COMP 551 Mini-Project 2
Image Classification Through MLP and CNN

Yang Kai Yam, Xu Michael, and Tian Yu Hao

School of Computer Science, McGill University

Abstract

In this project, we explored the multilayer perceptron (MLP) and convolutional neural network (CNN) models by doing image classification on the Fashion MNIST and CIFAR-10 datasets. We first implemented an MLP from scratch and evaluated its performance in different configurations. We determined a model that yielded the best results with an experimental accuracy of 85.7%*. We then trained and ran CNNs to compare them to MLP implementations of neural networks fitted to both the datasets. We found that CNNs perform better on average with image classification by a non-negligible amount, yet our best performing models were pre-trained ones (VGG11 for instance) taken from Pytorch's database.*

Introduction

With the goal of familiarizing ourselves with MLPs and CNNs, we trained various configurations of them on image data, with pixel inputs and class outputs. Our decision-making relating the "**goodness**" of a model was based mainly on the accuracy obtained on the test set, as well as the graphs of its training and cross-validation errors over training epochs. The latter helped us prevent overfitting, which is very common in neural networks due to their universal approximation property. This project was created in two main parts, the first being the MLP implementation on Fashion MNIST and the second the CNN implementation on CIFAR-10, with some combination of either or in some instances. In the end, we convinced ourselves of
1. The various optimization methods and their respective benefits on the accuracy of the model as presented below and...
2. The common knowledge that CNNs work better than MLPs in image classification problems.

Dataset

The datasets are collection of images of everyday life items and animals, which are represented in matrices each with dimensions 28 x 28 pixels. That remains true for both datasets, however CIFAR-10 instances do have an additional dimension in the RGB domain, making them 28 x 28 x 3 pixels per image. After a quick visualization, we scaled the datasets by the max pixel value and then normalized them according to the training data μ and σ as proposed in reference 1 of the appendix.

Results

A. Multilayer Perceptron (MLP)

An MLP is implemented and trained on Fashion MNIST to perform image classification with 10 classes output by 784 pixel 1D input. An Optimizer is built with a mini-batch SGD. The MLP model supports a wide configuration of different activations (ReLu, Sigmoid, Softplus) and layer designs. The final layer is softmax for probability calculation. The forward pass and backpropagation are implemented according to the chain rule in neural network graphs and prime functions.

A1. Different Weight Initialization Optimal weight initialization is found in each activation.

Figure 1 shows the training curve and test accuracy using ReLU. We find that the zeros, uniform, and Gaussian weight initialization methods deliver subpar performance in training, while "Xavier" and "Kaiming" offer the best training accuracies at 0.85. We have implemented a similar experiment using the other activations in Appendix A1, which show similar results. We see from the graph of training loss curves that Xavier and Kaiming normalization

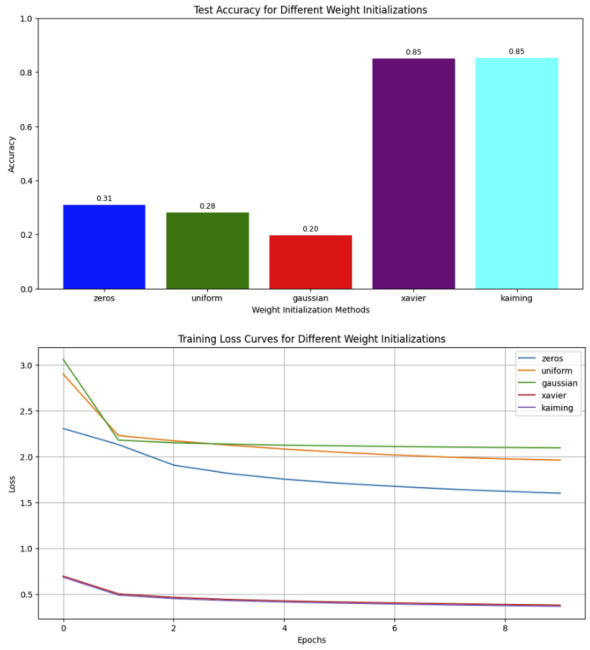


Figure 1: *The training curve and test accuracy chart with different weight initialization methods in ReLu*

make it so we start at much smaller values for the average loss across all training examples.

A2. Different Layer Structure Using ReLU, we observe the test accuracy in differently layered architectures.

Layer Structure	Accuracy Percentage
Only Softmax	71.35
1 Hidden Layers	85.27
2 Hidden Layers	85.31

Table 1: *Test accuracy of the relevant three architectures*

Table 1 and Figure 2 indicate that the model with no hidden layer (only softmax) obtains a relatively high loss curve and test accuracy of 0.72. However, the performance metrics seen between the models with 1 or 2 hidden layers present a marginal improvement for the deltas hovering at about 0.85. We believe they are related to the change in non-linearity and network complexity, which will be mentioned in the discussion.

A3. Different Activation Choices ReLU, Sigmoid, and Softplus are our study targets. Using the Kaiming weight initialization method and 2 hidden layers neural network with 128 units each (obviously good parameters), as well as the same learning rate of 0.1 and batch size 128 to

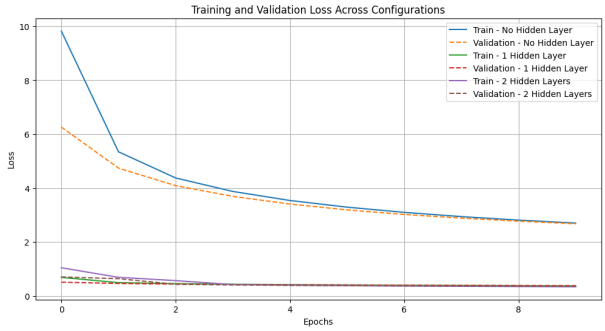


Figure 2: *The loss curve of different layer structures*

train all three models. The result is shown in Figure 3 that only the sigmoid activation is well-behaved, while ReLU and Softplus are barely activated.

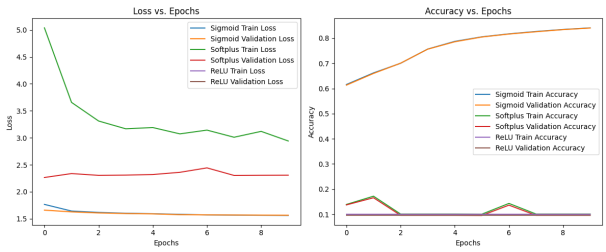


Figure 3: *Different Activation with the same learning rate of 0.01*

With that, we believe that the learning rate is sensitive for ReLU and Softplus, which the larger learning rate will make dying neutrons. So we decrease the learning rate, where both activations are activated and show surprising results. Figure 4 presents the test accuracies in descending order: ReLu (85.7), Softplus (84.9), Sigmoid (83.4), while Figure 5 shows the newly updated loss and training curves.

