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

KHOA HỌC MÁY TÍNH

BÁO CÁO GIỮA KỲ MÔN THỊ GIÁC MÁY TÍNH

Người hướng dẫn: **TS PHẠM VĂN HUY**

Người thực hiện: **NGUYỄN THỊ BẢO TRÂN – 52200089**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2025
TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

KHOA HỌC MÁY TÍNH

BÁO CÁO GIỮA KỲ MÔN THỊ GIÁC MÁY TÍNH

Người hướng dẫn: **TS PHẠM VĂN HUY**

Người thực hiện: **NGUYỄN THỊ BẢO TRÂN – 52200089**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2025

LỜI CẢM ƠN

Nhóm chúng em xin bày tỏ lòng biết ơn chân thành và sâu sắc đến thầy vì đã tận tâm hướng dẫn, truyền đạt kiến thức nền tảng và ứng dụng trong lĩnh vực học máy. Những bài giảng sinh động cùng những chia sẻ kinh nghiệm quý báu của thầy đã tạo cảm hứng và định hướng cho chúng em trong việc nghiên cứu, thực hành các kỹ thuật thị giác máy tính từ cơ bản đến nâng cao.

Quá trình học tập và thực hiện đồ án đã giúp chúng em hiểu sâu hơn về các phương pháp phân tích văn bản, mô hình học máy và ứng dụng deep learning trong thị giác máy tính. Đây là nền tảng quan trọng cho chúng em trong việc tiếp cận các vấn đề thực tiễn các ứng dụng học máy khác trong tương lai.

Do kiến thức còn hạn chế, báo cáo của chúng em không tránh khỏi thiếu sót. Chúng em rất mong nhận được góp ý từ thầy để hoàn thiện hơn trong quá trình học tập và nghiên cứu.

Một lần nữa, nhóm chúng em xin chân thành cảm ơn sự hướng dẫn tận tình của thầy cùng toàn thể giảng viên Khoa đã đồng hành cùng chúng em trong suốt thời gian qua. Chúng em xin chân thành cảm ơn ạ!

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

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

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

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

TP. Hồ Chí Minh, ngày 02 tháng 11 năm 2025

Tác giả

(ký tên và ghi rõ họ tên)

Nguyễn Thị Bảo Trân

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 giữa kỳ môn Thị giác máy tính tập trung xây dựng và đánh giá chương trình demo cho 5 mô hình học sâu gồm: **CNN**, **Faster R-CNN**, **Mask R-CNN**, **Denoising Autoencoder** và **GAN**. Thực nghiệm được triển khai trên môi trường **Kaggle GPU (Tesla P100 16GB, PyTorch 2.8.0 + CUDA 12.6.**

Với bộ dữ liệu **PennFudanPed**, báo cáo đánh giá ba hướng tiếp cận: (1) **CNN** phân loại nhị phân ảnh crop 64×64 (có người/không người) đạt **Accuracy 0.969** trên demo và **0.913** trên validation; (2) **Faster R-CNN** phát hiện người (bounding box) đạt **IoU 0.906** trên demo, trên validation có **IoU(mean) 0.843**, **Precision 0.961**, **Recall 1.000**; (3) **Mask R-CNN** phân đoạn instance đạt **Mask IoU 0.858** trên demo và **Mask IoU(mean) 0.878** trên validation. Về chi phí mô hình, số tham số lần lượt khoảng **11.18M (CNN)**, **41.08M (Faster R-CNN)** và **43.70M (Mask R-CNN)**.

Với bộ dữ liệu **Thumbnails**, báo cáo so sánh khả năng **khử nhiễu ảnh** giữa Autoencoder và GAN dưới nhiễu Gaussian (σ=0.05). Kết quả cho thấy **Autoencoder** cho chất lượng tái tạo tốt hơn (**PSNR ~29.8–29.9 dB, MSE ~0.0010**) so với **GAN** (**PSNR ~24.0–24.1 dB, MSE ~0.0039–0.0040**), đồng thời GAN có xu hướng xuất hiện artefact khi quá trình đối kháng chưa cân bằng. Nhìn chung, CNN phù hợp cho phân loại nhanh, Faster/Mask R-CNN giải quyết bài toán phức tạp hơn (detection/segmentation), và Autoencoder là lựa chọn ổn định hơn cho denoising trong thiết lập thí nghiệm hiện tại.

MỤC LỤC

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

DANH MỤC HÌNH

DANH MỤC BẢNG

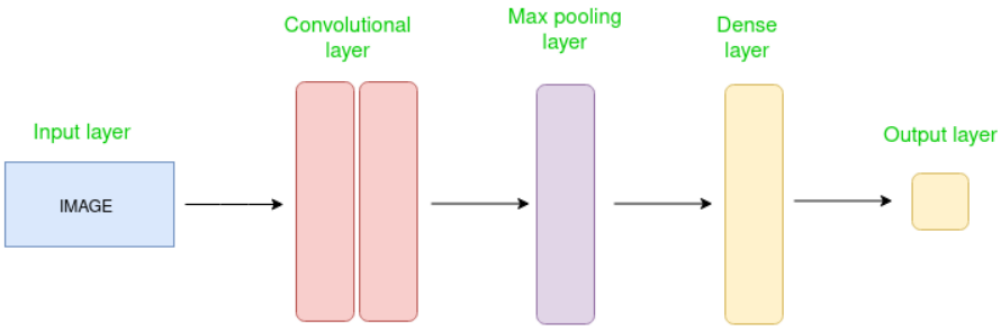
CHƯƠNG 1 – CƠ SỞ LÝ THUYẾT

1.1 Convolutional Neural Network (CNN)

1.1.1 Giới thiệu về CNN

Mạng nơ-ron tích chập (CNN) là phiên bản nâng cao của mạng nơ-ron nhân tạo (ANN) , chủ yếu được thiết kế để trích xuất các đặc điểm từ các tập dữ liệu ma trận dạng lưới. Điều này đặc biệt hữu ích cho các tập dữ liệu trực quan như hình ảnh hoặc video, nơi các mẫu dữ liệu đóng vai trò quan trọng. CNN được sử dụng rộng rãi trong các ứng dụng thị giác máy tính nhờ hiệu quả của chúng trong việc xử lý dữ liệu trực quan.

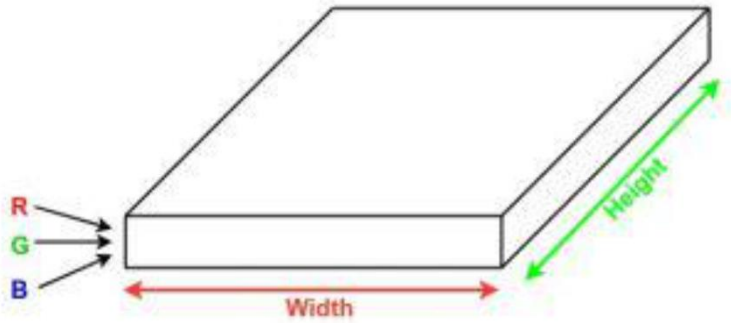
CNN bao gồm nhiều lớp như lớp đầu vào, lớp tích chập, lớp gộp và lớp kết nối đầy đủ.



Hình 1 Kiến trúc CNN

1.1.2 Quy trình hoạt động của CNN

Mạng nơ-ron tích chập là mạng nơ-ron chia sẻ các tham số của chúng. Nó có thể được biểu diễn dưới dạng một hình hộp chữ nhật có chiều dài, chiều rộng (kích thước của hình ảnh) và chiều cao (tức là kênh vì hình ảnh thường có các kênh đỏ, lục và lam).



Hãy tưởng tượng lấy một mảng nhỏ của hình ảnh này và chạy một mạng nơ-ron nhỏ, được gọi là bộ lọc hoặc hạt nhân trên đó, với K đầu ra và biểu diễn chúng theo chiều dọc.

Bây giờ, hãy trượt mạng nơ-ron đó trên toàn bộ hình ảnh, kết quả là chúng ta sẽ có một hình ảnh khác với chiều rộng, chiều cao và chiều sâu khác nhau. Thay vì chỉ có các kênh R, G và B, giờ đây chúng ta có nhiều kênh hơn nhưng chiều rộng và chiều cao lại nhỏ hơn. Phép toán này được gọi là Tích chập . Nếu kích thước miếng vá bằng với kích thước của hình ảnh, đó sẽ là một mạng nơ-ron thông thường. Nhờ miếng vá nhỏ này, chúng ta có ít trọng số hơn.

1.1.3 Các lớp xây dựng Convnet

Kiến trúc mạng nơ-ron tích chập hoàn chỉnh còn được gọi là covnets. Covnets là một chuỗi các lớp, và mỗi lớp biến đổi một thể tích này sang thể tích khác thông qua một hàm khả vi.

Lấy ví dụ bằng cách chạy covnets trên một hình ảnh có kích thước 32 x 32 x 3.

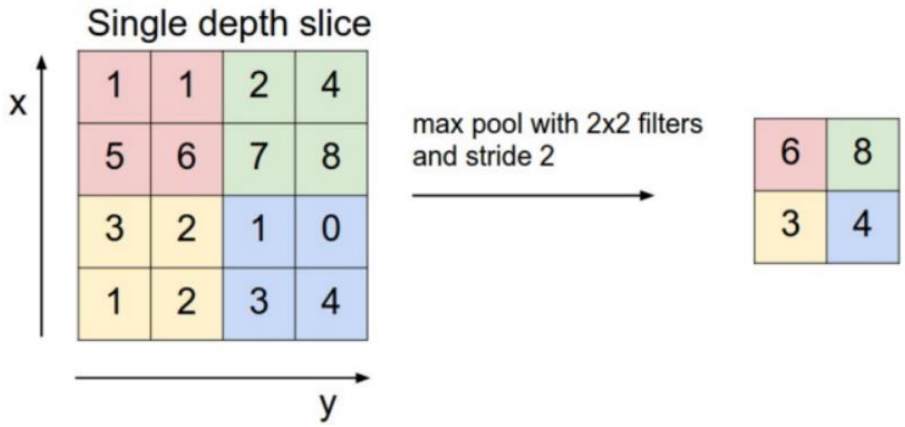
Lớp Đầu vào: Đây là lớp mà chúng ta nhập dữ liệu đầu vào cho mô hình. Trong CNN, đầu vào thường là một hình ảnh hoặc một chuỗi hình ảnh. Lớp này chứa dữ liệu đầu vào thô của hình ảnh với chiều rộng 32, chiều cao 32 và chiều sâu 3.

Lớp Tích chập : Đây là lớp được sử dụng để trích xuất đặc trưng từ tập dữ liệu đầu vào. Lớp này áp dụng một tập hợp các bộ lọc có thể học được, được gọi là hạt nhân

(kernel), cho các ảnh đầu vào. Các bộ lọc/hạt nhân này là các ma trận nhỏ hơn, thường có kích thước 2x2, 3x3 hoặc 5x5. Lớp này trượt trên dữ liệu ảnh đầu vào và tính tích vô hướng giữa trọng số hạt nhân và vùng ảnh đầu vào tương ứng. Đầu ra của lớp này được gọi là bản đồ đặc trưng. Giả sử chúng ta sử dụng tổng cộng 12 bộ lọc cho lớp này, chúng ta sẽ thu được thể tích đầu ra có kích thước 32 x 32 x 12.

Lớp Kích hoạt : Bằng cách thêm một hàm kích hoạt vào đầu ra của lớp trước, các lớp kích hoạt sẽ thêm tính phi tuyến tính vào mạng. Nó sẽ áp dụng một hàm kích hoạt từng phần tử cho đầu ra của lớp tích chập. Một số hàm kích hoạt phổ biến là RELU : max(0, x), Tanh , Leaky RELU , v.v. Thể tích không đổi, do đó thể tích đầu ra sẽ có kích thước 32 x 32 x 12.

Lớp Pooling : Lớp này được chèn định kỳ vào covnets và chức năng chính của nó là giảm kích thước của khối lượng, giúp tính toán nhanh hơn, giảm bộ nhớ và ngăn ngừa hiện tượng overfitting. Hai loại lớp pooling phổ biến là max pooling và average pooling . Nếu chúng ta sử dụng max pooling với bộ lọc 2 x 2 và bước nhảy 2, khối lượng kết quả sẽ có kích thước 16x16x12.



Làm phẳng: Các bản đồ đặc trưng kết quả được làm phẳng thành một vectơ một chiều sau các lớp tích chập và gộp để chúng có thể được chuyển vào một lớp được liên kết hoàn toàn để phân loại hoặc hồi quy.

Lớp kết nối đầy đủ: Lấy dữ liệu đầu vào từ lớp trước và tính toán nhiệm vụ phân loại hoặc hồi quy cuối cùng.

Lớp đầu ra: Đầu ra từ các lớp được kết nối đầy đủ sau đó được đưa vào hàm logistic để phân loại các tác vụ như sigmoid hoặc softmax, hàm này chuyển đổi đầu ra của từng lớp thành điểm xác suất của từng lớp.

1.1.4 Một số biến thể của CNN

1. **LeNet-5**

   LeNet-5 là một trong những mạng CNN đầu tiên được giới thiệu bởi Yann LeCun vào năm 1998, mở đường cho sự phát triển của các mô hình thị giác máy tính sau này. Những đặc điểm chính bao gồm:

   - Kiến trúc đơn giản gồm 7 tầng: 2 tầng tích chập, 2 tầng pooling, và 3 tầng fully-connected.
   - Sử dụng hàm kích hoạt **sigmoid/tanh** và hàm mất mát **cross-entropy**.
   - Được thiết kế cho bài toán nhận dạng chữ số viết tay (MNIST).

   LeNet-5 đặt nền móng cho kiến trúc CNN hiện đại.

2. **AlexNet**

   AlexNet, được phát triển bởi Alex Krizhevsky, Ilya Sutskever và Geoffrey Hinton (2012), đã đánh dấu bước ngoặt trong sự phát triển của học sâu.

   - Sử dụng **hàm kích hoạt ReLU** thay thế sigmoid để giảm hiện tượng biến mất gradient.
   - Tích hợp **Dropout** nhằm tránh overfitting.
   - Tận dụng **GPU song song** để huấn luyện trên tập dữ liệu lớn ImageNet.

AlexNet giúp giảm đáng kể lỗi phân loại và mở ra thời kỳ bùng nổ của Deep Learning.

3. **VGGNet**

   VGGNet được phát triển bởi nhóm nghiên cứu tại Đại học Oxford (Simonyan & Zisserman, 2014), nổi bật bởi kiến trúc đơn giản nhưng sâu hơn.

   - Sử dụng các bộ lọc **3×3** và **pooling 2×2** lặp lại nhiều lần.
   - Mạng gồm 16 hoặc 19 lớp (VGG16/VGG19).
   - Cấu trúc đồng nhất giúp dễ mở rộng và huấn luyện.

