Mini-Project 2:

Classification of Image Data with Multilayer Perceptrons and Convolutional Neural Networks

Lily Gostovic - 260958505
Mohammad Abdullah - 260980866
Justin Novick - 260965106

A report describing the findings of
Mini-Project 2 of COMP551

COMP551: Applied Machine Learning
McGill University
Montreal, Canada
October 31, 2023

1 Abstract

This project was focused on designing an MLP and a CNN and training them on 2 different image datasets, namely Fashion MNIST and CIFAR-10. Experiments were ran on both different models for tuning the hyper parameters which led to us using a learning rate of 0.001 for the MLP and a learning rate of 0.01 for the CNN to optimize computational time and see good convergence. Mini batch sizes of 256 were used throughout for all the experiments which is a normal batch size and widely used. Gradients were optimized using Stochastic Gradient Descent, and viable convergence was shown. Overall, the best performing MLP was the one having weights initialized through the "Kaiming" distribution, having 2 hidden layers (128 neurons in each layer), ReLu as an activation function, weak regularization and a batch size of 256. Each MLP model was trained for 300 epochs which was deemed sufficient due to the time constraints and computational resources at hand. Regardless, however, CNNs would continuously outperform the MLPs for both datasets, specifically CIFAR 10, and therefore could be considered the optimal way for training a model for these datasets.

2 Introduction

An implementation of an MLP was constructed using automatic gradient computation which was put to the test against CNNs created using the PyTorch library. Despite rigorous hyper parameter tuning, and having thrice as many epochs for training, the MLPs consistently under performed the CNNs. Through experiments 1, 2 and 3, it was observed that the weights for the model initialized using the "Kaiming" distribution were the best and that this distribution constitutes a strong prior for the model. More so, the best activation function is the ReLu function, and the ideal depth for the model is 2 hidden layers. Moreover, through experiments 4 and 5 it was deduced that regularization, both L1 and L2, did little benefit to the model and it was better for accuracy to train the model with normalized images. In the last couple of experiments, a CNN was used as a benchmark for the MLP's performance on both datasets. Then, through trial and error, we could surmise that using an MLP for the MNIST dataset would be feasible, as it was giving test accuracies above 85% consistently, but it would not be feasible to use the MLP model for predicting images from the CIFAR-10 dataset as the accuracy was in the low 40%'s. Furthermore, as an added bonus, we implemented pooling layers in the CNN to reduce overfitting and increase computational efficiency and make the model more robust for recognizing objects, which it did extremely well for the CIFAR dataset. Some relevant papers to these experiments include (Choromanska et al. 2014) and (Zhang et al. 2016). They contain the findings from research extending beyond the scope of this paper which are encouraged for further reading. Important graphs are displayed in line, while others can be found in the appendix.

3 Datasets

Two datasets were analyzed in this project. Both datasets were loaded into Google Colab and then vectorized for processing in Multi-Layer Perceptrons and Convolutional Neural Networks. Both datasets were then normalized using z-score normalization to ensure values centred around 0 within the range [-1,1]. The (10) classes are evenly distributed in both datasets, with each class having an equal number of samples over the whole dataset.
The Fashion-MNIST dataset contains images of Zalando's clothing items. It comprises a training set of 60,000 instances and a test set of 10,000 instances. Each instance represents a 28x28 gray scale image, tagged with one of 10 categories. Every image is 28 pixels tall and 28 pixels wide, totaling 784 pixels. Each pixel possesses a single associated value indicating its brightness, ranging from 0 to 255, where higher values signify darker shades. Figures 1-3 show sample images from the dataset, as well as distributions of the pixels and classes across the dataset.
The CIFAR10 dataset, unlike the Fashion-MNIST dataset, contains colored images. Each of these colored images falls into 1 out of 10 of the following categories: airplane, automobile (car), bird, cat, deer, dog, frog, horse, ship, truck. In total, the dataset has 70,000 images; 60,000 of the images used for training and 10,000 images used for testing. Each image is of the dimension 32x32x3, representing the width, height, and RGB values respectively. Each RGB value is a list with three elements, representing the intensity of red, blue, and green, respectively. The intensities take values from 0 to 255, inclusive. For each image, there are 1024 pixels, each with their own RGB encoding. Figures 4-5 show sample images from the dataset, as well as distributions of the pixels and classes across the dataset.

4 Results

4.1 Varying Weight Initialization

It was observed that the model using a weight initialization of all zeroes failed to learn at all (see Figure 6), staying at the exact same accuracy for all epochs. This is expected since MLPs learn using matrix multiplication, and zero multiplied by any number will always return zero, restricting any learning from taking place. The model using the Kaiming distribution weight initialization seen below in Figure 10, performed the best with the highest accuracy, and the least overfitting on average for all the models. In the table below, we display results for running the model for 300 epochs, which showed significant convergence for values of accuracy between the ranges of 85-88% for the Xavier (Figure 9) and Kaiming distributions. For the rest of the experiments we will be utilizing Kaiming distribution as it showed the most potential with the least amount of variation between the training and testing accuracies, as well as the fastest convergence.

Weight Initialization	Train Accuracy	Test Accuracy(%)
Zeros	10.00	10.00
Uniform	83.70	81.58
Gaussian	81.66	79.38
Xavier	89.56	86.91
Kaiming	88.47	86.41

Table 1: Varying Weight Initialization Effect on Test Accuracy

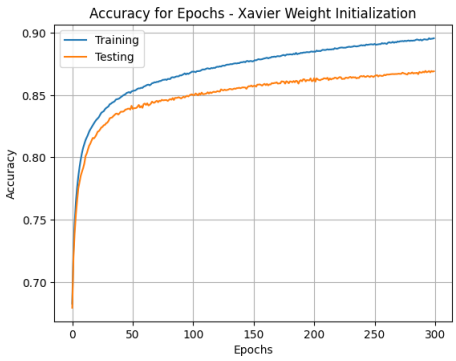


Figure 9: Xavier Weight Initialization

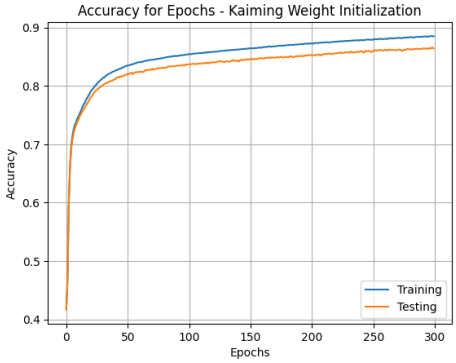


Figure 10: Kaiming Weight Initialization

4.2 Varying Number of Hidden Layers

It can be seen that the model with no hidden layers (Figure 11) did not significantly improve its accuracy while training, while the models with hidden layers showed significant learning. Additionally, it was observed that the more hidden layers a model had, the less overfitting the model experienced and the quicker it converged as well as having a higher accuracy. The results obtained are expected since we hypothesized that a "deeper" network would have an easier time learning the complex patterns and features for the dataset as it could apply more functions. However, the deeper network takes more learning time and therefore is more computationally expensive (as expected). In our case, the performance of both using 1 hidden layer (Figure 12) and 2 hidden layers (Figure 13) is very similar, despite the model having 2 hidden layers going through an initial accuracy slump when first training the model.

