# COMP 551 ASSIGNMENT 2

Antony Zhao, Naiwrita Sen, Shichen Li

October 31, 2023

## Abstract

*Neural networks are powerful tools to model complex and large datasets. In this project, we implemented a Multilayer Perceptron from scratch and used it to classify image data from the FashionMNIST Dataset and CIFAR-10 dataset. We conducted numerous experiments on the network by adjusting hidden layers and respective units, activation functions and the application of regularization. Our findings revealed our model performed best with 1 hidden layer, 128 units, employing Xavier initialization and ReLU activation. This setup attained an accuracy of 73% on our test set. We further implemented a convolutional neural network (CNN) with PyTorch and compared the performance of CNN and MLP on our datasets. Our results showed that results using CNN had a higher accuracy than the MLP model on both our datasets.*

## Introduction

This project aims to develop a Multilayer Perceptron and assess its performance on two different datasets. Our first dataset is the FashionMNIST dataset containing fashion products of 10 categories. In our experiments, we investigated various activation functions, the number of hidden layers, layer sizes, and hyper-parameters such as the learning rates, and L1 and L2 regularization. Our optimal model achieved an accuracy of 73% using 1 hidden layer, 128 units, Xavier initialization and ReLU activation. This might be because increasing the number of layers increases the accuracy but eventually, starts overfitting. Our investigation also revealed that using L2 regularization and no regularization had a similar effect on our dataset. We further experiment with convolutional neural network to compare the performance of CNN and MLP on our dataset.

For our second dataset, we use CIFAR10 dataset which consists of color images in 10 different classes, such as cats, birds, ships, etc. We evaluate the performance of MLP and CNN models on the dataset and our results show that CNN tends to have a better performance for image classification. We also noticed that by using the SGD optimizer and raising the momentum from zero, our accuracy improved. However, after a certain point, further increasing the momentum leads to a decline in accuracy which might be due to overfitting. As part of our investigation, we tested our dataset using a pre-trained ResNeXt model which achieved an accuracy of 98.56% on CIFAR-10 dataset, the highest accuracy of all three models tested in this study.

## Datasets

For our project, we utilized two primary datasets: the Fashion-MNIST dataset 1 and the CIFAR-10 dataset 2. Both datasets were loaded using the PyTorch framework.

The Fashion-MNIST dataset is a new standard dataset for computer vision and deep learning 3. It encompasses 70,000 labeled fashion images, with each image having a resolution of 28x28 pixels, amounting to 784 pixels per image. The dataset is partitioned by default into a training set of 60,000 images and a test set of 10,000 images. The images are classified into ten distinct categories: 'T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneaker', 'Bag', and 'Ankle boot'. These categories are represented numerically by integers from 0 to 9 for ease of reference.

The CIFAR-10 dataset contains 60,000 images that fall into ten unique categories, predominantly focusing on modes of transportation and animals: 'plane', 'car', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', and 'truck'. It's worth noting that these categories are mutually exclusive, ensuring each image is distinctly categorized. The default split of this dataset consists of 50,000 training images and 10,000 test images, a structure we adhered to throughout our experiments.

# Results

- **1. Adding different initialization**

In our first experiment, we ran our model with different weight initialization - Zero, Uniform[-1,1], Gaussian N(0,1), Xavier, and Kaiming. We used 1 hidden layer with 128 units and ReLU activation for this experiment. The loss versus epochs plot with different weight initialization is shown in Appendix Figure 1. By observing the training loss and test accuracy for each of the models, we concluded that using Xavier gave us the optimal performance with an accuracy of 74.06 %, as indicated in Table 1. Both Xavier and Kaiming initialization worked well will ReLU activation while others performed poorly on our dataset. Basic weight initialization methods, such as zero initialization and uniform initialization, have shortcomings such as the vanishing gradient problems. Better initialization methods, such as Xavier, have been developed to address these issues and our results reflect that very well.

| Initializations | Test Accuracy |
|---|---|
| Zero | 10 % |
| Uniform | 11.43 % |
| Gaussiam | 10.18 % |
| Xavier | 74.06 % |
| Kaiming | 54.98 % |

