ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

NHẬP MÔN HỌC MÁY VÀ KHAI PHÁ DỮ LIỆU - IT3190

BÀI TẬP LỚN:
Nhận Diện Chữ Số Viết Tay Bằng Neural Network

GVHD: PGS.TS. Thân Quang Khoát

Sinh viên:
Trần Lê Dũng - 20224964
Lê Trường Uy - 20225113
Nguyễn Vũ Minh Đức - 20224956
Nguyễn Văn Khỏe - 20225020
Nguyễn Thế Nhật Minh - 20225045

Hà Nội, ngày 1 tháng 12 năm 2024

Mục lục

Tóm tắt nội dung

Nhận dạng chữ số viết tay là một trong những bài toán cơ bản và phổ biến trong lĩnh vực học sâu. Đây là quá trình sử dụng các mô hình học máy để phân loại hình ảnh các chữ số viết tay thành các con số tương ứng từ 0 đến 9. Một ứng dụng tiêu biểu là bộ dữ liệu MNIST, với 70.000 hình ảnh chữ số viết tay đã được chuẩn hóa. Các mô hình nhận dạng chữ số có thể được sử dụng trong nhiều lĩnh vực như tự động hóa quy trình nhập liệu, nhận diện chữ số trên hóa đơn, xử lý biển số xe, hoặc tích hợp vào các hệ thống nhận dạng ký tự quang học (OCR). Tuy nhiên, việc huấn luyện mô hình này không chỉ giúp giải quyết bài toán nhận dạng chữ số mà còn là nền tảng quan trọng để hiểu cách xây dựng và tối ưu hóa mạng nơ-ron, làm tiền đề cho các nghiên cứu và ứng dụng học sâu phức tạp hơn.

1 Giới thiệu

Bài toán này tập trung vào việc phân loại các chữ số viết tay từ hình ảnh thành các số tương ứng (0-9), với ứng dụng phổ biến trong tự động hóa xử lý dữ liệu, nhận diện ký tự quang học (OCR), và số hóa tài liệu viết tay. Bộ dữ liệu MNIST, một tập hợp gồm 70.000 hình ảnh chữ số viết tay được chuẩn hóa, thường được sử dụng như một bài toán "nhập môn"để nghiên cứu và phát triển các mô hình học máy.

Trong bài tập này, nhóm nghiên cứu tập trung phát triển một hệ thống nhận dạng chữ số viết tay từ các phương pháp cơ bản đến các kỹ thuật nâng cao. Cụ thể, nhóm bắt đầu từ các phương pháp học máy truyền thống như K-Nearest Neighbors (KNN) và SVM (Support Vector Machines), sau đó dần chuyển sang các mô hình học sâu như Mạng nơ-ron truyền thẳng (Fully Connected Networks - FCN) và Mạng nơ-ron tích chập (Convolutional Neural Networks - CNN). Việc nâng cấp từ các phương pháp cơ bản lên các mô hình hiện đại không chỉ giúp nhóm hiểu rõ hơn về từng thuật toán mà còn kiểm nghiệm hiệu quả của chúng trên tập dữ liệu MNIST.

Nhóm cũng tập trung vào các kỹ thuật tiền xử lý dữ liệu như chuẩn hóa dữ liệu và tăng cường dữ liệu (data augmentation) để cải thiện độ chính xác của các mô hình. Kết quả của bài tập không chỉ giúp so sánh hiệu năng giữa các thuật toán mà còn thể hiện tiềm năng của học sâu trong việc giải quyết các bài toán nhận diện hình ảnh. Việc xây dựng hệ thống nhận dạng chữ số viết tay này đóng vai trò quan trọng, không chỉ trong việc làm quen với các thuật toán học máy và học sâu mà còn đặt nền móng cho các ứng dụng thực tế như nhận dạng văn bản, xử lý hình ảnh, và tích hợp vào các hệ thống thông minh.

2 Dữ liệu

2.1 Tổng quan dữ liệu

Tập dữ liệu MNIST là một tập hợp nổi tiếng trong lĩnh vực học máy, được thiết kế để hỗ trợ nghiên cứu và phát triển các hệ thống nhận diện chữ số viết tay. Tập dữ liệu này bao gồm 70.000 hình ảnh chữ số viết tay từ 0 đến 9, mỗi hình ảnh có kích thước 28x28 pixel, với 60.000 mẫu dùng để huấn luyện và 10.000 mẫu dùng để kiểm thử. Các hình ảnh được chuẩn hóa dưới dạng thang độ xám, mỗi pixel mang giá trị từ 0 đến 255.

Các đặc trưng của tập dữ liệu MNIST bao gồm:

- Dữ liệu đầu vào: Là ma trận 28x28 pixel, có thể được vector hóa thành một mảng 784 chiều khi làm việc với các thuật toán không sử dụng hình ảnh gốc.
- Nhãn (labels): Là một trong 10 số nguyên từ 0 đến 9, tương ứng với chữ số xuất hiện trong hình ảnh.

- Dấu hiệu và triệu chứng: Đau ngực, khó thở, buồn nôn, nôn
- Độ cân bằng : Các lớp (0-9) được phân bố tương đối đồng đều, đảm bảo không có sự thiên lệch trong quá trình huấn luyện.

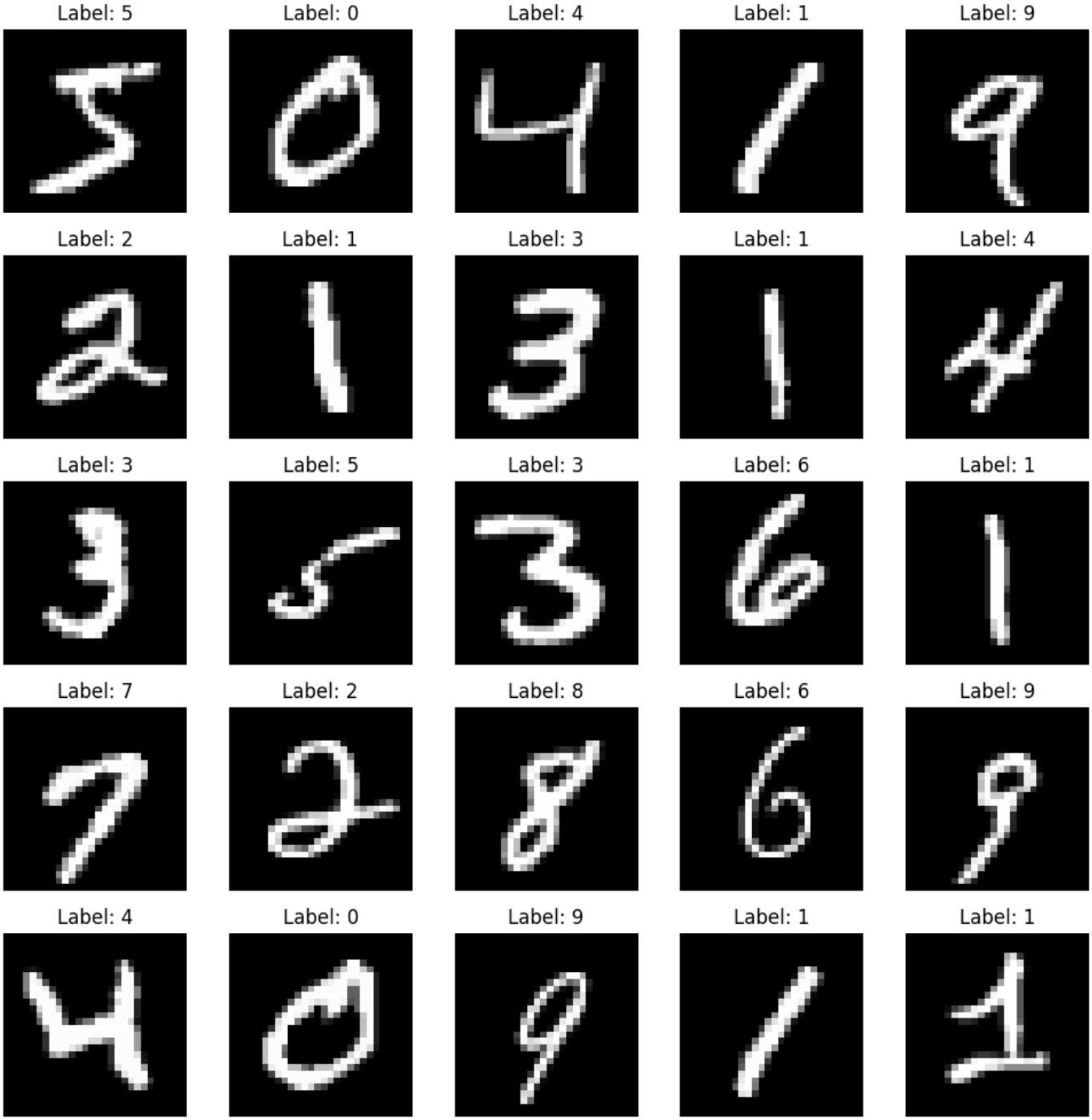
Trong bài tập này, nhóm tiến hành các bước xử lý và thử nghiệm tập dữ liệu MNIST như sau:

1. Chuẩn hóa dữ liệu : Giá trị pixel được chia cho 255 để chuẩn hóa về khoảng [0,1], giúp tăng tốc độ hội tụ của các mô hình.
2. Chọn phương pháp phân loại: Từ các thuật toán cơ bản như K-Nearest Neighbors (KNN) và SVM (Support Vector Machines), đến các mô hình học sâu như Fully Connected Networks (FCN) và Convolutional Neural Networks (CNN)
3. Tăng cường dữ liệu (augmentation): Thực hiện các phép biến đổi như xoay, co giãn, hoặc dịch chuyển hình ảnh để tăng sự đa dạng của tập dữ liệu huấn luyện.
4. Đánh giá hiệu năng : Sử dụng các chỉ số phổ biến như độ chính xác (accuracy), độ chính xác trung bình trên từng lớp (macro-averaged precision), và ma trận nhầm lẫn (confusion matrix).

Tập dữ liệu MNIST không chỉ phù hợp để so sánh hiệu năng của các thuật toán khác nhau mà còn cung cấp nền tảng cho việc nghiên cứu các phương pháp nâng cao hơn như mạng nơ-ron tísch chập hoặc thị giác máy tính phức tạp. Mục tiêu cuối cùng là xây dựng một hệ thống nhận dạng chữ số với độ chính xác cao, hỗ trợ các ứng dụng thực tế như nhận dạng ký tự quang học, xử lý hóa đơn, và số hóa dữ liệu viết tay.

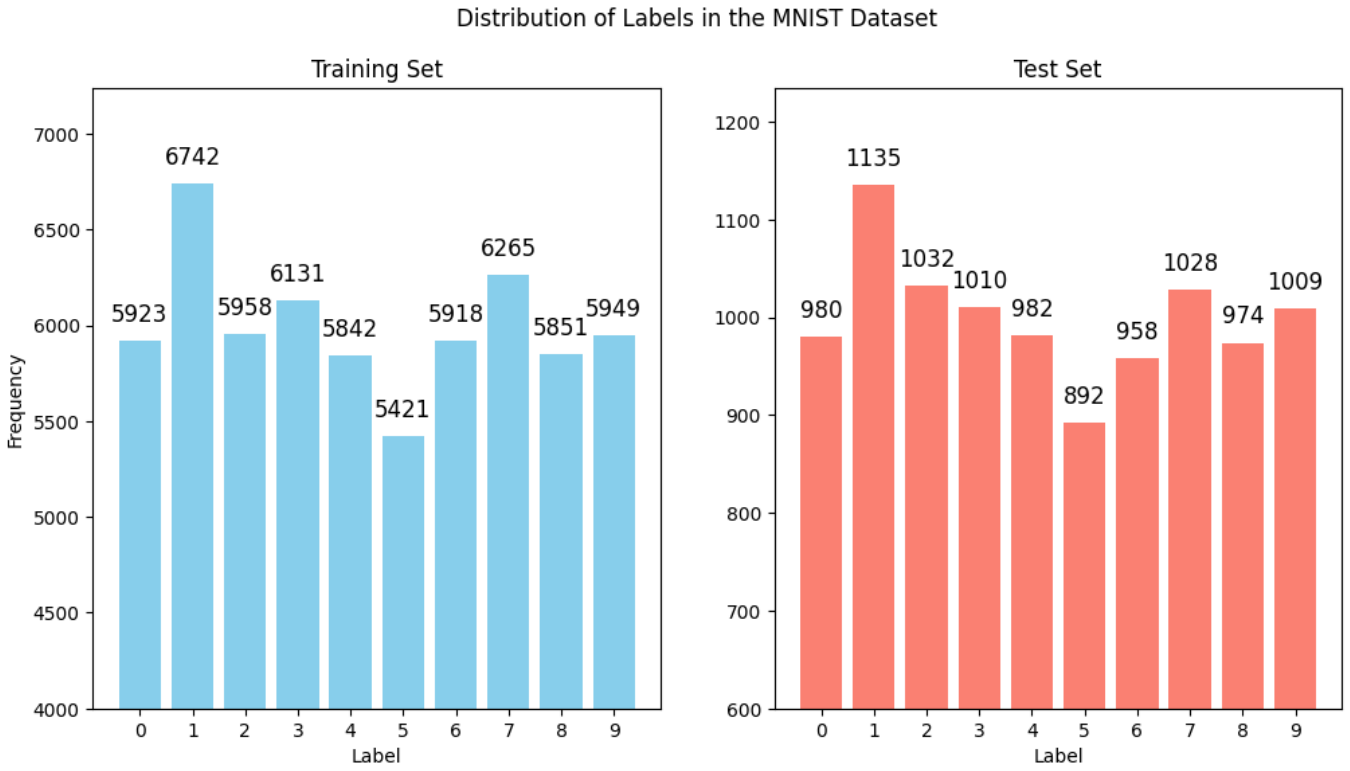
2.2 Phân tích dữ liệu

2.2.1 Dữ liệu



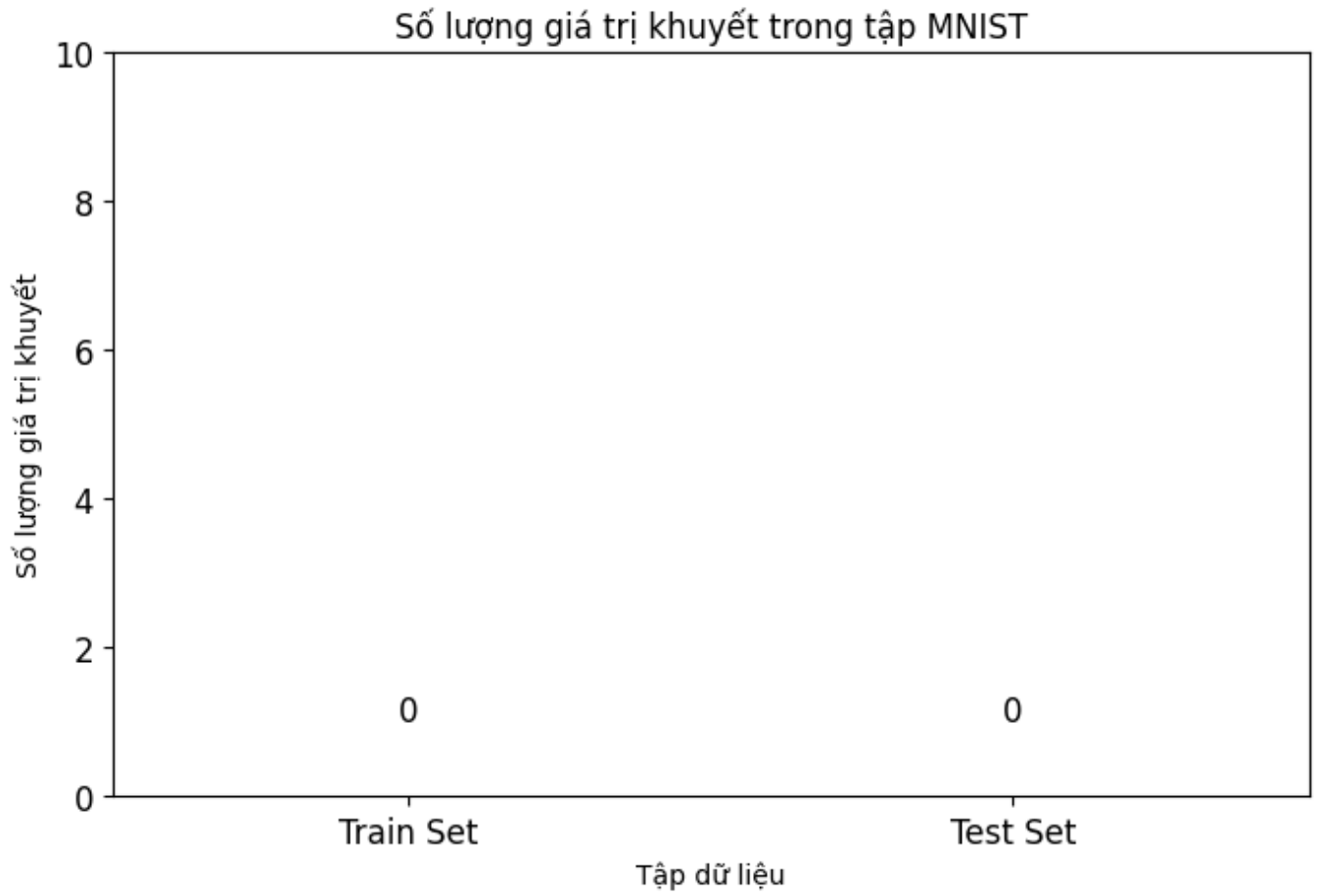
Đây là một số hình ảnh ví dụ trong bộ dữ liệu MNIST. Bộ dữ liệu này chứa hình ảnh của các chữ số viết tay, mỗi hình ảnh có kích thước 28x28 pixel. Các mẫu hình ảnh này giúp chúng ta huấn luyện các mô hình học máy để nhận dạng chữ số.

2.2.2 Độ lệch nhãn của dữ liệu



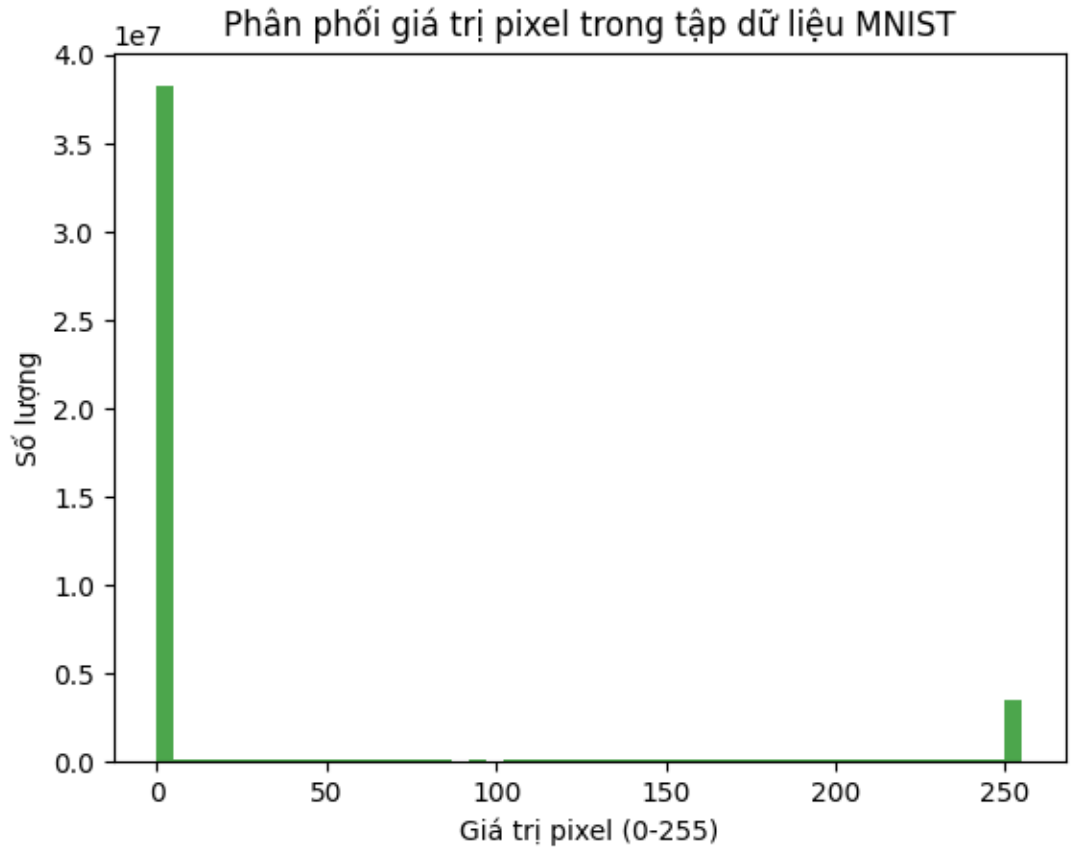
Đồ thị trên cho thấy độ lệch nhãn của bộ dữ liệu. Có thể thấy rằng độ lệch nhãn trong bộ dữ liệu là rất thấp, tức là các nhãn của dữ liệu phân bố đều trong các lớp (0-9) mà không có sự thiên lệch đáng kể. Điều này rất quan trọng trong việc huấn luyện mô hình vì nó giúp đảm bảo rằng mô hình không bị "lệch"khi học

