Đề tài 9: Xây dựng mô hình phân loại chữ số viết tay

Đại học Khoa học tự nhiên

Đại học quốc gia Hà Nội, VNU

Thành viên nhóm:
Đinh Mạnh Cường - 21001535
Nguyễn Tiến Đạt - 21001544
Nguyễn Quang Huy - 21001553

Giảng viên:
Cao Văn Chung

Hà Nội: Ngày 5 tháng 5 năm 2024

Mục lục

Danh sách hình vẽ

Chương 1

Giới thiệu

1.1 Lĩnh vực nhận diện chữ số viết tay

Nhận diện chữ số viết tay là một lĩnh vực nghiên cứu thu hút sự quan tâm đáng kể trong cộng đồng học máy và trí tuệ nhân tạo. Khả năng của máy tính để tự động nhận dạng và phân loại các chữ số do con người viết tay mang lại tiềm năng to lớn cho nhiều ứng dụng thực tế, bao gồm nhận dạng biển số xe, sắp xếp thư bưu điện, xử lý séc ngân hàng, và hỗ trợ người khiếm thị.

Mặc dù có sự tiến bộ đáng kể trong lĩnh vực này nhờ vào sự phát triển của các thuật toán học máy và thị giác máy tính, nhưng việc nhận diện chữ số viết tay vẫn đặt ra nhiều thách thức. Điều này là do sự đa dạng và phức tạp của các phong cách viết của con người, cũng như sự biến đổi của các điều kiện ánh sáng và nền tảng trong quá trình chụp ảnh.

Tuy nhiên, với sự tiến bộ trong các kỹ thuật học sâu như mạng nơ-ron tích chập (CNN), các nhà nghiên cứu đã đạt được những bước tiến quan trọng trong việc nâng cao hiệu suất của hệ thống nhận diện chữ số viết tay. Sự kết hợp giữa các phương pháp tiền xử lý dữ liệu, trích xuất đặc trưng và các mô hình học sâu đã giúp cải thiện đáng kể khả năng nhận diện và phân loại chữ số từ hình ảnh

1.2 Mục đích và Phạm vi nghiên cứu

Báo cáo này trình bày một nghiên cứu về việc xây dựng mô hình nhận diện chữ viết tay sử dụng các kỹ thuật về học máy. Mục tiêu của nghiên cứu là phát triển một mô hình có khả năng nhận diện chính xác các chữ số viết tay từ hình ảnh. Để đạt được mục tiêu này, báo cáo trình bày chi tiết các bước xây dựng mô hình như sau:

- Tiền xử lý dữ liệu: Phân tích tập dữ liệu gồm các hình ảnh chữ số viết tay, giảm chiều dữ liệu bằng phương pháp PCA, và nhóm các hình ảnh chữ số viết tay vào các cụm dựa trên đặc trưng hình ảnh.
- Trích xuất đặc trưng: Sử dụng các thuật toán phân cụm để trích xuất các đặc trưng hình ảnh từ các nhóm chữ số viết tay.
- Phân loại: Huấn luyện một mạng nơ-ron tích chập (CNN) để phân loại các hình ảnh chữ số viết tay dựa trên các đặc trưng được trích xuất.
- Đánh giá: Đánh giá hiệu suất của hệ thống bằng các chỉ số như độ chính xác, ma trận nhầm lẫn, recall và precision.

Chương 2

Giảm chiều dữ liệu bằng phương pháp PCA

2.1 Mô tả tập dữ liệu ảnh chữ số viết tay

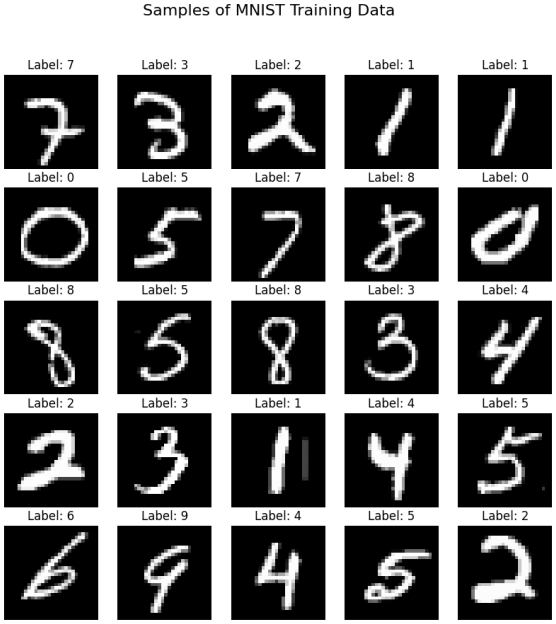
2.1.1 Tập dữ liệu MNIST

Tập dữ liệu MNIST là bộ cơ sở dữ liệu lớn nhất về chữ số viết tay được giới thiệu năm 1998 bởi Yann Lecun và cộng sự nhằm đánh giá các mô hình phân lớp.

MNIST bao gồm 2 tập con:

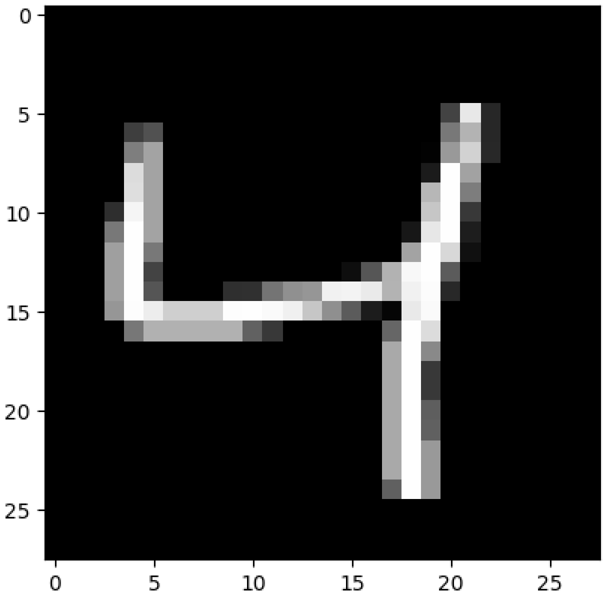
- Tập dữ liệu huấn luyện (training set): bao gồm tổng cộng 60,000 ví dụ về các chữ số viết tay từ 0 đến 9.
- Tập dữ liệu kiểm tra (test set): chứa 10,000 ví dụ khác nhau.

Tất cả đều được gãn nhán. Hình dưới đây là ví dụ về một số hình ảnh được trích ra từ MNIST.

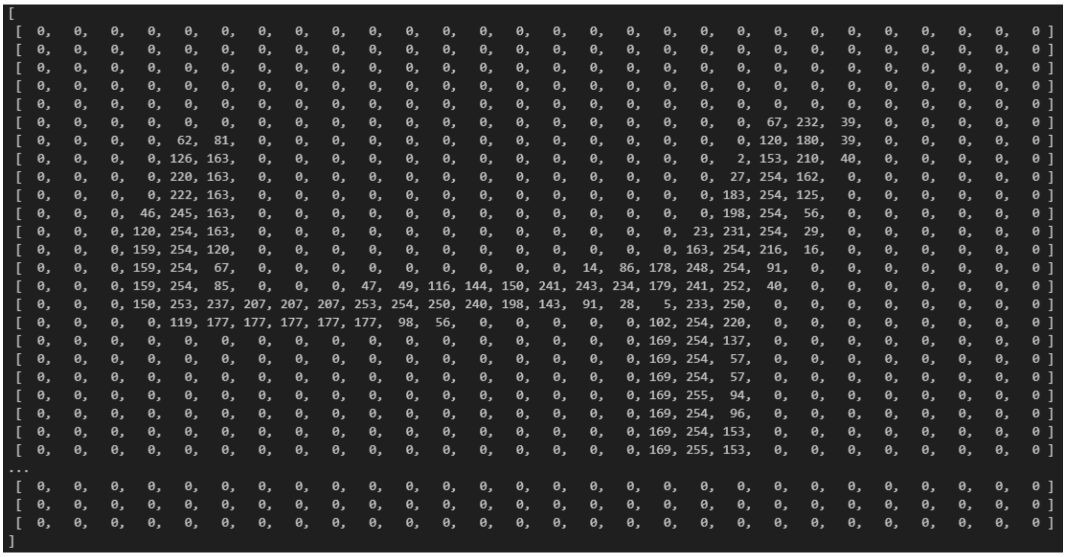


Hình 2.1: Ảnh chữ số viết tay

Mỗi bức ảnh là một ảnh đen trắng (có 1 channel), có kích thước 28x28 pixel (tổng cộng 784 pixels). Mỗi pixel mang một giá trị là một số tự nhiên từ 0 đến 255. Các pixel màu đen có giá trị bằng 0, các pixel càng trắng thì giá trị càng cao (nhưng không quá 255). Dưới dây là một ví dụ về chữ số 4 và giá trị các pixel của nó.



