COMP 551 Assignment 2: Classification of Image Data with Multilayer Perceptrons and Convolutional Neural Networks

Okyanus Gumus[1], Mihail Calitoiu[2], Eren Gurhan[3]

[1]McGill ID: 260900481.
[2]McGill ID: 260972537.
[3]McGill ID: 260973884.

Contributing authors: okyanus.gumus@mail.mcgill.ca; mihail.calitoiu@mail.mcgill.ca; eren.gurhan@mail.mcgill.ca;

Abstract

In this assignment a Convolutional Neural Network (CNN) and a Multi Layer Percoptron (MLP) was implemented for image recognition. Two data sets, Fashion MNIST and CIFAR-10, were used for the assignment, each with 10 different class labels. The datasets were vectorized to achieve the appropriate dimensions for the CNN. An investigation into the effects of different hyperparameters' (such as different weight types and learning rates) on test accuracy and test loss was conducted. The implemented CNN and MLP demonstrate expected behavior, with the CNN demonstrating significantly better performance for image recognition.

Keywords: Fashion MNIST, CIFAR-10, Convolutional Neural Networks, CNNs

1 Introduction

MNIST, and as a result Fashion MNIST, and CIFAR-10 are some of the most commonly used neural network datasets, each being used for different neural networks approaches. Fashion MNIST was used in some previous work, such as in [1–3] which focused on Convolutional Neural Networks (CNNs). CIFAR-10, on the other hand, was the subject of work such as [4–7], where implementations of different algorithms in CNNs was the main investigation topic. In this assignment, a multi-layer perceptron was implemented from scratch for image classification purposes. The perceptron was then tested using Fashion MNIST and CIFAR-10 testing sets.

This report is structured as follows: In Section 2, more information about the datasets is provided. The obtained results are found in Section 3. Discussion and conclusions are provided in Section 4,

followed by the statement of contribution in Section 5. Relevant screenshots and plots are provided in Appendix A.

2 Datasets

Fashion MNIST is an MNIST-like data set containing 70,000 images (60,000 training images and 10,000 testing images) of 28×28 grayscale clothing images, each associated with a label out of 10 classes. CIFAR-10, on the other hand, is a data set containing 60,000 32×32 colored images (50,000 training images and 10,000 testing images) with 10 different class labels.

For this assignment both the Fashion MNIST and CIFAR-10 data sets were vectorized to 64×64 to achieve the appropriate dimensions for the CNN. For the MNIST data set, consisting of 32 by 32 pixel images, the MLP's input layer width was set to 784 (32*32 pixels). In contrast, the CIFAR data set contains color images measuring 32x32 in size but also incorporating three distinct color channels: red, green, blue. Consequently, for CIFAR, the MLP had an input width of 3072 (3*32*32) representing the 3 channels flattened into a contiguous array, namely in order of red, blue, green. This distinction will be important for later sections. The test and training datasets were also normalized before implementing the neural networks.

3 Results

3.1 Task 1 & 2

For the assignment, we have downloaded the data set using built in PyTorch and Open ML libraries available in Python. As a 2nd step, the data sets have been vectored and separated into training and testing data sets.

For task 2, the MLP has been written from scratch and includes a fit and predict function. Our implementation of the MLP includes back propagation and mini-batch gradient descent. In order to evaluate the accuracy of our MLP model, we have also written an accuracy class.

For our MLP model, we have run multiple tests to find the best learning rate by trying 4 different values: 0.1, 0.05, 0.01, 0.005 over 400 epochs. As can be seen from Figure A1, the optimal learning is rate is determined to be 0.01. The validation accuracy steadily increases for a learning rate of 0.01, whereas the higher rate of 0.05 starts to overfit near epoch 100. A learning rate of 0.005 presents a good upward trend, but the average slope is lower, requiring more training time. As such, due to computational limitations, this learning rate parameter will be used throughout the assignment for the MLP model.

3.2 Task 3

3.2.1 Part 1:

This part of the assignment asks us to gain an understanding the effects of weights on our MLP model's accuracy.

As parameters, we have set our weights to be all zeros, uniform [-1 , 1], Gaussian N(0,1), Xavier, and Kaiming. To measure the accuracy, we have run our MLP model with each weight parameter and determined the accuracy under 400 epochs. The accuracy results have been plotted and are provided in Figure A2.

From the plot, we can observe that setting the weights to Kaiming results with higher accuracy and faster convergence on all epochs followed by Xavier, uniform, Gaussian, and all zeros. The plots are in line with our expectations.

