COMP 551 Mini-Project 2: Classification of Image Data with Multilayer Perceptrons and Convolutional Neural Networks

Raphael Fontaine (261051635), Mohammad Nafar (260899098), Winnie Yongru Pan (261001758)

***Abstract* – In this mini project, we explored the domain of deep learning by implementing a Multilayer Perceptron (MLP) from scratch and comparing it with Convolutional Neural Networks (CNNs) for image classification tasks using the Fashion MNIST and CIFAR-10 datasets. The primary objective was to gain practical experience in neural network creation and understand the impact of various architectural and training decisions.**

The project was divided into three main tasks: acquiring and preprocessing the datasets, implementing the MLP from scratch, and conducting experiments to analyze and compare the performance of MLP with CNN in the context of image classification. In the first task, the datasets were loaded and preprocessed to suit the needs of the experiments. In the second task, a basic MLP was created, setting the stage for deeper insights into neural network training. The third task involved a series of experiments exploring weight initialization, activation functions, regularization, and the performance comparison of MLPs with CNNs.

The results highlighted the significance of weight initialization on training dynamics and showed a performance advantage of CNNs over MLPs within the context of image classification tasks. Throughout this project, we not only built a foundational understanding of neural network behavior but also set a stage for further exploration in deep learning.

INTRODUCTION

In the context of machine learning and image classification, deep learning emerges as a powerful tool. Deep learning, characterized by multi-layered neural networks, has significantly contributed to advancements in image classification tasks. Multilayer Perceptrons (MLPs) and Convolutional Neural Networks (CNNs) in particular have had a great influence in this domain due to their capability to model complex patterns in data [1].

In this project, we implemented an MLP from scratch to classify image data and further compare its performance with that of CNNs. This was done using two real-world datasets: the Fashion MNIST dataset [2] and the CIFAR-10 dataset [3].

The Fashion MNIST dataset consists of 70,000 grayscale images spanning 10 fashion categories: T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, and Ankle boot [2]. Conversely, the CIFAR-10 dataset contains 60,000 color images spread over 10 varied classes, presenting a more complex scenario for classification given its colored visuals and diverse subject matter/labels which range from airplanes to ships to cats.

We began this project by first acquiring the data and subsequently vectorizing and normalizing the datasets to ensure that the data has the appropriate dimensions that will work with our MLP.

Following this, we implemented our MLP model from scratch and conducted several different experiments to evaluate its performance. MLPs have demonstrated respectable performance on image classification tasks when the images are preprocessed adequately, capable of achieving an 88.08% accuracy when tested on the Fashion MNIST dataset [4]. However, when it comes to understanding the arrangement and relation of parts in images, CNNs are better suited. In contrast to MLPs, CNNs utilize a series of filters to capture patterns, establishing them as a preferred choice for image-related tasks [5].

We explored different weight initialization methods and compared MLP models of varying complexity. We also tried regularization techniques and trained the model with unnormalized images. Shifting to CNNs, we saw improved performance on both Fashion MNIST and CIFAR-10 datasets. Lastly, we experimented with optimizer settings, comparing the effects of SGD and Adam optimizers on model performance.

This report details the methodologies and findings from our experiments with our MLP and CNN models. We observed the significant influence of weight initialization on training dynamics and confirmed CNNs' superior performance in image classification compared to MLPs. Furthermore, the MLP was extensively evaluated using various activation functions, including ReLU, Sigmoid, and Tanh, with Tanh producing the highest accuracy.

I. DATA SETS

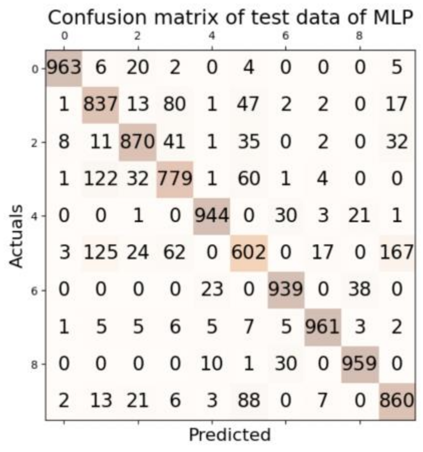
For this project, we were tasked with processing and analyzing two different datasets. The first dataset was the Fashion MNIST dataset. As aforementioned, this dataset consists of 70,000 grayscale images of 10 fashion categories [2]. A sample image from each one of these categories can be seen below in *Figure 1* in the appendix.