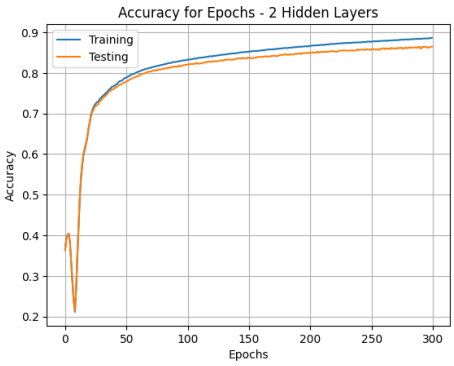


Figure 13: Two Hidden Layers with ReLU Activations

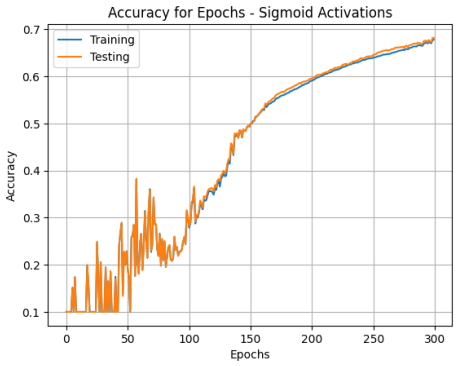


Figure 14: Sigmoid Activation Functions

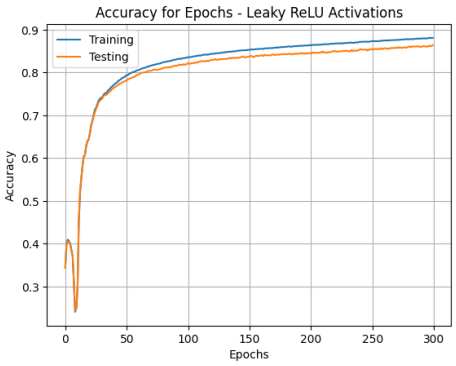


Figure 15: Leaky ReLU Activation Functions

4.3 Varying Activation Functions

This section explored the effect of varying activation functions on the hidden layers of the model. Experiments were done using the Sigmoid and Leaky ReLU activation functions. In both models, Softmax was used in the final layer since this data has multiclass targets. Graphs plotting the training accuracies over 300 epochs can be seen in Figures 14 and 15 above. It can be seen in comparing Figure 15 and Figure 13 that the leaky ReLU performs very similarly to the ReLU activation function. This is expected and aligns with what was seen in class. Figure 14 shows the training accuracy of the model using Sigmoid activations. It can be seen that the training is very unstable and does not trend in any particular way. This is as expected since the sigmoid activation function experiences issues like the vanishing gradient problem. Therefore, we can deduce that ReLu activtation is the way to go forward with our model.

4.4 Effects of L1 and L2 Regularization

L1 and L2 Regularization use different techniques to introduce noise in the model weight's parameters. Both use a regularization constant λ which alters the gradient in different ways; either pushing the weights close to zero, to zero (**L1**), or by introducing noise in the gradients of the weights (**L2**). However, the λ for L1 and L2 will be different, particularly the λ for L1 must be extremely small (0.0005) to allow the model to converge to a reasonable accuracy (See Figure 18), implying that most features (pixels) are important for fitting the model, and there is little redundancy in the features. However, for L2, we use a standard λ of 0.05 which just requires more noise, it takes longer to converge (curvier) which implies that it requires more epochs to converge reasonably (Figure 17). Notice in this graph, that the convergence will be to a lower accuracy than that of without using regularization (Figure 13)."Kaiming" distribution with 2 hidden layers, 128 units each is used for the models. Figure 16 shows L1 regularization with λ equal to 0.01.

4.5 Effects of Using Un-Normalized Images

In this section, the ideal MLP model (2 hidden layers, 128 units each) is used to train the MNIST dataset, but without normalizing the images. A lower accuracy is observed than that from using normalized images, which is around 83% for the testing data as well as the model requiring twice as many epochs to converge around 80% accuracy which is what we want, implying that for quicker and more accurate convergence, the data should be normalized to within the range of [-1,1] for each pixel. The results can be seen in (Figure 19).

4.6-7 CNN vs MLP Implementation on CIFAR-10 & MNIST with CNN Performance

The next section analyzes the implementation of a convolutional neural network (CNN), and the difference in accuracy when compared to the MLP implementations. CNNs are often favored over MLPs due to their ability to adaptive learn spatial features from inputs, and can process multi-channel inputs, advantageous for RGB grids. The PyTorch in the attached Google colab have 2 convolutional and 2 fully connected layers. First, a test

was conducted with the CNN on the MNIST dataset; this implementation only accepted one channel of input (grayscale). The first convolution had a 3x3 filter size, producing 32 output channels. The second convolution accepted the 32 input channels from the last output and produced 64 output channels with another 3x3 filter. Before the data was pushed to the fully connected layers, it was reshaped to 64x5x5 vectors, with the use of pooling. The first fully connected layer takes the data to 128 neurons, and the final layer takes the 128 neurons to 10 final neurons, representing the classes. As shown below, after 100 epochs of training, the CNN with the MNIST dataset was nearly perfect, with 93.40% testing accuracy as seen below (Figure 20). Most of the learning was achieved in the first 40 epochs. This experiment was done with an SGD optimizer which used a learning rate of 0.01, and momentum of 0.9. Regularization techniques such as dropout, L2, and random image flips were later tested, but didn't dramatically improve the testing accuracy. The CNNs 93.40% accuracy was noticeably better than the MLP's 86.41% accuracy (Figure 13). The next experiment sought to analyze the accuracy of the CNN for the CIFAR10 dataset, and compare it to the MLP implementation on CIFAR10. For the CIFAR10 dataset, a similar CNN approach was taken, but with 3 initial channels for RGB and data reshaped to 64x6x6 vectors. Additionally, the results shown below used a 25% dropout rate as well as random crops (with padding) and horizontal flips to help with generalization, remedying overfitting issues. On average, with each epoch, testing accuracy got incrementally better. After 100 epochs, the CNN output a 76.90% accuracy, as seen in (Figure 22), which dwarfed the 43.05% accuracy produced by the MLP for the CIFAR10 dataset (Figure 21). In summation, CNNs surpassed MLPs on both MNIST and CIFAR10 datasets, especially evident for CIFAR10 with its RGB channels. This highlights CNNs' superior capability in discerning non-linear image patterns, as present in the CIFAR dataset.