Hình 2.2: Ví dụ về chữ số 4



```
[
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0 ]
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0 ]
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0 ]
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0 ]
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0 ]
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  67, 232,  39,   0,   0,   0,   0,   0 ]
 [  0,   0,   0,   0,  62,  81,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 120, 180,  39,   0,   0,   0,   0,   0 ]
 [  0,   0,   0,   0, 126, 163,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   2, 153, 210,  40,   0,   0,   0,   0,   0 ]
 [  0,   0,   0,   0, 220, 163,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  27, 254, 162,   0,   0,   0,   0,   0,   0 ]
 [  0,   0,   0,   0, 222, 163,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 183, 254, 125,   0,   0,   0,   0,   0,   0,   0 ]
 [  0,   0,   0,  46, 245, 163,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 198, 254,  56,   0,   0,   0,   0,   0,   0,   0 ]
 [  0,   0,   0, 120, 254, 163,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  23, 231, 254,  29,   0,   0,   0,   0,   0,   0,   0 ]
 [  0,   0,   0, 159, 254, 120,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 163, 254, 216,  16,   0,   0,   0,   0,   0,   0,   0 ]
 [  0,   0,   0, 159, 254,  67,   0,   0,   0,   0,   0,   0,   0,   0,   0,  14,  86, 178, 248, 254,  91,   0,   0,   0,   0,   0,   0,   0 ]
 [  0,   0,   0, 159, 254,  85,   0,   0,   0,  47,  49, 116, 144, 150, 241, 243, 234, 179, 241, 252,  40,   0,   0,   0,   0,   0,   0,   0 ]
 [  0,   0,   0, 150, 253, 237, 207, 207, 207, 253, 254, 250, 240, 198, 143,  91,  28,   5, 233, 250,   0,   0,   0,   0,   0,   0,   0,   0 ]
 [  0,   0,   0,   0, 119, 177, 177, 177, 177, 177,  98,  56,   0,   0,   0,   0,   0, 102, 254, 220,   0,   0,   0,   0,   0,   0,   0,   0 ]
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 169, 254, 137,   0,   0,   0,   0,   0,   0,   0,   0 ]
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 169, 254,  57,   0,   0,   0,   0,   0,   0,   0,   0 ]
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 169, 254,  57,   0,   0,   0,   0,   0,   0,   0,   0 ]
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 169, 255,  94,   0,   0,   0,   0,   0,   0,   0,   0 ]
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 169, 254,  96,   0,   0,   0,   0,   0,   0,   0,   0 ]
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 169, 254, 153,   0,   0,   0,   0,   0,   0,   0,   0 ]
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 169, 255, 153,   0,   0,   0,   0,   0,   0,   0,   0 ]
...
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0 ]
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0 ]
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0 ]
]
```

Hình 2.3: Ma trận pixel của ví dụ về chữ số 4

2.1.2 Đặt vấn đề

Những tập dữ liệu lớn như MNIST (một tensor 3 chiều (n,28,28) nếu giữ nguyên cấu trúc ma trận cho mỗi ảnh, hoặc một ma trận 2 chiều (n,784) khi làm phẳng n ảnh) thường tiêu tốn nhiều bộ nhớ lưu trữ và thời gian huấn luyện. Vì vậy giảm số chiều dữ liệu là một bước quan trọng trong nhiều bài toán. Đây cũng được coi là một phương pháp nén dữ liệu.

Báo cáo này sẽ trình bày một phương pháp đơn giản nhất trong các thuật toán giảm số chiều dựa trên một mô hình tuyến tính: Phân tích thành phần chính (PCA).

2.2 Phương pháp PCA

2.2.1 Giới thiệu

Phương pháp phân tích thành phần chính (Principal Component Analysis - PCA) là phương pháp được dùng để giảm số chiều của một bộ dữ liệu mà trong đó có một số lượng lớn các biến có tương quan với nhau, trong khi vẫn giữ được lượng thông tin nhiều nhất có thể của bộ dữ liệu gốc.

Việc giảm số chiều được thực hiện bởi việc biến đổi tập hợp các biến gốc sang một tập hợp các biến mới, gọi là "các thành phần chính"(principal component). Các thành phần mới này không tương quan với nhau và có thứ tự sao cho một vài thành phần đầu tiên giữ lại được nhiều nhất thông tin có thể của bộ dữ liệu.

2.2.2 Một số ứng dụng phổ biến của PCA

- Dùng để nghiên cứu cấu trúc phương sai - hiệp phương sai của dữ liệu dựa trên việc phân tích ma trận phương sai của mẫu **S** hay ma trận hệ số tương quan **R**.
- Tạo ra một tập hợp biến mới (với một số lượng nhỏ hơn và không có tương quan với nhau). Tập hợp các biến mới này được sử dụng cho các phân tích thống kê khác (ví dụ như hồi quy bội).
- Phát hiện các điểm outliers hay phân cụm dữ liệu (cluster analysis).
- Kiểm tra giả định về phân phối chuẩn nhiều chiều (được sử dụng cho các phân tích thống kê nhiều chiều khác).

2.2.3 Cách tiếp cận PCA

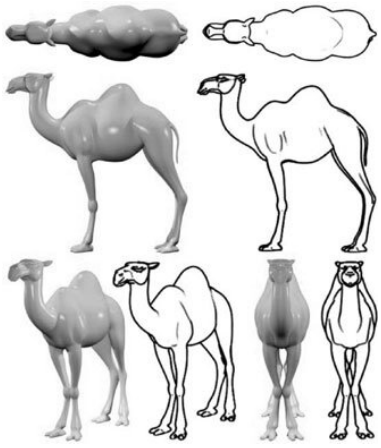
Đại số:

Các thành phần chính là tổ hợp tuyến tính các p biến gốc $X_1, X_2, ..., X_p$ sao cho:

- thành phần chính thứ nhất có phương sai lớn nhất có thể
- thành phần chính thứ hai có phương sai lớn nhất có thể và trực giao với thành phần chính thứ nhất
- ...

Hình học:

- Thực hiện phép xoay để chuyển về một hệ trục toạ độ mới có khả năng biểu diễn dữ liệu tốt hơn, cho phép khám phá được những liên kết tiềm ẩn trong bộ dữ liệu cũ mà ta không thể thấy rõ trên hệ trục toạ độ cũ. Các trục toạ độ mới phải được đảm bảo là luôn trực giao đôi một với nhau.



Hình 2.4: Hình ảnh minh hoạ cách tiếp cận hình học của phương pháp PCA

Ảnh trên minh hoạ cách tiếp cận hình học của phương pháp PCA. Rõ ràng việc chọn hệ trục toạ độ quan sát phần ngang lạc đà sẽ thu được nhiều thông tin hơn việc quan sát từ trên lưng lạc đà.

2.2.4 Các bước thực hiện PCA

1. Tính vector kỳ vọng của toàn bộ dữ liệu:

$$\overline{x} = \frac{1}{N}\sum_{n=1}^{N} x_n \tag{2.1}$$

2. Trừ mỗi điểm dữ liệu đi vector kỳ vọng của toàn bộ dữ liệu:

$$\hat{x}_n = x_n - \overline{x} \tag{2.2}$$

3. Tính ma trận hiệp phương sai:

$$S = \frac{1}{N}\hat{X}\hat{X}^T \tag{2.3}$$

4. Tính giá trị riêng bằng cách giải phương trình đặc trưng:

$$det(S - \lambda I) = 0 \tag{2.4}$$

Sau đó tính các vecto riêng u_i ứng với giá trị riêng λ_i, có $||u_i||_2 = 1$ của ma trận hiệp phương sai, sắp xếp chúng theo thứ tự giảm dần của trị riêng.

5. Chọn K vecto riêng ứng với K trị riêng lớn nhất để xây dựng ma trận U_k có các cột tạo thành một hệ trực giao. K vectors này, còn được gọi là các thành phần chính, tạo thành một không gian con gần với phân bố của dữ liệu ban đầu đã chuẩn hoá.