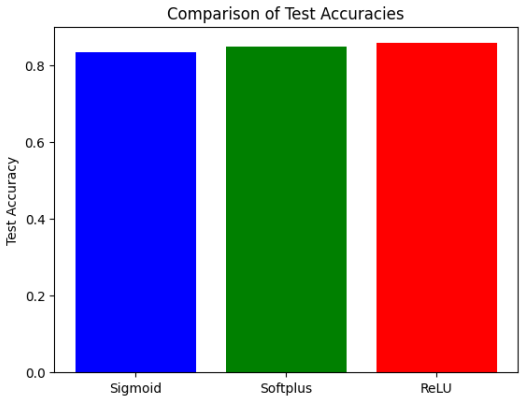


Figure 4: *The Test Accuracy of different activation*

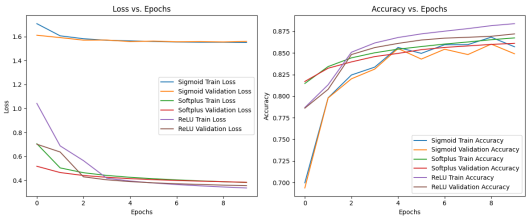


Figure 5: *Influence of lowering the learning rate on ReLU and Softplus*

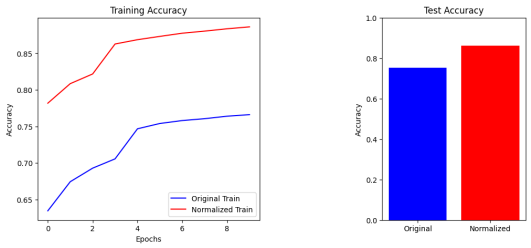


Figure 6: *Test accuracies for different normalizations*

A4. Influence of Regularization In this section, we explored the effect of regularizing our parameters throughout the learning. We implemented L2, L1, and normal SGD on a MLP with 2 hidden layers composed of 128 nodes. To hopefully minimize the influence of the other hyperparameters on every individual model, we kept the activation function as ReLU, as well having the learning rates and mini-batch sizes consistent through every model. Our findings are presented in Table 2. The training and cross-validation loss curves can be found in the appendix section A4. The model doesn't take well to not being regularized with a ReLU activation function.

Regularization	Test Accuracy
L2	0.806
L1	0.717
No	0.101

Table 2: *Test accuracies of the different regularizations*

A5. Influence of Normalization A neural network with similar parameters as presented in section A4 is initialized and trained on data which is not normalized. Its performance is then tested against the same model trained on data that was preprocessed with Kaiming normalization as was done in section A1. The model trained on the non-normalized dataset finished with a training accuracy of 0.76 compared to the normalized one with an accuracy of 0.88 as seen in Figure 6. The same trend can be seen in the test accuracies.

Extra: The Optimal Parameter Configuration Presented in Table 3 are the best parameters for every activation function we looked at. Note all of them follow the MLP structure with two 128-nodes hidden layers running L2 regularization for optimal performance for our use case.

Params	Sigmoid	ReLU	Softplus
batch size	128	128	128
lr	0.1	0.01	0.01
weight init	kaiming/xavier	kaiming	kaiming

Table 3: *Table of best experimental parameters for each activation*

B. Convolutional Neural Network

B1. CNN vs MLP on Fashion MNIST A CNN with 2 convolutional, 2 fully dense layers, 2 pooling layers, and one dropout layer is implemented with our choice of hyperparameters. We find that this model with a learning rate of 0.5 and a kernel size of 3 can achieve a test accuracy of 0.88 after 5 training epochs. This is better than the test accuracies offered by the previously trained MLP models within three percent. The improvement, while notable, isn't drastic. The reason for that could be related to the complexity of the data (or lackthereof), as the small sizes of the original images and their easily discernible patterns mean they aren't overly complicated to fit to. Our specific choice of hyperparameters can be explained by looking at section B1 of the appendix.

B2. CNN vs MLP on CIFAR-10 Referencing Table 3, we use ReLU parameters on our MLP with two hidden layers to train on the CIFAR-10 dataset. Figure 7 presents the training and CV accuracies that plateau around 0.55. Looking at the test accuracy of 0.48, we see that results obtained aren't stellar on this first run against the CIFAR-10 dataset. On the other hand, we initialize a similar CNN to the one presented in B1, and we obtain a better test accuracy of 0.66. Again, specific choices of parameters have been determined looking at Figure 8.

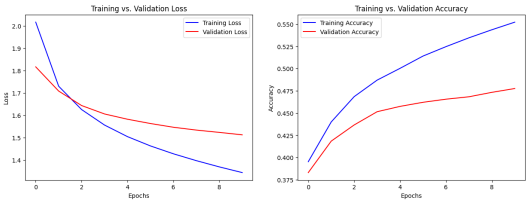


Figure 7: *Training metrics for MLP against CIFAR-10 dataset*

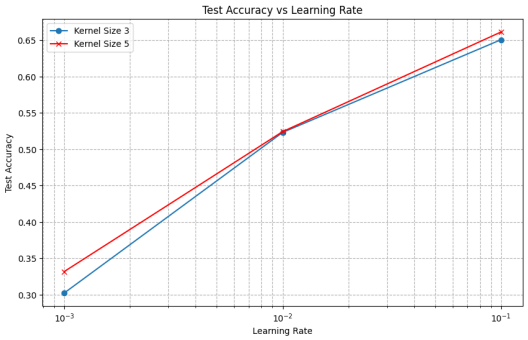


Figure 8: *Training metrics for CNN against CIFAR-10 dataset*

B3. Effects of Optimizers Using the same model, we now test for the effect of momentum in SGD optimizer vs the effect of the Adam optimizer for CNNs. Starting with the SGD optimization, we test the model with momenta in the range of $[0.3 - 0.9]$ with stepsize 0.3. In Figure 9, we see that the Adam optimizer's performance is significantly worse. This could be attributed to the fact that Adam adjusts its learning rate based on the initial several gradient vectors which may imply that our learning rate is not suitable for high dimension data like RGB images (lr = 0.00001). All in all, as we increase the momentum, the convergence speed and accuracy follows suit, however the loss becomes more unstable.

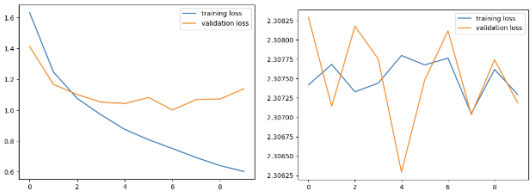


Figure 9: *Loss Curves for SGD (momentum = 0.6, left) and Adam (right). Y-axis represents loss and X-axis the epochs*

Extra: VGG11 In this section and the next, we explore the option of using pre-trained models to fit CIFAR-10 data. For the vgg11 model, we test for accuracy depending on the number of dense layers we add, represented in Figure 10. We choose to add two dense layers for the best accuracy. Following that, in Table 4, we have presented the aggregated best test accuracy results for the three models. There is a huge improvement using vgg11 instead of the other two.

Model	MLP	CNN	VGG11
Test Acc	0.48	0.66	0.82

Table 4: *Table of the relative performance of different models on CIFAR-10 data*

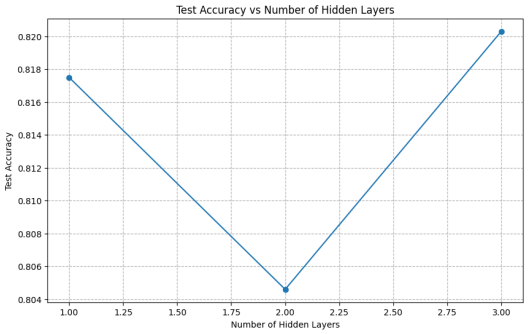


Figure 10: *Test accuracy for vgg11 against the number of hidden layers added*

Extra: ResNet We do the same as the previous section, but with ResNet-18 instead. The results are as presented in Figure 11. The best accuracy we obtained is 0.77 with zero hidden layers. The individual loss curves for each number of hidden layers added can be found in the appendix section B4. And once again, although it doesn't perform as well as VGG11, we have that the accuracy for the pre-trained models trained on CIFAR-10 is higher by a significant amount.

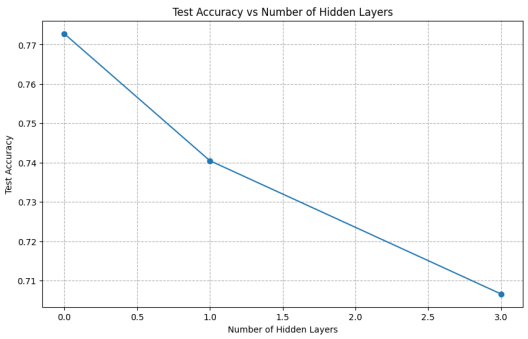


Figure 11: *Test accuracy for ResNet18 against the number of hidden layers added*

Discussion and Conclusion

Learning Rate Sensitivity in MLP Sigmoid is better working in relatively higer learning rate to avoid the vanishing gradient problem [2]. The ReLu and SoftPlus are started to be activated when applying the low learning rates, to prevent the zero derviatives [3]. It is believed ReLU and SoftPlus are highly sensitive to learning rates, that a higer learning rate can lead to very aggressive weight updates that there is a higher of risk of creating dying neurons in ReLU.

