COMP551-A3-Report

Jawdat Al-Jabi, Tal Smith, Steven Thao

November 2025

Abstract

This project investigates how model architecture and training choices affect performance on the FashionMNIST classification task. Beginning with simple multilayer perceptrons (MLPs), we examine the impact of network depth, activation functions, regularization, and input normalization. Our experiments show that deeper networks and nonlinearities (particularly Leaky-ReLU) consistently improve accuracy, while L1/L2 regularization provides little benefit when overfitting is not present. We also evaluate the role of data augmentation and find that common transformations such as rotations and translations reduce test accuracy, likely because they produce images that deviate from the clean, centered format of the test set. Transitioning to convolutional models yields substantial gains, with a custom CNN far outperforming the vanilla MLP. Finally, we test a pretrained ResNet and observe limited performance due to the mismatch between its CIFAR-10 training set and the FashionMNIST dataset. Overall, the results highlight the importance of architecture choice and data preprocessing when designing image classification pipelines.

1 Introduction

The goal of this project was to explore how different neural network architectures and training strategies perform on FashionMNIST, a widely used benchmark for image classification. Although the dataset is relatively simple (consisting of centered 28×28 grayscale images of clothing items) it provides a useful testing set for studying the effects of model depth, nonlinear activations, regularization, normalization, data augmentation, and convolutional structure on classification performance. By progressively moving from shallow multilayer perceptrons (MLPs) to deeper networks, experimenting with various activation functions, and incorporating both regularization and data preprocessing techniques, we aimed to understand which components meaningfully improve generalization and which can unexpectedly hinder it. We then compared these models against a small custom convolutional neural network (CNN) and a pretrained ResNet to examine the benefits of architectures that explicitly leverage spatial information. This systematic investigation allowed us to highlight the strengths and limitations of different design choices when working with image data.

2 Dataset

We used FashionMNIST for this project. This dataset is analogous to the standard MNIST dataset, but instead of being hand-drawn digits, it's a collection of pieces of clothing/fashion. Just like MNIST, fashion-MNIST is comprised of 28x28 greyscale images with integer pixel values ranging from 0 to 255.

After an early pre-processing step of diving the pixel values by 255, we looked at the mean and standard deviation to understand whether or not the pixel values were difficult to distinguish or not. We observed a mean of 0.2860 and a standard deviation of 0.3530, which represents a fairly large range.

Furthermore, we have discovered that the 10 classes had a very similar number of instances, meaning that class imbalance was not going to be an issue given proper randomization.

3 Results

3.1 Basic experimentation with vanilla MLPs

Since the images are 28x28, we expect an input size of 784 and an output of 10 classes (i.e. softmax). We started with 3 vanilla neural networks:

- No hidden layers
- 1 256-units hidden layer + ReLU
- 2 256-units hidden layers + ReLU

Respectively, we recorded the following metrics:

- Training accuracy: 0.6255, Testing accuracy: 0.6173
- Training accuracy: 0.6538, Testing accuracy: 0.6492
- Training accuracy: 0.8033, Testing accuracy: 0.7818

We can observe a steady increase in both the training and testing accuracies. We believe that this is a direct correlation with the use of nonlinearities and deeper networks. Since FashionMNIST is a non linearly-separable dataset, it makes sense that adding nonlinearity to our model makes it perform better. Furthermore, adding layers means increasing the depth of our network and we understand that increasing model depth allows for more nonlinear-function composition, which increases the expressiveness of the MLP, which implies a better classification.

3.2 Comparing 2 hidden layers with different activation functions

Using the last model above, the 2-hidden-layer model, we compared the ReLU-based model to those with tanh and leaky-ReLU.

Activation function	**Test Accuracy**
w/ ReLU (from 3.1)	0.7818
w/ Tanh	0.5598
w/ Leaky ReLU ($\alpha = 0.1$)	0.8092

Table 1: Test accuracies for various activation functions.

Interestingly, the Leaky-ReLU-based model performed slightly better than the ReLU one. Although we were intrigued by this result, we later understood that the benefit of Leaky-ReLU, being that it prevents vanishing gradients, is probably why it was able to converge better than its simpler counterpart, which can often get stuck in 0-gradient pitfalls.

3.3 Implementing L1 and L2 regularization

We independently implemented L1 and L2 regularization to the model above (2 hidden layers) and we obtained the following results:

Model	**Test Accuracy**
L1 Regularization	0.7700
L2 Regularization	0.7693

Table 2: Test accuracies for L1 and L2 regularization.

Using the same number of epochs for training (10), we observed similar (if not, slightly worse) results when compared to the base 2-layer model without regularization. This may happen because our model does not overfit the data, in which case regularization actually hinders its performance. In the event that our model would be complex enough to overfit the data, then regularization (of any kind) will generally improve the testing accuracy of the model.

3.4 Training with unnormalized images

Normalization of data is important in the context of model training, because it prevents the model from have any dominating gradients and all features work with the "same scale" of inputs. That being said, we tried training the model with an unnormalized version of FashionMNIST (i.e. the default data).

