COMP551 Mini-Project 2 Report

Joey Koay (260956108), Selina Wang (260978208), Estelle Lin (260949118)

October 31, 2023

Abstract

This research focuses on training neural networks for image classification. Our primary focus is on understanding and experimenting with various aspects of multilayer perceptrons (MLPs) and convolutional neural networks (CNNs) in the context of image classification. We explore how different training decisions affect the performance of these networks, leveraging both the Fashion MNIST and CIFAR-10 datasets.

Keywords
Machine Learning, Image Classification, Multilayer Perceptrons, Convolutional Neural Networks

1 Introduction

Image classification stands as a pivotal task in the realm of machine learning, necessitating a profound understanding of various neural network architectures and training methodologies. In this project, we delve into the intricacies of multilayer perceptrons (MLPs) and convolutional neural networks (CNNs), two prevalent architectures in the domain of deep learning, aiming to decipher how different training decisions impact their performance in image classification tasks. We employ the Fashion MNIST and CIFAR-10 datasets, which are widely recognized benchmarks in the field, used in numerous studies for evaluating machine learning models due to their complexity and variety in image content. Fashion MNIST and CIFAR-10 serve as critical benchmarks in machine learning, with Fashion MNIST offering a challenging environment for algorithm evaluation [1], and CIFAR-10 providing foundational insights for convolutional neural networks in image classification [2]. Brown et al. (2020) further emphasized the importance of optimization strategies across these datasets, highlighting their significant impact on model performance [3].

The most important finding of this project highlights the distinct behaviors and performance of MLPs and CNNs in image classification tasks, using the two datasets. The experiments demonstrated that while MLPs can be optimized for simpler tasks with appropriate adjustments in layers and activation functions, CNNs distinctly outperform in handling complex image datasets like CIFAR-10, showcasing their superior ability to capture spatial hierarchies and local patterns. The project also underscores the critical role of data normalization and the selection of suitable weight initialization methods, as well as the influence of momentum values in optimizers, in enhancing the models' learning efficiency and performance.

2 Datasets

2.1 Preprocessing Dataset

In both the Fashion MNIST and CIFAR-10 datasets, we did not remove any data points. Instead, we created a function that allows us to load the desired dataset, normalize, flatten, and turn the labels from base 10 numbers to one hot encoded label. For normalization, we followed a CS231n [4] (a Standford University class) data preprocessing methodology as suggested in the assignment outline. After obtaining the data, a mean subtraction is conducted. This process centers the data by making its mean closer to zero. Then, the standard deviation is calculated, which is then used to divide each data. This can help make the features more comparable when they have different scales and are located more closely to each other. This helps the models converge faster and improve their performance.

2.2 Fashion MNIST

The Fashion MNIST dataset consists of 60,000 data in the training set, and 10,000 data in the testing set. There are a total of 10 different labels, and each image is 28x28. Figure 7 helps visualize the data.

2.3 CIFAR-10

The CIFAR-10 dataset consists of 50,000 data in the training set, and 10,000 data in the testing set. There are a total of 10 different labels, and each image is 32x32. Figure 8 helps visualize the data.

3 Results

3.1 Different Weight Initialization

One of the many controllable parameters in MLP is the weight initialization. It determines how in-

formation is processed and transformed, thereby, allowing the MLP to capture patterns in data better. Each connection between neurons is associated with a weight, which determines the strength of the connection. As seen Figure 1, the best accuracy was produced by initializing the weights as Xavier or Kaiming. Whereas, if we initialize all the weights as zero, we are effectively "disconnecting" these neurons in the networks (in certain activation functions), therefore suggesting that none of the input is relevant.

