📌 שלב 1: טעינת הנתונים

כאן אנו טוענים את קובצי ה-MNIST הבינאריים (idx-ubyte.gz) ישירות מ-./mnist\_data באמצעות PyTorch.

python

Copy

Edit

import torch

import torchvision

import torchvision.transforms as transforms

# הגדרת ההמרות לנתונים (נורמליזציה והמרה לטנזורים)

transform = transforms.Compose([

transforms.ToTensor(), # המרת התמונות לטנזורים של PyTorch

transforms.Normalize((0.5,), (0.5,)) # נורמליזציה בין -1 ל-1 לשיפור האימון

])

# טעינת הנתונים מקובצי MNIST המקומיים (ללא הורדה מהאינטרנט)

trainset = torchvision.datasets.MNIST(root="./mnist\_data", train=True, transform=transform, download=False)

testset = torchvision.datasets.MNIST(root="./mnist\_data", train=False, transform=transform, download=False)

# יצירת DataLoader לקריאה נוחה במנות (batches)

trainloader = torch.utils.data.DataLoader(trainset, batch\_size=64, shuffle=True)

testloader = torch.utils.data.DataLoader(testset, batch\_size=64, shuffle=False)

print("✅ הנתונים נטענו בהצלחה!")

📌 שלב 2: הצגת דוגמה מהנתונים

נראה תמונה אחת מתוך סט האימון כדי לוודא שהנתונים נטענו נכון.

python

Copy

Edit

import matplotlib.pyplot as plt

import numpy as np

# לקיחת תמונה ודוגמית מהדאטה-לואדר

dataiter = iter(trainloader)

images, labels = next(dataiter)

# הצגת התמונה הראשונה

plt.imshow(images[0].squeeze(), cmap="gray")

plt.title(f"Label: {labels[0].item()}")

plt.show()

📌 שלב 3: הגדרת רשת נוירונים

נבנה רשת נוירונים פשוטה עם 3 שכבות צפופות (Fully Connected).

python

Copy

Edit

import torch.nn as nn

import torch.optim as optim

# הגדרת רשת נוירונים פשוטה

class SimpleNN(nn.Module):

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

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

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

self.fc2 = nn.Linear(128, 64) # שכבה שנייה

self.fc3 = nn.Linear(64, 10) # שכבה שלישית (10 קטגוריות - ספרות 0-9)

def forward(self, x):

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

x = torch.relu(self.fc1(x)) # פונקציית הפעלה ReLU

x = torch.relu(self.fc2(x))

x = self.fc3(x) # השכבה הסופית ללא ReLU כי CrossEntropyLoss עושה זאת אוטומטית

return x

# יצירת המודל

model = SimpleNN()

📌 שלב 4: הגדרת פונקציית הפסד ואופטימיזציה

נשתמש ב-CrossEntropyLoss (מתאים לבעיות סיווג) ובאופטימיזציה SGD.

python

Copy

Edit

# פונקציית הפסד (Cross Entropy מתאימה לסיווג של 10 מחלקות)

criterion = nn.CrossEntropyLoss()

# אופטימיזציה עם Stochastic Gradient Descent

optimizer = optim.SGD(model.parameters(), lr=0.01, momentum=0.9)

📌 שלב 5: אימון הרשת

נאמן את הרשת למשך 5 אפוקים (סבבים על כל הנתונים).

python

Copy

Edit

epochs = 5 # מספר סבבים

for epoch in range(epochs):

running\_loss = 0.0

for images, labels in trainloader:

optimizer.zero\_grad() # איפוס הגרדיאנטים

outputs = model(images) # הרצת המודל קדימה

loss = criterion(outputs, labels) # חישוב הפסד

loss.backward() # חישוב גרדיאנטים

optimizer.step() # עדכון המשקלים

running\_loss += loss.item()

print(f"Epoch {epoch+1}, Loss: {running\_loss / len(trainloader):.4f}")

print("✅ סיימנו לאמן את הרשת!")

📌 שלב 6: בדיקת הדיוק של המודל

נבדוק את אחוז הדיוק של הרשת על סט הבדיקה.

python

Copy

Edit

correct = 0

total = 0

with torch.no\_grad(): # אין צורך לשמור גרדיאנטים בבדיקה

for images, labels in testloader:

outputs = model(images)

\_, predicted = torch.max(outputs, 1) # מציאת התחזית עם ההסתברות הגבוהה ביותר

total += labels.size(0)

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

accuracy = 100 \* correct / total

print(f"🔵 דיוק המודל על סט הבדיקה: {accuracy:.2f}%")

📌 שלב 7: בדיקת תחזיות על תמונות בודדות

נראה כמה תחזיות שהמודל נותן:

python

Copy

Edit

# קבלת תמונות לדוגמה

dataiter = iter(testloader)

images, labels = next(dataiter)

# קבלת התחזיות מהמודל

outputs = model(images)

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

# הצגת 5 תמונות ותחזיותיהן

fig, axes = plt.subplots(1, 5, figsize=(10, 3))

for i in range(5):

axes[i].imshow(images[i].squeeze(), cmap="gray")

axes[i].set\_title(f"Pred: {predicted[i].item()}")

axes[i].axis("off")

plt.show()

📌 סיכום 🚀

🔹 שלב 1: טענו את הנתונים מקבצים בינאריים (idx-ubyte.gz) באמצעות torchvision.datasets.MNIST.

🔹 שלב 2: הצגנו תמונה אחת מהנתונים לוודא שהכל תקין.

🔹 שלב 3: בנינו רשת נוירונים בסיסית עם 3 שכבות.

🔹 שלב 4: הגדרנו פונקציית הפסד ואופטימיזציה (SGD).

🔹 שלב 5: אימנו את הרשת על MNIST ל-5 אפוקים.

🔹 שלב 6: בדקנו את הדיוק של המודל על סט הבדיקה.

🔹 שלב 7: הצגנו תחזיות על תמונות אמיתיות מסט הבדיקה.

✅ עכשיו יש לך מודל שלם לסיווג ספרות!