Table 1: Effect of Varying Initialization Methods on Accuracy

- **2. Adding different number of layers**

In this experiment, we ran three iterations of our model: 0 hidden layers, 1 hidden layer, and 2 hidden layers. The loss versus epochs plot for three different setups are shown in Appendix Figure 2. The MLPs with hidden layers used ReLU activation, and had a Softmax layer at the end. Each hidden layer had 128 units. We first monitored the training accuracy against the number of iterations and chose the optimal epochs after which training accuracy started to become constant. Then, we performed a hyperparameter tuning on each model to find the best learning rate. As shown in Table 2, the model with 1 hidden layer performed the best among the three and achieved an accuracy of 73 %. The model with 0 hidden layers performed the worst on our dataset. This might be because using softmax regression, with no hidden layers, does not accurately capture the complexity of our dataset. Therefore, adding more layers applies non-linearity to our model and captures complex features of our images such as edges. As a result, we get a better fit model. However, adding more layers eventually decreases the accuracy which might be due to the model overfitting.

| Hidden layers | Test Accuracy |
|---|---|
| 0 | 60 % |
| 1 | 73 % |
| 2 | 54 % |

Table 2: Effect of Varying Number of Layers on Test Accuracy

- **3. Adding different activations**

Using the same model as above with 2 hidden layers and 128 units, we now investigate the effects of varying activation functions on our models. We use two extra functions: Softplus and Sigmoid. The loss versus epochs plot for different activation functions is shown in Appendix Figure 3. The table of the effect of different activation functions on test accuracy is shown in Table 3. Our results reveal that ReLU function achieves the highest accuracy of 54% when compared to Sigmoid and Softplus functions. Sigmoid function performed very poorly on our dataset. This might be because of a vanishing gradient problem with Sigmoid functions. Also, Sigmoid functions are not zero-centred which can lead to undesirable asymmetry in the gradient updates for the weights. Therefore, activation functions such as Softmax, ReLU, and hyperbolic tangent might be more suitable for multi-class image classification.

| Activation Function | Test Accuracy |
|---|---|
| ReLU | 54 % |
| Logistic | 10 % |
| Softplus | 37 % |

Table 3: Effect of Activation Function on Test Accuracy

- **4. Adding different regularization (L1, L2)**

  In this experiment, we tested our model with different regularization: no regularization, L1 and L2 regularization. The results are shown below in Table 4. The loss versus epochs plot for each regularization method is as shown in Appendix Figure 4. We observe that the model with no regularization achieved the best score even though we expected L2 regularization to have the highest accuracy. One reason for this might be that neural networks have more than one minimum and with the learning rate and activation functions used, we might be converging to a local optimum when using L2 regularization. Further investigation needs to be conducted to test the effect of regularization and eliminate the possibility of an anomaly in the results.

| Regularization | Test Accuracy |
|---|---|
| None | 60 % |
| L1 | 10 % |
| L2 | 57 % |

Table 4: Effect of Adding Regularization on Test Accuracy

- **5. Adding unormalized inputs**

  For this experiment, we use 2 hidden layers with 128 units and ReLU activation, as before. However, our model was trained with unnormalized images. Therefore, the range of each value in the unit vector is between 0 and 255. As shown in Table 5, test accuracy with unnormalized images was lower than the normalized images, as expected. This is because if not normalized, some weights will have to adjust with significantly larger magnitudes than others. This can result in slow convergence or the model getting stuck in poor local minima, thus, leading to poor performance on the test set.

| Input Images | Test Accuracy |
|---|---|
| Normalized | 57 % |
| Unnormalized | 44 % |

Table 5: Performance of MLP vs CNN on FashionMNIST Dataset

- **6. Our MLP vs CNN (PyTorch) with dataset FashionMNIST**

  For this section, we implemented a CNN model and compared the performance of MLP and CNN models on our FashionMNIST dataset. The loss versus epochs plot for both models is shown in Appendix Figure 5. We observe that, as shown in Table 6, the CNN model gives a higher accuracy of 83.88%. This might be because of its ability to exploit spatial hierarchies and use shared weights to reduce the number of parameters used by the CNN model.

| Model | Test Accuracy |
|---|---|
| MLP | 74.43 % |
| CNN | 83.88 % |

Table 6: Performance of MLP vs CNN on FashionMNIST Dataset

- **7. Our MLP vs CNN (PyTorch) with dataset CIFAR-10**

The loss versus epochs plot for both models is shown in Appendix Figure 6. Using the CIFAR-10 dataset, even though both of the test accuracies decreased (in Table 7), the CNN model consistently outperformed the MLP model. This discrepancy is likely due to the inherent differences in how each model processes images. While the MLP model perceives an image as a flat vector of pixels, failing to capture the inherent structure and patterns within, the CNN model is specifically designed to address the complexities of image data. Consequently, CNNs tend to excel over MLPs when working with image datasets like CIFAR-10.