Setting the weights to all zeros results with the worst accuracy because the neurons in our MLP learn the same features due to symmetry. Uniform weights don't perform well either because it is known that gradients can sometimes be 0 during back propagation which makes it much more difficult to train. Gaussian bases should provide better results due to adding randomness in the weights; however, we observe that it performs slightly poorly than uniform weights. We suspect this happened because our data set was small and our model was not deep enough to put Gaussian bases forward. Xavier weights are initialized such that the variance of the weights are set to 1 over total inputs. Kaiming weights are randomly selected from a list of mean 0 and variance of 2 over the total number of inputs. Kaiming performs the best out of all weights because the randomness introduced works well with RELU activation which we used in our MLP model.

3.2.2 Part 2:

In this section we are asked to determine the effects of hidden layers for the accuracy of our model.

To determine the effect of hidden layers, we have developed 3 different MLPs that have no hidden layers, 1 hidden layer with 128 neurons, 2 hidden layers with 128 neurons each and 3 hidden layers with 128 neurons each. Since we are determining classes as our output result, we have also added a softmax stage at the end and recorded that accuracy of each model.

As can be seen from Figure A3, an increase in hidden layers results in a corresponding rise in the required number of epochs to achieve a comparable level of accuracy. At epoch 50, the MLPs with 0/1 hidden layers have an accuracy of 60%, whereas the MLPs with 2/3 hidden layers have almost half the accuracy. This is explained due to the larger number of weights, and these weights typically do not produce accurate predictions initially. After more training epochs, the four varying depth MLPs start performing similarly. Although we did not train the models for longer, we predict that the higher number hidden layer MLPs will outperform the smaller models, due to more weights and better feature differentiation/abstraction.

3.2.3 Part 3:

For this part, we have implemented a model with RELU, Leaky RELU, and Hyperbolic tangent. The results are provided in Figure A4. For each method, we have plotted the accuracy over 200 epochs. We determined that RELU and Leaky RELU perform almost exactly the same in terms of accuracy while the hyperbolic tangent performs worse as it has an accuracy of 50%.

RELU is known to set all negative values to 0 and pass positive values to the next layer. Leaky RELU, on the other hand is similar to RELU, but has a small gradient for negative values. The hyperbolic tangent inserts the values between -1 and 1. Since the the Tanh is sensitive to scale of the data. We think this leads to the vanishing gradient problems which causes the accuracy to be lower than RELU and Leaky RELU.

3.2.4 Part 4:

For this part we were asked to determine the effect of regularization on accuracy. We have run our model with no regularization, L1, and L2 regularization. The results are provided in Figure A5.

We have observed that no regularization and L1 regularization perform almost the same and can be considered more accurate than L2 regularization. We believe that L1 results with lower accuracy due to 3 possible reasons: sparsity, hyper-parameter tuning, and model shape. For sparsity, L1 regularization potentially might struggle to capture relevant patterns in the image dataset. For hyper-parameter tuning, if we use different learning rate / batch size, we can potentially mitigate the low accuracy convergence. For model shape, we believe that the current structure is not favourable for L1 regularization.

3.2.5 Part 5:

In this section, we have investigated the effect of normalization on our model accuracy. To test this relationship, we have ran our model with normalized and non-normalized data.

As can be seen in Figure A6, normalization leads to 30% higher accuracy at epoch 20; normalization ensures that all features/inputs have a balanced influence, promoting stable and consistent weight updates. This can be seen from Figure A6, where the accuracy converges smoother (and more vertically) than the non-normalized dataset. Moreover, we have also determined that it takes faster for the normalized model to converge. This result aligns with our previous understanding of the effects of normalization.

3.2.6 Part 6 & 7:

Firstly, we created a CNN trained on the Fashion MNIST dataset. Additionally, for Part 7, we implemented another CNN and trained it using the CIFAR 10 dataset. The accuracy graphs of the two CNNs are shown in Figure A7 and Figure A8, respectively.

As can be seen in Figure A8, the CNN's accuracy is better than the MLP (provided in Figure A9). While the CNN achieves 97% accuracy in 20 epochs, the MLP model achieves 35% accuracy in 400 epochs. This shows that implementing a CNN for the CIFAR dataset results with higher accuracy and lower time required to achieve a good accuracy rate.