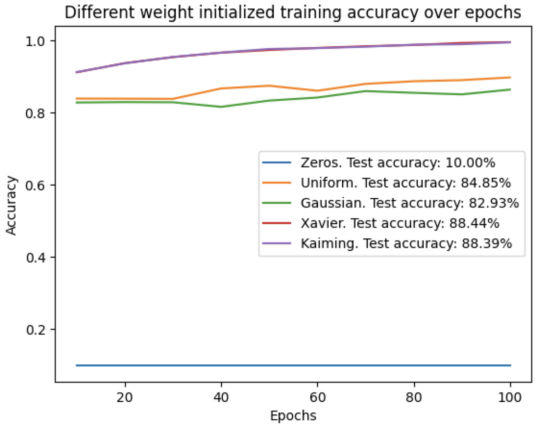

Figure 1: Result's accuracy with respect to different weight initialization (Fashion MNIST dataset)

3.2 Different Number of Hidden Layers

Another aspect of MLP that is controllable is the amount of hidden layers initialized. More hidden layers increase the model's ability to capture complex patterns and relationships in the data. However, it is computationally more intensive, requiring more time to train, and it needs more training data to perform well as it is prone to overfitting. Given that the Fashion MNIST dataset has 60,000 data, the a between zero and two hidden layers employed, there should be a higher accuracy associated to a larger amount of hidden layers. As seen in Figure 2, it is true.

3.3 Different Activation Function

When there are hidden layers in an MLP, we can employ activation functions to help decide which neuron should be activated or produce an output, and to what extent based on the weighted sum of its input. Some activation functions include identity, logistic, hyperbolic tangent, relu, leaky relu and softplus. For this experiment specifically, the hyperbolic tangent and leaky relu was employed to compare its performance. Each activation function has its own advantages and disadvantages. One problem that occurs is the "vanishing gradient" problem when the program reaches a state where nothing is being updated. When gradients are consistently zero, the weights of the associated neuron won't get updated during training, essentially rendering it non-responsive to learning. To prevent the program from entering such a state, activation functions like leaky ReLU will ensure that there will always be a non-zero output for every non-zero input. Figure 3 behaves as follows as hyperbolic tangent favors highly weighted inputs by rewarding them with even higher weighted outputs, whereas leaky relu is a linear relationship. Therefore, at lower epochs which is at a lower accuracy, the leaky relu model has a higher accuracy than the hyperbolic tangent model. Whereas, at higher epoch, which is at a higher accuracy, the leaky ReLU model has a higher accuracy

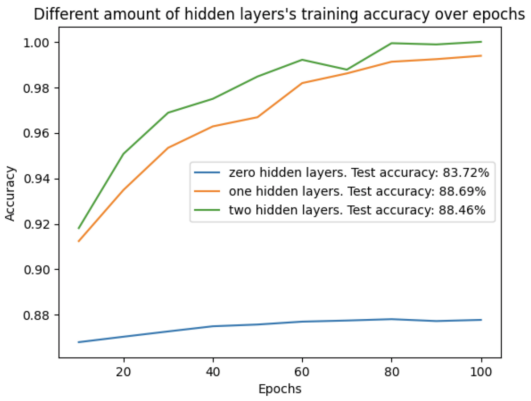

Figure 2: Result's accuracy with respect to different amount of hidden layers (Fashion MNIST dataset)

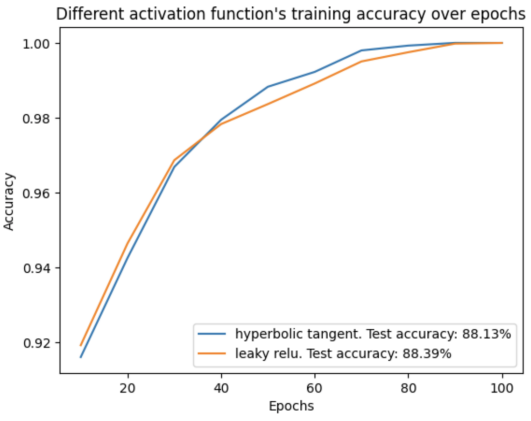

Figure 3: Accuracy comparison between Leaky Relu and Hyperbolic Tangent Activation Function)

3.4 L1 and L2 Regularization

Figure 4 demonstrates that the MLP achieves the highest accuracy without regularization, surpasses its performance with L2 regularization, and falls short when L1 regularization is applied. The performance disparity between different regularization

techniques in MLP for the Fashion MNIST dataset underscores the importance of model and data compatibility.

