# COMP 551 - MiniProject 2: Classification of Image Data with Multilayer Perceptrons and Convolutional Neural Networks, Fall 2023

Johan Trippitelli - 260917958, Minh Anh Trinh - 260853143, Chris Chong - 260976714

October 22, 2023

**Abstract:** The following mini project was conducted as part of the COMP 551 class at McGill university. It focuses on classifying image data using multilayer perceptrons and convolutional neural networks.

In the first task, two datasets are employed: the Fashion MNIST dataset and the CIFAR-10 dataset. Data is loaded into Python using the pytorch machine learning library, preprocessed, and utilized for the subsequent tasks.

The second task is implementing the multilayer perceptron machine learning model. The model was built from scratch using the knowledge obtained from COMP 551 lectures.

The third task is to perform experiments on the given data sets and examine the effects of various hyperparameters on the performance of the model. For this, the default training and testing sets were employed. In addition, in experiments 6-8, a pytorch-implemented convolutional neural network is compared against the MLP in terms of accuracy.

**Introduction:** The primary objective is to train and test a multilayer perceptron and analyze the Fashion-MNIST dataset and a CIFAR dataset, while varying the hyperparameters and observing the effects on performance. These hyperparameters include the weight initializations, the activation functions, regularization, and normalization. Later on, the MLP's performance is compared against a convolutional neural network in testing both datasets. Among other findings, it is shown that CNNs can outperform MLPs in terms of accuracy score with less effort in tuning hyperparameters. As well, increasing the momentum for stochastic gradient descent in CNNs improves the accuracy and stability of the final model.

**Datasets:** The two datasets used are the Fashion MNIST dataset and the CIFAR-10 dataset.

The Fashion-MNIST dataset has a total of 70,000 samples, each a 28x28 pixels large grayscale image that maps to a label which is one of 10 classes. These 10 classes include items such as t-shirts, trousers, pullovers, dresses, coats, sandals, shirts, sneakers, bags, and ankle boots. Each pixel in each image is represented by an integer between 0 and 255, with 0 being the lightest, and 255 being the darkest.[2]

The CIFAR-10 dataset has a total of 60,000 samples, each a 32x32 pixels large colour image that maps to a label which is one of 10 classes. These 10 classes include items such as airplanes, automobiles, birds, cats, deers, dogs, frogs, horses, ships, and trucks. Each image is represented by 3 32x32 large matrices, with the 3 matrices representing the amounts of color from red, green, and blue in each pixel. By default, the dataset is split into 50000 training images and 10000 test images.[1]

Some ethical concerns could arise in the manner in which these images were obtained, where they came from, what the motivations behind creating each dataset were. For example, if a certain brand was featured more prominently in one of the datasets for some reason or another.

**Results:**

In experiment 1 testing the different initializations was very conclusive. Using a learning rate of 0.1, a sigmoid activation function, and 1 hidden layer the results are shown in the bar graph in figure 1a. The data demonstrates that Xavier and Kaiming initializations outperformed the rest of the initializations. These methods got a test accuracy of roughly 0.85. On the other side of the spectrum we see that initializing our weights to 0 leads to poor training and a test accuracy of roughly 0.15.

Experiment 2: testing the different depths on the accuracy of predicting data from the Fashion-MNIST Dataset was a little tricky. When true relu was used the loss immediately became NAN and the results where inconclusive. This was deemed a drawback of relu creating "dead" neurons that fired 0 no matter the learning rate applied to the weights. To fix this an alternative function (leaky relu), kaiming initialization, and a small learning rate =

0.000001 were used for the training of the model. Without these modifications the training of our model resulted in NAN values of loss possibly from recurring neuron deactivation. With these new function definitions the model provided the results shown in figure 2d

Training Loss and Accuracy for the 0 layer model shown in 2a

Training Loss and Accuracy for the 1 layer model shown in 2b

Training Loss and Accuracy for the 2 layer model shown in 2c

By comparing the accuracy of model predictions on training and test data the experiment demonstrates that 1 hidden layer is best for this system.