   Tuy có số lượng tham số lớn (~138 triệu), VGGNet vẫn được sử dụng rộng rãi trong các ứng dụng trích xuất đặc trưng.

4. **GoogLeNet (Inception Network)**

   Được giới thiệu bởi Szegedy và cộng sự tại Google năm 2014, GoogLeNet là một cải tiến quan trọng trong việc tối ưu hóa số tham số.

   - Sử dụng **Inception module**, kết hợp các phép tích chập kích thước khác nhau (1×1, 3×3, 5×5) song song.
   - Dùng **1×1 convolution** để giảm chiều dữ liệu, giúp tiết kiệm tính toán.
   - Mạng có độ sâu 22 tầng nhưng chỉ khoảng 6,8 triệu tham số.

   GoogLeNet đạt hiệu suất cao hơn trong khi giảm đáng kể chi phí tính toán.

5. **ResNet (Residual Network)**

   ResNet được đề xuất bởi Kaiming He và cộng sự (2015) nhằm giải quyết vấn đề gradient biến mất khi mạng quá sâu.

- Giới thiệu **Residual Block** với kết nối tắt (skip connection) giúp gradient lan truyền dễ dàng hơn.
- Có thể huấn luyện mạng rất sâu (lên đến 152 tầng) mà vẫn hội tụ tốt.
- Đạt kết quả vượt trội trên ImageNet với độ lỗi top-5 chỉ 3.57%.

ResNet trở thành nền tảng cho nhiều mô hình sau này như Mask R-CNN, Faster R-CNN, và các mô hình y học.

6. **DenseNet**

   DenseNet (Gao Huang et al., 2017) mở rộng ý tưởng của ResNet bằng cách kết nối **mọi lớp với tất cả các lớp trước đó**.

   - Mỗi lớp nhận đầu vào là đặc trưng của toàn bộ các lớp trước.
   - Giúp **tái sử dụng đặc trưng**, giảm số tham số và cải thiện gradient.
   - Hiệu quả hơn trong huấn luyện mạng sâu, ít overfitting hơn.

   DenseNet tăng cường khả năng biểu diễn và tối ưu bộ nhớ.

7. **MobileNet**

   MobileNet (Google, 2017) được thiết kế cho các ứng dụng trên thiết bị di động và nhúng, với mục tiêu giảm kích thước và chi phí tính toán.

   - Sử dụng **Depthwise Separable Convolution** để tách riêng tích chập theo không gian và kênh.
   - Các phiên bản nâng cấp (V2, V3) cải thiện độ chính xác và hiệu năng.
   - Tối ưu cho môi trường hạn chế tài nguyên như điện thoại và IoT.

   MobileNet phù hợp cho phát hiện vật thể, nhận dạng khuôn mặt và xe tự hành nhẹ.

8. **EfficientNet**

EfficientNet (Mingxing Tan & Quoc V. Le, 2019) là biến thể CNN hiệu quả nhất hiện nay, được thiết kế bằng **Neural Architecture Search (NAS)**.

- Sử dụng phương pháp **Compound Scaling** để mở rộng đồng đều về độ sâu, chiều rộng và độ phân giải.
- Đạt độ chính xác cao hơn ResNet nhưng với ít tham số hơn nhiều.
- Có nhiều phiên bản (B0–B7) cân bằng giữa hiệu năng và tốc độ.

EfficientNet mang lại sự tối ưu vượt trội cho các ứng dụng thực tế như phân loại ảnh, y tế và Edge AI.

1.1.5 Thách thức của CNN

1. Thiếu dữ liệu huấn luyện chất lượng: CNN cần lượng dữ liệu lớn, đa dạng và được gán nhãn chính xác. Trong thực tế, việc thu thập và gán nhãn tốn kém, dữ liệu dễ nhiễu hoặc mất cân bằng.

2. Chi phí tính toán cao: Các mô hình CNN có hàng chục triệu tham số, đòi hỏi GPU mạnh và thời gian huấn luyện dài, khó triển khai trên thiết bị nhỏ.

3. Dễ overfitting và tổng quát kém: Hiệu suất mô hình thường giảm khi gặp dữ liệu mới trong điều kiện thực tế như ánh sáng hoặc góc chụp khác. Giải pháp là dùng Regularization, Dropout, Batch Normalization và cập nhật dữ liệu liên tục.

4. Khó học mối quan hệ toàn cục: CNN chủ yếu tập trung vào đặc trưng cục bộ, nên khó nắm bắt ngữ cảnh tổng thể của hình ảnh. Có thể kết hợp với Attention hoặc Transformer để cải thiện khả năng hiểu ngữ cảnh.

5**. Khó mở rộng và triển khai thực tế:** Khi áp dụng ở quy mô lớn như camera giám sát hoặc xe tự hành, cần cân bằng giữa tốc độ xử lý, độ chính xác và tài nguyên phần cứng.

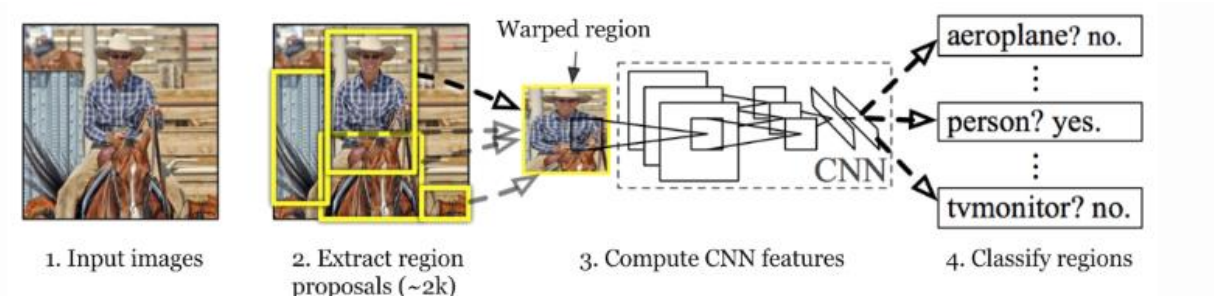
1.2 Region-based Convolutional Neural Networks (R-CNN)

1.2.1 Giới thiệu R-CNN

Mạng nơ-ron tích chập (CNN) truyền thống với các lớp được kết nối đầy đủ thường gặp khó khăn trong việc phát hiện đối tượng, đặc biệt là khi xử lý nhiều đối tượng có kích thước và vị trí khác nhau trong một hình ảnh. Phương pháp brute-force như áp dụng cửa sổ trượt trên ảnh để phát hiện đối tượng rất tốn kém về mặt tính toán, vì nó không thể mở rộng hiệu quả khi tần suất và độ biến thiên của đối tượng tăng lên.

Để vượt qua những thách thức này, R-CNN (Vùng có đặc điểm CNN) đã được giới thiệu. R-CNN mang đến một phương pháp tiếp cận thông minh hơn bằng cách sử dụng thuật toán tìm kiếm chọn lọc để tạo ra khoảng 2.000 đề xuất vùng từ một hình ảnh. Những đề xuất này có khả năng chứa các đối tượng và được xử lý riêng lẻ để phát hiện và định vị chúng hiệu quả hơn. R-CNN đã đánh dấu một bước tiến đáng kể trong lĩnh vực phát hiện đối tượng và đặt nền móng cho các mô hình phát hiện đối tượng nhanh hơn và chính xác hơn.

1.2.2 Quy trình hoạt động của R-CNN



1. **Hình ảnh đầu vào** : Bắt đầu với một hình ảnh đầu vào duy nhất chứa một hoặc nhiều đối tượng.
2. **Tạo đề xuất khu vực** : Sử dụng Tìm kiếm có chọn lọc để tạo khoảng 2.000 đề xuất khu vực (vị trí đối tượng tiềm năng).

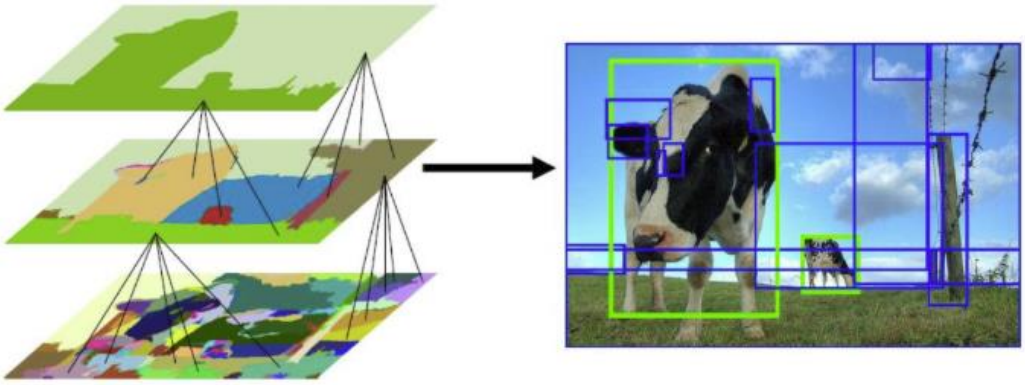
3. **Trích xuất đặc điểm và biến dạng** : Mỗi vùng được đề xuất sẽ được cắt và thay đổi kích thước (biến dạng) thành một kích thước cố định. Sau đó, mỗi vùng sẽ được truyền qua một mạng CNN để trích xuất các vectơ đặc điểm.
4. **Phân loại vùng** : Sử dụng các đặc điểm được trích xuất để phân loại từng vùng bằng SVM thành các loại đối tượng (ví dụ: người, ô tô) hoặc nền.

1.2.3 Phân tích cụ thể các bước thực hiện

1. Region Proposals (Đề xuất khu vực)

R-CNN bắt đầu bằng cách tạo ra các đề xuất vùng, là những phần nhỏ hơn của hình ảnh có thể chứa các đối tượng chúng ta đang tìm kiếm. Thuật toán sử dụng một phương pháp gọi là tìm kiếm chọn lọc, một phương pháp tiếp cận tham lam tạo ra khoảng 2.000 đề xuất vùng trên mỗi hình ảnh. Tìm kiếm chọn lọc cân bằng hiệu quả số lượng đề xuất trong khi vẫn duy trì khả năng thu hồi đối tượng cao, đảm bảo phát hiện đối tượng hiệu quả.

Bằng cách giới hạn số vùng để phân tích chi tiết, phương pháp này nâng cao hiệu suất tổng thể của R-CNN trong việc phát hiện các đối tượng trong hình ảnh.



2. Tìm kiếm có chọn lọc

Tìm kiếm Chọn lọc là một thuật toán tham lam tạo ra các đề xuất vùng bằng cách kết hợp các vùng phân đoạn nhỏ hơn. Thuật toán này lấy một hình ảnh làm đầu vào và

tạo ra các đề xuất vùng rất quan trọng cho việc phát hiện đối tượng. Phương pháp này mang lại những lợi thế đáng kể so với việc tạo đề xuất ngẫu nhiên bằng cách giới hạn số lượng đề xuất ở mức khoảng 2.000 trong khi vẫn đảm bảo độ thu hồi đối tượng cao.

Các bước của thuật toán:

1. **Tạo phân đoạn ban đầu** : Thuật toán bắt đầu bằng cách thực hiện phân đoạn phụ ban đầu của hình ảnh đầu vào.
2. **Kết hợp các vùng tương tự** : Sau đó, thuật toán này kết hợp đệ quy các hộp giới hạn tương tự thành các hộp lớn hơn. Độ tương đồng được đánh giá dựa trên các yếu tố như màu sắc, kết cấu và kích thước vùng.
3. **Tạo đề xuất vùng** : Cuối cùng, các hộp giới hạn lớn hơn này được sử dụng để tạo đề xuất vùng nhằm phát hiện đối tượng.

Thuật toán tìm kiếm có chọn lọc cung cấp một cách hiệu quả để xác định các vùng đối tượng tiềm năng, nâng cao hiệu quả tổng thể của quá trình phát hiện.

3. Trích xuất tính năng

Sau khi các đề xuất vùng được tạo ra, khoảng 2.000 vùng sẽ được trích xuất và biến dạng dị hướng thành kích thước đầu vào nhất quán mà CNN mong đợi (ví dụ: 224x224 pixel), sau đó được truyền qua CNN để trích xuất các đặc điểm.

Trước khi bẻ cong, kích thước vùng được mở rộng đến một kích thước mới, tạo ra 16 pixel ngữ cảnh trong khung bẻ cong. Mạng CNN được sử dụng là AlexNet và thường được tinh chỉnh trên một tập dữ liệu lớn như ImageNet để biểu diễn các đặc điểm chung.

Đầu ra của CNN là một vectơ đặc trưng có chiều cao biểu diễn nội dung của đề xuất vùng.

4. Phân loại đối tượng

Các vectơ đặc trưng được trích xuất từ các đề xuất vùng được đưa vào một bộ phân loại học máy riêng biệt cho từng lớp đối tượng quan tâm. R-CNN thường sử dụng Máy vectơ hỗ trợ (SVM) để phân loại. Đối với mỗi lớp, một SVM duy nhất được huấn luyện để xác định xem đề xuất vùng có chứa một thể hiện của lớp đó hay không.

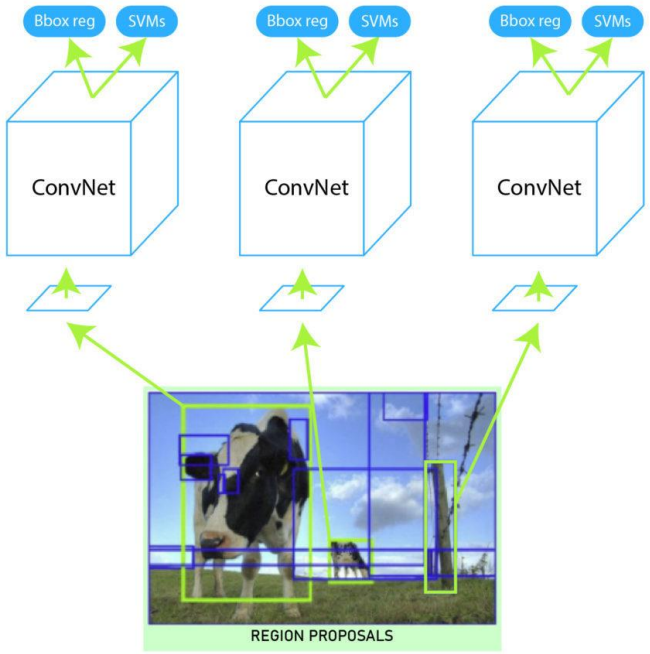
Trong quá trình huấn luyện, các mẫu dương là các vùng chứa một thể hiện của lớp. Các mẫu âm là các vùng không chứa.