Influence of weight initialization in MLP Different Activations have their optimal weight initialization. For all the activations, the zeros weight initialization obtains a bad result because it results in no gradient flow, making neurons symmetrical and hindering effective learning.

For the sigmoid activation, the methods (Uniform, Gaussian, Xavier, Kaiming) ensure the function operates in its sensitive range where gradients are more pronounced and prevent the vanishing gradients. For the ReLU and Softplus activations, they are linear of positive inputs. So, the Xavier and Kaiming initializations are highly suitable to keep activations' variance consistent across layers and prevent the issue of "dyling ReLU", making the inactive nodes.

Non-linearity and network depth in MLP They play important roles in our MLP design.

Non-Linearity: From Table 1, we observe a significant enhancement in test accuracy when we transition from a model with no hidden layers to one with a hidden layer. The reason is due to adding non-linearity. The model with no hidden layers simply uses softmax functions for training, which we posit is not enough to comprehend the intricate inter-relationships between pixels. Introducing even one single hidden layer can harness the potential of non-linearity. Non-linear activations enable a network to learn and represent intricate patterns, which a linear model would typically miss. So the model with a hidden layer can implement complex pattern recognition.

Network Depth: But changing from 1-hidden layer to 2-hidden layers did not result in a huge improvement, because of the simplicity of data and complexity of the network. Fashion MNIST is a 1 channel 1D pixels without RGB dataset, which is a relatively simple image input. This means the data might not have more intricate patterns that need deeper networks to capture. Alternatively, the depth might introduce more parameters, increasing the risk of overfitting if not managed correctly. Both results are expected because the hidden layer is expected to improve the accuracy by learning complex patterns, while the datasets are not extremely complex themselves, thereby using one hidden layer is adequate.

CNNs: Transitioning into a more complex dataset with the introduction of CIFAR-10, whose inputs are much higher-dimensional, we start seeing our simple MLP implementation suffer in terms of learning prowess. While the improvement on the accuracy of the CNN vs MLP against Fashion MNIST is marginal at best, there is an obvious gap that forms between the two looking at Table 4. This disparity created by the fact that CNNs have the additional ability to capture more complex and abstract patterns through convolutional layers is made clear the moment we increase the complexity of our features.

Our Best Runs: Putting it all together, it seems like the models that net us the best results by far are the pre-trained models as we have explored in the two latter extra sections of the results. In a future experiment, it would be interesting to test further the potential of these pre-trained models and different ways of initializing them.

Statement of Contribution

Yang Kai Yam - Implemented the architecture of the scratch MLP model, optimizer, and backpropagation algorithm (Task 2). Designed and ran the experiment for MLP parts (Experiment 3.1-3.3). Wrote the report for MLP results and discussions.
Xu Michael - Acquired and normalized the data. Worked on the CNN and pre-trained model part. Assisted other teammates to finalize the report.
Tian Yu Hao - Helped code and design the MLP from scratch section. Tabulating and putting together the results for the CNN part of the experiment. Wrote and finalized most of the report except for sections A1-A3.

Reference

1. Stanford class notes (CS231n Convolutional Neural Networks for Visual Recognition)
   https://cs231n.github.io/neural-networks-2/#datapre

2. Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 6(2), 107-116. https://doi.org/10.1142/S0218488598000094

3. Lu, L., Shin, Y., Su, Y., & Karniadakis, G. E. (2020). Dying ReLU and Initialization: Theory and Numerical Examples. arXiv preprint arXiv:1903.06733. https://arxiv.org/abs/1903.06733

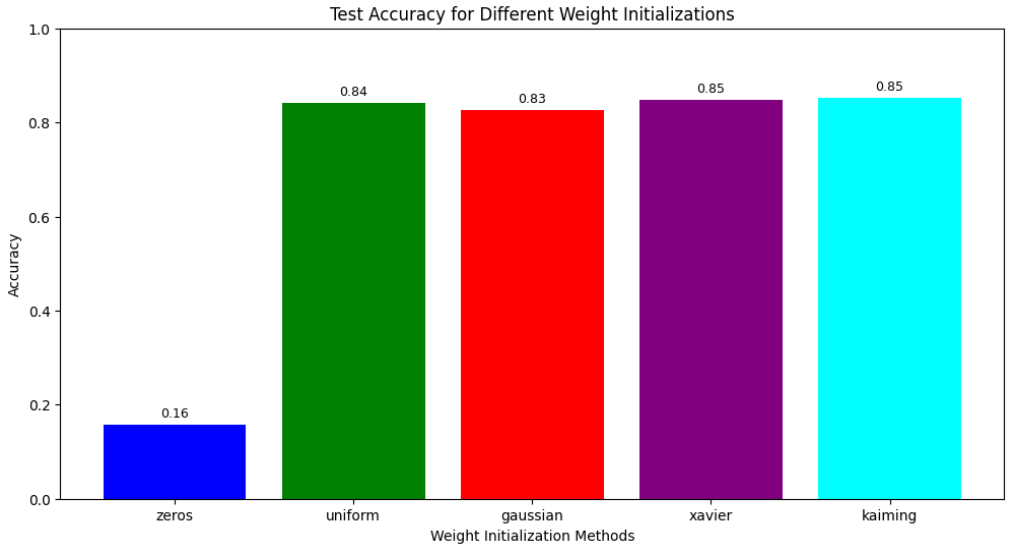
Appendix

A. Multilayer Perceptron (MLP)

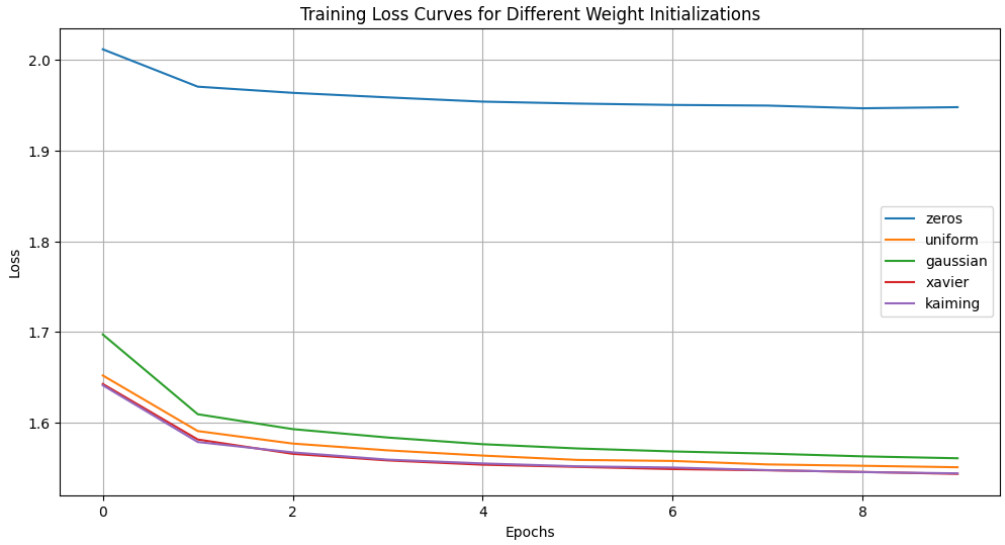
1. Different Weight Initialization For Each Activation

The report has shown the performance of different weight initializations in ReLu Activation. For a more comprehensive analysis, we have also analyzed the influence of the initialization methods in Sigmoid and Softplus respectively.

