COMP 551 Assignment 3 Report

Author Jinghan Zheng, Yunjie Xiao, Tailai Li

Abstract

In this mini-project, we implement a multilayer perceptron (MLP) from scratch and use it to classify images from the FashionMNIST dataset. The project involves building the full training pipeline, including forward propagation, backpropagation, gradient descent, and numerical evaluation. We then systematically study how network depth, activation functions, data normalization, and regularization affect classification accuracy. The experiments further extend to convolutional neural networks (CNNs), data augmentation with CNNs, and a pre-trained convolutional model, in which we freeze its feature extractor, and train a custom fully connected classifier on top. We also examined how layer width, CNN settings, and augmentation levels influence accuracy for creativity. We found that CNN achieves the best performance overall, while the pretrained ResNet18—though slightly worse than the CNN—still clearly outperforms all MLP variants.

Keywords: Multilayer Perceptron (MLP); FashionMNIST; Backpropagation; Activation Functions; Regularization; Normalization; Data Augmentation; Convolutional Neural Network (CNN); Pre-trained Models; Classification;

1 Introduction

Inspired by biological neural systems, neural networks have become fundamental to modern machine learning. In this project, we build a MLP from scratch and apply it to supervised image classification. Prior work includes a comparison of CNN architectures on FashionMNIST, where ResNet performed best [3], and a LeNet-5 variant that achieved over 98% accuracy on the same dataset [1].

The project systematically studies how architectural and optimization choices affect model performance. We first evaluate MLPs of different depths, then examine the impact of activation functions, normalization, and L1/L2 regularization. We also analyze the benefits and drawbacks of data augmentation. A simple CNN and CNN with data augmentation are implemented and compared with the MLPs. Finally, a pre-trained CNN is used by freezing its feature extractor and training a small classification head to contrast transfer learning with our earlier models. In the creativity experiments, we explored how hidden-layer width, CNN hyperparameters, and different augmentation strengths affect performance, identifying settings that help or hurt model accuracy.

The dataset we use, FashionMNIST [2], contains 70,000 images of 28×28 grayscale clothing items. Our key findings are that MLPs benefit mainly from a single nonlinear layer, while deeper MLPs, regularization, and heavy augmentation often hurt performance; in contrast, CNNs achieve the highest accuracy by leveraging spatial structure, and the pretrained ResNet18, though slightly below CNN, still outperforms all MLP variants. The creativity experiments further show that moderate model width, reasonable CNN hyperparameters, and light augmentation provide the best trade-off between accuracy and stability.

2 Dataset

Like we mentioned in the Introduction, FashionMNIST [2] serves as the dataset for all experiments in this project. It contains a total of 70,000 clothing images, with 60,000 allocated for training and 10,000 reserved for testing. Each sample is a small 28×28 grayscale picture that has been flattened into a 784-dimensional vector for use in the MLP models. Before training all models, the data is also normalized to stabilize gradient updates and improve convergence.

Class	0	1	2	3	4	5	6	7	8	9
Training Count	6000	6000	6000	6000	6000	6000	6000	6000	6000	6000
Test Count	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000

Table 1: Class distribution of the FashionMNIST training and test sets.

After performing an explanatory analysis to understand the data, we found that each class contributes exactly 6,000 images to the training set and 1,000 images to the test set, ensuring that every category is represented equally throughout the learning process. As a result, no class is overrepresented or underrepresented.

3 Results

3.1 Nonlinearity and Depth

Learning Rate	Val Acc (%)	Batch Size	Val Acc (%)
0.01	84.02	32	85.77
0.05	85.75	64	85.43
0.10	86.23	128	83.65
		256	83.72
		512	82.32

Table 2: Validation accuracy for different learning rates (2-layer MLP) and batch sizes (1-layer MLP).

Using a small grid search, we found that a learning rate of 0.10 produced the greatest validation accuracy (86.23%) among the tested values. For batch size, although smaller batches slightly improved accuracy, we selected 256 because it trains significantly faster while maintaining stable performance. These choices provide a practical balance between accuracy and computational efficiency, and were therefore used for all MLP experiments.

Model	Final Test Accuracy (%)
MLP (0 hidden layers)	83.54
MLP (1 hidden layer, 256 ReLU)	86.71
MLP (2 hidden layers, 256–256 ReLU)	85.56

Table 3: Test accuracy of the three MLP architectures in Task 3.1 after 10 training epochs.