5. Bounding Box Regressor (hồi quy hộp giới hạn)

Ngoài việc phân loại đối tượng, R-CNN còn thực hiện hồi quy hộp giới hạn. Với mỗi lớp, một mô hình hồi quy riêng biệt được huấn luyện để tinh chỉnh vị trí và kích thước của hộp giới hạn xung quanh đối tượng được phát hiện. Hồi quy hộp giới hạn giúp cải thiện độ chính xác của việc định vị đối tượng bằng cách điều chỉnh hộp giới hạn được đề xuất ban đầu để phù hợp hơn với ranh giới thực tế của đối tượng.

Để định vị chính xác khung giới hạn trong ảnh, chúng tôi sử dụng mô hình hồi quy tuyến tính bất biến theo tỷ lệ, được gọi là hồi quy khung giới hạn . Để huấn luyện mô hình này, chúng tôi sử dụng các cặp giá trị dự đoán và giá trị thực tế cho bốn chiều định vị (x,y,w,h). Ở đây, x và y biểu diễn tọa độ điểm ảnh của tâm hộp giới

hạn, trong khi w và h chỉ ra chiều rộng và chiều dài tương ứng của các hộp giới hạn. Phương pháp này nâng cao Độ chính xác trung bình (mAP) của kết quả lên 3-4%.

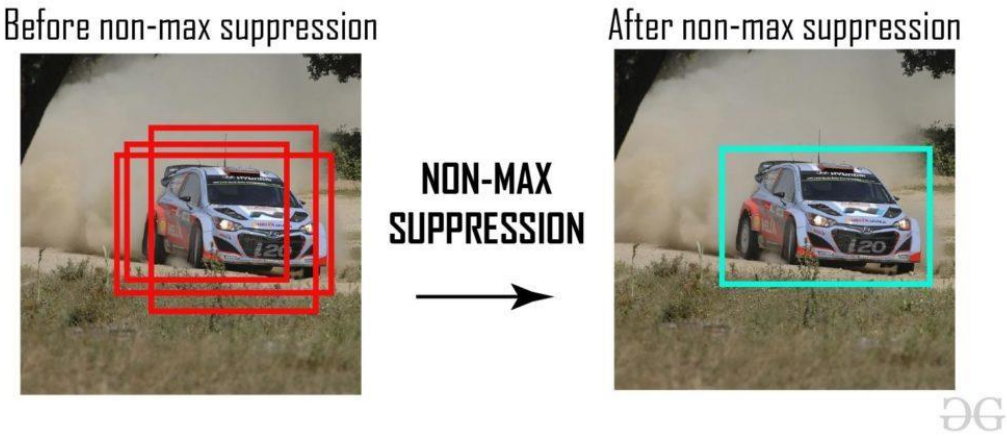


Để tối ưu hóa khả năng phát hiện hơn nữa, R-CNN áp dụng cơ chế ức chế không tối đa (NMS) :

- Loại bỏ các đề xuất có điểm tin cậy dưới ngưỡng (ví dụ: 0,5).
- Chọn vùng có xác suất cao nhất trong số các ứng viên cho mỗi đối tượng.
- Loại bỏ các vùng chồng lấn có IoU (Giao điểm trên hợp) trên 0,5 để loại bỏ các phát hiện trùng lặp, trong đó IoU được định nghĩa là:

$$IoU = \frac{Area\ of\ Overlap}{Area\ if\ Union}$$

Bằng cách kết hợp các đề xuất vùng, tìm kiếm có chọn lọc, trích xuất tính năng dựa trên CNN, phân loại SVM và tinh chỉnh hộp giới hạn, R-CNN đạt được độ chính xác cao trong việc phát hiện đối tượng, khiến nó phù hợp với nhiều ứng dụng khác nhau.



Sau đó, chúng ta có thể thu được kết quả đầu ra bằng cách vẽ các hộp giới hạn này trên hình ảnh đầu vào và dán nhãn các đối tượng có trong các hộp giới hạn.

1.2.4 Một số biến thể của R-CNN

1. Fast R-CNN

Fast R-CNN tối ưu hóa kiến trúc R-CNN bằng cách chia sẻ các phép tính giữa các đề xuất. Những cải tiến chính bao gồm:

- Xử lý một giai đoạn : Thay vì trích xuất các đặc điểm cho từng đề xuất vùng một cách độc lập, Fast R-CNN xử lý toàn bộ hình ảnh một lần thông qua CNN để tạo bản đồ đặc điểm. Các đề xuất vùng sau đó được trích xuất từ bản đồ đặc điểm chung này.
- Bộ phân loại Softmax : Fast R-CNN thay thế SVM bằng bộ phân loại softmax, cho phép đào tạo mạng từ đầu đến cuối.
- Cải thiện hồi quy hộp giới hạn : Fast R-CNN cải thiện quá trình hồi quy hộp giới hạn, dẫn đến độ chính xác định vị tốt hơn.

2. Faster R-CNN

Faster R-CNN tiếp tục cải tiến khuôn khổ R-CNN bằng cách tích hợp Mạng lưới Đề xuất Khu vực (RPN). Các tính năng chính bao gồm:

- **Mạng đề xuất khu vực** : RPN tạo ra các đề xuất khu vực chất lượng cao trực tiếp từ bản đồ đặc điểm do CNN tạo ra, loại bỏ nhu cầu tìm kiếm có chọn lọc.
- **Các tính năng tích chập được chia sẻ** : Cả RPN và mạng phát hiện đều chia sẻ các tính năng tích chập, giúp giảm đáng kể thời gian tính toán.
- **Tốc độ được cải thiện** : R-CNN nhanh hơn đạt tốc độ xử lý thời gian thực khoảng 0,1 giây cho mỗi hình ảnh trong khi vẫn duy trì độ chính xác phát hiện cao.

3.Mask R-CNN

Dựa trên Faster R-CNN, Mask R-CNN được giới thiệu để mở rộng mô hình nhằm thực hiện phân đoạn phiên bản. Các tính năng chính bao gồm:

- **Mặt nạ phân đoạn** : Ngoài các hộp giới hạn, Mask R-CNN còn dự đoán mặt nạ phân đoạn cho mỗi đối tượng được phát hiện, mang lại độ chính xác ở cấp độ pixel.
- **Mạng kim tự tháp tính năng (FPN)** : Mask R-CNN kết hợp FPN để cải thiện hiệu suất trên các đối tượng ở các tỷ lệ khác nhau, tăng cường độ chính xác phát hiện đối với các đối tượng nhỏ.
- **RoIAlign** : Kỹ thuật này thay thế RoIPooling để giải quyết các vấn đề không khớp, đảm bảo trích xuất tính năng tốt hơn cho từng vùng quan tâm.

4. Cascade R-CNN

Cascade R-CNN triển khai một khuôn khổ phát hiện đối tượng đa giai đoạn để cải thiện hiệu suất phát hiện. Các khía cạnh chính bao gồm:

- Phát hiện nhiều giai đoạn : Cascade R-CNN sử dụng một loạt các máy dò hoạt động ở các giai đoạn khác nhau, dần dần tinh chỉnh các đề xuất và cải thiện độ chính xác định vị.
- Cải thiện khả năng thu hồi và độ chính xác : Bằng cách giải quyết sự đánh đổi giữa khả năng thu hồi và độ chính xác ở từng giai đoạn, mô hình nâng cao hiệu suất phát hiện tổng thể, đặc biệt là trên các tập dữ liệu đầy thách thức.

1.3.4 Ứng dụng của R-CNN

Xe tự hành : R-CNN có thể phát hiện và phân loại nhiều vật thể khác nhau trên đường, chẳng hạn như người đi bộ, các phương tiện khác và biển báo giao thông, góp phần điều hướng an toàn hơn.

Hệ thống giám sát : Trong các ứng dụng bảo mật, R-CNN có thể xác định các hoạt động đáng ngờ bằng cách phát hiện và phân loại cá nhân và đối tượng theo thời gian thực.

Chụp ảnh y tế : R-CNN được sử dụng trong các ứng dụng y tế để xác định các bất thường trong hình ảnh chụp y tế, hỗ trợ chẩn đoán và điều trị sớm.

Thực tế tăng cường : R-CNN có thể cho phép nhận dạng đối tượng trong các ứng dụng thực tế tăng cường, nâng cao trải nghiệm của người dùng bằng cách phủ thông tin kỹ thuật số lên thế giới thực.

1.3.5 Điểm mạnh và yếu của R-CNN

1. Điểm mạnh

Phát hiện đối tượng chính xác : R-CNN cung cấp khả năng phát hiện đối tượng chính xác bằng cách tận dụng các đặc điểm tích chập dựa trên vùng. Nó hoạt động hiệu quả trong các tình huống đòi hỏi khả năng định vị và nhận dạng đối tượng chính xác.

Độ bền với các biến thể đối tượng : Mô hình R-CNN có thể xử lý các đối tượng có kích thước, hướng và tỷ lệ khác nhau, giúp chúng phù hợp với các tình huống thực tế có nhiều đối tượng và nền phức tạp.

Tính linh hoạt : R-CNN là một nền tảng đa năng có thể được điều chỉnh cho nhiều tác vụ phát hiện đối tượng khác nhau, bao gồm phân đoạn và theo dõi đối tượng. Bằng cách sửa đổi các lớp cuối cùng của mạng, bạn có thể tùy chỉnh R-CNN cho phù hợp với nhu cầu cụ thể của mình.

2. Điểm yếu

Độ phức tạp tính toán : R-CNN đòi hỏi tính toán chuyên sâu. Nó bao gồm việc trích xuất các đề xuất vùng, áp dụng CNN cho từng đề xuất, sau đó chạy các đặc trưng đã trích xuất thông qua một bộ phân loại. Quy trình nhiều giai đoạn này có thể chậm và tốn kém tài nguyên.

Suy luận chậm : Do xử lý tuần tự các đề xuất vùng, R-CNN tương đối chậm trong quá trình suy luận. Các ứng dụng thời gian thực có thể thấy độ trễ này không thể chấp nhận được.

Đề xuất vùng chồng lấn : R-CNN có thể tạo ra nhiều đề xuất vùng chồng lấn đáng kể, dẫn đến tính toán dư thừa và có khả năng ảnh hưởng đến hiệu suất phát hiện.

R-CNN không phải là mô hình End-to-End : Không giống như các kiến trúc phát hiện đối tượng hiện đại hơn như Faster R-CNN, R-CNN không phải là mô hình End-to-End. Nó bao gồm các mô-đun riêng biệt cho đề xuất vùng và phân loại, điều này

có thể dẫn đến hiệu suất không tối ưu so với các mô hình tối ưu hóa đồng thời cả hai tác vụ.

1.2.5 Thách thức của R-CNN

R-CNN phải đối mặt với một số thách thức trong quá trình triển khai:

Thuật toán Tìm kiếm Chọn lọc Cứng nhắc : Thuật toán tìm kiếm chọn lọc không linh hoạt và không liên quan đến bất kỳ quá trình học nào. Sự cứng nhắc này có thể dẫn đến việc tạo đề xuất vùng kém hiệu quả để phát hiện đối tượng.

Đào tạo tốn thời gian : Với khoảng 2.000 đề xuất ứng viên, việc đào tạo mạng trở nên tốn thời gian. Ngoài ra, nhiều thành phần cần được đào tạo riêng biệt, bao gồm kiến trúc CNN, mô hình SVM và bộ hồi quy hộp giới hạn. Quy trình đào tạo nhiều bước này làm chậm quá trình triển khai.

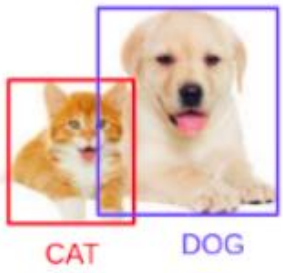
Không hiệu quả đối với các ứng dụng thời gian thực : R-CNN không phù hợp với các ứng dụng thời gian thực vì phải mất khoảng 50 giây để xử lý một hình ảnh duy nhất bằng bộ hồi quy hộp giới hạn.

Tăng yêu cầu về bộ nhớ : Lưu trữ bản đồ đặc điểm cho tất cả các đề xuất khu vực làm tăng đáng kể bộ nhớ đĩa cần thiết trong giai đoạn đào tạo.

1.3 Mask R-CNN

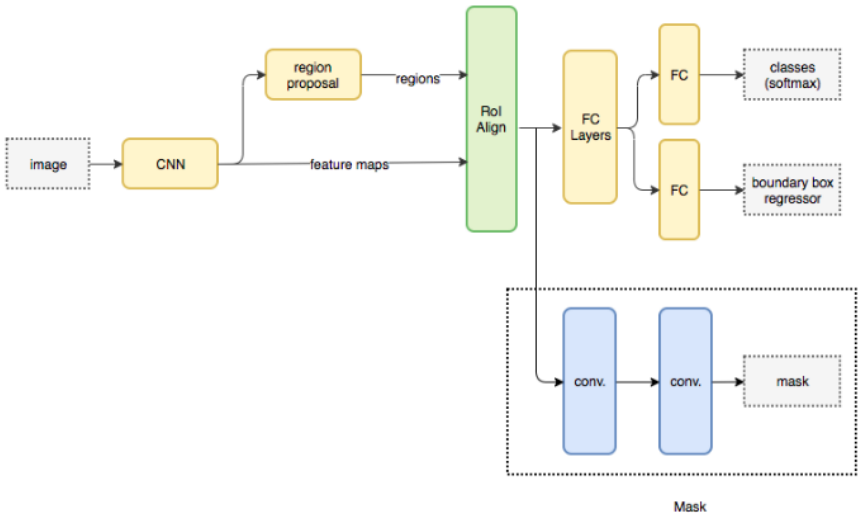
1.3.1 Giới thiệu về Mask R-CNN

Mask R-CNN (Mạng Nơ-ron Tích chập dựa trên Vùng Mặt nạ) là một phần mở rộng của kiến trúc Faster R-CNN, bổ sung một nhánh để dự đoán mặt nạ phân đoạn dựa trên các khả năng phát hiện đối tượng hiện có. Nó được giới thiệu để giải quyết nhiệm vụ phân đoạn thể hiện, trong đó mục tiêu không chỉ là phát hiện đối tượng trong ảnh mà còn phân đoạn chính xác các điểm ảnh tương ứng với từng thể hiện đối tượng.



Hình 2 Ví dụ về Mask R-CNN

Trong paper gốc của tác giả có nói đầy đủ như sau : "The Mask R-CNN framework is built on top of Faster R-CNN" . Có nghĩa là khi cho một hình ảnh vào ngoài việc trả ra label và bouding box của từng object trong một hình ảnh thì nó sẽ thêm cho chúng ta các object mask.