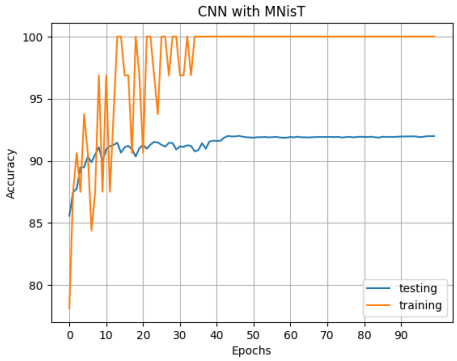


Figure 20: CNN with MNIST

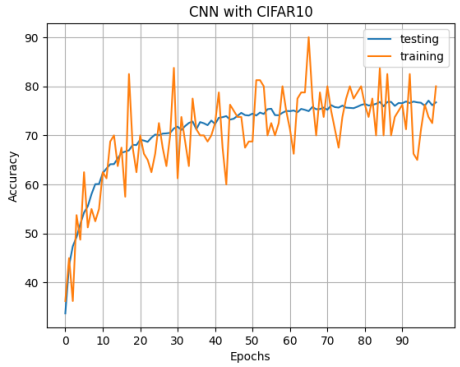


Figure 22: CNN with CIFAR10

4.8 Experiments with Different Optimizers

In the experiments mentioned above and below, a stochastic gradient descent optimizer was used to navigate the minimums with respect to the cost functions. For the previously mentioned experiments, a momentum parameter of 0.9 was used. The concept of momentum refers to using an exponential moving average of gradients to dictate how minimums of the cost function may be found; a higher momentum factor implies past gradients have a larger effect on current updates, while a lower momentum factor implies past gradients have a lesser effect on current updates. The experiments regarding momentum, as shown below, first tested with a momentum parameter of 0, and then 0.25, 0.75, 0.9, and 0.99. Their accuracies were 62.37%, 66.78%, 78.23%, and 73.11% respectively. We can see that 0.9 had both the most stable training and most accurate testing results. Lastly, an Adam optimizer was employed to see if this would result in different convergence patterns. Adam (Adaptive Moment Estimation) combines the benefits of the AdaGrad and RMSProp optimization algorithms. It maintains an exponentially decaying average of past gradients and squared gradients to adjust updates. We can see that with the same configuration as the former experiments, except for the optimizer, this led to a testing accuracy of 62.45%. While this looks relatively stable, it is still less ideal than standard momentum with everything else held constant. However, an investigation into different learning rates (i.e. 0.001), may be appropriate.

4.9 Exploring Pretrained Models

Out of further curiosity, 2 pre-trained models were used to investigate picture classification: ResNet-18 and SqueezNet. While ResNet-18 was designed to handle CIFAR10 images, SqueezNet was not, and required dimension compression. All of their convolutional layers were frozen and existing fully connected layers were removed. Subsequently, the following fully connected configurations were utilized [512], [512, 256], [512, 256, 128]. The first of the formerly mentioned was for simple comparison, the second was to allow for some abstraction before classification, and the third added a second hidden layer for an extra layer of abstraction to pick up embedded features. Due to a lack of GPU credits, the experiments stuck to 10 epochs. Although SqueezeNet did take less time to train than ResNet-18, it still took longer than the MLP, and didn't show any signs of convergence or testing accuracies above 10%. On the other hand, ResNet-18 was slower to train for more layers of abstraction and tended to be as quick for more simple configurations. The simpler configurations also did slightly better than the more complex ones, in terms of accuracy. At epoch 10, the 1 hidden layer configuration approached a training accuracy of 52.49% and testing accuracy of 48.23%, while the 2 hidden layer configuration approached a training accuracy of 50.28% and testing accuracy of 46.77%. It appears that the extra layer of abstraction could have been too complex for a smaller number of epochs, however, if more epochs were to be run it is possible the extra layer of abstraction could appear advantageous. This, however, still underscored optimal CNN implementation in terms of accuracy on the 10th epoch.

ResNet-18 output from 10 epochs:
Configuration [512, 256]:
Train Accuracies: [44.594, 47.482, 49.096, 49.79, 49.554, 50.528, 50.51, 51.768, 51.85, 52.49]
Validation Accuracies: [43.34, 45.86, 47.07, 47.38, 46.74, 46.82, 46.96, 47.5, 47.85, 48.23]

Configuration [512, 256, 128]:
Train Accuracies: [43.672, 46.582, 47.39, 48.318, 49.512, 51.174, 51.322, 50.834, 51.506, 50.288]
Validation Accuracies: [42.03, 44.16, 45.42, 46.03, 47.16, 47.7, 47.81, 47.74, 47.94, 46.77]

5 Discussion and Conclusion

Two primary datasets, Fashion-MNIST and CIFAR10, were explored with vectorization and normalization processes for neural network applications. During weight initialization experiments, the Kaiming distribution emerged as the most effective, producing the best accuracy. In-depth model assessments showed that networks with more hidden layers learned faster and exhibited less overfitting, while the ReLU activation function outperformed the Sigmoid. Regularization techniques revealed that L1 needed smaller lambda values for convergence, whereas L2 required more epochs. Convolutional Neural Networks (CNNs) significantly surpassed Multi-Layer Perceptrons (MLPs) in accuracy on both datasets, and experiments with various optimizers highlighted the efficacy of momentum in Stochastic Gradient Descent.

6 Statement of Contributions

All group members worked on all tasks together. Abdullah and Gostovic focused on the MLP Integration, while Novick was responsible for the CNN implementation. All group members ran tasks together synchronously.

References

Choromanska, Anna et al. (2014). "The Loss Surface of Multilayer Networks". In: *CoRR* abs/1412.0233. arXiv: 1412.0233. URL: http://arxiv.org/abs/1412.0233.

Zhang, Chiyuan et al. (2016). "Understanding deep learning requires rethinking generalization". In: *CoRR* abs/1611.03530. arXiv: 1611.03530. URL: http://arxiv.org/abs/1611.03530.