Sigmoid Activations: Except for Zeros, all the weight initializations perform well.

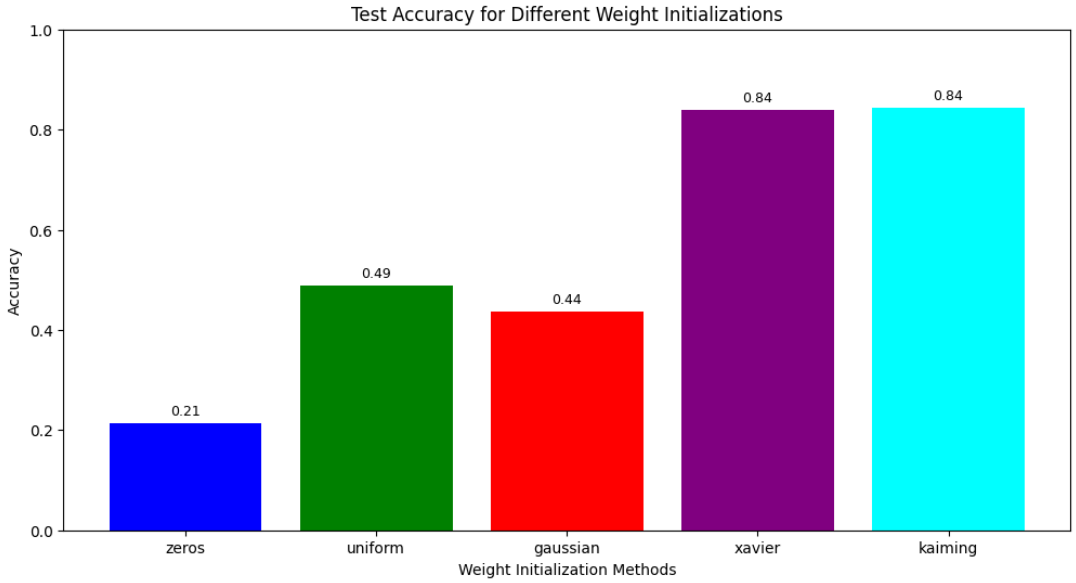


(Figure A1.1 The test accuracy in Sigmoid with different weight methods)

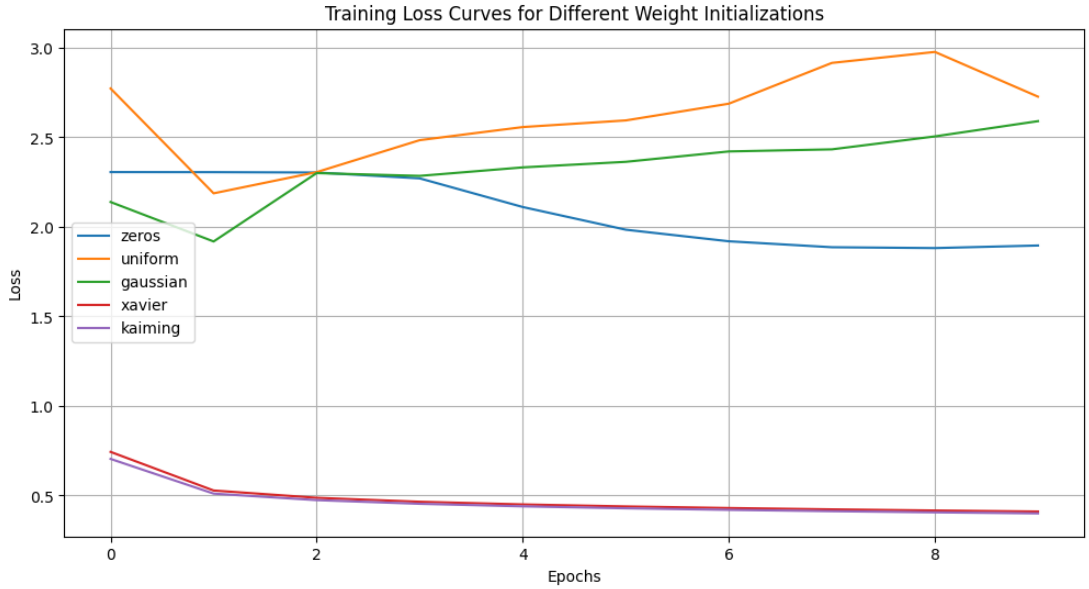


(Figure A1.2 The loss curves in Sigmoid with different weight methods)

SoftPlus Activations: The zeros, uniform, and Gaussian methods show inconsistent performance, while the Xavier and Kaiming are presenting pretty well and consistent results in test accuracy.



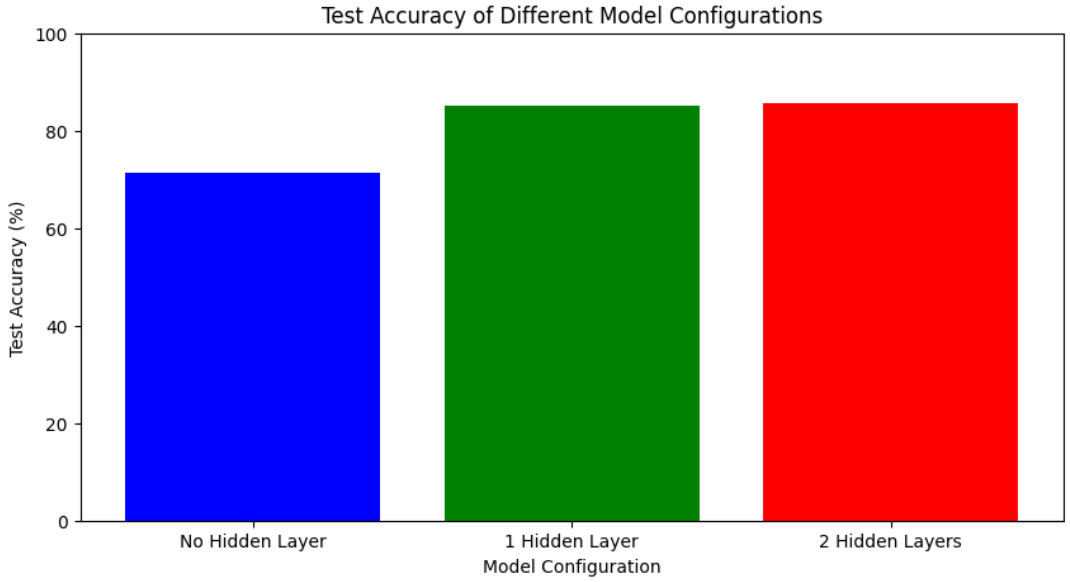
(Figure A1.3 The Test Accuracy in SoftPlus with different weight methods)



(Figure A1.4 The Loss Curves in SoftPlus with different weight methods)