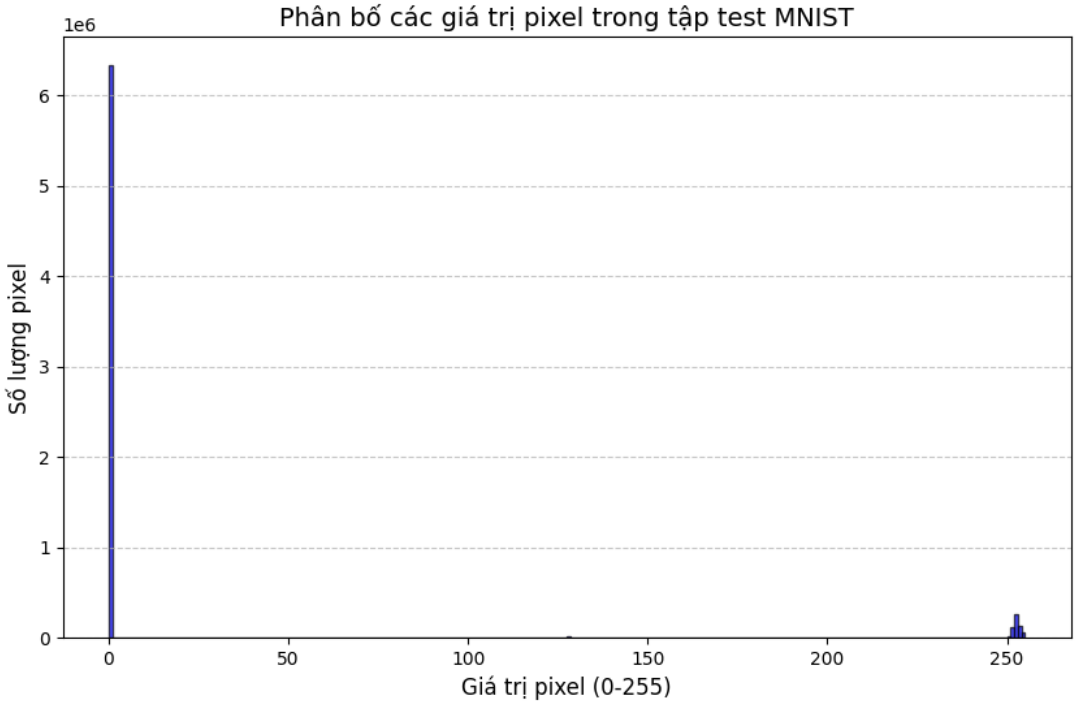
2.2.3 Dữ liệu khuyết



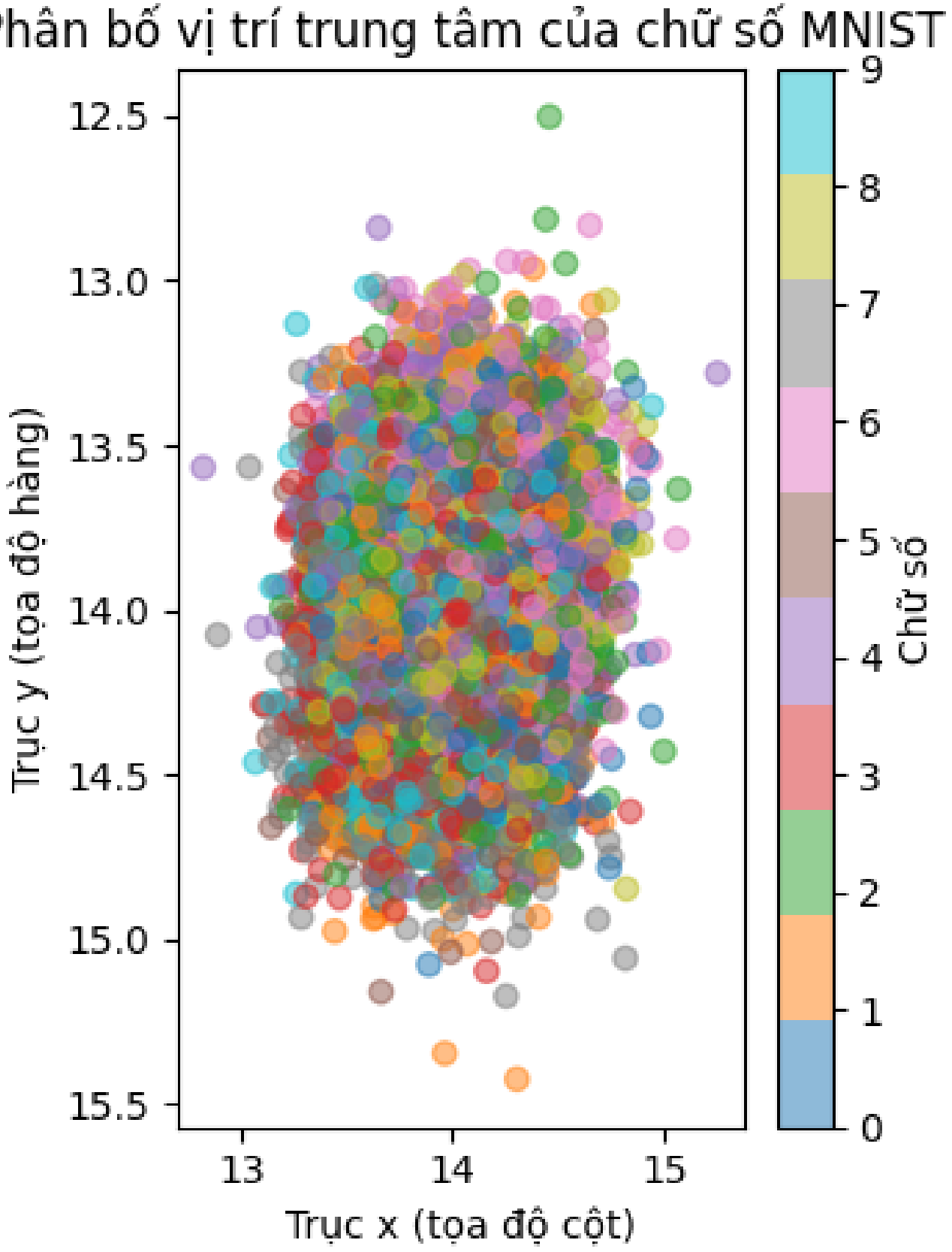
Đồ thị này cho thấy tình trạng thiếu dữ liệu (nếu có). Tuy nhiên, trong bộ dữ liệu MNIST, chúng ta không phát hiện thấy các giá trị khuyết. Điều này là một điểm mạnh, vì việc không có dữ liệu khuyết giúp mô hình học chính xác hơn và không cần phải xử lý phức tạp các giá trị thiếu.

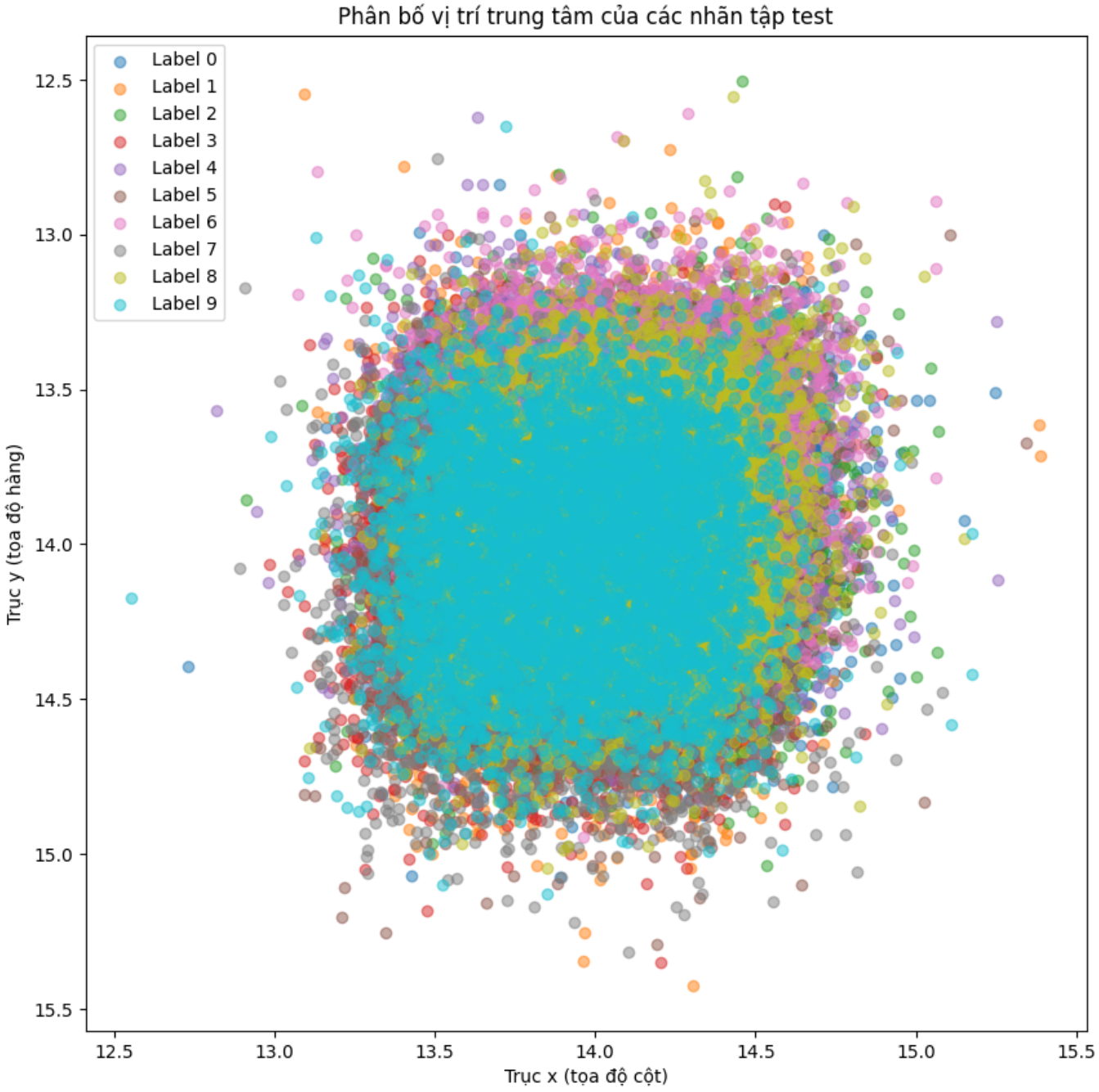
2.2.4 phân phối các thuộc tính



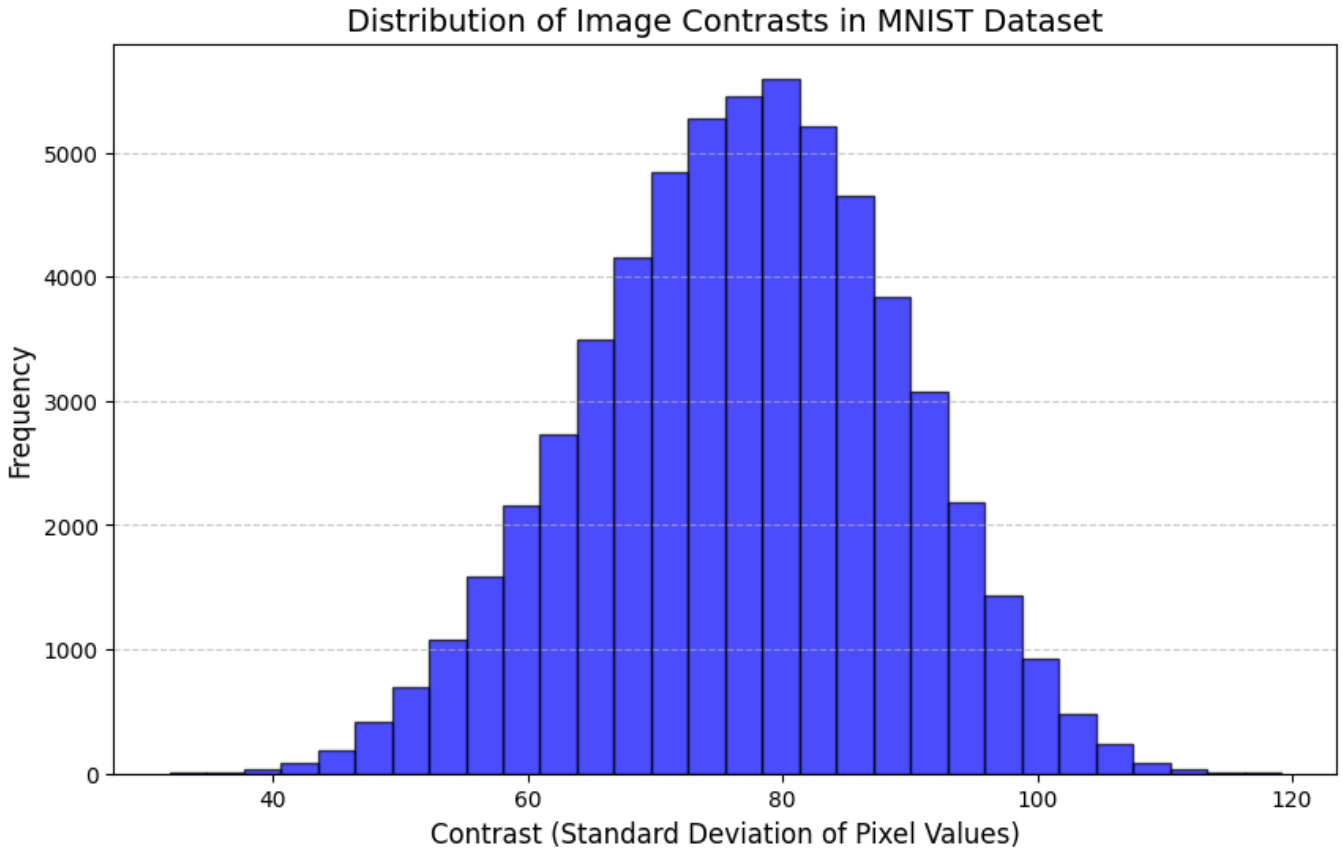


Biểu đồ trên cho thấy phân phối các thuộc tính trong bộ dữ liệu. Các giá trị pixel trong hình ảnh nằm trong khoảng từ 0 đến 255, và chúng ta có thể thấy sự phân bố của các giá trị này tương đối đều trong bộ dữ liệu. Điều này cho phép mô hình dễ dàng học cách nhận dạng chữ số mà không gặp khó khăn với sự biến thiên trong các giá trị pixel.


Phân bố vị trí trung tâm của chữ số MNIST
Trục y (tọa độ hàng)
12.5
13.0
13.5
14.0
14.5
15.0
15.5
13
14
15
Trục x (tọa độ cột)
Chữ số
0
1
2
3
4
5
6
7
8
9


Phân bố vị trí trung tâm của các nhãn tập test
Label 0
Label 1
Label 2
Label 3
Label 4
Label 5
Label 6
Label 7
Label 8
Label 9
Trục y (tọa độ hàng)
Trục x (tọa độ cột)
12.5
13.0
13.5
14.0
14.5
15.0
15.5

2.2.5 độ tương phản giữa các thuộc tính



$$\text{Contrast} = \sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

Biểu đồ độ tương phản này cho thấy độ lệch chuẩn của các giá trị pixel trong hình ảnh. Độ tương phản là một chỉ số quan trọng để đo lường sự phân bố ánh sáng và tối trong hình ảnh. Mức độ tương phản cao thường giúp hình ảnh dễ dàng phân biệt, trong khi độ tương phản thấp có thể gây khó khăn cho các mô hình nhận dạng. Đối với bộ dữ liệu MNIST, độ tương phản trung bình khá đồng đều, tạo điều kiện tốt cho việc nhận diện chữ số.

3 Tổng quan

3.1 Mạng Fully Connected (FC) Layer

Mạng Fully Connected (FC) là một loại mạng nơ-ron trong đó mỗi nơ-ron của một lớp được kết nối với tất cả các nơ-ron của lớp kế tiếp. Đây là thành phần cơ bản trong các mạng nơ-ron truyền thống và mạng nơ-ron sâu (DNN). Mỗi nơ-ron trong lớp này sẽ nhận đầu vào từ tất cả các nơ-ron ở lớp trước và truyền tín hiệu ra lớp sau. Các kết nối giữa các nơ-ron có trọng số riêng biệt và được học qua quá trình huấn luyện.

Cấu trúc của Mạng Fully Connected Layer Giả sử mạng có L lớp, trong đó lớp thứ l có n_l nơ-ron và lớp thứ $l-1$ có n_{l-1} nơ-ron. Mỗi nơ-ron trong lớp l nhận đầu vào từ tất cả các nơ-ron của

lớp $l-1$, và truyền tín hiệu tới lớp $l+1$. Các kết nối giữa các nơ-ron có trọng số riêng biệt $W^{(l)}$ và mỗi lớp có một bias $b^{(l)}$.

Công thức tính toán Giả sử lớp l có n_l nơ-ron và lớp $l-1$ có n_{l-1} nơ-ron. Đầu vào của lớp l là một vector $\mathbf{x}^{(l-1)}$, với mỗi $x_i^{(l-1)}$ là giá trị của nơ-ron tại lớp $l-1$. Trọng số kết nối giữa lớp $l-1$ và lớp l là ma trận trọng số $W^{(l)}$ có kích thước $n_l \times n_{l-1}$, và vector bias $b^{(l)}$ có kích thước n_l.

Công thức tính toán đầu ra của lớp l là:

$$\mathbf{z}^{(l)} = W^{(l)} \cdot \mathbf{x}^{(l-1)} + b^{(l)}$$

Trong đó, $\mathbf{z}^{(l)}$ là vector đầu ra chưa qua hàm kích hoạt (pre-activation).

Sau khi tính toán, đầu ra của lớp l là:

$$\mathbf{a}^{(l)} = f(\mathbf{z}^{(l)})$$

Trong đó, f là hàm kích hoạt, ví dụ như hàm ReLU, sigmoid, hoặc softmax, tùy thuộc vào bài toán.

Ví dụ về mạng Fully Connected Giả sử bạn có một mạng với ba lớp:

- Lớp đầu vào có 4 nơ-ron.
- Lớp ẩn có 3 nơ-ron.
- Lớp đầu ra có 2 nơ-ron (phù hợp cho bài toán phân loại nhị phân).

Công thức tính toán cho mạng này có thể mô tả như sau:
- Đầu vào $\mathbf{x} = [x_1, x_2, x_3, x_4]$. - Lớp ẩn có đầu ra $\mathbf{z}^{(1)} = W^{(1)} \cdot \mathbf{x} + b^{(1)}$, sau đó tính toán $\mathbf{a}^{(1)} = f(\mathbf{z}^{(1)})$.
- Lớp đầu ra có đầu ra $\mathbf{z}^{(2)} = W^{(2)} \cdot \mathbf{a}^{(1)} + b^{(2)}$, và cuối cùng tính toán $\mathbf{a}^{(2)} = f(\mathbf{z}^{(2)})$.

Ứng dụng Mạng fully connected được sử dụng rộng rãi trong các bài toán phân loại và hồi quy, đặc biệt là trong các mạng nơ-ron sâu (Deep Neural Networks, DNNs). Chúng cũng là một phần quan trọng của các mạng nơ-ron tích chập (CNN) và các mạng nơ-ron hồi tiếp (RNN), nơi lớp fully connected thường được sử dụng ở phần cuối của mạng để phân loại kết quả.

Mạng FC thường được áp dụng trong các bài toán nhận dạng hình ảnh, nhận dạng văn bản, và các ứng dụng khác trong học sâu.

3.2 Mạng Nơ-ron Tích chập (CNN)

Mạng Nơ-ron Tích chập (Convolutional Neural Network - CNN) là một loại mạng nơ-ron đặc biệt được thiết kế để xử lý dữ liệu có cấu trúc không gian như hình ảnh, video, hoặc tín hiệu thời gian. CNN đã đạt được nhiều thành tựu ấn tượng trong các bài toán nhận dạng hình ảnh, phân loại, phát hiện vật thể và nhiều ứng dụng khác.

Cấu trúc CNN Cấu trúc cơ bản của CNN gồm một loạt các lớp chập (convolutional layers) và các lớp pooling, với một hoặc nhiều lớp fully connected (FC) ở cuối để đưa ra kết quả dự đoán. Các thành phần chính của CNN bao gồm:

- **Lớp chập (Convolutional Layer):** Lớp chập là thành phần quan trọng nhất trong CNN. Lớp này sử dụng các bộ lọc (filters) hoặc kernel để quét qua dữ liệu đầu vào (chẳng hạn như hình ảnh), và tạo ra các ma trận đặc trưng (feature maps) của các đặc điểm như cạnh, hình dạng, và kết cấu trong hình ảnh.

- **Lớp pooling (Pooling Layer):** Lớp pooling có nhiệm vụ giảm kích thước không gian của dữ liệu (chẳng hạn, giảm độ phân giải của hình ảnh) để giảm bớt chi phí tính toán và tránh overfitting. Hai loại pooling phổ biến là Max Pooling và Average Pooling.
- **Lớp Fully Connected (FC):** Sau khi qua các lớp chập và pooling, dữ liệu được đưa qua một hoặc nhiều lớp fully connected, nơi mỗi neuron được kết nối với tất cả các neuron của lớp tiếp theo. Lớp FC giúp mạng học các đặc trưng trừu tượng và thực hiện phân loại hoặc dự đoán.
- **Lớp kích hoạt (Activation Function):** Các lớp chập và FC thường đi kèm với một hàm kích hoạt như ReLU (Rectified Linear Unit) hoặc Sigmoid để tạo ra tính phi tuyến cho mô hình.

Quá trình hoạt động của CNN Quá trình hoạt động của CNN có thể được mô tả qua các bước sau:

1. **Lớp Chập (Convolutional Layer):** Dữ liệu đầu vào (ví dụ: hình ảnh) sẽ được quét qua các bộ lọc. Mỗi bộ lọc sẽ tìm kiếm các đặc trưng trong hình ảnh như các cạnh, góc, và kết cấu.
2. **Lớp Pooling (Pooling Layer):** Sau mỗi lớp chập, lớp pooling được sử dụng để giảm kích thước của các đặc trưng đã phát hiện, từ đó giảm chi phí tính toán và giúp mô hình dễ dàng tổng quát hơn.
3. **Lớp Fully Connected (FC):** Các đặc trưng đã được xử lý qua các lớp chập và pooling sẽ được truyền vào các lớp fully connected để thực hiện các nhiệm vụ như phân loại hoặc dự đoán.

Các ưu điểm của CNN : Mạng Nơ-ron Tích chập có những ưu điểm sau:

- **Tính hiệu quả với dữ liệu hình ảnh:** CNN có khả năng tự động học các đặc trưng trong hình ảnh mà không cần sự can thiệp của con người.
- **Chia sẻ trọng số (Weight Sharing):** Các bộ lọc trong lớp chập được áp dụng trên toàn bộ hình ảnh, giúp giảm số lượng tham số và chi phí tính toán.
- **Tính bất biến (Invariant):** CNN có thể phát hiện các đặc trưng trong hình ảnh bất kể chúng ở đâu trong hình ảnh (đặc biệt là với sự kết hợp của các lớp pooling).

Ứng dụng của CNN Mạng Nơ-ron Tích chập đã được áp dụng thành công trong nhiều lĩnh vực, bao gồm:

- **Nhận dạng hình ảnh (Image Recognition):** CNN có thể phân loại và nhận dạng các đối tượng trong hình ảnh, như nhận dạng các chữ số, khuôn mặt, hoặc vật thể trong ảnh.
- **Phát hiện vật thể (Object Detection):** CNN cũng có thể phát hiện vị trí của các đối tượng trong hình ảnh, ví dụ như trong các ứng dụng nhận dạng biển số xe.
- **Phân tích video (Video Analysis):** CNN được sử dụng để nhận dạng các chuyển động và sự kiện trong video.
- **Y học (Medical Imaging):** CNN giúp phát hiện và chẩn đoán bệnh trong các hình ảnh y khoa, chẳng hạn như MRI hoặc X-ray.