Appendix

Figure 1: Fashion MNIST Sample Images

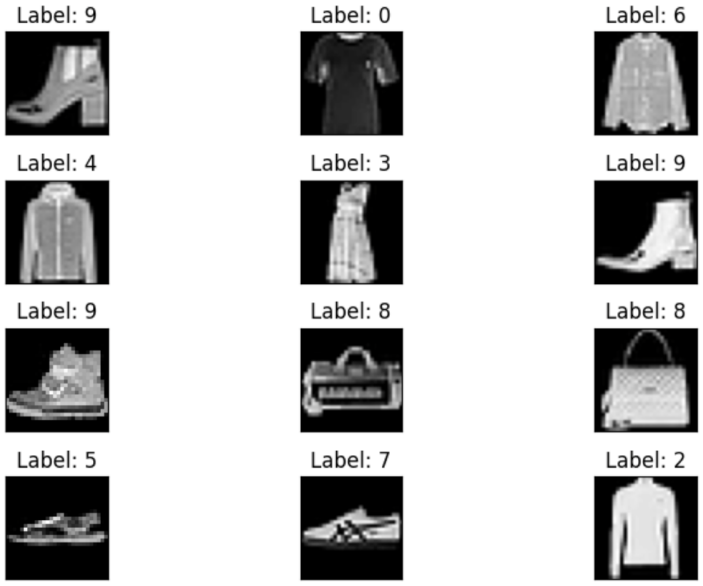


Figure 2: Fashion MNIST Pixel Distribution

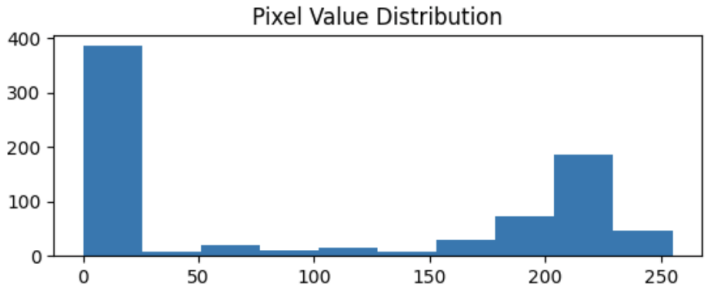


Figure 3: Fashion MNIST Class Distribution

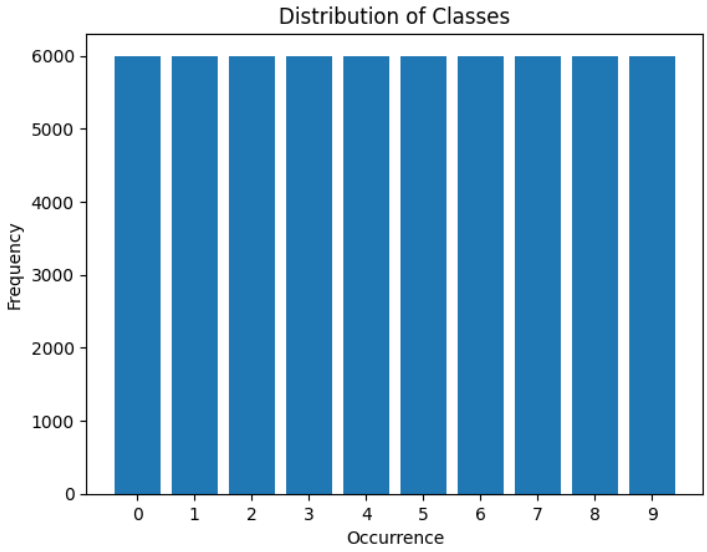


Figure 4: CIFAR10 Sample Images