The dataset's simplicity allows the MLP to generalize well without regularization, leading to the highest accuracy. L2 regularization, which penalizes large weights, results in a slight performance dip, indicating that the model's complexity is appropriate for the task and does not overfit. In contrast, L1 regularization enforces sparsity in weights and leads to a significant accuracy drop, suggesting that it may be overly restrictive and discard crucial features necessary for accurate predictions. This demonstrates that the choice and tuning of regularization methods should be meticulously done, considering the specific traits of both the dataset and the model.

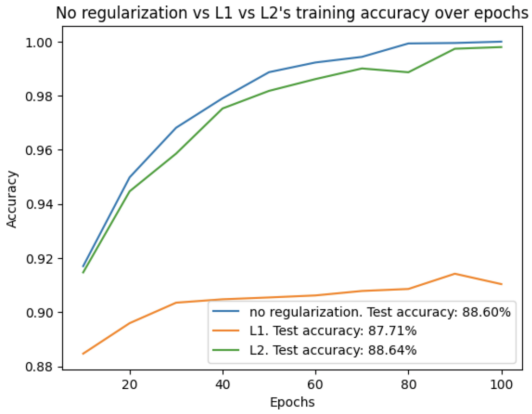

Figure 4: Accuracy comparison between no regularization, L1 regularization and L2 regularization)

3.5 Normalized vs unnormalized images

From Figure 5 we see that MLP with unnormalized images would cause a significant drop in accuracy. As mentioned in 2.1, normalization is an essential aspect of processing data as it can ensure that all pixel values are on a consistent scale, leading to faster and more stable training. This process mitigates the risk of issues such as vanishing or exploding gradients. In essence, normalization balances the input features, accelerates the convergence of the training process, and ultimately results in better model performance.

3.6 CNN with two convolutional and two fully connected layers

In this convolutional neural network experiment tailored for grayscale images, the first convolution layer has one input channel, 16 output channels with a 3x3 kernel, padding of 1, and a default stride of 1 to preserve spatial dimensions. The second convolution layer takes 16 input channels from the previous layer, has 32 output channels, and maintains the same kernel size, padding, and stride. Pooling layers follow each convolution to mitigate computational complexity under Google Colab's RAM constraints.

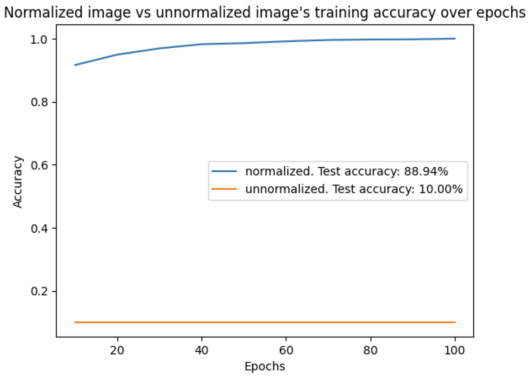

Figure 5: Accuracy comparison between normalized and unnormalized image)

Subsequently, the network transitions to fully connected layers: the first takes 1568 flattened features from the previous layer, outputting 128 layers, while the second transforms 128 input features into 10 outputs. Rectified Linear Unit (ReLU) activation follows each convolutional and the first fully connected layer, introducing non-linearity to capture complex patterns without altering data dimensions, ensuring efficient network performance within the available computational resources.

CNNs typically outperform MLPs in image classification due to key advantages. Firstly, CNNs employ local connectivity with convolutional filters, capturing spatial hierarchies and local patterns, whereas MLPs, connecting all neurons across layers, can miss these intricacies. Secondly, CNNs' parameter sharing reduces the total parameters, aiding generalization and curbing overfitting, unlike MLPs with non-shared parameters that can inflate with image dimensions. Lastly, while CNNs effectively utilize depth to discern complex features, deepening MLPs can pose training challenges due to the parameter surge.

