MiniProject 2: Classification of Image Data with Multilayer Perceptrons and Convolutional Neural Networks

Jiahang Wang, Li Jiang, Liting Chen

Abstract

This project explored the efficacy of various neural network configurations and optimization strategies in the domain of image classification, utilizing the Fashion MNIST and CIFAR-10 datasets as testbeds. Critical parameters under investigation included weight initialization techniques, model depth, model width, activation functions, regularization, kernel sizes, and optimization choices. Key findings highlighted the distinct advantage of Kaiming initialization for weight setting. Moreover, deeper architectures equipped with ReLU activations consistently outperformed shallower counterparts. The comparison between Multilayer Perceptrons (MLPs) and Convolutional Neural Networks (CNNs) underscored CNNs' superiority in handling image data, with notable performance improvements on both datasets. Among optimization techniques, while momentum-enhanced SGD exhibited top-tier results, the adaptability of the Adam optimizer was also commendable. Among different kernel sizes, the 3×3 performs best in our testing. The outcomes emphasize the criticality of informed architectural and parameter choices in neural network-driven image classification tasks.

1 Introduction

Deep learning architectures, notably neural networks, have displayed immense potential in a plethora of applications, especially in the realm of image classification. This project delved into the intricacies of various facets of neural networks, ranging from weight initialization strategies and depth influences to the importance of activation functions, regularization methods, and optimization choices. Our experimentation canvas comprised two primary datasets - the Fashion MNIST [Xiao et al., 2017] and CIFAR-10 [Krizhevsky et al., 2009]. Both datasets, while catering to the realm of image classification, offer different challenges and characteristics, enabling a comprehensive exploration.

The Fashion MNIST dataset, a benchmark in image classification tasks, revealed insightful outcomes. One of the salient findings was the clear superiority of Kaiming initialization in the context of weight initialization, outpacing other strategies and demonstrating high test accuracy. As we explored model depth, width, and the role of non-linearity, it was evident that wider and deeper models equipped with non-linear activations, specifically ReLU, showcased heightened performance. The comparative study between Multilayer Perceptrons (MLPs) and Convolutional Neural Networks (CNNs) on the same dataset accentuated the strengths of CNNs in handling image data, with CNNs achieving a test accuracy of nearly 90%, considerably outperforming MLPs. Turning our attention to the CIFAR-10 dataset, a more intricate image dataset, the dominance of CNNs was further solidified. Their adeptness at extracting complex image features was reflected in a training accuracy surpassing 73% and a test accuracy reaching above 70%, once again highlighting their advantage over MLPs. The analysis on the CIFAR-10 dataset revealed that a kernel size of 3×3 in CNNs offers optimal performance. Finally, the optimizer choices provided a nuanced understanding of the learning dynamics. While momentum proved to be a catalyst in SGD's performance, the Adam optimizer, known for its adaptability, emerged as a strong contender, slightly trailing high-momentum SGD but delivering commendable results.

2 Related work

Fashion MNIST Fashion MNIST has been widely adopted as a benchmark for evaluating machine learning algorithms' performance on image classification tasks. It has been especially appreciated as a more challenging alternative to the original MNIST dataset [Song et al., 2022]. Various studies have explored different machine learning and deep learning architectures to achieve high accuracy on this dataset. For instance, Huang et al. [2021] developed a novel personalized Cross-Silo federated learning method and validated it using Fashion MNIST. Evaluated on this dataset, Kadam et al. [2020] showed that CNN achieved high accuracy results on image classification.
CIFAR-10 CIFAR-10 dataset is designed for developing unsupervised and supervised algorithms for image classification. The dataset's complexity, arising from higher-resolution colored images, has made it a challenging and realistic benchmark for image classification tasks. Numerous works have analyzed various neural network architectures and training algorithms on CIFAR-10, providing valuable insights into optimizing model performance on complex image datasets [Berthelot et al., 2019]. A considerable amount of research has leveraged the CIFAR-10 dataset to explore unsupervised learning algorithms. For instance, Van Gansbeke et al. [2020] introduced SCAN, which achieved a notable increase in classification accuracy on CIFAR-10, marking a significant advancement in unsupervised learning.