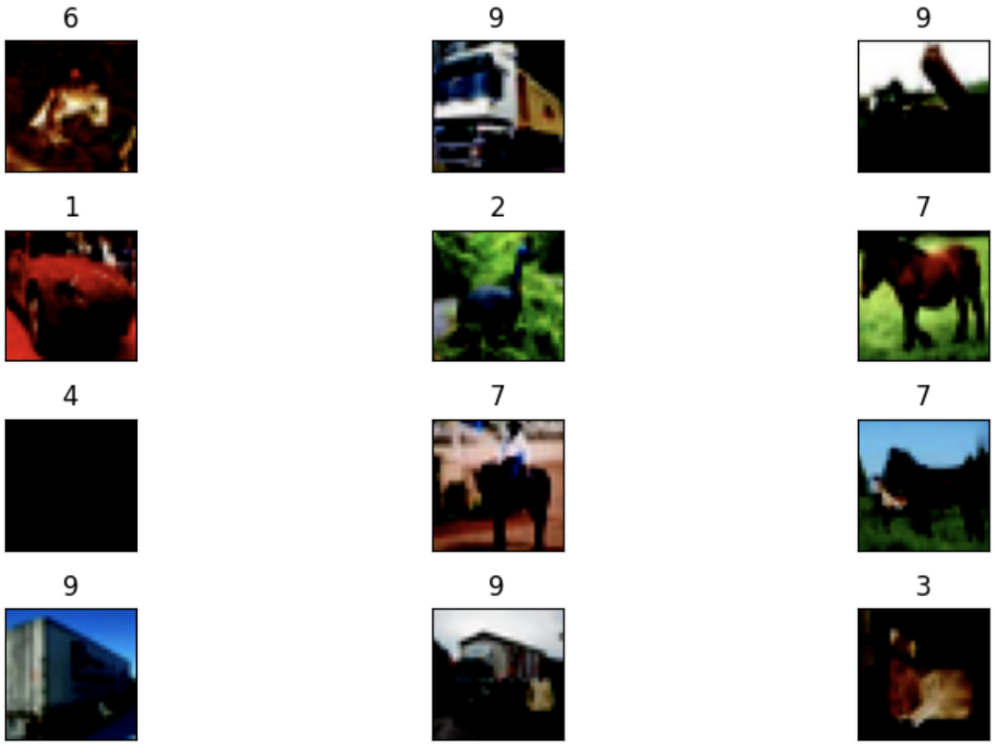


Figure 5: CIFAR10 Class Distribution

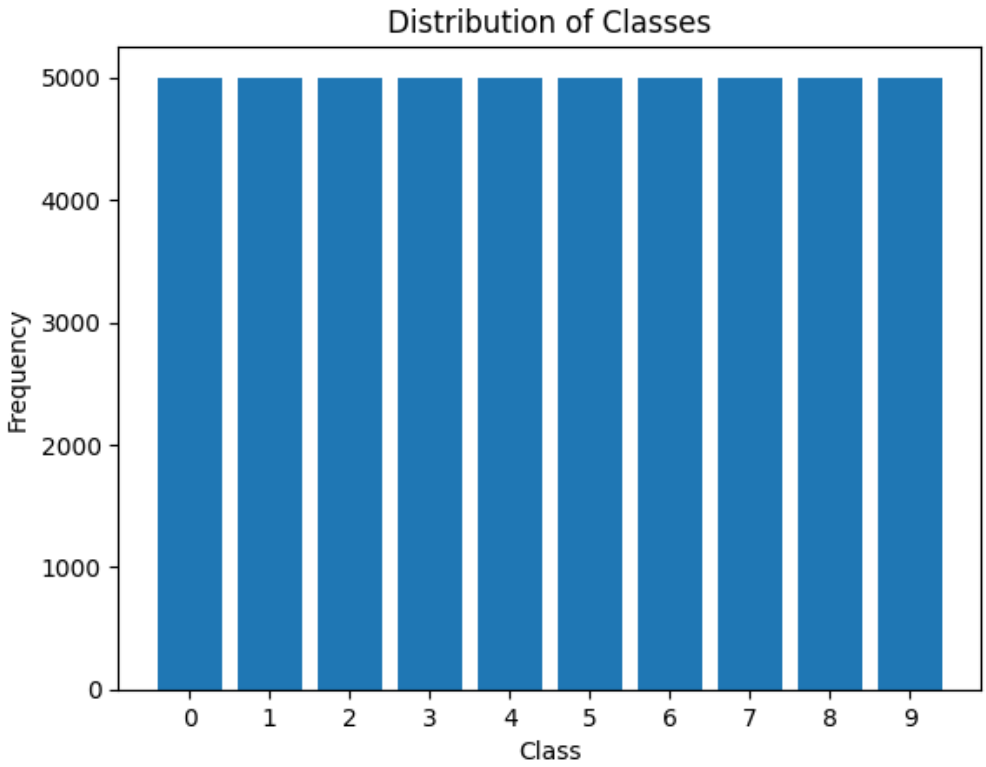


Figure 6: Zeros Weight Initialization

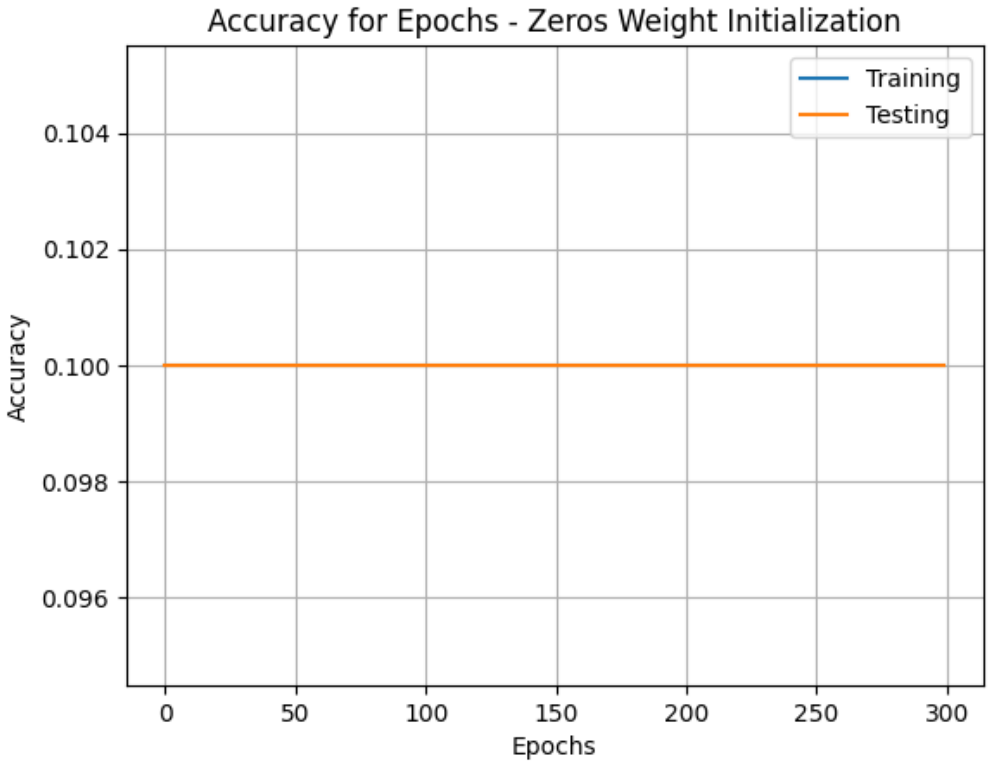


Figure 7: Uniform Weight Initialization

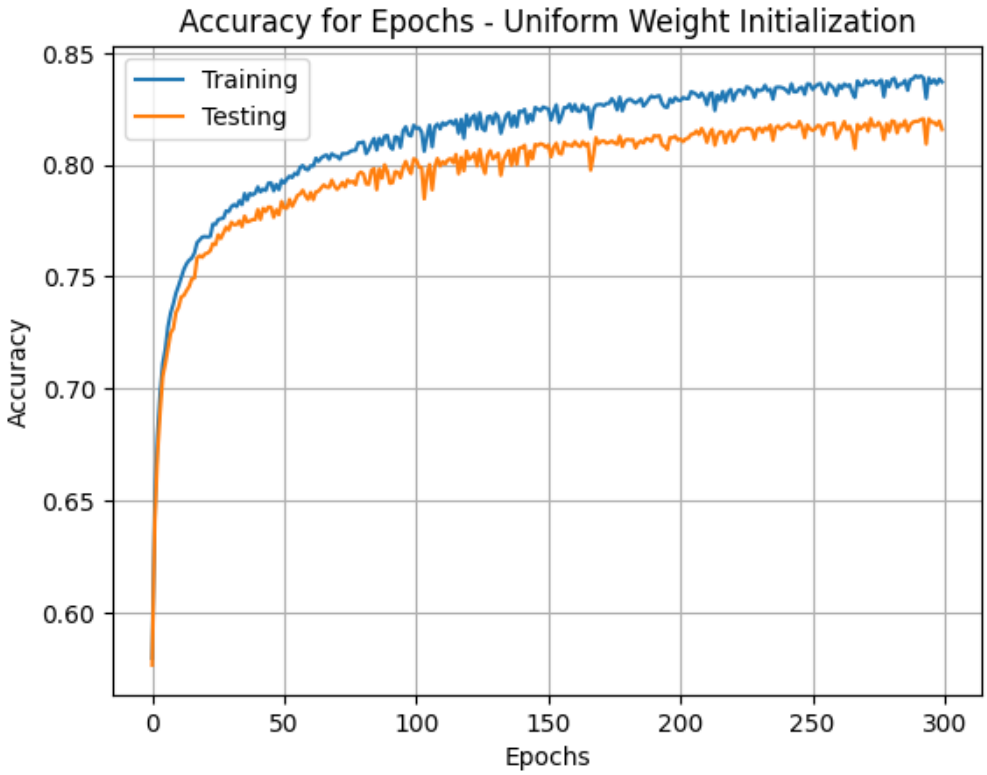