2. Difference of Different Layer Structures in ReLu

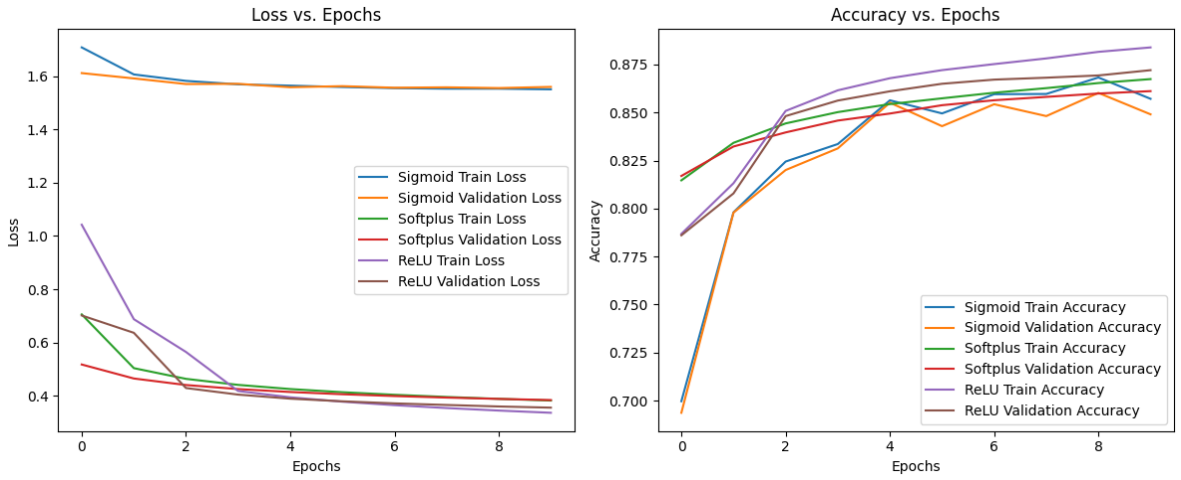
Additional information showing the test accuracy of different layer settings. The No-Hidden Layer is worse than the Having-Hidden Layers. However, it seems the network depth did not play an importance here.



(Figure A2. The Loss Curves in SoftPlus with different weight methods)

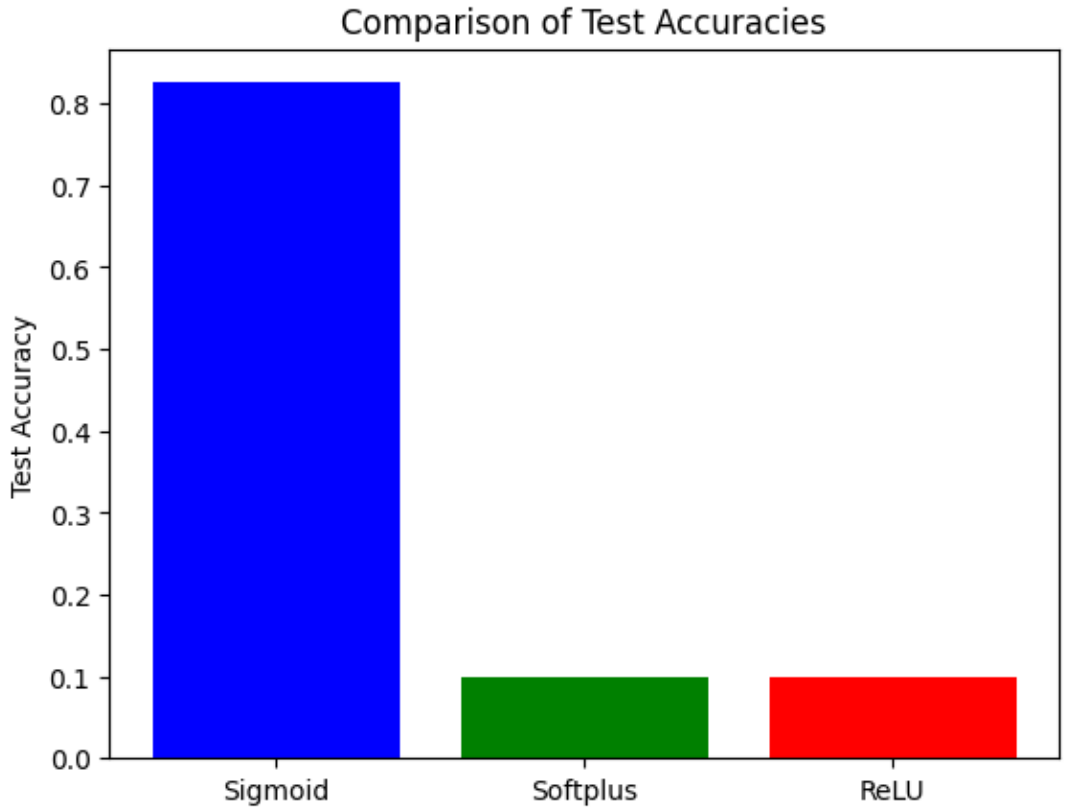
3. Performance of Different Activation Functions

Additional informations are shown here. The following cases show the loss curve and accuracy performance when sigmoid is using relatively high learning rate which is 0.01, while the relu and softplus are using relatively low learning rate which is 0.0001. The outputs are reasonable showing the expected trend.

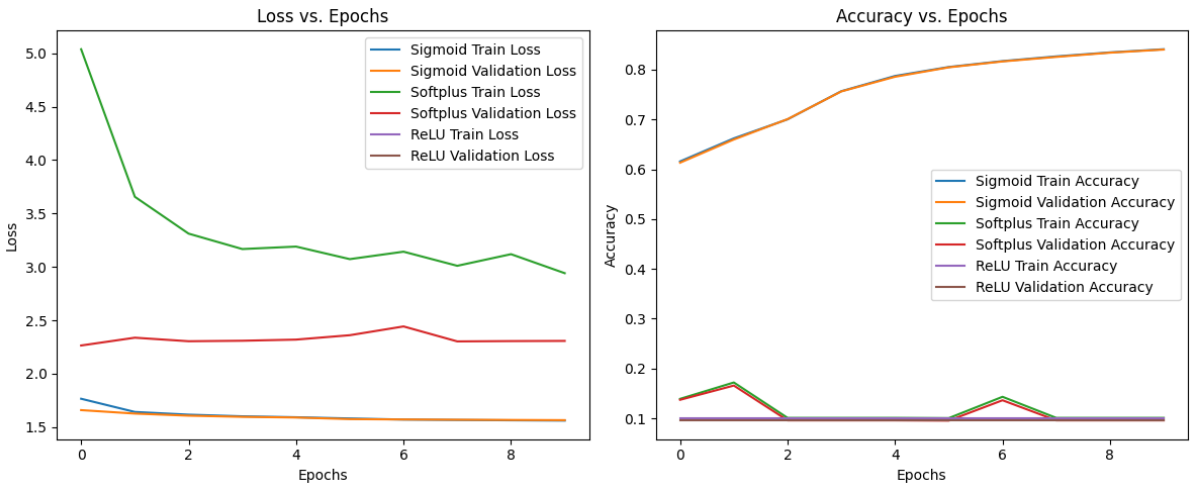


(Figure A3.1 The Loss and Learning Curves of different activations with valid LR)

The below two graphs show when the learning rates are inappropriately selected, the performance of all activations is hugely different. Because Softplus and ReLU are sensitive to learning rate. Using the same learning rate of 0.01 for the three models shows the worse performance like below.