6. Chiếu dữ liệu ban đầu đã chuẩn hoá $\hat{X}$ xuống không gian con tìm được.

7. Dữ liệu mới chính là toạ độ của các điểm dữ liệu trên không gian mới:

$$Z = U_K^T \hat{X} \tag{2.5}$$

Dữ liệu ban đầu có thể tính được xấp xỉ theo dữ liệu mới như sau:

$$x \approx U_K Z + \overline{x} \tag{2.6}$$

2.3 Thực nghiệm và kết quả

Input là file gzip "train-images-idx3-ubyte"gồm 60000 hình ảnh các chữ số viết tay từ 0 đến 9. Các hình ảnh được lưu dưới dạng grayscale, mỗi hình ảnh có kích thước 28x28 pixel.

Yêu cầu: Thực hiện việc rút gọn số chiều dữ liệu của tập dữ liệu input, sau đó hiển thị trực quan các phân lớp dữ liệu dưới dạng 2D.

Phương pháp đề xuất: Phân tích thành phần chính (PCA).

2.3.1 Thực hiện

Xây dựng chương trình dựa trên các bước thực hiện PCA:

Bước 1: Chuẩn hoá và tính ma trận hiệp phương sai của dữ liệu.

Bước 2: Tính hệ riêng của ma trận hiệp phương sai bằng cả 2 công cụ của thư viện Numpy.Linalg là SVD (khai triển kỳ dị) và Eigh (Khai triển hệ giá trị-vector riêng).

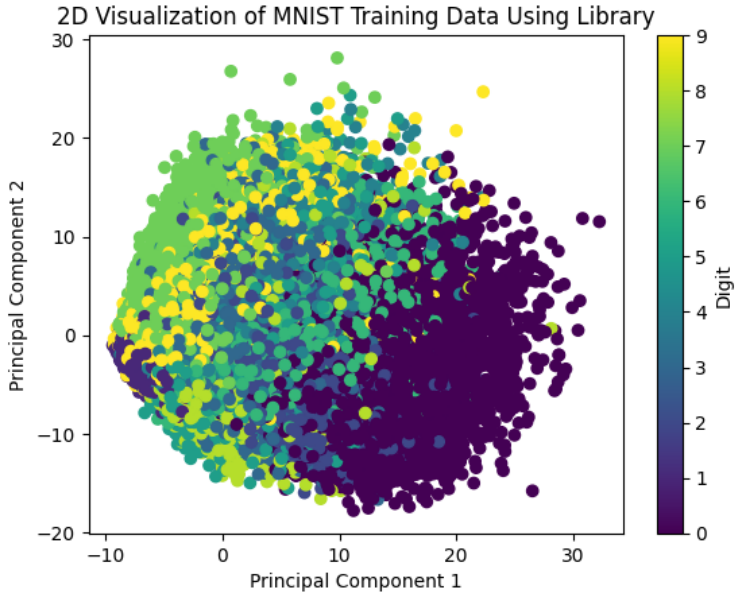
Bước 3: Chiếu dữ liệu (X) xuống không gian con 02 chiều ứng với 2 giá trị riêng lớn nhất (trị riêng [0] và trị riêng [1]).

Bước 4: Vẽ dữ liệu trong không gian 2 chiều mới thành lập để quan sát.

2.3.2 Kết quả và đánh giá phương pháp

Kết quả

Hình 2.5: Trực quan hoá dưới dạng 2D bằng PCA theo các bước xây dựng



Hình 2.6: Trực quan hoá dưới dạng 2D bằng PCA theo thư viện

Đánh giá

PC1 (khoảng -10 đến 30):

- Biểu thị sự phân bố của điểm dữ liệu theo hướng của phương sai lớn nhất trong dữ liệu sau khi giảm chiều.
- Liên quan đến các đặc điểm như độ dày tổng thể của nét vẽ chữ số, độ tương phản của hình ảnh.
- Giá trị cao: Nét vẽ dày, hình ảnh tương phản cao.
- Giá trị thấp: Nét vẽ mảnh, hình ảnh tương phản thấp.
- Giúp phân biệt các chữ số có hình dạng khác nhau.

PC2 (khoảng -20 đến 30):

- Biểu thị sự phân bố của điểm dữ liệu theo hướng của phương sai lớn thứ hai sau PC1.
- Liên quan đến các đặc điểm như độ nghiêng của chữ số, vị trí của chữ số trong khung hình.
- Giá trị cao: Chữ số bị nghiêng nhiều, nằm ở vị trí cao trong khung hình.
- Giá trị thấp: Chữ số đứng thẳng hơn, nằm ở vị trí thấp hơn trong khung hình.
- Giúp phân biệt các chữ số có hình dạng tương tự nhưng có độ nghiêng hoặc vị trí khác nhau.

Chương 3

Thuật toán phân cụm

3.1 Đề bài và vấn đề cần giải quyết

Nhiệm vụ là thực hiện phân cụm (clustering) trên tập dữ liệu ảnh chữ số viết tay được cung cấp. Mục đích là nhóm các ảnh tương tự về hình dạng, kích thước hoặc đặc trưng khác vào cùng một cụm để được các cụm có cùng chữ số. Sau khi phân cụm, yêu cầu hiển thị trực quan các cụm ảnh đã được phân loại bằng cách gán nhãn hoặc màu sắc khác nhau cho chúng. Việc phân cụm dữ liệu ảnh này là bước đầu tiên quan trọng để huấn luyện mô hình nhận dạng chữ số viết tay và tổ chức, quản lý dữ liệu hiệu quả hơn.

3.2 Mô hình sử dụng

3.2.1 Giới thiệu thuật toán Kmean

Thuật toán K-means [4] là thuật toán phân cụm không giám sát, chia dữ liệu thành K cụm dựa trên khoảng cách. Các bước chính bao gồm khởi tạo ngẫu nhiên K tâm cụm, gán điểm vào cụm gần nhất, tính lại tâm cụm mới và lặp lại cho đến khi không đổi. Thuật toán đơn giản, hiệu quả nhưng hạn chế là khó xác định số cụm K phù hợp, dễ bị kẹt cực trị cục bộ, nhạy cảm với nhiễu.

3.2.2 Nguyên lý hoạt động

a) Khởi tạo tâm cụm ban đầu

Đầu tiên, chúng tôi sẽ khởi tạo K tâm cụm ban đầu một cách ngẫu nhiên từ tập dữ liệu là một bước quan trọng trong thuật toán K-Means. Tuy nhiên, cách khởi tạo này cũng tiềm ẩn rủi ro và hạn chế nhất định.

Nếu các tâm cụm ban đầu được khởi tạo ở những vị trí "thuận lợi"trong không gian dữ liệu, thuật toán sẽ hội tụ nhanh và có khả năng cao đạt được kết quả phân cụm tối ưu. Ngược lại, nếu các tâm cụm ban đầu không phù hợp, thuật toán có thể bị kẹt ở cực trị cục bộ và đưa ra kết quả kém chất lượng.

b) Gán mỗi điểm dữ liệu vào cụm có tâm gần nhất.

Quá trình này thực hiện việc tính toán ma trận khoảng cách từ mỗi điểm dữ liệu đến tất cả các tâm đã được xác định trước đó. Sau đó, mỗi điểm dữ liệu sẽ được gán vào cụm chứa tâm gần nhất, tức là cụm mà khoảng cách từ điểm đó đến tâm của cụm đó là nhỏ nhất.

Để làm điều này, ta cần tính toán khoảng cách giữa mỗi điểm dữ liệu với mỗi tâm bằng cách sử dụng một phép đo khoảng cách Euclidean . Sau đó, ta chọn tâm gần nhất với mỗi điểm dữ liệu và gán điểm đó vào cụm của tâm đó. Quá trình này được thực hiện cho tất cả các điểm dữ liệu trong tập dữ liệu.

$$d(x, c_k) = ||x - c_k|| = \sqrt{(x_1 - c_{k1})^2 + (x_2 - c_{k2})^2 + \ldots + (x_s - c_{ks})^2} \tag{3.1}$$

Trong đó x là điểm dữ liệu, c_k là điểm trung tâm của cluster tương ứng, $d(x, c_k)$ là khoảng cách Euclid giữa điểm dữ liệu x và tâm cụm c_k, s là số chiều của một điểm dữ liệu.

c) Xác định trung tâm các cụm.