The second dataset was the CIFAR-10 dataset which consists of 60,000 32x32 color images in 10 different classes, with 6,000 images per class [3]. The classes include objects such as airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks. A sample image from each one of these classes can be seen below in *Figure 2* in the appendix.

Both datasets were vectorized and normalized for MLP compatibility. We verified data integrity by displaying sample images from each class and analyzed the label distribution to ensure balance, as imbalances can bias the MLP. Figure 3 in the appendix shows an even distribution across all labels in both datasets, which is ideal for unbiased MLP learning and prediction.

II. RESULTS

Following the implementation of our MLP from scratch, and prior to conducting any specific experiment on it, we evaluated the performance of the MLP through various metrics, such as the use of a confusion matrix. This was done as the confusion matrix is a crucial tool for evaluating the performance of a classification model. It provides a visual representation of the model's predictions versus the true labels, making it easier to identify where the model is performing well and where it's making mistakes. *Figure 4* below displays the confusion matrix associated with our MLP's test data.


Confusion matrix of test data of MLP
Actuals
Predicted

	0	1	2	3	4	5	6	7	8	9
0	963	6	20	2	0	4	0	0	0	5
1	1	837	13	80	1	47	2	2	0	17
2	8	11	870	41	1	35	0	2	0	32
3	1	122	32	779	1	60	1	4	0	0
4	0	0	1	0	944	0	30	3	21	1
5	3	125	24	62	0	602	0	17	0	167
6	0	0	0	0	23	0	939	0	38	0
7	1	5	5	6	5	7	5	961	3	2
8	0	0	0	0	10	1	30	0	959	0
9	2	13	21	6	3	88	0	7	0	860

Figure 4: Confusion Matrix of Test Data of MLP

Additionally, we also plotted the test and train performance of the MLP as a function of training epochs. These plots are essential for understanding how the model is learning from the training data and how it's performing on unseen test data over time (epochs). The plot of training and test loss over epochs and the plot of training and test accuracy over epochs can be seen below in *Figure 5* and *Figure 6*, respectively.