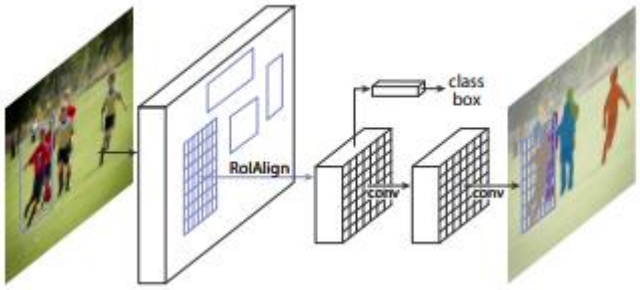


Hình 3 Sơ đồ hoạt động Mask R-CNN

1.3.2 Kiến trúc của Mask R-CNN

Mask R-CNN được Kaiming He và cộng sự đề xuất vào năm 2017. Nó rất giống với Faster R-CNN, ngoại trừ việc có thêm một lớp để dự đoán phân đoạn. Giai đoạn tạo đề xuất vùng giống nhau ở cả hai kiến trúc: giai đoạn thứ hai hoạt

động song song, dự đoán lớp, tạo hộp giới hạn cũng như xuất ra mặt nạ nhị phân cho mỗi RoI

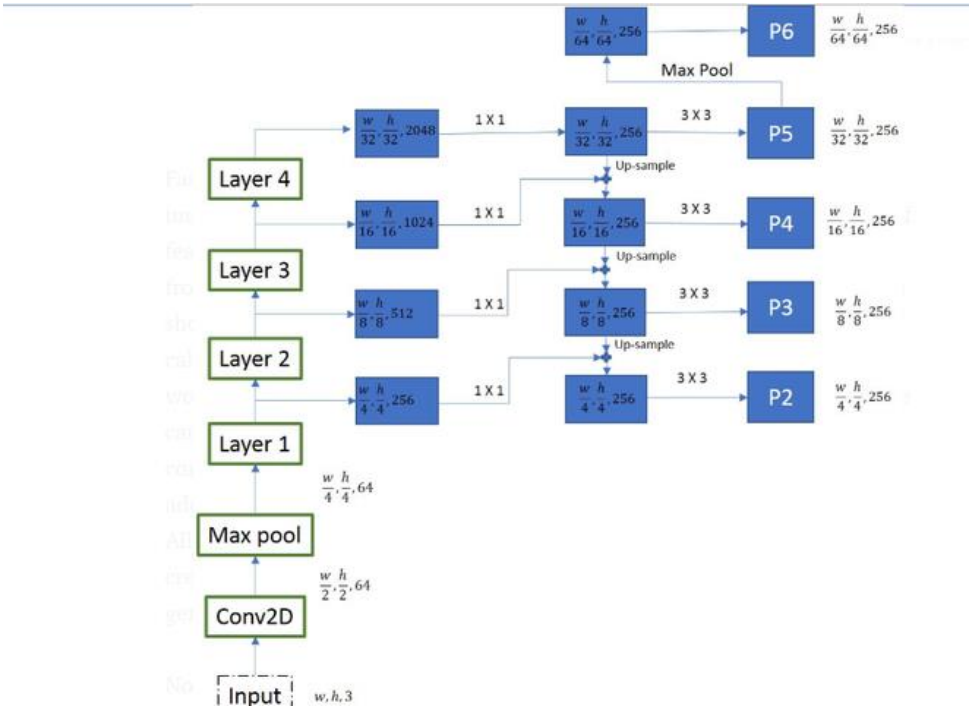


Hình 4 Kiến trục của Mask R-CNN

Kiến trúc của Mask R-CNN bao gồm :

- **Mạng xương sống (Backbone network)**

  Các tác giả của Mask R-CNN đã thử nghiệm với hai loại mạng xương sống. Loại thứ nhất là kiến trúc ResNet chuẩn (ResNet-C4) và loại còn lại là ResNet với mạng kim tự tháp đặc trưng. Kiến trúc ResNet chuẩn tương tự như Faster R-CNN, nhưng ResNet-FPN đã đề xuất một số cải tiến. Điều này bao gồm việc tạo RoI nhiều lớp. Mạng kim tự tháp đặc trưng nhiều lớp này tạo ra RoI ở nhiều quy mô khác nhau, giúp cải thiện độ chính xác của kiến trúc ResNet trước đây.

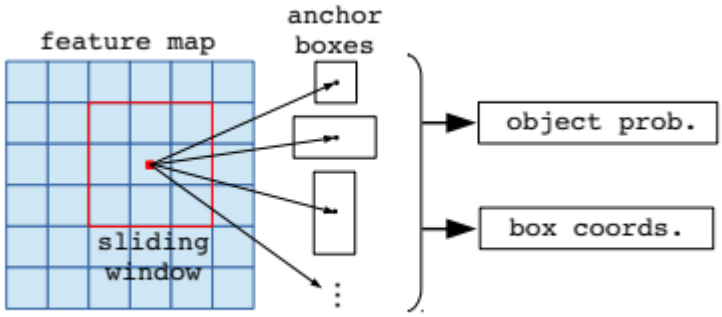


Hình 5 Kiến trúc xương sống của Mask R-CNN

Ở mỗi lớp, kích thước bản đồ đặc trưng được giảm đi một nửa và số lượng bản đồ đặc trưng được tăng gấp đôi. Chúng tôi lấy đầu ra từ bốn lớp (lớp - 1, 2, 3 và 4) . Để tạo ra các bản đồ đặc trưng cuối cùng, chúng tôi sử dụng một phương pháp được gọi là đường dẫn trên cùng-dưới cùng. Chúng tôi bắt đầu từ bản đồ đặc trưng trên cùng (w/32, h/32, 256) và làm việc theo cách của chúng tôi xuống các bản đồ lớn hơn, thông qua các hoạt động nâng cấp. Trước khi lấy mẫu, chúng tôi cũng áp dụng tích chập 1*1 để giảm số kênh xuống còn 256. Sau đó, số này được thêm từng phần tử vào đầu ra được lấy mẫu lên từ lần lặp trước. Tất cả các đầu ra đều phải chịu các lớp tích chập 3 X 3 để tạo ra 4 bản đồ đặc trưng cuối cùng (P2, P3, P4, P5) . Bản đồ đặc trưng thứ 5 (P6) được tạo ra từ hoạt động gộp tối đa từ P5 .

- **Mạng lưới đề xuất khu vực(Region Proposal Network)**

Toàn bộ bản đồ đặc trưng tích chập được tạo ra bởi lớp trước đó sẽ được truyền qua lớp tích chập 3*3 . Đầu ra của lớp này sau đó được truyền vào hai nhánh song song để xác định điểm số đối tượng và hồi quy tọa độ hộp giới hạn.



Hình 6 Anchor Generation Mask R-CNN

Ở đây, chúng ta chỉ sử dụng một bước anchor và 3 tỷ lệ anchor cho kim tự tháp đặc điểm (vì chúng ta đã có bản đồ đặc điểm có nhiều kích cỡ khác nhau để kiểm tra các đối tượng có nhiều kích cỡ khác nhau).

- **Biểu diễn mặt nạ(Mask Representation)**

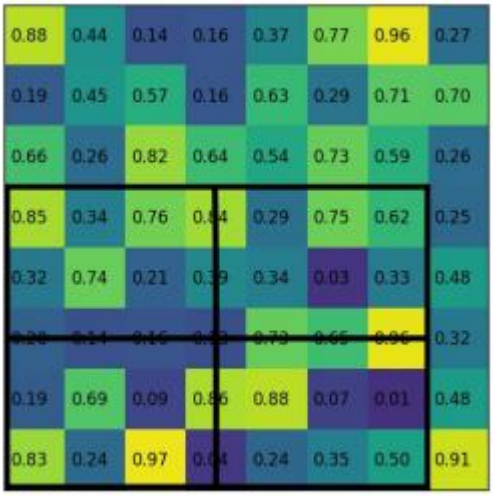
Mặt nạ chứa thông tin không gian về đối tượng. Do đó, không giống như các lớp hồi quy hộp giới hạn và phân loại, chúng ta không thể thu gọn đầu ra thành một lớp được kết nối hoàn toàn để cải thiện vì nó yêu cầu sự tương ứng giữa các điểm ảnh từ lớp trên. Mask R-CNN sử dụng một mạng được kết nối hoàn toàn để dự đoán mặt nạ. ConvNet này lấy RoI làm đầu vào và đưa ra biểu diễn mặt nạ m*m . Chúng tôi cũng nâng cấp mặt nạ này để suy luận trên hình ảnh đầu

vào và giảm các kênh xuống còn 256 bằng cách sử dụng phép tích chập 1*1 . Để tạo đầu vào cho mạng được kết nối hoàn toàn này để dự đoán mặt nạ, chúng ta sử dụng RoIAlign. Mục đích của RoIAlign là sử dụng để chuyển đổi các bản đồ đặc trưng có kích thước khác nhau do mạng đề xuất vùng tạo ra thành một bản đồ đặc trưng có kích thước cố định. Bài báo về Mask R-CNN đề xuất hai biến thể kiến trúc. Trong một biến thể, đầu vào của CNN tạo mặt nạ được truyền sau khi áp dụng RoIAlign (ResNet C4), nhưng trong một biến thể khác, đầu vào được truyền ngay trước lớp được kết nối hoàn toàn (Mạng FPN).

Nhánh tạo mặt nạ này là một mạng tích chập đầy đủ và nó đưa ra K * (m*m) , trong đó K là số lớp (một lớp cho mỗi lớp) và m=14 đối với ResNet-C4 và 28 đối với ResNet_FPN .

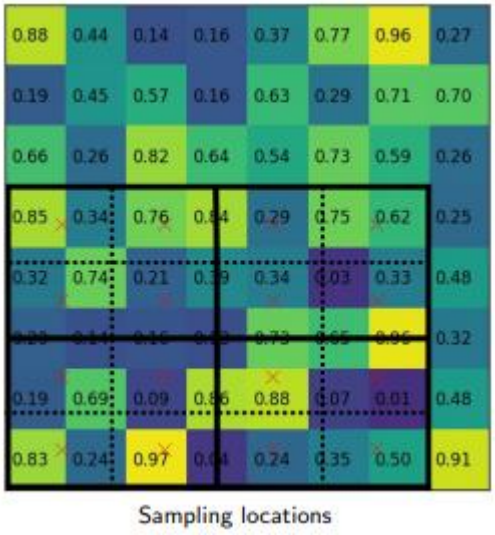
- **RoI Align**

RoI Align có cùng mục đích như RoI Pool, đó là tạo ra các vùng quan tâm có kích thước cố định từ các đề xuất vùng. Nó hoạt động theo các bước sau:



Hình 7 Căn chỉnh ROI

Với bản đồ đặc trưng của lớp Tích chập trước đó có kích thước h*w , hãy chia bản đồ đặc trưng này thành các lưới M * N có kích thước bằng nhau (chúng ta KHÔNG chỉ lấy giá trị số nguyên).



Hình 8 Sampling locations

Tốc độ suy luận của mặt nạ R-CNN là khoảng 2 fps , đây là tốc độ tốt khi xét đến việc bổ sung nhánh phân đoạn vào kiến trúc.

1.3.3 Qúa trình hoạt động của Mask R-CNN

Mask R-CNN được xây dựng dựa trên kiến trúc hai giai đoạn của Faster R-CNN, kết hợp thêm một nhánh để dự đoán mặt nạ phân đoạn cho mỗi đối tượng được phát hiện.

Cải tiến quan trọng nhất trong Mask R-CNN là sự ra đời của nhánh thứ ba, nhánh mặt nạ (mask branch), hoạt động song song với nhánh đề xuất vùng và nhánh phân loại hiện có. Sau khi các đề xuất vùng được tạo ra bởi mạng đề xuất vùng (RPN), nhánh mặt nạ chịu trách nhiệm dự đoán mặt nạ nhị phân cho mỗi vùng được đề xuất, phác thảo ranh giới chính xác ở cấp độ pixel của đối tượng. Điều này cho phép Mask R-CNN không chỉ xác định và phân loại đối tượng mà còn cung cấp mặt nạ phân đoạn chi tiết cho từng trường hợp.

Nhánh mặt nạ sử dụng cơ chế căn chỉnh từng điểm ảnh, thường được triển khai với phép gộp RoI được căn chỉnh theo không gian, để đảm bảo sự tương ứng chính xác giữa vùng được đề xuất và mặt nạ được tạo ra. Đầu ra cuối cùng là một tập hợp các hộp giới hạn, nhãn lớp và mặt nạ cấp độ điểm ảnh cho mỗi đối tượng được phát hiện trong ảnh.

1.3.4 Ưu và nhược điểm của Mask R-CNN

1. Ưu điểm

- Phân đoạn trường hợp chính xác: Mask R-CNN vượt trội trong việc cung cấp mặt nạ phân đoạn chính xác ở cấp độ pixel cho từng đối tượng được phát hiện.

- Tính linh hoạt: Có thể xử lý nhiều tác vụ trong một khuôn khổ duy nhất.
- Mạng đề xuất khu vực (RPN): Bằng cách kết hợp RPN, Mask R-CNN tạo ra các đề xuất khu vực một cách hiệu quả, cho phép tập trung vào các khu vực có liên quan và giảm đáng kể khối lượng tính toán so với các phương pháp quét toàn diện.
- Kiến trúc linh hoạt
- Hiệu suất tiên tiến: Mask R-CNN luôn đạt được kết quả tiên tiến trong các tiêu chuẩn phân đoạn phiên bản.

2. Nhược điểm

- Cường độ tính toán: Nhánh dự đoán mặt nạ làm tăng tải tính toán.
- Yêu cầu về tài nguyên: Mask R-CNN thường yêu cầu tài nguyên tính toán lớn như GPU.
- Thách thức về chú thích dữ liệu: Tốn nhiều công sức và khó tạo cho một số miền nhất định.
- Ứng dụng thời gian thực hạn chế: Mặc dù có độ chính xác cao, Mask R-CNN có thể không phù hợp với các ứng dụng thời gian thực có yêu cầu độ trễ nghiêm ngặt.

1.4.5 Ứng dụng của Mask R-CNN

Do có khả năng bổ sung để tạo mặt nạ phân đoạn, nó được sử dụng trong nhiều ứng dụng thị giác máy tính như:

- Ước tính tư thế con người
- Xe tự lái

- Bản đồ hình ảnh máy bay không người lái, v.v.

1.4 Autoencoder

1.4.1 Giới thiệu về Autoencoder

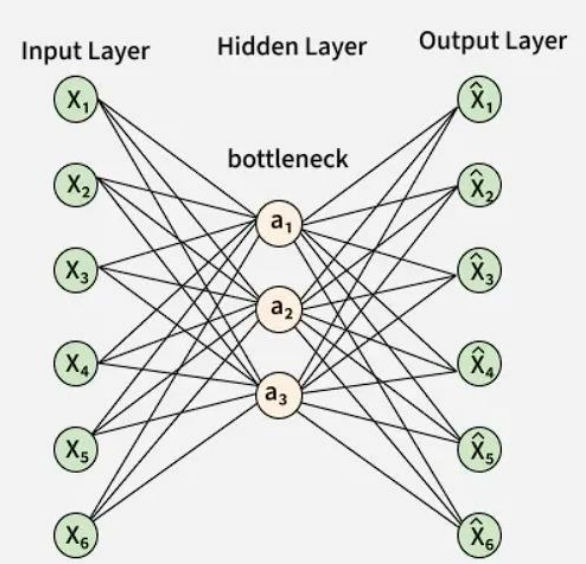
Autoencoder là một loại mạng nơ-ron đặc biệt, có khả năng học cách nén dữ liệu thành dạng nhỏ gọn và sau đó tái cấu trúc dữ liệu sao cho khớp nhất với dữ liệu đầu vào ban đầu. Chúng bao gồm:

- Encoder giúp nắm bắt các đặc điểm quan trọng bằng cách giảm chiều.
- Decoder xây dựng lại dữ liệu từ biểu diễn nén này.

