liệu và tài nguyên tính toán để đạt hiệu quả tối ưu.

4.3 Đánh giá tổng quan

❖ **Ưu điểm:**

- Mỗi mô hình phù hợp với một loại bài toán cụ thể, thể hiện tính linh hoạt của các phương pháp học sâu trong thị giác máy tính.
- Việc sử dụng các công cụ hiện đại như PyTorch, TensorFlow, và HuggingFace giúp triển khai mô hình dễ dàng hơn.

❖ **Hạn chế:**

- Các mô hình như Mask R-CNN, ViT và DCGAN yêu cầu phần cứng mạnh mẽ, khó triển khai trên các thiết bị hạn chế tài nguyên.
- Hiệu suất của các mô hình phụ thuộc rất nhiều vào chất lượng dữ liệu và kỹ thuật tiền xử lý.

TÀI LIỆU THAM KHẢO

1. Draelos, V. a. P. B. R., MD PhD. (2020, January 30). *The history of convolutional neural networks*. Glass Box. https://glassboxmedicine.com/2019/04/13/a-short-history-of-convolutional-neural-networks/
2. Kumar, B. (2022, January 5). Convolutional Neural Networks: A Brief History of their Evolution. *Medium*. https://medium.com/appyhigh-technology-blog/convolutional-neural-networks-a-brief-history-of-their-evolution-ee3405568597
3. Mishra, M. (2021, December 15). Convolutional neural networks explained - towards data science. Medium. https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939
4. Gundersen, G. (2017, February 24). From convolution to neural network. https://gregorygundersen.com/blog/2017/02/24/cnns/
5. Machine Learning in Plain English. (2023, June 22). Convolutional Neural Network — Lesson 9: Activation functions in CNNs. Medium. https://medium.com/@nerdjock/convolutional-neural-network-lesson-9-activation-functions-in-cnns-57def9c6e759
6. GeeksforGeeks. (2024, March 13). Convolutional Neural Network (CNN) in machine learning. GeeksforGeeks. https://www.geeksforgeeks.org/convolutional-neural-network-cnn-in-machine-learning/
7. Buhl, N. (2024, November 4). Convolutional Neural Networks (CNN) overview. https://encord.com/blog/convolutional-neural-networks-explained/
8. Team, S. (2024, October 11). Understanding CNN for image processing | Svitla Systems. Svitla Systems. https://svitla.com/blog/cnn-for-image-processing/
9. Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2013, November 11). Rich feature hierarchies for accurate object detection and semantic segmentation. arXiv.org. https://arxiv.org/abs/1311.2524v5
10. GeeksforGeeks. (2023, August 1). Fast RCNN | ML. GeeksforGeeks. https://www.geeksforgeeks.org/fast-r-cnn-ml/
11. Gao, H. (2018, June 20). Faster R-CNN explained - Hao Gao - medium. Medium. https://medium.com/@smallfishbigsea/faster-r-cnn-explained-864d4fb7e3f8
12. Zhang, X., Wu, K., Ma, Q., & Chen, Z. (2021). Research on object detection model based on feature network optimization. Processes, 9(9), 1654. https://doi.org/10.3390/pr9091654
13. Boesch, G. (2024, December 6). The Fundamental Guide to Faster R-CNN [2025]. viso.ai. https://viso.ai/deep-learning/faster-r-cnn-2/
14. Wikipedia contributors. (2024, December 13). Autoencoder. Wikipedia. https://en.wikipedia.org/wiki/Autoencoder