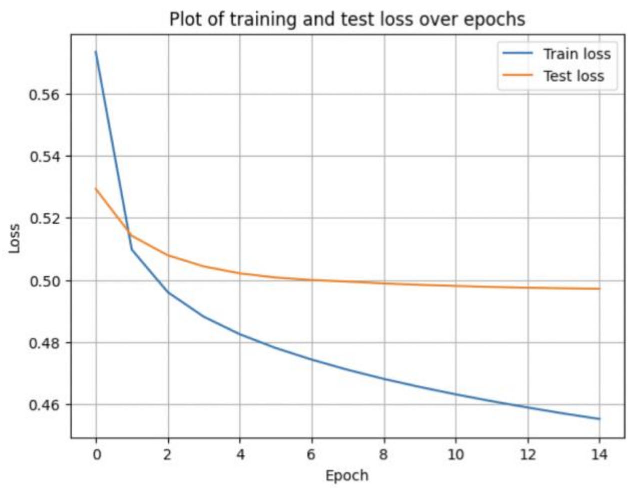


Figure 5: Plot of training and test loss over epochs

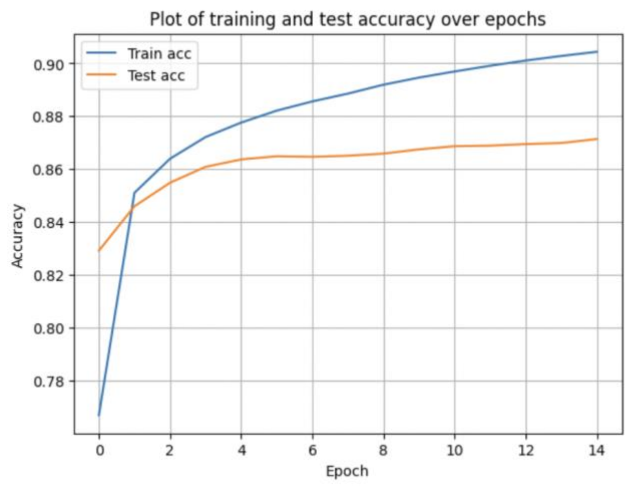


Figure 6: Plot of training and test accuracy over epochs

Several key observations can be made about the MLP based off figure 5 and figure 6. Firstly, Both the training and test metrics (loss and accuracy) are stabilizing or converging as epochs increase, suggesting that the model is reaching a point where further training might not lead to significant improvements. This also suggests that that the model might be approaching its best possible performance on this data with the current architecture and hyperparameters. Additionally, the plots also indicate that there is no sign of overfitting; typically, overfitting shows training metrics improving while test metrics deteriorate, but both metrics in our plots advance together. Achieving 91% training and 87% test accuracy suggests a robust MLP model, bolstering our confidence for subsequent experiments.

Furthermore, to fine-tune the hyperparameters of our MLP model, we performed an exhaustive grid search over a predefined set of potential hyperparameters. We repeated this process on several MLP architectures with different activation functions and varying numbers of hidden layers, using the specified parameters for each architecture. In doing so, we determined the optimal learning rate, learning rate decay and the best possible test accuracy for each given MLP architecture. *Table 0* in the appendix displays these results.

1. *MLPs with Different Weight Initialization Techniques*

In our initial experiment, we compared the impact of different weight initialization methods on the training and test accuracy of the Fashion MNIST dataset. We deployed five MLPs, each with 128 hidden units, initializing weights using the following techniques: All zeros; Uniform [-1, 1]; Gaussian N(0,1); Xavier; and Kaiming. The findings are summarized in Table 1.

Weight Initialization Technique	Test Accuracy on Fashion MNIST Dataset
All Zeros	10 %
Uniform [-1, 1]	73.73 %
Gaussian N(0,1)	75.07 %
Xavier	87.42 %
Kaiming	87.69 %

Table 1: Test Accuracy on Fashion MNIST Dataset for Different Weight Initialization Techniques

Evidently, from Table 1 an All-Zeros weight initialization technique results in the lowest test accuracy by far at 10% accuracy. The other techniques all present much higher accuracies. Uniform [-1, 1] yields an accuracy of 73.73%, while the Gaussian initialization with mean 0 and variance 1 achieves 75.07% accuracy. Notably, the Xavier and Kaiming initialization techniques demonstrate the highest accuracies, with 87.42% and 87.69% respectively, indicating their superior performance for this dataset.

2. *MLPs with Different Number of Hidden Layers*

In our subsequent experiment, we designed three distinct MLPs with varying hidden layers and compared their test accuracy on the Fashion MNIST dataset. The models were: 1) without hidden layers, 2) with one hidden layer of 128 units using ReLU activations, and 3) with two hidden layers, each with 128 units and ReLU activations. *Table 2* below showcases the test accuracy for each model on the dataset.

Number of Hidden Layers Used in MLP Model	Test Accuracy on Fashion MNIST Dataset
0	84.12 %
1	87.42 %
2	87.31 %

Table 2: Test Accuracy on Fashion MNIST Dataset for Different Number of Hidden Layers

Based off Table 2, we notice that the no hidden layer model scored 84.12%, while the MLP with one hidden layer achieves an accuracy of 87.42%, and the model with 2 hidden layers yields 87.31%. Typically, adding more hidden layers to a neural network allows it to capture more complex patterns and relationships in the data, which can lead to higher accuracy, however in this case, we notice that adding more layers doesn't always guarantee better results.

3. *MLPs with Different Activation Functions*

The next experiment involved taking the MLP model with 2 hidden layers from the previous experiment and creating two different copies of it with two different activation functions other than ReLU, as the initial MLP model uses ReLU. We decided to use the Sigmoid and Tanh activation functions. We trained these models on the Fashion MNIST dataset and subsequently compared their test accuracies with the model with ReLU activations. The test accuracies corresponding with each activation function is illustrated in *Table 3.*

Activation Function	Test Accuracy on Fashion MNIST Dataset
ReLU	87.31 %
Sigmoid	86.55 %
Tanh	87.11 %

Table 3: Test Accuracy on Fashion MNIST Dataset for Different Activation Functions

The ReLU-activated model achieved an accuracy of 87.31%, while replacing ReLU with the Sigmoid function resulted in a slightly worse performance of 86.55%. Additionally, the Tanh activation function registered a test accuracy of 87.11%. The results from Table 3 suggest that for this specific dataset and model architecture, ReLU may be the most optimal activation function among the three.