In experiment 3, we kept parameters consistent with experiment 2 but tested Leaky ReLU, Sigmoid, and Tanh activation functions. Tanh achieved the highest test accuracy at 0.48, while Sigmoid had the lowest at 0.10 (figure 3d). As mentioned in experiment 2, to prevent exploding gradients for Leaky ReLU, we used a small learning rate of 0.000001. However, this learning rate was too small for Sigmoid, resulting in stagnant training accuracy (figure 3b). Tanh's superior performance may be attributed to its zero-centered range, which combats the vanishing gradient problem that we may have with sigmoid, thus help with faster convergence during training.

In experiment 4, with a lambda value of 0.001, L2 regularization outperformed L1 regularization when applied independently to a model with two hidden layers, each having 128 units with ReLU activations. As shown in figure 3d, the testing accuracy for L2 regularization was 0.54, while L1 regularization achieved a testing accuracy of 0.45. Without any regularization, the model yielded a testing accuracy of 0.50. L2 regularization's superior performance may be attributed to its effective control of overfitting while retaining more features in the parameters, as opposed to L1 regularization, which is more aggressive in feature reduction.

Using the same model employed in experiment 4, in experiment 5 the model yielded significantly improved performance when working with unnormalized images. In fact, as we can see in figure 5c, its testing accuracy nearly doubled in comparison to using the same model with normalized images ranging from -1 to 1 as input (0.57 versus 0.30, respectively).

In experiment 6, the CNN increases accuracy when compared to using an MLP. The CNN with just 2 convolutional and 2 fully connected layers was able to achieve a test accuracy of 97.78% when compared to the MLP's accuracy of 70.44% on the same Fashion-MNIST Dataset. Since CNNs are able to use convolution layers, introducing inductive bias to MLP, it can achieve a higher performance, with much less effort or parameter tuning. This is shown in figures 6a and 6b

In experiment 7, with the CIFAR-10 dataset which has much more complex images, in the form of colored rather than black and white images, the difference is even more drastic. Since CNNs use kernels to associate nearby pixels with each other, it is able to get a much better result than the regular MLP. This is shown in figures 7a and 7b

In experiment 8, with the CIFAR-10 dataset, we can see that having as the momentum increases, this change in the hyperparameter will allow the training loss to start lower, and improve over time, with a similar slope, whereas with the adam optimizer, it will converge to a lower training loss faster. This is shown in figures 8a, 8b, and 8c

In terms of the accuracy changes, the adam optimizer has an accuracy score of 78.6% and the SGD optimizer has a best accuracy score of 59.23% while using a momentum of 0.8 in our testing.

