hw5 report

Problem & Data

本作業使用 Fashion-MNIST，共包含 60,000 筆訓練資料與 10,000 筆測試資料。每張影像為灰階、尺寸 28×28，共 10 個類別（T-shirt、Trouser、Pullover…）。資料統一縮放至[0,1]。我的實作將原始訓練集再切分為 training/validation（90%/10%），並將影像 reshape 為 CNN 可接受的(1,28,28) 格式。

preprocessing: 使用horizontal flip進行data segmentation。

Models

improved nn

script output: **159040**

```
Configuration: Namespace(train_path='train.csv', test_path='test4students.csv', batch_size=64, val_split=0.1, seed=42, epochs=40, lr=0.001, model_type='improved_nn', optimizer='adam', device=device(type='cuda', index=3), use_bn=1, use_dropout=0, use_residual=0, pooling='max', output_dir='results_nn_ablation/bn1_do0', use_plateau=1, gamma=0.5, plateau_patience=2)
Number of training batches: 844
Number of validation batches: 94
Number of test batches: 157
Model: IMPROVED_NN
Trainable Parameters: 159040
```

hand calculation:

layers	parameters count
Flatten	—
Linear(784→180)	141,120 + 180 = 141,300
bn1: BatchNorm1d(180)	180 + 180 = 360
Linear(180→90)	16,200 + 90 = 16,290
bn2: BatchNorm1d(90)	90 + 90 = 180
Linear(90→10)	900 + 10 = 910
Total	**159040**

improved cnn

script output: **162170**

```
Configuration: Namespace(train_path='train.csv', test_path='test4students.csv', batch_size=64, val_split=0.1, seed=42, epochs=40, lr=0.001, model_type='improved_cnn', optimizer='adam', device=device(type='cuda', index=3), use_bn=1, use_dropout=0, use_residual=0, pooling='avg', output_dir='results_cnn_ablation/bn1_do0_res0_poolavg', use_plateau=1, gamma=0.5, plateau_patience=2)
Number of training batches: 844
Number of validation batches: 94
Number of test batches: 157
Model: IMPROVED_CNN
Trainable Parameters: 162170
```

hand calculation:

layers	parameters count
Conv2d(1 → 24, kernel=3, padding=1)	conv: 1*24*3*3+24=240 batch norm: 24+24=48 240+48=288
Conv2d(24 → 48, kernel=3, padding=1)	conv: 24*48*3*3+48=10416 batch norm: 48+48=96 10416+96=10512
Linear(48*7*7→64)	2352*64=150,528 150528+64=150592
bn: BatchNorm1d(64)	64 + 64 = 128
Linear(64→10)	640+10 = 650
Total	**162170**

Training

所有模型統一使用以下訓練配置：

Optimizer

- Adam optimizer
- Learning rate: 1e-3
- Weight decay: 1e-4

LR Schedule

- ReduceLROnPlateau
- Patience: 2 epochs
- Factor: 0.5
- mode: min

Hyperparameters

- Epochs: 40
- Batch size: 64
- Loss: CrossEntropyLoss with label smoothing=0.1

Others

- data augmentation: horizontal flip
- seed: 42

Results

overview

我總共對NN執行了 $2 * 2 = 4$ 個實驗（**use_bn=0 or 1, use_dropout=0 or 1**）

並對CNN執行了 $2 * 2 * 2 * 2 = 16$ 個實驗 （**use_bn=0 or 1, use_dropout=0 or 1, use_residual=0 or 1, pooling=max or avg**）

並對BaselineNN/BaselineCNN個別執行了 1 次實驗

總共執行了 22 次實驗。

accuracy

我選擇用ImprovedCNN (bn1_do0_res0_poolavg)當作提交的版本，因為他的valiation score最高，為94.43。

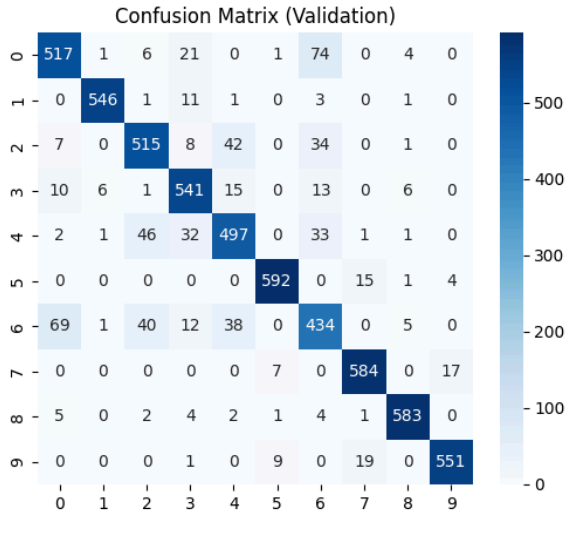
從 test accuracy 也可以看到，ImprovedCNN 整體的表現最好。

model	validation accuracy	test accuracy
BaselineNN	89.33	89.20
BaselineCNN	91.78	92.00