4. *MLPs with Different Regularization Techniques*

Next, using an MLP with 2 hidden layers each having 128 units with ReLU activations, we investigated the effects of L1 and L2 regularization on the accuracy of the models. We not only independently added L1 and L2 regularization to the network, but we also added both at the same time to see how this would affect the test accuracy. To adequately conduct this experiment, we once again performed an exhaustive grid search to determine the optimal L1 and L2 values for our MLP model when utilizing L1 regularization, L2 Regularization, or both. Table 4.1 in the Appendix showcases these optimal L1 and L2 values in the different cases of using L1 regularization, L2 Regularization, or both, while Table 4.2 below highlights the test accuracies on the Fashion MNIST dataset for different regularization techniques.

Regularization Technique Used	Test Accuracy on Fashion MNIST Dataset
No Regularization	87.31 %
L1-Regularization	87.42 %
L2-Regularization	87.21 %
L1 & L2	87.32 %

Table 4.2: Test Accuracy on Fashion MNIST Dataset for Different Regularization Techniques

Interestingly, the absence of regularization yields an accuracy of 87.31%, which is comparable to the results achieved using L1, L2, or a combination of both regularizations. Specifically, L1-Regularization yields a slightly increased accuracy of 87.42%, while L2-Regularization stands at 87.21%. A combined L1 & L2 regularization achieves an accuracy of 87.32% which is very similar to no regularization. The results suggest that overfitting might not be an issue in this model configuration as the introduction of regularization techniques did not significantly improve the model's performance. If overfitting was present, there would be a significant improvement in test accuracy when regularization is applied. However, since the accuracy remained relatively consistent regardless of the regularization technique used, it signifies that the base model without regularization was already generalizing well to unseen data.

5. *MLPs with Unnormalized Images*

In the next experiment, we trained an MLP with two hidden layers and 128 units each using ReLU activations on unnormalized images to evaluate its test accuracy against a model trained on normalized images. On the Fashion MNIST dataset, the unnormalized image model achieved an accuracy of *81.82%*, compared to *87.31%* for the normalized image model. This was expected as normalizing images scales pixel values to a standard range, typically [0,1], which can help the model converge faster during training. Unnormalized images may have varied and extreme pixel values, making it harder for the network to learn and possibly leading to less accurate predictions. The observed difference in test accuracy underlines the importance of preprocessing and normalizing data in neural network training.

6. *Comparing CNN with MLP on Fashion MNIST Dataset*

Subsequently, we decided to experiment with CNNs to determine how accurate they are relative to MLPs when tested against the same dataset; that is, the Fashion MNIST dataset. Thus, we created a CNN with 2 convolutional and 2 fully connected layers. We also set the number of units in the fully connected layers to be 128 and set the activation in all the layers to be ReLU. The CNN model was then trained on the Fashion MNIST dataset across 10 epochs, and the loss per each epoch was computed. Figure 7 below illustrates the relationship between the loss and the number of epochs during the training of the CNN model on the Fashion MNIST dataset. The corresponding values for each point in figure 6 are displayed in *Table 6* in the appendix.

