Lenet Build From Scratch

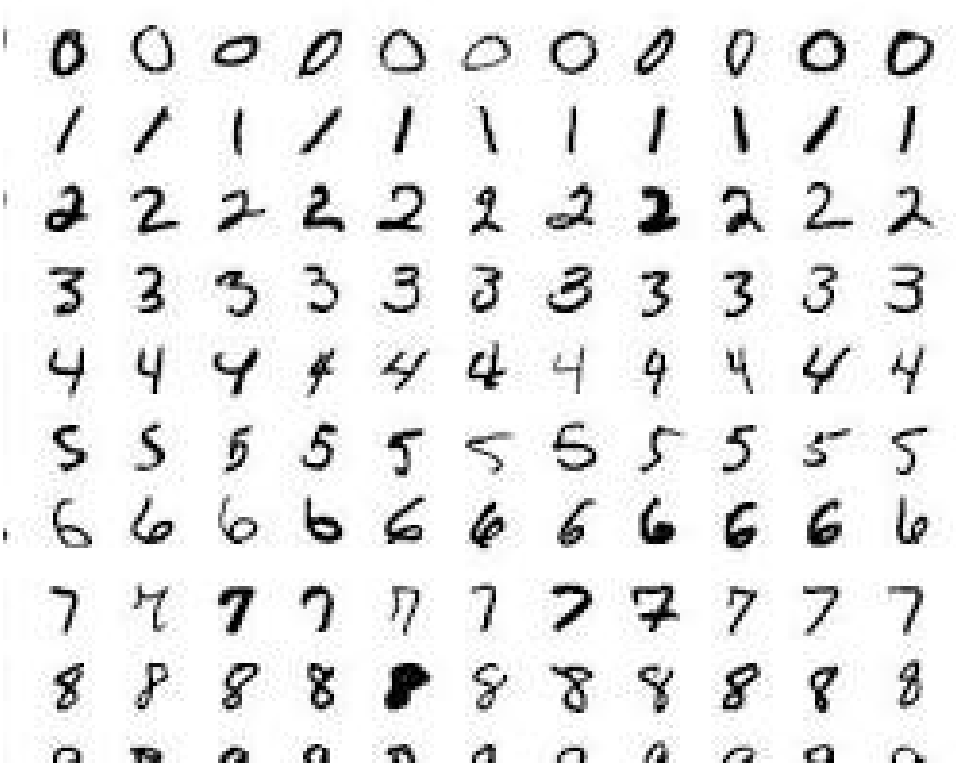
Lê Phạm Hoàng Trung

02/01/2025

1 Tập dữ liệu MNIST

1.1 Giới thiệu

MNIST (Modified National Institute of Standards and Technology database) là một tập dữ liệu hình ảnh kinh điển, đóng vai trò quan trọng trong lĩnh vực học máy và thị giác máy tính. Tập dữ liệu này chứa 70.000 hình ảnh chữ số viết tay, được chia thành 60.000 ảnh cho tập huấn luyện và 10.000 ảnh cho tập kiểm tra. Mỗi hình ảnh là ảnh xám kích thước 28x28 pixel, biểu diễn một chữ số từ 0 đến 9 và được gán nhãn sẵn với chữ số tương ứng.



Hình 1: Tập dữ liệu MNIST

Tính đơn giản và hiệu quả của MNIST khiến nó trở thành lựa chọn phổ biến cho việc huấn luyện và đánh giá các mô hình học máy, đặc biệt là các mạng nơ-ron. Tập dữ liệu này cung cấp một môi trường lý tưởng để nghiên cứu và giảng dạy về các thuật toán nhận dạng chữ số viết tay. Hơn nữa, MNIST thường được sử dụng như một benchmark để so sánh hiệu quả của các thuật toán học máy khác nhau.