Using the 10th-epoch results, adding nonlinear hidden layers clearly improves performance over the linear baseline: the model without hidden layers reaches 83.54%, while a single ReLU layer increases accuracy to 86.71%. The two-hidden-layer MLP performs slightly worse at 85.56%, suggesting that most of the gain comes from introducing nonlinearity, with additional depth offering limited benefit. This is consistent with class material: nonlinearity increases model expressivity, and small fluctuations, like the mild drop in the deeper model, are likely due to overfitting or limited training epochs.

3.2 Different Activation Functions

Activation Function	Final Test Accuracy (%)
ReLU (baseline)	85.56
tanh	82.28
Leaky-ReLU	84.65

Table 4: Comparison of activation functions for the 2-hidden-layer MLP (10 epochs).

Using the same two-hidden-layer MLP, we compared ReLU, tanh, and Leaky-ReLU using their 10th-epoch test accuracies. ReLU achieved the best performance (85.56%), followed by Leaky-ReLU (84.65%), while tanh performed noticeably worse at 82.28%. In principle, no activation is universally superior. ReLU variants typically train faster and avoid saturation(may be the reason why it is better here), whereas tanh can help when centered outputs benefit optimization. The underperformance of tanh here is likely due to slower convergence. Given that only 10 training epochs are provided, the result is expected.

3.3 L1 and L2 Regularization

Model / Regularization	Final Test Accuracy (%)
No regularization (ReLU baseline)	85.56
L1 ($\lambda = 10^{-5}$)	83.51
L2 ($\lambda = 10^{-4}$)	83.40

Table 5: Test accuracy of the baseline and L1/L2–regularized 2-layer MLPs (10 epochs).

Using the same two-hidden-layer MLP, we trained models with L1 and L2 regularization and compared them to the unregularized ReLU baseline. Unlike the previous case, both L1 (83.51%) and L2 (83.40%) regularization slightly reduced the final accuracy compared with the baseline (85.56%). This suggests that regularization was not beneficial here, likely because the model was not overfitting after only ten epochs. In this setting, the added penalties simply constrain the weights and slow learning, leading to lower accuracy rather than improved generalization. However, these results should be interpreted with caution, since the effect of regularization can depend strongly on the training duration and model size.

3.4 Unnormalized Images

Model Setting	**Final Test Accuracy (%)**
Normalized inputs	85.56
Unnormalized inputs	81.74

Table 6: Effect of input normalization on the 2-hidden-layer MLP (10 epochs).

Training the same two-hidden-layer MLP on unnormalized images caused a clear drop in performance, lowering the test accuracy from 85.56% to 81.74%. Without normalization, input values vary widely, so the gradient is unstable, and optimization becomes harder.

3.5 Data Augmentation

Setting	**Final Test Accuracy (%)**
Baseline (no augmentation)	85.56
With augmentation	78.72

Table 7: Effect of data augmentation on the 2-hidden-layer MLP (10 epochs).

Using the same two-hidden-layer MLP, training with data augmentation reduced the test accuracy from 85.56% to 78.72%. This decline is expected because augmentations such as flips and rotations distort pixel-level structure, which MLPs cannot handle well after flattening the images. While augmentation increases data diversity and can reduce overfitting, it distorts the structure, and it is harmful for models lacking spatial inductive biases, such as MLPs without convolutional layers. For example, horizontal flips or large rotations can distort clothing orientation in FashionMNIST, causing confusion for the model.

3.6 CNN

Learning Rate	**Train Acc (%)**	**Batch Size**	**Train Acc (%)**
0.001	94.06	32	93.68
0.005	92.15	64	93.59
0.010	88.31	128	92.78
0.100	10.02	256	91.38

Table 8: Validation of learning rates and batch sizes for the CNN.

To select hyperparameters for the CNN, we performed a small grid search over learning rates and batch sizes. A learning rate of 0.001 achieved the highest accuracy (94.06%), while larger learning rates quickly destabilized training. For batch size, 256 provided a good trade-off between accuracy (91.38%) and training speed. We therefore selected learning rate = 0.001 and batch size = 256 for the final CNN experiments.

Model	**Final Test Accuracy (%)**
Best MLP (1 hidden layer, 256 ReLU units)	86.71
CNN (2 conv layers + 1 FC layer with 256 units)	92.16

Table 9: Comparison of the best MLP and the CNN on FashionMNIST.