| Model | Test Accuracy |
|---|---|
| MLP | 34.37 % |
| CNN | 41.77 % |

Table 7: Performance of MLP vs CNN on FashionMNIST Dataset

- **8. Adding different optimizer (SGD and ADAM) on CNN (PyTorch)**

In our initial experiment, we evaluated the performance of a CNN on the CIFAR-10 dataset using PyTorch and two different optimizers: SGD and ADAM. The loss versus epochs plots for different momentum parameters are as shown in Appendix Figure 7 and 8.

For the SGD optimizer, we experimented with various momentum values: 0, 0.2, 0.4, 0.6, 0.8, and 1.0. Among these, a momentum of 0.3 yielded the highest accuracy, clocking in at 0.4684.

For the ADAM optimizer, we tested different learning rates: 0.0001, 0.001, 0.01, and 0.1. The optimal learning rate is 0.0001 achieving an accuracy of 0.4542. In a comprehensive comparison, the SGD optimizer performed better than the ADAM optimizer for CNN (PyTorch).

- **9. Pre-trained model (ResNet), and adding different number of layers on it.**

We employed a pre-trained ResNeXt50_32x4d model. Three distinct linear layer architectures were explored, each based on the feature layers of the ResNeXt50 model. These fully connected architectures consist of 1, 2, and 3 layers respectively. The exact dimensions for each of these configurations are detailed in Table 8.

| Number of layers | Dimension of each layer |
|---|---|
| 1 | $2048\times512$ |
| 2 | $2048\times256, 256\times10$ |
| 3 | $2048\times512, 512\times256, 256\times10$ |

Table 8: Part 3.9 Dimension of the linear layers

Initially, we evaluated our enhanced concatenated model with all feature layers set to non-trainable, including both convolutional and other layers. After 20 epochs, this configuration achieved an accuracy of approximately 45%. However, given the unique architecture of the ResNeXt model, we proceeded to freeze only the convolutional layers, leaving other layers like batch normalization trainable. Using the same linear layer structures, the model exhibited a significant improvement in accuracy after 20 epochs. A comparison of this accuracy with results from previous models is presented in Table 9. Note that the time for each model is under different number of epochs for them to reach best accuracy. The loss versus epochs graph is as shown in Appendix Figure 9. It is obvious that the pre-trained model if only freeze convolutional layers, performs better than all previous models in terms of accuracy. However, as of training time, the ResNeXt takes significantly longer, this is possibly due to the fact that ResNeXt has a way deeper and wider structure than what we were implementing for the MLP and Pytorch CNN model. Based on our testing data, the model with only one single fully connected layer performed the best. It achieved an accuracy of 98.56% in 20 epochs consuming around 5min and 58sec. In theory, a smaller number of layers should lead to a faster training time if the dimensions of the layers are similar, but for our test, the time taken is actually longer. This can be explained by the fact that the training

time can vary and since we let the %timeit function only run the code once, the time count might have a large variance.

| Model | Accuracy | Training time | Number of Epochs |
|---|---|---|---|
| MLP | 34.37% | 58 sec | 500 |
| Pytorch CNN | 42.77% | $\sim 6min11sec$ | 300 |
| ResNeXt with 1 linear layer | 98.56% | $\sim 5min58sec$ | 20 |
| ResNeXt with 2 linear layer | 98.21% | $\sim 5min45sec$ | 20 |
| ResNeXt with 3 linear layer | 98.07% | $\sim 5min52sec$ | 20 |

Table 9: Part 3.9 Accuracy and Training Time

## Discussion and Conclusion

Multilayer Perceptron (MLP) models have gained widespread popularity in image classification tasks within the realm of machine learning4. In our recent project, we trained various configurations of the MLP model using the Fashion-MNIST dataset 1 as well as CIFAR-10. Our implementation offers flexibility in terms of initialization patterns, the number of hidden layers, activation functions, regularization techniques, and the option to normalize the input dataset. Our research primarily revolved around understanding the impact of these variations on the model's accuracy.