We believe CNNs are more accurate than the MLP due to the fact that convolution is used in the CNN. Since the MLP does not consider the spatial arrangement of features it performs poorly. Another improvement of the CNN architecture is weight sharing which results in fewer parameters and decreasing the risk of over-fitting. Additionally, this also decreases the computation requirement when training the model.

In terms of speed, we believe the CNN to outperform the MLP due to the pooling layer applied to the CNN. We know that pooling down samples the dimensions of input in each layer and reduces the sensitivity to small changes thus allowing the model to generalize better.

3.2.7 Part 8:

In this part, we are asked to determine the effects of changing the momentum on our CNN model as well as determining the effect of using an ADAM optimizer. The results are provided in Figure A10.

As can be seen from Figure A10, we observe the highest-to-lowest accuracy when the momentum is set to 0.9, 0.5, 0, and 2. Moreover we have also determined that the Adams optimizer performs poorly compared to the SGD optimizer.

We believe the CNN performs better with higher momentum (up to a reasonable level) because it becomes easier for the model to escape local minimum thus allowing the model to explore the parameter space more effectively.

From the plot, we can infer that as the momentum level increases up to an understandable value (less than 1.0), the accuracy of the model also increases. Moreover, it can be concluded that as the momentum of the model increases, we introduce oscillations into the equation. These oscillations can be visibly seen when plotting the loss values as the training continues. In terms of convergence speed, we observe a positive correlation between momentum rate and time spent to reach 20 epochs.

Table 1 Time spent to complete 20 epocs (in seconds)

Momentum 0	**Momentum 0.5**	**Momentum 0.9**	**Momentum 2**
465.35	470.51	470.68	479.52

When we compare the Adams optimizer instead of SGD optimizer, we observe lower accuracy results. It is observed that it takes slightly less time to converge, but lacks performance in terms of accuracy. We believe this to be the case due to gradient scaling and sensitivity to noisy gradients. Adam optimizer is known to use squared gradients and this can multiply the impact of noisy gradients in our model. Furthermore, we also suspect that vanishing gradients could also be a cause for lower accuracy and higher completion time for Adams optimizer.

4 Discussion and Conclusions

In this assignment, we have investigated how different hyperparameters effect the accuracy and convergence time for MLP and CNN networks.

For the MLP, we conclude that selecting the best learning rate requires multiple trials and can be a good starting point to observe how our model reacts under different conditions. Moreover, we understood the impact of how activation layers, depth of network, regularization, and normalization effect accuracy, loss, and time required to reach an optimal accuracy level.

In terms of CNN's, we have explored the impact of momentum and optimization methods on our accuracy and overall performance of the network. We concluded that increasing the momentum to a certain level helps improve accuracy and that we should always take our dataset into account when determining which optimization method to choose.

In terms of the comparison between MLP and CNN's, we understood that CNN's perform better on image classification due to their convolution stage in terms of time and accuracy.

5 Statement of Contribution

The contributions of the group members are as follows:

- Okyanus Gumus: Writing the assignment report.
- Mihail Calitoiu: Implementing the ML models.
- Eren Gurhan: Analyzing the datasets.