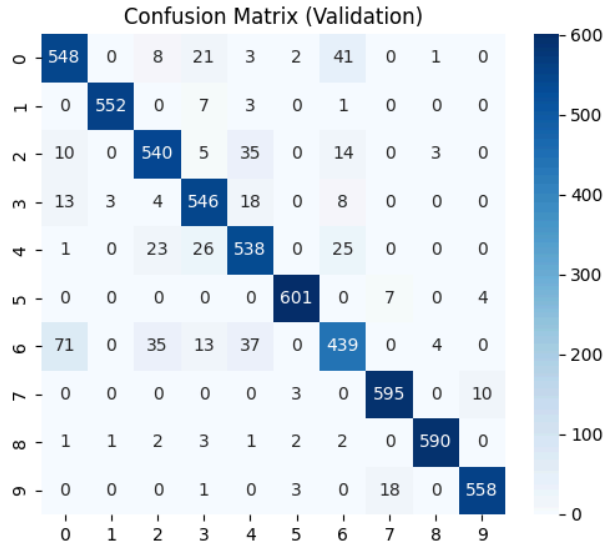
ImprovedNN (bn1_do0)	90.50	89.90
ImprovedCNN(bn1_do0_res0_poolavg)	94.43	94.50

confusion matrix

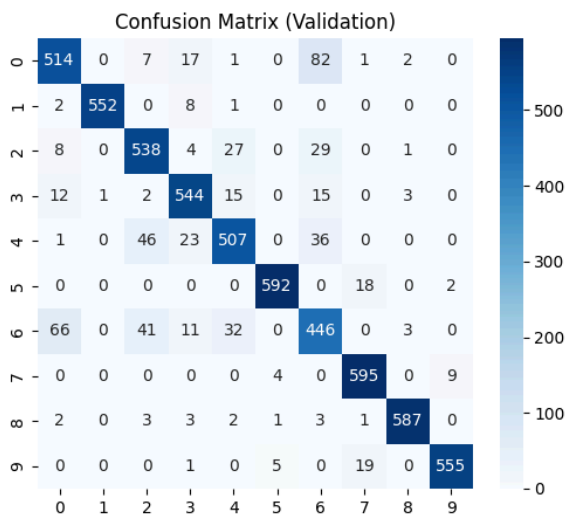
- BaselineNN



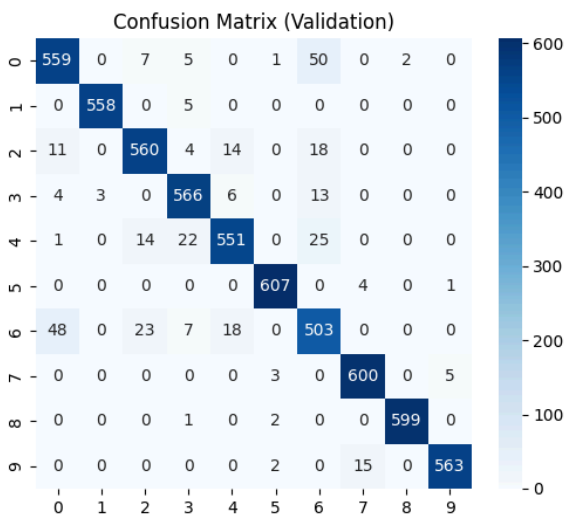
- BaselineCNN



- ImprovedNN(bn1_do0)

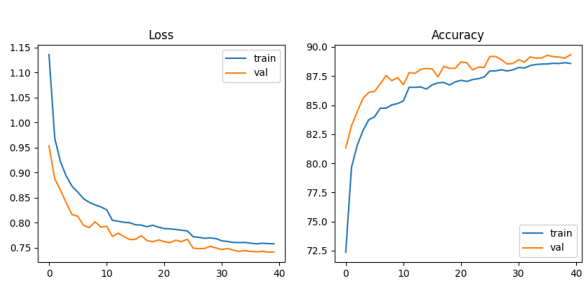


- ImprovedCNN(bn1_do0_res0_poolavg)