As shown in Figure 9, CNN have accuracies of over 90 percent, however, it is worth noticing that the MLP seems to slightly outperform CNN in this case. This unexpected scenario could be due to the dataset's simplicity, which may favor the less complex MLP, reducing the risk of overfitting. The CNN's performance might be hampered by a suboptimal architecture, inadequate hyperparameter settings, or insufficient training duration. This suggests a need for further investigation and optimization of the CNN model for this specific task.

3.6.1 Additional experiments

In the conducted experiments, varying the hyperparameters of a Convolutional Neural Network (CNN) showcased distinct impacts on model performance. Increasing the number of filters enhanced the model's ability to capture diverse features from the input, improving accuracy (see Figure 10), though this could potentially lead to overfitting if taken to an extreme [5]. Larger kernel sizes initially benefited the model by encapsulating more global information, but this advantage plateaued, highlighting a balance between capturing broader patterns and potentially losing fine-grained details [6] (see Figure 11). Strides, when increased, reduced the spatial resolution of the feature maps [7], resulting in a drop in accuracy due to the loss of spatial information, showcasing a trade-off between computational efficiency and model performance (see Figure 13). Padding's incremental addition preserved spatial dimensions and considered border information, leading to improved accuracy (see Figure 12), yet there's a cautionary note on potential noise introduction with excessive padding [8]. Optimally tuning these parameters necessitates a careful balance, tailored to the specific dataset and task at hand. From Figure 14, the graph depicts the relationship between the size of the training dataset (power of 10) and the resulting test accuracy. As the number of training images increases, the test accuracy also rises. Initially, with fewer images, the model struggles to generalize well, resulting in lower accuracy. However, as more data is introduced, the model becomes better equipped to capture underlying patterns and generalize to unseen data, leading to higher accuracy. The trend aligns with the common understanding that larger training datasets typically enable machine learning models to perform better, as they can learn more diverse features and reduce overfitting [9]. The log scale on the x-axis also emphasizes the diminishing returns of adding more data after a certain point.

3.7 CNN vs MLP for CIFAR10 Dataset

For the CIFAR10 dataset, an MLP was trained using 2 hidden layers with the Kaiming initializer, relu activation function, and no regularization. As shown in Figure 15, the performance of MLP is consistently low with the accuracy being no more than 10 percent. Meanwhile, a CNN trained using 2 convolutional and 2 fully connected layers with the Kaiming initializer and ReLU activation function is giving an accuracy of around 70 percent, as shown in Figure 16. Thus, CNN outperforms MLP significantly for the CIFAR10 dataset as discussed in task 3.6.

In the specific context of the CIFAR-10 dataset, which comprises 60,000 32x32 color images distributed across 10 classes, the advantages of CNNs become even more evident. The small dimensionality of the images in CIFAR-10 necessitates a model that can efficiently capture local patterns and hierarchies, making CNNs a more suitable choice. While MLPs require flattening the image data into a high-dimensional input vector, resulting in a loss of spatial information, CNNs process the image in its original 3D shape, preserving spatial relationships and reducing dimensionality through layers. This architectural difference fundamentally equips CNNs to provide superior performance in classifying CIFAR-10 images compared to MLPs, given proper network design and hyperparameter tuning.

