```
from torchvision.datasets import MNIST
from torchvision import transforms
import torch

transform = transforms.Compose([transforms.ToTensor(), transforms.Normalize((0.5,), (0.5,))])

train_dataset = MNIST(root="./data", train=True, transform=transform, download=True)
test_dataset = MNIST(root="./data", train=False, transform=transform, download=True)
```

```
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz
Failed to download (trying next):
<urlopen error [Errno 11002] getaddrinfo failed>

Downloading https://ossci-datasets.s3.amazonaws.com/mnist/train-images-idx3-ubyte.gz
Downloading https://ossci-datasets.s3.amazonaws.com/mnist/train-images-idx3-ubyte.gz to ./data\MNIST\raw\train-images-idx3-ubyte.gz
100.0%
Extracting ./data\MNIST\raw\train-images-idx3-ubyte.gz to ./data\MNIST\raw

Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz
Failed to download (trying next):
<urlopen error [Errno 11002] getaddrinfo failed>

Downloading https://ossci-datasets.s3.amazonaws.com/mnist/train-labels-idx1-ubyte.gz
Downloading https://ossci-datasets.s3.amazonaws.com/mnist/train-labels-idx1-ubyte.gz to ./data\MNIST\raw\train-labels-idx1-ubyte.gz
100.0%
Extracting ./data\MNIST\raw\train-labels-idx1-ubyte.gz to ./data\MNIST\raw

Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz
Failed to download (trying next):
<urlopen error [Errno 11002] getaddrinfo failed>

Downloading https://ossci-datasets.s3.amazonaws.com/mnist/t10k-images-idx3-ubyte.gz
Downloading https://ossci-datasets.s3.amazonaws.com/mnist/t10k-images-idx3-ubyte.gz to ./data\MNIST\raw\t10k-images-idx3-ubyte.gz
100.0%
Extracting ./data\MNIST\raw\t10k-images-idx3-ubyte.gz to ./data\MNIST\raw

Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz
100.0%Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz to ./data\MNIST\raw\t10k-labels-idx1-ubyte.gz
Extracting ./data\MNIST\raw\t10k-labels-idx1-ubyte.gz to ./data\MNIST\raw
```

```
from torch.utils.data import DataLoader

train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=64, shuffle=False)
```