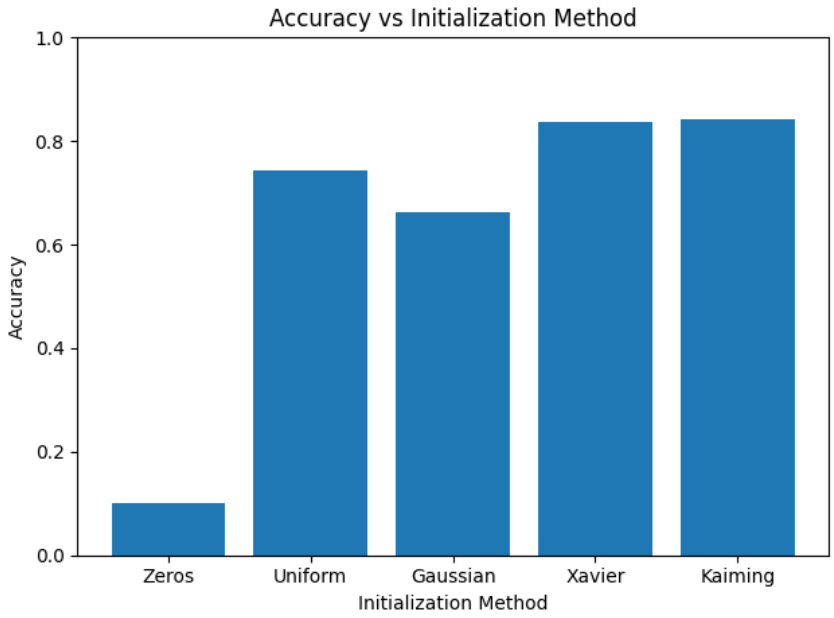
**Discussion and Conclusion:** Conclusions that can be drawn from experiment 1 are as follows: initialization with zeros should not be used as it reduces the accuracy of our model significantly; initialization using Kaiming and Xavier methods is preferable to any other method when it comes to sigmoid activation functions. When using RELU activation it is important to use Kaiming initialization to reduce the possibility of NAN errors. Results for expirement 2 show that it is best to utilize 1 hidden layer for this particular model. However, the performance of 2 layers and 0 layers was close behind. Perhaps if different activation functions were used this conclusion would not hold. Therefore layer determination requires more detailed testing. In Experiment 3, Tanh outperforms Leaky ReLU and Sigmoid, with Sigmoid performing the worst. This might be due to the small learning rate used for Leaky ReLU to prevent exploding gradients, but was too small for Sigmoid, resulting in stagnant training accuracy. In experiment 4, L2 regularization yielded superior results in comparison to L1 regularization and the absence of regularization. In Experiment 5, unnormalized images significantly improve performance as opposed to normalized images. Results for experiments 6 and 7 clearly show that CNNs are superior in performance and efficiency when compared to MLPs, in that hyperparameters don't have to be tuned much to match or beat performance given by MLPs. Results for experiment 8 show that using the adam optimizer will give a higher accuracy score when

compared to an SGD optimizer.

**Statement of Contributions:** Johan and Minh Anh implemented the MLP model. Minh Anh did experiments 3,4, and 5. Chris did descriptions and ethical analyses of the datasets, and experiments 6,7, and 8.

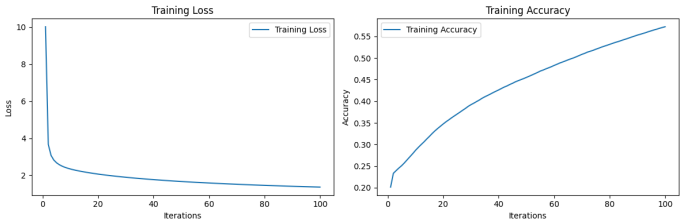
## References and Appendix:

**Figure 1:**

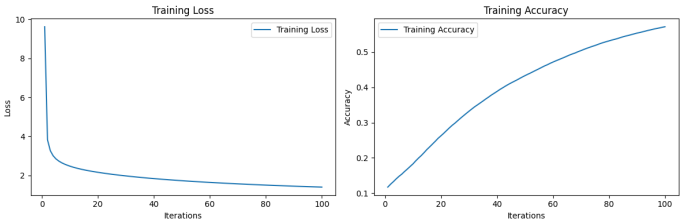



(a) Expirement 1: Performance depending on different weight initializations

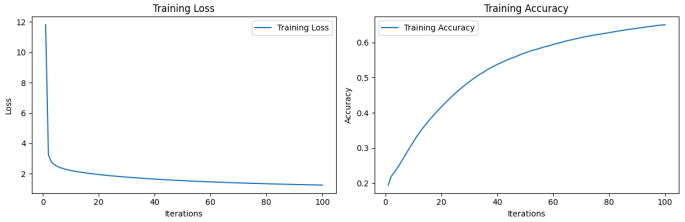
**Figure 2:**



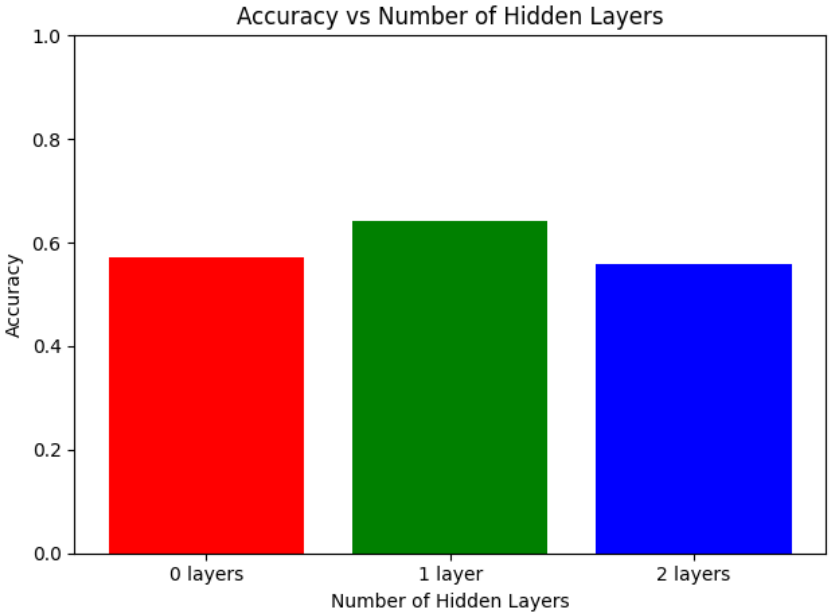

(a) Experiment 2: Performance on training for 0 depth




(b) Experiment 2: Performance on training for 1 depth

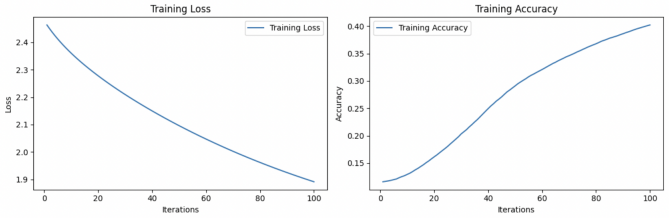



(c) Experiment 2: Performance on training for 2 depth

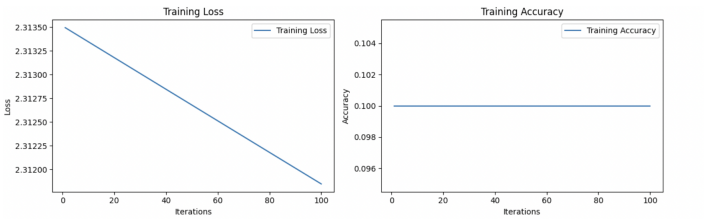



(d) Experiment 2: Final results on test data

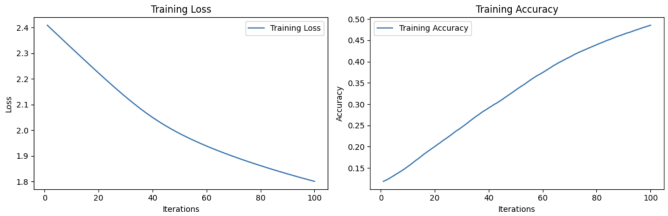
**Figure 3:**



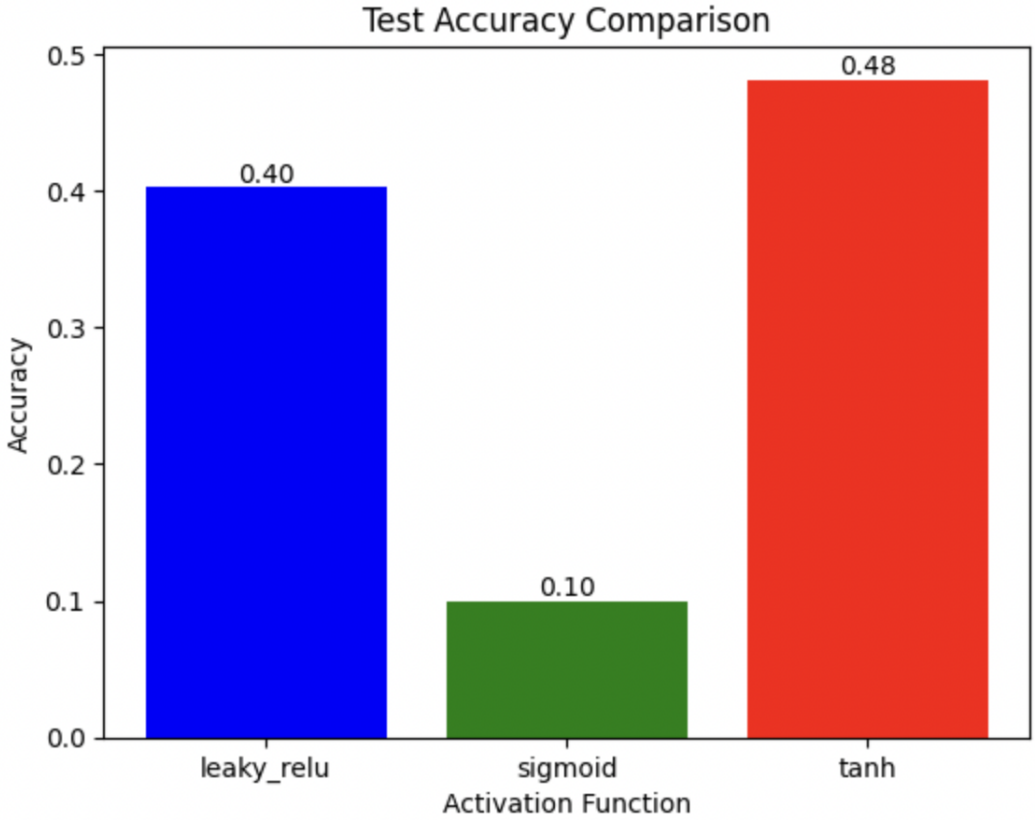