3.8 Effects of Optimizer on Performance

As shown in Figure 6, the graph underscores the influence of momentum on the performance dynamics of a neural network trained via the SGD optimizer across 10 epochs. **Convergence Speed:** Momentum accumulates past gradients to guide the model's next steps, with higher values enabling faster navigation through the loss landscape, leading to quicker accuracy gains in initial epochs [10]. For instance, the trajectory for momentum 0.9 shows a notably faster rise in the early epochs compared to its counterparts, corroborating the principle that a higher momentum can expedite convergence. **Final Accuracy:** By the culmination of the 10 epochs, it is observed that networks with momentum values of 0.9 and 0.8 outperform the others in terms of accuracy. Higher momenta let the model take larger steps towards the global minimum, bypassing suboptimal local minima and potentially enhancing overall accuracy. **Stability:** While momentum hastens convergence, it might induce oscillations, especially if the gradient surpasses the minima, as observed in the initial epochs of the 0.5 and 0.6 momentum curves. Such fluctuations suggest oscillations around optimal weights, signifying reduced stability. In contrast, the 0.8 and 0.9 curves showcase stability with their smooth trajectories. Fundamentally, the graph underscores the multifaceted role of increased momentum: accelerating convergence, potentially improving final accuracy, and influencing stability.

Momentum, primarily linked with SGD, propels convergence by factoring in past gradients [11]. Meanwhile, in the Adam optimizer, Beta values determine exponential decay rates for these gradients. Particularly, Beta1 manages the decay of the first moment, akin to a gradient's moving average. This balance between past and recent gradients, governed

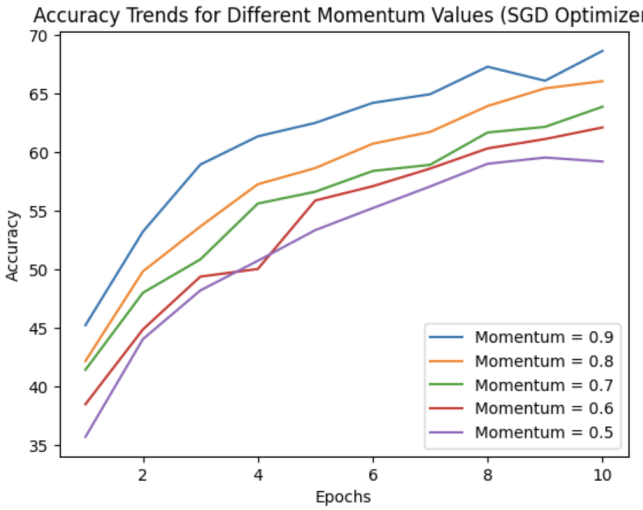

Figure 6: Accuracy Trend for different momentum value (SGD)

by adjusting Beta1, affects both the speed of convergence and the stability of the optimization. Thus Adams optimizer trend based on beta1 values was experimented with for interest.; Comparing the accuracy trends of SGD with varied momentum values to the Adam optimizer with different Beta1 values, both show faster convergence with higher parameter values. However, Adam, especially with Beta1 values of 0.9 and 0.8, exhibits a steadier increase in accuracy across epochs (see Figure 17). By the 10th epoch, both optimizers achieve higher accuracy with increased momentum or Beta1 values. Notably, the Adam optimizer's training curves are smoother, indicating a more stable learning process than the SGD, which showcases minor fluctuations, especially at lower momentums. This suggests that Adam might offer more consistent and stable training compared to SGD.

3.9 Bonus

In the conducted experiment, the model exhibited an accuracy of 60% on the CIFAR-10 dataset, as illustrated in Figure 18. When compared to the best-performing Multi-Layer Perceptron (MLP) model and a standard Convolutional Neural Network (CNN) model developed in this study, the pre-trained model underperformed by approximately 30% in terms of accuracy. This discrepancy in performance can be attributed to the utilization of the ResNet50 architecture, initially pre-trained on the ImageNet dataset. ImageNet encompasses a diverse and complex array of high-resolution images, significantly differing from the smaller, more specific images found in CIFAR-10.

In an effort to tailor the pre-trained model to the CIFAR-10 classification task, additional fully connected layers were incorporated. Despite this modification, the model's performance was hindered by the reliance on the frozen convolutional layers, which provide features not ideally suited for the CIFAR-10 dataset. Consequently, the performance of the fully connected layers was limited, resulting in a less than optimal classification accuracy.