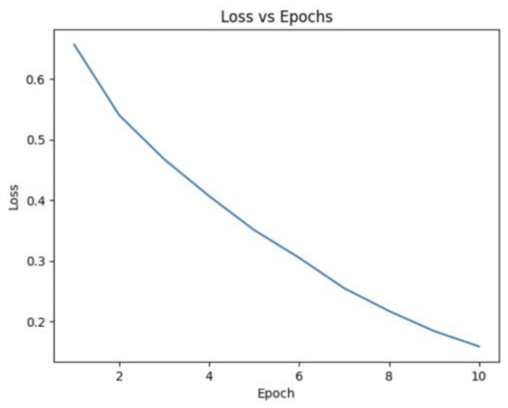


Figure 6: Loss vs Epochs for CNN Model on Fashion MNIST Training Dataset

Figure 6 shows a clear downward trend in the loss as the number of epochs increases. This suggests that the model is effectively learning from the training data and is becoming progressively better at making predictions. Moreover, this indicates a successful training process where the model has learned and improved its performance over time. After training the CNN model using the Fashion MNIST training dataset, we evaluated its performance on the test set, achieving an accuracy of 91.69%. When we compare the test accuracy of the CNN to that of the MLP model with two hidden layers of 128 units each and ReLU activations (87.31%), it's evident that the CNN achieves a higher accuracy. This is expected as CNNs are specifically designed for image data, making them more suitable for image tasks than MLPs. Their inherent structure allows for recognizing patterns regardless of the position in the image, often leading to better performance in image classification compared to MLPs.

7. *Comparing CNN with MLP on CIFAR-10 Dataset*

For the next experiment, we trained our MLP and CNN models using the CIFAR-10 dataset and subsequently compared their test accuracies to determine which of these two models yield the best test accuracy when tested against the CIFAR-10 dataset.

To optimize this experiment, we conducted an exhaustive grid search to fine-tune the parameters of the MLP model to compare its highest test accuracy with the test accuracy of the CNN model. And regarding the CNN model, we trained in a similar fashion as in experiment 6 and obtained its loss across 10 epochs. Table 7.1 and 7.2 in the appendix display the results of the exhaustive grid search for MLP and the loss values of CNN across each epoch when trained using the CIFAR-10 training dataset. Figure 7 in the appendix depicts the plot of these loss values against each epoch (similar to that of Figure 6).

From Table 7.1, we notice that best accuracy achieved by the MLP was 39.41%. However, when we tested the CNN on the test dataset, it achieved an accuracy of 70.89%. Evidently, the CNN clearly outperforms the MLP on the CIFAR-10 dataset. Once again, as aforementioned, this is expected as their inherent structure allows for recognizing patterns regardless of the position in the image, often leading to better performance in image classification compared to MLPs.

8. *The Use of SGD Optimizer on CIFAR-10 Dataset*

In the next experiment, we investigated the effects of an optimizer on performance with respect to the CIFAR- 10 dataset. We initially employed an SGD optimizer with a momentum factor set to zero, and then we increased this value to 0.9 to gauge its influence on the network's convergence speed, final test accuracy, and stability. We then used an Adam optimizer and compared its convergence speed, final accuracy and stability against the results achieved with SGD. To quantify each optimizer’s convergence speed, we computed the average absolute rate of change in loss (AARCL) between successive epochs; and to quantify each optimizer’s stability, we calculated the variance of the rate of change in loss (VRCL). Table 8.1, 8.2, and 8.3 in the appendix display the loss values of CNN across each epoch for SGD with momentum 0, SGD with momentum 0.9, and Adam, respectively. Furthermore, the calculations to determine the ARRCL between successive epochs and VRCL for each optimizer can be found under Calculation 8.1, 8.2, and 8.3 in the appendix; for SGD with momentum 0, SGD with momentum 0.9, and Adam, respectively.

Table 8.4 below showcases the final results.

Optimizer	Momentum value	AARCL	VRCL	Test Accuracy
SGD	0	0.1202	0.0122	61.92 %
SGD	0.9	0.1404	0.0163	71.48 %
Adam	-	0.0795	0.0012	71.56 %

Table 8.4: AARCL, VRCL, and Test Accuracy for Different Optimizers

SGD with momentum had the quickest convergence (AARCL of 0.1404), yet Adam outperformed in test accuracy (71.56%) and stability (VRCL of 0.0012). While SGD converges rapidly, Adam balances better accuracy and stability.

9. *The Use of ResNet on CIFAR-10 Dataset*

For this experiment, we loaded a pre-trained model (ResNet), froze all the convolutional layers, and removed all the fully connected ones. We then added 3 fully connected layers tailored for a 10-class output like CIFAR-10. After training for 10 epochs on the CIFAR-10 dataset, the model achieved an accuracy of 39.15%. While this is slightly below the 39.41% of the top performing MLP from experiment 7, it's notably lower than the 70.89% achieved by the regular CNN in the same experiment.