(a) Experiment 3: Training performancefor Leaky Relu




(b) Experiment 3: Training performance for Sigmoid

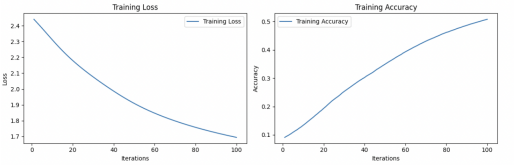



(c) Experiment 3: Training performance for Tanh

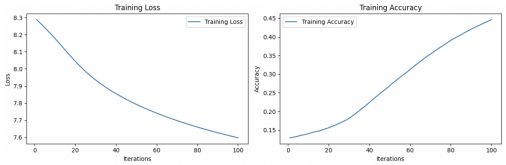



(d) Experiment 3: Final results on test data

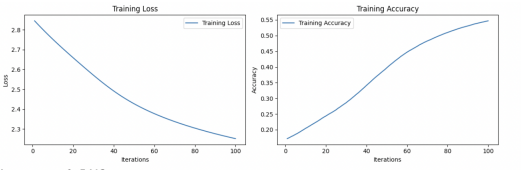
**Figure 4:**



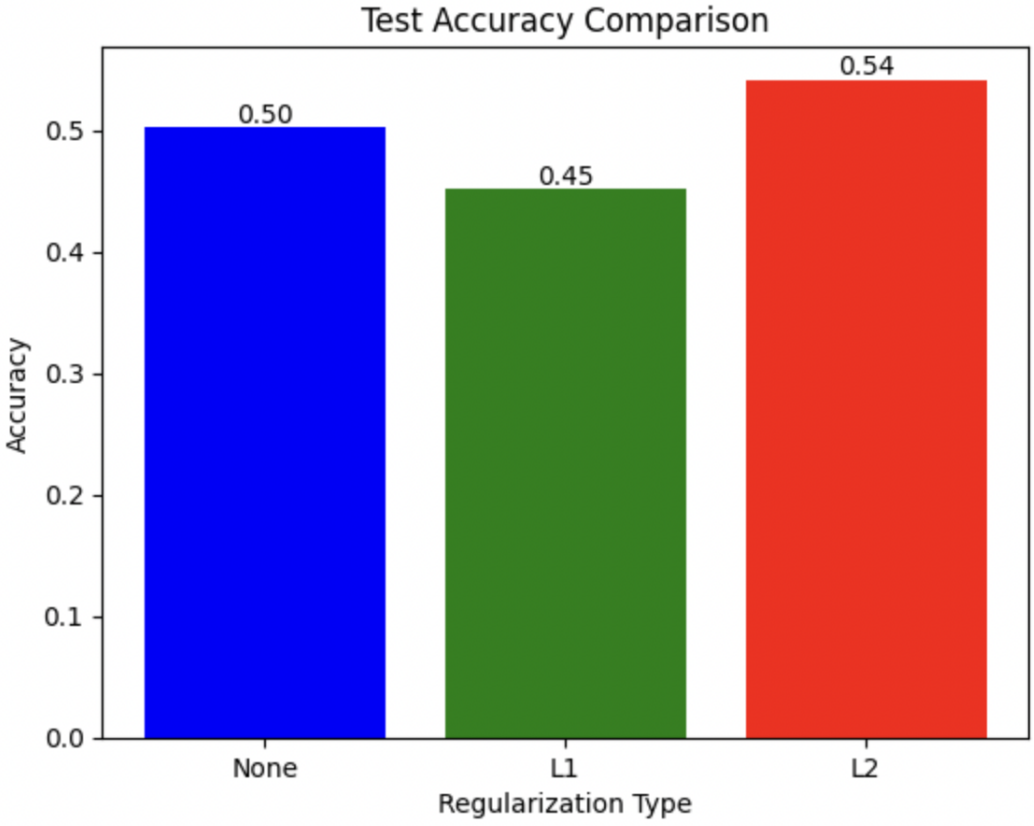

(a) Experiment 4: Training performance for no regularization




(b) Experiment 4: Training performance for L1 regularization

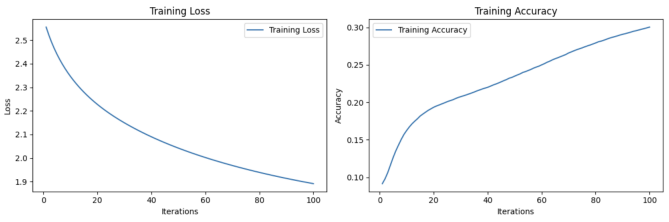



(c) Experiment 4: Training performancefor L2 regularization

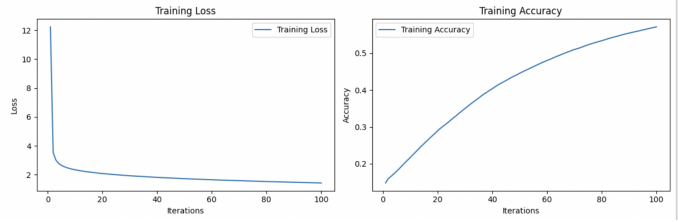