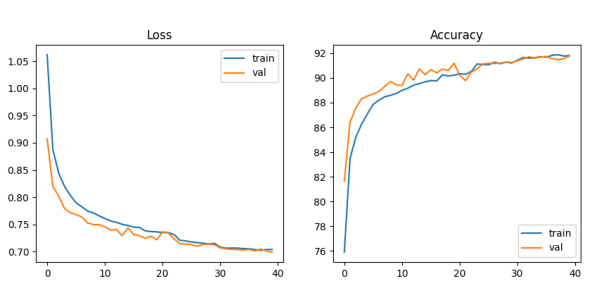


learning curves

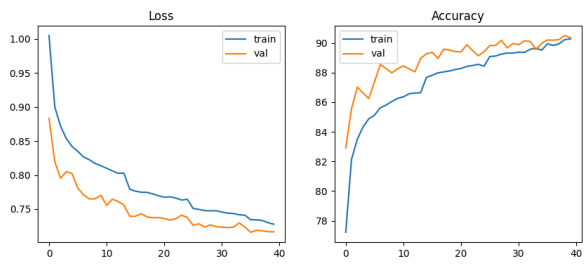
- BaselineNN



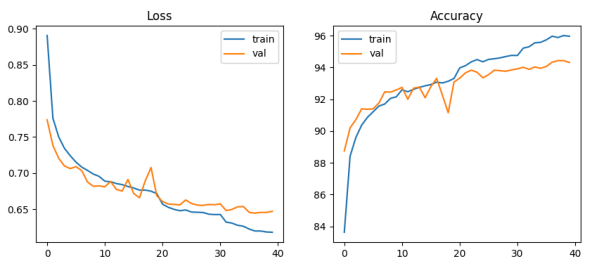
- BaselineCNN



- ImprovedNN(bn1_do0)



- ImprovedCNN(bn1_do0_res0_poolavg)



detailed ablation table

總共分為三個部分，part 1對nn做ablation，總共2*2=4種組合；part 2對cnn做ablation，總共2*2*2*2=16種；part 3則是baseline nn/cnn，總共2種。同時我呈現了扣除baseline前、後的原始參數、準確度數據。

experiment	use_bn	use_dropout	use_residual	pooling	params	Δparams vs 162,170	best_val_acc	Δaccuracy vs 94.43
Part 1: improved nn ablation								
bn0_do0	0	0	0	max	158500	-3,670	90.02	-4.41
bn0_do1	0	1	0	max	158500	-3,670	89.93	-4.50
bn1_do0	1	0	0	max	159040	-3,130	90.50	-3.93
bn1_do1	1	1	0	max	159040	-3,130	90.37	-4.06
Part 2: improved cnn ablation								
bn0_do0_res0_poolavg	0	0	0	avg	161898	-272	92.47	-1.96
bn0_do0_res0_poolmax	0	0	0	max	161898	-272	93.28	-1.15
bn0_do0_res1_poolavg	0	0	1	avg	163146	+976	92.33	-2.10
bn0_do0_res1_poolmax	0	0	1	max	163146	+976	92.87	-1.56
bn0_do1_res0_poolavg	0	1	0	avg	161898	-272	92.60	-1.83
bn0_do1_res0_poolmax	0	1	0	max	161898	-272	93.18	-1.25
bn0_do1_res1_poolavg	0	1	1	avg	163146	+976	92.20	-2.23
bn0_do1_res1_poolmax	0	1	1	max	163146	+976	93.20	-1.23
bn1_do0_res0_poolavg	1	0	0	avg	162170	0	94.43	0.00
bn1_do0_res0_poolmax	1	0	0	max	162170	0	94.10	-0.33
bn1_do0_res1_poolavg	1	0	1	avg	163418	+1,248	93.98	-0.45
bn1_do0_res1_poolmax	1	0	1	max	163418	+1,248	94.07	-0.36
bn1_do1_res0_poolavg	1	1	0	avg	162170	0	94.38	-0.05
bn1_do1_res0_poolmax	1	1	0	max	162170	0	94.05	-0.38
bn1_do1_res1_poolavg	1	1	1	avg	163418	+1,248	94.08	-0.35
bn1_do1_res1_poolmax	1	1	1	max	163418	+1,248	93.97	-0.46
Part 3: baseline nn/cnn								
baseline_nn	0	0	0	none	80506	-81,664	89.33	-5.10
baseline_cnn	0	0	0	none	76218	-85,952	91.78	-2.65

Discussion

Findings

整體實驗結果顯示，在參數量相近的情況下（例如 Improved NN ≈ 159k 參數、Improved CNN ≈ 162k 參數），CNN 都優於 NN，通常能多取得約 2–4% 的 validation accuracy。差異主要來自 CNN 的 inductive biases：locality 與 weight sharing 使其能有效捕捉 Fashion-MNIST 中的局部形狀特徵，而 NN 即使擁有相同等級的參數量，也難以從扁平化的輸入中學到等量的空間資訊。