Mô hình được huấn luyện bằng cách giảm thiểu lỗi tái cấu trúc bằng các hàm mất mát như Lỗi Bình Phương Trung Bình (Mean Squared Error) hoặc Entropy Chéo Nhị Phân (Binary Cross-Entropy). Các hàm này được áp dụng trong các tác vụ như loại bỏ nhiễu, phát hiện lỗi và trích xuất đặc trưng, trong đó việc nắm bắt biểu diễn dữ liệu hiệu quả là rất quan trọng.

1.4.2 Kiến trúc của Autoencoder

Kiến trúc của bộ mã hóa tự động bao gồm ba thành phần chính hoạt động cùng nhau để nén và sau đó tái tạo dữ liệu như sau:



Hình 9 Kiến trúc của Autoencoder

1. Encoder

Nó nén dữ liệu đầu vào thành dạng nhỏ hơn, dễ quản lý hơn bằng cách giảm số chiều của dữ liệu trong khi vẫn bảo toàn thông tin quan trọng. Nó có ba lớp:

- Lớp đầu vào : Đây là nơi dữ liệu gốc đi vào mạng. Dữ liệu này có thể là hình ảnh, văn bản hoặc bất kỳ dữ liệu có cấu trúc nào khác.
- Lớp ẩn : Các lớp này thực hiện một loạt các phép biến đổi trên dữ liệu đầu vào. Mỗi lớp ẩn áp dụng các trọng số và hàm kích hoạt để nắm bắt các mẫu quan trọng, từ đó giảm dần kích thước và độ phức tạp của dữ liệu.
- Đầu ra (Không gian tiềm ẩn) : Bộ mã hóa xuất ra một vectơ nén được gọi là biểu diễn tiềm ẩn hoặc mã hóa. Vector này nắm bắt các đặc điểm quan trọng của dữ liệu đầu vào ở dạng cô đọng, giúp lọc nhiễu và dư thừa.

2. Bottleneck (Latent Space)

Đây là lớp nhỏ nhất của mạng, biểu diễn phiên bản nén nhất của dữ liệu đầu vào. Nó đóng vai trò như nút thắt thông tin, buộc mạng phải ưu tiên các đặc điểm quan trọng nhất. Biểu diễn nén này giúp mô hình học được cấu trúc cơ bản và các mẫu chính của dữ liệu đầu vào, cho phép khái quát hóa tốt hơn và mã hóa dữ liệu hiệu quả hơn.

3. Decoder

Nó có nhiệm vụ lấy biểu diễn nén từ không gian tiềm ẩn và tái cấu trúc lại thành dạng dữ liệu gốc.

- Lớp ẩn : Các lớp này dần dần mở rộng vectơ tiềm ẩn trở lại không gian nhiều chiều hơn. Thông qua các phép biến đổi liên tiếp, bộ giải mã cố gắng khôi phục hình dạng và chi tiết dữ liệu ban đầu.
- Lớp Đầu ra : Lớp cuối cùng tạo ra đầu ra được tái tạo, nhằm mục đích giống nhất với đầu vào ban đầu. Chất lượng tái tạo phụ thuộc vào khả năng giảm thiểu sự khác biệt giữa đầu vào và đầu ra của bộ mã hóa-giải mã trong quá trình huấn luyện.

4. Loss Function in Autoencoder Training

Trong quá trình huấn luyện, mục tiêu của bộ mã hóa tự động là giảm thiểu tổn thất tái tạo, đo lường mức độ khác biệt giữa đầu ra tái tạo và đầu vào ban đầu. Việc lựa chọn hàm tổn thất phụ thuộc vào loại dữ liệu đang được xử lý:

Sai số bình phương trung bình (MSE) : Thường được sử dụng cho dữ liệu liên tục. Nó đo lường chênh lệch bình phương trung bình giữa dữ liệu đầu vào và dữ liệu được tái tạo.

Cross-Entropy nhị phân : Được sử dụng cho dữ liệu nhị phân (giá trị 0 hoặc 1). Tính toán sự khác biệt về xác suất giữa đầu ra ban đầu và đầu ra được tái tạo.

Trong quá trình huấn luyện, mạng sẽ cập nhật trọng số bằng cách sử dụng lan truyền ngược để giảm thiểu tổn thất tái tạo này. Nhờ đó, mạng học cách trích xuất và giữ lại những đặc điểm quan trọng nhất của dữ liệu đầu vào được mã hóa trong không gian tiềm ẩn.

5. Efficient Representations in Autoencoders

Việc ràng buộc bộ mã hóa tự động giúp nó học các đặc điểm có ý nghĩa và cô đọng từ dữ liệu đầu vào, từ đó mang lại các biểu diễn hiệu quả hơn. Sau khi huấn luyện, chỉ phần mã hóa được sử dụng để mã hóa dữ liệu tương tự cho các tác vụ trong tương lai. Nhiều kỹ thuật khác nhau được sử dụng để đạt được điều này như sau:

Giữ các lớp ẩn nhỏ : Việc giới hạn kích thước của mỗi lớp ẩn buộc mạng phải tập trung vào các tính năng quan trọng nhất. Các lớp nhỏ hơn giúp giảm sự dư thừa và cho phép mã hóa hiệu quả.

Chính quy hóa : Các kỹ thuật như chính quy hóa L1 hoặc L2 thêm các điều khoản phạt vào hàm mất mát. Điều này ngăn ngừa hiện tượng quá khớp bằng cách loại bỏ các trọng số quá lớn, giúp đảm bảo mô hình học được các biểu diễn tổng quát và hữu ích.

Khử nhiễu: Trong quá trình khử nhiễu, nhiễu ngẫu nhiên được thêm vào đầu vào trong quá trình huấn luyện. Bộ mã hóa tự động học cách loại bỏ nhiễu này trong quá trình tái tạo, giúp tập trung vào các đặc điểm cốt lõi, không nhiễu và cải thiện độ tin cậy.

Điều chỉnh các hàm kích hoạt : Điều chỉnh các hàm kích hoạt có thể thúc đẩy tính thưa thớt bằng cách chỉ kích hoạt một vài nơ-ron tại một thời điểm. Tính thưa thớt này làm giảm độ phức tạp của mô hình và buộc mạng chỉ nắm bắt các đặc điểm có liên quan nhất.

1.4.3 Cách loại Autoencoder

1. ***Denoising Autoencoder***

Bộ mã hóa tự động khử nhiễu được đào tạo để xử lý các đầu vào bị hỏng hoặc nhiễu, học cách loại bỏ nhiễu và hỗ trợ tái tạo dữ liệu sạch. Nó ngăn mạng lưới chỉ ghi nhớ đầu vào và khuyến khích học các tính năng cốt lõi.

2. ***Sparse Autoencoder***

Bộ mã hóa tự động thưa thớt chứa nhiều đơn vị ẩn hơn các đặc trưng đầu vào nhưng chỉ cho phép một vài nơ-ron hoạt động đồng thời. Độ thưa thớt này được kiểm soát bằng cách đặt một số đơn vị ẩn về 0, điều chỉnh hàm kích hoạt hoặc thêm một hình phạt thưa thớt vào hàm mất mát.

3. ***Variational Autoencoder***

Bộ mã hóa tự động biến thiên (VAE) đưa ra các giả định về phân phối xác suất của dữ liệu và cố gắng học một phép xấp xỉ tốt hơn. Nó sử dụng phương pháp giảm dần gradient ngẫu nhiên để tối ưu hóa và học phân phối của các biến tiềm ẩn. Chúng được sử dụng để tạo dữ liệu mới, chẳng hạn như tạo hình ảnh hoặc văn bản chân thực.

Nó giả định rằng dữ liệu được tạo ra bởi Mô hình đồ họa có hướng và cố gắng tìm hiểu một phép tính gần đúng $q_\phi = (z|x)$ đến thuộc tính có điều kiện $q_\theta = (z|x)$ Ở đâu ϕ và θ lần lượt là các tham số của bộ mã hóa và bộ giải mã.

4. Convolutional Autoencoder

Bộ mã hóa tự động tích chập sử dụng mạng nơ-ron tích chập (CNN) được thiết kế để xử lý hình ảnh. Bộ mã hóa trích xuất các đặc điểm bằng các lớp tích chập và bộ giải mã tái tạo hình ảnh thông qua quá trình giải tích chập, còn được gọi là lấy mẫu nâng cao.

1.5 Generative Adversarial Network(GAN)

1.5.1 Giới thiệu về GAN

Mạng Đối kháng Sinh sinh (GAN) giúp máy móc tạo ra dữ liệu mới, chân thực bằng cách học hỏi từ các ví dụ hiện có. Được giới thiệu bởi Ian Goodfellow và nhóm của ông vào năm 2014, GAN đã thay đổi cách máy tính tạo ra hình ảnh, video, nhạc và nhiều thứ khác. Không giống như các mô hình truyền thống chỉ nhận dạng hoặc phân loại dữ liệu, GAN mang đến một phương pháp sáng tạo bằng cách tạo ra nội dung hoàn toàn mới, gần giống với dữ liệu thực tế. Khả năng này đã hỗ trợ nhiều lĩnh vực như nghệ thuật, trò chơi điện tử, chăm sóc sức khỏe và khoa học dữ liệu. Trong bài viết này, chúng ta sẽ tìm hiểu thêm về GAN và các khái niệm cốt lõi của nó.

1.5.2 Kiến trúc của GAN

1. *Generator Model*

Bộ tạo(Generator) là một mạng nơ-ron sâu lấy nhiễu ngẫu nhiên làm đầu vào để tạo ra các mẫu dữ liệu thực tế như hình ảnh hoặc văn bản. Nó học các mẫu dữ liệu cơ bản bằng cách điều chỉnh các tham số nội bộ trong quá trình huấn luyện thông qua lan truyền ngược . Mục tiêu của nó là tạo ra các mẫu mà bộ phân biệt phân loại là thực.

Chức năng mất mát của máy phát điện: Máy phát điện cố gắng giảm thiểu tổn thất này:

$$\boldsymbol{J_G = -\frac{1}{m}\sum_{i=1}^{m} \log D\,(G(z_i))}$$

Trong đó J_G đo lường mức độ máy phát điện đánh lừa bộ phân biệt tốt như thế nào. $G(z_i)$ là mẫu được tạo ra từ tiếng ồn ngẫu nhiên z_i, là xác suất ước tính của bộ phân biệt rằng mẫu được tạo ra là mẫu thực.

Generator model nhằm mục đích tối đa hóa $D(G(z_i)$ nghãi là nó muốn bộ phân biệt dữ liệu giả của nó thành dữ liệu thật(xác suất gần bằng 1)

2. *Discriminator Model*

Bộ phân biệt hoạt động(Discriminator) như một bộ phân loại nhị phân, giúp phân biệt giữa dữ liệu thực và dữ liệu được tạo ra. Nó học cách cải thiện khả năng phân loại thông qua quá trình huấn luyện, tinh chỉnh các tham số để phát hiện mẫu giả chính xác hơn. Khi xử lý dữ liệu hình ảnh, bộ phân biệt sử dụng các lớp tích chập hoặc các kiến trúc liên quan khác giúp trích xuất các đặc điểm và nâng cao khả năng của mô hình

Chức năng mất mát của bộ phân biệt: Bộ phân biệt cố gắng giảm thiểu tổn thất này:

$$\boldsymbol{J_D == -\frac{1}{m}\sum_{i=1}^{m} \log D\left(G(z_i)\right) -= -\frac{1}{m}\sum_{i=1}^{m} \log(1 - D\,(G(z_i))}$$

Trong đó J_G đo lường mức độ phân biệt mẫu thật và mẫu giả của bộ phân biệt . $G(z_i)$ là mẫu giả từ generator, $D(x_i)$là xác suất ước tính của discriminator cho rằng mẫu x_i là mẫu thực. $D(G(x)_i)$ là xác suất của discriminator cho rằng mẫu giả là thật.

Discriminator muốn phân biệt chính xác dữ liệu là dữ liệu thực(tối đa hóa $\mathbf{log}\,\boldsymbol{D}\left(\boldsymbol{G}(\boldsymbol{z_i})\right)$) và dữ liệu giả mạo như giả mạo (tối đa hóa $\mathbf{log}(\mathbf{1}-\boldsymbol{D}\,(\boldsymbol{G}(\boldsymbol{z_i}))$

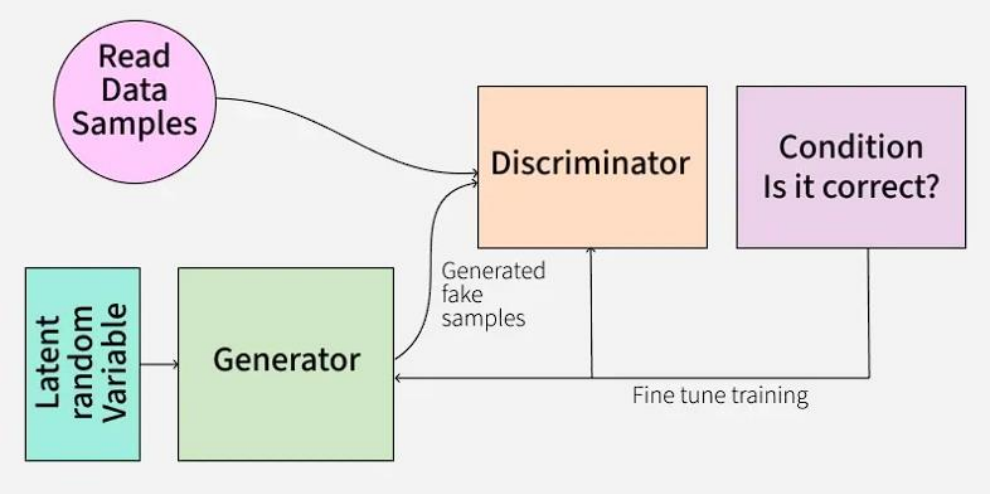
3. *MinMax Loss*

GANs are trained using a MinMax Loss between the generator and discriminator:

$$min_G max_D(G,D) = [E_{x\sim p_{data}}[\log D(x)] + E_{x\sim p_{z(z))}}[\log 1 - D(g(z))]]$$

Trong đó G is mạng Generator và D là mạng Discriminator. p_{data} phân phối dữ liệu thực. $p_z(z)$ phân phối nhiễu ngẫu nhiên(thường là bình thường hoặc đồng đều). D(x) là ước tính của người phân biệt dữ liệu thật. D(G(Z)) là ước tính của người phân biệt dữ liệu được tạo ra.