(d) Experiment 4: Final results on test data

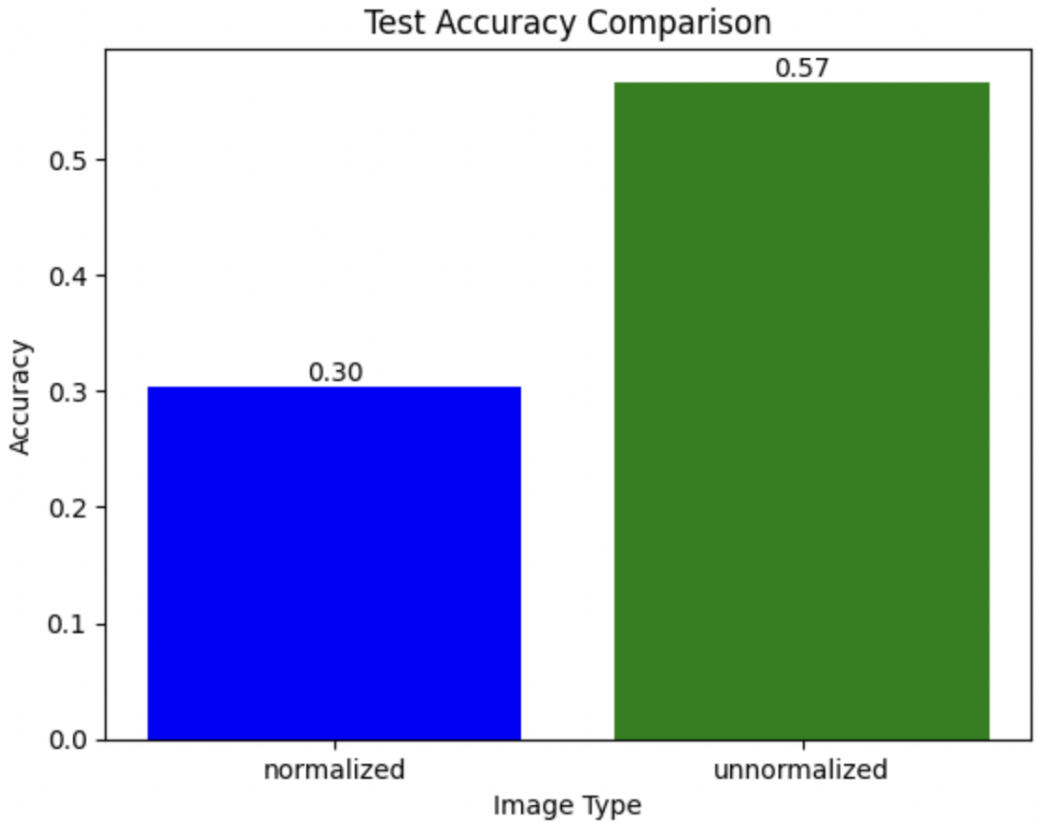
**Figure 5:**




(a) Experiment 5: Training performance for normalized data




(b) Experiment 5: Training performance for unnormalized data




(c) Experiment 5: Final results on test data

**Figure 6:**

(a) Experiment 6: Performance of CNN on fashion dataset

```
fashion_testloader = fashion_data_test_load
fashion_accuracy_list = []

correct = 0
total = 0
with torch.no_grad():
  for images, labels in fashion_testloader:
      images, labels = images.to(device), labels.to(device)

      #images = images.view(-1, 28*28)

      outputs = net(images)
      _, predicted = torch.max(outputs.data, dim=1)

      correct += (predicted == labels).sum().item()
      total += labels.size(0)
fashion_accuracy_list.append((100 * correct / total))

print(f"CNN on Fashion - Test Accuracy: {(100 * correct / total)} %%")
```

```
CNN on Fashion - Test Accuracy: 97.78 %%
```

```
y_pred_mnist = mlp.predict(X_test_mnist)
mlp.evaluate_acc(y_pred_mnist, y_test_mnist)
```

```
Accuracy: 0.7044
```

(b) Experiment 6: Performance of MLP on fashion dataset

**Figure 7:**

```
y_pred_cifar = mlp.predict(X_test_cifar)
mlp.evaluate_acc(y_pred_cifar, y_test_cifar)
```

```
Accuracy: 0.3549
```