3 Datasets

3.1 Dataset 1: Fashion MNIST

The Fashion MNIST dataset contains grayscale images of various clothing items, segmented into ten categories. Each image, as demonstrated in Figure 1, is represented with a resolution of 28x28 pixels and labeled with corresponding categories, such as shoes, shirts, and bags, identified by unique numeric codes ranging from 0 to 9. The pixel distribution of these images, highlighted in the figure for Picture 3 (a dress), indicates the pixel intensity values' spread. Most of the pixel values cluster around -1, representing the background (or absence of clothing), with a secondary peak closer to the positive range, which corresponds to the actual apparel. This bimodal distribution is consistent across images and reflects the inherent contrast between the clothing items and the background, assisting in data-driven tasks such as classification and pattern recognition.

3.2 Dataset 2: CIFAR-10

The CIFAR-10 dataset comprises 60,000 color images spread across 10 distinct classes. Each class is represented by 6,000 images, all of which have a resolution of 32x32 pixels. These images are divided methodically: 50,000 serve as the training set while the remaining 10,000 are for testing. For ease of processing, the training set is further subdivided into five batches, each containing 10,000 images. It's noteworthy that while the test batch maintains an even distribution with 1,000 images from every class, the training batches might display some variability, with certain classes being more predominant than others. However, in aggregate, the training data ensures a balanced representation with 5,000 images from each category. The classes within the CIFAR-10 dataset encompass a diverse range of objects, each providing unique challenges and insights for model training. This dataset, with its intricate composition and varied visual content, has been instrumental in advancing research in image classification and understanding model behaviors on multi-class problems. As illustrated in Figure 2, the dataset encapsulates a myriad of scenes, from fauna like dogs and cats to mechanical constructs like ships and trucks. Analyzing the pixel distribution for an exemplary image, a ship, we observe a wide spectrum of pixel intensity values, indicative of the rich, diverse content in the dataset. Given its comprehensive nature and diverse content, CIFAR-10 stands as a robust benchmarking tool for researchers in the realm of image recognition and processing.

4 Results

4.1 Analysis of Weight Initialization Strategies

The impact of weight initialization in neural networks is a critical aspect that can govern the rate of convergence, and even determine whether the model converges at all. To analyze it, we create several MLPs with two hidden layers each having 128 units, initializing the weights as (1) all zeros (2) Uniform [-1, 1] (3) Gaussian N(0,1) (4) Xavier (5) Kaiming. After training these models, we plot the training curves and test accuracy on the Fashion MNIST dataset in Figure 3. During training, zero initialization shows gradual, less efficient learning. Both Xavier and Kaiming initializations demonstrate consistent and efficient learning trends, with Kaiming being particularly effective. As for test accuracy, (1)Zero initialization results in the lowest performance due to redundant neuron outputs. (2) Uniform initialization offers moderate accuracy. (3) Gaussian initialization surpasses zero and uniform methods.(4) Xavier initialization, balancing gradient scales across layers, notably improves performance. (5) Kaiming initialization, optimized for ReLU activations, achieves the highest accuracy.

4.2 Analysis of Model Depth and Non-linearity

In this experiment, we conducted three iterations of our model, encompassing scenarios with 0 hidden layers, 1 hidden layer, and 2 hidden layers. The MLPs featuring hidden layers utilized ReLU activations and incorporated a Softmax layer at the end to facilitate classification. Each hidden layer consisted of 128 units. As depicted in Figure 4, the model with 0 hidden layers demonstrates the lowest test accuracy. Without any hidden layer, the model essentially becomes a linear classifier, limiting its capability to capture complex patterns in the data. Introducing a single hidden layer with 128 units and ReLU activation substantially improves the test accuracy. The model with 2 hidden layers achieves the highest accuracy. The additional layer offers the network more capacity to capture intricate data relationships. This increased depth, coupled with non-linearity, offers a richer hypothesis space for the model to explore. These results align with the understanding that deeper models with non-linear activations can better capture complex data patterns, albeit at the risk of overfitting if not regulated.