Máy phát điện cố gắng giảm thiểu tổn thất này (để đánh lừa bộ phân biệt) và bộ phân biệt cố gắng tối đa hóa tổn thất này (để phát hiện hàng giả một cách chính xác).



Hình 10 Cấu trúc của GAN

1.5.3 Cách thức hoạt động của GAN

GAN được huấn luyện bằng cách cho hai mạng lưới: Generator (G) và Discriminator (D) cạnh tranh và cải thiện cùng nhau. Dưới đây là quy trình từng bước.

1. *Generator's First Move*

Bộ tạo bắt đầu với một vectơ nhiễu ngẫu nhiên như số ngẫu nhiên. Nó sử dụng nhiễu này làm điểm khởi đầu để tạo ra một mẫu dữ liệu giả, chẳng hạn như hình ảnh được tạo ra. Các lớp bên trong của bộ tạo biến đổi nhiễu này thành thứ trông giống như dữ liệu thật.

2. *Discriminator's Turn*

   Bộ phân biệt nhận được hai loại dữ liệu:

   - Các mẫu thực tế từ tập dữ liệu đào tạo thực tế.
   - Mẫu giả được tạo ra bởi máy phát điện.

Nhiệm vụ của D là phân tích từng dữ liệu đầu vào và xác định xem đó là dữ liệu thật hay do G tạo ra. Nó đưa ra điểm xác suất từ 0 đến 1. Điểm 1 cho thấy dữ liệu có khả năng là thật và điểm 0 cho thấy đó là giả.

3. *Adversarial Learning*
   - Nếu bộ phân biệt phân loại chính xác dữ liệu thật và giả, nó sẽ hoạt động tốt hơn.
   - Nếu máy phát điện đánh lừa bộ phân biệt bằng cách tạo ra dữ liệu giả giống thật, nó sẽ nhận được bản cập nhật tích cực và bộ phân biệt sẽ bị phạt vì đưa ra quyết định sai.
4. *Generator's Improvement*
   - Mỗi lần bộ phân biệt nhầm dữ liệu giả thành dữ liệu thật, máy phát điện sẽ học hỏi từ thành công này.
   - Qua nhiều lần lặp lại, máy phát điện sẽ cải thiện và tạo ra các mẫu giả thuyết phục hơn.
5. *Discriminator's Adaptation*

- Bộ phân biệt cũng liên tục học hỏi bằng cách tự cập nhật để phát hiện dữ liệu giả tốt hơn.
- Sự tương tác qua lại liên tục này khiến cả hai mạng lưới ngày càng mạnh mẽ hơn theo thời gian.

6. *Training Progression*
   - Khi quá trình đào tạo tiếp tục, máy phát điện sẽ trở nên rất thành thạo trong việc tạo ra dữ liệu thực tế.
   - Cuối cùng, bộ phân biệt phải vật lộn để phân biệt thật và giả, cho thấy GAN đã đạt đến trạng thái được đào tạo tốt.
   - Ở thời điểm này, trình tạo có thể tạo ra dữ liệu tổng hợp chất lượng cao có thể được sử dụng cho nhiều ứng dụng khác nhau.

1.5.4 Các loại GAN

1. Vanilla GAN

Vanilla GAN là loại GAN đơn giản nhất. Nó bao gồm:

- Cả bộ generator và bộ discriminator đều được xây dựng bằng cách sử dụng perceptron nhiều lớp (MLP).
- Mô hình tối ưu hóa công thức toán học của nó bằng cách sử dụng phương pháp giảm dần độ dốc ngẫu nhiên (SGD).

Mặc dù mang tính nền tảng, Vanilla GAN có thể gặp phải các vấn đề như:

- Thu gọn chế độ : Máy phát điện tạo ra các loại đầu ra giới hạn nhiều lần.
- Đào tạo không ổn định : Bộ tạo và bộ phân biệt có thể không cải thiện trơn tru.

2. Conditional GAN (CGAN)

Conditional GAN (CGAN) bổ sung một tham số điều kiện để hướng dẫn quá trình tạo dữ liệu. Thay vì tạo dữ liệu ngẫu nhiên, chúng cho phép mô hình tạo ra các loại đầu ra cụ thể.

Hoạt động của CGAN:

- Biến có điều kiện (y) được đưa vào cả bộ tạo và bộ phân biệt.
- Điều này đảm bảo rằng trình tạo sẽ tạo ra dữ liệu tương ứng với điều kiện nhất định (ví dụ: tạo hình ảnh của các đối tượng cụ thể).
- Bộ phân biệt cũng nhận được nhãn để giúp phân biệt giữa dữ liệu thật và dữ liệu giả.

Ví dụ : Thay vì tạo ra bất kỳ hình ảnh ngẫu nhiên nào, CGAN có thể tạo ra một đối tượng cụ thể như chó hoặc mèo dựa trên nhãn.

3. Deep Convolutional GAN (DCGAN)

GAN tích chập sâu (DCGAN) là một trong những loại GAN phổ biến nhất được sử dụng để tạo hình ảnh.

Chúng quan trọng vì chúng:

- Sử dụng Mạng nơ-ron tích chập (CNN) thay vì các perceptron nhiều lớp đơn giản (MLP).
- Các lớp gộp tối đa được thay thế bằng bước tiến tích chập giúp mô hình hiệu quả hơn.
- Các lớp được kết nối hoàn toàn sẽ bị loại bỏ, cho phép hiểu rõ hơn về hình ảnh trong không gian.

DCGAN thành công vì chúng tạo ra hình ảnh chân thực, chất lượng cao.

4. Laplacian Pyramid GAN (LAPGAN)

Laplacian Pyramid GAN (LAPGAN) được thiết kế để tạo ra hình ảnh chất lượng cực cao bằng cách sử dụng phương pháp đa độ phân giải.

Hoạt động của LAPGAN:

- Sử dụng nhiều cặp bộ tạo-phân biệt ở các cấp độ khác nhau của kim tự tháp Laplacian.
- Đầu tiên, hình ảnh được lấy mẫu xuống ở mỗi lớp của kim tự tháp và được nâng cấp lại bằng GAN có điều kiện (CGAN).
- Quá trình này cho phép hình ảnh dần dần tinh chỉnh các chi tiết và giúp giảm nhiễu cũng như cải thiện độ rõ nét.

Nhờ khả năng tạo ra hình ảnh có độ chi tiết cao, LAPGAN được coi là phương pháp vượt trội để tạo ra hình ảnh chân thực.

5. Super Resolution GAN (SRGAN)

GAN siêu phân giải (SRGAN) được thiết kế để tăng độ phân giải của hình ảnh chất lượng thấp trong khi vẫn giữ được chi tiết.

Hoạt động của SRGAN:

- Sử dụng mạng nơ-ron sâu kết hợp với hàm mất mát đối nghịch.
- Cải thiện hình ảnh có độ phân giải thấp bằng cách thêm các chi tiết tốt hơn, giúp hình ảnh trông sắc nét và chân thực hơn.
- Giúp giảm các lỗi nâng cấp hình ảnh thường gặp như mờ và vỡ hình.

CHƯƠNG 2 – XÂY DỰNG CHƯƠNG TRÌNH DEMO

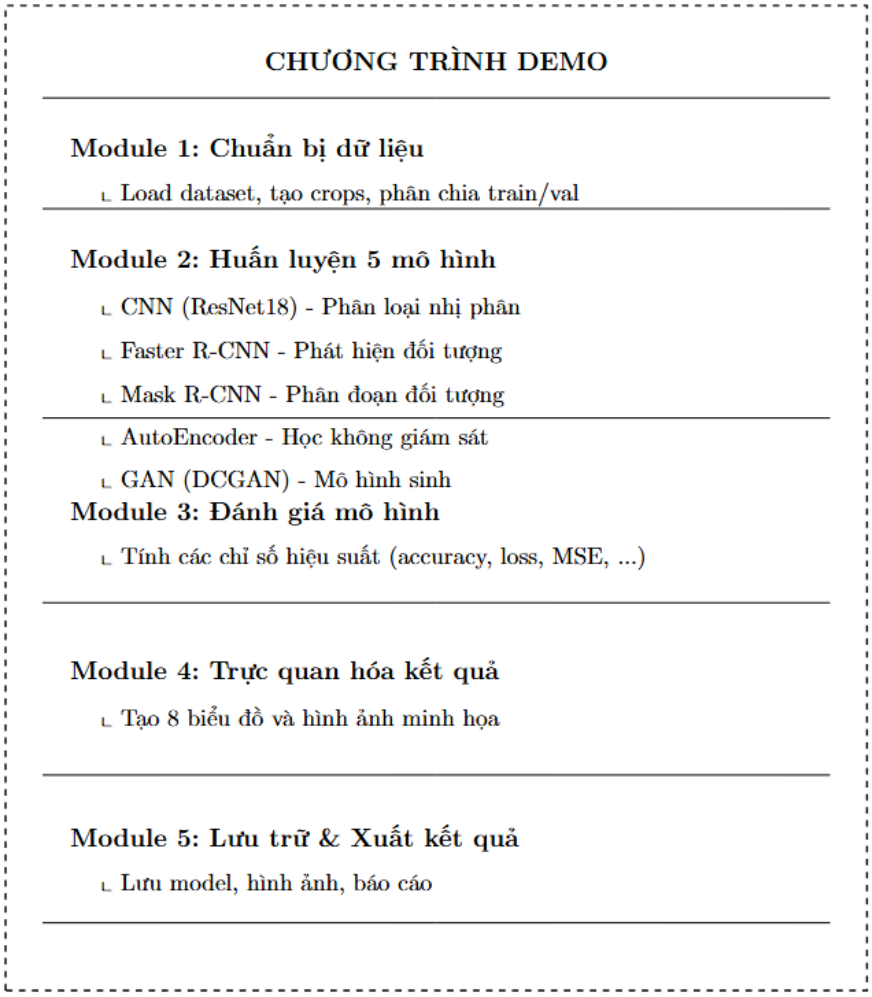
2.1 Giới thiệu

Để đánh giá hiệu quả của năm mô hình học sâu trong bài toán phát hiện người bộ hành, em xây dựng một chương trình demo toàn diện cho phép huấn luyện, đánh giá và so sánh các mô hình trên tập dữ liệu Penn-Fudan Pedestrian. Chương trình được xây dựng bằng Python 3.10+ với framework PyTorch, tối ưu hóa để chạy trên GPU (CUDA 11.8+) nhằm giảm thời gian huấn luyện.

2.2 Kiến trúc Hệ Thống

2.2.1 Các module chính

Chương trình demo được chia thành 5 module chính:



2.2.2 Công Nghệ Sử Dụng

Công Nghệ	Phiên Bản	Mục Đích
Python	3.10+	Ngôn ngữ lập trình chính
PyTorch	1.13+	Framework học sâu
torchvision	0.14+	Các mô hình detection/segmentation
Matplotlib	3.7+	Trực quan hóa dữ liệu
NumPy	1.24+	Xử lý mảng
PIL/Pillow	9.0+	Xử lý ảnh

Pandas	2.0+	Phân tích dữ liệu

2.3 Chuẩn Bị Dữ Liệu

2.3.1 Tập Dữ Liệu Dataset

Nguồn dữ liệu: Tập dữ liệu Penn-Fudan Pedestrian dùng cho các model CNN, Faster RCNN, Mask RCNN gồm:

- **124 ảnh** toàn cảnh độ phân giải 640×480 pixels
- **Khoảng 1000 người** được ghi chú vị trí và khoanh vùng
- **Định dạng:** PNG với mặt nạ (mask) nhị phân tương ứng

Cấu trúc dữ liệu:

```
PennFudanPed/
├── PNGImages/          (124 ảnh gốc)
│   ├── FudanPed00001.png
│   ├── FudanPed00002.png
│   └── ...
├── PedMasks/           (124 mặt nạ)
│   ├── FudanPed00001_mask.png
│   ├── FudanPed00002_mask.png
│   └── ...
└── (Được tạo bởi chương trình)
    ├── crops64/        (Tất cả crops 64×64)
    ├── crops64_pos/    (Crops dương - người)
└── crops64_neg/    (Crops âm - nền)
```

Nguồn dữ liệu: Tập dữ liệu Thumnails dùng cho các model Autoencoder, GAN gồm:

- **70000 ảnh** mặt người toàn cảnh độ phân giải 128×128 pixels
- **Định dạng:** PNG

2.3.2 Xử Lý Dữ Liệu

Bước 1: Trích xuất Bounding Boxes từ Mask

```
def load_target(mask_p):
    mask = np.array(Image.open(mask_p))
    obj_ids = np.unique(mask)[1:]  # Loại bỏ nền (0)

    # Tạo mask nhị phân cho từng người
    masks = (mask[..., None] == obj_ids).astype(np.uint8).transpose(2,0,1)

    # Trích xuất bounding box từ mask
    boxes = []
    for m in masks:
        pos = np.argwhere(m)
        y1, x1 = pos.min(0)
        y2, x2 = pos.max(0)
        boxes.append([x1, y1, x2, y2])

    return boxes, labels, masks
```

Bước 2: Tạo Crops 64×64

Từ mỗi bounding box, chúng em cắt ảnh con và resize về 64×64:

- Số lượng: ~1000+ crops từ người, ~3000+ crops từ nền
- Mục đích: Dùng cho CNN, AutoEncoder, GAN

Bước 3: Chia Tập Dữ Liệu

Sau khi chuẩn bị dữ liệu, em chia tập thành các phần:

- Tỷ lệ: 80% training, 20% validation
- Shuffle: Ngẫu nhiên hóa để tránh overfitting
- Batch size:
    - CNN: 32
    - Faster R-CNN: 2 (vì ảnh lớn)
    - Mask R-CNN: 2

- AutoEncoder: 64
- GAN: 64

2.4 5 mô hình deep learning

2.4.1 Mô Hình 1: CNN (Convolutional Neural Network)

❖ **Mục tiêu**

Mô hình CNN được dùng để giải bài toán **phân loại nhị phân** trên ảnh crop 64×64:

- **1 (Positive):** crop có người đi bộ (cắt từ Ground Truth bounding box)
- **0 (Negative):** crop nền (random crop không đè lên người)

❖ **Kiến trúc:**

```
ResNet18 với 2 lớp đầu ra (người/không phải người)

Đầu vào (64×64×3)
    ↓
Conv Block 1: [64 filters, 3×3, stride 2]
Conv Block 2: [128 filters, 3×3, stride 2]
Conv Block 3: [256 filters, 3×3, stride 2]
Conv Block 4: [512 filters, 3×3, stride 2]
    ↓
Global Average Pooling
    ↓
Fully Connected: 512 → 2 (softmax)
    ↓
Đầu ra (2 xác suất lớp)
```