Tính lại các tâm cụm mới bằng cách lấy trung bình của tất cả các điểm trong cùng cụm sau khi đã được cập nhật.

$$m_i = \frac{1}{|C_i|} \sum_{x \in C_i} x \tag{3.2}$$

Trong đó C_i là cluster thứ i, $m_i$là điểm trung tâm của cluster tương ứng.

d) Điều kiện dừng của thuật toán

Lặp lại bước (b) và (c) cho đến khi các tâm cụm không thay đổi nữa (hội tụ) hoặc đạt số vòng lặp tối đa. Để đánh giá sự hội tụ ta cần xem xét sự thay đổi của tâm cụm mới so với tâm cụm cũ bằng hàm mất mát [5]:

$$\text{Error} = \sum_{i=1}^{n} \sum_{k=1}^{c} r_{nk} ||\mathbf{x}_i - \mathbf{c}_k||_2^2 \tag{3.3}$$

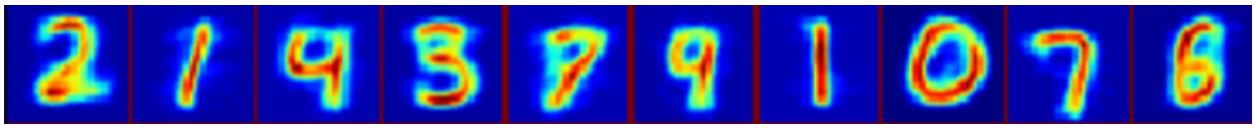
Trong đó C là số lượng cụm, $||\mathbf{x}_i - \mathbf{c}_k||_2^2$ là bình phương khoảng cách Euclid giữa điểm dữ liệu x và tâm cụm c_k, r_{nk} = 1 nếu x_i thuộc k và r_{nk} = 0 nếu x_i không thuộc k.

Tổng của các bình phương khoảng cách này đo lường tổng sai số bình phương giữa mỗi điểm dữ liệu và tâm cụm tương ứng. Mục tiêu của thuật toán K-Means là tìm các tâm cụm sao cho tổng sai số này là nhỏ nhất. Khi cập nhật vị trí của các tâm cụm và gán lại các điểm dữ liệu vào các cụm, hàm mất mát này được sử dụng để đánh giá hiệu suất của mô hình K-Means. Điều này giúp thuật toán hội tụ đến một giải pháp tối ưu hơn theo thời gian.

3.3 Thực nghiệm và kết quả

3.3.1 Đầu vào

Đầu vào là dữ liệu gồm 1000 ảnh chữ số viết tay có kích thước 28x28 tức là 784 pixels được gán vào biến dữ liệu X.



Hình 3.1: Hình ảnh minh họa chữ viết tay

Trước khi thực hiện phân cụm ta cần phải xác định rõ đâu là điểm dữ liệu phân cụm. X ở đây sẽ là một ma trận có chiều dạng (1000,784) tức là mỗi hàng sẽ là dữ liệu 784 pixel của mỗi ảnh còn số cột là 1000 ảnh được thêm vào. Ta coi mỗi hàng của ma trận X là một điểm dữ liệu và nhiệm vụ là dựa vào sự tương đồng đặc tính của mỗi ảnh để phân cụm chúng. Dữ liệu hình ảnh có 10 chữ số đồng nghĩa với việc chúng tôi sẽ phân các hình ảnh thành 10 cụm.

3.3.2 Thực hiện

Bước 1: Chọn ngẫu nhiên 10 tâm cho dữ liệu X với mỗi tâm là ma trận 28x28 tương đương với 784 điểm ảnh (pixel).

Bước 2: Tính toán ma trận khoảng cách D, trong đó mỗi phần tử $D[i, j]$ là khoảng cách Euclid giữa điểm dữ liệu thứ i và điểm tâm thứ j, với i chạy từ 1 đến 1000 (số lượng ảnh) và j chạy từ 1 đến 10 (số lượng tâm). Đảm bảo rằng ma trận D có kích thước (1000, 10), tương ứng với khoảng cách của 1000 ảnh đến mỗi tâm trong số 10 tâm.

$$D = \begin{pmatrix} d_{1,1} & d_{1,2} & \cdots & d_{1,10} \\ d_{2,1} & d_{2,2} & \cdots & d_{2,10} \\ \vdots & \vdots & \ddots & \vdots \\ d_{1000,1} & d_{1000,2} & \cdots & d_{1000,10} \end{pmatrix} \tag{3.4}$$

Bước 3: Tìm giá trị khoảng cách nhỏ nhất của mỗi điểm dữ liệu (ảnh) đến các tâm cụm. Tạo một danh sách E có độ dài 1000, trong đó $E[i]$ lưu trữ chỉ số của tâm cụm mà điểm dữ liệu thứ i gần nhất với nó, có nghĩa là:

$$E[i] = \underset{j}{\operatorname{argmin}}\, D[i, j]$$

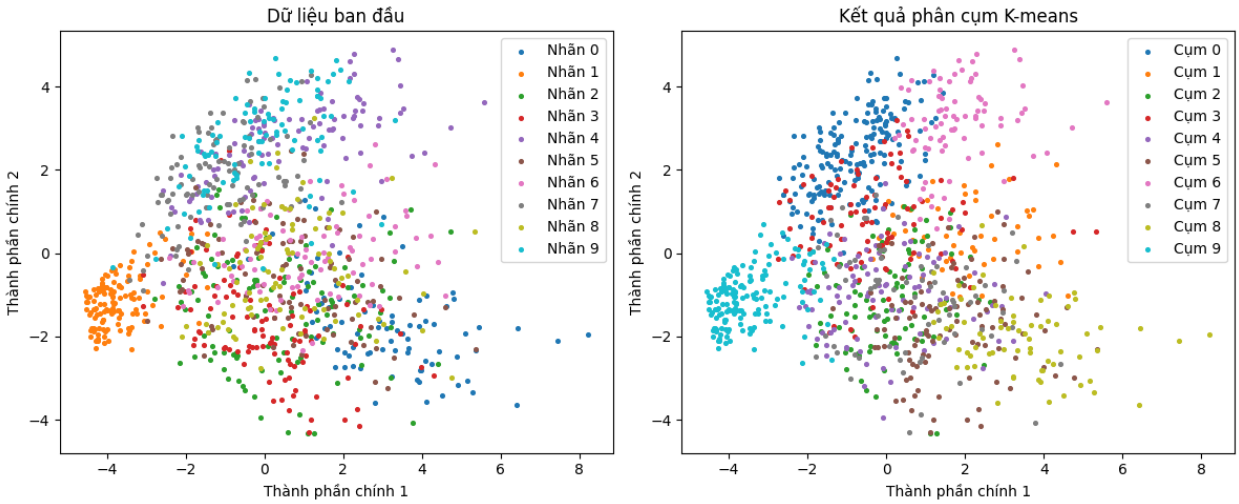
với i chạy từ 1 đến 1000 và j chạy từ 1 đến 10.

Bước 4: Tính toán tâm cụm mới bằng cách lấy trung bình của tất cả các điểm dữ liệu (ảnh) trong cùng một cụm. Sau khi tính toán được tâm cụm mới, lặp lại Bước 2 và Bước 3 với tập hợp tâm cụm mới vừa tính được. Quá trình này được lặp lại cho đến khi đạt điều kiện dừng, ví dụ như khi tâm cụm không thay đổi và đạt số lần lặp tối đa.

3.3.3 Kết quả và đánh giá mô hình

a) Biểu đồ điểm

Biểu đồ điểm là công cụ trực quan hiệu quả để biểu diễn kết quả phân cụm khi dữ liệu có số chiều bằng 2. Để xử lý dữ liệu đa chiều, chúng tôi sẽ sử dụng PCA để giảm xuống 2 chiều, sau đó vẽ biểu đồ điểm để đánh giá hiệu quả phân cụm trước và sau khi áp dụng thuật toán.



Hình 3.2: Biểu đồ phân cụm điểm của chữ viết tay

Nhận xét: Dựa trên kết quả hiển thị, có thể thấy rằng thuật toán K-means đã phân cụm dữ liệu một cách tương đối tốt. Mỗi cụm được phân tách rõ ràng và các điểm dữ liệu trong cùng một cụm có sự tập trung về mặt vị trí. Tuy nhiên, khi so sánh với dữ liệu ban đầu đã được gán nhãn, việc gán nhãn màu cho các cụm sau khi phân cụm không hoàn toàn trùng khớp với nhãn màu ban đầu. Điều này có thể là do quá trình phân cụm chỉ dựa trên đặc trưng vị trí mà không tính đến đặc trưng màu sắc. Do đó, mặc dù các cụm được phân tách tốt về mặt vị trí, nhưng việc gán nhãn màu cho chúng có thể không chính xác hoàn toàn so với dữ liệu gốc.

a) Biểu đồ hình ảnh

Một cách đánh giá trực quan khác, có thể thể hiện tốt hơn kết quả phân cụm, là hiển thị các mẫu ảnh đại diện cho mỗi cụm. Cụ thể, sau khi áp dụng thuật toán K-means để phân cụm dữ liệu, chúng ta sẽ chọn ngẫu nhiên 20 ảnh từ mỗi

cụm và hiển thị chúng trên một bảng ảnh. Trong bảng ảnh này, mỗi dòng tương ứng với một cụm, và các ảnh trong cùng một dòng đều thuộc về cụm đó.