Normalization of data	**Test Accuracy**
Without normalization	0.7476
With normalization (from 3.1)	0.7818

Table 3: Test accuracies when using normalized and unnormalized data in training.

We can see that normalizing the training data does improve the test accuracy of the model.

3.5 Implementing Data Augmentation

Data Augmentation increases the dataset size by introducing some linear transformations to the current dataset. This increases the models' robustness to noise and should theoretically allow it to perform better at test-time.

This snippet of code represents the random transformations we imposed on the training data:

```
transforms.RandomHorizontalFlip(p=0.5),
transforms.RandomRotation(10),
transforms.RandomAffine(degrees=0,␣translate=(0.1,␣0.1)),
```

Which means we applied some random flips, rotations, and translation transformations to our data. We obtained the following results after using this augmented data:

Data Augmentation	**Test Accuracy**
Without augmentation (from 3.3)	0.7693
With augmentation	0.6873

Table 4: Test accuracies when using augmented data in training.

This was our most interesting result. We see a significant dip in the testing accuracy after augmenting our training data. After further thought, however, we came to the conclusion that augmenting the training set might have actually deprived the model of test-set-like data. The FashionMNIST dataset is made up of completely "straight" and centered fashion items. This means that augmenting the data by rotating and translating the images might have "misled" the model for what the fashion items in the test set actually look like. As a result, the model gets trained with a smaller sample of the test-like images, which could have negatively impacted the testing accuracy.

3.6 Implementing a Convolutional Neural Network

Since we are working with images, it makes more sense to use a model that is better suited for the task. Convolutional Neural Networks are made to process data with spatial relationships, like images. We have implemented a very simple CNN model with PyTorch:

- A conv. layer with a 3x3 kernel, 1 padding, ReLU out:
  ((28x28 input) $\rightarrow$ 32 channels $\times$ (28x28))

- A 2x downsampling with maxpooling:
  (32 channels $\times$ (28x28) $\rightarrow$ 32 channels $\times$ (14x14))
- Another conv. layer with a 3x3 kernel, 1 padding, ReLU out:
  (32 channels $\times$ (14x14) $\rightarrow$ 64 channels $\times$ (14x14))
- Another 2x downsampling with maxpooling:
  (64 channels $\times$ (14x14) $\rightarrow$ 64 channels $\times$ (7x7))
- Flattened to 3136 ($7 \times 7 \times 64$)
- A FC layer with ReLU output
- A dropout layer with $p = 0.5$
- Another FC layer that outputs to 10 classes

Model type	**Test Accuracy**
Vanilla MLP (from 3.1)	0.7818
CNN	0.9196

Table 5: Test accuracies between vanilla MLP and CNN.

We observe a significant improvement in the testing accuracy when switching to a CNN, and this is expected, because we know that CNNs perform better on images when compared head to head with MLPs.

3.7 CNN with Data Augmentation

Now we introduce data augmentation when training the CNN. We used the same model as above, and we obtained the following results after applying the same data augmentation operations as above:

Data Augmentation	**Test Accuracy**
Without augmentation (from 3.6)	0.9196
With augmentation	0.9033

Table 6: Test accuracies in CNN when using augmented data in training.

Similarly to 3.5, we observe an unexpected dip in the performance of the model when introducing augmented data. Again, this may come from testing with data that does not resemble the training data as much as pre-augmentation.

3.8 Using ResNet, a pretrained CNN

ResNet is a CNN architecture that was proposed in 2015 by a team of researchers from Microsoft. It was a very powerful architecture that was able to perform really well on CIFAR-10, a dataset of 60000 32x32 color images. We used ResNet in this project by freezing the kernel weights and only training the FC layers with augmented data. We ended up with the following results:

Model	**Test Accuracy**
Vanilla MLP (from 3.5)	0.6873
CNN (from 3.6)	0.9033
ResNet	0.7018

Table 7: Test accuracies through various models (with data augmentation).

This may come as a surprise at first glance, but we realize that ResNet's kernels were trained on CIFAR-10, whose classes do not match FashionMNIST's at all. That being said, we may now understand the poorer performance compared to the CNN trained from scratch on our dataset.

4 Discussion and Conclusion

Below are some key takeaways from this project:

- The deeper the Neural Network, the better (more expressive) it is at test-time.
- Introducing nonlinearity (especially Leaky-ReLU) further helps with test metrics.
- Regularization does not improve a model that is not overfitting already.
- Data normalization is very important when training neural networks.
- Data augmentation does not improve the performance of the model if the testing set does not really vary (all centered and straight).
- Convolutional Neural Networks are, in fact, much better at classifying images than vanilla neural networks.
- ResNet (or any pretrained model), although generally very powerful, does not work on images it has not really seen before.

5 Statement of Contributions

Part	**Contributor**
Part 1	Jawdat Al-Jabi
Part 2	Steven Thao
Part 3	Tal Smith and Jawdat Al-Jabi
Code Review	All
Project Write-Up	All

Table 8: Project Contributions