Appendix A Screenshots

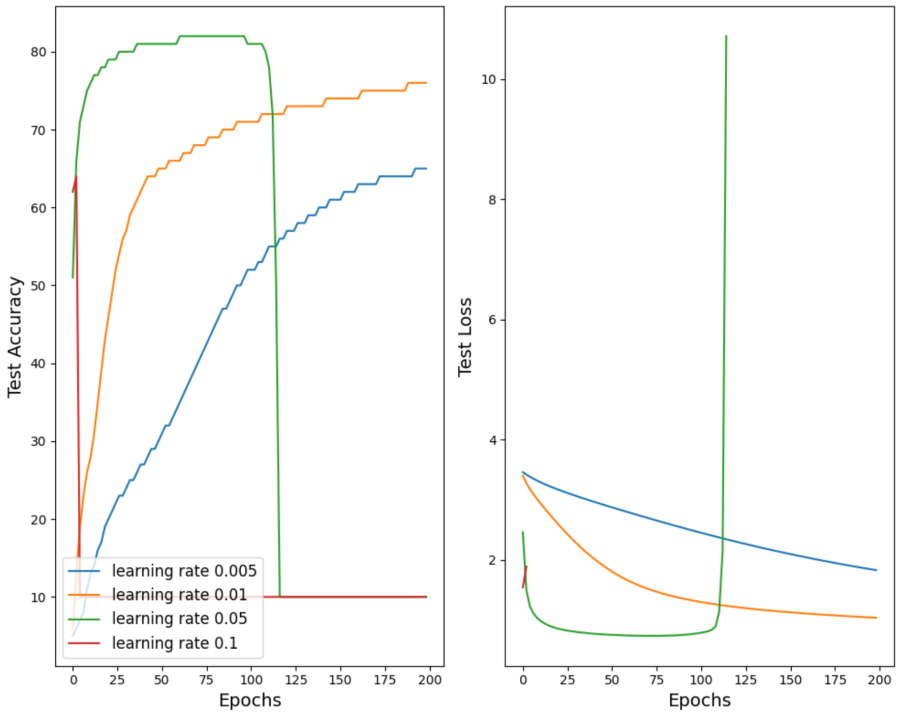

Fig. A1 Test accuracy and test loss with different learning rates.

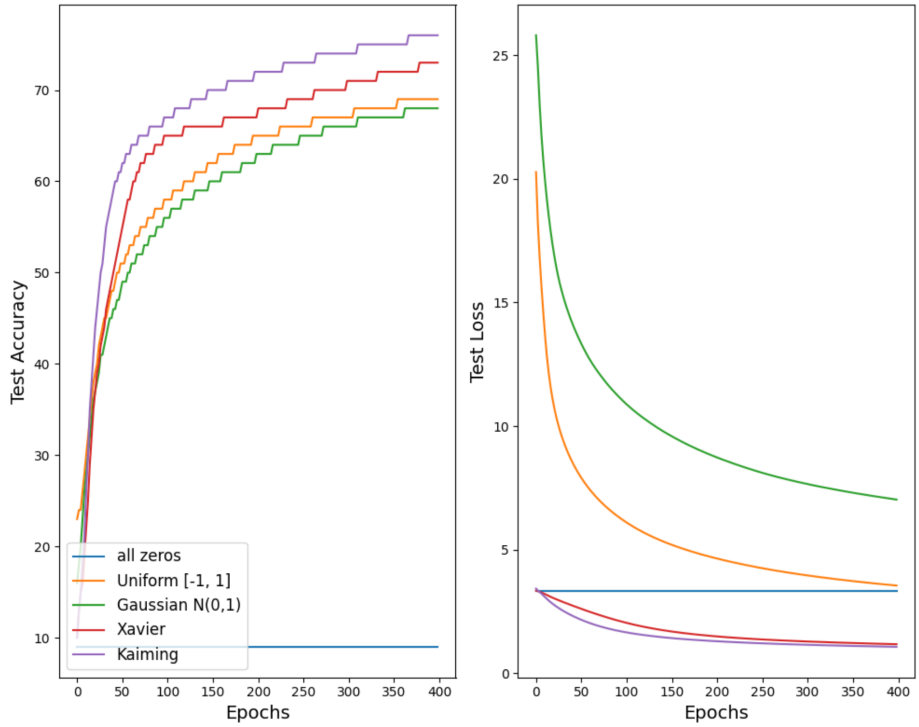

Fig. A2 Test accuracy and test loss with different weights.