4.3 Analysis of Model Width

In this experiment, we used the 2-layer MLP of the last experiment and tested the effect of model width on the performance. As illustrated in Figure 5, as the width of the hidden layer increases from 32 to 128 units, there's a marked improvement in model performance. The test accuracy jumps from 55.25% for a width of 32 units to 73.44% for 128 units. This suggests that a more complex model with a larger number of units in the hidden layer can capture more intricate patterns in the data, leading to enhanced performance. However, further increasing the width to 256 units results in a marginal drop in accuracy to 73.11%, indicating a potential onset of diminishing returns in accuracy gains as the model width continues to grow. This could also hint at the model beginning to overfit to the training data or simply becoming overly complex without any tangible benefits in terms of generalization to the test data.

4.4 Analysis of Activation Function

We took the last model above, the one with 2 hidden layers, and created two different copies of it in which we used sigmoid and tanh as activations. After training on Fashion MNIST, we compared their test accuracies. As depicted in Figure 6, the model using the sigmoid activation function demonstrates the lowest test accuracy. This might be due to the vanishing gradient problem where, for very high or low values of the input, the gradient becomes very small, hampering the model's learning. Another squashing function, tanh compresses the outputs to the range [-1, 1]. It performs better than sigmoid but still lags behind the ReLU and Leaky ReLU functions. This can be attributed to it still being susceptible to the vanishing gradient problem, albeit to a lesser extent than sigmoid. The model using ReLU activations outperforms both sigmoid and tanh. ReLU introduces non-linearity and speeds up the training without the risk of vanishing gradients for positive values. Its simplicity and effectiveness in practice make it a popular choice. Leaky ReLU allows small negative values when the input is less than zero. The model with Leaky ReLU has comparable accuracy to the one with the regular ReLU. Leaky ReLU addresses the dying ReLU problem where some neurons might never activate and thus never update during training. If the results deviate from expectations, potential reasons could include differences in model initialization, learning rates, or even the specific data distribution of the dataset in use.

4.5 Influence of L1 and L2 Regularizations

We employed an MLP comprising two hidden layers, each with 128 units and ReLU activations. L1 and L2 regularizations were introduced independently to this architecture. As illustrated in Figure 7, the introduction of both L1 and L2 regularizations led to enhanced test accuracies of 73.70% and 73.39% respectively, relative to the non-regularized model's 71.84%. Regularizations act as a constraint on the network, preventing it from fitting the training data too closely, thereby offering better generalization on unseen data. Without regularization, the model might fit closely to the training data, making it susceptible to overfitting, especially with more complex models. L1 regularization tends to produce a model where many weight parameters are zero, effectively leading to a simpler, sparse model. L2 regularization penalizes large weights, ensuring that no particular weight has an overly dominant influence on the model's predictions. The choice between L1 and L2 often depends on the specific dataset and problem at hand, but as seen here, both can offer beneficial impacts on model performance.

4.6 The Impact of Data Normalization

Normalization plays a pivotal role in training deep learning models. It improves the convergence speed and the overall performance, allowing the model to learn from data more effectively and generalize better to new data. When trained with normalized data, the model achieves an accuracy of 86.03%, while it drops to 84.03% for the unnormalized data, as depicted in Figure 8. This suggests that normalized data helps the model train more effectively, potentially due to smoother loss landscapes and better convergence properties. A similar trend is observed with test data. The model achieves an accuracy of 85.13% with normalized data compared to 82.57% without normalization. The decrease in accuracy for the unnormalized data indicates that normalization not only aids in training but also in the model's ability to generalize to unseen data. The variances underscore the importance of preprocessing steps like normalization in model training.

4.7 Comparative Analysis: MLP vs CNN on FashionMNIST

Utilizing a similar approach, a CNN was crafted using PyTorch. This network was designed with 2 convolutional layers followed by 2 fully connected layers. The number of units in the fully connected layers was set at 128, and ReLU activation was used across all layers. The CNN was then trained on the Fashion MNIST dataset to gauge its efficacy.
As depicted in Figure 9, the MLP achieves a training accuracy of 85.49% and a testing accuracy of 83.91%. These values indicate a relatively strong performance of the MLP on this dataset, with only a minor difference between the training and testing accuracy, suggesting limited overfitting. The CNN demonstrates a training accuracy of 91.36% and a testing accuracy of 89.71%. This highlights the enhanced capability of CNNs in handling image data, with their specialized architecture that takes advantage of spatial hierarchies and local patterns in images. The disparity in accuracy between the MLP and CNN is evident.

