

Week 10 HW#5: Build & Compare NN vs CNN (Image Classification)

Release: Nov 6, 2025

Due: Nov 26, 2025 23:59:59 (Kaggle), Nov 27, 2025 23:59:59 (Code submission)

Work mode: Individual (you may discuss high-level ideas; code must be your own)

1) Learning objectives

By the end of this assignment you should be able to:

1. Design, train, and debug a simple **fully connected neural network (NN/MLP)** and a **convolutional neural network (CNN)** for the **same** classification task.
 2. **Count parameters** for Linear/Conv; relate capacity to over/underfitting.
 3. Empirically evaluate the effects of **Batch Normalization (BN)**, **Residual connections**, **Dropout**, and **Pooling**.
 4. Practice **reproducible experiments** and communicate findings clearly.
-

2) Dataset (required)

Use: Fashion-MNIST (same split for NN & CNN).

About the dataset

- **Type:** Zalando article images (10 clothing categories)
- **Images:** 28×28 **grayscale** (single channel), centered, size-normalized
- **Splits:** **60,000** training images, **10,000** test images
- **Classes (10):** T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot

Input to models

- **NN (MLP):** **flatten** each 28×28 to a 784-D vector.
 - **CNN:** keep as tensor of shape **(B, 1, 28, 28)**.
-

3) What you must build

A. Baselines (start here)

1. Baseline NN (MLP)

- Architecture: Input → Linear → ReLU → Linear → ReLU → Linear → Softmax (in loss)
- No BN/Dropout initially.

2. Baseline CNN

- Example: [Conv(3×3, 32)] → ReLU → [Conv(3×3, 64)] → ReLU → **MaxPool(2×2)** → Flatten → Linear → ReLU → Linear → Softmax (in loss)
- No BN/Dropout/Residual initially.

B. Your designs (improve baselines)

- Propose **one improved NN** and **one improved CNN**.
- You **must** run ablations for: **BN**, **Residual (skip) block(s)**, **Dropout**, and **Pooling** (where

applicable).

- Keep parameter counts within limits (see §4).

C. Training protocol (apply to all models)

- Optimizer: Adam or SGD (your choice; justify).
 - Epochs: ≥ 15
 - Batch size: 64–256.
 - Learning rate: start from $1e-3$ (Adam) or 0.1 (SGD with decay) and tune.
 - **Validation split**: hold out 5–10% of training for model selection & early stopping.
 - **Random seed**: fix and report (e.g., 42).
-

4) Parameter counting

Layer formulas (trainable params only):

- **Linear(in, out)**: $\text{in} \times \text{out} + \text{out}$
- **Conv2d(in_c, out_c, kH, kW, bias=True)**: $(\text{kH} \times \text{kW} \times \text{in}_c) \times \text{out}_c + \text{out}_c$

Reporting

Report the **exact trainable parameter count** for every model in your table (you may show hand calculations and include a script output).

Note: hand calculations by layer formulas are required but for BatchNorm/Residual Block... script output is acceptable.

Percentile-based efficiency bonus

After everyone submits, we will compute distributions **separately for NN and CNN** (on the same dataset):

- Let **P10(params)** be the 10th-percentile parameter count, and **P90(acc)** be the 90th-percentile **test accuracy**.
- If your model's **parameter count** $< \text{P10(params)}$ and your **test accuracy** $\geq \text{P90(acc)}$ for that method (NN or CNN), you earn an **efficiency bonus**.
- This encourages **small-yet-accurate** designs in a fair, class-relative way and replaces the earlier fixed parameter budgets.

5) Experiments you must run (Ablations)

For **each method (NN & CNN)**, perform the following toggles **one at a time** starting from your improved design:

1. **+BN** vs **-BN** (keep everything else identical).
2. **+Residual** vs **-Residual** (CNN only).
3. **+Dropout** vs **-Dropout**.
4. **Pooling**: MaxPool/AvgPool vs **stride-2 conv** vs **no downsampling** (CNN only).

Keep training steps (epochs, LR schedule) identical across toggles. Report the **best validation accuracy, final test accuracy, training curves, and param counts**.

6) What to submit

A. Report (PDF, ≤ 5 pages) Include the following sections (use the headings):

1. **Problem & Data:** dataset choice (Fashion-MNIST), class examples, split strategy, preprocessing.
2. **Models:** detailed layer tables (shape, kernel, stride, padding), **parameter counts** (show calculations and verification by script output).
3. **Training:** optimizer, LR schedule, epochs, batch size.
4. **Results:**
 - Accuracy (val/test), confusion matrix, learning curves (loss/accuracy vs epoch).
 - Ablation table: BN/Residual/Dropout/Pooling effects (Δ accuracy, Δ params).
5. **Discussion:** compare NN vs CNN on the **same** data. When does CNN help and why? Link to **inductive biases** (locality, weight sharing). Discuss capacity/regularization trade-offs and the role of BN/Dropout/Residual/Pooling. Any other findings?

B. Kaggle

- On Kaggle, your submission is a CSV file `pred.csv` of predicted class labels for all test images.
- Link: <https://www.kaggle.com/competitions/hw-5-comparing-between-nn-cnn-using-fashion-mnist>

C. Code (zip)

- Either .py or .ipynb is fine; include a short README with run instructions.

Submission packaging (E3 System)

- Compress your **code and report** together; **do not** include trained models or datasets.
- **Filename (required):** <studentID>_hw5.zip
- Only the **last submission** will be graded.
- **Reasonableness check:** if your code is not reasonable (e.g., missing vital parts, cannot run, or results clearly inconsistent with the report), your final grade will be **multiplied by 0.9**.
- Your .zip should contain **only**:
 - code/ (or top-level scripts): .py and/or .ipynb
 - report.pdf

7) Grading rubric (100 pts)

- **Correctness & Reproducibility (25):** code runs with provided seed; results match report; clean repo.
- **Design & Parameter Accounting (20):** valid architectures; accurate param counts; clear layer tables.
- **Experimental Rigor (25):** proper splits, learning curves, confusion matrix.
- **Analysis & Insight (20):** compares NN vs CNN, interprets BN/Residual/Dropout/Pooling effects; error analysis grounded.
- **Presentation (10):** clear writing, figures/tables, ≤5 pages, professional formatting.

Extra credit (+10)

- **Parameter-efficient & accurate:** For **each** of NN and CNN, if your model's params are **below the class 10th percentile** while its **test accuracy is at or above the class 90th percentile** (computed after all submissions), you receive **+5 points per method** (up to **+10** total). Ties at the threshold are counted as meeting the criterion.