III. DISCUSSION AND CONCLUSION

In this project, we focused on the topic of neural network architectures, primarily comparing MLPs and CNNs in image classification tasks. Through various rigorous experiments, CNNs consistently emerged as the superior choice, predominantly due to their ability to identify spatial hierarchies in images. Training dynamics, such as decreasing loss with increasing epochs, showcased the effectiveness of our training strategies. The choice of optimizer, specifically between SGD with momentum and Adam, highlighted a trade-off between convergence speed and final accuracy. Using a pre-trained ResNet model showed us the challenging aspects of using existing models for new tasks. Changes to these models must match the specific needs of the data we're working with. Despite some successes, no single model or strategy was optimal across all the datasets, emphasizing the importance of iterative validation in machine learning. For future endeavors, exploring deeper CNN architectures, more exhaustive hyperparameter searches, and other pre-trained models might yield further insights.

IV. STATEMENT OF CONTRIBUTION

All team members contributed equally to the project.

V. REFERENCES

[1] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

[2] FashionMNIST - Torchvision 0.16 documentation.
https://pytorch.org/vision/stable/generated/torchvision.datasets.FashionMNIST.html

[3] CIFAR10 - Torchvision main documentation.
https://pytorch.org/vision/main/generated/torchvision.datasets.CIFAR10.html

[4] Image classification using multilayer Perceptron. (n.d.). https://jingpeilu.github.io/doc/253HW2_Report.pdf

[5] Krizhevsky, A. , Learning Multiple Layers of Features from Tiny Images.
https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