Figure 8: Gaussian Weight Initialization

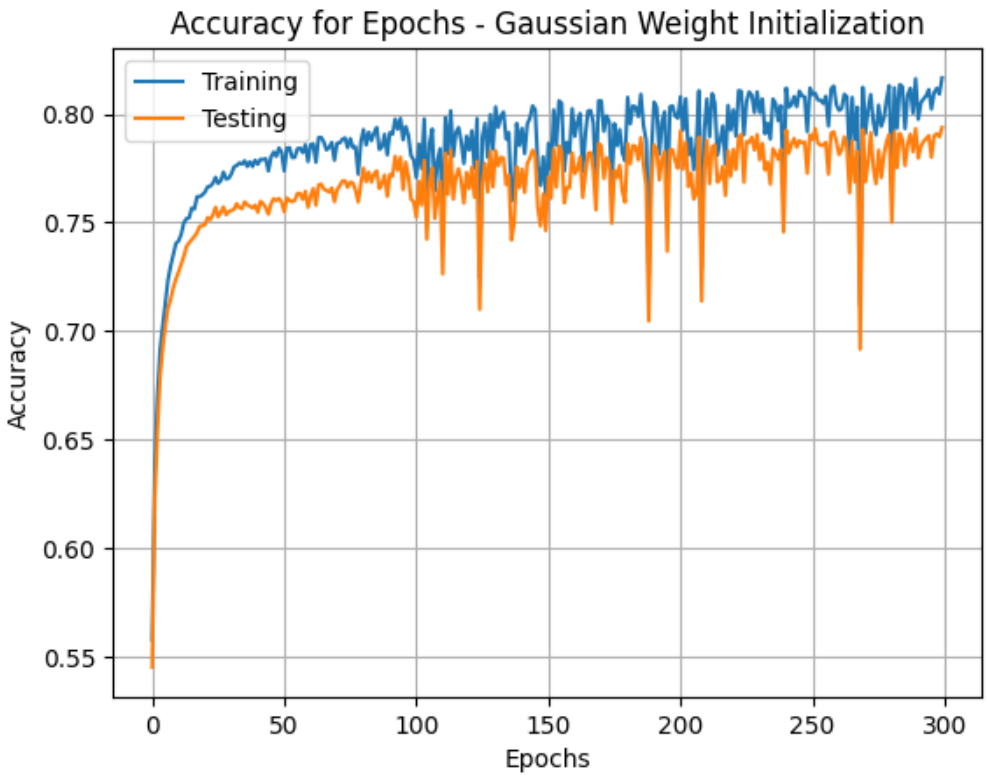


Figure 11: No Hidden Layers

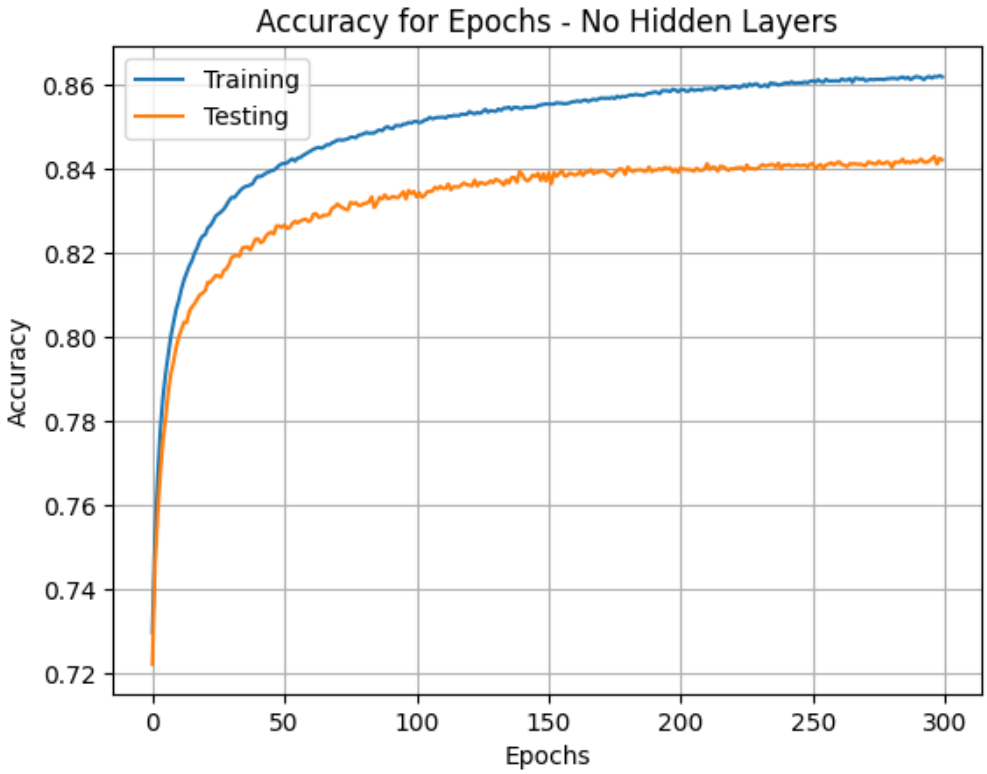


Figure 12: One Hidden Layers

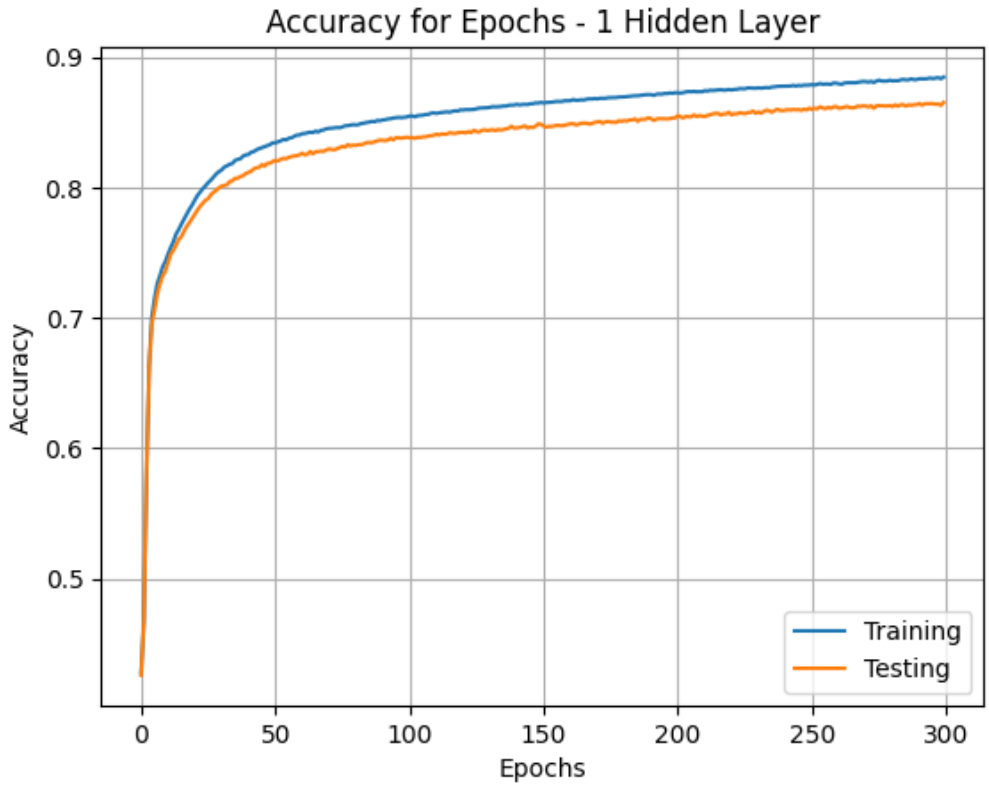


Figure 16: L1 Regularization $\lambda = 0.01$

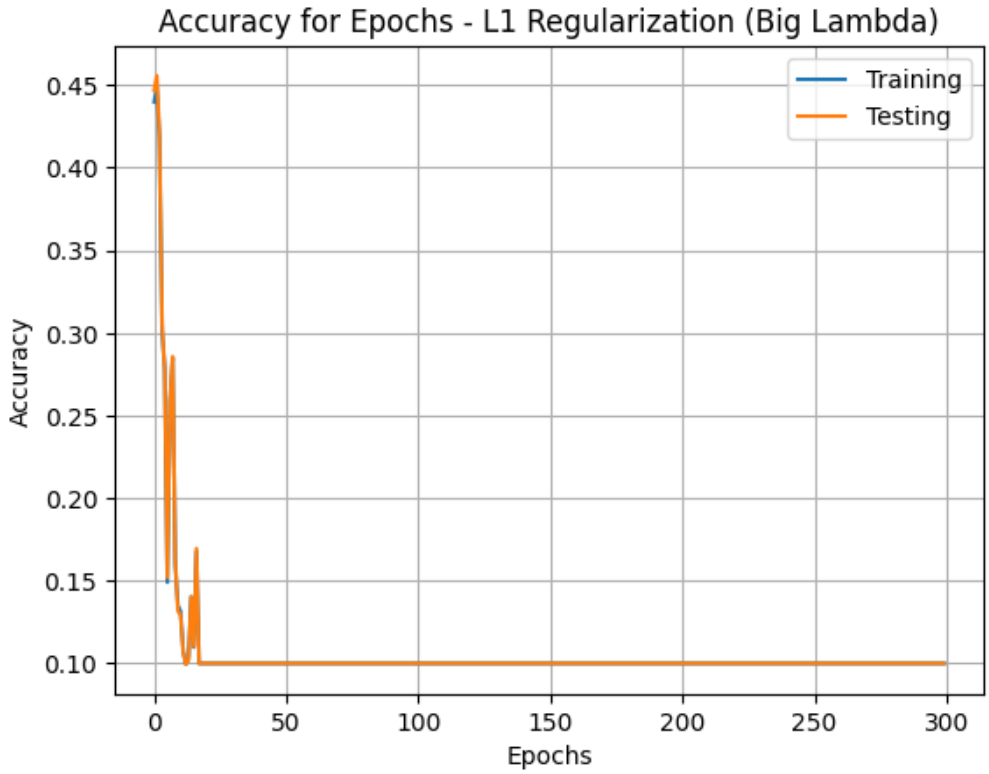