Hình 3.3: Biểu đồ phân cụm dạng hình ảnh của chữ viết tay

Nhận xét: Từ biểu đồ hình ảnh, ta có thể nhận thấy rằng thuật toán phân cụm đã hoạt động tương đối tốt trong việc nhóm các chữ số tương tự vào cùng một cụm. Tuy nhiên, vẫn tồn tại một số nhầm lẫn giữa các chữ số có hình dạng gần giống nhau, chẳng hạn như số 7 với số 2 hoặc số 3 với số 8. Điều này cho thấy cần phải xem xét và điều chỉnh thuật toán K-means để phù hợp hơn với đặc thù của bài toán phân cụm ảnh chữ số viết tay.

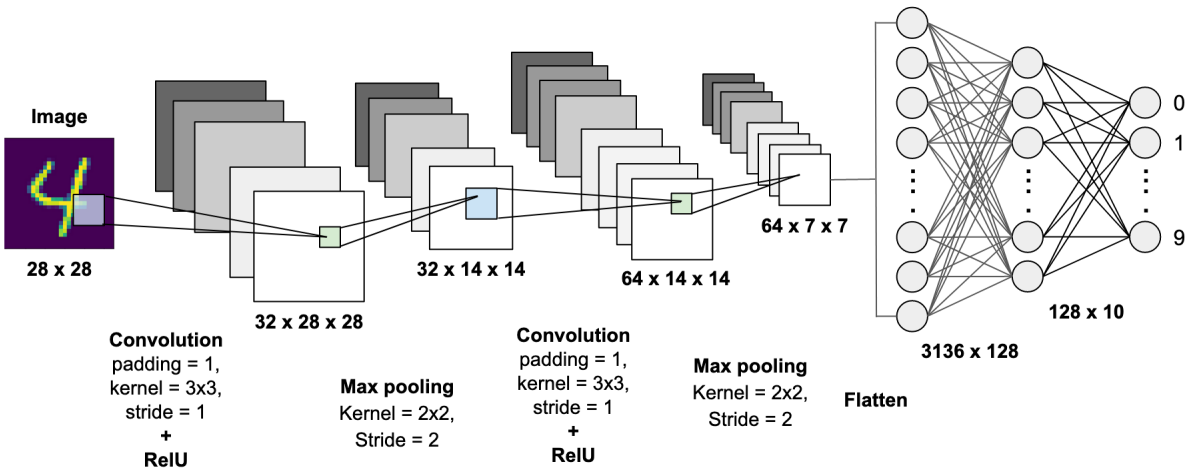
Chương 4

Xây dựng mô hình mạng Nơ-ron tích chập

4.1 Giới thiệu

Mạng Nơ-ron tích chập(CNN) là một trong những phương pháp mạnh mẽ nhất trong lĩnh vực xử lý ảnh và đã đạt được những thành tựu đáng kể trong nhiều ứng dụng nhận diện hình ảnh.

Phần này đề xuất một kiến trúc CNN cụ thể, bao gồm ba tầng tích chập hỗn hợp, mỗi tầng kết hợp với các lớp kích hoạt ReLU [5] và lớp Max Pooling [5]. Tiếp đó là hai tầng kết nối đầy đủ (fully connected layers) [5] được thiết kế với số units phù hợp để trích xuất và biểu diễn thông tin từ các đặc trưng đã được học từ ảnh. Cuối cùng, chúng tôi sử dụng một tầng quyết định được kích hoạt bởi hàm softmax để phân loại chính xác chữ số trong ảnh. [6]



Hình 4.1: Mô hình CNN

4.2 Feedforward

4.2.1 Lớp đầu vào

Trong quá trình Feedforward của mạng nơ-ron, lớp đầu vào chịu trách nhiệm tiếp nhận dữ liệu đầu vào và truyền nó qua mạng. Trong bài toán phân loại hình ảnh chữ số viết tay MNIST, kích thước của mỗi hình ảnh là 28x28 pixel và có một kênh màu (ảnh xám).

Do đó, lớp đầu vào của mô hình CNN được cấu hình để chấp nhận dữ liệu hình ảnh có kích thước 28x28x1, trong đó:

- 28x28 là kích thước của hình ảnh.
- 1 đại diện cho số kênh màu, vì ảnh là ảnh xám, nên chỉ có một kênh màu.

Lớp đầu vào không thực hiện bất kỳ phép biến đổi nào trên dữ liệu đầu vào, nó chỉ chuyển tiếp nó đến tầng tích chập đầu tiên của mạng.

Trong quá trình feedforward của mạng nơ-ron, dữ liệu di chuyển từ lớp đầu vào qua các lớp tiếp theo. Trong mạng CNN, quá trình này thường bắt đầu với tầng tích chập đầu tiên. Đầu ra của tầng tích chập này trở thành đầu vào cho các tầng tích chập tiếp theo. Kết quả cuối cùng của các tầng tích chập là đầu vào cho tầng kết nối đầy đủ (fully connected layer). Sau đó, kết quả của tầng kết nối đầy đủ được chuyển vào một lớp fully connected layer cuối cùng, thường kèm theo hàm kích hoạt softmax để tính xác suất của các lớp đầu ra. Đây là quy trình thông thường trong mạng CNN được sử dụng cho bài toán phân loại.

4.2.2 Tầng tích chập

Tầng tích chập là một phần quan trọng của mạng nơ-ron hồi quy tích chập (CNN), được sử dụng để trích xuất các đặc trưng từ dữ liệu hình ảnh. Tầng này hoạt động bằng cách di chuyển một bộ lọc (kernel hoặc filter) qua toàn bộ hình ảnh đầu vào và tính toán tích chập giữa kernel và mỗi phần của hình ảnh. [5]

Cho một hình ảnh đầu vào có kích thước $W_{\text{in}} \times H_{\text{in}} \times D_{\text{in}}$, với:

- W_{in} và H_{in} là chiều rộng và chiều cao của hình ảnh.
- D_{in} là số lượng kênh (hoặc chiều sâu) của hình ảnh.

Tầng tích chập sử dụng K bộ lọc có kích thước $F \times F \times D_{\text{in}}$, với F là kích thước của kernel và D_{in} là số lượng kênh của hình ảnh đầu vào. Khi tiến hành tích chập, mỗi bộ lọc sẽ di chuyển qua toàn bộ hình ảnh và tính toán tích chập tại từng vị trí. Kết quả của quá trình này là một feature map (bản đồ đặc trưng) có kích thước $W_{\text{out}} \times H_{\text{out}} \times K$, với W_{out} và H_{out} là kích thước của feature map và K là số lượng bộ lọc.

Công thức tính toán feature map $O_{i,j,k}$ tại vị trí (i, j) trên kênh thứ k là:

$$O_{i,j,k} = \sum_{l=0}^{F-1} \sum_{m=0}^{F-1} \sum_{n=0}^{D_{\text{in}}-1} I_{i+l,j+m,n} \times K_{l,m,n,k} + b_k$$

Trong đó:

- $I_{i+l,j+m,n}$ là giá trị của pixel tại vị trí $(i+l, j+m)$ trên kênh thứ n của hình ảnh đầu vào.
- $K_{l,m,n,k}$ là trọng số của kernel tại vị trí (l, m) trên kênh thứ n của bộ lọc thứ k.
- b_k là thiên vị (bias) của bộ lọc thứ k.

Tính toán feature map như vậy sẽ tạo ra một biểu diễn của hình ảnh đầu vào dưới dạng các đặc trưng được học được bởi mạng CNN.

Ví dụ:

Cho ma trận đầu vào I kích thước 4×4 và một bộ lọc K kích thước 3×3 như sau:

$$I = \begin{bmatrix} 1 & 2 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 2 & 2 & 0 & 1 \\ 1 & 0 & 2 & 2 \end{bmatrix}, \quad K = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

Quá trình tích chập sẽ diễn ra như sau:

- Tại vị trí $(0,0)$:

$$O_{0,0} = (1\times 1)+(2\times 0)+(1\times 1)+(0\times 0)+(1\times 1)+(0\times 0)+(2\times 1)+(2\times 0)+(0\times 1) = 5$$

- Tại các vị trí khác sẽ tính tương tự và di chuyển bộ lọc qua ma trận đầu vào.

