from tqdm import tqdm # ✅ הוספת progress bar

import torch

import torch.nn as nn

import torch.optim as optim

import torchvision

import torchvision.transforms as transforms

import matplotlib.pyplot as plt

# בדיקה אם GPU זמין

device = torch.device("cuda" if torch.cuda.is\_available() else "cpu")

print(f"Using device: {device}")

# 🔹 1️⃣ טוענים את הנתונים של MNIST

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

train\_dataset = torchvision.datasets.MNIST(root="./data", train=True, download=True, transform=transform)

test\_dataset = torchvision.datasets.MNIST(root="./data", train=False, download=True, transform=transform)

train\_loader = torch.utils.data.DataLoader(train\_dataset, batch\_size=64, shuffle=True)

test\_loader = torch.utils.data.DataLoader(test\_dataset, batch\_size=64, shuffle=False)

# 🔹 2️⃣ מגדירים את מודל הרשת הנוירונית

class NeuralNet(nn.Module):

def \_\_init\_\_(self):

super(NeuralNet, self).\_\_init\_\_()

self.fc1 = nn.Linear(28 \* 28, 128) # שכבה ראשונה

self.relu = nn.ReLU() # פונקציית הפעלה

self.fc2 = nn.Linear(128, 10) # שכבת פלט (10 ספרות)

def forward(self, x):

x = x.view(-1, 28 \* 28) # הפיכת תמונה לווקטור

x = self.fc1(x)

x = self.relu(x)

x = self.fc2(x)

return x

# יוצרים את המודל ומעבירים ל-GPU אם אפשר

model = NeuralNet().to(device)

# 🔹 3️⃣ מגדירים פונקציית הפסד ואופטימיזציה

criterion = nn.CrossEntropyLoss()

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

# 🔹 4️⃣ מאמנים את המודל עם Progress Bar

num\_epochs = 5

for epoch in range(num\_epochs):

model.train()

total\_loss = 0

progress\_bar = tqdm(train\_loader, desc=f"Epoch {epoch+1}/{num\_epochs}", leave=True)

for images, labels in progress\_bar:

images, labels = images.to(device), labels.to(device)

optimizer.zero\_grad()

outputs = model(images)

loss = criterion(outputs, labels)

loss.backward()

optimizer.step()

total\_loss += loss.item()

progress\_bar.set\_postfix(loss=total\_loss / len(train\_loader))

print(f"Epoch [{epoch+1}/{num\_epochs}], Loss: {total\_loss/len(train\_loader):.4f}")

# 🔹 5️⃣ בודקים את הדיוק על סט הבדיקה

model.eval()

correct = 0

total = 0

with torch.no\_grad():

for images, labels in tqdm(test\_loader, desc="Evaluating"):

images, labels = images.to(device), labels.to(device)

outputs = model(images)

\_, predicted = torch.max(outputs, 1)

total += labels.size(0)

correct += (predicted == labels).sum().item()

accuracy = 100 \* correct / total

## print(f"Accuracy on test set: {accuracy:.2f}%") 🎯 הסבר על החלקים המרכזיים בקוד

1. **הורדת הנתונים של MNIST**
   * טוען את הנתונים מהאינטרנט ומעביר אותם לפורמט Tensor.
   * מבצע **נורמליזציה** לערכים בטווח [-1,1] כדי להאיץ את האימון.
2. **בניית רשת הנוירונים**
   * שכבת קלט של 28x28 = 784 נוירונים.
   * שכבה נסתרת (Hidden Layer) עם 128 נוירונים ו-ReLU.
   * שכבת פלט עם 10 נוירונים (כי יש 10 ספרות 0-9).
3. **אימון הרשת**
   * משתמש ב-**Adam Optimizer** ו-**CrossEntropyLoss**.
   * עובר על כל הסט **5 אפוקים** (epochs).
4. **בדיקה של המודל**
   * מחשב את **רמת הדיוק (%)** על סט הבדיקה.
   * מציג כמה דוגמאות מנובאות על המסך.