This experimental outcome underscores the crucial balance required between leveraging pre-trained features and ensuring sufficient model adaptability when applying transfer learning across varied datasets. The findings suggest that while pre-trained models can expedite the training process and potentially enhance performance, careful consideration and adjustment are necessary to align the model with the specific characteristics of the target dataset, ensuring optimal results.

4 Discussion

Effective weight initialization in MLPs is vital, with Xavier and Kaiming methods preventing extreme gradients for optimal performance. Adding layers boosts MLP performance but also raises computational demands and overfitting risks, requiring a balanced strategy. Leaky ReLU proved more efficient than hyperbolic tangent in addressing vanishing gradients, and the simplicity of Fashion MNIST may not necessitate strong regularization.

CNNs displayed exceptional performance on the complex CIFAR-10 dataset, showcasing their strength in intricate image classification tasks and highlighting their potential. The experiments underscored the importance of data normalization and revealed the significant impact of momentum values on the convergence speed and accuracy of SGD optimizers, emphasizing the necessity for careful tuning to balance performance and stability.

Conclusions

This research offered valuable insights into neural network training for image classification, emphasizing the importance of choosing the right architecture, and initialization method, and ensuring data normalization. The findings highlight CNNs' effectiveness in complexity and MLPs' adaptability with proper adjustments.

Statement of Contributions

Joey Koay, Selina Wang, and Estelle Lin worked collaboratively on this mini-project for both the coding aspect and the writeup.

Appendix

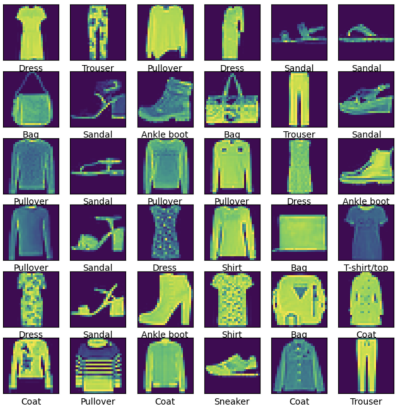

Figure 7: Visualization of Fashion MNIST

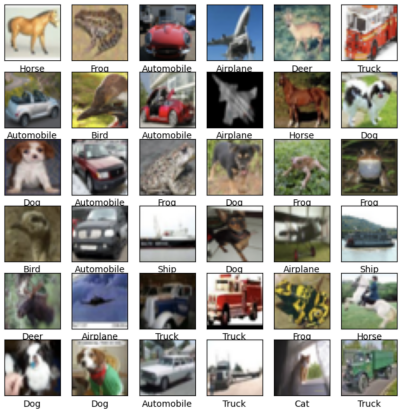

Figure 8: Visualization of CIFAR-10

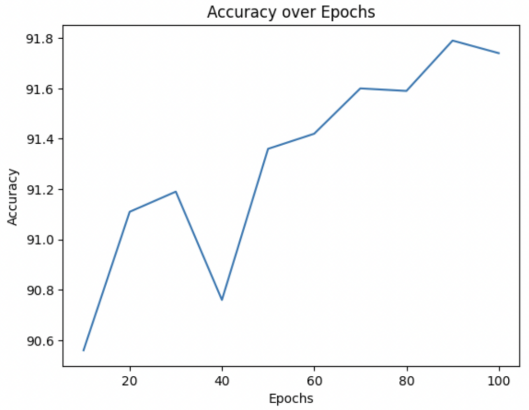

Figure 9: CNN Accuracy for Fashion MNIST Dataset)

