Classification of Image Data using MLP and CNN

Nicholas Dahdah (260915130), Jonayed Islam (260930902), Jasper Yun (260651891)

Abstract

Computer vision has become an important application of machine learning, with widespread use in autonomous driving algorithms, medical diagnoses, and even cashier-less grocery stores. Image classification is at the core of computer vision, and we implemented multilayer perceptron (MLP) models and convolutional neural network (CNN) models to classify images on the Fashion-MNIST dataset. We used the torchvision module to pre-process the data for each model. Our best MLP and CNN models achieved maximum training accuracies of 76.3% and 96.0%, respectively, with corresponding test accuracies of 75.5% and 88.9%. The CNN outperformed the MLP by a vast margin at the cost of higher model complexity. The MLP performed best with zero hidden layers, but we limited training times during experimentation. Given more training time, the MLP with two hidden layers using ReLU activation would likely perform slightly better. We also found that MLP models with nonlinear activation functions did not lead to significantly better performance, thus the dataset is "linear enough" for simple models to still perform well. While this performance does not surpass human expert performance, our classifiers can be used to quickly classify images.

Introduction

Image classification is a popular application of machine learning (ML) at the heart of computer vision. For humans, constantly monitoring and analyzing massive volumes of image data is time-consuming and inefficient. However, ML models can quickly and cost-effectively perform this task. We implemented multilayer perceptron (MLP) and convolutional neural network (CNN) models for image classification using the Fashion-MNIST dataset [1]. The creators of the dataset [1] compared the performance of classifiers on the Fashion-MNIST and MNIST datasets, achieving up to 89.7% test accuracy on the Fashion-MNIST dataset with an SVC with C=10. The highest-performing MLP achieved a test accuracy of 87.1% [1]. In [2], they achieved a test accuracy of over 98% using the CNN-based LeNet-5.

Datasets

The Fashion-MNIST dataset was used. This dataset contains 28 x 28 grayscale images with each image representing one of ten clothing item types such as a shirt, a shoe, or a bag. The training set contains 60,000 labeled examples, while the test set contains 10,000 labeled examples. Both the training and test sets have perfectly equal distributions of each of the ten classes. The data was loaded, normalized, and shuffled using the torchvision module. As the original images contained 784 pixels with each having a value between 0 and 256, we normalized the pixel values to be between 0 and 1. For the MLP, the images were flattened from a 28*28 matrix into a vector of size 784. For training the CNN model, the image matrixes were transformed into tensors which were then sent to a GPU for parallelized computing.

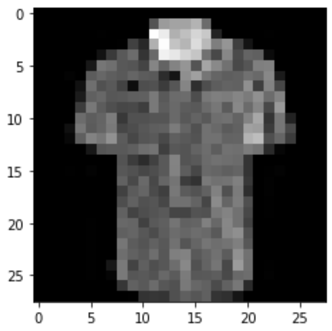

Fig. 1. Example of a Shirt from the Fashion-MNIST Training Set

Results

Of all the models tested, the CNN with optimized hyperparameters performed the best, with a test accuracy of 88.9%. The results for all models are presented in Table 1. We note that the models in Table 1 have been trained on design-chosen hyperparameters that we believe best balance the accuracy-latency trade-off. The choice of model hyperparameters is justified by the results in tables 2 and 3.

Table 1. MLP and CNN Training and Test Accuracies

Model	Training Accuracy	Test Accuracy
MLP with no hidden layers (LR=0.2, epochs=100)	76.29%	75.49%

MLP with 1 hidden layer and ReLU activation (LR=0.2, epochs=100)	68.74%	67.98%
MLP with 2 hidden layers and ReLU activation (LR=0.3, epochs=200)	69.31%	68.64%
MLP with 2 hidden layers and tanh activation (LR=0.3, epochs=200)	72.57%	71.52%
MLP with 2 hidden layers and Leaky-ReLU activation (LR=0.3, epochs=200)	67.49%	66.35%
MLP with 2 hidden layers and sigmoid activation* (LR=0.3, epochs=200)	49.09%	49.55%
MLP with 2 hidden layers and ReLU activation with drop-out (LR=0.3, epochs=200)	65.23%	64.04%
MLP with 2 hidden layers and ReLU activation with drop-out and weight decay* (LR=0.3, epochs=200)	60.26%	59.63%
MLP with 2 hidden layers and ReLU activation (Unnormalized data, LR=0.1, epochs=200)	12.68%	9.74%
Convolutional neural network with optimized hyperparameters	96.01%	88.87%