Kết quả sẽ là một feature map O kích thước 2×2 như sau:

$$O = \begin{bmatrix} 5 & 6 \\ 6 & 7 \end{bmatrix}$$

Sau khi tính toán tích chập giữa kernel và mỗi phần của hình ảnh, một bước quan trọng tiếp theo là thêm thiên vị vào kết quả tích chập. Thiên vị (bias) là một tham số được học trong quá trình huấn luyện mô hình và giúp điều chỉnh đầu ra của tầng tích chập.

Giả sử kết quả tích chập tại mỗi vị trí (i,j) được ký hiệu là $O_{i,j}$. Sau khi tính toán, chúng ta thêm thiên vị b vào mỗi $O_{i,j}$ như sau:

$$O_{i,j} = O_{i,j} + b$$

Ở đây, b là một vector thiên vị bias có cùng kích thước với kết quả tích chập O. Cụ thể, nếu O là một ma trận kích thước $m\times n$, thì b sẽ là một vector kích thước $m\times n$, trong đó mỗi phần tử của b được thêm vào tương ứng với mỗi phần tử của O.

Việc này giúp tăng tính linh hoạt và khả năng học của mạng, bằng cách cho phép các tầng tích chập tạo ra các biểu diễn phức tạp hơn từ dữ liệu đầu vào.

Trong ví dụ trước về tích chập một phần của hình ảnh, kết quả tích chập sẽ được cộng với thiên vị b. Ví dụ:

$$O = \begin{bmatrix} 5+b & 6+b \\ 6+b & 7+b \end{bmatrix}$$

Ở đây, b_k là thiên vị bias của bộ lọc thứ k.

Kết quả này sẽ tạo thành feature map cuối cùng và được đưa qua hàm kích hoạt để tạo ra biểu diễn cuối cùng của hình ảnh.

4.2.3 Hàm kích hoạt ReLU

ReLU (Rectified Linear Unit) là một hàm kích hoạt phổ biến trong các mạng neural, đặc biệt là trong các tầng tích chập của mạng CNN. Hàm này giúp tăng tính phi tuyến tính của mô hình và giảm vấn đề biến mất đạo hàm (vanishing gradient) trong quá trình lan truyền ngược. Nguyên do của việc tăng mức độ phi tuyến tính của mô hình thì ở phần trước, tầng tích chập; tất cả các phép toán được thực hiện đều là phép toán tuyến tính. [5]

ReLU được định nghĩa bởi công thức toán học sau:

$$f(x) = \max(0,x)$$

Trong ví dụ trước về tầng tích chập, chúng ta đã tính toán feature map O sau khi áp dụng tích chập và thiên vị bias như sau:

$$O = \begin{bmatrix} 5+b & 6+b \\ 6+b & 7+b \end{bmatrix}$$

Khi áp dụng hàm kích hoạt ReLU lên feature map O bao gồm thiên vị bias, ta có:

$$f(O) = \begin{bmatrix} 5+b & 6+b \\ 6+b & 7+b \end{bmatrix}$$

Ở đây, vì tất cả các giá trị trong feature map O là số không âm, nên khi áp dụng hàm kích hoạt ReLU, chúng không thay đổi. Cụ thể, hàm kích hoạt ReLU không ảnh hưởng đến các giá trị dương và chuyển đổi các giá trị âm thành 0.

4.2.4 Maxpooling

Maxpooling là một kỹ thuật quan trọng trong mạng neural tích chập (CNN), được sử dụng để giảm kích thước của feature map và trích xuất các đặc trưng quan trọng của mẫu dữ liệu. [5] [6]

Maxpooling thực hiện việc chia feature map thành các ô nhỏ không chồng chéo (ví dụ: ô kích thước 2×2 hoặc 3×3) và chọn giá trị lớn nhất từ mỗi ô làm đại diện cho ô đó. Kết quả là một feature map mới với kích thước giảm đi một cách đáng kể. [5]

Cho một feature map F kích thước $W \times H \times D$, với: - W và H là chiều rộng và chiều cao của feature map. - D là số lượng kênh của feature map. Maxpooling thực hiện việc chia feature map thành các ô $n \times n$ (thường là 2×2 hoặc 3×3) và chọn giá trị lớn nhất từ mỗi ô. Kết quả là một feature map mới với kích thước giảm xuống $\frac{W}{n} \times \frac{H}{n} \times D$.

Ví dụ:

Áp dụng maxpooling lên feature map sau:

$$F = \begin{bmatrix} 5 & 6 & 8 & 9 \\ 6 & 4 & 2 & 3 \\ 8 & 5 & 7 & 4 \\ 3 & 2 & 1 & 6 \end{bmatrix}$$

Với kích thước của mỗi ô maxpool là 2×2, ta chia feature map thành các ô như sau:

$$\left[\begin{array}{cc|cc} 5 & 6 & 8 & 9 \\ 6 & 4 & 2 & 3 \\ \hline 8 & 5 & 7 & 4 \\ 3 & 2 & 1 & 6 \end{array}\right]$$

Sau đó, chúng ta chọn giá trị lớn nhất từ mỗi ô:

$$\begin{bmatrix} 6 & 9 \\ 8 & 7 \end{bmatrix}$$

Kết quả là một feature map mới có kích thước giảm đi 2 lần so với feature map ban đầu.

Maxpooling không chỉ giúp giảm kích thước của feature map, mà còn giúp giảm nguy cơ overfitting bằng cách giảm số lượng tham số cần học trong mạng. Bằng cách giữ lại các đặc trưng quan trọng nhất từ mỗi vùng của hình ảnh, maxpooling tạo ra tính bất biến về vị trí. Điều này có nghĩa là dù một đối tượng di chuyển trong hình ảnh, các đặc trưng quan trọng sẽ vẫn được giữ lại. Điều này cải thiện khả năng tổng quát hóa của mô hình và giúp tránh overfitting, khiến cho mô hình hiệu quả hơn khi đối mặt với dữ liệu mới.

4.2.5 Dropout

Dropout là một kỹ thuật regularization phổ biến trong mạng neural, được sử dụng để giảm overfitting và cải thiện khả năng tổng quát hóa của mô hình. Kỹ thuật này hoạt động bằng cách ngẫu nhiên loại bỏ một số lượng ngẫu nhiên các đơn vị (units) trong quá trình huấn luyện, tức là đặt giá trị của chúng bằng 0. [5] [7]

Trong mỗi lượt huấn luyện, mỗi đơn vị sẽ bị tắt với xác suất được xác định trước, thường là giữa 0.2 và 0.5. Quá trình này tạo ra một mạng con khác nhau cho mỗi lượt huấn luyện, giúp mô hình học các đặc trưng phân loại dữ liệu một cách tổng quát hơn.

Dropout giúp mô hình trở nên chống lại overfitting bằng cách làm cho mỗi đơn vị phải học các đặc trưng một cách độc lập với các đơn vị khác. Điều này thúc đẩy các đơn vị học cách phản ứng với một loạt các ngữ cảnh khác nhau, thay vì chỉ phụ thuộc vào các đặc trưng cụ thể của một lớp hoặc một nhóm đơn vị.

Dropout thường được áp dụng sau các tầng fully connected (hoặc dense) trong mạng neural, nhưng cũng có thể

được sử dụng sau các tầng tích chập. Trong quá trình dự đoán (inference), khi mô hình không còn trong quá trình huấn luyện, không có dropout được áp dụng và tất cả các đơn vị được sử dụng.

4.2.6 Dàn phẳng (Flatten)

Phần dàn phẳng (Flatten) trong mạng nơ-ron tích chập (CNN) có nhiệm vụ chuyển đổi các feature map 2D thành một vector 1 chiều trước khi đưa vào các tầng kết nối đầy đủ. Đây là một bước cần thiết để biến đổi dữ liệu sao cho có thể thực hiện các phép toán đại số tuyến tính trên các nơ-ron đầu vào của tầng kết nối đầy đủ.

Trong ví dụ về mạng CNN đã đề cập ở phần Maxpooling, chúng ta đã có một feature map kích thước 2×2 sau khi áp dụng các tầng tích chập và Maxpooling. Dưới đây là phần dàn phẳng (Flatten) của feature map này:

$$\text{Feature map trước Flatten} = \begin{bmatrix} 5 & 6 \\ 6 & 7 \end{bmatrix}$$

Sau khi áp dụng phần Flatten, feature map trở thành một vector 1 chiều:

$$\text{Feature map sau Flatten} = [5, 6, 6, 7]$$

Vector này sau đó được đưa vào các tầng kết nối đầy đủ để thực hiện các phép toán tuyến tính. Cuối cùng, tầng softmax có thể được áp dụng để chuyển đổi đầu ra của mạng thành các xác suất ứng với từng lớp.