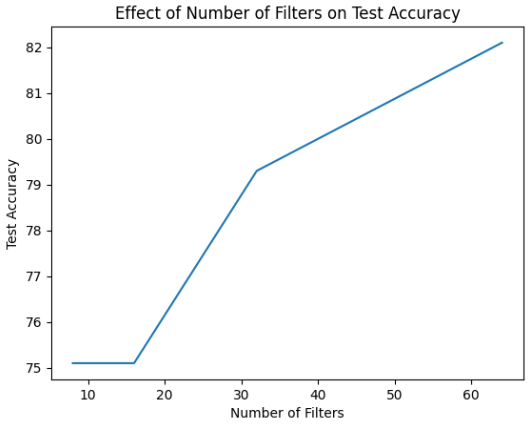

Figure 10: Impact of Number of Filters on Test Accuracy

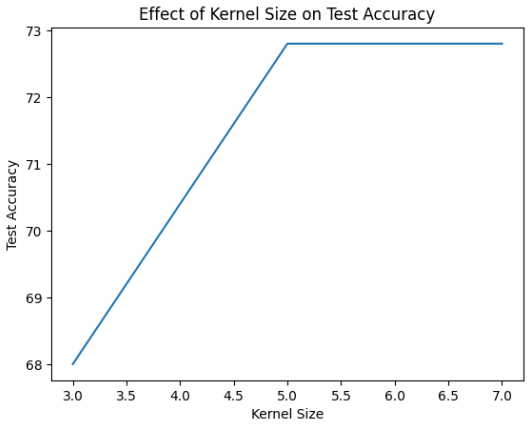

Figure 11: Impact of Kernel Size on Test Accuracy

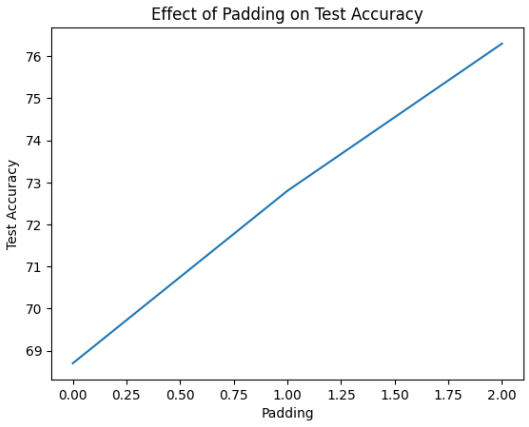

Figure 12: Impact of Padding on Test Accuracy

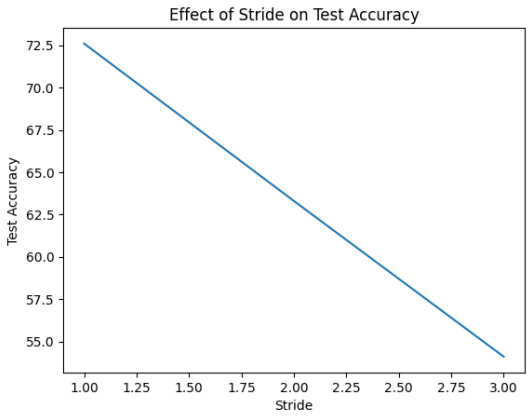

Figure 13: Impact of Stride on Test Accuracy

Figure 14: Test Accuracy of CNN Across Log-Scaled Training Set Sizes

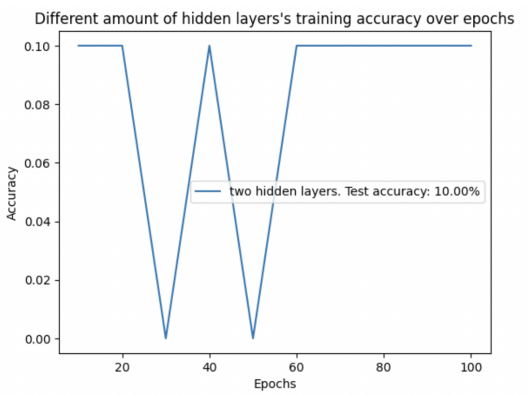

Figure 15: MLP Accuracy for CIFAR10 Dataset)