❖ **Thông số huấn luyện:**

- Epochs: 10
- Optimizer: Adam (learning rate: 1e-3)
- Loss function: Cross-Entropy Loss
- Batch normalization: Có
- Dropout: Không

❖ **Kết quả mong đợi :**

- Phân loại nhanh crops 64×64
- Accuracy cao (>85%) trên validation set
- Có thể sử dụng trực tuyến (real-time)

2.4.2 Mô Hình 2: Faster R-CNN

❖ **Mục tiêu**

Thực hiện **phát hiện đối tượng (bounding box)** cho lớp:

- **Person = 1** (background = 0)

❖ **Kiến trúc*:***

```
ResNet50 + FPN (Feature Pyramid Network)
    ↓
Region Proposal Network (RPN)
    ↓
ROI Pooling
    ↓
Classification Head (2 lớp)
    ↓
Bounding Box Regression
    ↓
Đầu ra: boxes + scores
```

❖ **Thông số huấn luyện:**

Epochs: 6

Optimizer: SGD (learning rate: 0.005, momentum: 0.9)

Loss function: Smooth L1 + Cross-Entropy

Pre-trained: COCO weights

Backbone: ResNet50 + FPN

Cách sử dụng:

```
det_model = fasterrcnn_resnet50_fpn(weights="DEFAULT")
in_features = det_model.roi_heads.box_predictor.cls_score.in_features
det_model.roi_heads.box_predictor = FastRCNNPredictor(in_features, 2)

optimizer = torch.optim.SGD(
```

```
        [p for p in det_model.parameters() if p.requires_grad],
        lr=0.005, momentum=0.9, weight_decay=1e-4
    )

    for epoch in range(6):
        det_model.train()
        for imgs, targets in train_dl:
            imgs = [im.to(device) for im in imgs]
            targets = [{k:v.to(device) for k,v in t.items()} for t in
targets]
            loss_dict = det_model(imgs, targets)
            loss = sum(loss_dict.values())
            loss.backward()
            optimizer.step()
```

❖ **Kết quả mong đợi:**

- Phát hiện vị trí người trong ảnh
- Trả về bounding boxes có độ tin cậy
- Thích hợp cho surveillance/monitoring

2.4.3 Mô Hình 3: Mask R-CNN

❖ **Mục tiêu**

Thực hiện **phân đoạn từng cá thể (instance segmentation)**:

- Vừa dự đoán **bbox**, vừa dự đoán **mask** cho từng người.

❖ **Kiến trúc:**

```
Faster R-CNN Base
    ↓
ROI Align (thay ROI Pooling)
    ↓
Mask Head (FCN để dự đoán mask)
    ↓
Đầu ra: boxes + scores + masks
```

❖ **Thông số huấn luyện:**

Epochs: 6

Optimizer: SGD (learning rate: 0.005, momentum: 0.9)

Loss function: mask + classification + regression

Mask Head: 256-d hidden layer + FCN decoder

Backbone: ResNet50 + FPN

Cách sử dụng:

```
seg_model = maskrcnn_resnet50_fpn(weights="DEFAULT")

# Thay mask predictor cho 2 lớp
in_features_mask =
seg_model.roi_heads.mask_predictor.conv5_mask.in_channels
seg_model.roi_heads.mask_predictor = MaskRCNNPredictor(
    in_features_mask, 256, 2
)

for epoch in range(6):
    seg_model.train()
    for imgs, targets in train_dl:
        loss_dict = seg_model(imgs, targets)
        loss = sum(loss_dict.values())
        loss.backward()
        optimizer.step()
```

❖ **Kết quả mong đợi:**

- Phân đoạn từng người riêng biệt
- Trả về mask nhị phân cho mỗi người
- Áp dụng: đếm tập trung đông người
- Tốc độ: ~7 FPS trên GPU
- Chính xác hơn Faster R-CNN

2.4.4 Mô Hình 4: AutoEncoder (Denoising AutoEncoder)

❖ **Mục tiêu :**

Mô hình AutoEncoder được sử dụng nhằm học biểu diễn nén (latent representation) của ảnh người đi bộ, đồng thời khử nhiễu ảnh đầu vào. Mô hình học cách tái tạo ảnh gốc từ ảnh bị nhiễu thông qua quá trình nén – giải nén.

❖ **Kiến trúc mô hình :**

AutoEncoder gồm hai phần chính: **Encoder** và **Decoder**.

Encoder (Nén đặc trưng):

- Conv2d (3 → 32), kernel 4×4, stride 2, padding 1 + ReLU
- Conv2d (32 → 64), kernel 4×4, stride 2, padding 1 + ReLU
- Conv2d (64 → 128), kernel 4×4, stride 2, padding 1 + ReLU

→ Đầu ra Encoder là **bottleneck feature map kích thước 8×8×128**

Decoder (Giải nén & tái tạo ảnh):

- ConvTranspose2d (128 → 64), kernel 4×4, stride 2, padding 1 + ReLU
- ConvTranspose2d (64 → 32), kernel 4×4, stride 2, padding 1 + ReLU
- ConvTranspose2d (32 → 3), kernel 4×4, stride 2, padding 1 + Sigmoid

→ Ảnh đầu ra có cùng kích thước với ảnh gốc (64×64×3)

Cấu trúc Bottleneck

- Kích thước: **8 × 8 × 128**
- Chức năng: nén thông tin hình ảnh người đi bộ vào không gian đặc trưng thấp chiều, giữ lại các đặc trưng quan trọng nhất.

❖ **Thông số huấn luyện**

Số epoch: 10

Optimizer: Adam

Learning rate: 1e-3

Loss function: Mean Squared Error (MSE)

Số lớp: 3 lớp Encoder + 3 lớp Decoder

❖ **Quy trình huấn luyện**

- Ảnh đầu vào được **thêm nhiễu Gaussian** trước khi đưa vào mô hình.

- Mô hình học cách tái tạo lại ảnh sạch từ ảnh nhiễu.
- Loss được tính bằng **MSE giữa ảnh tái tạo và ảnh gốc**.
- Trọng số được cập nhật sau mỗi batch.

❖ **Kết quả mong đợi**

- Mô hình học được biểu diễn nén hiệu quả cho ảnh người đi bộ
- **MSE < 0.03**, thể hiện khả năng tái tạo tốt
- Ảnh đầu ra rõ cấu trúc, giảm nhiễu đáng kể

❖ **Ứng dụng**

- Phát hiện bất thường (Anomaly Detection)
- Nén dữ liệu hình ảnh
- Tiền xử lý dữ liệu cho các mô hình nhận dạng khác

2.4.5 Mô Hình 5: GAN-DCGAN kết hợp Gradient Penalty

❖ **Mục tiêu**

Mô hình GAN được sử dụng để **sinh ảnh người đi bộ giả có tính logic**, phục vụ cho:

- Data Augmentation
- Mở rộng tập dữ liệu huấn luyện
- Bảo vệ quyền riêng tư (Privacy-preserving dataset)

❖ **Kiến trúc mô hình :**

Kiến trúc Generator

Đầu vào:

- Vector nhiễu ngẫu nhiên (latent vector) kích thước **64 chiều**

Các tầng sinh ảnh:

- ConvTranspose2d (64 → 512), kernel 4×4 + ReLU
- ConvTranspose2d (512 → 256), kernel 4×4 + ReLU
- ConvTranspose2d (256 → 128), kernel 4×4 + ReLU
- ConvTranspose2d (128 → 3), kernel 4×4 + Tanh

→ Sinh ảnh đầu ra kích thước **64×64×3**

Kiến trúc Discriminator (Critic)

- **Đầu vào:** Ảnh 64×64×3
- **Các tầng phân biệt:**
  - Conv2d (3 → 64), kernel 4×4 + LeakyReLU
  - Conv2d (64 → 128), kernel 4×4 + BatchNorm + LeakyReLU
  - Conv2d (128 → 256), kernel 4×4 + BatchNorm + LeakyReLU
  - Conv2d (256 → 1), kernel 4×4

→ Đầu ra là **điểm đánh giá ảnh thật/giả**

❖ **Thông số huấn luyện**

- **Epochs**: 10
- **Optimizer Generator**: Adam (lr = 2e-4, $\beta_1 = 0.5$, $\beta_2 = 0.999$)
- **Optimizer Discriminator**: Adam (lr = 2e-4, $\beta_1 = 0.5$, $\beta_2 = 0.999$)
- **Loss function**: Binary Cross-Entropy with Logits
- **Latent dimension**: 64

❖ **Quy trình huấn luyện**

1. Discriminator được huấn luyện để phân biệt ảnh thật và ảnh giả

2. Generator học cách sinh ảnh để đánh lừa Discriminator
3. Gradient Penalty được sử dụng nhằm:
    - Ổn định quá trình huấn luyện
    - Giảm hiện tượng mode collapse
4. Hai mạng được huấn luyện luân phiên theo từng batch

❖ **Kết quả mong đợi**

- Sinh ảnh người đi bộ có cấu trúc hợp lý
- Hình dáng và tỷ lệ cơ thể tương đối tự nhiên
- Loss của Generator và Discriminator dần ổn định

❖ **Ứng dụng**

- Data Augmentation cho các mô hình phát hiện người
- Tạo dữ liệu huấn luyện thay thế dữ liệu thật
- Nghiên cứu GAN và mô hình sinh ảnh

CHƯƠNG 3 – KẾT QUẢ THỰC NGHIỆM

3.1 Cài đặt môi trường

Thực nghiệm được triển khai trên bộ dữ liệu **PennFudanPed** (bài toán người đi bộ), nhằm đánh giá 3 hướng tiếp cận:

1. **CNN** cho bài toán **phân loại nhị phân** (có người / không người) trên ảnh crop 64×64
2. **Faster R-CNN** cho bài toán **phát hiện đối tượng** (bounding box)
3. **Mask R-CNN** cho bài toán **phân đoạn instance** (mask từng người)

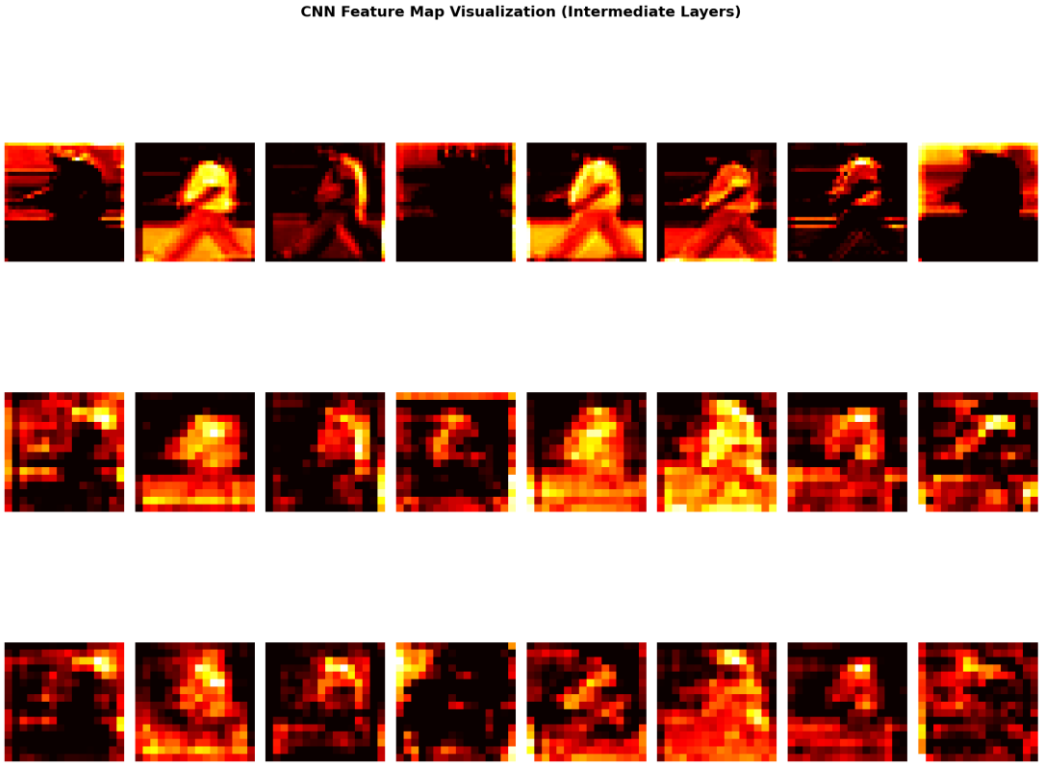
Với bộ dữ liệu Thumbnails nhằm đánh giá so sánh khả năng khữ nhiễu ảnh dựa trên 2 model Autoencoder và Gan

Môi trường thực nghiệm (Kaggle GPU):

- PyTorch: **2.8.0 + CUDA 12.6**
- GPU: **Tesla P100-PCIE-16GB**, VRAM khoảng **17.06 GB**
- Huấn luyện và lưu toàn bộ kết quả vào thư mục /kaggle/working

3.2 Kết quả thực nghiệm trên PennFudanPed

3.2.1 Kết quả thực nghiệm CNN (Binary Classification)



Hình 11 CNN Features Map

Hình 12 CNN Results

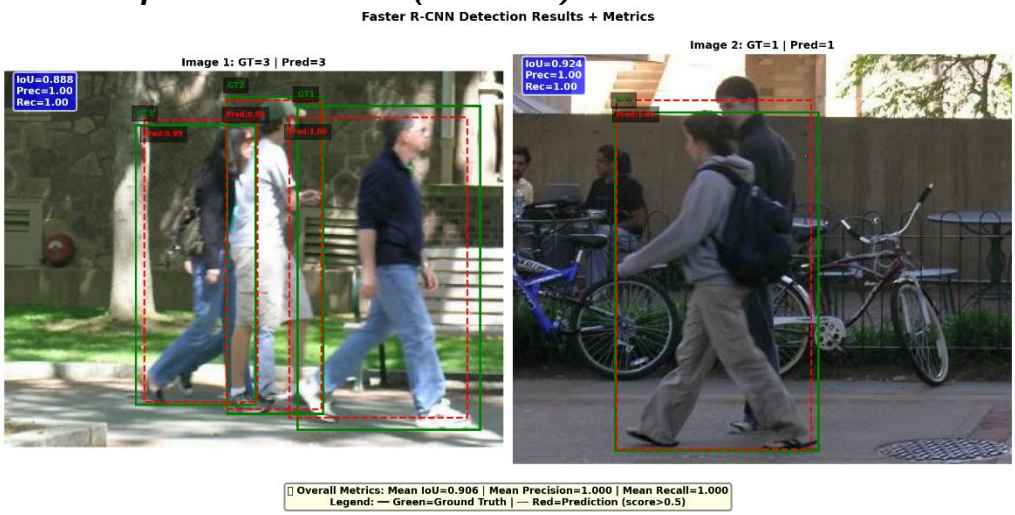
Kết quả demo/đánh giá:

- **Accuracy: 0.969**
- **Precision: 1.000**
- **Recall: 0.950**
- **F1-score: 0.974**

Nhận xét:

- Precision = 1.000 cho thấy mô hình hầu như **không dự đoán nhầm background thành "có người"** (ít false positive).
- Recall = 0.950 nghĩa là vẫn còn **bỏ sót một số ảnh có người** (false negative nhẹ), nhưng tổng thể F1 cao (0.974) → mô hình phân loại ổn định.

