13. PyTorch

PyTorch הינה ספרייה ייעודית עבור למידה עמוקה, המאפשרת עבודה נוחה עם דאטה, בנייה פשוטה של מודלים והתממשקות קלה למכונות GPUs. הספרייה משתמשת במבני נתונים הנקראים טנזורים (torch.tensor), שהם למעשה מערכים רב ממדיים. הטנזורים דומים במהות ובסינטקס ל-NumPy array, אלא שהטנזורים יכולים לרוץ על גבי GPUs וכך לבצע אימונים בצורה מהירה. בפרק זה נעבור על החלקים השונים שיש בכל מודל של למידה עמוקה: טעינת דאטה, בניית ארכיטקטורה, הגדרת שיטת האופטימיזציה, אימון וטסט ושמירה וטעינה של מודלים. כפרק מבוא, נעבור על אופן העבודה עם משתנים מסוג torch.tensor.

13.1 Tensors and DataLoaders

13.1.1 Tensors in PyTorch

אתחול של טנזורים יכול להיעשות בשתי דרכים עיקריות – על ידי הגדרת משתנה קיים כטנזור או על ידי אתחול מפורש של משתנה מסוג טנזור. כאשר כבר קיים משתנה, בין אם זה איבר יחיד ובין אם זה מערך חד/רב ממדי, ניתן פשוט לעשות קאסטינג מפורש:

```
data = [[1, 2],[3, 4]]
x_data = torch.tensor(data)
```

במידה והמשתנה הוא כבר מסוג NumPy array, ניתן להשתמש בפונקציה מובנית על מנת לקבל טנזור:

```
np_array = np.array([[1, 2],[3, 4]])
x_np = torch.from_numpy(np_array)
```

דרך אחרת לאתחל טנזורים הינה על ידי אתחול טנזור באופן מפורש. לשם כך ראשית יש לקבוע את הממד שלו (Shape), ולאחר מכן לקבוע כיצד יאותחל אותו טנזור, שהממד שלה כבר ידוע. ערכי האתחול הנפוצים הינה אפסים, אחדות, ומספרים רנדומליים מהתפלגות אחידה:

```
shape = (2,3)
rand_tensor = torch.rand(shape)
ones_tensor = torch.ones(shape)
zeros_tensor = torch.zeros(shape)
```

מלבד הממד של הטנזור, ניתן גם לקבוע בפירוש את ה-type של כל איבר, כאשר הדיפולט הוא torch.float32. כמו כן, ניתן לקבוע לאיזה מכשיר יקושר הטנזור – CPU או GPU. ניתן לעשות זאת על ידי הוספת הארגומנט "device" או לחילופין לאחר הגדרת הטנזור לקשר אותו למכשיר על ידי הפקודה ("cuda")to..