Ví dụ về CNN Một ví dụ cơ bản về cấu trúc CNN có thể bao gồm:

- **Lớp chập đầu tiên:** Nhận một hình ảnh đầu vào và áp dụng các bộ lọc để tìm các đặc trưng như cạnh và kết cấu.
- **Lớp pooling:** Sử dụng max pooling hoặc average pooling để giảm kích thước của đặc trưng.
- **Lớp chập thứ hai:** Áp dụng các bộ lọc khác để tìm các đặc trưng phức tạp hơn từ các đặc trưng đã được tìm thấy trong lớp trước.
- **Lớp fully connected:** Cuối cùng, các đặc trưng được đưa vào một lớp fully connected để thực hiện phân loại.

Mô hình CNN mạnh mẽ nhờ khả năng tự động học và nhận diện các đặc trưng quan trọng trong dữ liệu hình ảnh mà không cần sự can thiệp thủ công.

4 So sánh giữa Mạng Nơ-ron Tích chập (CNN) và Mạng Nơ-ron Toàn Kết Nối (FCN)

Trong học sâu, CNN và FCN đều là các kiến trúc mạng nơ-ron phổ biến, nhưng chúng có sự khác biệt rõ rệt về cấu trúc và ứng dụng. Dưới đây là một so sánh giữa hai loại mạng này:

4.1 Mạng Nơ-ron Tích chập (CNN)

Ưu điểm:

- **Phân tích dữ liệu hình ảnh hiệu quả:** CNN đặc biệt hiệu quả trong việc xử lý và phân loại dữ liệu hình ảnh nhờ vào các lớp tích chập (convolutional layers), giúp phát hiện các đặc trưng của hình ảnh mà không cần phải thiết kế thủ công.
- **Tính chia sẻ trọng số:** Các bộ lọc trong CNN chia sẻ trọng số, giúp giảm thiểu số lượng tham số và tiết kiệm tài nguyên tính toán.
- **Khả năng học đặc trưng tự động:** CNN có thể học các đặc trưng hình ảnh tự động từ dữ liệu mà không cần người dùng phải xác định trước.

Hạn chế:

- **Cần nhiều dữ liệu huấn luyện:** CNN yêu cầu một lượng lớn dữ liệu để huấn luyện một cách hiệu quả và tránh overfitting.
- **Đòi hỏi tài nguyên tính toán lớn:** Việc huấn luyện CNN yêu cầu phần cứng mạnh mẽ, đặc biệt là GPU, và thời gian huấn luyện có thể lâu.
- **Khó giải thích:** Mạng CNN có thể rất phức tạp và khó giải thích, đặc biệt khi các lớp học các đặc trưng trừu tượng.

4.2 Mạng Nơ-ron Toàn Kết Nối (FCN)

Ưu điểm:

- **Cấu trúc đơn giản:** FCN có cấu trúc đơn giản hơn so với CNN, với mỗi lớp đều có kết nối đầy đủ giữa các neuron, điều này giúp học các mối quan hệ phức tạp giữa các đầu vào.
- **Hiệu quả trong các bài toán dự đoán và phân loại:** FCN có thể được sử dụng cho các bài toán phân loại, hồi quy hoặc dự đoán trong các lĩnh vực như tài chính, y tế, và sinh học.
- **Không cần quá nhiều dữ liệu huấn luyện:** FCN không yêu cầu lượng dữ liệu lớn như CNN để đạt được kết quả tốt trong các bài toán đơn giản.

Hạn chế:

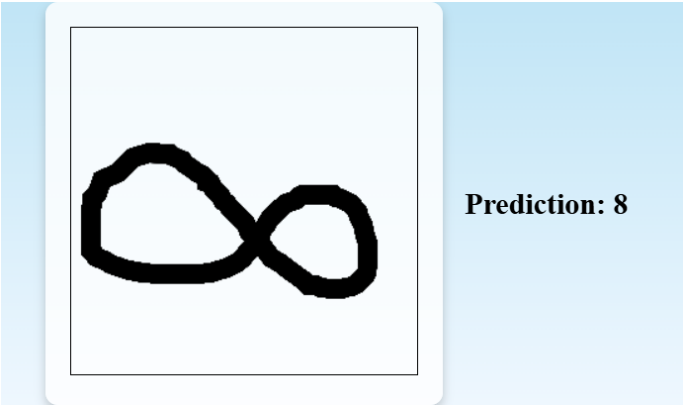
- **Không hiệu quả với dữ liệu hình ảnh phức tạp:** Mặc dù có thể áp dụng cho các bài toán hình ảnh, FCN không thể hiệu quả như CNN trong việc nhận dạng hình ảnh phức tạp.
- **Dễ bị overfitting:** Vì tất cả các lớp đều kết nối đầy đủ, FCN dễ dàng bị overfitting nếu không sử dụng các phương pháp điều chỉnh (regularization) như dropout.
- **Không học được đặc trưng từ dữ liệu:** FCN không thể tự động học các đặc trưng đặc trưng của hình ảnh như CNN, do đó thường yêu cầu các đặc trưng đã được thiết kế từ trước.

4.3 So sánh CNN và FC

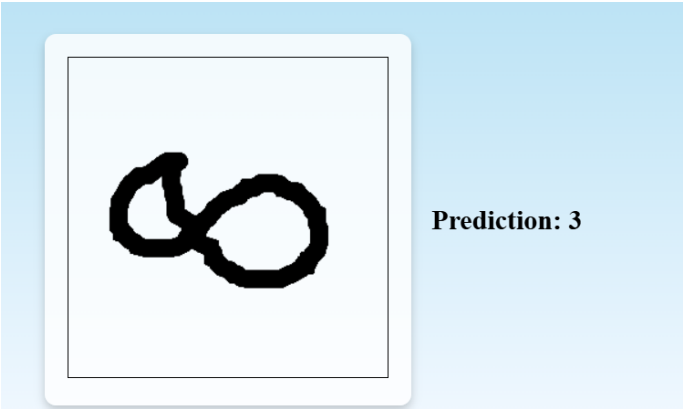
Tiêu chí	CNN	FCN
Ứng dụng chính	Xử lý hình ảnh, nhận diện hình ảnh	Dự đoán, phân loại, hồi quy
Cấu trúc	Các lớp tích chập, lớp pooling, lớp fully connected	Các lớp fully connected, không có lớp tích chập
Dữ liệu huấn luyện	Cần dữ liệu lớn và đa dạng	Có thể làm việc với dữ liệu ít hơn
Tài nguyên tính toán	Cần GPU và tài nguyên tính toán mạnh mẽ	Thấp hơn so với CNN
Khả năng học đặc trưng tự động	Có khả năng học đặc trưng từ dữ liệu	Không học đặc trưng tự động
Khả năng giải thích	Khó giải thích vì các lớp trừu tượng	Dễ hiểu và giải thích hơn

Bảng 2: So sánh giữa Mạng Nơ-ron Tích chập (CNN) và Mạng Nơ-ron Toàn Kết Nối (FCN)

Tính bất biến trong không gian:



Dự đoán của mô hình CNN đối với số 8 nằm ngang.



Dự đoán của mô hình FC đối với ố 8 nằm ngang.

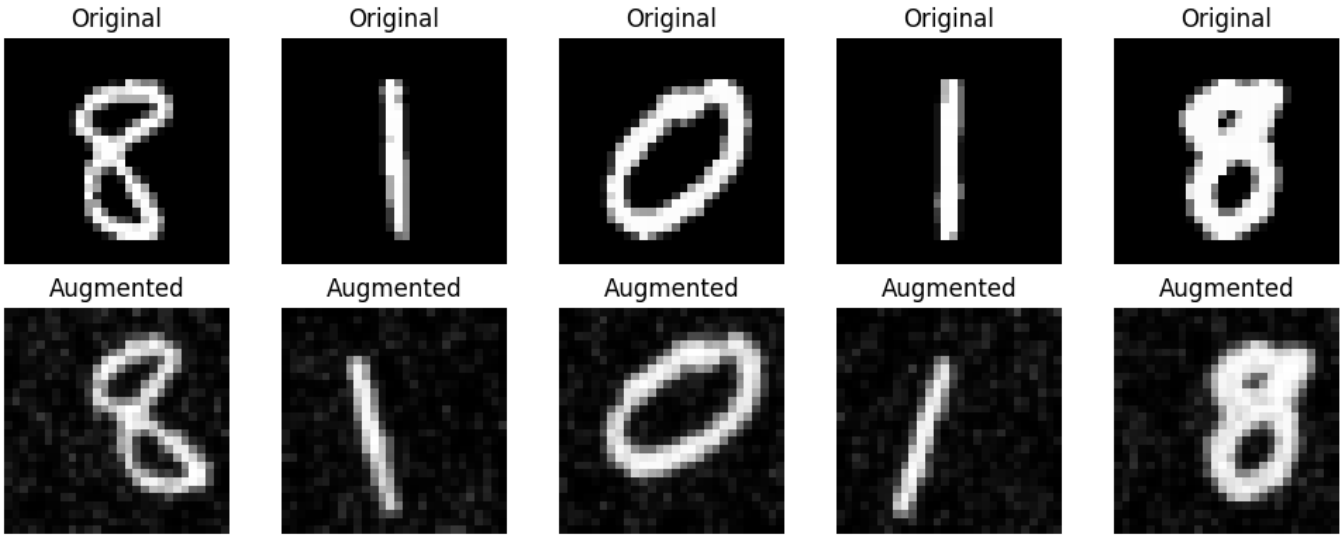
Kết luận Tóm lại, CNN và FCN đều là những mạng nơ-ron mạnh mẽ nhưng được thiết kế cho các mục đích khác nhau. CNN thích hợp cho các bài toán liên quan đến nhận dạng hình ảnh và học các đặc trưng từ dữ liệu, trong khi FCN có cấu trúc đơn giản hơn và phù hợp với các bài toán dự đoán và phân loại không đụng phải dữ liệu hình ảnh phức tạp. Lựa chọn giữa CNN và FCN phụ thuộc vào loại bài toán, tài nguyên tính toán và dữ liệu có sẵn.

5 Thuật toán đề xuất

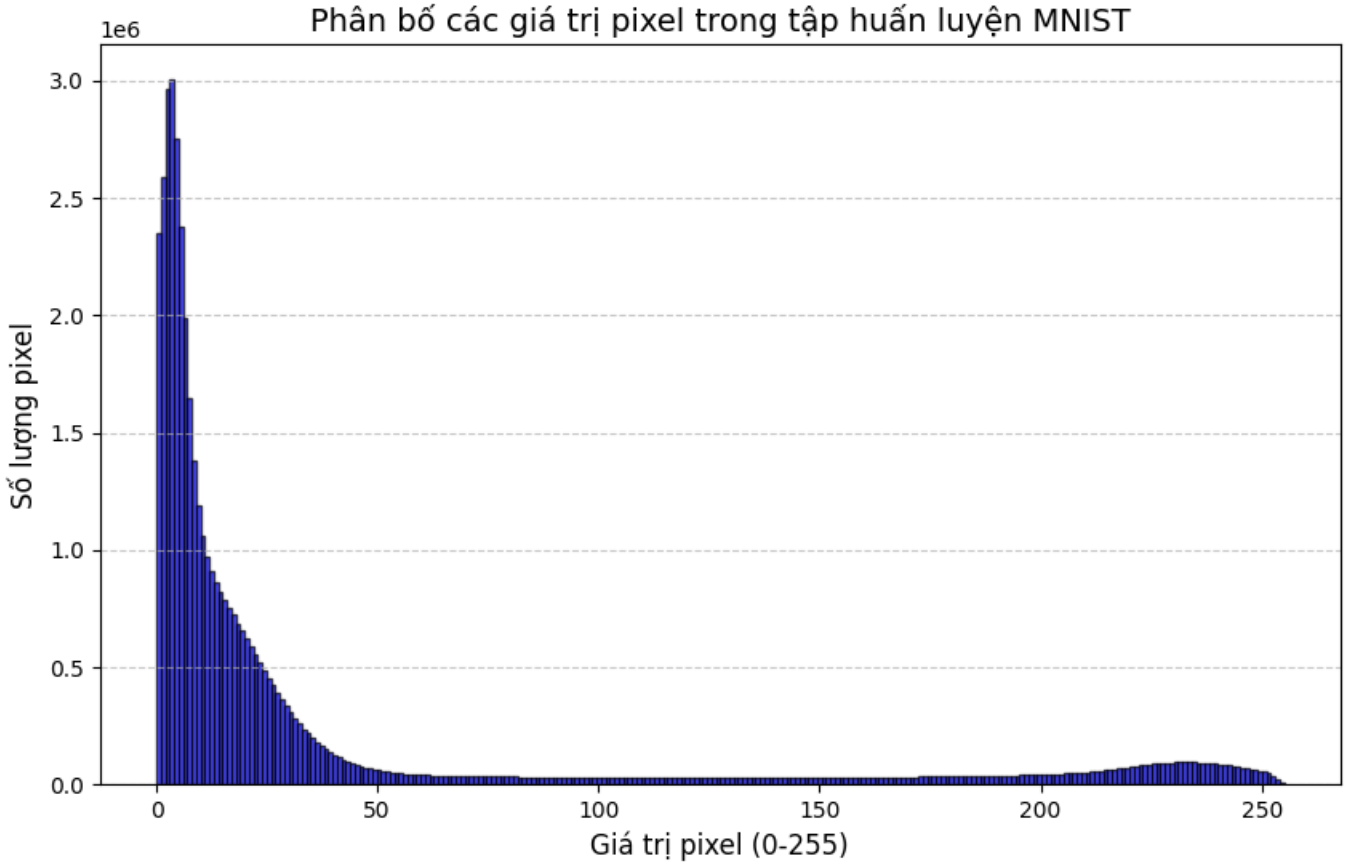
5.1 Tiền xử lý dữ liệu

5.1.1 Vấn đề đối với dữ liệu ban đầu

Dựa trên các hình ảnh trực quan về dữ liệu ở phần 2.2 cho thấy được dữ liệu ban đầu rất đẹp, không có hiện tượng mất cân đối dữ liệu và độ phân bố giữa các số trong bức ảnh nằm ở chính giữa nhưng để hướng đến một mô hình mạnh hơn và sát với thực tế hơn thì cần thiết phải căn chỉnh một chút để có thể tổng quát hóa cao hơn như thêm nhiễu , dịch chuyển chữ số sang các góc của hình ảnh,tạo độ nghiêng trong khoảng -30 đến 30 độ :



Độ phân bố pixed trên tập dữ liệu mới :



5.1.2 Với từng mô hình

Các mô hình KNN, SVM, FCN và CNN đều có yêu cầu khác nhau về cách xử lý dữ liệu đầu vào từ bộ dữ liệu MNIST. Dưới đây là sự khác biệt cụ thể:

- **FCN (Fully Connected Network):**
  - Tương tự như KNN và SVM, dữ liệu cũng cần được làm phẳng thành vector 784.
  - Việc chuẩn hóa dữ liệu về khoảng $[0, 1]$ cũng được áp dụng để cải thiện hiệu quả huấn luyện.

- **CNN (Convolutional Neural Network):**
  - Không yêu cầu làm phẳng dữ liệu đầu vào. CNN hoạt động trực tiếp trên dữ liệu ở định dạng ma trận $28 \times 28 \times 1$ (thêm chiều thứ ba để biểu diễn kênh màu, trong trường hợp này là ảnh grayscale với 1 kênh).
  - Sau khi qua các lớp tích chập (*convolutional layers*) và gộp (*pooling layers*), dữ liệu mới được làm phẳng thành một vector để đưa vào các lớp fully connected (nếu có).
  - Dữ liệu đầu vào cũng được chuẩn hóa về khoảng $[0, 1]$, nhưng việc làm phẳng diễn ra bên trong kiến trúc mạng chứ không yêu cầu xử lý trước từ phía người dùng.

Tóm lại: Trong khi KNN, SVM,... và FCN yêu cầu dữ liệu đầu vào phải được làm phẳng và chuẩn hóa trước khi đưa vào mô hình, CNN có thể làm việc trực tiếp với dữ liệu ở dạng ma trận. Tuy nhiên, dữ liệu cần được chuẩn hóa trong mọi trường hợp để tối ưu hóa quá trình huấn luyện.

5.2 Các vấn đề dẫn đến dùng Neural Network

Các mô hình học máy truyền thống như KNN, SVM có hiệu quả trong những bài toán phân loại có số lượng đặc trưng nhỏ và dữ liệu vừa phải. Tuy nhiên, trong bài toán phân loại ảnh, dữ liệu thường có kích thước lớn (như hình ảnh 28x28 từ MNIST, tương đương 784 đặc trưng khi làm phẳng) và có tính chất không gian (spatial features), khiến các mô hình này gặp một số hạn chế:

- **Kích thước dữ liệu lớn:** Với mỗi hình ảnh có độ phân giải cao, số lượng đặc trưng có thể lên đến hàng nghìn hoặc hàng triệu. KNN và SVM truyền thống không thể xử lý hiệu quả với số lượng đặc trưng lớn này, đặc biệt là khi phải tính toán khoảng cách hoặc độ tương tự trong không gian đặc trưng lớn.
- **Tính chất không gian của ảnh:** Các mô hình như KNN và SVM không tận dụng được mối quan hệ không gian giữa các pixel trong ảnh. Trong khi đó, ảnh có sự phụ thuộc không gian mạnh mẽ (như sự liên kết giữa các pixel gần nhau), điều mà các mô hình học sâu như CNN có thể khai thác hiệu quả.
- **Hiệu quả tính toán:** KNN cần tính toán khoảng cách giữa điểm dữ liệu và tất cả các điểm khác trong tập huấn luyện, điều này gây tốn kém về mặt tính toán, đặc biệt khi số lượng dữ liệu rất lớn. Tương tự, SVM với hàm hạt nhân có thể gặp khó khăn trong việc xử lý tập dữ liệu lớn, khiến việc huấn luyện trở nên chậm và tốn tài nguyên.
- **Khả năng tổng quát:** Các mô hình truyền thống như KNN và SVM dễ bị overfitting khi xử lý dữ liệu phức tạp như ảnh, trong khi các mô hình học sâu (như CNN) có khả năng tổng quát tốt hơn nhờ vào các kỹ thuật như regularization và pooling.

Vì các lý do trên, việc áp dụng các mô hình học máy truyền thống như KNN và SVM vào bài toán phân loại ảnh không mang lại hiệu quả tối ưu. Các mô hình học sâu như CNN (Convolutional Neural Network) đã chứng tỏ khả năng vượt trội trong việc khai thác các đặc trưng không gian và học được các biểu diễn mạnh mẽ từ dữ liệu ảnh.

5.2.1 Nhận xét

Như vậy với tập dữ liệu mới dự đoán sẽ đòi hỏi một mô hình phức tạp và có khả năng tổng hợp hóa cao hơn

5.3 Các kỹ thuật giảm Overfiting

5.3.1 L2 Regularization

L2 regularization, còn được gọi là *Ridge Regularization* trong hồi quy tuyến tính, là một kỹ thuật thêm vào mục tiêu tối ưu hóa một thuật toán học máy để ngăn chặn hiện tượng overfitting. Kỹ thuật này thực hiện việc phạt các trọng số của mô hình bằng cách cộng một hình thức phạt vào hàm mất mát.

Giả sử chúng ta có một mô hình hồi quy tuyến tính, nơi $\mathbf{X}$ là ma trận đặc trưng (features) và $\mathbf{y}$ là vectơ nhãn (labels), hàm mất mát trong hồi quy tuyến tính là:

$$\mathcal{L} = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Trong đó y_i là nhãn thực tế và $\hat{y}_i = \mathbf{x}_i^T\mathbf{w} + b$ là giá trị dự đoán, với $\mathbf{w}$ là vectơ trọng số và b là hệ số chệch.

Khi áp dụng L2 regularization, ta thêm vào một phần phạt dựa trên bình phương của các trọng số $\mathbf{w}$:

$$\mathcal{L}_{\text{regularized}} = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

Ở đây, $\|\mathbf{w}\|_2^2 = \sum_{j=1}^{m} w_j^2$ là L2 norm của trọng số, và λ là tham số điều chỉnh (regularization parameter), kiểm soát độ mạnh của phạt. Giá trị λ càng lớn, sự phạt càng mạnh, và mô hình sẽ trở nên ít phức tạp hơn, giúp giảm overfitting.

L2 regularization giúp giảm thiểu giá trị tuyệt đối của các trọng số, từ đó tạo ra một mô hình đơn giản hơn và ít nhạy cảm với nhiễu trong dữ liệu.

5.3.2 Dropout

Dropout là một kỹ thuật regularization được sử dụng trong huấn luyện mạng nơ-ron để giảm thiểu overfitting. Phương pháp này hoạt động bằng cách ngẫu nhiên loại bỏ (set to zero) một phần các đơn vị (neurons) trong mỗi lớp của mạng trong mỗi bước huấn luyện, qua đó làm cho mô hình không phụ thuộc quá nhiều vào bất kỳ đơn vị nào cụ thể.

Giả sử chúng ta có một mạng nơ-ron với các lớp ẩn. Trong mỗi lần huấn luyện, đối với mỗi neuron trong lớp ẩn, ta thực hiện một phép biến đổi ngẫu nhiên: với xác suất p, ta "bỏ qua"neuron đó (set giá trị của neuron về 0), và với xác suất $1 - p$, neuron này vẫn hoạt động bình thường.

Kỹ thuật này có thể được mô tả như sau:

$$\mathbf{h}^{(l)} = f(\mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}) \cdot \mathbf{r}^{(l)}$$

Ở đây: - $\mathbf{h}^{(l)}$ là đầu ra của lớp l. - $f(\cdot)$ là hàm kích hoạt (activation function). - $\mathbf{W}^{(l)}$ và $\mathbf{b}^{(l)}$ là trọng số và hệ số chệch của lớp l. - $\mathbf{r}^{(l)}$ là vector dropout với các giá trị ngẫu nhiên trong $\{0, 1\}$, nơi mỗi phần tử có xác suất p là 0 (tức là bị loại bỏ) và xác suất $1 - p$ là 1 (tức là không bị loại bỏ).

Trong quá trình huấn luyện, dropout giúp giảm sự phụ thuộc vào các đặc trưng cụ thể của lớp ẩn và tăng cường khả năng tổng quát của mô hình. Sau khi huấn luyện, trong giai đoạn dự đoán (inference), ta sử dụng tất cả các neuron mà không áp dụng dropout nữa, và thay vì làm việc với xác suất dropout, ta nhân các trọng số với $(1 - p)$ để bù đắp cho việc dropout trong huấn luyện.

Dropout giúp tạo ra một tập hợp các mô hình con khác nhau trong mỗi bước huấn luyện, và khi kết hợp kết quả từ tất cả các mô hình này, mô hình tổng thể có khả năng tổng quát tốt hơn.

