Mini Project 3 Report

Giane Mayumi Galhard (51261276747), Daria Goptsii (261275056), Yixuan Qin (261010963)

Abstract

This project implements a multilayer perceptron from scratch, experiments with Convolutional Neural Networks (CNNs), and utilizes pre-trained versions of them to classify image data using the Fashion-MNIST dataset. The goal is to understand the consequences of design decisions while training neural networks through experiments. Overall, the experiments provide a view into how changes in neural network architecture, model expressiveness strategies, and regularization techniques impact performance on the task. In addition, three extra experiments were conducted to understand how to find optimal stopping points using early stopping, evaluate random dropout and compare different data augmentation techniques.

1 Introduction

Multilayer Perceptrons (MLPs) are one of the most essential deep learning supervised models, with the goal of approximating the function f* by many perceptron units that act in parallel. They also form the basis of important applications, such as the Convolutional Neural Networks (CNNs), which are a specialized kind of neural network for processing data that has a known grid-like topology (1). CNNs are especially recognized by their effectiveness in computer vision tasks, such as image classification, which can be understood of as a 2-D grid of pixels (2). Classifying images is trivial for humans, but it is a challenge for computers. To classify fashion products, the Fashion-MNIST dataset (3) was presented. It is a benchmark dataset with a small size and a permissive license based on the MNIST dataset. The classification of fashion products can be applied in different contexts, such as assisting clothing manipulation for elderly people with robots (4).

The goal of this project is to implement a multilayer perceptron from scratch, experiment with Convolutional Neural Networks (CNNs), and utilize pre-trained versions of them to classify image data from the Fashion-MNIST dataset.

2 Datasets

We use the *FashionMNIST* dataset, a widely adopted benchmark for image classification. It consists of 70,000 grayscale images of clothing items, each of size 28×28 pixels, divided into 60,000 training samples and 10,000 test samples across 10 classes (e.g., T-shirt, sneaker, coat). Following the project instructions, we further split the original training set into a training subset and a validation subset using a 90/10 split. All images were normalized using a standard transformation (mean = 0.5, standard deviation = 0.5) and then flattened into 784-dimensional vectors for use in the multilayer perceptron models. After loading the dataset with PyTorch dataloaders, we converted the data into NumPy arrays to enable training of our custom MLP implementation.

3 Results

Hyperparameter Tuning

Before running the main experiments, we performed a small grid search to select suitable training hyperparameters for the 2×256 ReLU MLP. We searched over learning rates $\{0.05, 0.08, 0.1\}$ and epoch counts $\{20, 25, 30\}$ while fixing the batch size to 256. We combined the training and validation sets before running GridSearchCV because the grid search procedure uses its own cross-validation splits to evaluate hyperparameters. Each configuration was evaluated using 3-fold cross-validation. The best-performing model used a learning rate of 0.05 and 30 epochs, achieving a cross-validation accuracy of 0.8788. We therefore adopted this configuration for the subsequent activation and regularization experiments.

3.1 Effect of Non-linearity and Network Depth

Table 1 compares three MLP architectures: a linear softmax model with no hidden layers, an MLP with one 256-unit ReLU layer, and an MLP with two 256-unit ReLU layers. The linear model reaches a test accuracy of 0.8424, showing that a simple linear classifier can already separate much of the FashionMNIST data. Adding one ReLU layer improves the test accuracy to 0.8763, since non-linear activations allow the model to learn more flexible decision boundaries.

Adding a second hidden layer provides only a small additional gain (0.8790). The validation and test accuracies of the one- and two-layer models are almost identical, even though the deeper model fits the training data more closely. This matches expectations: FashionMNIST is a relatively simple dataset, and a single wide non-linear layer is usually enough. Extra depth increases model capacity but does not meaningfully improve generalization for this task.