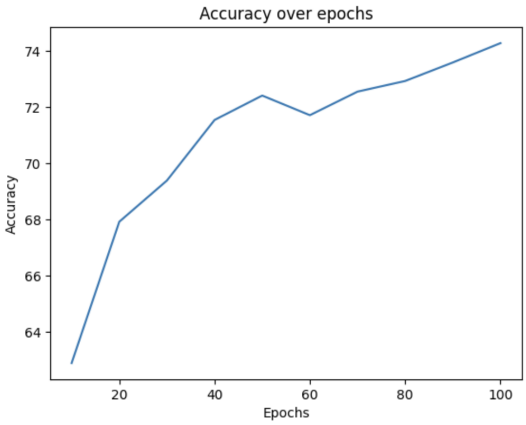

Figure 16: CNN Accuracy for CIFAR10 Dataset)

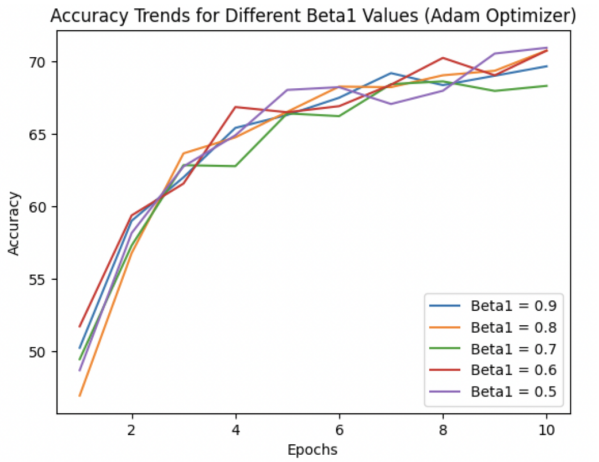

Figure 17: Accuracy Trend for different Beta value (Adam)

```
[7,   100] loss: 0.911
[7,   200] loss: 0.884
[7,   300] loss: 0.892
[7,   400] loss: 0.932
[7,   500] loss: 0.970
[7,   600] loss: 0.918
[7,   700] loss: 0.926
[8,   100] loss: 0.831
[8,   200] loss: 0.834
[8,   300] loss: 0.861
[8,   400] loss: 0.883
[8,   500] loss: 0.868
[8,   600] loss: 0.889
[8,   700] loss: 0.919
[9,   100] loss: 0.781
[9,   200] loss: 0.801
[9,   300] loss: 0.839
[9,   400] loss: 0.845
[9,   500] loss: 0.848
[9,   600] loss: 0.849
[9,   700] loss: 0.842
[10,   100] loss: 0.778
[10,   200] loss: 0.780
[10,   300] loss: 0.789
[10,   400] loss: 0.796
[10,   500] loss: 0.794
[10,   600] loss: 0.799
[10,   700] loss: 0.809
Finished Training
Accuracy of the network on the 10000 test images: 60 %
```

Figure 18: Open-ended question results (bonus)

References

[1] Xiao, H., Rasul, K., & Vollgraf, R. (2017). Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. arXiv preprint arXiv:1708.07747.

[2] Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Technical Report.

[3] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. arXiv preprint arXiv:2005.14165.

[4] Stanford University. (n.d.). Neural Networks Part 2: Setting up the Data and the Loss. Retrieved, from https://cs231n.github.io/neural-networks-2/#datapre

[5] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In Advances in Neural Information Processing Systems (pp. 1097-1105).

[6] Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv preprint arXiv:1409.1556.

[7] Springenberg, J. T., Dosovitskiy, A., Riedmiller, M., & Brox, T. (2014). Striving for Simplicity: The All Convolutional Net. arXiv preprint arXiv:1412.6806.

[8] Dumoulin, V., & Visin, F. (2016). A guide to convolution arithmetic for deep learning. arXiv preprint arXiv:1603.07285.

[9] Sun, C., Shrivastava, A., Singh, S., & Gupta, A. (2017). Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. In Proceedings of the IEEE International Conference on Computer Vision (pp. 843-852).

[10] Qian, N. (1999). On the momentum term in gradient descent learning algorithms. Neural Networks, 12(1), 145-151.

[11] Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In International Conference on Machine Learning (pp. 1139-1147)