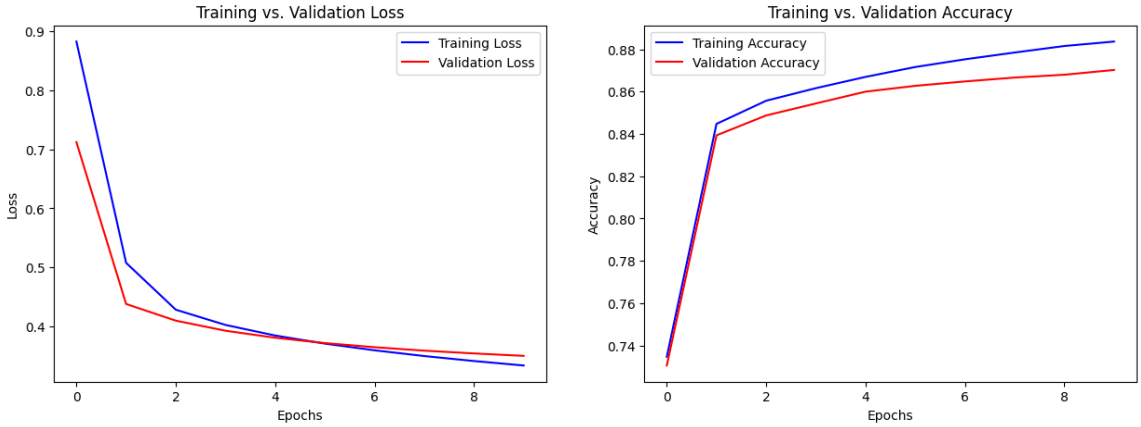
(Figure A3.2 The Test Accuracy in different activations with inappropriate LR)



(Figure A3.3 The Loss and Learning Curves of different activations with inappropriate LR)

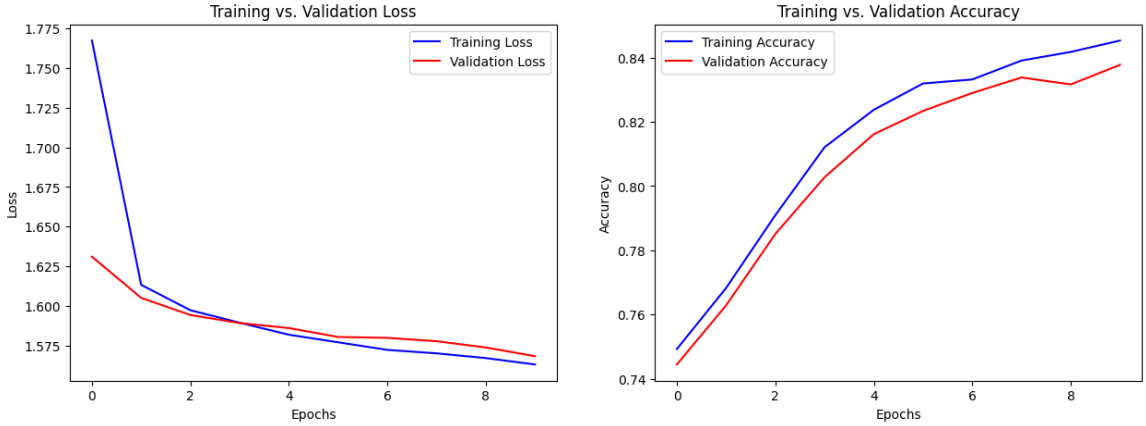
4. Best Learning Curves of Each Activation

Relu



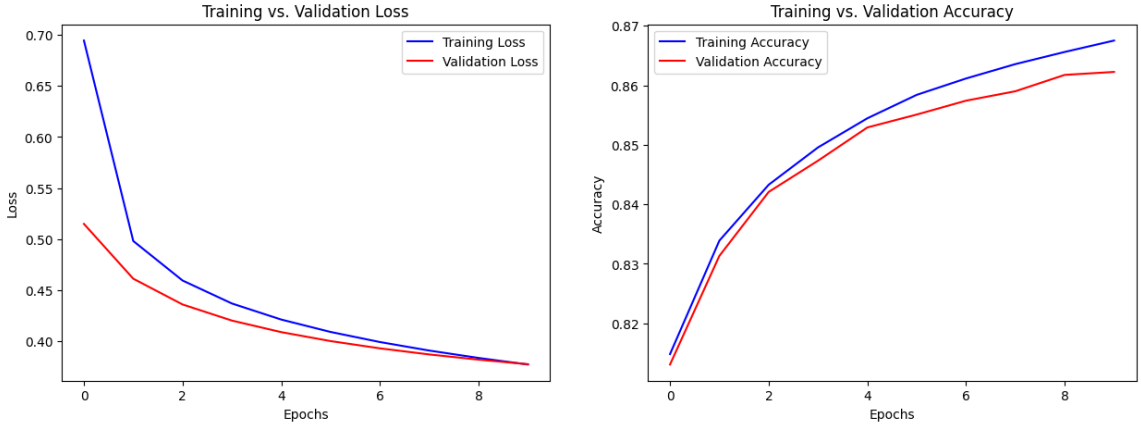
(Figure A4.1 The loss and learning curve of Relu using best parameters)

Sigmoid



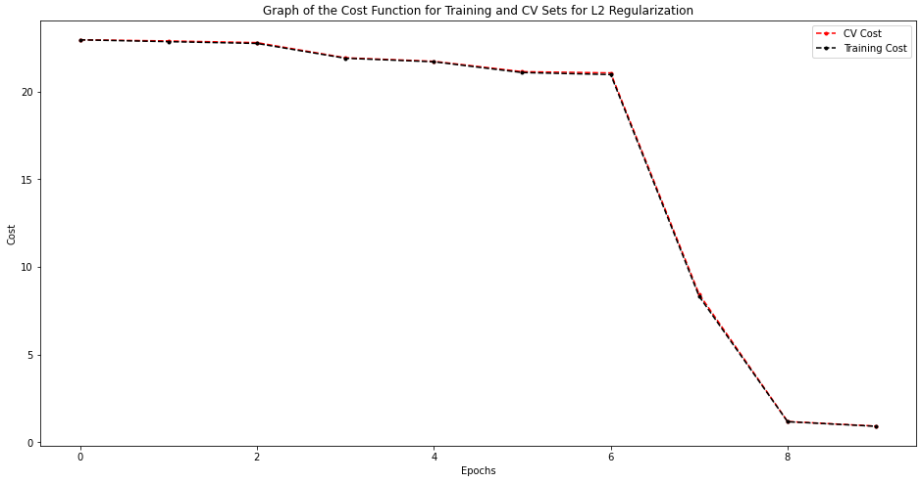
(Figure A4.2 The loss and learning curve of Sigmoid using best parameters)

SoftPlus

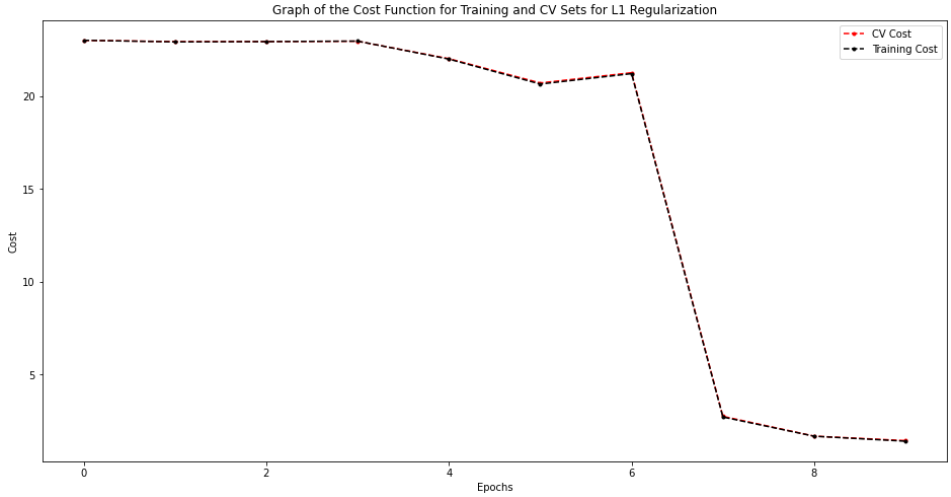


(Figure A4.3 The loss and learning curve of SoftPlus using best parameters)