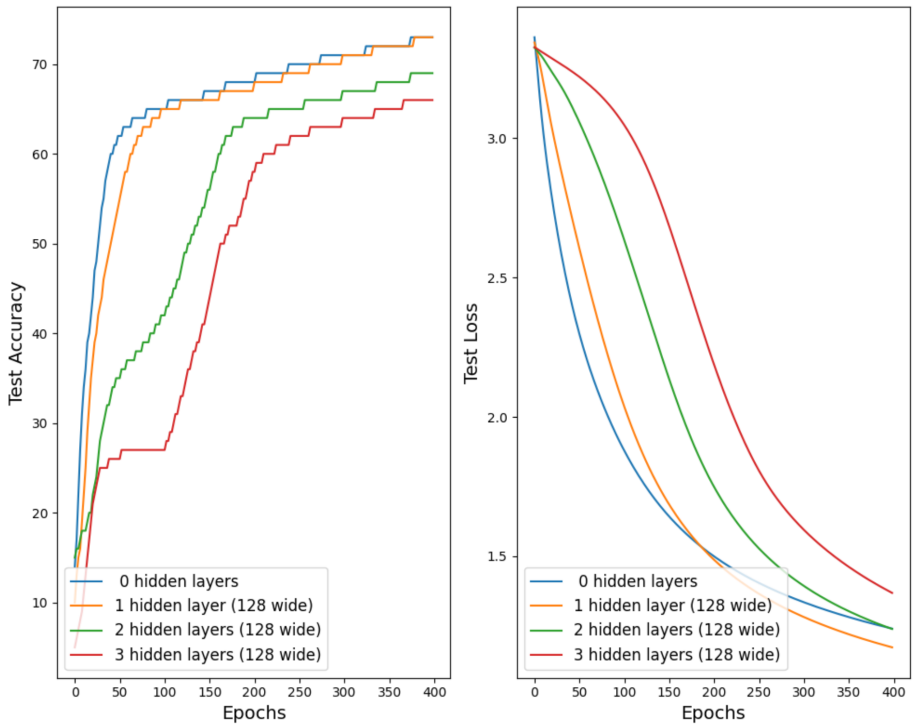

Fig. A3 Test accuracy and test loss with different layers.

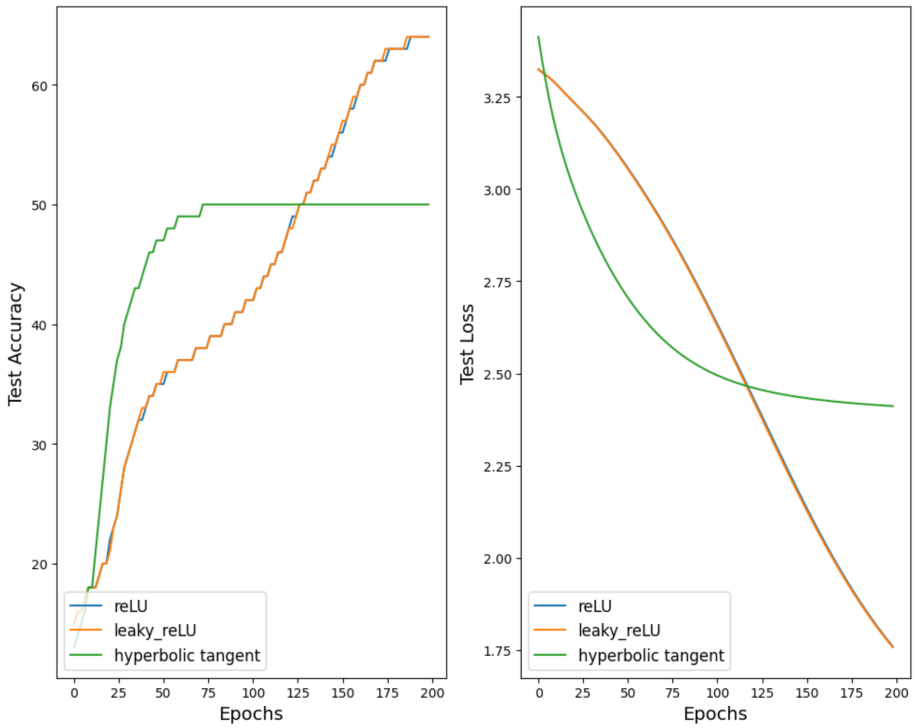

Fig. A4 Test accuracy and test loss with different weights.

