知能プログラミング演習 I 第 6 回レポート

XXXX 年 YYYY 月 ZZZZ 日　学籍番号　氏名

課題１

forward 関数を実行して各ステップでのテンソルサイズを確認した結果：

- 入力時: torch.Size([100, 1, 28, 28])
- conv 層通過後: torch.Size([100, 6, 26, 26])
- ReLU 適用後: torch.Size([100, 6, 26, 26])
- MaxPool2d 適用後: torch.Size([100, 6, 13, 13])
- Flatten 適用後: torch.Size([100, 1014])
- 線形層出力: torch.Size([100, 10])

サイズが変化する理由について考察：

- 最初は 100 個のバッチで 28x28 の 1 チャンネル画像
- 畳み込み層で 3x3 のフィルタ、パディング無しなので各辺が 2 減って 26x26 になる
- ReLU は要素ごとの処理なのでサイズは変わらない
- MaxPooling で 2x2 の窓で最大値を取るので各辺が半分の 13x13 になる
- Flatten で全て 1 次元に並べるので 6*13*13=1014 次元のベクトルになる
- 最後の線形層で 10 クラス分類用の 10 次元に変換される

課題２

課題の指示に従って CNN クラスを実装した。最初、線形層の入力サイズの計算で少し悩んだが、課題 1 のサイズ確認方法を参考にして 6*7*7=294 として設定した。

実行して生成された error.pdf を見ると、青い線が訓練ロス、赤い線がテスト精度を表している。

観察したこと：

- 訓練ロスは順調に下がっている
- テスト精度は 97.4%→ 98.3%→ 97.6%と推移
- epoch 2 で一番良い結果が出た後、少し下がった
- 全体的には高い精度を維持している
- 思ったより簡単な構造でも良い結果が出て驚いた

課題３

Fashion MNIST は普通の MNIST より難しそうなので、いろいろ試してみた。
最初はもっと複雑なネットワークを作ったが、計算時間がかかりすぎたので簡単にした。最終的な設定：

- 畳み込み層を 3 つ（16, 32, 64 チャネル）
- 各層の後に ReLU と MaxPooling
- Dropout で過学習を防ぐ（0.5 に設定）
- 全結合層は 128 次元経由で 10 クラスに分類
- 学習率は 0.001 の Adam を使用
- 5 エポックで学習

実行結果：

- 最終的に 89.1%の精度を達成
- 最初は 81.5%だったが徐々に改善
- Fashion MNIST は服の種類を分類するので通常の MNIST より難しいと思っていたが、思ったより良い結果になった
- ドロップアウトが効いているようで、安定して学習できた