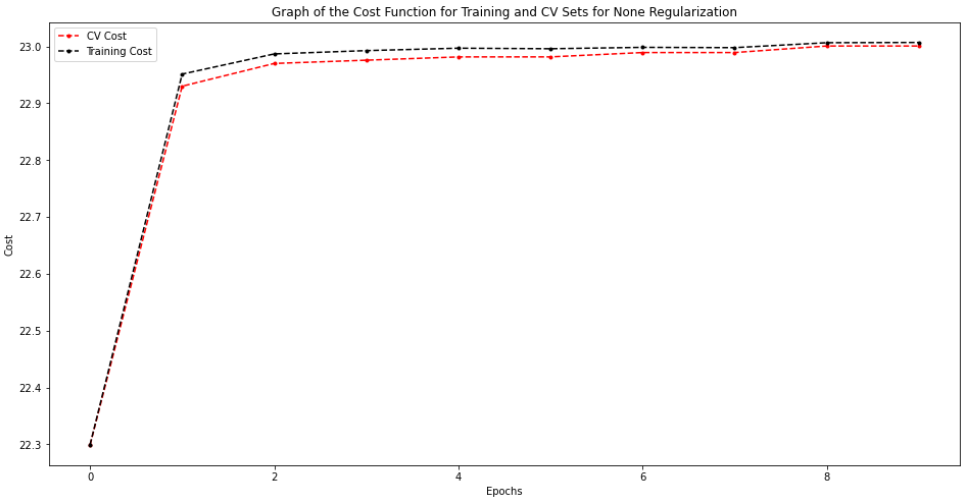
4. Effects of regularization



(Figure A4. 1 Mean loss curves for L2 regularization. 0.806 Accuracy. ReLU activation)

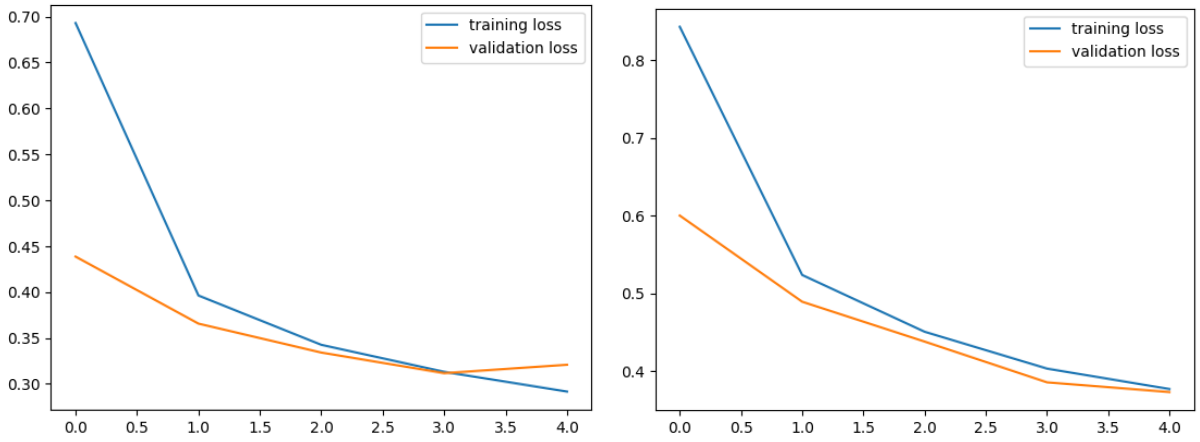


(Figure A4. 2 Mean loss curves for L1 regularization. 0.717 Accuracy. ReLU activation)

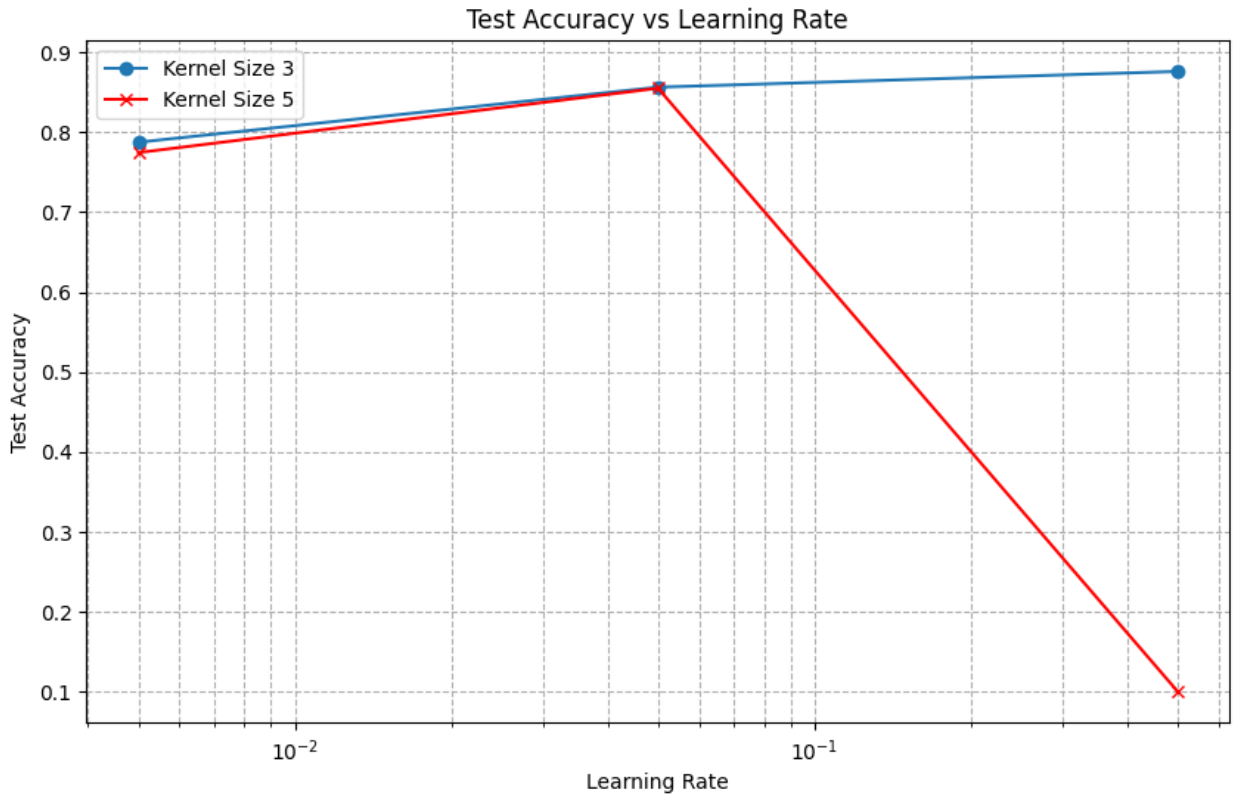


(Figure A4. 3 Mean loss curves for no regularization. 0.101 Accuracy. ReLU activation)

B. CNN



(Figure B1.1 Loss Curves for CNN trained on Fashion MNIST for differing learning rates. LR = 0.5 (left) and LR = 0.05 (right))



(Figure B1.2 Testing different hyperparameters against test accuracy. We choose the most appropriate learning rate and kernel size from this)