Figure 17: L2 Regularization $\lambda = 0.05$

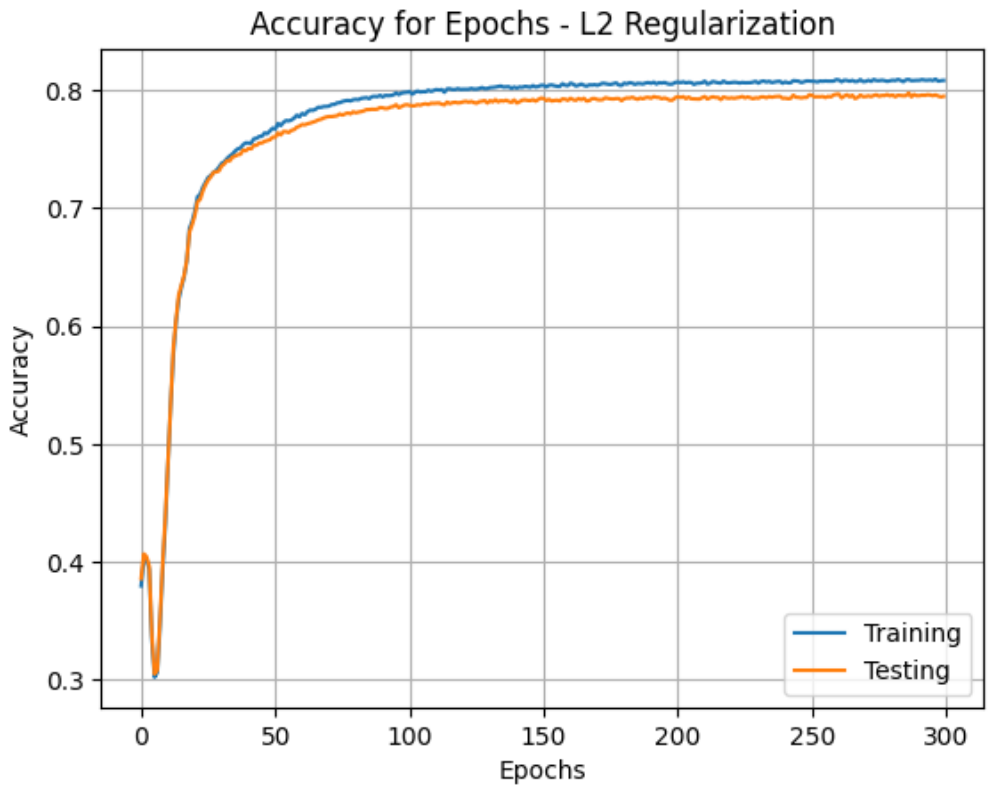


Figure 18: L1 Regularization $\lambda = 0.0005$

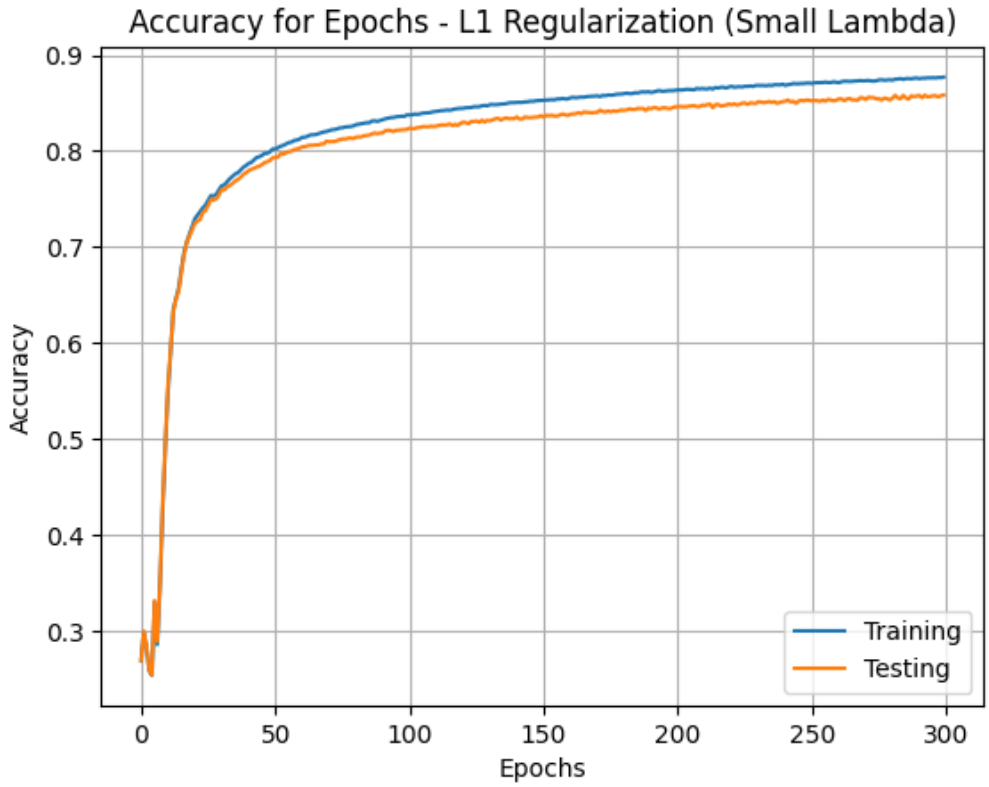


Figure 19: Unnormalized MNIST data training

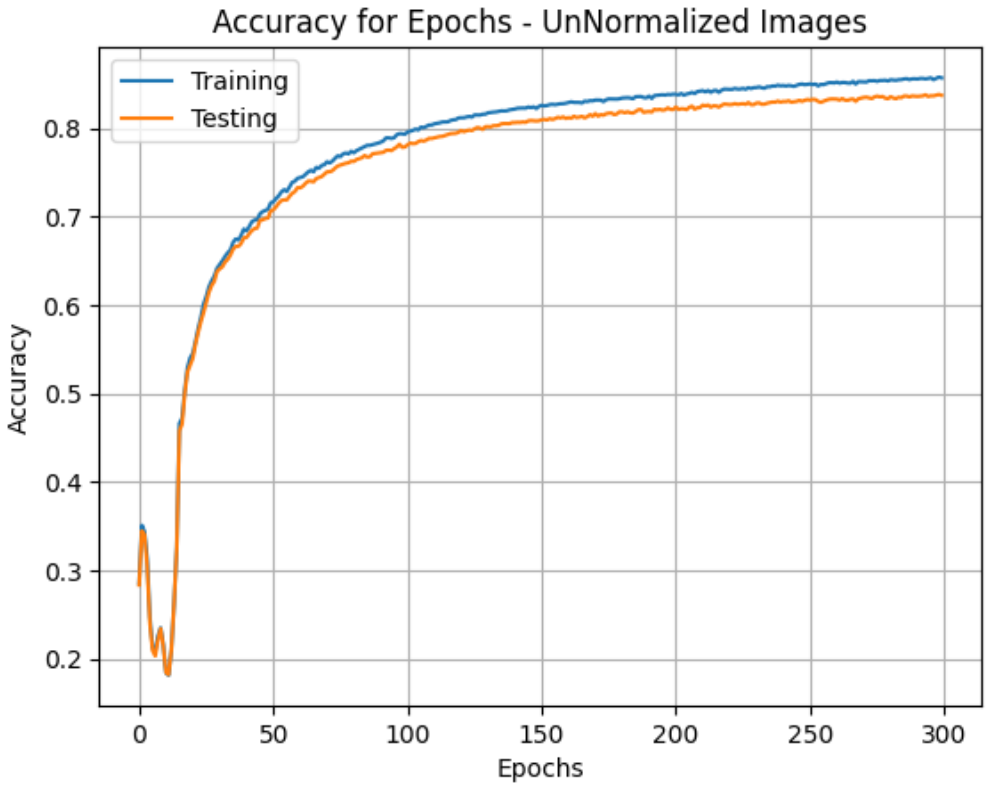


Figure 21: MLP with CIFAR10