3.2.2 Kết quả Faster R-CNN (Detection)



Hình 13 Faster R-CNN Results

Kết quả demo/đánh giá:

- **Detection IoU: 0.906**
- **Precision: 1.000**
- **Recall: 1.000**

Nhận xét:

- Precision và Recall đều đạt 1.000 → mô hình phát hiện đúng hầu hết đối tượng với ngưỡng score sử dụng.
- IoU 0.906 thể hiện **box dự đoán khớp tốt** với ground truth (sai lệch biên không đáng kể).

3.2.3 Kết quả Mask R-CNN (Segmentation)

Hình 14 Mask RCNN Result

Kết quả demo/đánh giá:

- **Mask IoU: 0.858**

Nhận xét:

Mask IoU ~0.858 cho thấy mô hình phân đoạn khá tốt, tuy có thể còn sai lệch nhẹ ở **biên đối tượng** (rìa người, vùng chồng lấp, hoặc vùng nền phức tạp).

3.2.4 Đánh giá định lượng tổng hợp (Validation)

Kết quả tổng hợp trên tập validation:

Nhóm mô hình	**Chỉ số chính**	**Giá trị**
CNN	Accuracy	**0.913**
Faster R-CNN	IoU (mean)	**0.843**
Faster R-CNN	Precision	**0.961**
Faster R-CNN	Recall	**1.000**
Mask R-CNN	Mask IoU (mean)	**0.878**

Bảng 1 Bảng kết quả tổng hợp độ đo trên Validation

Nhận xét:

- Detection có Recall = 1.000 → mô hình **ít bỏ sót người**, phù hợp các bài toán ưu tiên "không bỏ qua đối tượng".
- Mask IoU (mean) 0.878 khá cao, phản ánh chất lượng phân đoạn tốt trên validation.
- CNN Accuracy 0.913 trên validation là mức ổn định cho bài toán nhị phân, nhất là khi dữ liệu bị lệch lớp (67.5% positive).

3.2.5 Phân tích chi phí mô hình (số tham số)

Thống kê tham số:

- **CNN: ~11.18M** parameters
- **Faster R-CNN: ~41.08M** parameters

- **Mask R-CNN: ~43.70M** parameters

Nhận xét:

- CNN nhẹ hơn nhiều, phù hợp bài toán phân loại nhanh (cost thấp).
- Faster/Mask R-CNN nặng hơn do backbone + FPN + head (và Mask head), đổi lại giải quyết được bài toán phức tạp hơn (detection/segmentation).

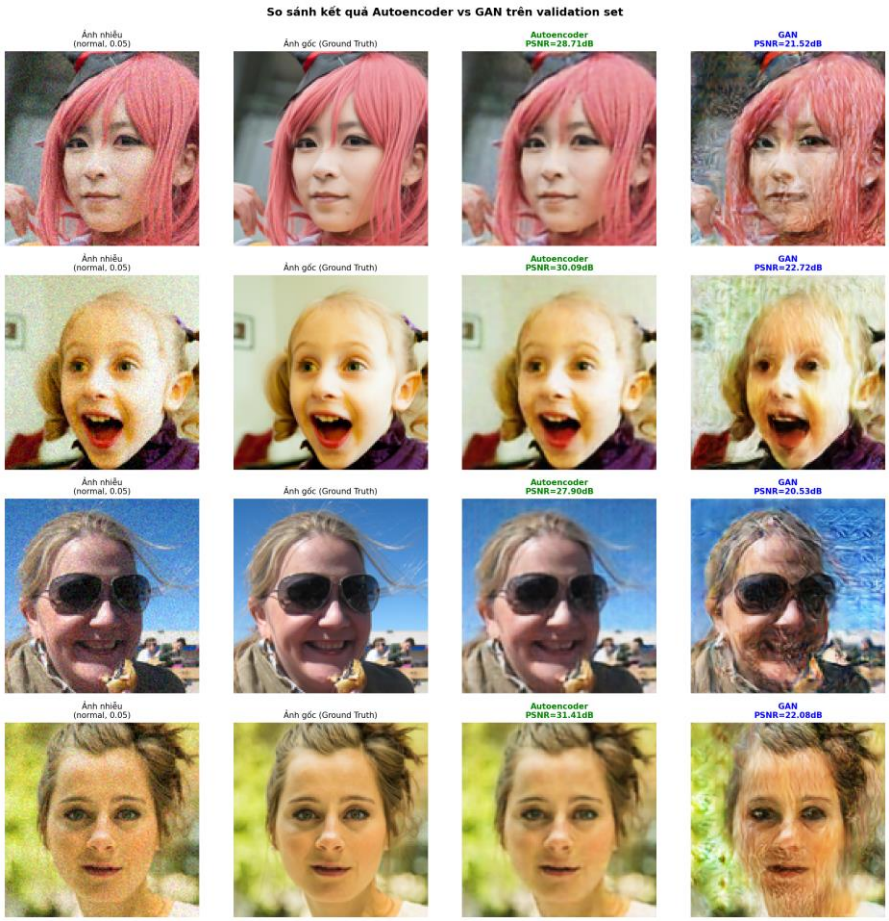
3.2 Kết quả thực nghiệm trên Thumbnails

Thực nghiệm trên bộ dữ liệu **Thumbnails** nhằm đánh giá khả năng **khử nhiễu ảnh (image denoising)** theo hai hướng:

- **Autoencoder**: học tái tạo ảnh sạch từ ảnh nhiễu (reconstruction-based).
- **GAN**: denoising theo hướng đối kháng (adversarial), gồm Generator sinh ảnh khử nhiễu và Discriminator phân biệt ảnh thật/giả.

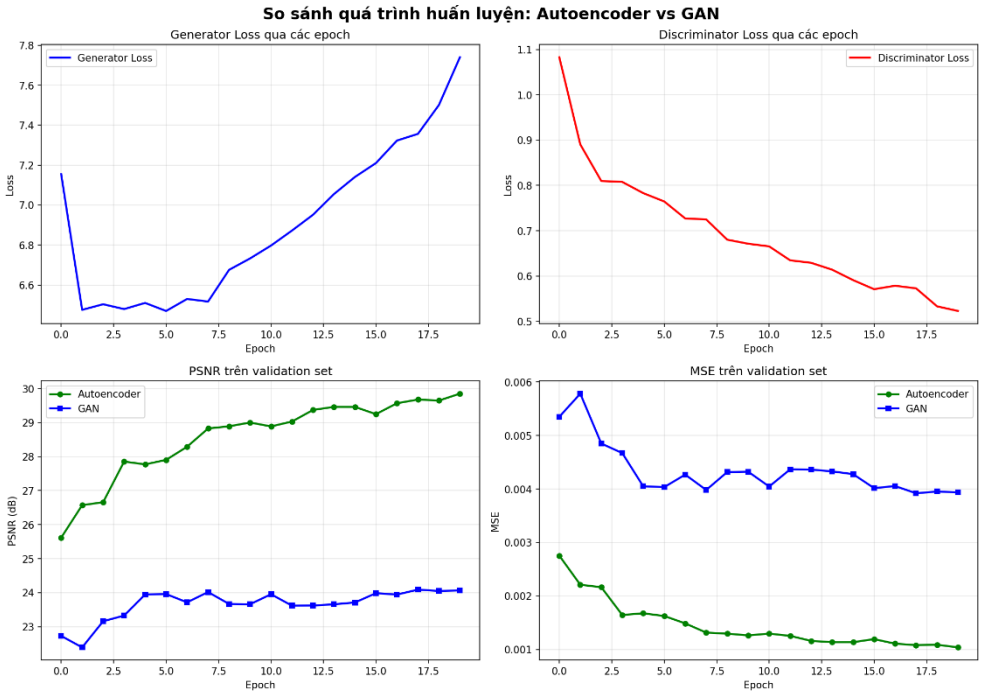
Trong thí nghiệm, ảnh được thêm **Gaussian noise (normal, $\sigma = 0.05$)** và đánh giá bằng:

- **PSNR (dB)**: càng cao càng tốt
- **MSE**: càng thấp càng tốt



Hình 15 So sánh kết quả Autodencoder và Gan trên validation set

Hình minh hoạ gồm bốn cột: ảnh nhiễu, ảnh gốc (ground truth), ảnh khôi phục bởi Autoencoder và ảnh khôi phục bởi GAN; kết quả cho thấy Autoencoder đạt chất lượng tái tạo tốt hơn (PSNR cao hơn), trong khi GAN xuất hiện artefact và biến dạng texture ở một số vùng chi tiết.



Hình 16 So sánh quá trình huấn luyện Autoencodr và Gan

PSNR của Autoencoder tăng dần và MSE giảm ổn định, trong khi GAN có PSNR thấp hơn và MSE cao hơn; đồng thời đường cong Generator/Discriminator loss cho thấy quá trình đối kháng chưa cân bằng hoàn toàn, ảnh hưởng đến chất lượng khôi phục.

3.2.1 Nhận xét kết quả Autoencoder

Autoencoder thể hiện khả năng khử nhiễu tốt và ổn định. Quan sát trực quan cho thấy ảnh sau khôi phục giảm rõ hạt nhiễu, đồng thời giữ được cấu trúc chính của đối tượng (khuôn mặt, biên vùng sáng/tối và bố cục nền). Tuy nhiên, do tối ưu theo hướng tái tạo (reconstruction-based), ảnh đầu ra đôi khi có xu hướng **mịn hoá nhẹ** ở các vùng texture nhỏ.

Kết quả demo/đánh giá (validation):

- **PSNR** tăng dần theo epoch và đạt xấp xỉ **~29.8–29.9 dB** ở giai đoạn cuối.
- **MSE** giảm và ổn định quanh **~0.0010**.

- Trên các mẫu minh hoạ, PSNR của Autoencoder đạt khoảng: **28.71 dB**, **30.09 dB**, **27.90 dB**, **31.41 dB**.

Nhận xét:

- PSNR cao (~30 dB) và MSE thấp (~0.001) cho thấy Autoencoder **tái tạo ảnh sạch hiệu quả** trong điều kiện nhiễu σ=0.05.
- Kết quả nhìn chung **ít biến dạng và ít artefact**, phù hợp cho các bài toán yêu cầu đầu ra ổn định, trung thực với ảnh gốc.

3.2.2 Nhận xét kết quả GAN

Trong cấu hình huấn luyện hiện tại, mô hình GAN cho kết quả khử nhiễu **chưa tốt bằng Autoencoder**. Quan sát trực quan cho thấy ảnh khôi phục thường xuất hiện **artefact/biến dạng texture** (vệt bất thường, vùng chi tiết bị "bệt" hoặc biến dạng), khiến mức độ tương đồng với ảnh gốc giảm.

Kết quả demo/đánh giá (validation):

- **PSNR** của GAN dao động quanh **~23–24 dB**, cuối quá trình đạt khoảng **~24.0–24.1 dB**.
- **MSE** giảm nhưng vẫn tương đối cao, ổn định quanh **~0.0039–0.0040**.
- Trên các mẫu minh hoạ, PSNR của GAN khoảng: **21.52 dB**, **22.72 dB**, **20.53 dB**, **22.08 dB**.

Nhận xét:

- PSNR thấp hơn đáng kể và MSE cao hơn phản ánh chất lượng khôi phục **kém ổn định** so với Autoencoder.
- Từ đường cong loss, Discriminator có xu hướng giảm đều trong khi Generator loss tăng về cuối, gợi ý trạng thái đối kháng **chưa cân bằng**, dễ dẫn đến việc Generator tạo artefact thay vì tái tạo trung thực ảnh sạch.

3.2.3 So sánh tổng hợp Autoencoder và GAN

Tổng hợp định lượng và định tính cho thấy Autoencoder vượt trội trong thí nghiệm này:

- **Autoencoder:** PSNR ≈ **~29.8–29.9 dB**, MSE ≈ **~0.0010** → khử nhiễu tốt, ổn định, ít artefact.
- **GAN:** PSNR ≈ **~24.0–24.1 dB**, MSE ≈ **~0.0039–0.0040** → chất lượng thấp hơn, xuất hiện artefact và biến dạng chi tiết.

Như vậy, với mức nhiễu **σ=0.05** và quy trình huấn luyện hiện tại, **Autoencoder là lựa chọn phù hợp hơn** cho bài toán khử nhiễu trên bộ dữ liệu Thumbnails.

TÀI LIỆU THAM KHẢO

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017. (Original conference: NeurIPS 2012).

[3] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[4] C. Szegedy *et al*., “Going deeper with convolutions,” in *Proc. CVPR*, 2015.

[5] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. CVPR*, 2016.

[6] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proc. CVPR*, 2017.

[7] A. G. Howard *et al*., “MobileNets: Efficient convolutional neural networks for mobile vision applications,” *arXiv preprint arXiv:1704.04861*, 2017.

[8] M. Tan and Q. V. Le, “EfficientNet: Rethinking model scaling for convolutional neural networks,” in *Proc. ICML*, 2019.

[9] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *Proc. CVPR*, 2014.

[10] R. Girshick, “Fast R-CNN,” in *Proc. ICCV*, 2015.

[11] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” in *Proc. NeurIPS*, 2015.

[12] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask R-CNN,” in *Proc. ICCV*, 2017.

[13] I. Goodfellow *et al*., “Generative adversarial nets,” in *Proc. NeurIPS*, 2014.

[14] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, “Extracting and composing robust features with denoising autoencoders,” in *Proc. ICML*, 2008.

[15] P. Dollár, C. Wojek, B. Schiele, and P. Perona, “Pedestrian detection: A benchmark,” in *Proc. CVPR*, 2009. (Tham khảo nền tảng benchmark pedestrian; dùng để đối chiếu bối cảnh).

[16] Penn-Fudan Pedestrian Database, *Dataset description/website* (dùng làm tập dữ liệu cho detection/segmentation người đi bộ).

[17] Torchvision Documentation, *Detection models: Faster R-CNN / Mask R-CNN* (tham khảo API triển khai trong PyTorch/torchvision).

[18] GeeksforGeeks / DataCamp / AWS / NTTuan8, các bài viết tổng quan MLP/RNN/CNN (tham khảo bổ trợ kiến thức nền).