B2. CNN vs MLP on CIFAR-10

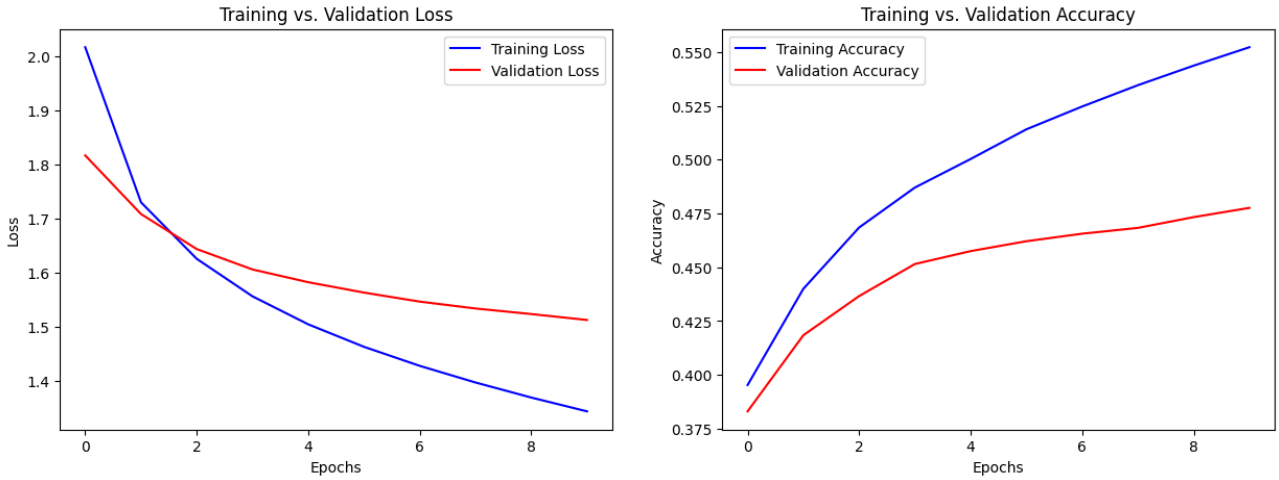


Figure B2.1 Loss curves and training + CV accuracies graphs of MLP

B3. SGD vs Adam Optimizations

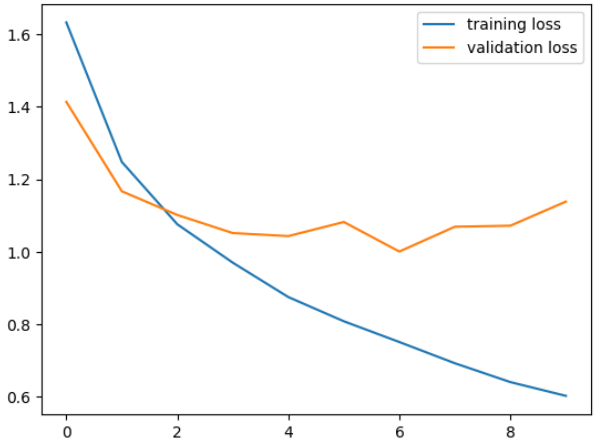


Figure B3.1 Loss curves for training and CV graph for SGD optimization with momentum = 0.3

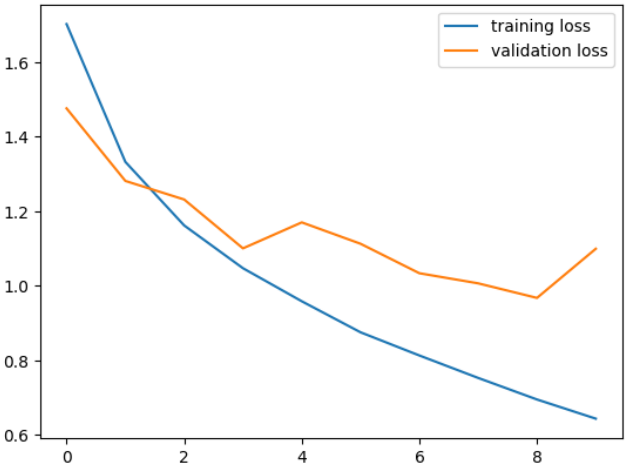


Figure B3.2 Loss curves for training and CV graph for SGD optimization with momentum = 0.6

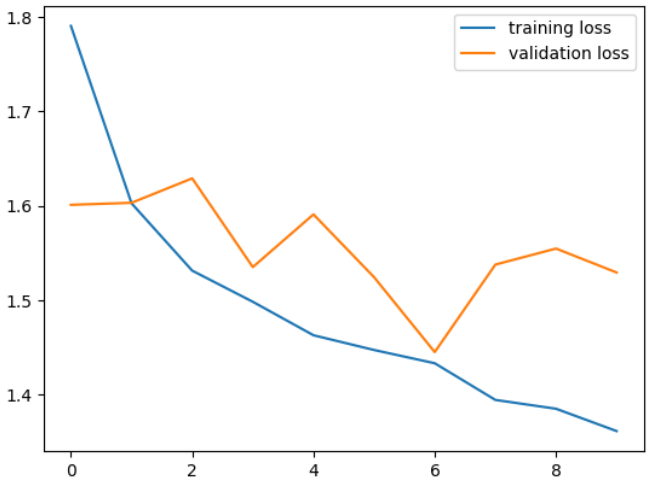


Figure B3.3 Loss curves for training and CV graph for SGD optimization with momentum = 0.9
Note: For all 3 graphs, the Y-axis represents the loss value and X-axis represents the epochs

B4. Extras: Pre-Trained Models on CIFAR-10 (ResNet18)

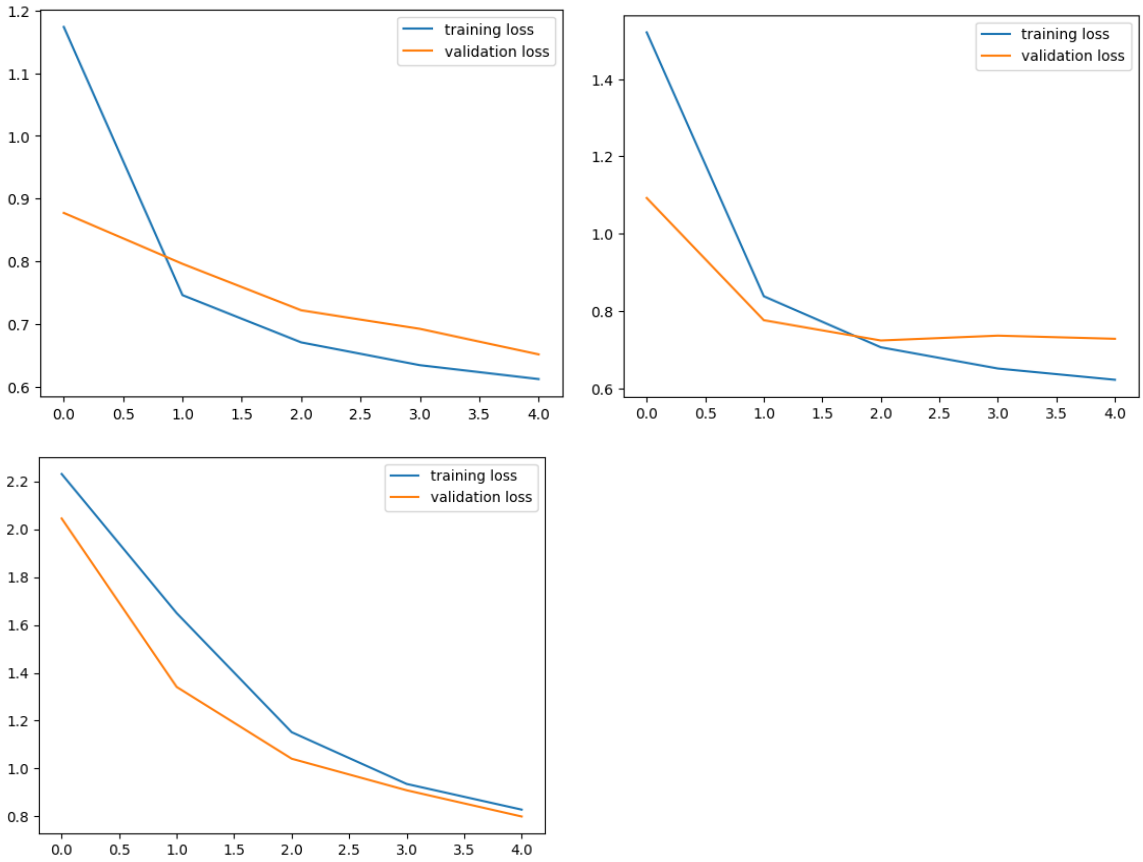


Figure B4.1 Graphs of the loss curves for ResNet18 with zero, one, and three hidden layers respectively (counting from left to right and up down). The Y-axis represents the loss value, and the X-axis the number of epochs