4.2.7 Tầng kết nối đầy đủ và Softmax

Tầng kết nối đầy đủ (fully connected layer) [5] thường được sử dụng để biến đổi các đặc trưng đã được trích xuất từ các tầng trước đó thành dự đoán cuối cùng. Đầu ra của tầng này thường được đưa qua hàm softmax để chuyển đổi thành các xác suất cho mỗi lớp.

Hàm Softmax:

Hàm softmax là một hàm kích hoạt được sử dụng phổ biến trong tầng đầu ra của các mạng neural để chuyển đổi các giá trị thành các xác suất. Công thức của hàm softmax cho một vector đầu vào z là:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Trong đó: - $\sigma(z)_i$ là phần tử thứ i của vector xác suất đầu ra. - z_i là phần tử thứ i của vector đầu vào z. - K là số lượng lớp.

Hàm softmax biến đầu vào thành một phân phối xác suất, với tổng của tất cả các phần tử trong vector xác suất bằng 1.

Áp dụng hàm Softmax:

Sau khi tính toán đầu ra từ tầng kết nối đầy đủ, chúng ta áp dụng hàm softmax để chuyển đổi thành các xác suất cho mỗi lớp. Điều này giúp mô hình của chúng ta tạo ra dự đoán chính xác về lớp của dữ liệu đầu vào.

Ví dụ:

Cho $z = [5, 6, 6, 7]$ ở phần trước là đầu ra sau khi áp dụng dàn phẳng (flatten) từ feature map được tạo ra từ tầng maxpooling trong một mô hình CNN. Sau đó, áp dụng hàm kích hoạt softmax, chúng ta có:

$$\sigma(z) = \left[\frac{e^5}{e^5 + e^6 + e^6 + e^7}, \frac{e^6}{e^5 + e^6 + e^6 + e^7}, \frac{e^6}{e^5 + e^6 + e^6 + e^7}, \frac{e^7}{e^5 + e^6 + e^6 + e^7} \right]$$

Đây là xác suất của mỗi lớp được dự đoán dựa trên đầu ra của tầng kết nối đầy đủ và được tính bằng hàm softmax.

4.2.8 Hàm mất mát

Trong quá trình feedforward của mạng neural, hàm mất mát đánh giá sự khác biệt giữa đầu ra dự đoán của mô hình và giá trị thực tế. Hàm mất mát thường được chọn dựa trên loại bài toán mà mạng neural đang giải quyết. Trong bài toán phân loại, hàm mất mát phổ biến là cross-entropy loss.

Cross-entropy loss được tính bằng cách so sánh phân phối xác suất của đầu ra dự đoán và phân phối xác suất của nhãn thực tế. Nó đo lường sự tương đồng giữa hai phân phối xác suất này, với mục tiêu là giảm thiểu khoảng cách giữa

chúng.

Xét hai phân phối xác suất: P là phân phối xác suất thực tế và Q là phân phối xác suất dự đoán. Khi sử dụng cross-entropy loss, chúng ta tính toán giá trị mất mát bằng công thức:

$$J(P,Q) = -\sum_i P(i)\log(Q(i))$$

Trong đó: - $P(i)$ là xác suất của lớp i trong nhãn thực tế. - $Q(i)$ là xác suất của lớp i trong đầu ra dự đoán.

Mục tiêu của quá trình huấn luyện là tối thiểu hóa giá trị của hàm mất mát này, tức là làm cho đầu ra dự đoán Q càng gần với nhãn thực tế P càng tốt. Điều này đồng nghĩa với việc cải thiện hiệu suất của mô hình trong việc phân loại dữ liệu.

4.3 Backpropagation

4.3.1 Tính toán giảm đạo hàm(Gradient Descent)

Trong thuật toán Gradient Descent, mục tiêu là điều chỉnh các tham số của mô hình để giảm thiểu giá trị của hàm mất mát $J(w)$. Để làm điều này, ta sử dụng đạo hàm của hàm mất mát theo từng tham số w, ký hiệu là $\frac{\partial J}{\partial w}$. Trong đó $J(w)$ là hàm mất mát (loss function) đánh giá hiệu suất của mô hình dựa trên dữ liệu huấn luyện, và w là tập hợp các trọng số mạng neural. Với mỗi bước lặp, trọng số w được cập nhật theo công thức sau:

$$w = w - \alpha\frac{\partial J}{\partial w}$$

Trong đó:

- α là learning rate, quyết định tốc độ học của thuật toán.
- $\frac{\partial J}{\partial w}$ là đạo hàm của hàm mất mát theo từng tham số w.

Quá trình này tiếp tục lặp lại cho đến khi đạt được điều kiện dừng hoặc số lần lặp đã đủ. Gradient Descent giúp di chuyển các tham số theo hướng giảm thiểu hàm mất mát, từ đó tối ưu hóa mô hình.

4.3.2 Tối ưu hóa trọng số

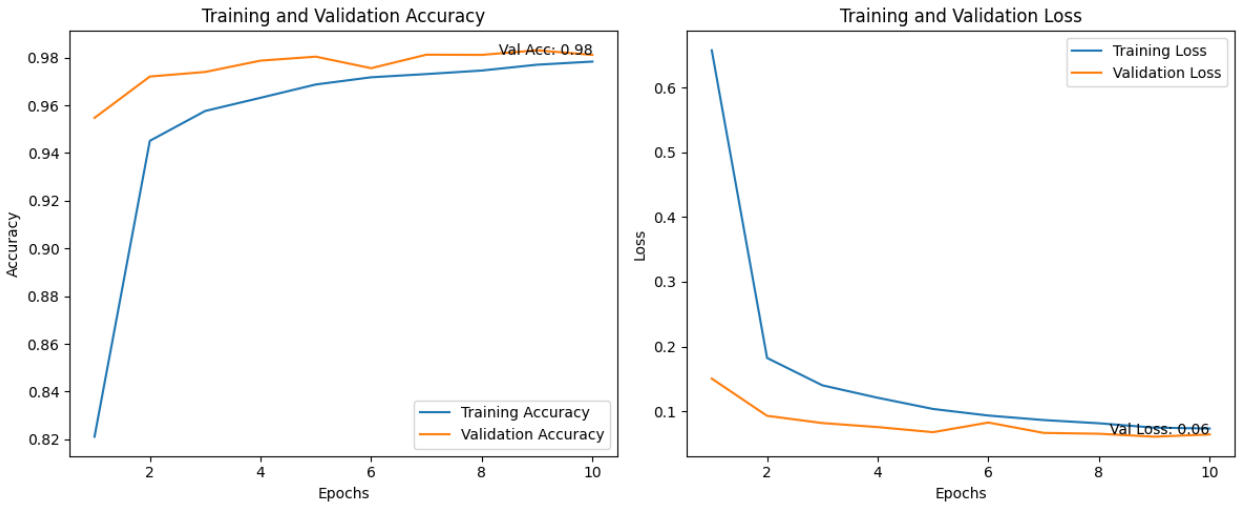
Trong quá trình huấn luyện mô hình CNN, một phần quan trọng là điều chỉnh các trọng số để giảm thiểu hàm mất mát và tăng hiệu suất của mô hình. Các thuật toán tối ưu hóa trọng số được sử dụng để thực hiện điều này, trong đó có Adam, một trong những phương pháp tối ưu hóa phổ biến trong lĩnh vực học máy.

Adam (Adaptive Moment Estimation) kết hợp cả momentum và RMSProp để cải thiện tốc độ học và hiệu suất của thuật toán Gradient Descent. Nó sử dụng hai giá trị momentum (vận tốc) và RMSProp (root mean square propagation) để cập nhật trọng số. Cụ thể, Adam tính toán một giá trị động của learning rate cho từng trọng số mạng neural, giúp điều chỉnh tốc độ học dựa trên gradient và lịch sử của gradient trong quá trình huấn luyện.

4.4 Kết quả và đánh giá mô hình

Sau quá trình huấn luyện mô hình, việc kiểm tra tính hiệu quả của mạng nơ-ron tích chập (CNN) đòi hỏi việc sử dụng biểu đồ và các độ đo hiệu suất đánh giá. Biểu đồ dưới đây mô tả sự thay đổi của độ chính xác và mất mát trong quá trình huấn luyện mô hình CNN cho thấy mô hình đang học hỏi và cải thiện hiệu suất. Tuy nhiên, độ chính xác và mất mát trong tập validation không đồng điệu với độ chính xác và mất mát trong tập training, cho thấy dấu hiệu overfitting. Cần điều chỉnh các siêu tham số và áp dụng các kỹ thuật chống overfitting để cải thiện hiệu suất mô hình trên dữ liệu validation và tập dữ liệu test độc lập.