5.3.3 EarlyStop

Early Stopping là một kỹ thuật regularization trong huấn luyện mạng nơ-ron, được sử dụng để ngừng quá trình huấn luyện khi hiệu suất trên tập kiểm tra không còn cải thiện nữa, từ đó giúp ngăn chặn overfitting. Phương pháp này giúp tránh việc mô hình trở nên quá phức tạp và chỉ học thuộc dữ liệu huấn luyện mà không có khả năng tổng quát tốt với dữ liệu mới.

Giả sử chúng ta có một hàm mất mát $\mathcal{L}$ được tính trên cả tập huấn luyện và tập kiểm tra trong mỗi epoch. Trong quá trình huấn luyện, chúng ta sẽ theo dõi giá trị của hàm mất mát trên tập kiểm tra. Nếu sau một số epoch liên tiếp, giá trị hàm mất mát trên tập kiểm tra không cải thiện, hoặc bắt đầu tăng lên, huấn luyện sẽ được dừng lại.

Giới thiệu về Early Stopping trong bài toán huấn luyện mạng nơ-ron:

1. Theo dõi hiệu suất: Đầu tiên, hiệu suất của mô hình trên tập kiểm tra được theo dõi sau mỗi epoch. Giả sử $\mathcal{L}_{\text{val}}(t)$ là giá trị hàm mất mát trên tập kiểm tra tại epoch t.

2. Định nghĩa số epoch không cải thiện: Sau một số lượng epoch nhất định không có sự cải thiện (được gọi là patience), quá trình huấn luyện sẽ dừng lại. Nếu hiệu suất trên tập kiểm tra không cải thiện trong một số epoch liên tiếp, việc huấn luyện sẽ ngừng và trả về mô hình tốt nhất đã được tìm thấy.

3. Kỹ thuật Early Stopping có thể được mô tả như sau:

$$\text{if } \mathcal{L}_{\text{val}}(t) > \mathcal{L}_{\text{val}}(t-1) \text{ for some } t > T_{\text{patience}}, \text{ then stop training.}$$

Ở đây: - $\mathcal{L}_{\text{val}}(t)$ là hàm mất mát trên tập kiểm tra tại epoch thứ t. - T_{patience} là số epoch mà mô hình có thể tiếp tục huấn luyện mà không có sự cải thiện.

Lợi ích của Early Stopping: - Ngăn chặn overfitting: Khi huấn luyện tiếp tục lâu hơn cần thiết, mô hình có thể bắt đầu học các đặc điểm nhiễu và không tổng quát tốt trên dữ liệu chưa thấy. Early stopping giúp dừng huấn luyện trước khi điều này xảy ra. - Tiết kiệm thời gian huấn luyện: Khi mô hình không còn cải thiện, việc tiếp tục huấn luyện là không cần thiết. Early stopping giúp tiết kiệm thời gian và tài nguyên tính toán.

Early stopping kết hợp với các kỹ thuật regularization khác như dropout hay L2 regularization giúp nâng cao hiệu quả huấn luyện và cải thiện khả năng tổng quát của mô hình.

5.4 Các thuật toán tối ưu hóa

5.4.1 Gradient decent

Gradient Descent (GD) là thuật toán tối ưu hóa cơ bản nhất, hoạt động bằng cách sử dụng đạo hàm của hàm mất mát theo trọng số để cập nhật các trọng số theo hướng giảm dần giá trị hàm mất mát.

Cập nhật trọng số theo công thức:

$$\mathbf{w} = \mathbf{w} - \eta \nabla J(\mathbf{w})$$

Trong đó:

- $\mathbf{w}$ là trọng số của mô hình,
- η là learning rate,
- $\nabla J(\mathbf{w})$ là gradient của hàm mất mát $J(\mathbf{w})$ theo trọng số $\mathbf{w}$.

5.4.2 Gradient decent with momentum

Gradient Descent with Momentum là một cải tiến của Gradient Descent, nơi động lượng được thêm vào để giúp quá trình cập nhật trọng số ổn định hơn và tăng tốc quá trình hội tụ.

Cập nhật trọng số với động lượng:

$$v_t = \beta v_{t-1} + (1 - \beta)\nabla J(\mathbf{w})$$

$$\mathbf{w} = \mathbf{w} - \eta v_t$$

Trong đó:

- v_t là động lượng tại bước t,
- β là hệ số điều chỉnh động lượng.

5.4.3 Adam

Adam là một thuật toán tối ưu hóa kết hợp giữa **Momentum** và **RMSProp**. Adam sử dụng các ước lượng động lượng bậc 1 và bậc 2 để điều chỉnh tốc độ học cho mỗi trọng số.

Cập nhật trọng số với Adam:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\nabla J(\mathbf{w})$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)(\nabla J(\mathbf{w}))^2$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

$$\mathbf{w} = \mathbf{w} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

Trong đó:

- m_t và v_t lần lượt là các ước lượng động lượng bậc 1 và bậc 2,
- β_1 và β_2 là các hệ số điều chỉnh động lượng và ước lượng bậc 2,
- ϵ là một hằng số nhỏ để tránh chia cho 0.

5.5 Các hàm kích hoạt

5.5.1 Hàm kích hoạt Sigmoid

Hàm kích hoạt **Sigmoid** là một hàm logistic, được định nghĩa như sau:

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

Trong đó:

- e là cơ số của logarit tự nhiên,
- x là đầu vào của hàm sigmoid.

Hàm sigmoid có giá trị đầu ra nằm trong khoảng $(0, 1)$, thường được sử dụng cho các bài toán phân loại nhị phân.

5.5.2 Hàm Kích Hoạt ReLU (Rectified Linear Unit)

Hàm kích hoạt **ReLU** là một trong những hàm kích hoạt phổ biến trong học sâu, được định nghĩa như sau:

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

Trong đó:

- x là đầu vào của hàm ReLU.

Hàm ReLU giúp tránh vấn đề "vanishing gradient"và có xu hướng giúp mô hình hội tụ nhanh hơn.

5.5.3 Hàm Kích Hoạt Softmax

Hàm kích hoạt **Softmax** thường được sử dụng trong các bài toán phân loại đa lớp. Hàm softmax chuyển đổi một vector giá trị đầu vào thành một xác suất, với tổng của các giá trị xác suất là 1. Nó được định nghĩa như sau:

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}$$

Trong đó:

- x_i là phần tử thứ i trong vector đầu vào,
- N là số lượng phần tử trong vector đầu vào.

Hàm Softmax thường được sử dụng trong lớp cuối cùng của mạng nơ-ron để tính xác suất phân loại cho mỗi lớp trong bài toán phân loại đa lớp.

5.6 Loss Function

Cross-Entropy Loss là một hàm loss phổ biến trong các bài toán phân loại, đặc biệt là khi đầu ra được biểu diễn dưới dạng xác suất. Công thức tổng quát của hàm Cross-Entropy Loss như sau:

$$L = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{C} y_{ij} \log(\hat{y}_{ij})$$

Trong đó:

- N: Tổng số mẫu trong tập dữ liệu.
- C: Số lượng lớp (categories).
- y_{ij}: Nhãn thực tế (ground truth), là một giá trị nhị phân (0 hoặc 1) biểu thị lớp đúng của mẫu i trong lớp j.
- $\hat{y}_{ij}$: Xác suất dự đoán của lớp j cho mẫu i (được tính qua softmax).

Đối với bài toán phân loại nhị phân (binary classification), công thức được đơn giản hóa thành:

$$L = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

Trong đó:

- y_i: Nhãn thực tế (0 hoặc 1) cho mẫu i.
- $\hat{y}_i$: Xác suất dự đoán $P(y_i = 1)$ từ mô hình.

Hàm Cross-Entropy Loss giúp mô hình tối ưu hóa để dự đoán xác suất gần với nhãn thực tế nhất.

5.7 Khác

5.7.1 Giới thiệu về GridSearch

GridSearch là một phương pháp tìm kiếm lưới để xác định siêu tham số tối ưu cho mô hình học máy. Phương pháp này sẽ thử tất cả các kết hợp của các siêu tham số đã được chỉ định từ trước, tính toán và đánh giá hiệu suất của mô hình với từng bộ siêu tham số, sau đó chọn bộ siêu tham số cho kết quả tối ưu nhất. Mặc dù GridSearch có thể rất tốn thời gian nếu số lượng siêu tham số và các giá trị của chúng lớn, nhưng nó là một phương pháp đơn giản và hiệu quả.

5.7.2 Giới thiệu về Genetic Algorithm (Giải thuật di truyền)

Genetic Algorithm (GA) là một phương pháp tối ưu hoá dựa trên việc mô phỏng nguyên lý chọn lọc tự nhiên và tiến hoá trong sinh học. Nó thường được sử dụng để tìm kiếm tham số tối ưu cho các vấn đề phức tạp, bao gồm tìm kiếm siêu tham số trong mô hình mạng neuron.

Cách hoạt động và cài đặt GA trong việc tìm kiếm siêu tham số tối ưu

Khởi tạo quần thể (Population Initialization) Một quần thể (*population*) được khởi tạo ngẫu nhiên. Mỗi cá thể (*individual*) đại diện cho một cấu hình của mô hình mạng neuron.
Ở bài toán này, một cấu hình của mạng neuron bao gồm: số lượng neuron mỗi lớp, tốc độ học (*learning rate*), kích thước batch (*batch size*), tỉ lệ *drop rate* của các lớp *dropout*, hệ số phạt L2 (*weight decay*) và số vòng lặp huấn luyện (*epochs*).

Hàm đánh giá (Evaluate Function) Đánh giá chất lượng của từng cá thể dựa trên hiệu suất của nó. Trong bài toán này, độ chính xác (*accuracy*) trên tập kiểm định (*validation set*) được sử dụng làm tiêu chuẩn đánh giá.
Chất lượng của mỗi cá thể sẽ được lưu trữ trong một bộ nhớ tạm để tăng tốc độ đánh giá và khắc phục vấn đề hạn chế tài nguyên GPU.

Chọn lọc (Selection) Chọn các cá thể tốt nhất dựa trên giá trị hàm đánh giá (*evaluate function*) để tạo ra thế hệ mới. Hai phương pháp chọn lọc phổ biến:

- *Roulette Wheel Selection*: Xác suất chọn mỗi cá thể dựa trên giá trị hàm đánh giá của nó. Các cá thể có giá trị đánh giá cao sẽ có khả năng được chọn lớn hơn.
- *Tournament Selection*: Chọn ngẫu nhiên một nhóm cá thể và so sánh chúng. Cá thể tốt nhất trong nhóm này sẽ được chọn để tạo thế hệ tiếp theo.

Trong bài toán này, nhóm lựa chọn *Tournament Selection* với kích thước giải đấu lớn để đẩy nhanh quá trình hội tụ. Một nửa số cá thể ban đầu sẽ được giữ lại.

Lai ghép (Crossover) Kết hợp hai cá thể cha mẹ để tạo ra con mới. Thông qua quá trình này, thông tin sẽ được chia sẻ giữa các cá thể trong quần thể.

Đột biến (Mutation) Thay đổi ngẫu nhiên một số giá trị trong cá thể nhằm duy trì tính đa dạng của quần thể, đồng thời tránh rơi vào bẫy cực trị cục bộ.

Thay thế (Replacement) Tạo ra thế hệ mới bằng cách thay thế các cá thể cũ kém hiệu quả bằng các cá thể mới. Những cá thể mới này được tạo ra thông qua quá trình lai ghép ngẫu nhiên giữa các cá thể chiến thắng từ giai đoạn *Selection*.

5.7.3 Phương pháp khởi tạo trọng số: He và Xavier Initialization

He Initialization He Initialization là phương pháp khởi tạo trọng số được thiết kế đặc biệt cho các hàm kích hoạt dạng ReLU (hoặc biến thể của ReLU). Công thức để khởi tạo trọng số như sau:

$$W \sim \mathcal{N}(0, \frac{2}{n_{in}})$$

Trong đó:

- W là trọng số của mạng.
- n_{in} là số lượng đầu vào của một neuron.
- $\mathcal{N}(0, \sigma^2)$ là phân phối chuẩn với trung bình bằng 0 và phương sai σ^2.

Xavier Initialization Xavier Initialization, hay Glorot Initialization, được thiết kế để giữ cho phương sai của đầu ra không đổi qua từng lớp trong mạng neuron. Điều này giúp giảm hiện tượng gradient vanishing và exploding, phù hợp với các hàm kích hoạt có giá trị đối xứng, như sigmoid hoặc tanh. Công thức khởi tạo trọng số:

$$W \sim \mathcal{U}(-\sqrt{\frac{1}{n_{in}}}, \sqrt{\frac{1}{n_{in}}})$$

hoặc

$$W \sim \mathcal{N}(0, \frac{1}{n_{in}})$$

Trong đó:

- $\mathcal{U}(a, b)$ là phân phối đều trong khoảng $[a, b]$.
- Các tham số khác giống như trong He Initialization.

5.7.4 Giới thiệu về Cross-Validation

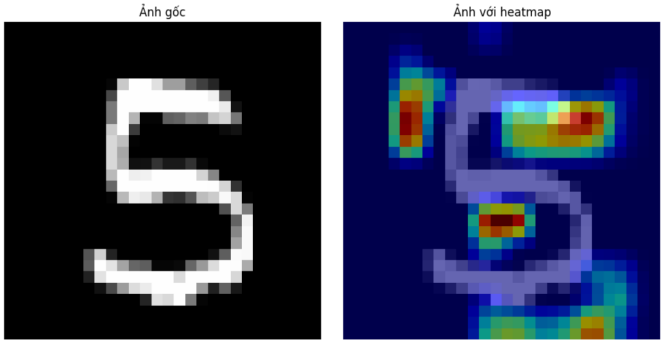
Cross-Validation là một kỹ thuật đánh giá mô hình, thường được sử dụng để ước lượng hiệu suất của mô hình trên dữ liệu chưa thấy (unseen data). Kỹ thuật phổ biến nhất là k-fold cross-validation, trong đó dữ liệu được chia thành k phần bằng nhau, và mô hình được huấn luyện k lần với mỗi phần dữ liệu làm tập kiểm tra một lần. Cross-validation giúp tránh hiện tượng overfitting và cung cấp một cách tiếp cận tổng quát hơn khi đánh giá mô hình.

5.7.5 Giới thiệu về Hold-out

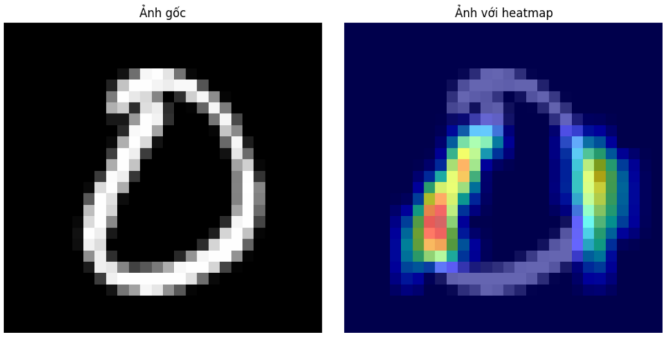
Hold-out là một phương pháp phân chia dữ liệu thành hai tập: tập huấn luyện và tập kiểm tra. Dữ liệu huấn luyện được sử dụng để huấn luyện mô hình, trong khi dữ liệu kiểm tra được sử dụng để đánh giá mô hình sau khi huấn luyện. Phương pháp này đơn giản và nhanh chóng, nhưng có thể không cung cấp một ước lượng chính xác về hiệu suất mô hình khi dữ liệu bị chia không đồng đều.

5.8 Giới thiệu về GradCam

Grad-Cam(Gradient-weighted Class Activation Mapping) là một mô hình giải thích và trực quan hóa quyết định của mạng nơ-ron. Mục đích của Grad-Cam là để hiển thị những vùng quan trọng nhất ảnh hưởng đến quyết định(hay đầu ra) của mạng tích chập CNN bằng cách tạo bản đồ heatmap để chỉ ra mức độ đóng góp của từng khu vực ảnh.



Hình 1: Mạng nhận diện số 5 thông qua các chi tiết được Grad-Cam tô đậm.



Hình 2: Mạng nhận diện số 0 thông qua các chi tiết được Grad-Cam tô đậm.

6 Kết quả và đánh giá

6.1 Kết Quả

6.1.1 Tổng quan về các số đo

Confusion matrix (ma trận nhầm lẫn) là một công cụ quan trọng trong bài toán phân loại nhị phân. Nó cung cấp thông tin về số lượng dự đoán đúng và sai ở mỗi lớp. Một confusion matrix trong bài toán phân loại nhị phân bao gồm bốn phần chính:

- **True Positive (TP):** Dự đoán đúng lớp dương tính.
- **True Negative (TN):** Dự đoán đúng lớp âm tính.
- **False Positive (FP):** Dự đoán sai lớp âm tính thành dương tính.
- **False Negative (FN):** Dự đoán sai lớp dương tính thành âm tính.

Các độ đo phân loại nhị phân khác nhau có xuất phát điểm từ confusion matrix này. Trong bài toán phân loại nhị phân, chúng ta cần đánh giá hiệu suất của mô hình phân loại. Dưới đây là các độ đo phổ biến được sử dụng:

Accuracy (Độ chính xác) : Độ chính xác là tỷ lệ giữa số lượng dự đoán đúng và tổng số mẫu. Nó được tính bằng công thức:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Tuy nhiên, độ chính xác không phản ánh đúng hiệu suất của mô hình khi dữ liệu bị thiên lệch. Ví dụ, trong trường hợp dữ liệu có sự mất cân bằng giữa các lớp, mô hình có thể dự đoán một lớp ít phổ biến hơn với độ chính xác cao mà không phải là một mô hình tốt.

Precision (Độ chính xác dương tính) : Precision đo lường khả năng của mô hình phân loại đưa ra dự đoán chính xác cho các điểm dương tính. Công thức tính precision như sau:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Precision cao đồng nghĩa với việc mô hình có ít dự đoán sai dương tính, tuy nhiên, nó không quan tâm đến việc mô hình có bỏ sót bao nhiêu điểm dương tính thực sự (*false negative*).

Recall (Độ phủ) : đo lường khả năng của mô hình phân loại tìm ra các điểm dương tính thực sự. Công thức tính recall như sau:

$$\text{Recall} = \frac{TP}{TP + FN}$$

Recall cao đồng nghĩa với việc mô hình tìm ra nhiều điểm dương tính thực sự, tuy nhiên, nó không quan tâm đến việc mô hình có dự đoán sai bao nhiêu điểm âm (*false positive*).

F1-score : là một độ đo kết hợp giữa *precision* và *recall*, giúp đánh giá hiệu suất tổng thể của mô hình phân loại. Công thức tính F1-score như sau:

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

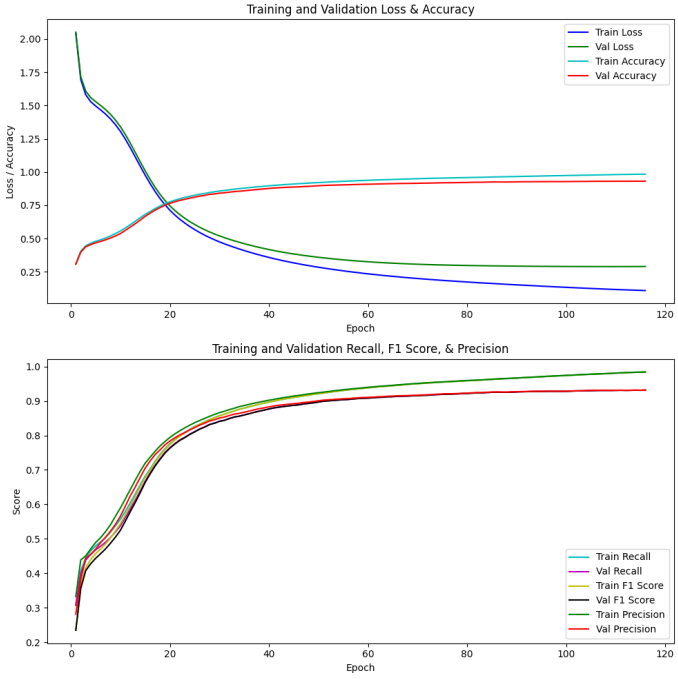
F1-score là một con số trung bình điều hòa (*harmonic mean*) của *precision* và *recall*. Nó giúp cân nhắc cả việc dự đoán chính xác các điểm dương tính và tìm ra các điểm dương tính thực sự.

6.1.2 Kết quả

Các mạng dưới đây đều có lớp cuối đi qua hàm softmax

Mạng Fully connected layers : Dưới đây là một số kiến trúc và kết quả thu được:

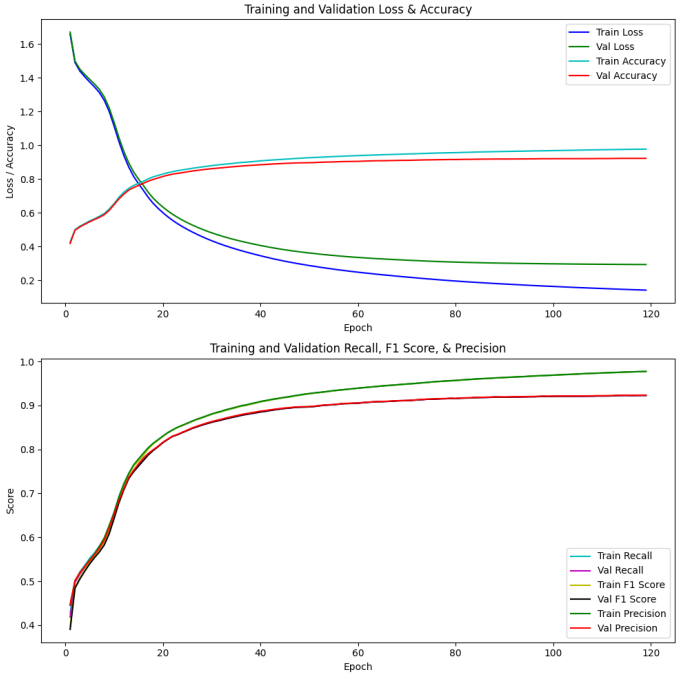
- **Hàm kích hoạt: sigmoid. Thuật toán tối ưu: gradient descent. Các kỹ thuật khác: L2 Regularization, EarlyStop.**



Train Accuracy: 98.90% **Loss train:** 0.1 **Test Accuracy:** 89.36% **Loss test:** 0.48496

Siêu tham số: 2 hidden layers với lần lượt 512 và 128, learning rate: 0.001, batch size: 128, epochs: 116, weight decay: 0.00001.