4.8 Comparative Analysis: MLP vs CNN on CIFAR-10

In a continued exploration of neural architectures, the CIFAR-10 dataset was employed as the experimental backdrop. For this analysis, both an MLP and a CNN (keeping the architectural constraints) were trained on this dataset. Figure 10 indicates that while the MLP achieved a training accuracy of 66.80%, the CNN notably outperformed it with an accuracy of 73.18%. This significant uplift can be credited to CNNs' ability to handle image data, exploiting spatial features and intricate patterns which simple MLP architectures might overlook. Transitioning to the test dataset, the MLP's performance dwindled to a modest 51.89%. The CNN, though trained with a higher accuracy, realized a test accuracy of 70.76%. Despite a decline, CNN's performance was markedly superior to the MLP, underlining the former's robustness and adaptability to unseen data. In summation, when operating on intricate image datasets like CIFAR-10, CNNs demonstrate clear supremacy in extracting complex features and delivering commendable generalization.

4.9 Impact of Different Kernel Sizes

The selection of kernel size is a crucial aspect of CNN design, significantly influencing its capability to capture spatial features. To investigate this, different kernel sizes were tested, ranging from 1×1 to 7×7. As depicted in Figure 11, the test accuracy undergoes noticeable variations with changing kernel sizes. A 1×1 kernel achieves an accuracy of 88.92%, serving as a baseline. Upon increasing the kernel size to 3×3, there is a perceptible boost in performance, registering an accuracy of 91.2%. This improvement can be attributed to the kernel's expanded receptive field, which allows the model to recognize more complex patterns. The 5×5 kernel presents a slight decline in performance with an accuracy of 90.78%, hinting that there might be an optimal kernel size range for this dataset. Lastly, the 7×7 kernel size, despite its extensive field of view, exhibits a further dip in accuracy, settling at 90.26%. From this analysis, it is evident that while enlarging the kernel size can augment the model's ability to perceive larger patterns, there exists an optimal range beyond which the gains diminish. For the CIFAR-10 dataset, a kernel size of 3×3 appears to strike a balance, offering the most favorable performance. It is essential, however, to consider the specificities of each dataset and task before conclusively determining the optimal kernel size.

4.10 Impact of Optimizer Choices

The impact of optimizer choice on the CIFAR-10 dataset was thoroughly examined. Utilizing the SGD optimizer, momentum was initially set to zero and subsequently increased to 0.5 and 0.9 for further tests. We also evaluated the model's performance with the ADAM optimizer. As delineated in Figure 12, we observed divergent convergence patterns. The no-momentum SGD exhibited a steady decline in loss. However, integrating momentum—specifically at 0.5 and 0.9—accelerated convergence, with the latter displaying the most pronounced loss reduction. The Adam optimizer, leveraging the combined advantages of momentum and adaptive learning rates, also demonstrated swift convergence, closely matching the performance of high-momentum SGD.
Turning to test accuracies, the SGD without momentum logged an accuracy of 61.52%. The incorporation of momentum dramatically elevated this metric, with the 0.9 momentum SGD achieving a commendable 71.85%. Notably, the Adam optimizer, renowned for its self-adjusting learning rates, registered a test accuracy of 70.96%. While it slightly lagged behind the high-momentum SGD, it surpassed the other setups. In summary, the role of momentum is undeniably crucial in expediting convergence and augmenting model accuracy. Moreover, the Adam optimizer, due to its intricate learning rate modulation, stands out as a powerful optimizer choice.