從 ablation studies 中亦可觀察到：BatchNorm 是最具一致性的正向因素，可提升 0.5–1% accuracy 並穩定訓練；Pool 的選擇對性能影響最大，其中 avg pooling 在本架構中表現最佳，而 stride pooling 與無 pooling 皆有明顯下降。相對地，dropout 在本 dataset 與模型規模下反而會造成輕微 underfitting，使 performance 略為下降。Residual connection 在本次較淺的 CNN 上幾乎沒有帶來額外效益。Confusion matrix 亦反映了 CNN 對含較多空間結構的類別（如 Pullover、Coat、Shirt）改善最為顯著，說明其特化於影像資料的架構優勢。綜合來看，最佳模型組合（BN + no dropout + no residual + avg pooling）代表了在中等深度 CNN 上最有效的正則化策略，而整體結果也清楚展示出 CNN 在影像分類任務上的結構性優勢與模型設計對泛化性能的影響。

Analysis

本次分析以 **ImprovedCNN (bn1_do0_res0_poolavg)** 作為基準模型，其最佳驗證準確率為 **94.43%**，參數量 **162,170**。下面的結果我都以這個模型為比較的基準，並直接引用 ablation table 中的數據。

1. BatchNorm — 對性能影響最大

   移除 BatchNorm 的結果如下：

   - 實驗：`bn0_do0_res0_poolavg`
   - Val Acc：**92.47%**
   - Δ Accuracy：**−1.96%**
   - Params：**161,898**（−272）

   這是所有 ablation 裡性能下降最多的項目。

   證明**BN 是本模型中影響最關鍵的因素，可以穩定訓練並提升泛化能力，是改進效果最顯著的元素。**

2. Pooling — AvgPool 的效果大於 MaxPool

   將 AvgPool 換成 MaxPool：

   - 實驗：`bn1_do0_res0_poolmax`
   - Val Acc：**94.10%**

- Δ Accuracy：**−0.33%**
- Params：**162,170**（差異 0）

AvgPool 在本架構中較能保留有效特徵，性能比 MaxPool 稍高。

從無 BN 的組合中也觀察到同樣現象：

- `bn0_do0_res0_poolavg → 92.47%`
- `bn0_do0_res0_poolmax → 93.28%`
  （Max 反而更高，但兩者皆遠低於含 BN 的模型）

說明 pooling 的效果 **會與 BN 交互影響**，但 AvgPool 在最佳組合中仍佔上風。

3. Residual Connection — 增加參數但性能略降

   加入 residual：

   - 實驗：`bn1_do0_res1_poolavg`
   - Val Acc：**93.98%**
   - Δ Accuracy：**−0.45%**
   - Params：**163,418**（+1,248）

   在此 relatively shallow CNN 中，Residual 並未帶來增益，反而使性能下降 0.45%。

   顯示 **Residual 的好處在淺層架構中有限**。

4. Dropout — 輕微 underfitting，效果負向

   加入 Dropout 的結果如下：

   - 實驗：`bn1_do1_res0_poolavg`
   - Val Acc：**94.38%**
   - Δ Accuracy：**−0.05%**
   - Params：162,170（相同）

   Dropout 讓性能輕微下降 0.05%，幾乎可視為無差。

   在這個 dataset（Fashion-MNIST）模型的參數規模下，**Dropout 帶來 underfitting，而非改善泛化。**

綜合結論

透過 ablation，可以得到以下的排序：

Component	Δ Accuracy	結論
BatchNorm	1.96%	影響最大、最必需
Pooling (Avg > Max)	0.33%	AvgPool 優於MaxPool
Residual	0.45%	對CNN的效果有限
Dropout	0.05%	會導致輕微的 underfitting

最佳模型組合為：BN + No Dropout + No Residual + AvgPool

即 ImprovedCNN(bn1_do0_res0_poolavg)，其驗證準確率達 **94.43%**。

這個結果明確展示：

- **BN 提供穩定訓練與最佳泛化效果**
- **Pooling 類型次之，AvgPool 在本架構中最適合**
- **Residual 在淺層 CNN 中沒有必要**
- **Dropout 在中等規模模型會導致 underfitting**

Reproducibility

- 關於重現結果：
  - 我嘗試過set_seed()，但會發現用mac的mps跑，和cuda(cu118)跑出來的結果會不一樣。這點是無法避免的，因此，若要結果完全一致，必須使用torch=2.7.1+cu118的版本，如下所述：

    ```
    def set_seed(seed):
        random.seed(seed)
        torch.manual_seed(seed)
        np.random.seed(seed)
        if torch.cuda.is_available():
            torch.cuda.manual_seed_all(seed)
            torch.backends.cudnn.deterministic = True
            torch.backends.cudnn.benchmark = False
    ```

  - 我使用torch=cu118。因此如果torch版本不一樣，可能結果會有微小的差異。如果裝置一樣，結果是可以重現的。

```
Package                Version
---------------------- -----------
torch                  2.7.1+cu118
```

- 執行方式：參考readme.md