The CNN clearly outperforms all MLP variants. Test accuracy increases from about 86.7% for the best MLP to 92.2% for the CNN. This gain is expected, since convolution and pooling layers exploit the 2D spatial structure of images, while the MLP only sees flattened pixels.

3.7 CNN with Data Augmentation

Setting	Final Test Accuracy (%)
CNN (no augmentation)	92.16
CNN (with augmentation)	91.28

Table 10: Effect of data augmentation on the CNN (10 epochs).

Using the same CNN architecture, adding data augmentation (random horizontal flips and small rotations) produced a slight decrease in accuracy, from 92.16% to 91.28%. This drop is expected because the CNN is already robust to small spatial variations, and the FashionMNIST classes are relatively sensitive to orientation. Augmentation adds a bit of overhead due to the extra image transformations, so training takes slightly longer. Overall, augmentation makes training slower and performance slightly worse.

3.8 Pretrained Model

Head	Train Acc (%)
1-layer	81.51
2-layer	**82.19**
3-layer	80.43

FC head depth (2 epochs)

LR	Train Acc (%)
1×10^{-4}	**84.37**
5×10^{-4}	84.37
1×10^{-3}	84.32
5×10^{-3}	81.70

Learning rate search

Batch Size	Train Acc (%)
32	84.19
64	85.25
128	85.57
256	**86.04**

Batch size search

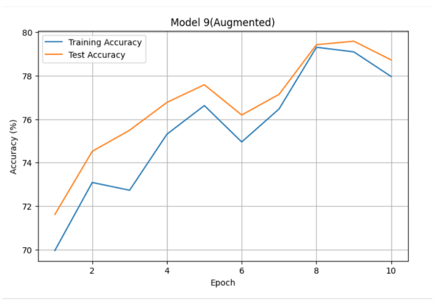
Table 11: Hyperparameter sweeps for the ResNet18 transfer-learning setup

Using the frozen ResNet18 backbone, the 2-layer FC head performed best, and a learning rate of 10^{-4} gave the most stable accuracy. Batch size 256 achieved the highest score and fastest training. Thus, we selected the 2-layer head, learning rate 10^{-4}, and batch size 256 for all transfer-learning experiments.

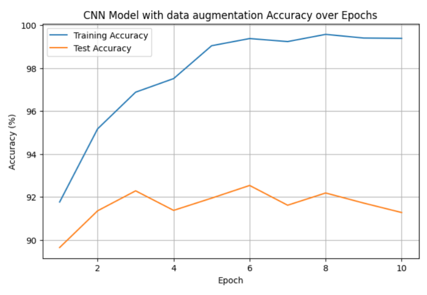
Model (with Q5 augmentation)	Final Test Accuracy (%)	Avg. Epoch Time (s)
MLP (2 hidden layers, ReLU)	78.72	5.2
CNN (2 conv + 1 FC layer)	91.28	11.9
ResNet18 transfer learning	85.83	100.6

Table 12: Comparison of accuracy and average runtime per epoch using the same augmentation pipeline.

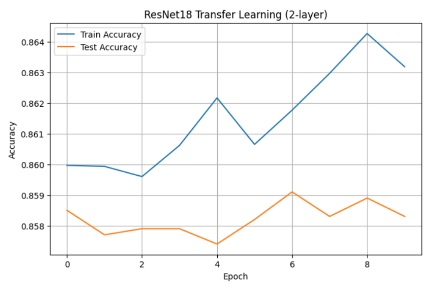
With data augmentation, the CNN achieves the best accuracy (91.28%), The pre-trained ResNet18 reaches 85.83%, better than augmented MLP (78.72%) but still below the custom CNN. In terms of running speed, the MLP is fastest, the CNN is moderately slower, and the ResNet18 model is by far the slowest due to its deep backbone.