In our experiments, the best-performing MLP configuration utilized the "Xavier" initialization pattern, incorporated 1 hidden layer, employed the "ReLU" activation function, used L2 regularization, and normalized its inputs. Furthermore, increasing the number of training images lead to better performance of the MLP model. However, even with this optimal configuration, our MLP fell short in accuracy when compared to a Convolutional Neural Network (CNN) developed using PyTorch. This observation held true for both the Fashion-MNIST and CIFAR-10 datasets. For instance, with the CIFAR-10 dataset, our CNN achieved an accuracy of 0.4295, while the MLP only reached 0.3437. This stark contrast underscores the superior performance of CNN models in image classification tasks.

Diving deeper into the performance metrics of the CNN, we found that using the "SGD" optimizer yielded better results than the "ADAM" optimizer. Remarkably, the "SGD" optimizer with a momentum setting of 0 proved to be the most effective.

Moving forward, a pivotal area of our research will be to explore the underlying reasons for the pronounced advantage that CNN models have over MLP models in image classification endeavours.

## Statement of Contributions

Each team member significantly contributed across all phases of the project, ensuring a holistic involvement in preprocessing, modeling, and documentation.

# Appendix

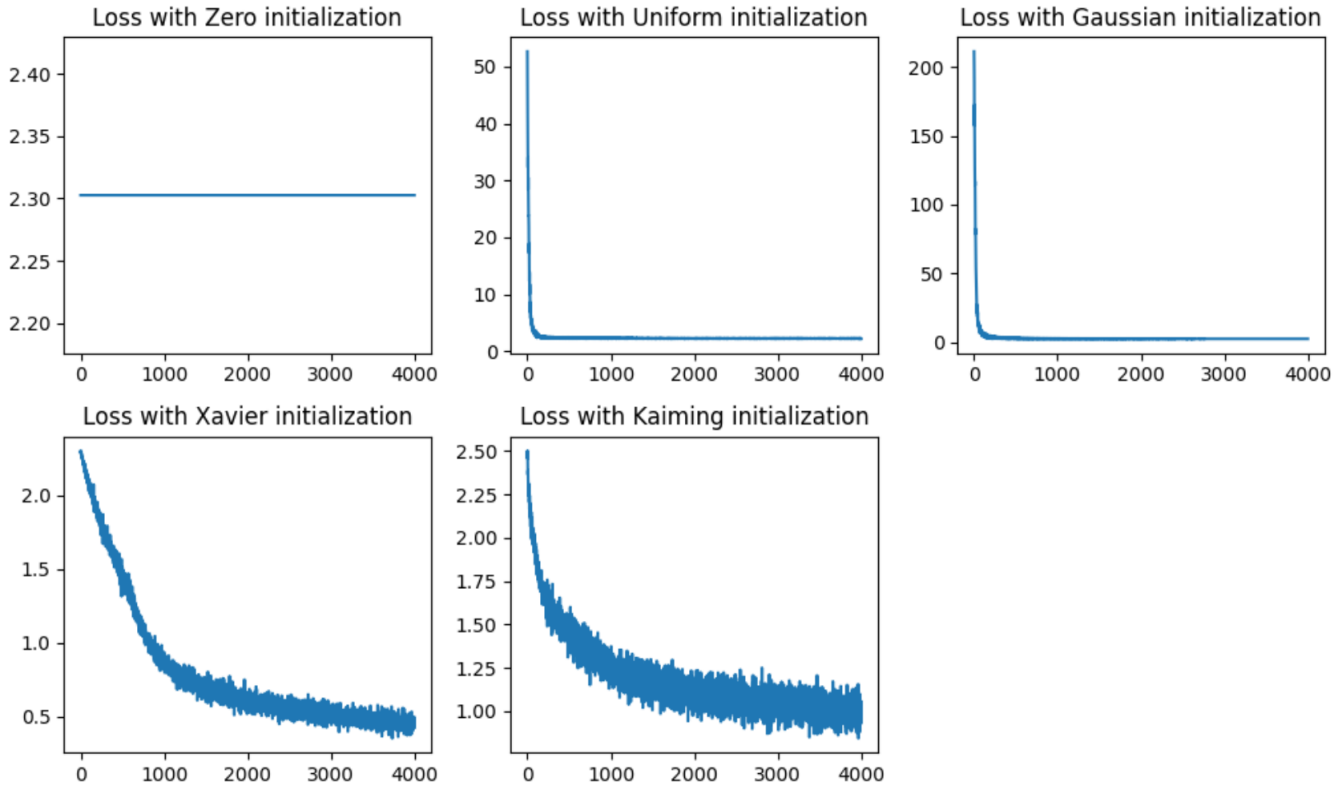


Figure 1: Training Loss With Different Weight Initialization

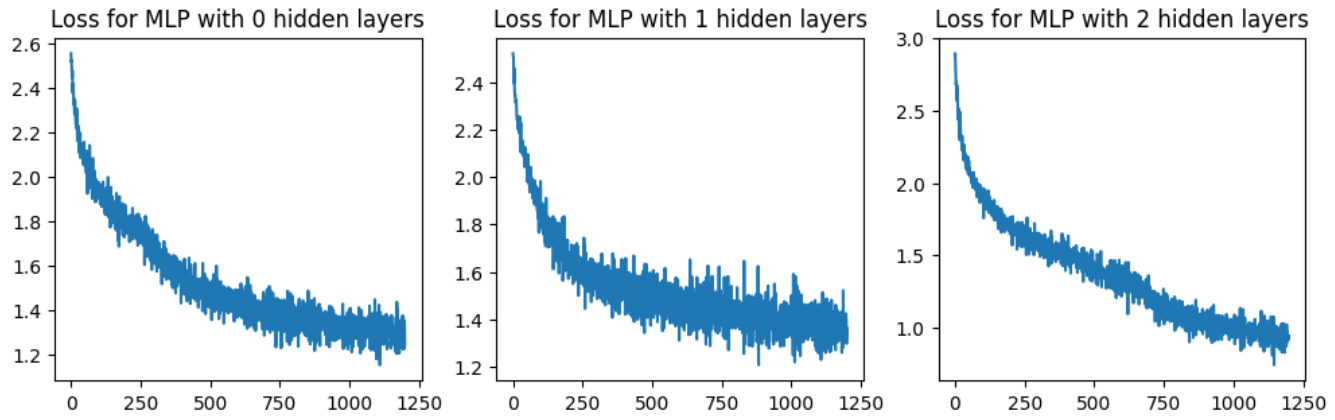


Figure 2: Training Loss With Different Number of Layers

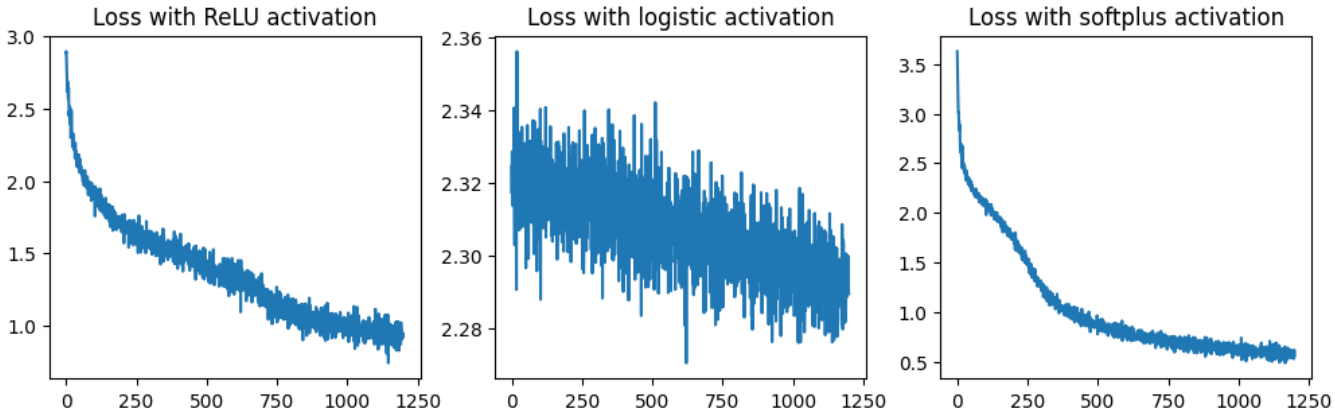


Figure 3: Training Loss With Different Activation Functions

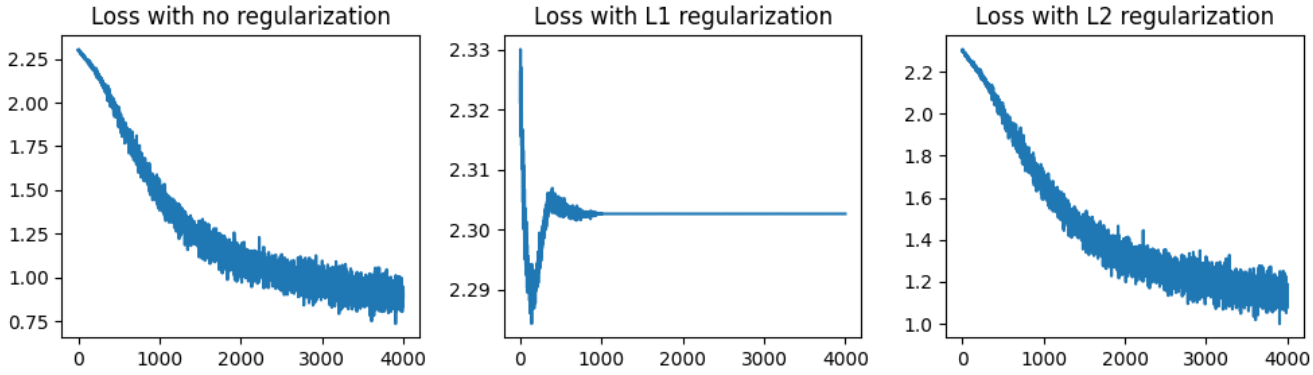


Figure 4: Loss with different regularization

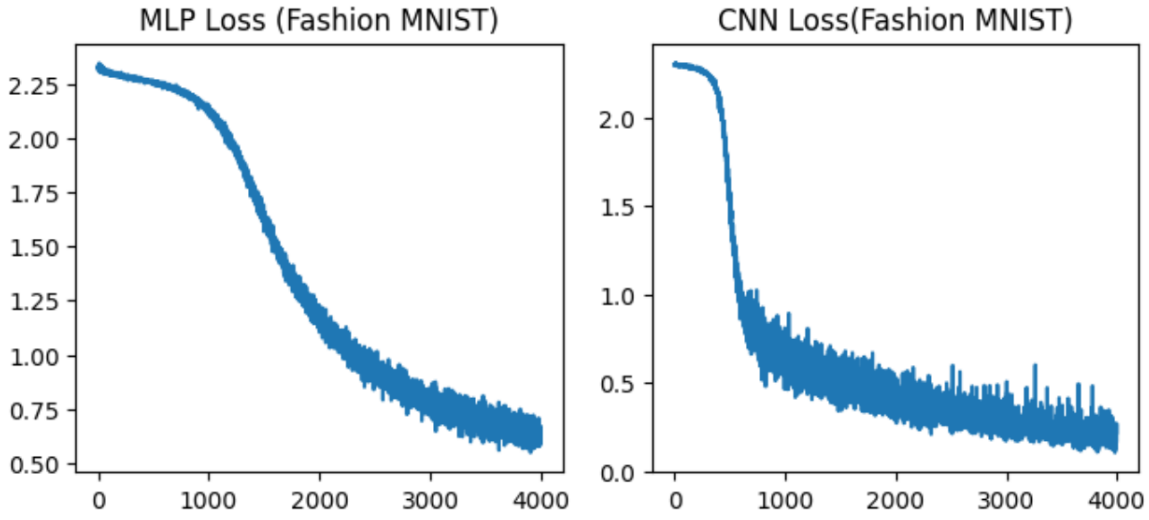


Figure 5: Our MLP vs. CNN (Dataset FashionMNIST)