Tiếp đố là bảng tổng kết các chỉ số đo từ mô hình bao gồm: Epochs, Accuracy, Precision, Recall và Loss.



Hình 4.2: Biểu đồ minh họa sự biến đổi của độ chính xác và mất mát trong quá trình huấn luyện của mạng khi số lượng epochs tăng

Epoch	Accuracy	Precision	Recall	Loss
1	0.6830	0.99	0.99	1.7628
2	0.9344	0.99	0.99	0.2110
3	0.9563	0.96	0.99	0.1444
4	0.9631	0.99	0.97	0.1207
5	0.9659	0.99	0.98	0.1107
6	0.9707	0.99	0.96	0.0965
7	0.9721	0.99	0.99	0.0902
8	0.9761	0.99	0.98	0.0776
9	0.9759	0.98	0.97	0.0754
10	0.9775	0.95	0.99	0.0721

Bảng 4.1: Bảng biểu mô tả chỉ số đánh giá mô hình

- "Epochs"là số thứ tự của epoch.
- "Accuracy"là tỷ lệ dự đoán chính xác trên tập huấn luyện.
- "Precision"là tỷ lệ các dự đoán dương tính đúng so với tất cả các dự đoán dương tính.
- "Recall"là tỷ lệ các dự đoán dương tính đúng so với tất cả các thực thể dương tính.
- "Loss"là giá trị hàm mất mát trên tập huấn luyện, một thước đo của sự không chính xác trong dự đoán của mô hình.

Đồng thời ta thu được ma trận lỗi sau:

$$\begin{bmatrix} 1162 & 0 & 2 & 1 & 0 & 0 & 2 & 0 & 1 & 7 \\ 0 & 1315 & 3 & 0 & 0 & 0 & 1 & 2 & 0 & 1 \\ 0 & 4 & 1158 & 2 & 0 & 0 & 0 & 4 & 6 & 0 \\ 0 & 0 & 16 & 1184 & 0 & 9 & 0 & 4 & 4 & 2 \\ 3 & 0 & 0 & 0 & 1154 & 0 & 0 & 0 & 2 & 17 \\ 1 & 3 & 3 & 3 & 0 & 1059 & 5 & 2 & 11 & 17 \\ 5 & 2 & 0 & 0 & 2 & 0 & 1165 & 0 & 3 & 0 \\ 1 & 2 & 21 & 0 & 1 & 0 & 0 & 1267 & 0 & 7 \\ 1 & 0 & 9 & 0 & 5 & 2 & 0 & 2 & 1130 & 11 \\ 3 & 1 & 0 & 1 & 5 & 1 & 1 & 1 & 1 & 1180 \end{bmatrix}$$

Đặc điểm của ma trận lỗi (Confusion Matrix):

- Mỗi hàng trong ma trận đại diện cho nhãn thực tế của dữ liệu. Ví dụ, hàng đầu tiên biểu thị cho nhãn "0", hàng thứ hai biểu thị cho nhãn "1", và tiếp tục cho đến hàng cuối cùng biểu thị cho nhãn "9".
- Mỗi cột trong ma trận đại diện cho nhãn được dự đoán bởi mô hình. Tương tự như hàng, cột đầu tiên biểu thị cho nhãn "0", cột thứ hai biểu thị cho nhãn "1", và tiếp tục cho đến cột cuối cùng biểu thị cho nhãn "9".
- Độ chính xác toàn cục (Accuracy): Độ chính xác tổng thể của mô hình có thể được tính bằng tổng các phần tử trên đường chéo chính (các dự đoán đúng) chia cho tổng số lượng điểm dữ liệu. Trong trường hợp này, tổng số lượng dự đoán đúng là (1162 + 1315 + 1158 + 1184 + 1154 + 1059 + 1165 + 1267 + 1130 + 1180) = 11494. Tổng số lượng điểm dữ liệu là tổng của tất cả các phần tử trong ma trận lỗi, tức là 12000. Vì vậy, độ chính xác toàn cục là 11494/12000 = 0.9578.
- Lỗi phân loại: Các ô nằm ngoài đường chéo chính của ma trận lỗi thể hiện số lượng dự đoán sai cho mỗi cặp chữ số. Ví dụ, có 17 điểm dữ liệu của chữ số "4"đã bị mô hình dự đoán thành chữ số "9". Bằng cách này, ta có thể xác định được các cặp chữ số mà mô hình thường nhầm lẫn với nhau.

Chương 5

Kết Luận

Trong báo cáo này, chúng tôi đã tiến hành xây dựng mô hình nhận diện chữ viết tay sử dụng các kỹ thuật máy học và thị giác máy tính. Qua quá trình nghiên cứu và thực nghiệm, chúng tôi đã đạt được những kết quả quan trọng sau:

- Giảm chiều dữ liệu bằng phương pháp PCA: Chúng tôi đã mô tả tập dữ liệu ảnh chữ số viết tay MNIST và áp dụng phương pháp PCA để giảm chiều dữ liệu. Kết quả thực nghiệm đã cho thấy hiệu quả của việc giảm chiều dữ liệu trong việc xử lý hình ảnh chữ số viết tay.
- Thuật toán phân cụm: Chúng tôi đã áp dụng thuật toán K-Means để phân cụm hình ảnh chữ số viết tay dựa trên đặc trưng hình ảnh. Kết quả đã cho thấy khả năng phân loại chính xác của thuật toán trong việc nhóm các hình ảnh vào các cụm tương ứng.
- Xây dựng mô hình CNN phân loại hình ảnh: Chúng tôi đã giới thiệu mô hình Mạng nơ-ron tích chập và áp dụng nó để phân loại hình ảnh chữ số viết tay. Qua quá trình huấn luyện và đánh giá, mô hình đã đạt được độ chính xác cao và khả năng nhận diện chữ số viết tay tốt.

Tổng kết, nghiên cứu này đã đưa ra những kết quả tích cực trong việc áp dụng các kỹ thuật học máy vào việc nhận diện chữ số viết tay. Các phương pháp và mô hình đã được xây dựng và đánh giá có thể được áp dụng trong các ứng dụng thực tế khác nhau, đồng thời tạo ra cơ sở cho các nghiên cứu và phát triển tiếp theo trong lĩnh vực này.

Chương 6

Phân công nhiệm vụ

Nhiệm vụ	Phân công	Tiến độ
Tìm hiểu về phương pháp giảm chiều dữ liệu (Ý 1)	Nguyễn Tiến Đạt	Đã xong
Tìm hiểu về thuật toán phân cụm (Ý 2)	Đinh Mạnh Cường	Đã xong
Tìm hiểu về mô hình CNN (Ý 4)	Nguyễn Quang Huy	Đã xong
Viết báo cáo về phương pháp PCA giảm chiều dữ liệu	Nguyễn Tiến Đạt	Đã xong
Viết báo cáo về thuật toán phân cụm K-mean	Đinh Mạnh Cường	Đã xong
Viết báo cáo về mô hình CNN	Nguyễn Quang Huy	Đã xong
Làm slides về phương pháp PCA giảm chiều dữ liệu	Nguyễn Tiến Đạt	Đã xong
Làm slides về thuật toán phân cụm K-mean	Đinh Mạnh Cường	Đã xong
Làm slides về mô hình CNN	Nguyễn Quang Huy	Đã xong

Bảng 6.1: Nhiệm vụ của từng thành viên trong nhóm

Tài liệu tham khảo

[1] Bishop, Christopher M. “Pattern recognition and Machine Learning.”, Springer (2006). Chapter 12: Continues Latent Variables. 559-569.

[2] Ian T. Jolliffe and Jorge Cadima. (2016, 13 April). "Principal component analysis: a review and recent developments".

[3] Ng, A. (2020). "Lecture Notes for Topic: Clustering". In CS229 Machine Learning Course. Stanford University (Book)

[4] Bishop, Christopher M. "Pattern Recognition and Machine Learning.", Springer (2006). Chapter 9: Mixture Models and EM. Section 9.1: K-means Clustering. (Book).

[5] Jain, M., Kaur, G., Quamar, M. P., and Gupta, H. (2021, February 17-19). "Handwritten digit recognition using CNN". In 2021 International *Conference on Innovative Practices in Technology and Management (ICIPTM)* (pp. 3-4). IEEE.

[6] Hossain, Md. A., and Ali, Md. M. (2019). "Recognition of handwritten digit using convolutional neural network (CNN)". *Global Journal of Computer Science and Technology: D Neural and Artificial Intelligence, 19(2)*, 1-10.

[7] Nguyen, T. D., and Vo, B. N. (2021). "Handwritten Vietnamese character recognition using convolutional neural networks".*Pattern Recognition Letters*, 153, 102-111.