5 Discussion and Conclusions

Our exploration into neural network configurations emphasized the intricate balance between architecture design and optimization for effective image classification. Kaiming initialization and CNNs showcased notable advantages, indicating their suitability for such tasks. While momentum-enhanced SGD performed commendably, Adam's adaptability cannot be overlooked. The divergent performance across datasets underscores the importance of context in choosing network parameters. In conclusion, while specific configurations might outshine others in certain scenarios, the paramount importance lies in understanding dataset characteristics and aligning them with network design and optimization strategies for optimal performance. We report the hyperparameters in the appendix.
In this project, Jiahang Wang and Li Jiang primarily concentrated on the coding aspects, while Liting Chen was chiefly responsible for drafting this report.

References

David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. *Advances in neural information processing systems*, 32, 2019.

Yutao Huang, Lingyang Chu, Zirui Zhou, Lanjun Wang, Jiangchuan Liu, Jian Pei, and Yong Zhang. Personalized cross-silo federated learning on non-iid data. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 7865–7873, 2021.

Shivam S Kadam, Amol C Adamuthe, and Ashwini B Patil. Cnn model for image classification on mnist and fashion-mnist dataset. *Journal of scientific research*, 64(2):374–384, 2020.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. Learning from noisy labels with deep neural networks: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Scan: Learning to classify images without labels. In *European conference on computer vision*, pages 268–285. Springer, 2020.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

A Appendix

A.1 Hyperparameters

For MLP

Parameter	**Value**
Hidden Layers	2
Units per Layer	128, 128
Activation Function	ReLU
Initialization	Kaiming
Regularization	L2
Learning Rate	1e-1
Epochs	1000
Batch Size	64

For CNN

Parameter	**Value**
Learning Rate	0.01
Optimizer	SGD
Momentum	0.9
Loss Function	CrossEntropy
Batch Size	64
Number of Epochs	5
Conv1 In Channels	1
Conv1 Out Channels	32
Conv1 Kernel Size	3x3
Conv2 Out Channels	64
Conv2 Kernel Size	3x3
Pooling Kernel Size	2x2
FC1 In Features	64 * 7 * 7
FC1 Out Features	128
FC2 Out Features	10