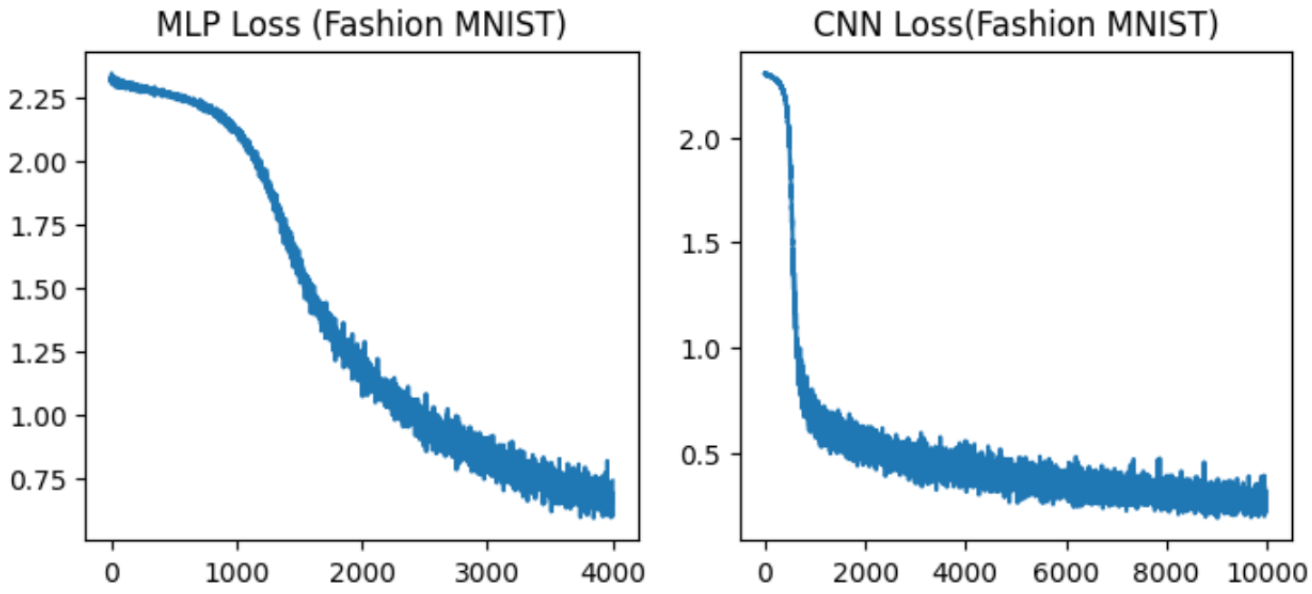


Figure 6: Our MLP vs. CNN (Dataset CIFAR-10)