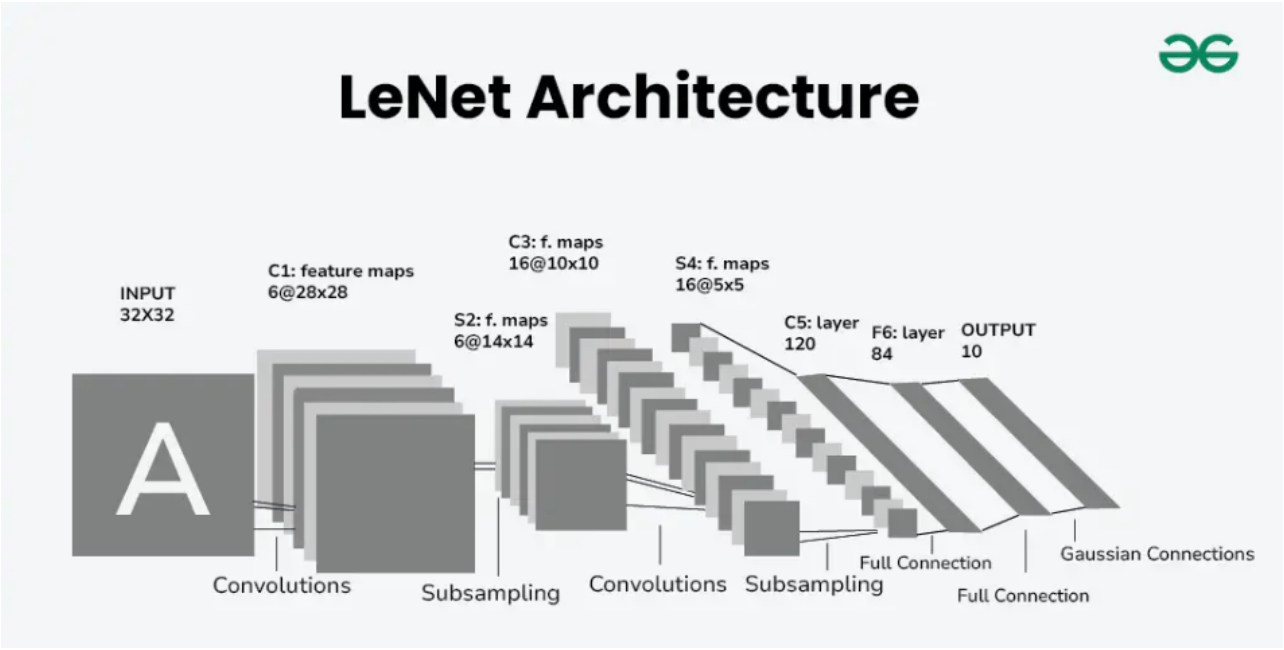
1.2 Cách tải tập dữ liệu MNIST

Sử dụng đoạn code được cung cấp bên dưới, dùng thư viện PyTorch tải tập dữ liệu MNIST về:

```
import torchvision
import torch
import torch.nn as nn
import torchvision.transforms as transforms

transform = transforms.ToTensor()

# MNIST - Train data
trainset = torchvision.datasets.MNIST(root='./data', train=True,
                                      download=True, transform=transform)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=32,
                                          shuffle=True, num_workers=2)

# MNIST - Test data
testset = torchvision.datasets.MNIST(root='./data', train=False,
                                     download=True, transform=transform)
testloader = torch.utils.data.DataLoader(testset, batch_size=32,
                                         shuffle=False, num_workers=2)

# Classes
classes = ('0', '1', '2', '3', '4', '5', '6', '7', '8', '9')
```

2 Mô hình LeNet-5

LeNet-5 là một mạng nơ-ron convolutional (CNN) tiên phong được đề xuất bởi Yann LeCun và cộng sự vào năm 1998. Đây là một trong những mô hình CNN đầu tiên và có ảnh hưởng lớn đến sự phát triển của lĩnh vực học sâu. LeNet-5 được thiết kế ban đầu để nhận dạng chữ số viết tay trong hình ảnh với tập dữ liệu MNIST.



Hình 2: Kiến trúc mô hình LeNet-5

LeNet-5 bao gồm các lớp sau:

- Lớp convolutional (C1): Sử dụng 6 kernel kích thước 5x5 với stride là 1, tạo ra 6 feature map kích thước 28x28.
- Lớp pooling (S2): Sử dụng average pooling với kernel 2x2 và stride là 2, giảm kích thước feature map xuống 14x14.
- Lớp convolutional (C3): Sử dụng 16 kernel kích thước 5x5 với stride là 1, tạo ra 16 feature map kích thước 10x10.
- Lớp pooling (S4): Sử dụng average pooling với kernel 2x2 và stride là 2, giảm kích thước feature map xuống 5x5.
- Lớp fully connected (C5): Kết nối 120 nơ-ron với tất cả các nơ-ron từ lớp trước.
- Lớp fully connected (F6): Kết nối 84 nơ-ron với lớp C5.
- Lớp đầu ra: Kết nối 10 nơ-ron (tương ứng với 10 chữ số) với lớp F6, sử dụng hàm softmax để phân loại.