A.2 Experimental Plots

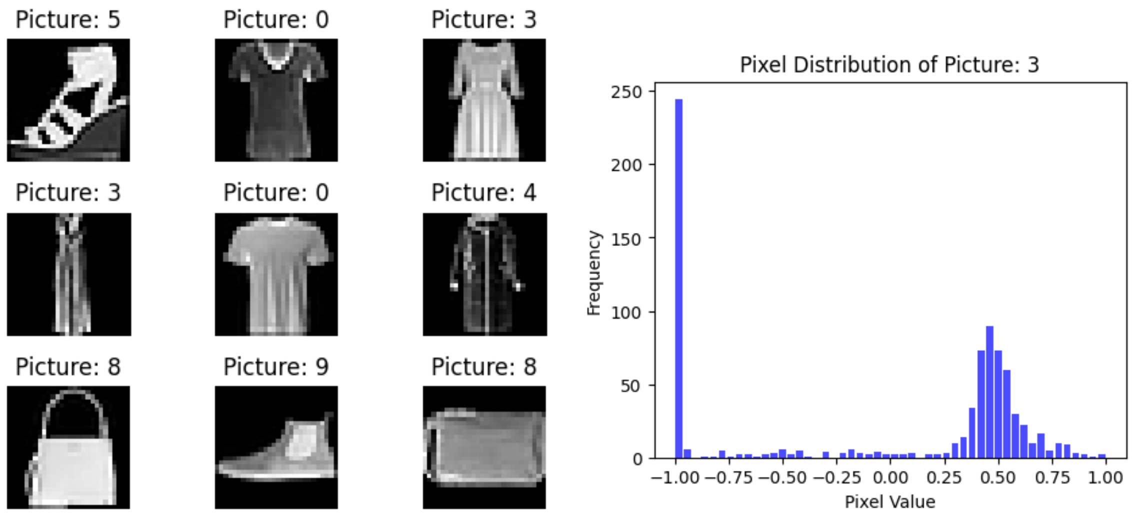


Figure 1: **Left:** Sample pictures **Right:**Pixel Distribution of Picture: 3

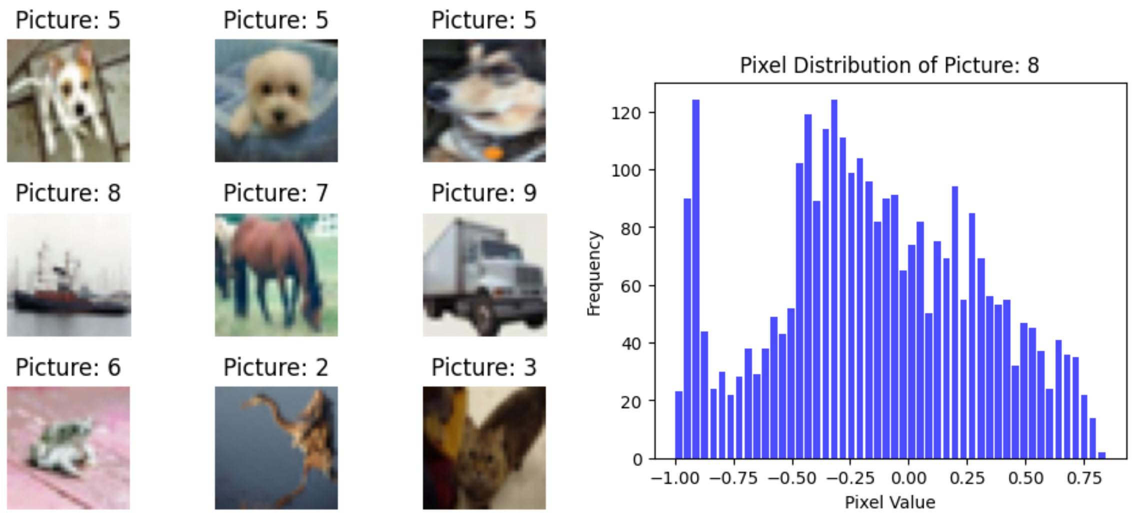
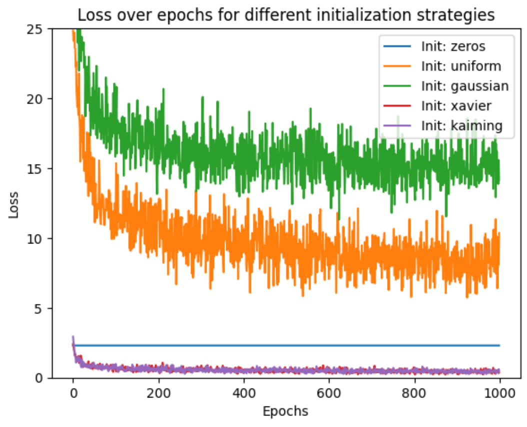


Figure 2: **Left:** Sample pictures **Right:**Pixel Distribution of Picture: 8



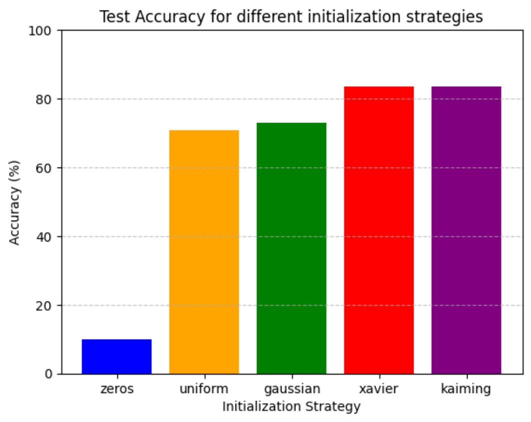


Figure 3: **Left:** Loss over epochs for different initialization strategies **Right:**Test Accuracy for different initialization strategies