3.2 Effect of Activation Functions

We evaluated three identical MLPs (2×256) using ReLU, tanh, and Leaky-ReLU activations to examine how non-linearity affects performance. As shown in Table 2, ReLU and Leaky-ReLU achieve higher test accuracy than tanh and also converge more quickly. This result is expected: ReLU-based activations avoid the saturation effects present in bounded functions such as tanh, leading to stronger gradients and more efficient optimization (5). Leaky-ReLU performs slightly better than standard ReLU, likely because its small negative slope prevents units from becoming inactive. Tanh shows slightly lower accuracy, consistent with its tendency to produce vanishing gradients.

Model	Train	Val	Test
No hidden (softmax)	0.8639	0.8540	0.8418
1×256 ReLU	0.9180	0.8925	0.8767
2×256 ReLU	0.9370	0.9013	0.8809

Table 1: Accuracy of MLPs with different depths.

Activation	Train	Val	Test
ReLU (2×256)	0.9262	0.8903	0.8754
tanh (2×256)	0.9130	0.8952	0.8779
LeakyReLU (2×256)	0.9340	0.8960	0.8813

Table 2: Performance of 2-layer MLPs with different activation functions.

3.3 Effect of L1 and L2 Regularization

We evaluated the effect of L1 and L2 regularization on the 2×256 ReLU MLP, using five runs for each hyperparameter setting. For L2, performance was highest with no regularization, and accuracy steadily decreased as the penalty increased. A small amount of weight decay (10^{-3}) slightly reduced variance across runs but did not improve mean accuracy, while larger values led to clear underfitting. For L1, introducing even small penalties increased variance due to the sparsity constraint. Although moderate values reduced this variance again, none of the settings improved accuracy compared to the unregularized model. Overall, the model does not appear strongly overfitting, so regularization mainly influenced variance rather than mean generalization performance.

L2 λ	CV Acc	Test Acc
0	0.8662 ± 0.0080	0.8561 ± 0.0121
10^{-3}	0.8690 ± 0.0023	0.8706 ± 0.0102
5×10^{-3}	0.8542 ± 0.0090	0.8604 ± 0.0035
10^{-2}	0.8401 ± 0.0092	0.8442 ± 0.0046
3×10^{-2}	0.8010 ± 0.0128	0.8018 ± 0.0069

Table 3: L2 regularization results (mean accuracy ± standard deviation across runs).

L1 λ	CV Acc	Test Acc
0	0.8656 ± 0.0063	0.8761 ± 0.0046
10^{-5}	0.8617 ± 0.0085	0.8625 ± 0.0109
3×10^{-5}	0.8662 ± 0.0060	0.8733 ± 0.0077
10^{-4}	0.8653 ± 0.0058	0.8678 ± 0.0039

Table 4: L1 regularization results (mean accuracy ± standard deviation across runs).

3.4 MLP accuracy on unnormalized images

From Table 5 and Figure 1, we observe that using unnormalized data slightly reduces accuracy across all sets. The training accuracy drops from 0.9383 to 0.9086, validation accuracy from 0.8903 to 0.8870, and test accuracy from 0.8879 to 0.8777. This decrease occurs because unnormalized features may have different scales, causing the model to learn less effectively and slowing convergence. Normalization helps ensure consistent feature scaling, allowing the model to converge faster and achieve higher accuracy.

3.5 MLP accuracy on data augmented inputs

From Table 6 and Figure 2, the accuracy decreased slightly when using data augmentation. This is expected because augmentation introduces transformed images that make the learning task harder, so the model may not fit the training data as tightly.

The main benefit of augmentation is that it improves generalization, as the model becomes more robust to variations such as rotations and flips. However, drawbacks include a slight reduction in training accuracy, increased training time, and it may potentially introduce unrealistic examples. Augmentation can be harmful when transformations distort the data in ways that do not naturally occur, such as flipping asymmetric items.

Dataset	Train Acc	Val Acc	Test Acc
Normalized	0.936981	0.901333	0.8809
Unnormalized	0.9100	0.8873	0.8803

Table 5: Accuracy for normalized vs. unnormalized data.