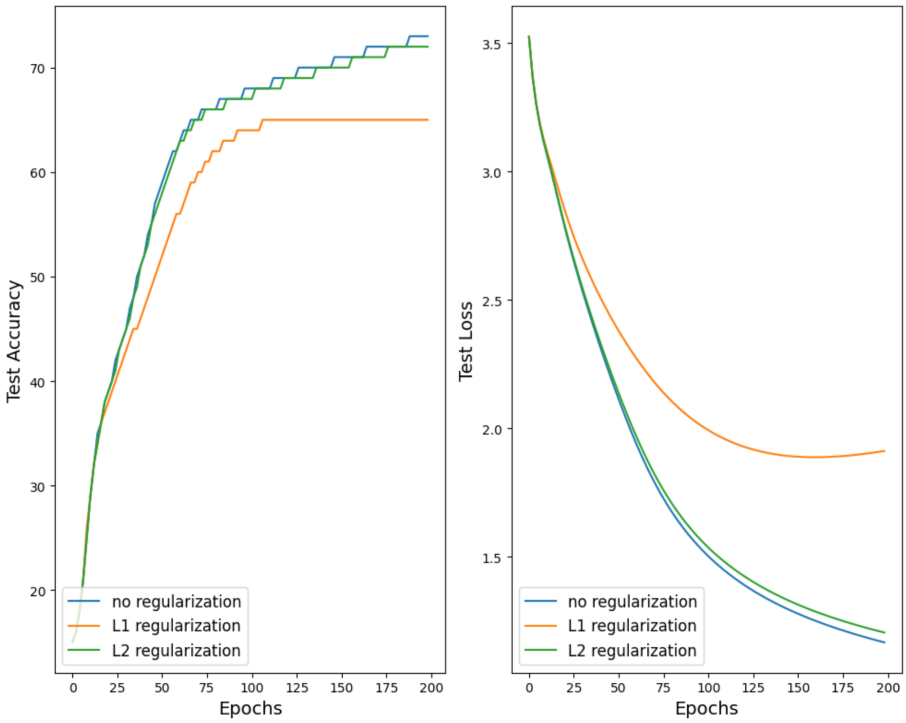

Fig. A5 Test accuracy and test loss with different regularizations.

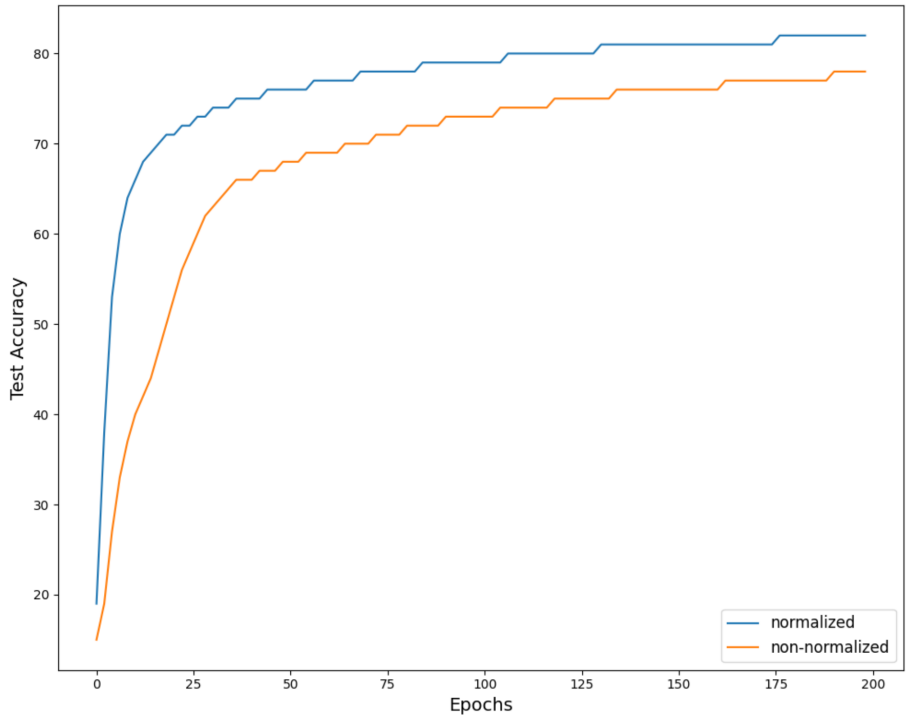

Fig. A6 Test accuracy with and without normalization.