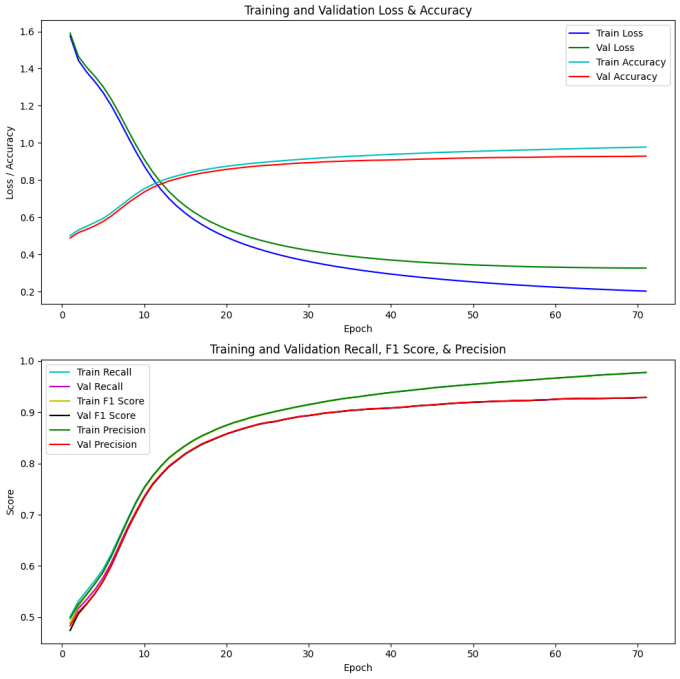
- **Hàm kích hoạt: sigmoid. Thuật toán tối ưu: gradient descent with momentum. Các kỹ thuật khác: L2 Regularization, EarlyStop.**



Train Accuracy: 98% **Loss train:** 0.1 **Test Accuracy:** 89.75% **Loss test:** 0.39374

Siêu tham số: 1 hidden layers với 512 noron, learning rate:0.00001 ,momentum:0.99,, batch size: 128, epochs: 116, weight decay: 0.00001.

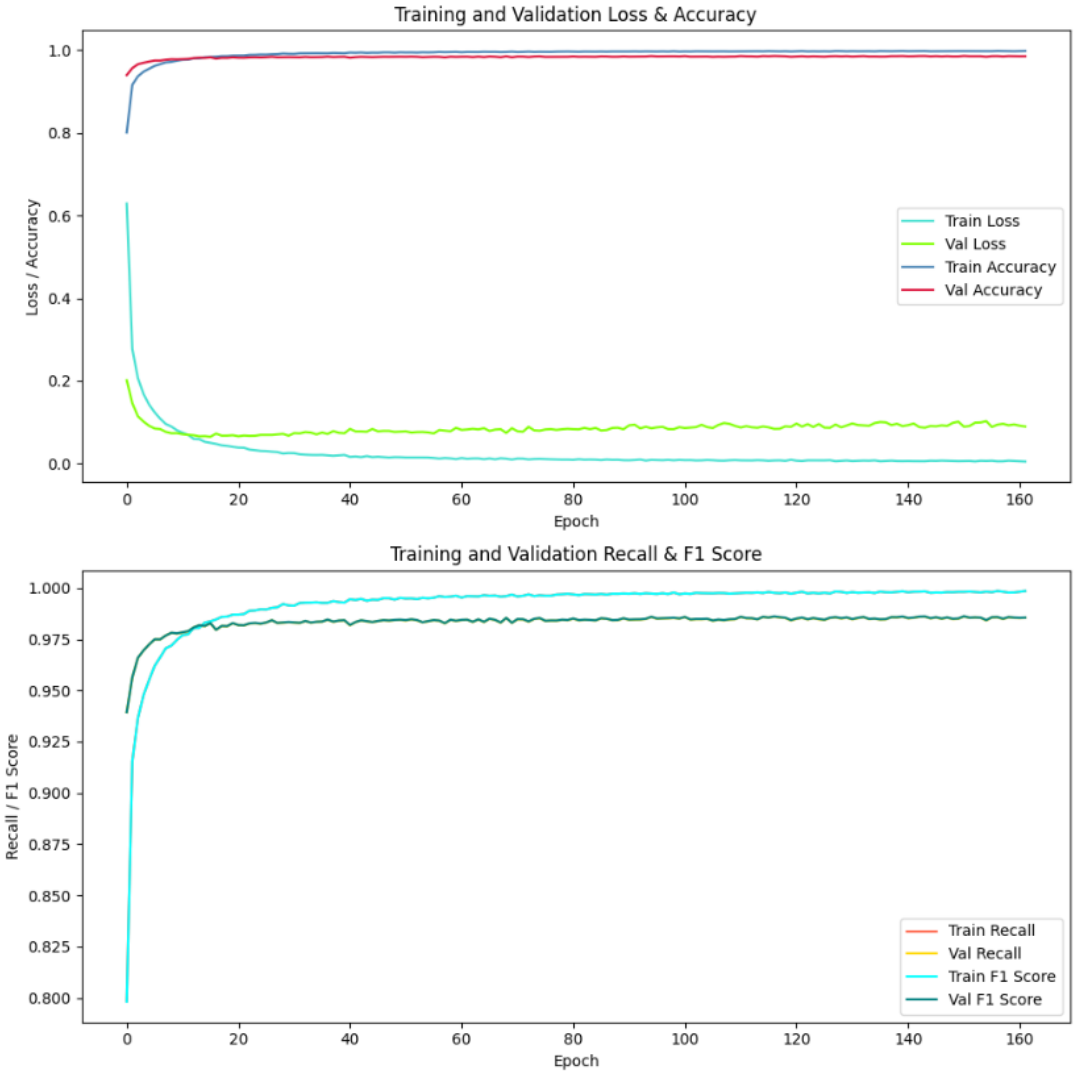
- **Hàm kích hoạt: sigmoid. Thuật toán tối ưu: adam. Các kỹ thuật khác: L2 Regularization, EarlyStop.**

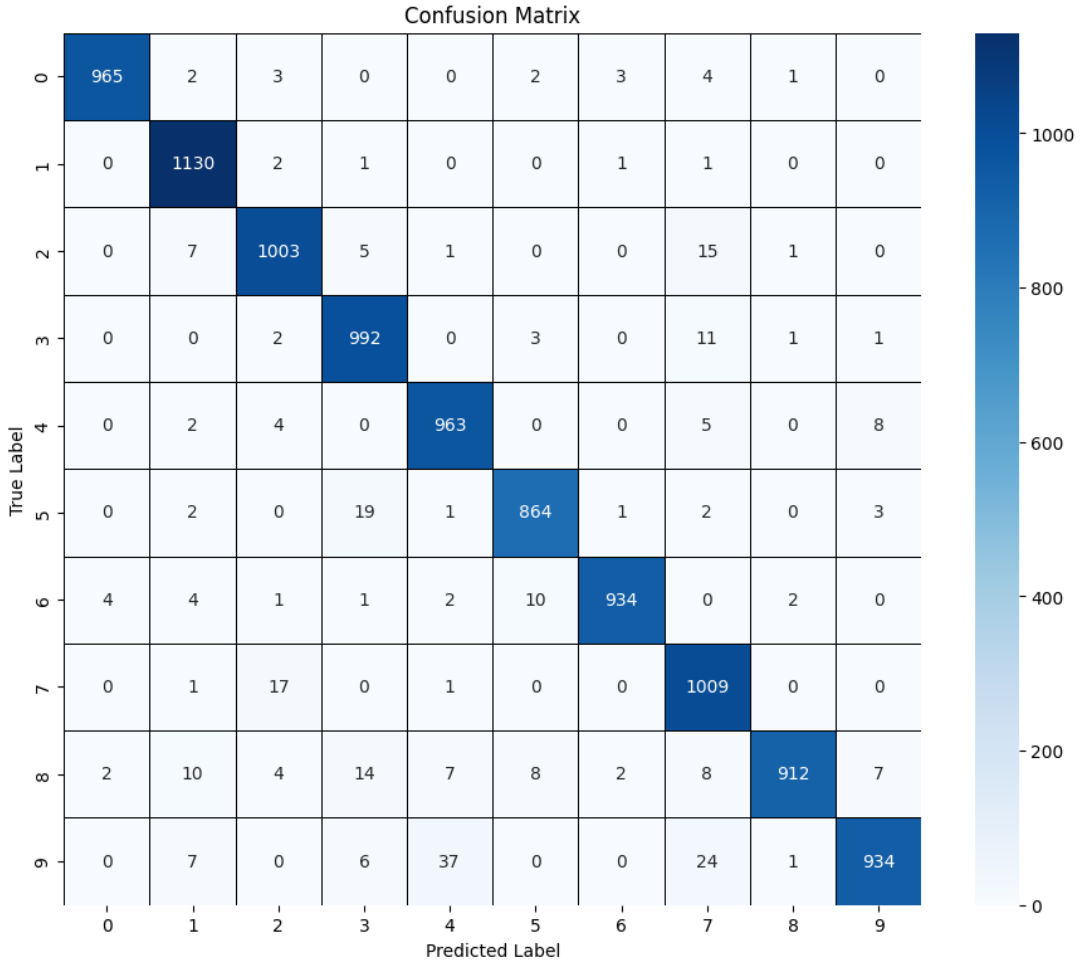


Train Accuracy: 98.1% **Loss train:** 0.1988 **Test Accuracy:** 82.95% **Loss test:** 0.7936

Siêu tham số: 1 hidden layers với 512 noron, learning rate:0.00008 , batch size: 32, epochs: 71, weight decay: 0.00001.

- **Hàm kích hoạt: ReLU. Thuật toán tối ưu: adam. Các kỹ thuật khác: L2 Regularization, EarlyStop, Dropout.**



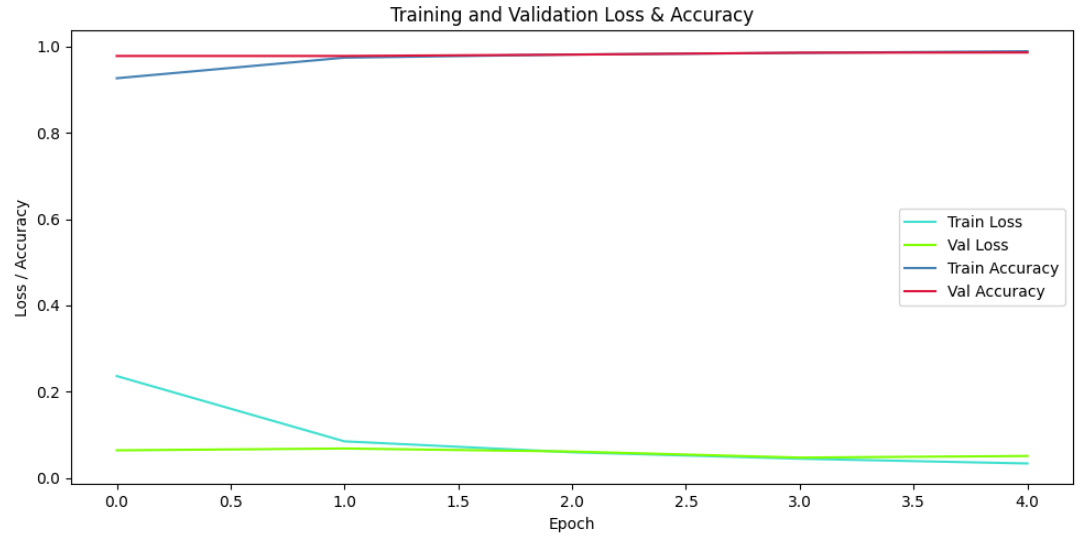


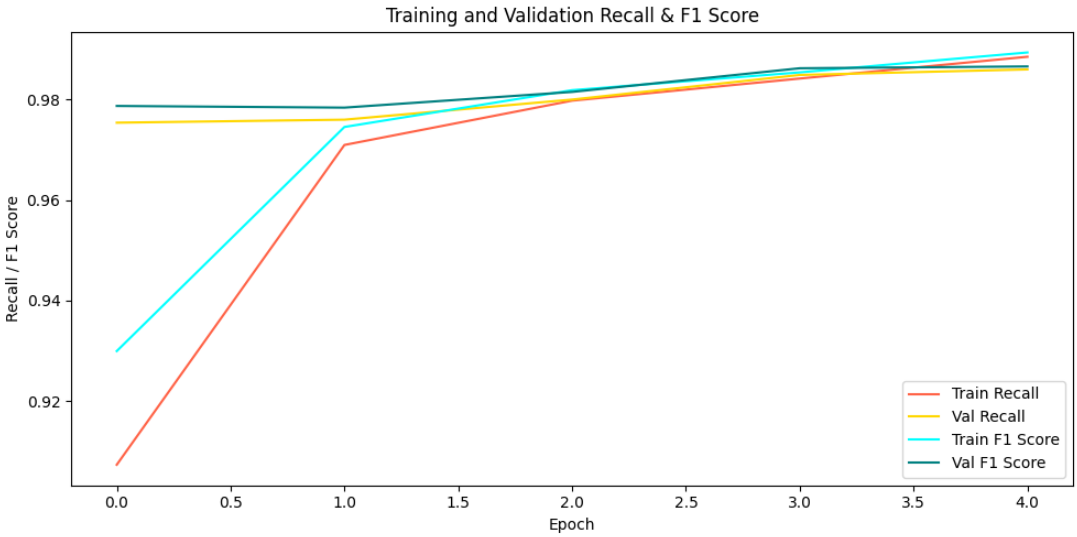
Train Accuracy: 99.85% **Loss train:** 0.0048 **Test Accuracy:** 98.54% **Loss test:** 0.0859

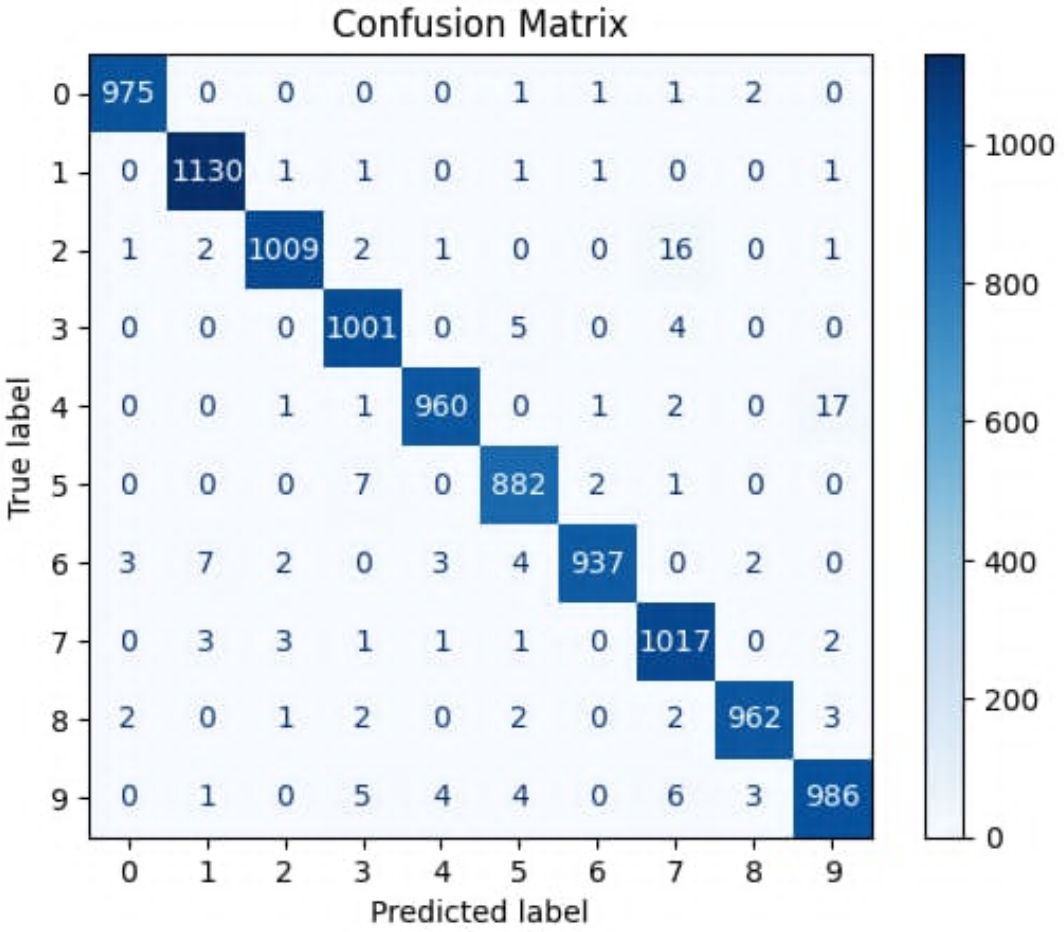
Siêu tham số: 3 hidden layers với [512,512,256] neuron, learning rate = 1e-4, batch size = 32, 250 epochs, weight decay: [1e-5,0,1e-5], drop rate = [0.5,0,0.3]

Mạng Convolutional Neural Network.

- **Hàm kích hoạt:** ReLU.
- **Thuật toán tối ưu:** Adam.
- **Các kỹ thuật khác:** Dropout (0.2), EarlyStopping (patience = 2, restore best weights).


Training and Validation Loss & Accuracy
Loss / Accuracy
Epoch
Train Loss
Val Loss
Train Accuracy
Val Accuracy


Training and Validation Recall & F1 Score
Recall / F1 Score
Epoch
Train Recall
Val Recall
Train F1 Score
Val F1 Score



Siêu tham số: Mô hình gồm 2 convolutional layers, learning rate mặc định của Adam = 0.01, batch size: 16, epochs tối đa: 5.
Train Accuracy: 99.66%, **Precision:** 99.66%, **Recall:** 99.66%, **F1 Score:** 99.66%
Test Accuracy: 98.78%, **Precision:** 98.79%, **Recall:** 98.78%, **F1 Score:** 98.78%

6.2 Nhận xét

7 Mã nguồn

7.1 Thư viện sử dụng

Dưới đây là các thư viện Python được sử dụng trong mô hình:

```
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.datasets import mnist
```

7.2 Cấu trúc mã nguồn

Chương trình được chia thành 4 phần, bao gồm: Phân tích bộ dữ liệu, Tiền xử lý dữ liệu, Tìm siêu tham số, Đánh giá mô hình

- https://colab.research.google.com/drive/1TMUHlJ-rTymWYOW9uDbZI_uBUzDnA2cX?usp=sharing Mã nguồn cho mô hình mạng kết nối toàn phần với hàm tối ưu Gradient Descent, hàm kích hoạt Sigmoid.
- https://colab.research.google.com/drive/1jHf52C6xqcd3d-kPQC9XQSrMrweeB_4G?usp=sharing Mã nguồn cho mô hình mạng kết nối toàn phần với hàm tối ưu Gradient Descent with Momentum, hàm kích hoạt Sigmoid.
- https://colab.research.google.com/drive/1u9gWU-O49FmbGTZoLrvYzp-KsYMHod6n?usp=sharing Mã nguồn cho mô hình mạng kết nối toàn phần với hàm tối ưu Adam, hàm kích hoạt Sigmoid.
- https://colab.research.google.com/drive/1qr6zVWgtxaGdDYwsNgOv9ukyqAKUVOks?usp=sharing Mã nguồn cho mô hình mạng kết nối toàn phần với hàm tối ưu Adam, hàm kích hoạt ReLU.
- https://colab.research.google.com/drive/1K1t60i3T_4fl21Jz4VI5P6M3s8IB_7CZ?usp=sharing&fbclid=IwZXh0bgNhZWOCMTAAAR1KJV_aIgkANgDMW1RJc4GZSONJ54MZwU82ZDpVBLWp4sEX2IwFrXGOq3A_aem_x6P690Ao1Xi-NR2arsh57Q#scrollTo=wIZ7hr_4Ms0k Mã nguồn cho mô hình mạng Convolutional Neural Network.
- https://www.kaggle.com/code/khenguyn/yolov3expansionproject Mã nguồn cho mô hình mạng YOLOv3 nhận diện nhiều chữ số trong một bức ảnh.
- https://colab.research.google.com/drive/1qyCXdf4FyDEfK-1bCKc44QyGcONPYJgO?usp=sharing&fbclid=IwY2xjawHIk7BleHRuA2FlbQIxMAABHR6dLXvOqdwz9pOTH_ShSLbFqkZGikPbKU9yOqqQipdCGgV3Saem_moRxcmPkZ3J1AZxN9BDewA#scrollTo=Jddi1jhRTp3H Mã nguồn cho xây dựng mô hình Convolutional Neural Network thủ công.
- https://l.facebook.com/l.php?u=https%3A%2F%2Fgithub.com%2Fgittbin%2FML_image_recognize3Ffbclid%3DIwZXh0bgNhZWOCMTAAAR31KFp7bcIAdd2JmCkDVTVN3Io6RNoWQPBXYJdYljgwnVSLBZ5X7L5Aaem_vuS4fjCFagsJGW5IJ9lWhA&h=AT2yJ8PrzfM8rS5teuZT7-5bCT33l_SrazPkT5wFYcYPi4eHV4Ob_ohsG2Ylfwi7bDmG74tcvcUOZUbsRzk7Cj-oe6cZJsD6Zjl39HYZVqTUs8iIUFppZ8maDdqdM-FPiwChcw Mã nguồn trò chơi áp dụng FC và CNN.
- https://l.facebook.com/l.php?u=https%3A%2F%2Fdrive.google.com%2Ffile%2Fd%2F1Iv7BCn_MnOCVj88RIXwjuYyT18e78oSM%2Fview%3Ffbclid%3DIwZXh0bgNhZWOCMTAAAR2sQJgQ7X7_SkFMV_26OdeoMfWWqAoRHCrmU5Fdq08CU5fp7CL6dXYGAkQ_aem_xa9Wv2Gc4vtju-kL95iXtw&h=AT2yJ8PrzfM8rSSrazPkT5wFYcYPi4eHV4Ob_ohsG2Ylfwi7bDmG74tcvcUOZUbsRzk7Cj-oe6cZJsD6Zjl39HYZVqTUs8iIUFp Mô hình CNN
- https://drive.google.com/file/d/1SfifxbQObKQKsoOfqV8iC77qtO2wwF_T/view?usp=sharing Mô hình FC

8 Mở rộng bài toán

8.1 Phân loại ảnh MNIST và hạn chế

Trong các chương trước, bài toán phân loại các chữ số viết tay trên bộ dữ liệu MNIST đã được thực hiện thành công bằng các mô hình mạng nơ-ron tích chập (CNN). Kết quả đạt được là khả năng phân loại chính xác các chữ số từ 0 đến 9 với độ chính xác cao. Tuy nhiên, bài toán phân loại ảnh MNIST có một số hạn chế như sau:

- Ảnh đầu vào chỉ chứa một chữ số duy nhất ở vị trí trung tâm, kích thước cố định là 28×28.

- Không xử lý được trường hợp có nhiều chữ số trong cùng một ảnh hoặc chữ số nằm ở các vị trí khác nhau.
- Không cung cấp thông tin về vị trí và kích thước của các chữ số trong ảnh.

Để khắc phục các hạn chế này và mở rộng khả năng ứng dụng thực tế, đề tài tiếp tục phát triển thành bài toán phát hiện đối tượng số trong ảnh đen trắng .