(a) Experiment 7: Performance of MLP on cifar dataset

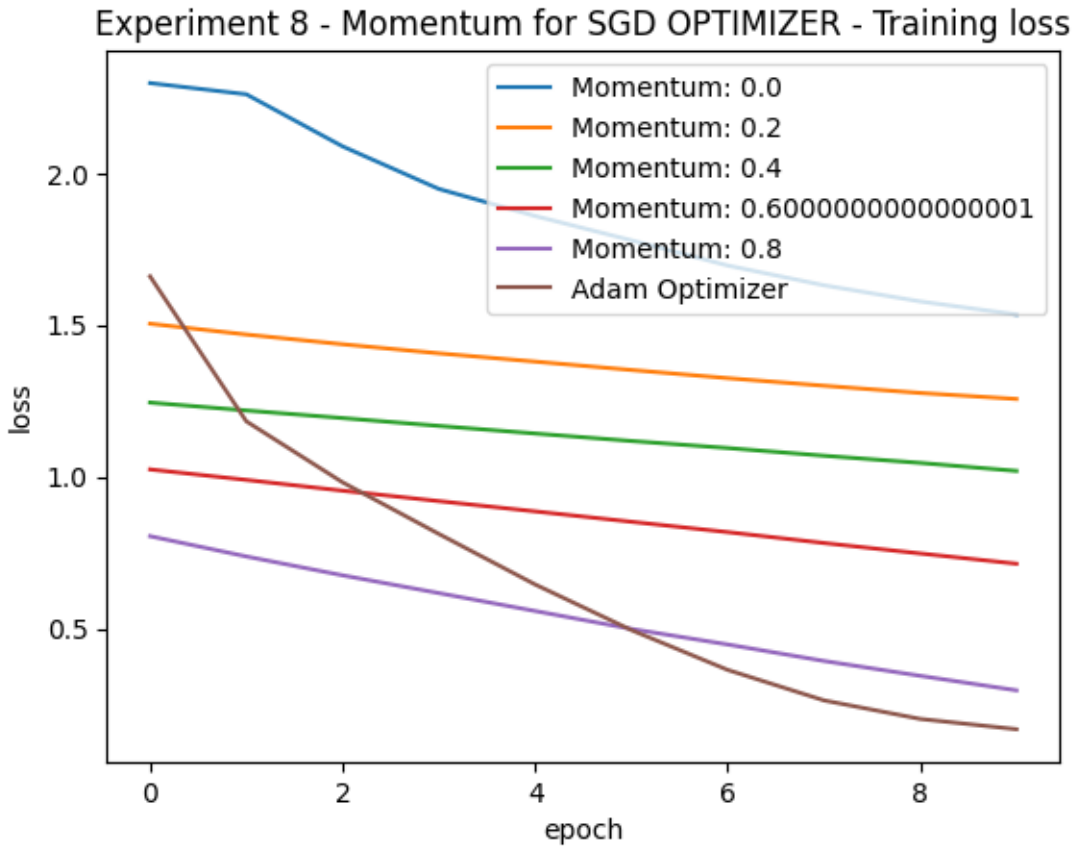
```
Test Accuracy: 59.23 %%
```

(b) Experiment 7: Performance of CNN on cifar dataset

**Figure 8:**

```
Model using Momentum 0.0 has an accuracy score of: 43.74
Model using Momentum 0.2 has an accuracy score of: 52.44
Model using Momentum 0.4 has an accuracy score of: 56.88
Model using Momentum 0.6000000000000001 has an accuracy score of: 59.19
Model using Momentum 0.8 has an accuracy score of: 59.23
```

(a) Experiment 8: Varying Momentum Accuracy Results for CNN




(b) Experiment 8: Training Loss for Varying Momentums on SGD vs Adam optimizer for CNN

```
adam_accuracy_list = []

with torch.no_grad():
  for images, labels in testloader:
      images, labels = images.to(device), labels.to(device)

      #images = images.view(-1, 28*28)

      outputs = net(images)
      _, predicted = torch.max(outputs.data, dim=1)

      correct += (predicted == labels).sum().item()
      total += labels.size(0)
adam_accuracy_list.append((100 * correct / total))

print(f"Test Accuracy: {(100 * correct / total)} %%")
```

```
Test Accuracy: 78.59333333333333 %%
```

(c) Experiment 8: Accuracy Results for Adam Optimizer on CNN

## References

[1] University of Toronto. The cifar-10 dataset, 2009.

[2] Zalando Research. Kaggle - fashion mnist, 2017.