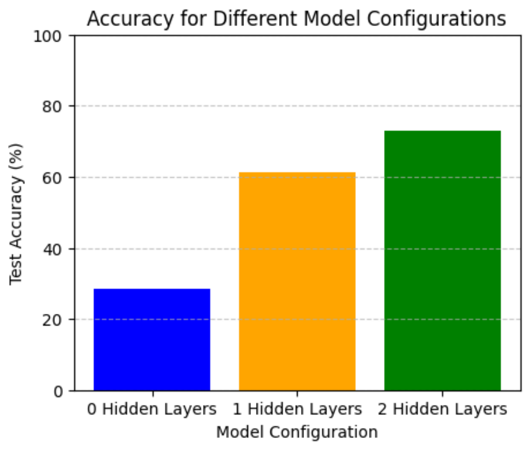


Figure 4: Accuracy for models with different depths

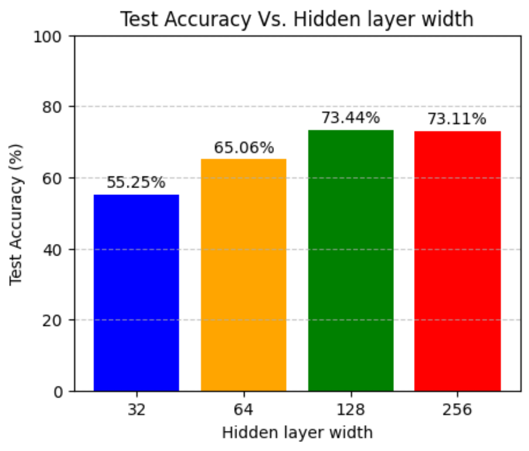
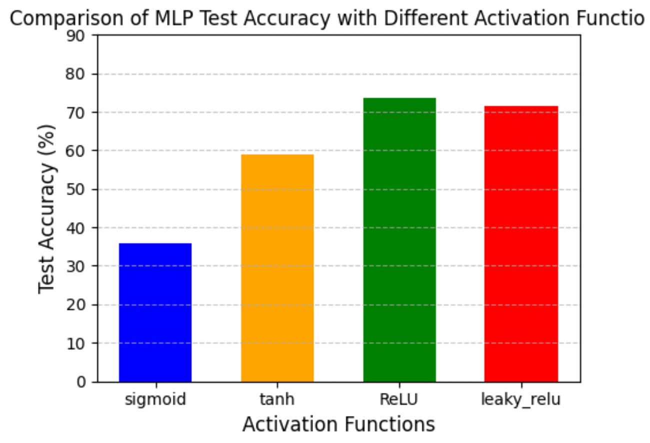