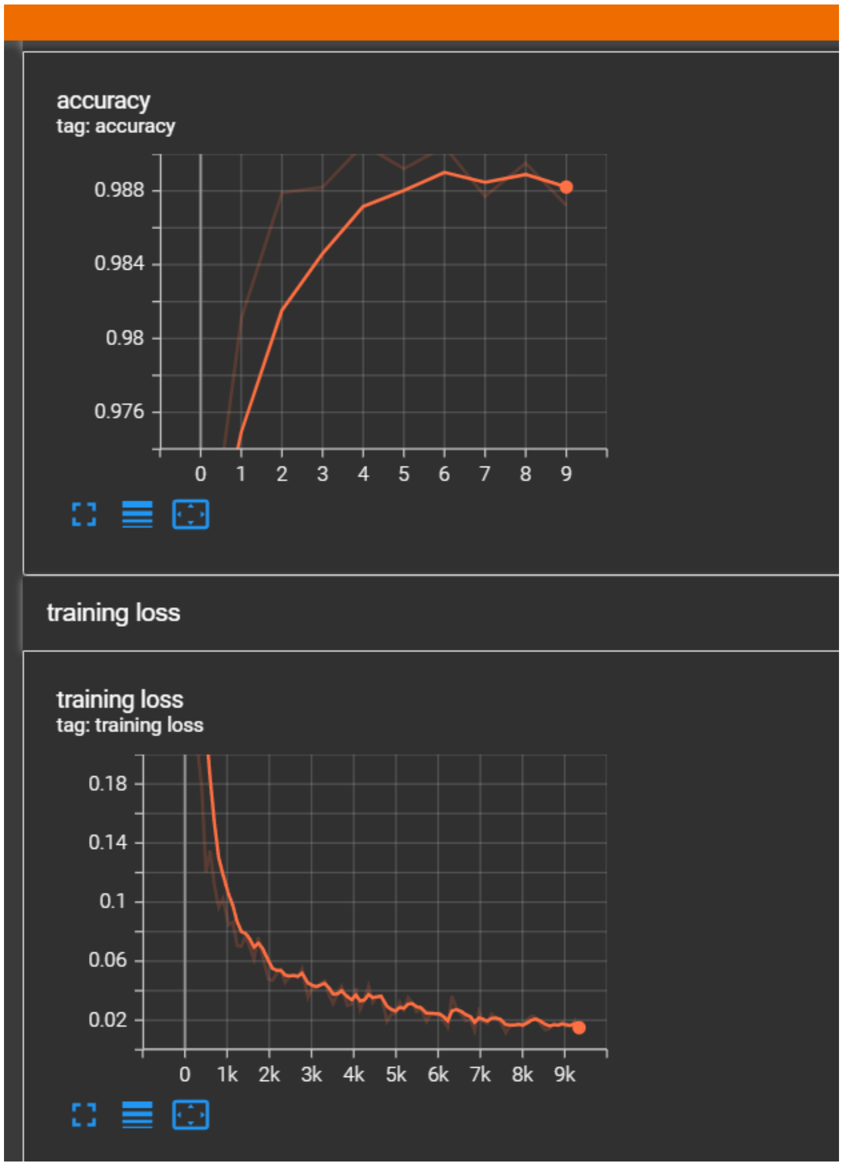
```
import torch.nn as nn

class LeNet5(nn.Module):
    def __init__(self):
        super(LeNet5, self).__init__()

        self.conv1 = nn.Conv2d(1, 6, kernel_size=5)
        self.conv2 = nn.Conv2d(6, 16, kernel_size=5)
        self.fc1 = nn.Linear(16*4*4, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = F.relu(F.max_pool2d(self.conv1(x), 2))
        x = F.relu(F.max_pool2d(self.conv2(x), 2))
        x = x.view(-1, 16*4*4)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return F.log_softmax(x, dim=1)
```

```
from torch.utils.tensorboard import SummaryWriter

# Crea un directorio de resumen para TensorBoard (p.ej., 'runs/lenet5_experiment_1')
writer = SummaryWriter('runs/lenet5_experiment_2')
```

```
import torch.nn.functional as F
import torch.optim as optim
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = LeNet5()
model.to(device)
# Añadir el modelo a TensorBoard
images, labels = next(iter(train_loader))
writer.add_graph(model, images.to(device))


criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)



num_epochs = 10
for epoch in range(num_epochs):
    model.train()
    running_loss = 0.0
    for i, (data, target) in enumerate(train_loader):
        data, target = data.to(device), target.to(device)

        optimizer.zero_grad()
        output = model(data)
        loss = criterion(output, target)
        loss.backward()
        optimizer.step()

        running_loss += loss.item()
        if i % 100 == 99:  # Registrar cada 100 lotes
            writer.add_scalar('training loss', running_loss / 100, epoch * len(train_loader) + i)
            running_loss = 0.0

    # Evaluación
    model.eval()
    total_correct = 0
    TP = 0
    TN = 0
    FP = 0
    FN = 0

    with torch.no_grad():
        for data, target in test_loader:
            data, target = data.to(device), target.to(device)
            output = model(data)
            _, predicted = torch.max(output.data, 1)
            total_correct += (predicted == target).sum().item()

            # Calculo para clasificación binaria
            TP += ((predicted == 1) & (target == 1)).sum().item()
            TN += ((predicted == 0) & (target == 0)).sum().item()
            FP += ((predicted == 1) & (target == 0)).sum().item()
            FN += ((predicted == 0) & (target == 1)).sum().item()

    accuracy = total_correct / len(test_dataset)
    precision = TP / (TP + FP) if TP + FP != 0 else 0
    recall = TP / (TP + FN) if TP + FN != 0 else 0
    f1 = 2 * (precision * recall) / (precision + recall) if precision + recall != 0 else 0

    writer.add_scalar('accuracy', accuracy, epoch)
    writer.add_scalar('precision', precision, epoch)
    writer.add_scalar('recall', recall, epoch)
    writer.add_scalar('f1', f1, epoch)

    print(f"Epoch {epoch+1}/{num_epochs}, Accuracy: {accuracy:.4f}, Precision: {precision:.4f}, Recall: {recall:.4f}, F1-score: {f1:.4f}")
```

```
Epoch 1/10, Accuracy: 0.9716, Precision: 1.0000, Recall: 1.0000, F1-score: 1.0000
Epoch 2/10, Accuracy: 0.9833, Precision: 0.9991, Recall: 1.0000, F1-score: 0.9996
Epoch 3/10, Accuracy: 0.9873, Precision: 1.0000, Recall: 1.0000, F1-score: 1.0000
Epoch 4/10, Accuracy: 0.9889, Precision: 0.9991, Recall: 1.0000, F1-score: 0.9996
Epoch 5/10, Accuracy: 0.9893, Precision: 0.9991, Recall: 1.0000, F1-score: 0.9996
Epoch 6/10, Accuracy: 0.9914, Precision: 1.0000, Recall: 1.0000, F1-score: 1.0000
Epoch 7/10, Accuracy: 0.9894, Precision: 1.0000, Recall: 1.0000, F1-score: 1.0000
Epoch 8/10, Accuracy: 0.9910, Precision: 0.9991, Recall: 1.0000, F1-score: 0.9996
Epoch 9/10, Accuracy: 0.9884, Precision: 0.9991, Recall: 1.0000, F1-score: 0.9996
Epoch 10/10, Accuracy: 0.9891, Precision: 1.0000, Recall: 1.0000, F1-score: 1.0000
```

LeNet5 Learning


accuracy
tag: accuracy
0.988
0.984
0.98
0.976
0
1
2
3
4
5
6
7
8
9
training loss
training loss
tag: training loss
0.18
0.14
0.1
0.06
0.02
0
1k
2k
3k
4k
5k
6k
7k
8k
9k