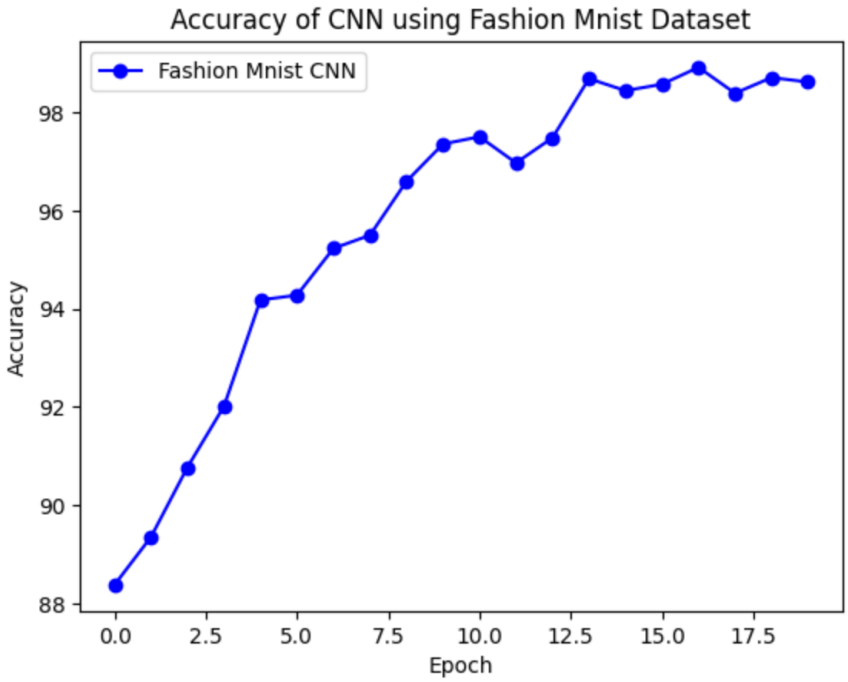

Fig. A7 Accuracy of the CNN with the Fashion MNIST dataset.

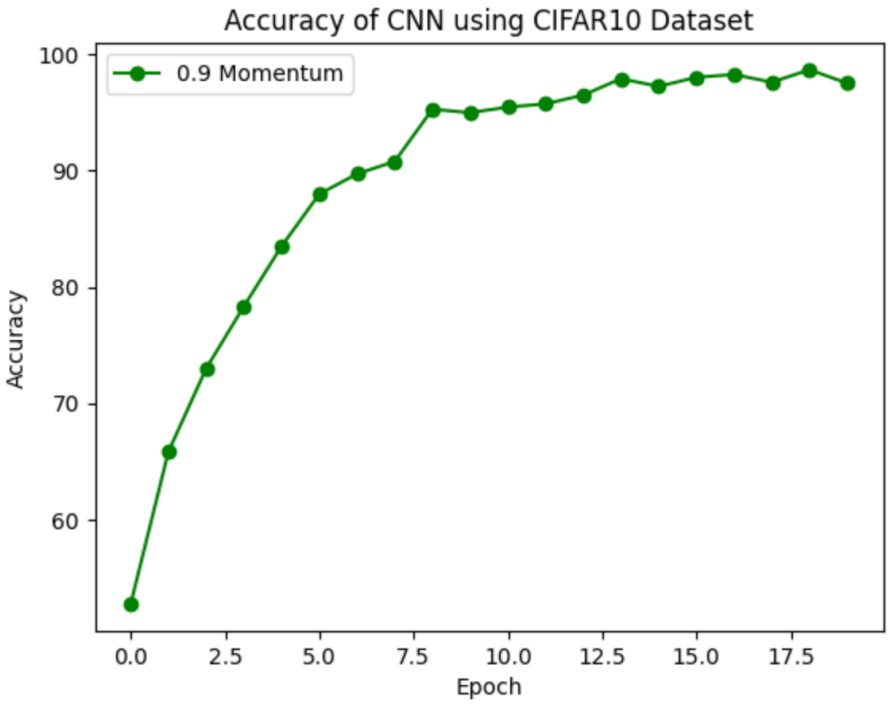

Fig. A8 Accuracy of the CNN with the CIFAR-10 dataset.

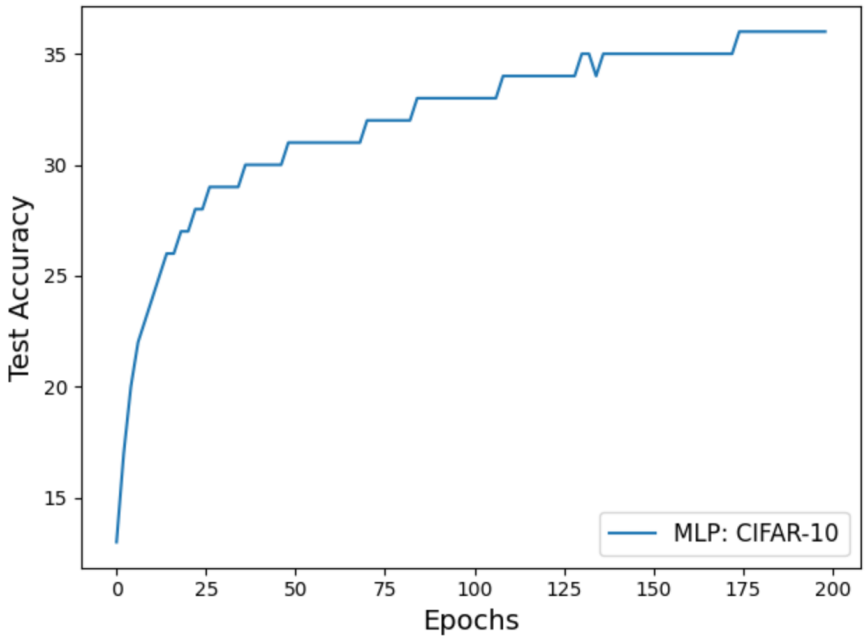

Fig. A9 Accuracy of the MLP with the CIFAR-10 dataset.