* Supplementary experiments

We note that since the training and test accuracy of each model are very similar, none of the models are overfitting. A graph of training accuracy as a function of epoch is shown in Fig. 2, demonstrating the learning process of a 2-layer MLP with ReLU activation and a learning rate of 0.3. In this figure, the model steps over the minimum gradient of the loss function periodically but reaches a strong training accuracy after 200 training epochs. Given enough compute time, this issue could be rectified by using a smaller learning rate and more training epochs. Please note that, in general, we did not train the models for more than 250 epochs, as suggested by the TA, David Venuto. For reference, please see the following Ed discussion post: https://edstem.org/us/courses/18448/discussion/1344162.

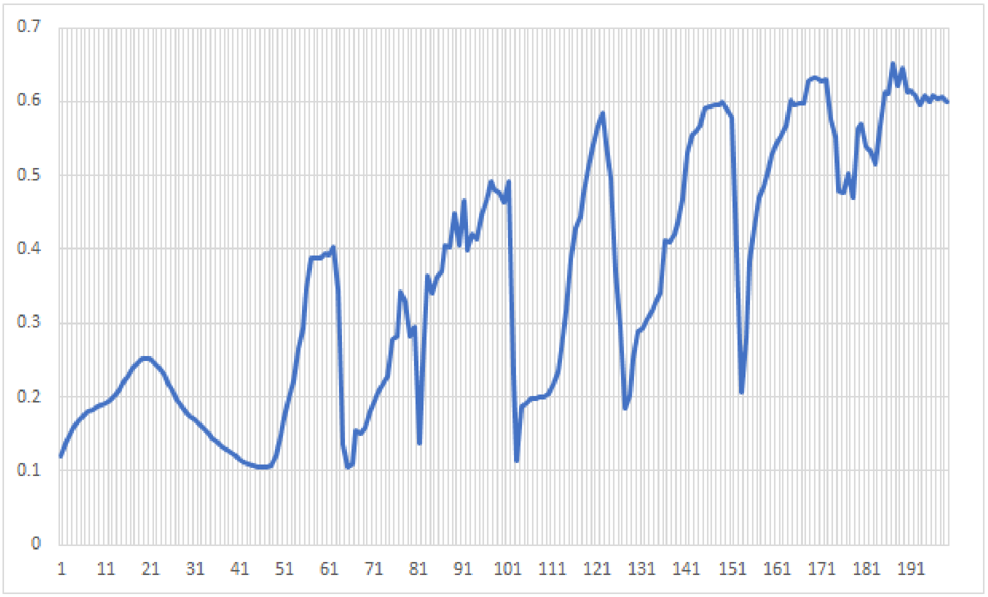

Fig. 2. Training Accuracy vs. Epochs of MLP with 2 hidden layers and ReLU activation (LR=0.3, epochs=200)

Table 2. MLP Test Accuracies During Hyperparameter Tuning

0 Hidden Layers

Learning Rate (Epochs = 50)	
Learning Rate	Test Accuracy
0.05	68.05%
0.1	71.88%
0.2	**72.24%**

Epochs (LR = 0.2)	
Number of epochs	Test Accuracy
10	60.86%
50	72.24%
100	**75.49%**

1 Hidden Layer (ReLU)

Learning Rate (Epochs = 50)	
Learning Rate	Test Accuracy
0.05	45.15%
0.1	58.14%
0.2	**61.97%**

Epochs (LR = 0.2)	
Number of epochs	Accuracy
10	23.11%
50	61.97%
100	**67.98%**

2 Hidden Layers (ReLU)

Learning Rate (Epochs = 200)	
Learning Rate	Test Accuracy
0.1	43.87%
0.2	59.74%
0.3	**68.64%**

Epochs (LR = 0.3)	
Number of epochs	Test Accuracy
50	19.85%
200	**68.64%**
400*	75.30%

* Note we did not use 400 epochs for later experiments due to the high computation time. Therefore, we used 200 epochs since it provided sufficient test accuracy.