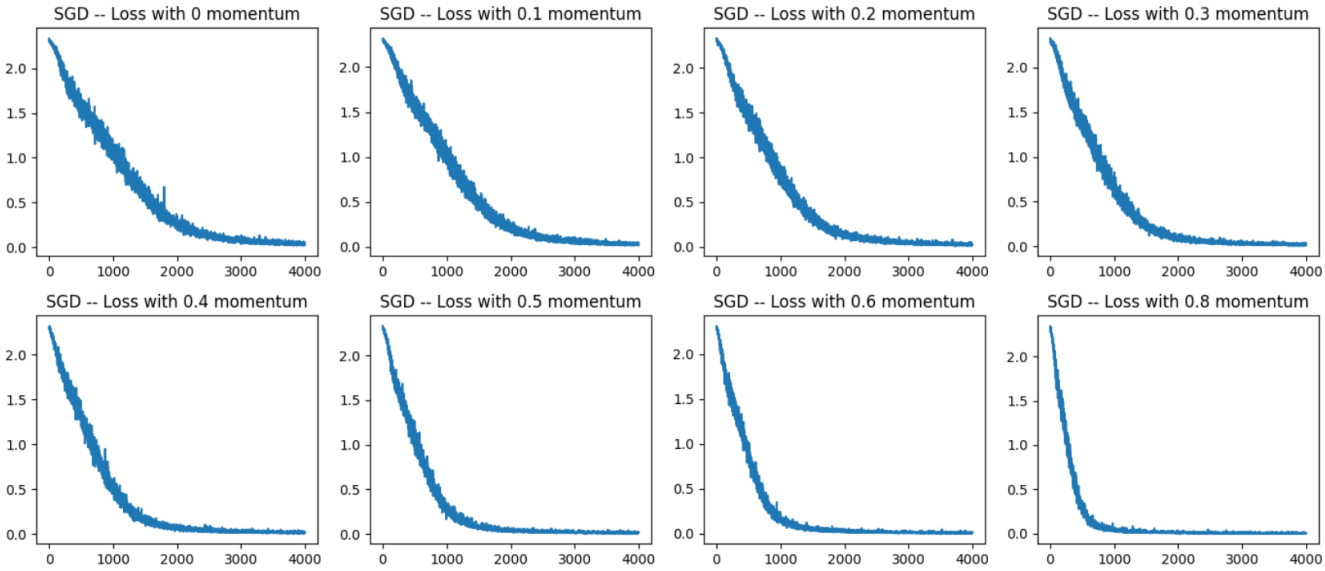


Figure 7: Training Loss With SGD Using Different Momentum Parameter

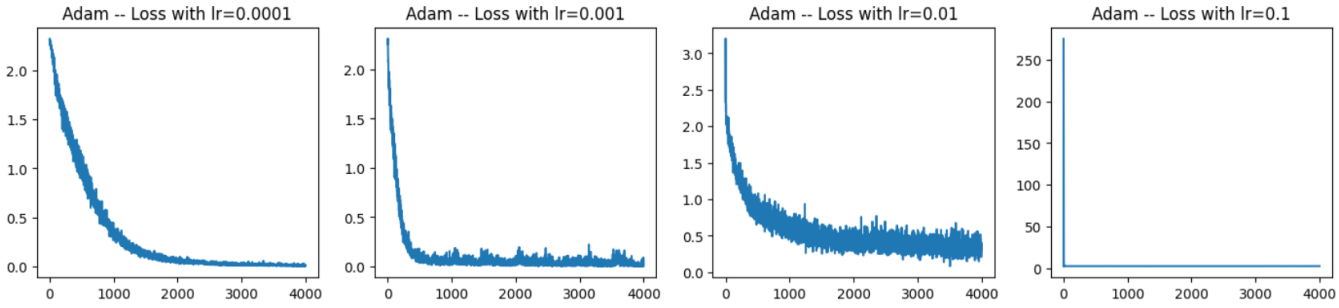


Figure 8: Training Loss With Adam using Different Momentum Parameter

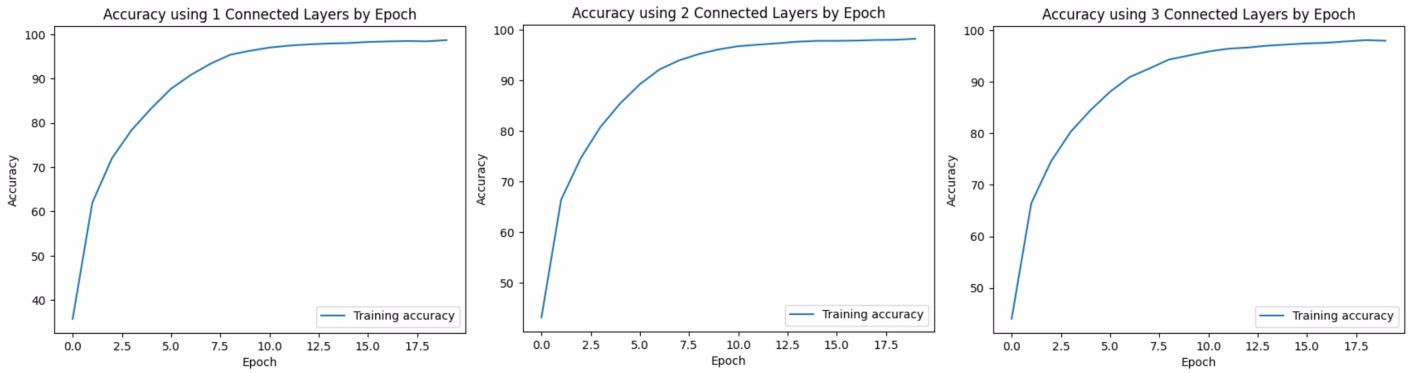


Figure 9: Accuracy using different Connected Layers by Epoch

## References

1. Zalandoresearch. (n.d.). Zalandoresearch/fashion-mnist: A mnist-like fashion product database. benchmark. GitHub. Retrieved April 4, 2022, from https://github.com/zalandoresearch/fashion-mnist

2. Learning Multiple Layers of Features from Tiny Images, Alex Krizhevsky, 2009.

3. Brownlee, Jason. "Deep Learning CNN for Fashion-Mnist Clothing Classification." Deep Learning CNN for Fashion-MNIST Clothing Classification, Jason Brownlee, 27 Aug. 2020, https://machinelearningmastery.com/to-develop-a-cnn-from-scratch-for-fashion-mnist-clothing-classification/.

4. Khalid, Irfan Alghani. "Multilayer Perceptron for Image Classification." Medium, Irfan Alghani Khalid, 10 May 2021, https://towardsdatascience.com/multilayer-perceptron-for-image-classification-5c1f25738935.

## Acronyms

**CNN** Convolutional Neural Network.

**MLP** Multilayer Perceptron.

**SGD** Stochastic Gradient Descent.