Figure 5: Accuracy for models with different widths



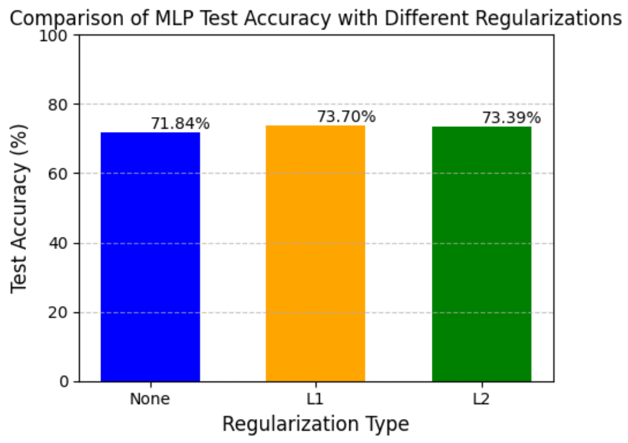


Figure 6: Comparison of MLP test accuracy with different activation functions

Figure 7: Comparison of MLP test accuracy with different regularizations

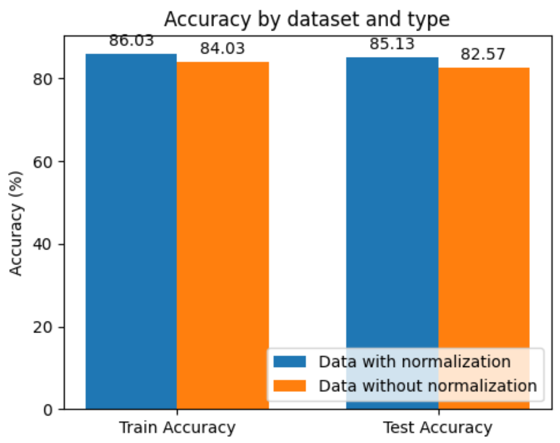


Figure 8: Accuracy by dataset and type

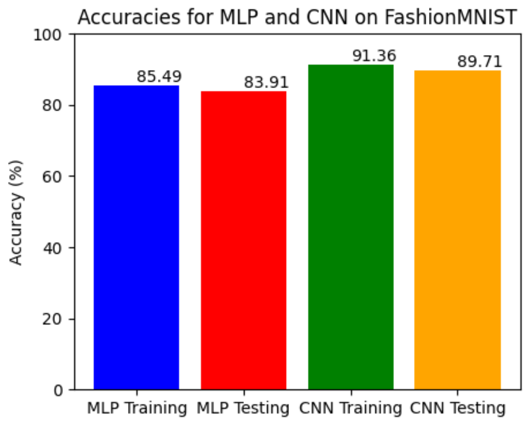


Figure 9: Accuracies for MLP and CNN on FashionMNIST

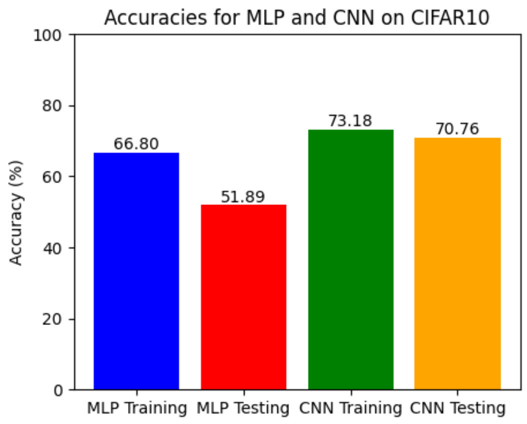
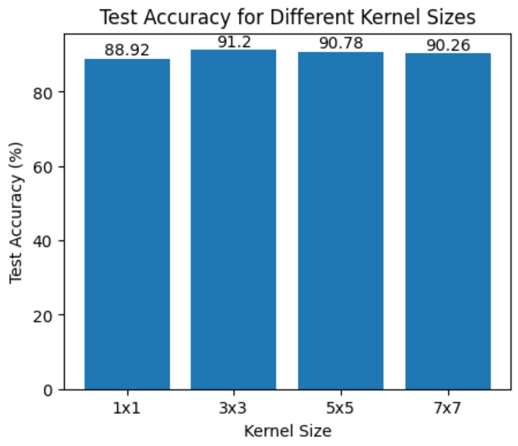


Figure 10: Accuracies for MLP and CNN on CIFAR10



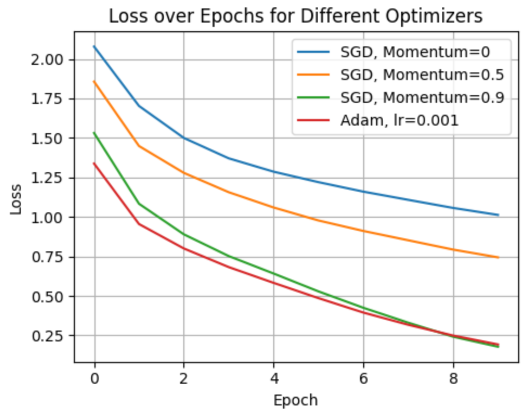


Figure 11: Test Accuracy for Different Kernel Sizes

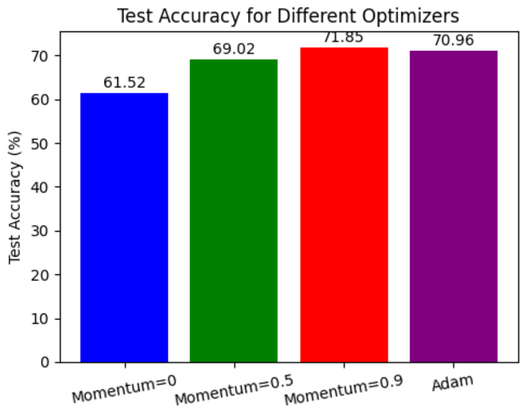


Figure 12: **Left:** Loss over epochs for different optimizes **Right:** Test Accuracy for different optimizers