Based on the results of the experiments in Table 2, we selected the optimal hyperparameters. For the learning rate, we simply picked the value that yielded the highest accuracy since changing the learning rate does not affect the training time of the MLP. For the number of training epochs, we had to choose the hyperparameter that best balanced the accuracy and the training time of the MLP. This was particularly relevant when evaluating a two-layer MLP, where we decided to run our final test experiments using 200 training epochs instead of 400, despite 400 epochs yielding a higher accuracy. We made this decision because the model took a very long time to train with 400 epochs.

Task 3.1

The performance of MLPs with no hidden layers, one hidden layer with 128 hidden units and ReLU activation, and two hidden layers with 128 hidden units each and ReLU activations can be found in rows 1-3 of Table 1. We notice that for the chosen values of learning rate and maximum gradient iterations (or epochs), the model with no hidden layers performs best with a test accuracy of 75.49%. The corresponding confusion matrix is shown in Fig. 3. From the confusion matrix, we see strong identification of all articles of clothing except shirts, which the model instead misclassified as T-shirts, pullovers, or coats.

In theory, the non-linearities introduced in the models with hidden layers should allow the models to fit non-linear data. We believed that this would increase testing accuracy since we believe that Fashion-MNIST was not linearly separable. Knowing that the model with no hidden layers attained a 75.49% test accuracy, we can say that Fashion-MNIST is “linear enough” to be fit well with a linear model. In practice, the models with non-linear activations did not perform better. However, we believe that this is due to training epochs and not the introduction of non-linearities. Given enough epochs, we believe it would perform better.

We initially believed that a deeper model with non-linearities would be more expressive and thus, would be more accurate. We believe we did not get this result because deeper models require more training epochs to achieve higher testing accuracy, and so, given the equitable amount of training time we allocated to our models, the more complicated models were not trained to completion. We suspect that given more training epochs, the deeper models would be able to generalize better, as long as they do not overfit (which is possible due to their high expressiveness).

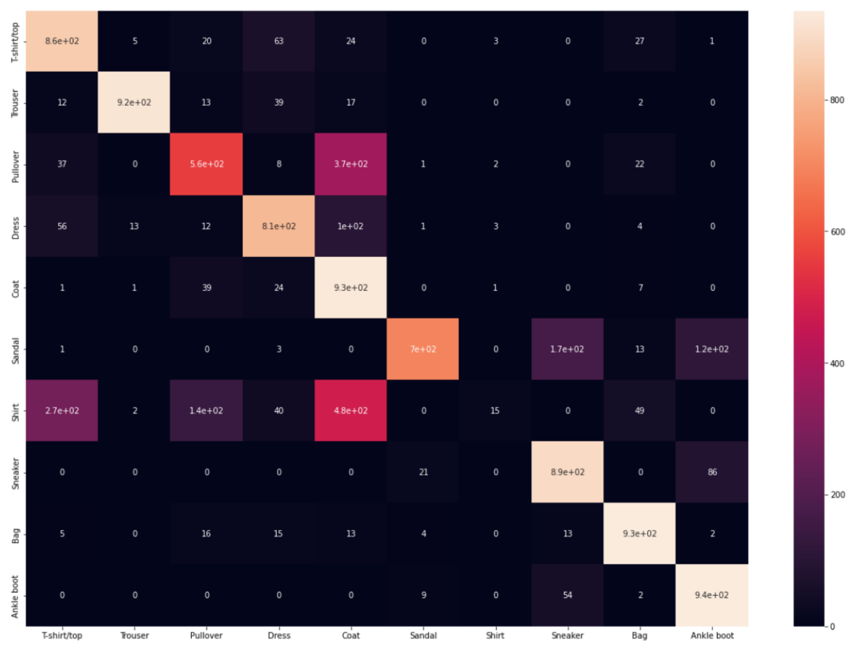

Fig. 3. Confusion Matrix of the Fashion-MNIST Prediction on Test Set with optimized 0-Layer MLP Model

Task 3.2

We compared the performance of two-layer MLPs with different activation functions. The non-linear activations functions we tested were ReLU, tanh, Leaky-ReLU, and, additionally, sigmoid. The tanh activation led to the best testing accuracy (71.52%), followed by ReLU (68.64%), then Leaky-ReLU (66.35%), and then sigmoid (49.55%). The tanh function works best in our case due to its steeper gradients compared to the other activation functions, which increases the rate of learning. Tanh, however, does suffer from vanishing gradients. ReLU and Leaky-ReLU performed similarly, despite their noticeable difference when values are negative. This implies that ReLU's zone-of-no-learning does not affect the overall performance. Sigmoid performed the worst due to its non-steep gradient, which corresponded to low rates of learning. We expected tanh, ReLU, and Leaky-ReLU to perform better than sigmoid. We expected tanh to perform worse than ReLU and Leaky-ReLU due to vanishing gradients, but it did not end up being an issue.