8.2 Mở rộng bài toán phát hiện đối tượng

Thay vì chỉ phân loại một chữ số đơn lẻ, bài toán mới có mục tiêu:

- Phát hiện nhiều chữ số trong cùng một ảnh đen trắng.
- Xác định vị trí (tọa độ bounding box) và loại chữ số (từ 0 đến 9) của từng đối tượng trong ảnh.
- Xử lý ảnh có kích thước lớn hơn, như 64×64, với chữ số nằm ở các vị trí ngẫu nhiên.

Bài toán này thuộc nhóm các bài toán phát hiện đối tượng trong thị giác máy tính. giải pháp được chọn để giải quyết bài toán này là Yolov3

8.3 Tổng Quan Dữ Liệu

Trong phần này, chúng tôi sẽ giới thiệu về dataset được sử dụng trong nghiên cứu này, bao gồm các chi tiết về nguồn gốc dữ liệu, quy trình chuẩn bị, và cách thức dữ liệu được sử dụng trong các thí nghiệm. Dataset này sẽ được áp dụng để huấn luyện mô hình YOLOv3 cho việc nhận diện các chữ số viết tay từ bộ dữ liệu MNIST.

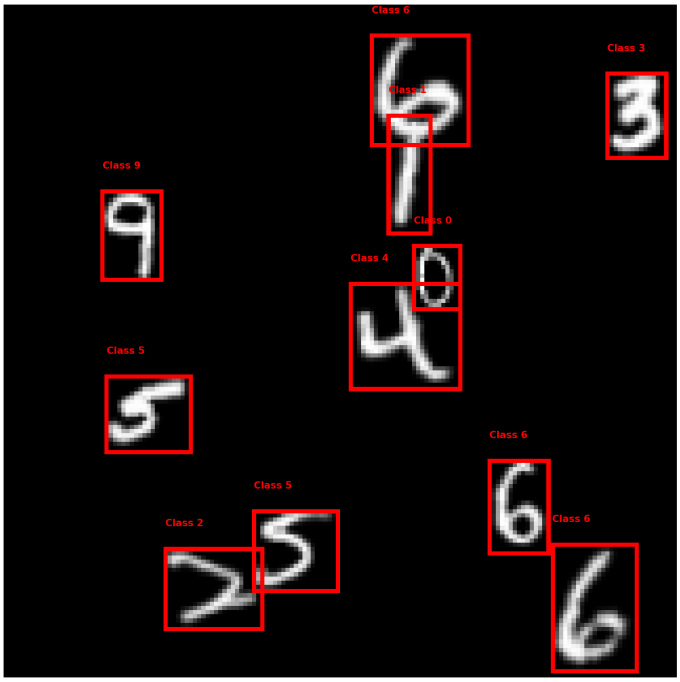
8.3.1 Cách Dataset Được Tạo Ra

Dataset được tạo ra bằng cách sử dụng dữ liệu từ bộ MNIST, bộ dữ liệu chứa các hình ảnh của các chữ số viết tay từ 0 đến 9. Tuy nhiên, không giống như MNIST gốc chỉ bao gồm một chữ số trong mỗi ảnh, dataset của chúng tôi chứa nhiều chữ số trong cùng một ảnh, với mỗi chữ số được bao quanh bởi một bounding box. Các bước để tạo dataset bao gồm:

- **Chọn ngẫu nhiên các chữ số từ bộ dữ liệu MNIST:** Mỗi chữ số trong bộ MNIST được chọn ngẫu nhiên và resize để phù hợp với kích thước yêu cầu.
- **Xoay và chèn chữ số vào ảnh mới:** Chữ số được xoay ngẫu nhiên trong phạm vi từ $-10°$ đến $10°$ và sau đó được chèn vào ảnh đen mới với kích thước cố định 160×160.
- **Tạo bounding boxes cho mỗi chữ số:** Các bounding boxes được tính toán xung quanh từng chữ số và lưu lại thông tin về chúng trong các tệp nhãn theo định dạng YOLO.
- **Tránh chồng lấn giữa các bounding boxes:** Để đảm bảo các bounding boxes không chồng lấn quá nhiều, thuật toán *Intersection over Union* (IoU) được sử dụng để kiểm tra mức độ chồng lấn và loại bỏ các trường hợp không hợp lệ.

Dataset được chia thành các thư mục con:

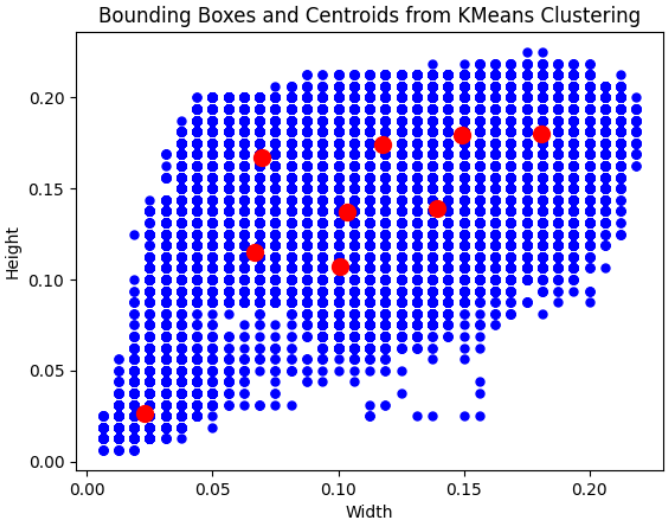
- `images/`: Chứa các hình ảnh đã tạo ra.
- `labels/`: Chứa các tệp nhãn, mỗi tệp tương ứng với một hình ảnh trong thư mục `images`.



Hình 3: Một hình ảnh mẫu

8.3.2 Phân tích dữ liệu

Phân tích phân bố kích thước của bounding boxes Phân tích sự phân bố kích thước của các bounding box trong tập dữ liệu, xác định các kích thước bounding box phổ biến.

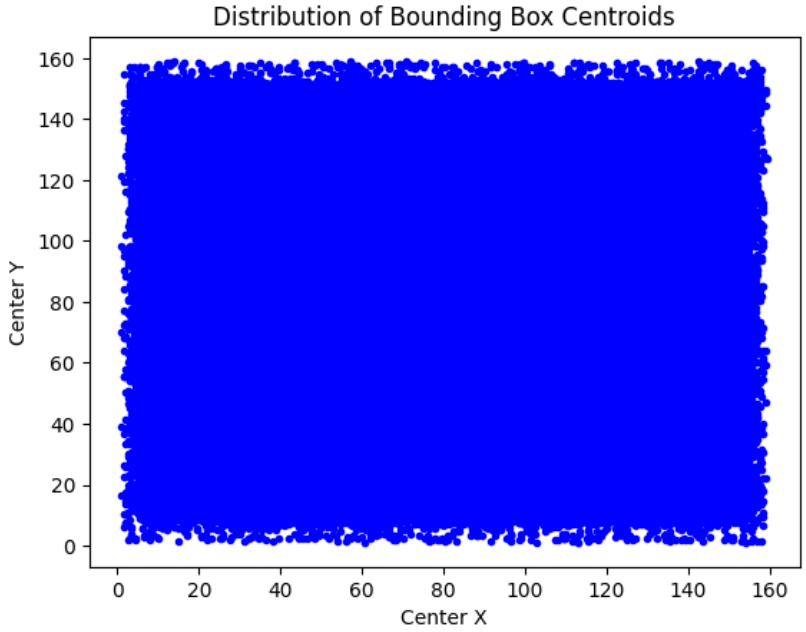


Hình 4: Phân bố kích thước của các bounding box trong dataset.

Dựa trên biểu đồ phân bố kích thước của các bounding box, ta có thể nhận thấy các đối tượng có kích thước nhỏ chiếm ít phần trong dataset so với các đối tượng lớn hơn. Điều này có thể ảnh hưởng đến việc huấn luyện, đặc biệt là khi mạng không được huấn luyện đầy đủ với các đối tượng có kích thước nhỏ.

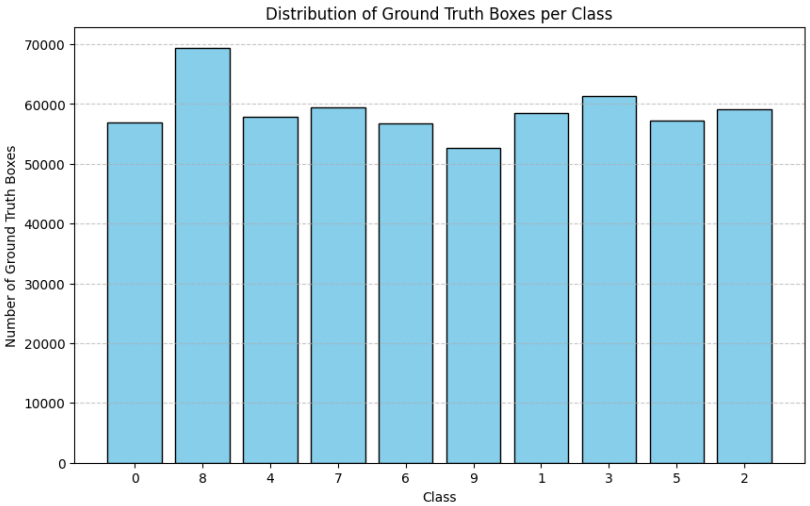
Phân tích sự phân bố tâm của bounding boxes Xem xét phân bố không gian của các tâm bounding box trong các ảnh để đánh giá khả năng phân bố các đối tượng trong ảnh.

Biểu đồ cho thấy các tâm bounding box phân bố khá đồng đều trong không gian ảnh.



Hình 5: Sự phân bố của các tâm bounding box trong không gian ảnh.

Tỷ lệ giữa các lớp đối tượng Phân tích tỷ lệ giữa các lớp đối tượng trong dataset, xác định sự cân bằng hoặc mất cân bằng giữa các lớp và ảnh hưởng của chúng đến hiệu suất huấn luyện.

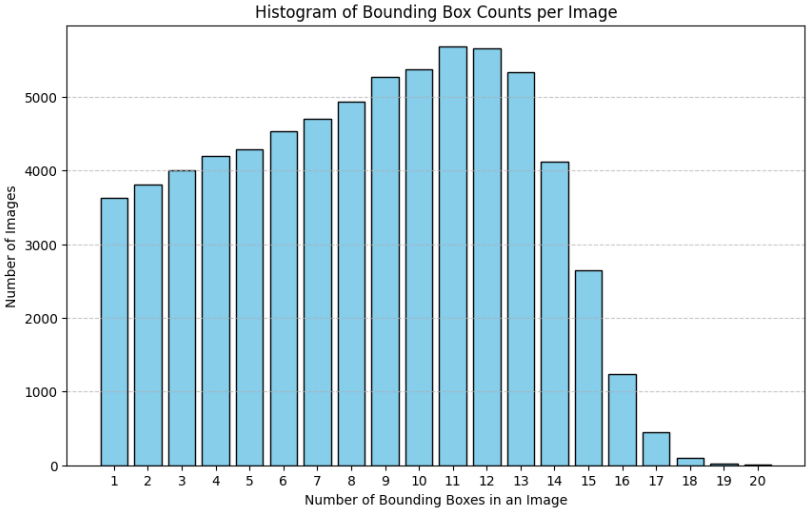


Hình 6: Tỷ lệ giữa các lớp đối tượng trong dataset.

Như có thể thấy từ biểu đồ trên, sự phân bố các lớp trong dataset là khá cân bằng.

Phân tích số lượng bounding box trên mỗi ảnh Phân tích số lượng bounding boxes trên mỗi ảnh trong dataset, từ đó đánh giá độ phức tạp của mỗi ảnh và tác động của nó đối với quá trình huấn luyện.

Biểu đồ trên thể hiện số lượng bounding boxes xuất hiện trong mỗi ảnh. Ta có thể nhận thấy sự phân bố của số lượng bounding boxes không đồng đều: một số ảnh có rất ít bounding boxes, trong khi một số ảnh khác lại có nhiều bounding boxes. Điều này có thể ảnh hưởng đến khả năng học của mô hình, đặc biệt là trong các ảnh có quá nhiều bounding boxes hoặc quá ít đối tượng. Việc điều chỉnh sự phân bố này có thể cải thiện hiệu suất huấn luyện.



Hình 7: Biểu đồ số lượng bounding box trên mỗi ảnh trong dataset.

8.3.3 Chuẩn bị Dữ liệu và Tiền xử lý

Trong quá trình huấn luyện mô hình YOLO, việc chuẩn bị và tiền xử lý dữ liệu là rất quan trọng để đảm bảo rằng dữ liệu đầu vào được chuẩn hóa và định dạng phù hợp. Dữ liệu trong bài toán YOLO bao gồm các ảnh và nhãn, trong đó nhãn chứa thông tin về các bounding box (vùng chứa đối tượng). Mỗi bounding box được mô tả bằng các thông số: lớp đối tượng, tọa độ trung tâm (x, y), và kích thước (chiều rộng và chiều cao).

8.3.4 Thu thập và Tiền xử lý Dữ liệu

Dữ liệu được lấy từ các thư mục chứa ảnh và nhãn. Mỗi ảnh có định dạng '.png' và mỗi tệp nhãn tương ứng có định dạng '.txt'. Trong tệp nhãn, mỗi dòng mô tả một bounding box, bao gồm: lớp đối tượng, tọa độ trung tâm của bounding box, chiều rộng và chiều cao của bounding box, với các giá trị được phân tách bởi dấu cách.

Dữ liệu sẽ được chuẩn hóa theo các bước sau:

- Đọc ảnh: Các ảnh được đọc từ thư mục ảnh dưới dạng grayscale. Sau đó, các giá trị pixel trong ảnh được chuẩn hóa về phạm vi từ 0 đến 1 bằng cách chia giá trị pixel cho 255.0.
- Đọc nhãn: Các nhãn được đọc từ các tệp '.txt'. Mỗi tệp nhãn chứa thông tin về các đối tượng trong ảnh, với mỗi dòng chứa các giá trị: lớp đối tượng, tọa độ trung tâm của bounding box (x, y), và kích thước (chiều rộng và chiều cao).
- Khởi tạo targets cho các scale: Trong YOLO, chúng ta sử dụng nhiều kích thước lưới (scale) khác nhau để dự đoán các bounding box. Các scale này được xác định bởi tham số $S = [5, 10, 20]$, tương ứng với các kích thước lưới khác nhau trong ảnh. Mỗi scale sẽ có một số lượng anchor boxes xác định(được xác định nhờ vào phân cụm k-mean), và các anchor boxes này sẽ được gán với các ground truth có iou lớn nhất với anhchors box.
- Tính toán IoU (Intersection over Union): Để xác định anchor box nào phù hợp với một ground truth, ta tính toán giá trị IoU giữa ground truth và các anchor boxes. Anchor box nào có IoU lớn nhất với ground truth và lớn hơn một ngưỡng xác định (ví dụ: 0.7),đồng thời tại scale đó cell chưa có ground truth nào được gắn với anchor box đó, anchor box đó sẽ được sử dụng gắn với ground truth này .
- Chuyển đổi và chuẩn hóa thông tin nhãn: Các tọa độ (x, y) và kích thước (w, h) của bounding box trong nhãn được chuyển đổi về hệ quy chiếu của lưới, tức là tỷ lệ của tọa độ và kích thước

so với kích thước của lưới. Sau đó, thông tin này được gán vào các target tại các vị trí tương ứng trong lưới.

- Đảm bảo tính đầy đủ của các anchor boxes: Mỗi bounding box sẽ chỉ được gán với một anchor box tại mỗi scale, và nếu một anchor box đã được gán cho một bounding box, nó sẽ không được gán cho bất kỳ bounding box nào khác tại cùng scale.

Kết quả Chuẩn bị Dữ liệu

Sau khi tiến hành các bước tiền xử lý, mỗi ảnh sẽ có một tập các target (một cho mỗi scale). Các target này chứa thông tin về các bounding box trong ảnh, bao gồm: sự hiện diện của đối tượng (1 hoặc 0), các giá trị bình thường hóa của tọa độ và kích thước của bounding box, và chỉ số lớp đối tượng. Các target này sẽ được sử dụng trong quá trình huấn luyện để cập nhật các tham số của mô hình YOLO, nhằm tối ưu hóa khả năng phát hiện đối tượng trong ảnh.

8.4 Giới thiệu về YOLOv3

YOLOv3 (You Only Look Once version 3) là một trong những mô hình phát hiện đối tượng phổ biến nhất hiện nay nhờ vào:

- Tốc độ xử lý nhanh: YOLOv3 chỉ cần một lần duyệt qua ảnh đầu vào để phát hiện tất cả các đối tượng.
- Độ chính xác cao: YOLOv3 sử dụng backbone Darknet-53 với residual connections để cải thiện khả năng trích xuất đặc trưng.
- Phát hiện đa tỉ lệ: Mô hình có khả năng phát hiện các đối tượng có kích thước nhỏ, trung bình và lớn thông qua kiến trúc Feature Pyramid Network (FPN).

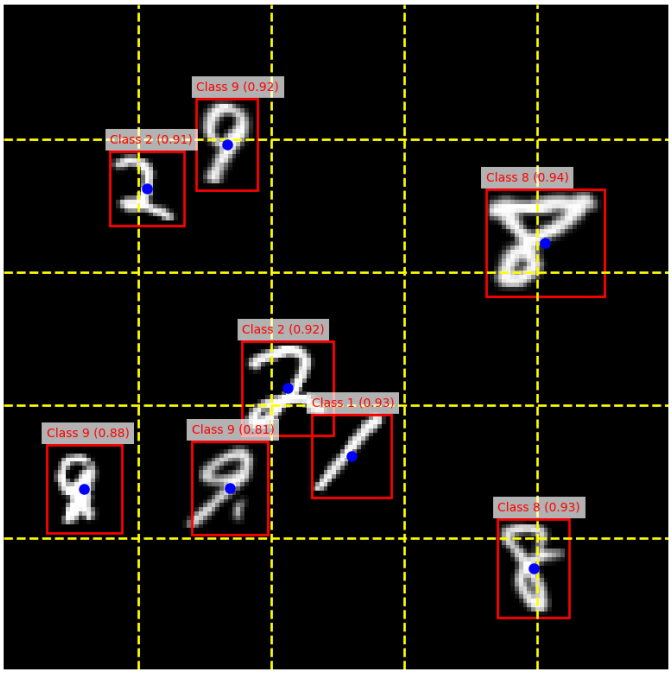
8.5 Khái quát phương thức YOLO hoạt động

Ý tưởng chính của YOLO (You Only Look Once) là chia ảnh đầu vào thành $S \times S$ ô vuông (cell). Mỗi cell chịu trách nhiệm phát hiện các *bounding box* có tâm nằm trong cell đó. Với ý tưởng như vậy, đầu vào và đầu ra của mô hình được mô tả như sau:

- **Đầu vào:** Là một hoặc nhiều ảnh có kích thước `batch_size` × `num_channels` × `width` × `height`. Ví dụ:
  - Trong bài toán hiện tại khi huấn luyện batch size được chọn là 32, đầu vào sẽ là một tensor kích thước $32 \times 1 \times 160 \times 160$.
  - Khi dự đoán từng ảnh một, đầu vào sẽ có kích thước $1 \times 1 \times 160 \times 160$.
- **Đầu ra:** Tensor đầu ra trước tiên phải có $S \times S$ chiều, biểu diễn các ô cell trong ảnh. Mỗi ô cell cần có các thông tin sau:
  1. **Toạ độ tâm của bounding box:** Mỗi ô cell dự đoán tọa độ tâm của *bounding box* nằm trong cell đó. Do đó, đầu ra khi này cần chứa thông tin về toạ độ tâm:

     $$S \times S \times 2$$

     với 2 giá trị biểu diễn tọa độ tâm (x, y).



Hình 8: Mẫu cách yolo hoạt động

2. **Kích thước bounding box:** Để xác định được *bounding box*, cần thêm độ rộng và chiều cao (width, height). Khi đó, đầu ra mở rộng thành:

$$S \times S \times (2+2) = S \times S \times 4$$

3. **Xác suất có object:** Không phải lúc nào các ô cell cũng chứa đối tượng. Do đó, cần thêm một giá trị biểu diễn khả năng có một object xuất hiện trong bounding box đó. Đầu ra tiếp tục mở rộng thành:

$$S \times S \times (4+1) = S \times S \times 5$$

4. **Phân loại đối tượng:** Để phân loại đối tượng trong bounding box, cần thêm một vector xác suất có độ dài `num_classes`, biểu thị xác suất object bên trong `bounding_box` là 1 instance của 1 nhãn lớp cụ thể. Khi đó, đầu ra của một ô cell sẽ là:

$$S \times S \times (5 + \texttt{num\_classes})$$

5. **Dự đoán nhiều bounding box:** Vì một cell có thể chứa tâm của nhiều đối tượng, mỗi cell cần dự đoán nhiều hơn một bounding box. Giả sử mỗi cell dự đoán `num_bounding_box` bounding box, đầu ra mở rộng thành:

$$\texttt{num\_bounding\_box} \times S \times S \times (5 + \texttt{num\_classes})$$

- **Tổng quát đầu ra cho toàn bộ batch:** Cuối cùng, khi huấn luyện với `batch_size` ảnh, đầu ra sẽ có dạng:

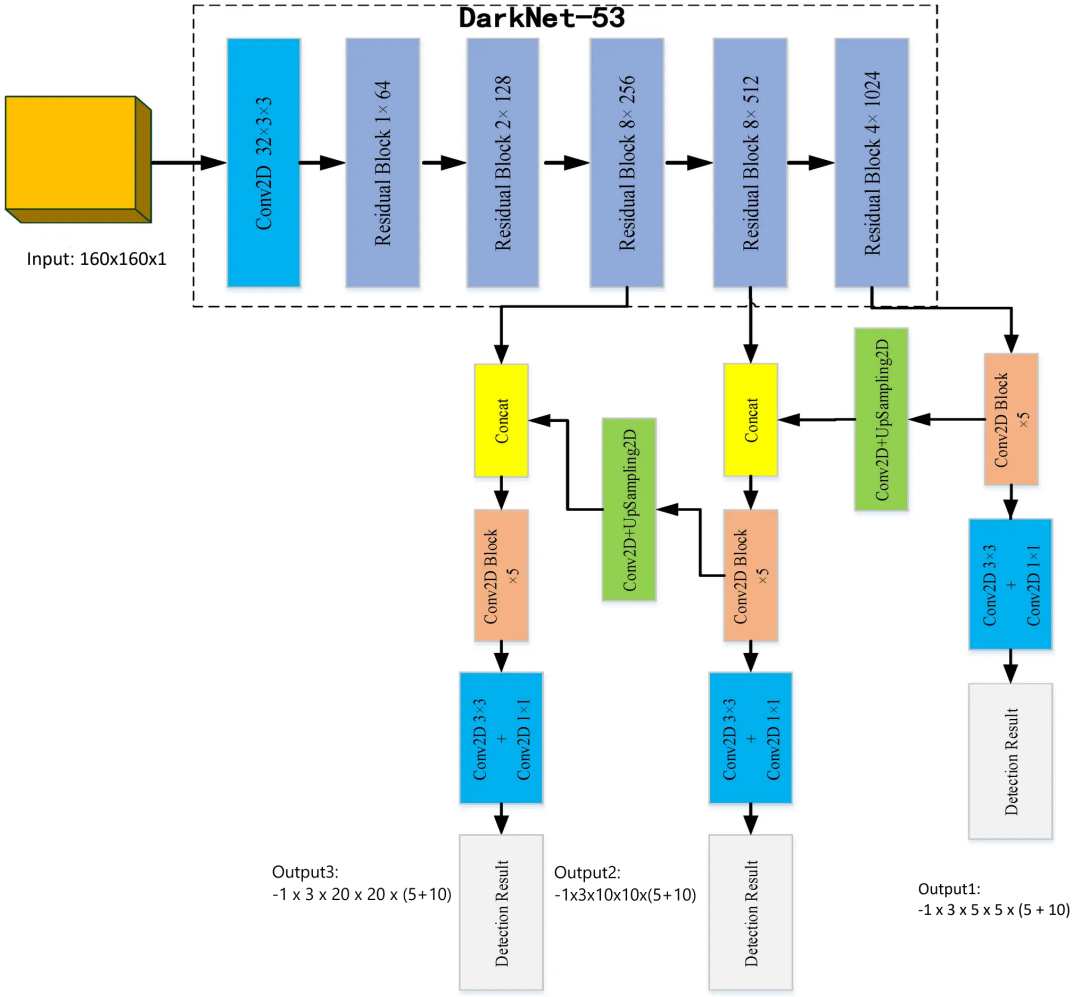
$$\texttt{batch\_size} \times \texttt{num\_bounding\_box} \times S \times S \times (5 + \texttt{num\_classes})$$

Với cấu trúc đầu vào và đầu ra như trên, YOLO thực hiện phát hiện và phân loại đối tượng một cách trực tiếp và hiệu quả thông qua một mạng nơ-ron duy nhất.