VI. APPENDIX

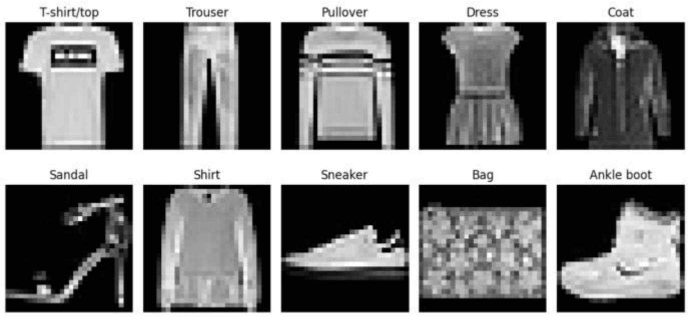


Figure 1: Sample Images from Each Category in the Fashion MNIST dataset

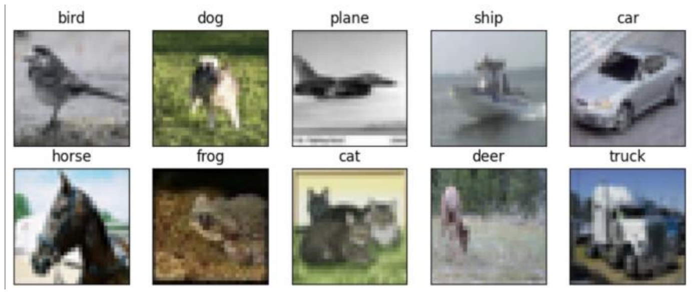


Figure 2: Sample Images from Each Class in the CIFAR-10 dataset

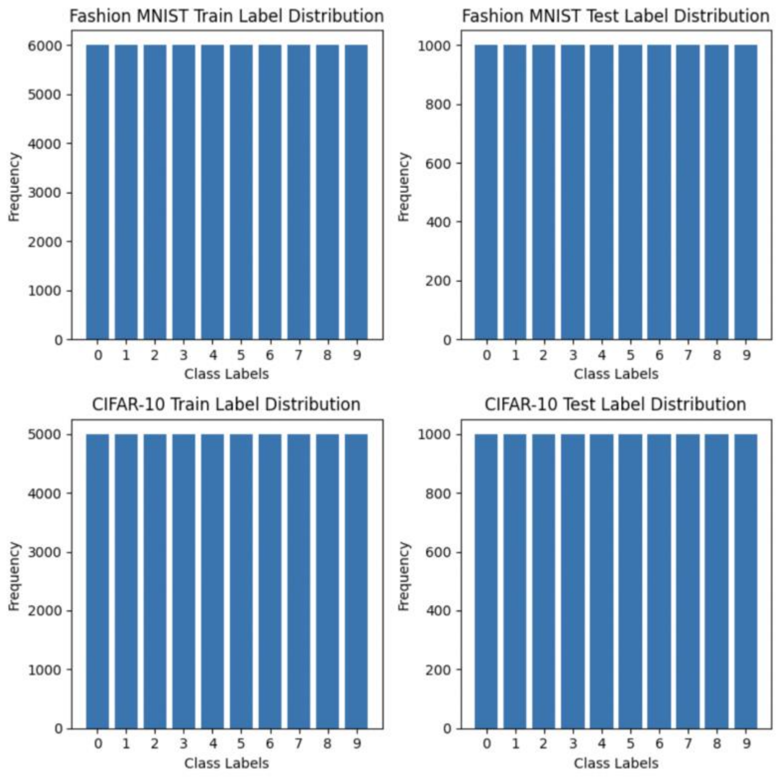


Figure 3: Label Distribution of Training and Testing data in both Fashion MNIST & CIFAR-10

Activation Function	No. of Hidden Layers	Learning Rate	Learning Rate Decay	Best Accuracy
ReLu	0	0.01	0.01	84.18 %
ReLu	1	0.01	0.00	87.47 %
ReLu	2	0.005	0.01	87.39 %
Sigmoid	2	1.00	0.0001	86.62 %
Tanh	2	0.1	0.01	87.23 %

Table 0: Learning Rate, Learning Rate Decay & Best Accuracy for different MLP Architectures Using Fashion MNIST Dataset

Regularization Technique	Optimal L1 Value	Optimal L2 Value
L1-Regularization	0.00005	0
L2-Regularization	0	0.00001
L1 & L2	0.00005	0.000001

Table 4.1: Optimal L1 and L2 Values for Different Regularization Techniques

Epoch	Loss
1	0.65636580
2	0.54017617
3	0.46755453
4	0.40684774
5	0.35087515
6	0.30523514
7	0.25512229
8	0.21741519
9	0.18442148
10	0.15912487

Table 6: Loss Values of CNN Against Fashion MNIST Dataset Across Each Epoch

Activation Function	No. of Hidden Layers	Learning Rate	Learning Rate Decay	Best Accuracy
Sigmoid	2	0.5	0	35.99 %
ReLU	2	0.001	0	38.85 %
Tanh	2	0.01	0	38.15 %
Sigmoid	1	0.1	0	39.22 %
ReLU	1	0.001	0	39.41 %
Tanh	1	0.01	0	37.91 %

Table 7.1: Learning Rate, Learning Rate Decay & Best Accuracy for different MLP Architectures Using CIFAR-10 Dataset

Epoch	Loss
1	1.314374515
2	0.943174235
3	0.786808855
4	0.681082265
5	0.587616566
6	0.5043130443
7	0.4252312746
8	0.3579109602
9	0.2853771568
10	0.23147067041