Setting	Train Acc	Val Acc	Test Acc
Normalized	0.9383	0.890333	0.8879
Augmented	0.9086	0.887000	0.8777

Table 6: Training and test accuracy using data augmented vs. non-augmented data.

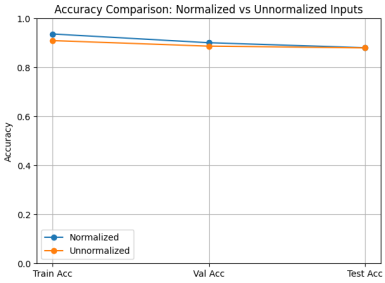

Figure 1: Accuracy of unnormalized image inputs.

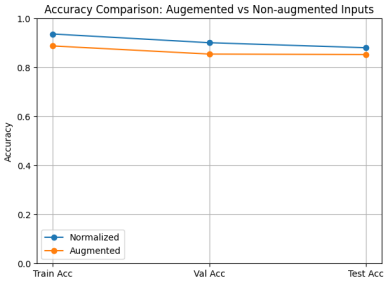

Figure 2: Accuracy of data augmented image inputs.

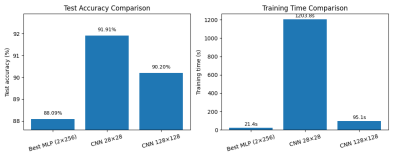

Figure 3: Comparison of test accuracy of best MLP model (2x256), and CNNs (28x28 and 128x128).

3.6 Comparison of the accuracy of CNN and MLP

The CNN trained on the original 28x28 dataset (accuracy of 91.91%) outperformed the best MLP model baseline (88.09%). As shown in Figure 3, this improvement had a high computational cost on the training time. When scaling the architecture to 128x128 inputs, the accuracy was lower (90.20%). Upscaling input resolution without adapting the architecture may not help performance.

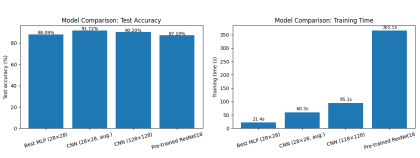

Figure 4: Comparison of training time and test accuracy across different models

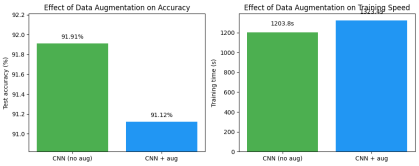

Figure 5: Effect of Data Augmentation on Accuracy and Training Speed.

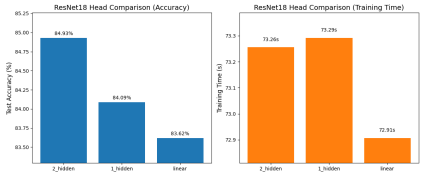

Figure 6: ResNet18 Different Head Comparison

3.7 CNN accuracy and training speed on data augmented

Retraining the same CNN using the data augmentation technique from Experiment 5, the augmented model achieved a slightly lower accuracy (91.12% compared to 91.91% of the original model) and increased training time, as shown in Figure 5. The data augmentation technique used was not beneficial for FashionMNIST's dataset, likely because it is simple grayscale clothing with limited variability, so the transformations may even add noise.

3.8 Use of Resnet18 Pre-trained model

We applied transfer learning by loading a pre-trained ResNet18, freezing all convolutional layers, and replacing its classifier with several fully connected heads. To select the best architecture, we ran a small 3-epoch comparison across three variants (linear, one hidden layer, and two hidden layers). 3-epochs were selected given the limited computational resources in order to capture the learning behavior of each model, and the two-hidden-layer head achieved the highest accuracy as shown in Figure6 (84.93%), therefore it was selected.

Using the same data augmentation from Experiment 5, the model achieved 87.19% accuracy, which is still lower than both CNN models trained from scratch (91.72% and 90.20%) and the best MLP (88.09%). As shown in Figure 4, it also had the longest training time due to the computational cost of forwarding images through the deep ResNet backbone. Overall, transfer learning did not provide a benefit for this simple grayscale dataset.