8.6 Giới thiệu khái quát về kiến trúc mạng YOLOv3

8.6.1 Kiến trúc tổng quát của YOLOv3

YOLOv3 (You Only Look Once version 3) sử dụng kiến trúc mạng *Darknet-53*, một mạng nơ-ron tích chập (CNN) bao gồm tổng cộng 53 lớp tích chập. Mạng Darknet-53 được xây dựng từ sự kết hợp của các khối tích chập tiêu chuẩn và khối *Residual*, giúp huấn luyện mạng nơ-ron sâu trở nên hiệu quả hơn.



Hình 9: Kiến trúc mạng yolov3

Cấu trúc các tầng trong Darknet-53 Darknet-53 được chia thành các khối tích chập chính với số lượng lớp và kích thước khác nhau. Cấu trúc này bao gồm:

- **Tầng đầu vào**: Sử dụng một lớp tích chập ban đầu với kích thước kernel 3×3 và stride là 1.
- **Các khối Residual**: Darknet-53 bao gồm 5 cụm khối Residual, mỗi cụm có số lượng khối Residual khác nhau (1, 2, 8, 8, 4). Trong đó:
  - Mỗi khối Residual bao gồm hai lớp tích chập liên tiếp với kích thước kernel 3×3 và 1×1.
  - Các khối Residual được kết nối với nhau thông qua *skip connections* (kết nối bỏ qua).
- **Giảm kích thước đặc trưng (downsampling)**: Thay vì sử dụng các tầng pooling, Darknet-53 sử dụng các lớp tích chập với stride = 2 để giảm kích thước đặc trưng. Việc sử dụng tích chập stride = 2 mang lại nhiều lợi ích như:

- **Tham số học được**: Convolution stride = 2 có các tham số trong bộ lọc tích chập, cho phép mạng học cách giảm chiều dữ liệu một cách tối ưu, phù hợp với nhiệm vụ cụ thể.
- **Bảo toàn thông tin tốt hơn**: Convolution stride = 2 giảm kích thước nhưng vẫn kết hợp thông tin từ các pixel lân cận một cách *tinh tế* thông qua các bộ lọc học được, giúp mạng giữ lại nhiều thông tin hơn so với MaxPooling.
- **Khả năng truyền gradient tốt hơn**: Các tham số học được trong convolution stride = 2 giúp truyền gradient tốt hơn so với MaxPooling, vì MaxPooling là một phép toán cố định và đôi khi làm giảm độ nhạy với gradient.
- **Hạn chế mất mát thông tin**: MaxPooling chỉ chọn giá trị lớn nhất trong vùng và bỏ qua các giá trị còn lại, điều này có thể gây mất mát thông tin đáng kể.

Sử dụng các tầng Convolution thay vì Fully Connected Network Thay vì làm phẳng đầu ra và sử dụng mạng Fully Connected Network (FCN) như trong các mô hình truyền thống, YOLOv3 tiếp tục sử dụng các tầng convolution để duy trì cấu trúc không gian của *feature map*. Việc này mang lại một số lợi ích như:

- **Giữ lại cấu trúc không gian**: Các tầng convolution giữ nguyên mối liên kết không gian giữa các đặc trưng trên *feature map*, trong khi việc làm phẳng trong FCN có thể làm mất thông tin không gian quan trọng.
- **Giảm số lượng tham số**: Nhờ chia sẻ tham số trong các bộ lọc convolution, số lượng tham số của mạng giảm đáng kể so với việc sử dụng các lớp fully connected, từ đó tăng tốc độ tính toán.
- **Tính linh hoạt với kích thước ảnh đầu vào**: Do không có các lớp fully connected, YOLOv3 có thể xử lý các ảnh đầu vào với kích thước khác nhau mà không cần thay đổi kiến trúc mạng.

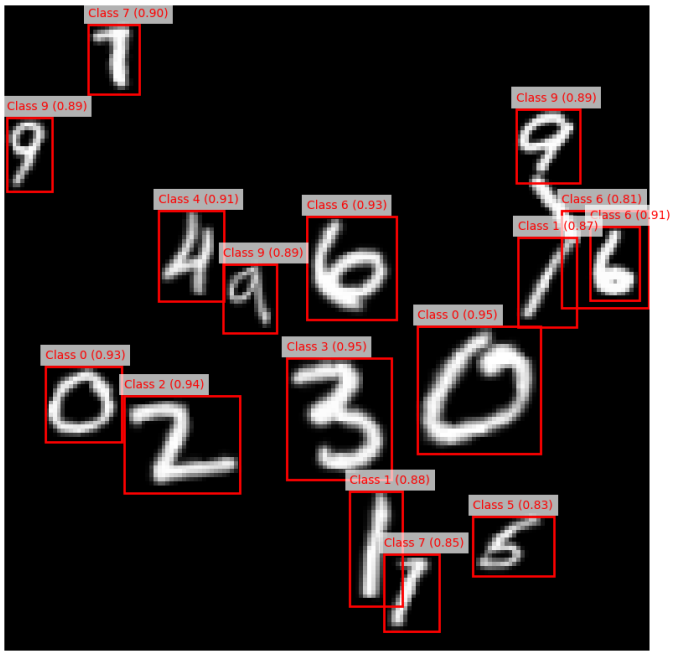
Hạn chế của việc không sử dụng fully connected layers Tuy nhiên, việc không sử dụng fully connected layers cũng có một số hạn chế:

- **Giảm khả năng học các kết nối toàn cục**: Fully connected layers có khả năng học các mối liên hệ toàn cục giữa các đặc trưng trên *feature map*, trong khi convolution chủ yếu tập trung vào các kết nối cục bộ.
- **Khó khăn trong việc tóm tắt toàn bộ thông tin**: Do chỉ sử dụng convolution, YOLOv3 dự đoán *bounding boxes* và *class scores* tại từng vị trí trên *feature map*. Điều này có thể tạo ra các kết quả dư thừa (dự đoán tại nhiều vị trí gần nhau) và yêu cầu các kỹ thuật như Non-Maximum Suppression (NMS) để loại bỏ các *bounding boxes* không cần thiết.

8.4.1 Kết hợp thông tin đa tầng bằng Feature Pyramid Network (FPN)

YOLOv3 muốn đưa ra dự đoán cho các đối tượng ở các tỉ lệ khác nhau, tuy nhiên gặp phải các thách thức:

- **Lớp nông (shallow layers)**: Chứa các đặc trưng cục bộ chi tiết như cạnh, góc, và kết cấu, nhưng thiếu ngữ cảnh tổng thể. Dự đoán trực tiếp từ tầng này có thể dẫn đến kết quả không chính xác.
- **Lớp sâu (deep layers)**: Tích lũy được nhiều thông tin ngữ nghĩa (semantic), giúp nhận diện các đối tượng lớn và bối cảnh hình ảnh. Tuy nhiên, kích thước không gian của feature map bị giảm mạnh, khiến khó phát hiện các đối tượng nhỏ.



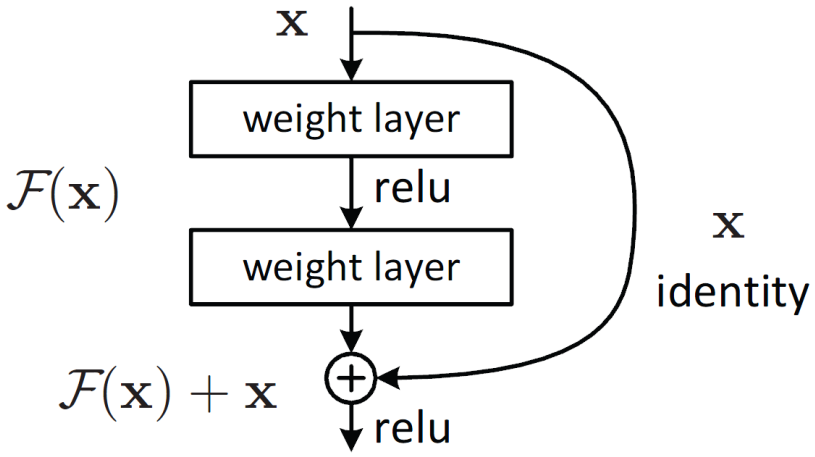
Hình 10: Chú ý thấy ràng số 7 bì nhận diện nhầm thành số 1 do tính cục bộ của lớp convolution

Giải pháp của YOLOv3 YOLOv3 áp dụng kiến trúc Feature Pyramid Network (FPN) để kết hợp thông tin từ các tầng cao và tầng thấp:

- **Trích xuất thông tin từ ba mức tỉ lệ**:
  - **Tầng cao**: Dự đoán đối tượng lớn (feature map nhỏ nhất).
  - **Tầng trung**: Dự đoán đối tượng vừa (feature map trung bình).
  - **Tầng thấp**: Dự đoán đối tượng nhỏ (feature map lớn nhất).
- **Kết hợp thông tin qua upsampling**:
  - YOLOv3 thực hiện *upsampling* (phóng to feature map) từ tầng cao để kết hợp thông tin ngữ nghĩa với các đặc trưng chi tiết của tầng thấp hơn.
  - Các feature maps từ tầng cao được nối (concatenate) với feature maps từ tầng thấp để tạo ra thông tin kết hợp đa tỉ lệ.
- **Phát hiện đa mức**: Sau khi kết hợp thông tin, YOLOv3 tối ưu hóa ba mức tỉ lệ để dự đoán bounding boxes và class scores:
  - **Tầng thấp**: Phát hiện đối tượng nhỏ.
  - **Tầng trung**: Phát hiện đối tượng vừa.
  - **Tầng cao**: Phát hiện đối tượng lớn.

8.6.2 Phân tích khối Residual trong YOLOv3

Khối *Residual* đóng vai trò quan trọng trong kiến trúc Darknet-53. Mỗi khối Residual bao gồm một kết nối bỏ qua (skip connection), giúp thông tin và gradient có thể truyền qua mạng dễ dàng hơn.



Hình 11: Cấu trúc khối Residual trong Darknet-53

Cấu trúc và hoạt động của khối Residual Mỗi khối Residual bao gồm hai nhánh chính:

- **Nhánh chính**: Thực hiện biến đổi phi tuyến thông qua các lớp tích chập liên tiếp.
- **Nhánh bỏ qua (skip connection)**: Truyền trực tiếp đầu vào đến đầu ra mà không thay đổi.

Công thức biểu diễn khối Residual như sau:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x}, \tag{1}$$

trong đó:

- $\mathbf{x}$ là đầu vào của khối Residual.
- $\mathcal{F}(\mathbf{x}, \{W_i\})$ là kết quả của các phép tích chập và hàm phi tuyến (ReLU).
- $\mathbf{y}$ là đầu ra của khối Residual.
- Dấu cộng (+) biểu diễn kết nối bỏ qua, giúp đưa thông tin đầu vào trực tiếp tới đầu ra.

Vai trò của khối Residual Khối Residual giúp YOLOv3 khắc phục nhiều vấn đề thường gặp trong mạng nơ-ron sâu:

- **Giảm thiểu vấn đề vanishing gradient**: Kết nối bỏ qua đảm bảo gradient có thể lan truyền ngược qua các lớp mà không bị suy giảm.
- **Tăng cường khả năng huấn luyện**: Nhờ skip connection, mạng có thể học các biến đổi phi tuyến phức tạp một cách dễ dàng hơn.
- **Tích lũy đặc trưng**: Kết nối bỏ qua cho phép mạng tổng hợp và tích lũy các đặc trưng từ các lớp trước, giúp tăng khả năng biểu diễn của mạng.
- **Giảm overfitting**: Khối Residual giúp ổn định quá trình huấn luyện và tránh tình trạng quá khớp khi mạng trở nên rất sâu.

8.4.2 Sự kết hợp giữa YOLOv3 và Darknet-53

YOLOv3 và Darknet-53 là sự kết hợp hoàn hảo trong việc nhận diện đối tượng nhờ vào việc sử dụng mạng CNN sâu với các đặc trưng mạnh mẽ từ Darknet-53. Mạng Darknet-53 cung cấp khả năng trích xuất đặc trưng mạnh mẽ với các residual blocks và cấu trúc mạng sâu, trong khi YOLOv3 tận dụng các đặc trưng này để dự đoán các đối tượng một cách chính xác và nhanh chóng trong các bức ảnh.

9 Hàm Loss YOLOv3

Hàm loss trong YOLOv3 bao gồm một số thành phần khác nhau, mỗi thành phần có vai trò riêng biệt trong việc tối ưu hóa mô hình. Hàm loss tổng quát của YOLOv3 bao gồm các số hạng sau:

$$\mathcal{L} = \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{\text{obj}} \left[(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \right]$$

$$+\lambda_{\text{obj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{\text{obj}} \cdot \text{BCEWithLogits}(C_i, \hat{C}_i)$$

$$+\lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{\text{noobj}} \cdot \text{BCEWithLogits}(C_i, \hat{C}_i)$$

$$+\lambda_{\text{classes}} \sum_{i=0}^{S^2} 1_{\text{obj}} \cdot \text{CrossEntropy}(p_i, \hat{p}_i),$$

10 Số Hạng Tọa Độ trong Hàm Loss của YOLO

Hàm loss phần tọa độ trong YOLO bao gồm việc dự đoán chính xác tọa độ trung tâm (x, y) và kích thước (w, h) của bounding box. Các đại lượng được chuẩn hóa như sau:

- (x, y): Tọa độ trung tâm của bounding box được chuẩn hóa tương đối với ô *cell* mà bounding box thuộc về.
- (w, h): Chiều rộng và chiều cao của bounding box được chuẩn hóa theo kích thước của toàn bộ ảnh.

10.1 Loss Tọa Độ

Số hạng loss tọa độ có dạng như sau:

$$\begin{aligned}\text{Loss}_{\text{coord}} = \lambda_{\text{coord}} &\sum_{i=1}^{S^2} \sum_{j=1}^{B} 1_{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\ &+ \lambda_{\text{coord}} \sum_{i=1}^{S^2} \sum_{j=1}^{B} 1_{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]\end{aligned}$$

Trong đó:

- λ_{coord}: Trọng số điều chỉnh tầm quan trọng của số hạng tọa độ trong tổng hàm loss.
- 1_{obj}: Là một chỉ báo bằng 1 nếu bounding box j tại ô i chứa object, ngược lại là 0.
- x_i, y_i: Tọa độ trung tâm thực tế của bounding box trong ô i.

- $\hat{x}_i, \hat{y}_i$: Tọa độ trung tâm được dự đoán.
- w_i, h_i: Chiều rộng và chiều cao thực tế của bounding box.
- $\hat{w}_i, \hat{h}_i$: Chiều rộng và chiều cao được dự đoán.

10.2 Chuẩn Hóa Tọa Độ và Kích Thước

- Tọa độ trung tâm (x, y) được chuẩn hóa trong khoảng $[0, 1]$ với góc trên bên trái của ô cell là $(0, 0)$ và góc dưới bên phải là $(1, 1)$.
- Chiều rộng và chiều cao (w, h) sử dụng căn bậc hai để cân bằng giữa các bounding box lớn và nhỏ:
$$\sqrt{w}, \sqrt{h}.$$
Việc này giúp giảm độ chênh lệch giữa các bounding box có kích thước lớn và nhỏ, làm cho quá trình tối ưu hóa ổn định hơn.

11 Giải thích Số Hạng Object Loss (obj_loss)

Số hạng object loss trong YOLO được sử dụng để tối ưu hóa giá trị confidence của bounding box có chứa object.

11.1 Công thức tính C_i

Giá trị confidence thực tế C_i của bounding box tại ô i được định nghĩa là:

$$C_i = \text{Pr} \cdot \text{IOU},$$

trong đó:

- Pr: Xác suất có object xuất hiện trong bounding box.
- IOU: Intersection over Union giữa bounding box dự đoán và ground truth.

11.2 Giải thích Các Đại Lượng

- Xác suất Pr:
$$\text{Pr} = P(\text{object present}),$$
là xác suất có object trong bounding box tại ô i. Nếu bounding box chứa object, Pr $= 1$; ngược lại, Pr $= 0$.
- Intersection over Union (IOU):
$$\text{IOU} = \frac{\text{Area of Overlap}}{\text{Area of Union}},$$
trong đó:
  - Area of Overlap: Diện tích giao giữa bounding box dự đoán và ground truth.
  - Area of Union: Diện tích hợp của bounding box dự đoán và ground truth.

  IOU nằm trong khoảng $[0, 1]$, với:
  - IOU $= 1$: Hai bounding box hoàn toàn trùng khớp.
  - IOU $= 0$: Hai bounding box không có vùng giao nhau.

11.3 Số hạng Object Loss

Hạng tử object loss được tính bằng công thức:

$$\lambda_{\text{obj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{\text{obj}} \cdot \text{BCEWithLogits}(C_i, \hat{C}_i),$$

trong đó:

- $\hat{C}_i$: Giá trị confidence dự đoán bởi mô hình.
- $C_i = \Pr \cdot \text{IOU}$: Giá trị confidence thực tế.
- $\text{BCEWithLogits}(C_i, \hat{C}_i)$: Hàm Binary Cross Entropy với Logits, giúp tối ưu hóa dự đoán $\hat{C}_i$ về gần C_i.
- 1_{obj}: Chỉ báo bằng 1 nếu bounding box chứa object; ngược lại bằng 0.

12 Phân Tích Hạng Tử No Object Loss (noobj_loss)

Hạng tử No Object Loss được sử dụng để tối ưu hóa confidence score $\hat{C}_i$ cho các bounding box không chứa object. Điều này giúp mô hình giảm thiểu các dự đoán sai ở các vùng background.

12.1 Công Thức No Object Loss

$$\mathcal{L}_{\text{noobj}} = \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{\text{noobj}} \cdot \text{BCEWithLogits}(C_i, \hat{C}_i),$$

trong đó:

- λ_{noobj}: Trọng số điều chỉnh mức độ ảnh hưởng của no object loss so với các thành phần khác trong hàm loss.
- 1_{noobj}: Là chỉ báo bằng 1 nếu bounding box j trong ô i không chứa object, ngược lại bằng 0.
- $C_i = 0$: Vì không có object trong bounding box, giá trị confidence thực tế được gán bằng 0.
- $\hat{C}_i$: Confidence score dự đoán bởi mô hình cho bounding box không chứa object.
- $\text{BCEWithLogits}(C_i, \hat{C}_i)$: Hàm Binary Cross Entropy with Logits, có công thức:

$$\text{BCEWithLogits}(C_i, \hat{C}_i) = -\left[C_i \cdot \log(\sigma(\hat{C}_i)) + (1 - C_i) \cdot \log(1 - \sigma(\hat{C}_i))\right],$$

  với $\sigma(\hat{C}_i) = \frac{1}{1+e^{-\hat{C}_i}}$ là hàm sigmoid.

12.2 Ý Nghĩa Của No Object Loss

- Mục tiêu: Đảm bảo confidence score dự đoán $\hat{C}_i$ cho các bounding box không chứa object tiệm cận về 0, giảm thiểu các false positives (nhận diện nhầm object ở vùng background).
- Trọng số λ_{noobj}: Vì có rất nhiều bounding box không chứa object, hạng tử này thường chiếm tỷ lệ lớn trong tổng hàm loss. Do đó, λ_{noobj} được sử dụng để giảm ảnh hưởng quá mức của nó, giúp cân bằng với các hạng tử khác như tọa độ và class loss.

13 Giải Thích Công Thức Class Loss

Hạng tử class loss được sử dụng để tối ưu hóa dự đoán nhãn class của object tại mỗi ô lưới (i) có chứa object.

13.1 Công Thức Class Loss

$$\mathcal{L}_{\text{class}} = \lambda_{\text{classes}} \sum_{i=0}^{S^2} 1_{\text{obj}} \cdot \text{CrossEntropy}(p_i, \hat{p}_i),$$

trong đó:

- λ_{classes}: Trọng số điều chỉnh mức độ ảnh hưởng của class loss so với các thành phần khác trong hàm loss.
- 1_{obj}: Chỉ báo nhị phân, bằng 1 nếu ô lưới i chứa object, bằng 0 nếu ngược lại.
- p_i: Vector ground truth xác suất của các class tại ô i, được biểu diễn bằng one-hot encoding.
- $\hat{p}_i$: Vector xác suất dự đoán của các class bởi mô hình, sau khi áp dụng hàm softmax.

13.2 Ý Nghĩa Class Loss

- Mục tiêu: Tối ưu hóa để xác suất dự đoán $\hat{p}_i$ khớp với nhãn thực p_i, giúp mô hình phân loại đúng class của object.
- Trọng số λ_{classes}: Do chỉ tính class loss cho các ô chứa object ($1_{\text{obj}} = 1$), trọng số này giúp cân bằng tầm quan trọng của class loss với các thành phần khác trong hàm loss.