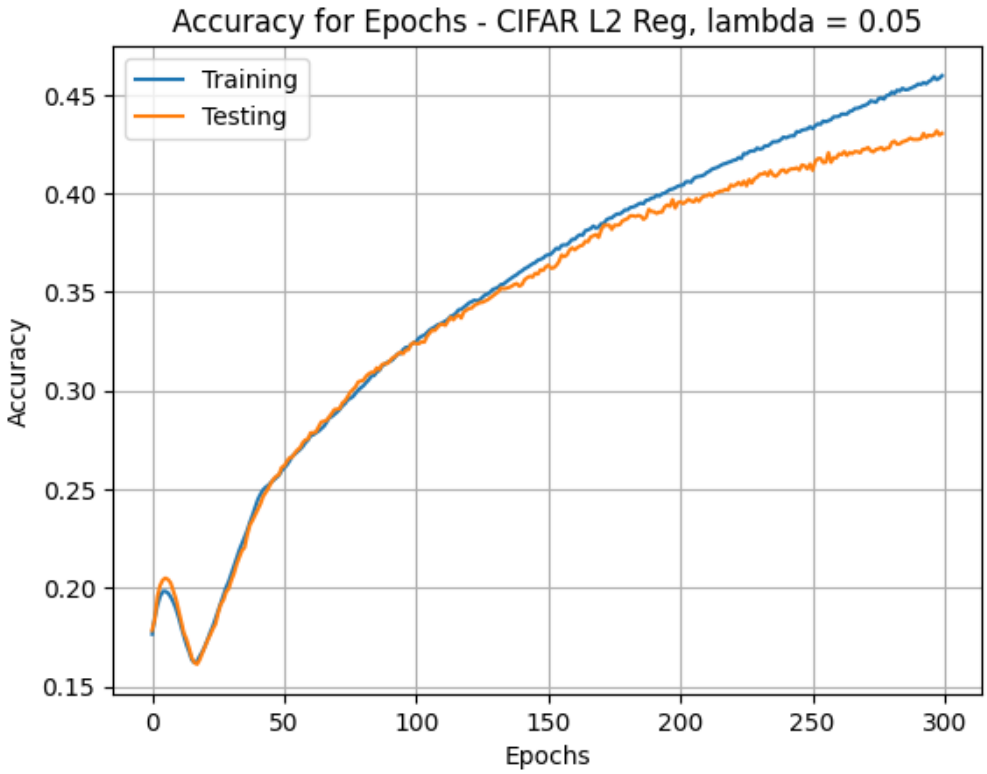


Figure 23: No momentum

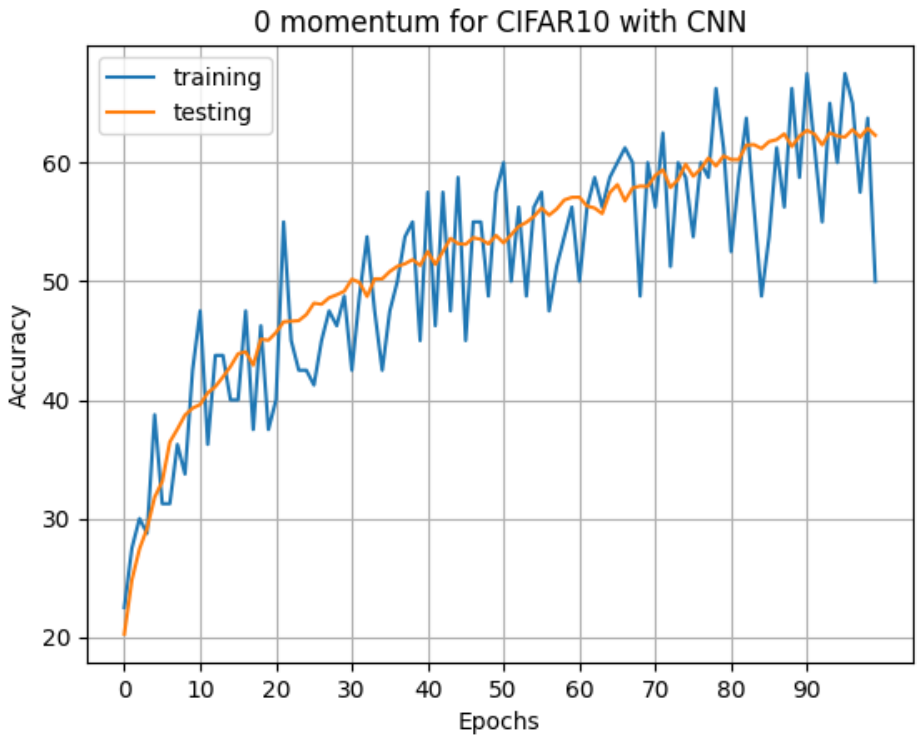
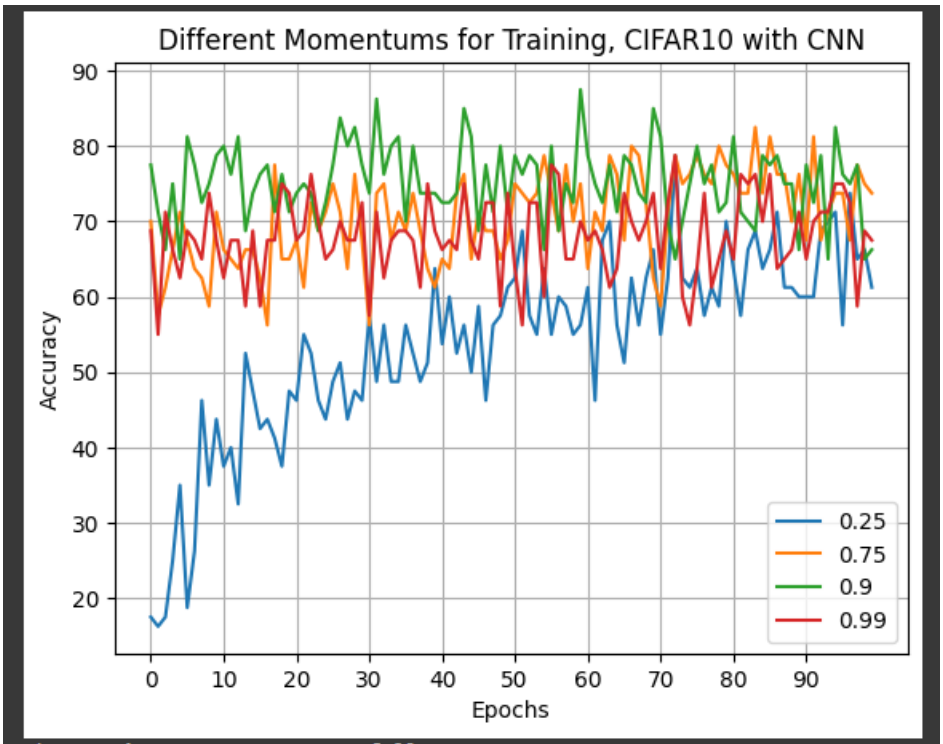


Figure 24: Different Momentums



The testing otuputs were as follows:

Momentum:	Accuracy
0.25	66.78%
0.75	76.34%
0.9	78.23%
0.99	73.11%

Figure 25: Adam