Task 3.3

We then added drop-out regularization and, additionally, weight decay. Drop-out is implemented by randomly eliminating a hidden unit with a probability of 5% (which can be adjusted). Weight decay penalizes weights by a factor proportional to the square of the corresponding weight. An MLP with two hidden layers, ReLU activation, and drop-out regularization achieved a test accuracy of 64.04%, which is very similar to the same model without drop-out (68.64%). An MLP with two hidden layers, ReLU activation, drop-out regularization, and weight decay achieved a test accuracy of 59.63%. These regularization techniques negatively affected accuracy due to the fact that the models were not overfitting to begin with. As such, they reduced the expressiveness of the model unnecessarily. We note that these techniques would positively affect the test accuracy if the model was overfitting to the training data.

Task 3.4

Using unnormalized images, the model was unable to accurately predict the outputs of both seen and unseen data. The training accuracy was 12.68% and the test accuracy was 9.74%. Normalization, in general, speeds up learning. From the achieved accuracies, we observe that the model does not have the ability to generalize as it has not sufficiently learned the data. Given more training epochs, the model may be able to achieve a higher training and test accuracy using unnormalized data.

Task 3.5

We implemented a CNN model and tuned four of its hyperparameters: learning rate, input batch size, number of training epochs, and choice of optimizer. We found that a learning rate of 0.005 was optimal. We also saw that a smaller batch size of 200 performed better on the test set as it helped reduce overfitting. Allowing the model to train for 250 epochs gave it more time to converge on a solution. The choice of optimizer was the hyperparameter that had the most significant impact as Adam led to an 86.7% accuracy while SGD and AdaGrad had an accuracy of 72.6% and 72.7%, respectively. Due to limited computing capabilities, we tuned each hyperparameter one at a time by setting all the other hyperparameters to a constant value. These results are presented in Table 3.

Table 3. CNN Test Accuracies During Hyperparameter Tuning

Learning Rate (other hyperparameters const.)	
Learning Rate	Test Accuracy
0.001	86.6%
0.005	**88.6%**
0.01	88.3%

Batch Size (other hyperparameters const.)	
Batch Size	Test Accuracy
200	**88.3%**
2000	85.3%
5000	84.2%

Epochs (other hyperparameters const.)	
Number of epochs	Test Accuracy
50	86.0%
100	87.2%
250	**88.8%**

Optimizer (other hyperparameters const.)	
Optimizer	Test Accuracy
Stochastic Gradient Descent	72.6%
Adam	**86.7%**
AdaGrad	72.7%

After tuning the CNN model, we achieve a test accuracy of 88.9%. The corresponding confusion matrix is shown in Fig. 4. When looking at the confusion matrix, we notice that most of the model's misclassifications occur between the shirt, T-shirt, and coat categories, much like the MLP. This makes sense as these items often have a similar appearance. For example, a coat has the same appearance as a shirt, but it may be larger.