13.3 Mối Quan Hệ Giữa Các Số Hạng và Lựa Chọn Siêu Tham Số λ

Hàm loss tổng quát của YOLO bao gồm các thành phần chính:

$$\mathcal{L} = \mathcal{L}_{\text{coord}} + \mathcal{L}_{\text{obj}} + \mathcal{L}_{\text{noobj}} + \mathcal{L}_{\text{class}},$$

trong đó:

$$\mathcal{L}_{\text{coord}} = \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \right],$$

$$\mathcal{L}_{\text{obj}} = \lambda_{\text{obj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{\text{obj}} \cdot \text{BCEWithLogits}(C_i, \hat{C}_i),$$

$$\mathcal{L}_{\text{noobj}} = \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{\text{noobj}} \cdot \text{BCEWithLogits}(C_i, \hat{C}_i),$$

$$\mathcal{L}_{\text{class}} = \lambda_{\text{classes}} \sum_{i=0}^{S^2} 1_{\text{obj}} \cdot \text{CrossEntropy}(p_i, \hat{p}_i).$$

Mối Quan Hệ Giữa Các Số Hạng

1. Hạng tử tọa độ ($\mathcal{L}_{\text{coord}}$): - Có vai trò quan trọng để tối ưu hóa vị trí và kích thước của bounding box. - Nếu λ_{coord} quá nhỏ, mô hình có thể không học tốt tọa độ chính xác, dẫn đến bounding box sai lệch. - Ngược lại, nếu λ_{coord} quá lớn, nó có thể làm giảm tầm quan trọng của các thành phần khác.

2. Hạng tử object ($\mathcal{L}_{\text{obj}}$): - Xác suất dự đoán object xuất hiện (C_i) cần được tối ưu để phản ánh chính xác mức độ tự tin của mô hình. - λ_{obj} cần được lựa chọn sao cho cân bằng với hạng tử $\mathcal{L}_{\text{noobj}}$, tránh mô hình ưu tiên các ô có object mà bỏ qua các ô background.

3. Hạng tử no object ($\mathcal{L}_{\text{noobj}}$): - Do số lượng các ô không chứa object lớn hơn nhiều so với các ô có object, hạng tử này chiếm tỷ lệ lớn trong tổng loss. - Vì vậy, λ_{noobj} thường được chọn nhỏ hơn λ_{obj}, để giảm thiểu ảnh hưởng tiêu cực của việc học trên các ô không chứa object.

4. Hạng tử class ($\mathcal{L}_{\text{class}}$): - Phụ thuộc vào chất lượng dự đoán class của mô hình. - Với bài toán có nhiều class, việc dự đoán chính xác class trở nên khó khăn hơn, nên λ_{classes} cần được điều chỉnh phù hợp để đảm bảo mô hình học tốt các nhãn.

13.3.1 Lựa Chọn Siêu Tham Số λ

13.3.2 Trọng Số λ_{coord}

Trọng số λ_{coord} có ảnh hưởng lớn đến hiệu quả học của mô hình, đặc biệt là trong giai đoạn huấn luyện đầu tiên. Khi bắt đầu huấn luyện, các bounding box được dự đoán bởi mô hình thường ngẫu nhiên và có độ chính xác thấp, do đó, IOU (Intersection over Union) giữa các bounding box dự đoán và ground truth rất nhỏ.

Nếu trọng số λ_{obj} được đặt quá cao trong giai đoạn này, mô hình sẽ nhanh chóng tập trung vào việc giảm thiểu sai số đối với xác suất có đối tượng (object), dẫn đến việc mô hình học rất nhanh về xác suất mà không chú trọng vào việc tối ưu hóa vị trí và kích thước của bounding box. Điều này có thể gây ra các vấn đề như:

- Huấn luyện không hiệu quả: Mô hình sẽ không cải thiện được chất lượng của các bounding box trong những bước huấn luyện đầu tiên.
- Gradient Exploding: Khi mô hình quá tập trung vào việc giảm xác suất của các ô có object mà không chú trọng đến tọa độ, gradient có thể trở nên quá lớn, dẫn đến hiện tượng gradient exploding. Điều này sẽ khiến cho quá trình huấn luyện không ổn định, có thể buộc phải dừng huấn luyện do các giá trị gradient quá lớn.

Do đó, trọng số λ_{coord} cần được đặt đủ cao (ví dụ: $\lambda_{\text{coord}} = 5$) để đảm bảo rằng mô hình sẽ tập trung vào việc cải thiện tọa độ của các bounding box ngay từ những bước huấn luyện đầu tiên. Việc đặt giá trị này cao giúp đảm bảo rằng trong giai đoạn đầu, mô hình sẽ không bị "mất cân bằng"khi chỉ tối ưu hóa cho xác suất có đối tượng mà bỏ qua các lỗi trong việc dự đoán bounding box.

2. Trọng số λ_{obj} và λ_{noobj}: - λ_{obj}: Giá trị vừa phải, thường là 1. - λ_{noobj}: Giá trị nhỏ hơn nhiều, ví dụ $\lambda_{\text{noobj}} = 0.5$ hoặc $\lambda_{\text{noobj}} = 0.1$, để giảm tác động của background.

3. Trọng số λ_{classes}: - Với số lượng class lớn, λ_{classes} cần được điều chỉnh tăng để đảm bảo mô hình học tốt các nhãn class.

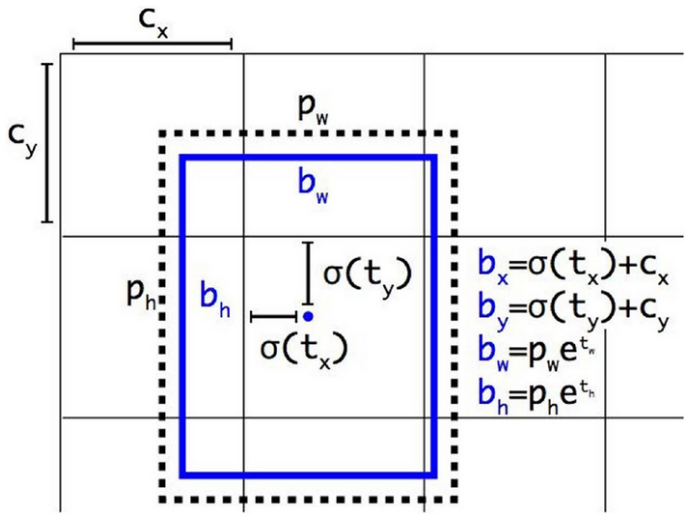
Chuẩn hoá đầu ra của YOLOv3

Trong YOLOv3, đầu ra của mạng là một tensor chứa tọa độ của các bounding box, điểm số objectness, và xác suất của các lớp. Để tăng tốc quá trình huấn luyện và cải thiện độ hội tụ, các giá trị đầu ra thô được chuẩn hoá trước khi sử dụng để dự đoán các bounding box và điểm số objectness. Quá trình chuẩn hoá được mô tả bằng các phương trình sau:

$$b_x = \sigma(t_x) + c_x$$

$$b_y = \sigma(t_y) + c_y$$
$$b_w = p_w \exp(t_w)$$
$$b_h = p_h \exp(t_h)$$
$$Pr(\text{object}) = \sigma(t_o)$$

Trong đó: - t_x, t_y, t_w, t_h, t_o là các đầu ra thô của mạng. - c_x, c_y, p_w, p_h là các hằng số điều chỉnh được



Hình 12: Chuẩn hoá đầu ra của mạng

xác định bởi vị trí và kích thước của các anchor boxes. - Hàm sigmoid σ được áp dụng cho các giá trị t_x, t_y, t_o để chuẩn hoá chúng về khoảng (0, 1). - Hàm mũ exp được áp dụng cho t_w và t_h để đảm bảo giá trị luôn dương, giúp mô hình dự đoán kích thước bounding box một cách hiệu quả.

Giải thích lý do sử dụng sigmoid và exponential:

- **Hàm sigmoid:** Hàm sigmoid được sử dụng cho các đầu ra tọa độ t_x, t_y và điểm số objectness t_o vì nó ánh xạ giá trị về khoảng (0, 1). Điều này phù hợp để dự đoán xác suất hoặc tỷ lệ phần trăm của các giá trị trong một ô lưới. Tuy nhiên, khi tâm của bounding box nằm gần rìa của ô lưới, việc sử dụng sigmoid có thể làm giảm độ chính xác trong dự đoán, do hàm này bị giới hạn ở phạm vi (0, 1).

- **Hàm exponential (exp):** Hàm exp được áp dụng cho t_w và t_h nhằm đảm bảo giá trị chiều rộng và chiều cao của bounding box luôn dương. Việc này giúp mô hình có thể dự đoán kích thước bounding box với độ chính xác cao.

Hỗ trợ của Anchor Boxes: Anchor boxes đóng vai trò quan trọng trong việc huấn luyện YOLOv3. Chúng cung cấp các khung tham chiếu với kích thước cố định, giúp mạng học được kích thước và hình dạng của các đối tượng một cách nhanh chóng mà không cần học lại từ đầu. Các anchor boxes giảm thiểu sự không khớp giữa đầu ra của mạng và các giá trị thực, qua đó cải thiện khả năng dự đoán vị trí và kích thước bounding box.

Vấn đề với việc sử dụng sigmoid khi tâm bounding box ở rìa ô lưới: Hàm sigmoid giúp giới hạn giá trị dự đoán tọa độ trong khoảng [0, 1], nhưng khi tâm bounding box nằm gần rìa của một ô lưới, điều này có thể làm giảm độ chính xác. Các thay đổi nhỏ ở giá trị t_x hoặc t_y trong trường hợp này có thể dẫn đến sự sai lệch lớn trong tọa độ tuyệt đối của bounding box. Để giảm thiểu vấn đề này, anchor boxes hỗ trợ bằng cách cung cấp các điểm tham chiếu, qua đó giúp cải thiện độ chính xác trong dự đoán bounding box cho các đối tượng gần rìa của ô lưới.

Kết Luận

Sự cân bằng giữa các trọng số $\lambda_{\text{coord}}, \lambda_{\text{obj}}, \lambda_{\text{noobj}}, \lambda_{\text{classes}}$ rất quan trọng trong việc cải thiện hiệu năng của mô hình YOLO. Việc lựa chọn các giá trị này cần dựa trên dữ liệu và bài toán cụ thể, thường thông qua thử nghiệm và điều chỉnh siêu tham số.

8.8 Đánh giá mô hình bằng mAP

Để đánh giá hiệu suất của mô hình phát hiện đối tượng, một trong những chỉ số phổ biến nhất là mAP (mean Average Precision). Chỉ số này tính toán mức độ chính xác của mô hình bằng cách đo lường diện tích dưới đường cong Precision-Recall (PR curve) cho từng lớp đối tượng. Mỗi đối tượng được đánh giá thông qua giá trị Intersection over Union (IoU) giữa bounding box dự đoán và bounding box thực tế.

8.8.1 Tại sao sử dụng map thay cho việc dùng recall hay precision làm độ đo chất lượng mô hình

Khi đánh giá hiệu suất của mô hình phát hiện đối tượng, việc chỉ sử dụng Precision hoặc Recall một cách độc lập có thể không phản ánh chính xác khả năng của mô hình trong các tình huống thực tế.

- Precision cao nhưng Recall thấp: Khi Precision cao mà Recall thấp, mô hình có thể chỉ ra một số ít dự đoán rất chính xác (TP), nhưng lại bỏ sót phần lớn các đối tượng thực tế (chưa được phát hiện). Ví dụ, mô hình có thể chỉ đưa ra một bounding box duy nhất, và nó khớp chính xác với một đối tượng trong ảnh. Tuy nhiên, do không phát hiện được các đối tượng còn lại, mô hình này vẫn sẽ có Recall thấp. Điều này có thể xảy ra khi mô hình chỉ phát hiện được một số ít đối tượng trong khi bỏ qua nhiều đối tượng khác, dẫn đến kết quả là không đủ thông tin để đưa ra đánh giá chính xác.

- Recall cao nhưng Precision thấp: Mặt khác, khi Recall cao nhưng Precision thấp, mô hình có thể đưa ra rất nhiều dự đoán, nhưng chỉ một phần trong số đó thực sự là chính xác. Điều này có thể dẫn đến hiện tượng mô hình đưa ra nhiều dự đoán sai (FP), nhưng vẫn có một số dự đoán đúng (TP). Chẳng hạn, mô hình có thể phát hiện hầu hết các đối tượng trong ảnh, nhưng lại đưa ra rất nhiều bounding box không chính xác. Kết quả là Recall cao nhưng Precision thấp, và mô hình vẫn có thể không thực sự tốt vì tỷ lệ các dự đoán sai là quá lớn.

- Cân bằng Precision và Recall: Vì vậy, việc chỉ đánh giá mô hình bằng một trong hai chỉ số trên không thể đưa ra cái nhìn đầy đủ về hiệu suất của mô hình. Precision và Recall thực sự phản ánh hai khía cạnh khác nhau của mô hình: Precision đánh giá khả năng dự đoán đúng của mô hình, trong khi Recall đánh giá khả năng phát hiện đầy đủ các đối tượng thực tế. Một mô hình lý tưởng cần phải đạt được một sự cân bằng giữa hai yếu tố này, tức là vừa có khả năng dự đoán chính xác, vừa có khả năng phát hiện tất cả các đối tượng.

13.4 Phương pháp đánh giá mới: F1-Score và mAP

Để đánh giá mô hình một cách toàn diện, một phương pháp hợp lý là sử dụng F1-Score, là trung bình hài hòa giữa Precision và Recall. F1-Score được tính bằng công thức:

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

F1-Score là một chỉ số mạnh mẽ vì nó cân bằng giữa hai yếu tố, giảm thiểu tác động của việc chỉ tập trung vào một trong hai chỉ số mà bỏ qua yếu tố còn lại.

Mặc dù F1-Score hữu ích, trong bài toán phát hiện đối tượng, **mAP** (Mean Average Precision) vẫn là chỉ số chính được sử dụng vì nó không chỉ đánh giá Precision và Recall mà còn tính toán diện tích dưới đường cong Precision-Recall cho từng lớp đối tượng, cung cấp cái nhìn tổng quan về khả năng phát hiện của mô hình. mAP tính toán sự thay đổi của Precision trong suốt phạm vi Recall, giúp đánh giá mô hình ở các mức độ khác nhau của sự phát hiện đối tượng.

13.5 Cách tính mAP

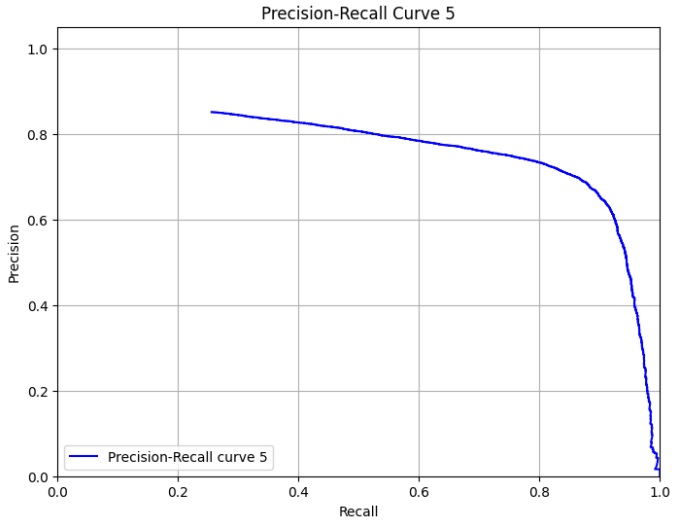
Cách tính mAP được thực hiện theo các bước sau:

- Xếp các bounding box theo confidence score giảm dần: Đối với mỗi nhãn lớp, ta sẽ xếp tất cả các bounding box dự đoán theo thứ tự giảm dần của confidence score (độ tin cậy). Việc này giúp ưu tiên các dự đoán có độ tin cậy cao hơn.
- Tính toán IoU với các Ground Truth: Với mỗi bounding box dự đoán, tính toán giá trị IoU với tất cả các bounding box ground truth có cùng lớp trong cùng một ảnh. Để xác định xem một dự đoán có đúng hay không (True Positive - TP) hay sai (False Positive - FP), ta sẽ so sánh IoU của dự đoán đó với các ground truth có cùng lớp. Nếu IoU vượt qua ngưỡng $IoU_{\text{threshold}}$, dự đoán được coi là TP. Nếu không, nó được coi là FP.
- Cập nhật TP và FP: Dự đoán là TP nếu nó không trùng với dự đoán đã được ghép đôi với ground truth trong ảnh đó. Nếu IoU không đạt ngưỡng, hoặc nếu đã có một dự đoán khác với IoU cao hơn, nó sẽ là FP.
- Tính Precision và Recall tại mỗi bước: Sau khi phân loại các dự đoán là TP hoặc FP, ta tính toán Precision và Recall tại mỗi bước:

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{\text{Tổng số đối tượng thực tế}}$$

- Vẽ đường cong Precision-Recall: Với từng nhãn lớp, ta vẽ đường cong Precision-Recall bằng cách sử dụng giá trị Precision và Recall đã tính được. Đoạn đường cong này thể hiện mối quan hệ giữa Precision và Recall khi thay đổi số lượng TP và FP.



Hình 13: Precision-Recall Curve class 5

- Tính Average Precision (AP): Để tính AP, ta tính diện tích dưới đường cong Precision-Recall. AP được tính bằng cách tích lũy diện tích dưới đường cong tại các điểm khác nhau của Recall:

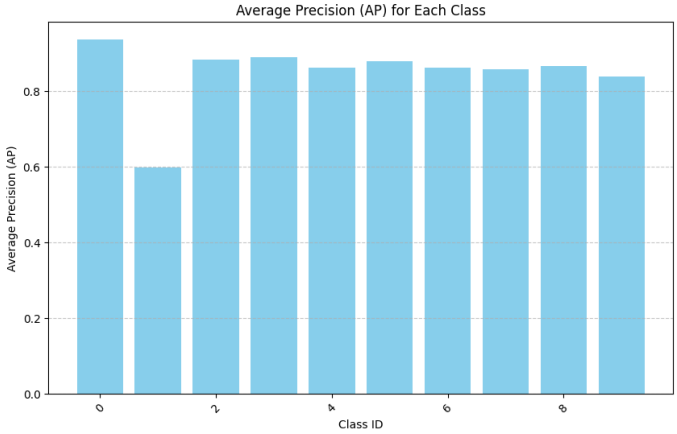
$$\text{AP} = \sum_{i=1}^{N} (R_i - R_{i-1}) \cdot P_i$$

Trong đó P_i và R_i là Precision và Recall tại điểm thứ i, và N là số lượng điểm trong đường cong.

- Tính mAP: Mean Average Precision (mAP) là giá trị trung bình của các AP tính được cho tất cả các lớp đối tượng. mAP được tính bằng:

$$\text{mAP} = \frac{1}{N} \sum_{c=1}^{N} \text{AP}_c$$

trong đó N là số lượng lớp đối tượng.



Hình 14: Ap at Different confident threshsold

13.6 Ý nghĩa của mAP

mAP cung cấp một cái nhìn tổng thể về hiệu suất của mô hình trong việc phát hiện các đối tượng trong hình ảnh. Một giá trị mAP cao cho thấy mô hình có khả năng phát hiện và phân loại chính xác các đối tượng trong nhiều tình huống khác nhau. Tuy nhiên, mAP có thể bị ảnh hưởng bởi các yếu tố như chất lượng của dữ liệu huấn luyện, độ phân giải của ảnh, và ngưỡng IoU được chọn.

Các mô hình với mAP cao thường thể hiện khả năng xử lý tốt các đối tượng trong các điều kiện thực tế, bao gồm cả các đối tượng nhỏ, các đối tượng bị che khuất, và các đối tượng có sự thay đổi về hình dạng hoặc kích thước.

14 Kết luận

Nhận diện chữ viết tay là một trong những bài toán nổi bật trong lĩnh vực học máy và thị giác máy tính. Tập dữ liệu MNIST, với 60,000 ảnh huấn luyện và 10,000 ảnh kiểm tra, là một nguồn tài nguyên phổ biến để thử nghiệm và đánh giá các mô hình học máy. Trong nghiên cứu này, nhóm đã triển khai một hệ thống nhận diện chữ viết tay sử dụng các phương pháp học sâu để đạt được độ chính xác cao trên tập dữ liệu MNIST.

Để xây dựng mô hình, nhóm sử dụng mạng neural sâu (deep neural network) với các lớp ẩn và hàm kích hoạt ReLU, kết hợp với các kỹ thuật tối ưu hóa như gradient descent. Bằng việc áp dụng các phương pháp điều chỉnh học như dropout và early stop, nhóm đã có thể giảm thiểu hiện tượng overfitting và cải thiện độ chính xác của mô hình.

Kết quả thu được cho thấy mô hình của nhóm có thể phân loại chính xác hơn 98% các chữ viết tay trên tập kiểm tra, vượt qua nhiều phương pháp truyền thống. Điều này chứng minh rằng các mô hình học sâu có thể xử lý hiệu quả các bài toán nhận diện hình ảnh phức tạp.

Trong tương lai, nhóm dự định mở rộng nghiên cứu để cải thiện mô hình bằng cách sử dụng các kiến thức chuyên môn trong lĩnh vực nhận dạng chữ viết tay, cũng như áp dụng các mô hình phức tạp hơn như Convolutional Neural Networks (CNNs) và các kỹ thuật học không giám sát (unsupervised learning) để tối ưu hóa độ chính xác. Hơn nữa, nhóm sẽ nghiên cứu việc áp dụng các chiến lược học tăng cường (transfer learning) để tăng cường khả năng nhận diện trong các tình huống thực tế với dữ liệu không đồng nhất.

15 Đóng góp

Thành viên	Vai trò	Công việc
Trần Lê Dũng	Nhóm trưởng	Viết báo cáo, viết mã nguồn, làm slide, thuyết trình
Nguyễn Vũ Minh Đức	Thành viên	Viết báo cáo, viết mã nguồn, làm slide
Lê Trường Uy	Thành viên	Viết báo cáo, viết mã nguồn, làm slide
Nguyễn Văn Khỏe	Thành viên	Viết báo cáo, viết mã nguồn, làm slide, thuyết trình
Nguyễn Thế Nhật Minh	Thành viên	Viết báo cáo, viết mã nguồn, làm slide, thuyết trình

Tài liệu

[1] Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification".-https://arxiv.org/pdf/1502.01852v1

[2] *Softmax Regression*. `https://machinelearningcoban.com/2017/02/17/softmax/`

[3] *Yolov3 an incremental improvement*. `https://ui.adsabs.harvard.edu/abs/2018arXiv180402767R/abs`

[4] *Hàm kích hoạt*. `https://arxiv.org/pdf/1502.01852v1`