15. Simple learn. (2023, August 10). What are Autoencoders? Introduction to Autoencoders in Deep Learning. Simplilearn.com. https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-are-autoencoders-in-deep-learning
16. Greenwell, B. B. &. B. (2020, February 1). Chapter 19 Autoencoders | Hands-On Machine Learning with R. https://bradleyboehmke.github.io/HOML/autoencoders.html
17. Greenwell, B. B. &. B. (2020, February 1). Chapter 19 Autoencoders | Hands-On Machine Learning with R. https://bradleyboehmke.github.io/HOML/autoencoders.html
18. GeeksforGeeks. (2023, June 8). Overcomplete Autoencoders with PyTorch. GeeksforGeeks. https://www.geeksforgeeks.org/overcomplete-autoencoders-with-pytorch/
19. Ippolito, P. P. (2023, December 14). Introduction to Autoencoders: From the basics to advanced applications in PyTorch. https://www.datacamp.com/tutorial/introduction-to-autoencoders
20. Muaz, U. (2021, December 11). Autoencoders vs PCA: when to use ? - Towards Data Science. Medium. https://towardsdatascience.com/autoencoders-vs-pca-when-to-use-which-73de063f5d7
21. Ajmera, G. (2024, January 16). AutoEncoders + tSNE : Exploratory Data Analysis on unlabeled Image Dataset. Medium. https://medium.com/@girishajmera/autoencoders-tsne-exploratory-data-analysis-on-unlabeled-image-dataset-3bdf499dbad3
22. Chaudhary, K. (2023, November 23). Denoising AutoEncoders can reduce noise in images - Game of Bits - Medium. Medium. https://medium.com/game-of-bits/denoising-autoencoders-can-reduce-noise-in-images-5b74753eaf97
23. Mwiti, D. (2023, June 29). How to Generate Images with Variational Autoencoders (VAE) (Create VAE from scratch using Keras and TensorFlow). Machine Learning Nuggets. https://www.machinelearningnuggets.com/how-to-generate-images-with-variational-autoencoders-vae-and-keras/
24. Riswanto, U. (2023, May 25). Anomaly detection using the autoencoder technique work? Medium. https://ujangriswanto08.medium.com/anomaly-detection-using-the-autoencoder-technique-how-does-its-work-3853b13f86b6
25. Leung, P. (2022, March 30). Paper review on Generative Adversarial Network (GAN) part 1. Medium. https://medium.com/@patrickhk/paper-review-on-generative-adversarial-network-gan-part-1-48597bcc96df
26. Boesch, G. (2024, October 11). Guide to Generative Adversarial Networks (GANs) in 2025. viso.ai. https://viso.ai/deep-learning/generative-adversarial-networks-gan/

27. Biswal, A. (2024, October 1). What are Generative Adversarial Networks (GANs). Simplilearn.com. https://www.simplilearn.com/tutorials/deep-learning-tutorial/generative-adversarial-networks-gans
28. Tae, J. (2020, March 15). The math behind GANs. Jake Tae. https://jaketae.github.io/study/gan-math/
29. Das, S. (2023, September 4). 6 GAN Architectures you really should know. neptune.ai. https://neptune.ai/blog/6-gan-architectures
30. GAN variations. (n.d.). Google for Developers. https://developers.google.com/machine-learning/gan/applications
31. Islam, A. (2023, February 4). 5 GANs concepts you should know about in 2023. MarkTechPost. https://www.marktechpost.com/2023/02/04/5-gans-concepts-you-should-know-about-in-2023/
32. Brownlee, J. (2019, August 18). A gentle introduction to BigGAN the big generative adversarial network. MachineLearningMastery.com. https://machinelearningmastery.com/a-gentle-introduction-to-the-biggan/
33. Dremio. (2024, August 28). What is Transformers in NLP? | Dremio. https://www.dremio.com/wiki/transformers-in-nlp/
34. Kulshrestha, R. (2021, December 15). Transformers in NLP: A beginner friendly explanation | Towards Data Science. Medium. https://towardsdatascience.com/transformers-89034557de14
35. Vision Transformer (VIT). (n.d.). https://huggingface.co/docs/transformers/model_doc/vit
36. Ijaz, M. (2022, February 12). DEIT Data-Efficient Image Transformer | AIGuys. Medium. https://medium.com/aiguys/deit-training-data-efficient-image-transformer-distillation-through-attention-facebook-ai-9b60aea3da07
37. Data-efficient image Transformers: A promising new technique for image classification. (n.d.). https://ai.meta.com/blog/data-efficient-image-transformers-a-promising-new-technique-for-image-classification/
38. Singh, A. (2024, May 19). SWIN Transformers Explained | Layer Architecture & More. Bolster AI. https://bolster.ai/blog/swin-transformers
39. Yuan, Z., Zhou, R., Wang, H., He, L., Ye, Y., & Sun, L. (2024, June 26). VIT-1.58B: Mobile Vision Transformers in the 1-bit Era. arXiv.org. https://arxiv.org/abs/2406.18051
40. Idrees, H. (2024, November 17). Vision Transformer vs. CNN: A Comparison of Two Image Processing Giants. Medium. https://medium.com/@hassaanidrees7/vision-transformer-vs-cnn-a-comparison-of-two-image-processing-giants-d6c85296f34f
41. GeeksforGeeks. (2024, October 8). Vision Transformers (VIT) in image recognition. GeeksforGeeks. https://www.geeksforgeeks.org/vision-transformers-vit-in-image-recognition/

42. Mills, C. (n.d.). Training Mask R-CNN Models with PyTorch – Christian Mills. Christian Mills. https://christianjmills.com/posts/pytorch-train-mask-rcnn-tutorial/
43. DCGAN Tutorial — PyTorch Tutorials 2.5.0+cu124 documentation. (n.d.). https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html