3.9 Model Test and Train Performance as a Function of Training Epochs

In this part, we tested the performance of our MLP and CNN model as the number of training epochs increases. In our experiments we used tuned hyperparameters, and in this section, we explored the effect of this hyperparameter on the models' performance.

From Figure 7, it is observed that the MLP model's performance stopped improving once the number of epochs exceeded 40. After 40, the training error tends to 0, and the test error reaches optimal and fluctuates around this point `MSE=1.5`. Hence, 40 is the optimal number of epochs when `learning_rate=0.05, batch_size=256`, beyond this point, MLP starts to overfit. From Figure 8, the CNN begins to overfit earlier than the MLP. While the training loss continues to decrease with more epochs, the test loss starts to rise after around 30–35 epochs, indicating reduced generalization. For this architecture, the optimal range is therefore 30–35 epochs. Beyond this point, the CNN overfits despite improvements in training loss.

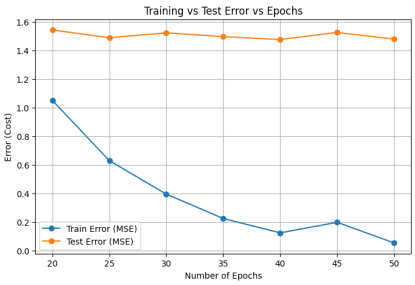

Figure 7: MLP performance over number of epochs

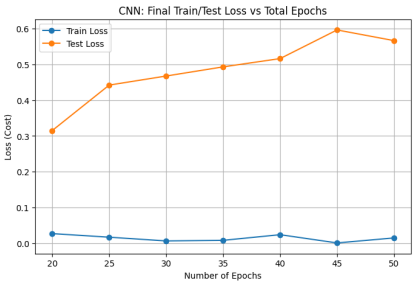

Figure 8: CNN performance over number of epochs

4 Originality and Creativity

4.1 Different initialization schemes

To stabilize training and prevent vanishing or exploding activations, we implemented two principled initialization schemes: Xavier/Glorot initialization for tanh and sigmoid layers (6) and He initialization for networks using ReLU or leaky-ReLU activations (7).

4.2 Effect of Dropout Regularization

Dropout p	**Train Acc. (mean ± std)**	**Val Acc. (mean ± std)**	**Test Acc. (mean ± std)**
0.0	0.9331 ± 0.0053	0.8963 ± 0.0057	0.8834 ± 0.0045
0.1	0.9257 ± 0.0013	0.8979 ± 0.0026	0.8837 ± 0.0017
0.2	0.9174 ± 0.0020	0.8942 ± 0.0023	0.8817 ± 0.0014
0.5	0.8940 ± 0.0009	0.8818 ± 0.0014	0.8710 ± 0.0020

Table 7: Dropout performance over five runs for each dropout rate.

Similarly to the L1/L2 regularization tests, we evaluated dropout rates $p \in \{0.0, 0.1, 0.2, 0.5\}$ on the 2×256 ReLU MLP, using five runs per setting. Higher dropout levels consistently lowered training accuracy because dropping more activations reduces the effective capacity of the model during training. Validation and test accuracy showed only a very small change at $p = 0.1$, indicating that a mild amount of dropout has little effect on the model's behaviour. For $p = 0.2$ and especially $p = 0.5$, both validation and test accuracy decreased, reflecting reduced capacity rather than improved generalization. Overall, dropout did not provide performance gains, consistent with earlier observations that the model is not significantly overfitting.