Table 7.2: Loss Values of CNN Against CIFAR-10 Dataset Across Each Epoch

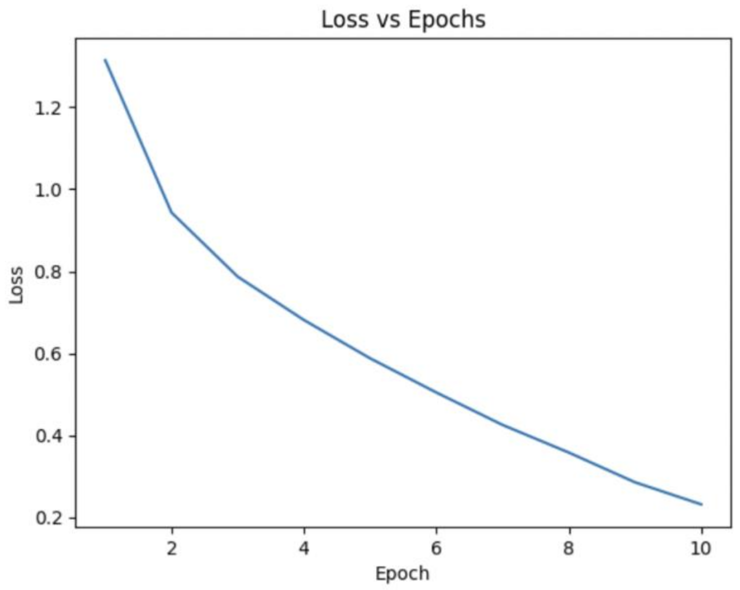


Figure 7: Loss vs Epochs for CNN Model on CIFAR-10 Training Dataset

Epoch	Loss
1	1.897176317
2	1.506223612
3	1.339422759
4	1.228563468
5	1.141635927
6	1.072150790
7	1.0123496821
8	0.9559292238
9	0.9066190813
10	0.8643230581

Table 8.1: Loss Values of CNN Against CIFAR-10 Dataset Across Each Epoch (SGD With Momentum 0)

Calculation 8.1

$$AARCL = \frac{\Delta\, loss}{\Delta\, epoch} = \frac{(|1.8972 - 1.5062| \ + \ |1.5062 - 1.3394| \ + \ |1.3394 - 1.2286| + \ldots + |0.9066 - 0.8643|)}{9}$$

$$AARCL = \frac{\Delta\, loss}{\Delta\, epoch} = \frac{(0.3910 \ + \ 0.1668 \ + \ 0.1109 + \ldots + 0.0423)}{9} = \mathbf{0.1202}$$

$$VRCL = \frac{\sum(rate\ of\ change - average\ rate\ of\ change)^2}{n}$$

$$VRCL = \frac{(0.3910 - 0.1202)^2 \ + \ (0.1668 - 0.1202)^2 + \ldots + (0.0423 - 0.1202)^2}{9} \approx \mathbf{0.0122}$$

Epoch	Loss
1	1.416222412
2	0.987153996
3	0.815650893
4	0.673838977
5	0.5563100888
6	0.4501428571
7	0.3446544530
8	0.2651904651
9	0.2001371108
10	0.1543524815

Table 8.2: Loss Values of CNN Against CIFAR-10 Dataset Across Each Epoch (SGD With Momentum 0.9)

Calculation 8.2

$$AARCL = \frac{\Delta\, loss}{\Delta\, epoch} = \frac{(|1.4162 - 0.9872|\ +\ |0.9872 - 0.8157|\ +\ |0.8157 - 0.6738| + \ldots + |0.2001 - 0.1544|)}{9}$$

$$AARCL = \frac{\Delta\, loss}{\Delta\, epoch} = \frac{(0.4291\ +\ 0.1715\ +\ 0.1418 + \ldots + 0.0458)}{9} = \mathbf{0.1404}$$

$$VRCL = \frac{\sum(rate\ of\ change - average\ rate\ of\ change)^2}{n}$$

$$VRCL = \frac{(0.3910 - 0.1202)^2\ +\ (0.1668 - 0.1202)^2 + \ldots + (0.0423 - 0.1202)^2}{9} \approx \mathbf{0.0163}$$

Epoch	Loss
1	0.7654945285
2	0.6452936031
3	0.5409853251
4	0.4509056692
5	0.3704220174
6	0.2857749723
7	0.2250828607
8	0.1701379255
9	0.1448338590
10	0.1210582250

Table 8.3: Loss Values of CNN Against CIFAR-10 Dataset Across Each Epoch (Adam Optimizer)

Calculation 8.3

$$AARCL = \frac{\Delta\, loss}{\Delta\, epoch} = \frac{(|0.7655 - 0.6453| + |0.6453 - 0.5410| + |0.5410 - 0.4509| + \ldots + |0.1448 - 0.1211|)}{9}$$

$$AARCL = \frac{\Delta\, loss}{\Delta\, epoch} = \frac{(0.1202 + 0.1043 + 0.0901 + \ldots + 0.0238)}{9} = \mathbf{0.0795}$$

$$VRCL = \frac{\sum(rate\ of\ change - average\ rate\ of\ change)^2}{n}$$

$$VRCL = \frac{(0.1202 - 0.0795)^2 + (0.1043 - 0.0795)^2 + \ldots + (0.0238 - 0.0795)^2}{9} \approx \mathbf{0.0012}$$