Lab 5: MLP Hyperparameter Evaluation Report

Course: Introduction to Artificial Intelligence (Summer 2025)
Task: Checking how different settings affect an MLP for image classification.
Dataset: FashionMNIST

1. Introduction

Here's a rundown of the experiments we did for Lab 5. The main goal was to build a Multilayer Perceptron (MLP) and see how tweaking key settings – hyperparameters – changed how well the model performed. We followed the plan for Variant 4 from the lab instructions, looking specifically at learning rate, mini-batch size, hidden layers, and loss functions using the FashionMNIST dataset.

2. How We Did It (Methodology)

We started with a baseline setup for the MLP. Then, we ran a series of tests where we changed just one setting at a time, keeping everything else the same as the baseline. The train.py script handled the training, splitting the data (50k for training, 10k for validation), building the model, and checking its performance. Each setup was trained for 10 epochs using the basic Stochastic Gradient Descent (SGD) optimizer. We judged performance based on the final accuracy on the validation set.

Baseline Setup:

- Learning Rate (lr): 0.01
- Batch Size (bs): 32
- Hidden Layers (hl): [128] (Just one hidden layer with 128 neurons)
- Loss Function (loss_fn): Cross Entropy (CE)

Baseline Final Validation Accuracy: 84.36%

3. Results and Observations

Let's look at the final validation accuracy we got when changing each setting.

3.1. Learning Rate (LR) Test

(Baseline: lr=0.01, bs=32, hl=[128], loss_fn=CE)

- lr=0.1: **87.49%** (~+3% vs baseline) - *Best*
- lr=0.01: 84.36% (Baseline)
- lr=0.001: 79.12% (~-5% vs baseline)

What we saw: The learning rate controls how big the steps are when the model learns. A higher rate (0.1) actually gave the best results in our 10-epoch runs, probably because it helped the model learn faster in that short time. The really low rate (0.001) didn't do well, suggesting the steps were too tiny to get very far in just 10 epochs.
Though I thought a smaller value would be better, but maybe it's within the margin of error or due to 10 epochs, higher numbers of runs might have been netter.

3.2. Batch Size (BS) Test

(Baseline: lr=0.01, bs=32, hl=[128], loss_fn=CE)

- bs=1: **88.35%** (~+4% vs baseline) - *Best*
- bs=32: 84.36% (Baseline)
- bs=256: 79.82% (~-4.5% vs baseline)

What we saw: Batch size is about how many examples the model looks at before updating itself. Using just one example at a time (bs=1, basically pure SGD) gave the best accuracy. It seems like the frequent, noisy updates might have helped the model explore better or avoid getting stuck in bad spots within our 10 epochs. The large batch size (256) didn't work well; maybe the updates were too smooth and led it to a poor solution, or maybe it just needed a different learning rate.

3.3. Hidden Layers (HL) Test

(Baseline: lr=0.01, bs=32, hl=[128], loss_fn=CE)

- hl=[] (Linear Model): 83.98% (~-0.4% vs baseline)
- hl=[128] (Baseline): 84.36%
- hl=[256, 128]: **85.19%** (~+1% vs baseline) - *Best*

What we saw: Hidden layers give the model power to learn complex patterns. Adding a second layer gave a small boost (~1%). Interestingly, the model with no hidden layers (just a linear model) wasn't much worse than our baseline single-layer model.
Again I expected bigger improvements but I think fewer epochs are to blame. Still there is a small improvement.

3.4. Loss Function (LossFn) Test
(Baseline: lr=0.01, bs=32, hl=[128], loss_fn=CE)

- loss_fn=CE: **84.36%** (Baseline) - *Best*
- loss_fn=MSE: 69.19% (~-15% vs baseline)
- loss_fn=MAE: 62.51% (~-22% vs baseline)

What we saw: The loss function tells the model what to aim for. Cross Entropy (CE) is made for classification problems like this, and it worked best. Mean Squared Error (MSE) and Mean Absolute Error (MAE) are really for regression (predicting numbers, not categories). Trying to use them here, even with some adjustments, gave much worse results (~15-22% lower accuracy). They just aren't the right tool for the job.

Seems about ok.

4. Conclusions

So, after running these tests one setting at a time:

- **Learning Rate and Batch Size** seemed to matter most for getting good accuracy within our 10-epoch limit. A fairly high learning rate (0.1) and a tiny batch size (1) gave the best scores, which was a bit different from what you might typically expect. This suggests faster learning and maybe some helpful noise from the updates were key in this short timeframe.
- **Network Structure (Hidden Layers)** didn't change things as much. The small difference was observed though.
- **Loss Function Choice** was super important. Cross Entropy was clearly the way to go, while the regression losses (MSE, MAE) performed poorly, just as expected.

The best individual settings we found were lr=0.1 (87.49% accuracy) and bs=1 (88.35% accuracy). Trying to combine the best settings might lead to even better results.

Previous run was a cross grid search to find the best method however it ended up taking a very long time even with 5 epochs only, and with 5 epochs the **data was not stable enough to comment. But here are the conclusions of the previous report version.**

1. Learning Rate

- High rate (0.1): Led to rapid initial decrease but unstable oscillations/divergence. Final Val Acc: 87.49%
- Medium rate (0.01): Provided smooth convergence. Final Val Acc: 84.36%
- Low rate (0.001): Was stable but too slow; validation accuracy lagged. Final Val Acc: 79.12%

2. Batch Size

- Batch=1 (online): Produced very noisy gradients but achieved the highest final accuracy due to regularizing noise. Final Val Acc: 88.35%
- Batch=32: Balanced stable gradients with update frequency—our recommended default. Final Val Acc: 84.36%
- Batch=256: Yielded smooth trajectories but lower final accuracy and slower generalization. Final Val Acc: 79.82%

3. Number & Width of Hidden Layers

- 0 layers (linear model): Peaked early, indicating underfitting. Final Val Acc: 83.98%
- 1 hidden layer (128 neurons): Boosted accuracy; non-linearity is beneficial. Final Val Acc: 84.36%
- 2 hidden layers (256⟶128): Reached the highest accuracy among layer tests but added ~30% training time; diminishing returns. Final Val Acc: 85.19%

4. Loss Function

- Cross-Entropy: Fastest convergence and highest final accuracy. Final Val Acc: 84.36%
- MSE: Slower training, less stable gradients, significantly lower accuracy. Final Val Acc: 69.19%
- MAE: Very slow and erratic—unsuitable for classification. Final Val Acc: 62.51%

Recommended Best Setup

- Learning rate: 0.01
- Batch size: 32
- Hidden layers: 2 (256⟶128)
- Loss function: Cross-Entropy