4.3 Introducing early stopping in MLP

Since we added L1 and L2 regularization to our MLP implementation, we find it also interesting to explore the effect of early stopping on the model's accuracy. The early stopping level ranges from 0-4, which indicates the number of epochs (0-20) to run after there are no improvements. The mapping from early stopping level to number of epochs is:

```
self.level_to_patience = {0: None, 1: 3, 2: 5, 3: 10, 4: 20}
```

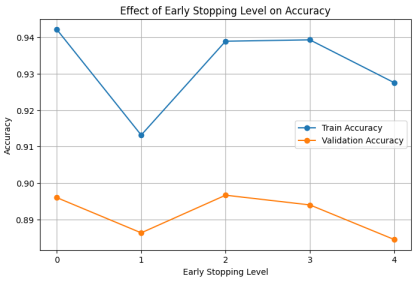

Figure 9: Effect of early stopping level on accuracy

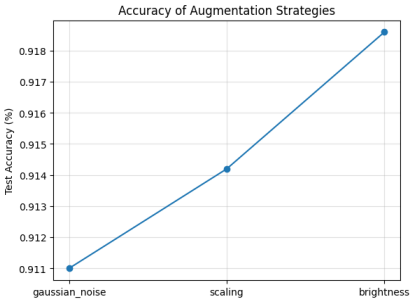

Figure 10: Accuracy for different augmentation techniques on CNN

From Figure 9, when there is no early stopping (level 0), the accuracy is high because the model is fully learned. Once early stopping is added, if patience is too small (level 1, 3 epochs), the model stops training before it has fully learned the patterns, and the accuracy drops significantly. The model reaches a sweet point at `the beginning of _stopping_level=2` (5 epochs). From this point, increasing patience starts hindering the accuracy since the model starts to overfit.

4.4 Finding the best data augmentation techniques for CNN

To identify the best augmentation technique for this task, three different methods were tested: scaling (80% and 120%), brightness adjustment (+- 40%), and Gaussian noise on the CNN from Experiment 6. Each model was trained for 3 epochs to see how the transformations affect the model performance in the early stages of learning and to save computational resources. The results (Figure 10), scaling achieved the highest accuracy, followed by brightness and noise.

5 Discussion and Conclusion

Our results show that Non-linear hidden layers with ReLU activation achieve best performance for our MLP model on the Fashion-MNIST dataset. Normalization improves accuracy by keeping input features on a consistent scale. On the other hand, regularization has limited effect on accuracy because the dataset is not highly prone to overfitting. Data augmentation also slightly lowers training accuracy, but can improve model robustness by exposing it to realistic variations. The CNN model surpass MLPs in image prediction by exploiting spatial correlations in images. In addition, like regularization, adding dropout and early stopping had little to none benefit.

6 Statement of Contributions

Daria Goptsii was responsible for data acquisition, Experiments 1–3, analysis of the dropout effect, and implementation of different initialization schemes. She also contributed to running multiple training configurations and summarizing their results. Giane Mayumi was responsible for experiments 6, 7, 8, and an extra experiment of augmentation techniques. Yixuan was responsible for experiments 4, 5, 9 and exploring the effect of early stopping on accuracy. Overall, all team members helped each other during implementation, discussed the results together, and contributed to the report. Given the limit of pages for the report, additional plots are available on the project's notebook.

References

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[2] Y. Zhao, G. Wang, C. Tang, C. Luo, W. Zeng, and Z. Zha, "A battle of network structures: An empirical study of cnn, transformer, and MLP," *CoRR*, vol. abs/2108.13002, 2021. [Online]. Available: https://arxiv.org/abs/2108.13002

[3] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *CoRR*, vol. abs/1708.07747, 2017. [Online]. Available: http://arxiv.org/abs/1708.07747

[4] O. Nocentini, J. Kim, M. Z. Bashir, and F. Cavallo, "Image classification using multiple convolutional neural networks on the fashion-mnist dataset," *Sensors*, vol. 22, no. 23, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/23/9544

[5] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 315–323. [Online]. Available: https://proceedings.mlr.press/v15/glorot11a/glorot11a.pdf

[6] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256. [Online]. Available: https://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1026–1034. [Online]. Available: https://arxiv.org/abs/1502.01852