3 Yêu cầu

Ở phần này, bạn sẽ đóng vai trò giống như Yann Lecun, tạo ra mô hình LeNet-5 "from scratch", và áp dụng nó trên tập dữ liệu MNIST ở trên để đánh giá mô hình. Cụ thể, các bạn sẽ thực hiện các yêu cầu bên dưới

- Tải tập dữ liệu MNIST (sử dụng đoạn code đã cung cấp ở phần giới thiệu dữ liệu)

- Xây dựng mô hình LeNet-5 theo kiến trúc đã được mô tả, sử dụng PyTorch hoặc Tensorflow đều được (khuyến khích PyTorch). Hàm kích hoạt giữa các layer nên là hàm phi tuyến (ví dụ hàm ReLU)
- Đánh giá trên tập dữ liệu MNIST (nên cải thiện mô hình sao cho accuracy $\geq 95\%$)
- **Bonus:** Thử sáng tạo mô hình theo ý bạn, có thể áp dụng các kỹ thuật thêm như **Early Stopping, Image Processing, ...** để tăng độ chính xác.

4 Gợi ý

4.1 Xây dựng mô hình

```
class Lenet5(nn.Module):
    def __init__(self):
        super(Lenet5, self).__init__()
        # Create the first Conv2d layer with input channels = 1, output channels = 6,
        # kernel size = 5, stride = 1, and padding = 2 to maintain the input size.
        self.conv1 = nn.Conv2d(1, 6, kernel_size=5, stride=1, padding=2)
        self.relu1 = nn.ReLU()  # Add ReLU activation function after the first convolution.
        self.pool1 = nn.AvgPool2d(kernel_size=2, stride=2)  # Apply AvgPool to reduce spatial dimensions by half

        # Do more to finish ...

    def forward(self, x):
        # Pass the input through the first Conv2d layer, ReLU, and AvgPool.
        x = self.conv1(x)  # Apply the first convolution.
        x = self.relu1(x)  # Apply ReLU activation.
        x = self.pool1(x)  # Apply AvgPool to reduce dimensions.

        # Do more to finish ...
```

4.2 Xây dựng hàm huấn luyện

```
def train_and_evaluate(model, trainloader, testloader, criterion, optimizer, num_epochs=2):
    for epoch in range(num_epochs):
        model.train()
        for i, data in enumerate(trainloader):
            inputs, labels = data
            optimizer.zero_grad()
            outputs = model(inputs)
            loss = criterion(outputs, labels)
            loss.backward()
```

```
            optimizer.step()
            if i % 2000 == 1999:
                print(f"Epoch {epoch+1}, batch {i+1}, loss: {loss.item()}")

        model.eval()
        correct = 0
        total = 0
        with torch.no_grad():
            for data in testloader:
                images, labels = data
                outputs = model(images)
                _, predicted = torch.max(outputs.data, 1)
                total += labels.size(0)
                correct += (predicted == labels).sum().item()
        accuracy = 100 * correct / total
        print(f"Epoch {epoch+1}, accuracy: {accuracy:.2f}%")

criteria = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(trunk.parameters(), lr=0.01)

# Pass parameters to run :DDD
train_and_evaluate(trunk, trainloader, testloader, criteria, optimizer,
    num_epochs=30)
```

4.3 Đầu ra của mô hình

Ở đây, mình khuyến khích các bạn nên tự custom và tinh chỉnh mô hình, ví dụ như thay đổi hàm kích hoạt, thay đổi layer pool. Quan trọng, các bạn có thể để số units (tức input channels, output channels) tuỳ ý, miễn giữ nguyên units ở đầu vào và đầu ra cho hợp lệ là được. Tuy nhiên, nên đảm bảo accuracy lớn hơn 95%.