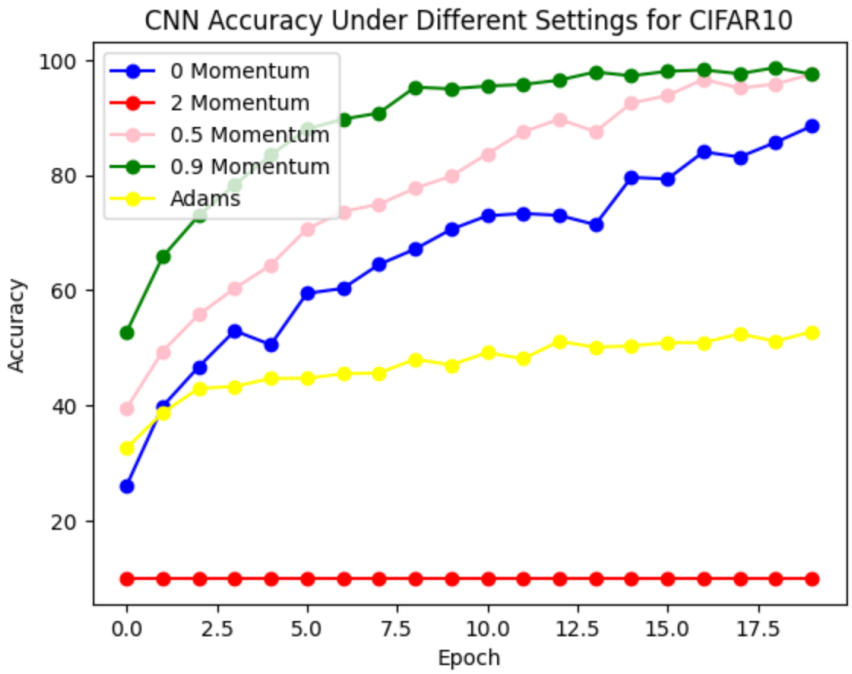

Fig. A10 Accuracy of the CNN with different momentum.

References

[1] Xhaferra, E., Cina, E., Toti, L.: Classification of standard fashion mnist dataset using deep learning based cnn algorithms. In: 2022 International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), pp. 494–498 (2022). https://doi.org/10.1109/ISMSIT56059.2022.9932737

[2] Kayed, M., Anter, A., Mohamed, H.: Classification of garments from fashion mnist dataset using cnn lenet-5 architecture. In: 2020 International Conference on Innovative Trends in Communication and Computer Engineering (ITCE), pp. 238–243 (2020). https://doi.org/10.1109/ITCE48509.2020.9047776

[3] Teow, M.Y.W.: Experimenting deep convolutional visual feature learning using compositional subspace representation and fashion-mnist. In: 2020 IEEE 2nd International Conference on Artificial Intelligence in Engineering and Technology (IICAIET), pp. 1–6 (2020). https://doi.org/10.1109/IICAIET49801.2020.9257819

[4] Thakkar, V., Tewary, S., Chakraborty, C.: Batch normalization in convolutional neural networks — a comparative study with cifar-10 data. In: 2018 Fifth International Conference on Emerging Applications of Information Technology (EAIT), pp. 1–5 (2018). https://doi.org/10.1109/EAIT.2018.8470438

[5] Doon, R., Kumar Rawat, T., Gautam, S.: Cifar-10 classification using deep convolutional neural network. In: 2018 IEEE Punecon, pp. 1–5 (2018). https://doi.org/10.1109/PUNECON.2018.8745428

[6] Aslam, S., Nassif, A.B.: Deep learning based cifar-10 classification. In: 2023 Advances in Science and Engineering Technology International Conferences (ASET), pp. 01–04 (2023). https://doi.org/10.1109/ASET56582.2023.10180767

[7] Tan, H.H., Hann Lim, K.: Minibatch approximate greatest descent on cifar-10 dataset. In: 2018 IEEE-EMBS Conference on Biomedical Engineering and Sciences (IECBES), pp. 320–323 (2018). https://doi.org/10.1109/IECBES.2018.8626708