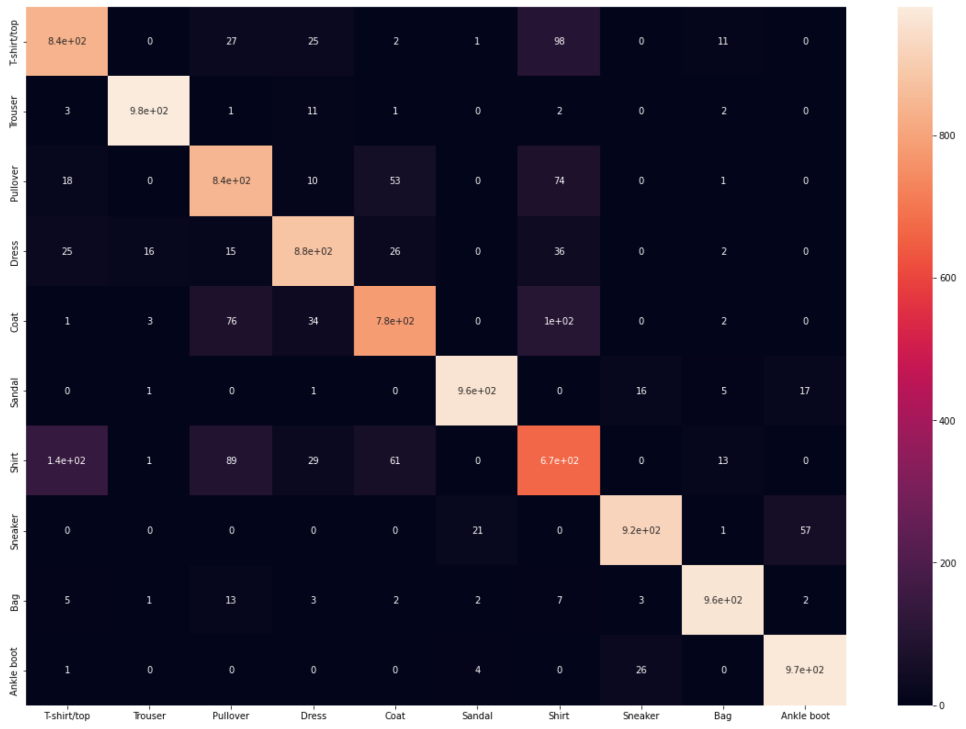

Fig. 4. Confusion Matrix of the Fashion-MNIST Prediction on Test Set with optimized CNN Model

Task 3.6

Based on our experiments in Table 1 and our choice of hyperparameters in Table 2, we constructed an MLP architecture that we believed would perform as well as possible. This architecture consists of 2 hidden layers, each with tanh activation. We chose tanh because of how well it performed in Task 3.2. We trained the model with a learning rate of 0.3 and let it train for 400 epochs, unlike our other models. As expected, this model outperformed all the MLP models in Table 1, achieving a training accuracy of 76.68% and test accuracy of 76.03%. This MLP does an adequate job of classifying fashion images but does not achieve the same level of training and test accuracy as a tuned CNN (96.01% and 88.87%, respectively).

Discussion and Conclusion

Overall, the CNN model performed better than the MLP models on the Fashion-MNIST dataset. We note that data normalization is essential for the MLP to be able to perform image classification efficiently. For the MLP, we further note the effects of changing the learning rate and the number of training epochs. Increasing the learning rate and increasing the number of training epochs improved the training and test accuracy. However, increasing the learning rate too high will cause the model to miss the minimum of the training loss and potentially diverge. Lastly, we note the effect of drop-out regularization and weight decay. In our experiments, these two types of regularization did not provide a significant change in accuracy because our models are not overfitting. If the models were overfitting (due to more training epochs), we believe that drop-out and weight decay would help. For the CNN, we note the effect of the batch size and choice of optimizer. A low batch size improved accuracy. Choosing Adam as an optimizer greatly improved the CNN's ability to classify images. Both the MLP and CNN performed well on the Fashion-MNIST dataset, but would not be more accurate than a human evaluator. Therefore, these models should only be used for quick classification.

It is also important to acknowledge the computational limits that we faced when completing this project. We were limited by both memory availability and runtime. Our accuracy was limited by the number of training epochs (< 250). Furthermore, when performing hyperparameter tuning on both the MLP and CNN, we were forced to tune one parameter at a time as grid search would be too computationally intensive and would take much too long.

To improve the performance of these models in future work, it would be interesting to explore different model variations, given more compute resources. For MLPs, this would involve increasing the number of hidden layers and hidden units. It would also involve varying the activation functions so that not every hidden layer has the same activation function. It would also be interesting to add convolutional and pooling layers with different dimensions to the CNN. These modifications to the architectures of the MLP and CNN could potentially yield better test accuracy.

Statement of Contributions

Nicholas and Jasper divided the work required for the training and experimentation of the MLP. Jonayed processed the data and handled the training and experimentation for CNN.

References

[1] H. Xiao, K. Rasul and R. Vollgraf. “Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms,” arXiv:1708.07747 [cs], Sep. 2017.

[2] M. Kayed, A. Anter and H. Mohamed, "Classification of Garments from Fashion MNIST Dataset Using CNN LeNet-5 Architecture," 2020 International Conference on Innovative Trends in Communication and Computer Engineering (ITCE), 2020, pp. 238-243, doi: 10.1109/ITCE48509.2020.9047776.