(a) Model 1: MLP



(b) Model 2: CNN

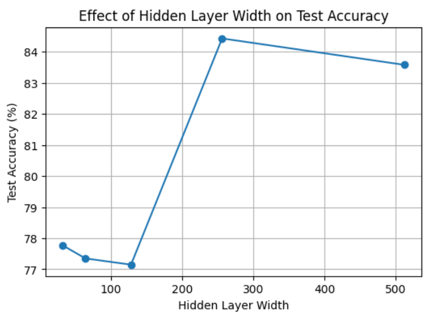


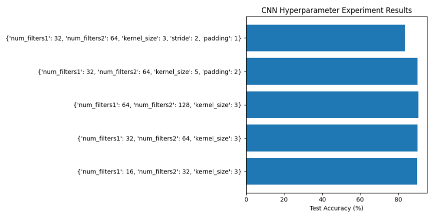
(c) Model 3: Transfer Learning

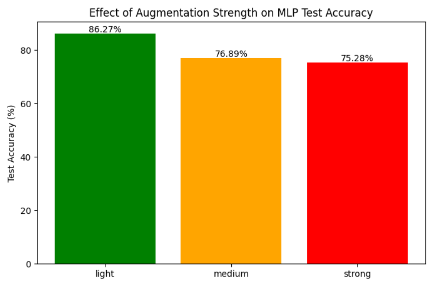
Figure 1: Training and test accuracy curves for the three models over 10 epochs.

The three plots above visualize how training and test accuracy evolve over the 10 training epochs for MLP, CNN, and pretrain data augmentation model. We wrote graphs for each model in the notebook, but we may not report them in every task to save space for the paper.

3.9 Creativity Experiments







1. Investigate the effect of width of hidden layers for MLP. Use one hidden layer MLP architecture and vary the number of neurons in the hidden layer [32, 64, 128, 256, 512]. It is observed that smaller hidden layers tend to underfit the data, resulting in low accuracy since the model lacks enough capacity to learn the patterns. Larger hidden layers have much better performance, but after a certain point (256 neurons), the accuracy goes down slightly due to overfitting. The optimal width here is 256 neurons, balancing capacity and generalization.

2. Investigate the effect of CNN's convolutional hyperparameters. Compare different kernel size1, number of filters1 and filters2, and train for one epoch with learning rate 1e-3. It is observed that with the same kernel size, increasing the number of filters improves accuracy as the model learns more features. With fixed number of filters, larger kernel sizes perform better as they capture more spatial features. The best hyperparameter combination is kernel size 5, number of filters1 64, and number of filters2 128.

3. Investigate the effect of different data augmentation levels on MLP. Three levels of data augmentation: light (random rotation 3°, ColorJitter(brightness=0.1)), medium (random rotation 10°, RandomCrop(28,padding=2)), and heavy (random rotation 15°, RandomCrop(28,padding=4), RandomHorizontalFlip(p=0.5)). It is observed that light augmentation performs best, while medium and heavy augmentation lead to worse performance due to distorting the images too much. The optimal augmentation level is light, which produces more data while preserving the characteristics.

4 Discussion and Conclusion

Our experiments show that architectural choices strongly shape performance on FashionMNIST. MLPs benefit mainly from adding a single nonlinear layer, while deeper MLPs, regularization, and heavy augmentation often hurt accuracy due to limited spatial awareness. CNNs leverage image structure much more effectively, achieving significantly higher accuracy than all MLP variants. When applying the same augmentation pipeline, the CNN experiences only a small accuracy drop, and transfer learning with a frozen ResNet18 provides a strong baseline but remains slightly below our custom CNN. The creativity experiments show that moderate model width and light augmentation help, while overly large models or strong augmentation hurt performance. Looking forward, fine-tuning deeper layers of the pretrained ResNet18, experimenting with stronger CNN architectures, and running longer training epochs are all good extensions to investigate. Further improvements could also come from more systematic hyperparameter tuning, exploring modern regularization methods such as dropout or label smoothing, and evaluating the models on harder image datasets to test generalization beyond FashionMNIST.

5 Statement of Contributions

Jinghan and Yunjie are responsible for organizing the data, implementing the model, and running the experiments. Tailai is responsible for cleaning the data, gathering the results, and writing the report.

References

[1] Mohammed Kayed, Ahmed Anter, and Hadeer Mohamed. Classification of garments from fashion mnist dataset using cnn lenet-5 architecture. In *2020 International Conference on Innovative Trends in Communication and Computer Engineering (ITCE)*, pages 238–243, 2020.

[2] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms, 2017.

[3] Yue Zhang. Evaluation of cnn models with fashion mnist data. In *Proceedings of the Conference on Computer Vision Applications*, 2019. Venue not specified in original source.