COMP 551 ASSIGNMENT 3

April 4, 2022

Mingze Li, Shelley Xia, Yiwei Cao

Abstract

In the present report, we implement a MLP model with mini-batch gradient descent from scratch and achieve the highest accuracy of 89.06%. We find that the model archives higher testing accuracy with more training epochs. Implementing drop-out regularization reduces the variance of the model performance, for there is no significant observed overfitting as the number of epochs grows among the models with drop-out regularization. Furthermore, our experiments show that compared to models with one or two hidden layers, models with no hidden layers are more prone to overfitting. Additionally, we find that recruiting more hidden units (i.e. increasing model width) improves model performance. Feeding the model normalized input data leads to a much better model performance than using unnormalized input data as well.

Introduction

Multilayer Perceptron (MLP) models, a branch of artificial neural networks, take inspiration from the biological nervous system which uses a dynamic system of the activation and passing of electrical signals to perform complicated tasks. Here we implement a multilayer perceptron from scratch and analyze its performance on image classification using the Fashion-MNIST dataset with varying activation function, regularization and normalization methods. A dataset of 60,000 28x28 grayscale images of 10 fashion categories with 10,000 test set images, the Fashion-MNIST dataset is a standard dataset for deep learning algorithms in image classification [1]. Up until the submission of this report, the best performing model on this dataset is the Fine-Tuning DARTS, with an accuracy rate of 96.91% [2]. Their model takes a Neural Architecture Search approach, which automates the design of artificial neural networks instead of hand-designing the architecture [3]. Models using this method have been shown to have the capability of outperforming some hand-designed architectures[4].

Our multilayer perceptron model reaches its highest accuracy performance of 89.06%, with 1 hidden layer, 256 hidden units, 200 training epochs, and Relu activation function. We designed multiple experiments to test the effects of model hyperparameter and design have on the model performance. We observed that increasing the model width, the number of training epochs and using data normalization all lead to increased accuracy rate. Additionally, we find that models with 0 hidden layers are more prone to overfitting than models with 1 or 2 hidden layers. To inquire into the influence of regularization methods on model variance, we implemented a drop-out regularization and found that it indeed reduces the model variance and prevents overfitting. To our surprise, we did not find a linear relationship between the number of hidden layers and model accuracy. As for the effects of different activation functions on model performance, we found that ReLu, leaky-ReLU and tanh activation functions lead to similar model accuracy, which is not too surprising when considering all three have been proven to be successful and commonly used activation models for neural networks. Lastly, we experimented with convolutional neural networks with the help of Pytorch and achieved an accuracy of 87.03% with only five training epochs. If given more time, we are sure that we would achieve higher testing accuracy with more training epochs.

Datasets

For this project, we used the Fashion-MNIST dataset [1]. The Fashion-MNIST is a new standard dataset for computer vision and deep learning [5]. It was created after the well-known MNIST handwritten digit datasets in order to provide a more complex and challenging benchmark dataset for machine learning tasks. Its inputs are Zalando's article images and outputs are one of the ten types of clothing to which these images belong, such as T-shirts, trousers, pullovers, and dresses. Specifically, each input instance is a 28x28 grayscale image. There are 60,000 training instances and 10,000 testing instances in total.

Given the popularity of this dataset, many machine learning libraries have included it in their databases. Hence, we took advantage of Keras and loaded the dataset from it directly. For simplicity and comparability, we kept the default train and test partitions. Since MLP takes in inputs in vectorized forms, we vectorized each input so it is a 1-D vector of size 784 instead of a 2-D matrix of size 28x28.

Unnormalized data could be seen as unbalanced weights given to equally important features. This imbalance would then traverse through layers of MLP and result in a biased model. Hence, data normalization is crucial prior to training. We performed 3 kinds of data normalization on vectorized data [6]. First, for each dimension of input, we subtracted its mean from it and divided it by its standard deviation. Then each dimension had a mean of 0 and values between -1 and 1. The resulting data was named zero-centered data. Secondly, we tried Principle Component Analysis (PCA), which focused on a subset of features that contain the most variance. The resulting data was named PCA reduced data. Thirdly, we did a whitening transformation, which changed each vector to a white noise vector. The resulting data was named white data. After each of these normalizations, we visualized changes by displaying the first 5 instances.

Results

1. Non-linearity and depth

We trained three models with different numbers of hidden layers (0, 1, and 2) and ReLU activation function. The latter two models have 128 units in each hidden layer. We observed that the accuracies on the testing data are 82.71%, 84.21%, and 84.08%, for the 0, 1, and 2 hidden layer(s) models respectively. Although the accuracies for the 1 and 2 hidden layers model are slightly higher than the 0 hidden layers model, this result is surprising since the performances of the three models are similar. We expected to see a significant improvement by increasing the depth of the neural network and using non-linear activation functions.

Notably, we also observed that the model with no hidden layer is more prone to overfitting (Figure 1). As the number of epochs increases, the accuracy of this model on the training set and test set quickly diverges.

Additionally, we found that for all three models, setting the learning rate to 0.1 yields the best result compared to 0.001 and 0.01 (Table 1). We then used this learning rate for all other models in later parts the the task.

Figure 1. Model performance as a function of epoch

(a) 0 hidden layer (b) 1 hidden layer (c) 2 hidden layers

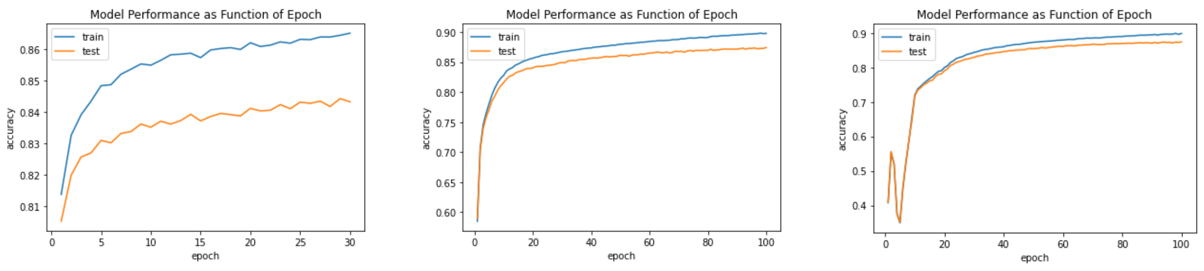


Table 1: prediction accuracies for different models (columns) and different learning rates (rows)

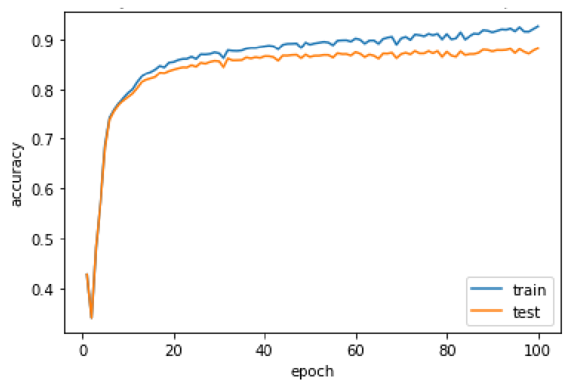
	0 hidden layer	1 hidden layer	2 hidden layers
alpha = 0.001	65.88%	59.14%	22.98%
alpha = 0.01	76.64%	70.51%	34.35%
alpha = 0.1	82.71%	84.21%	84.08%

2. Leaky-ReLU and tanh activation functions

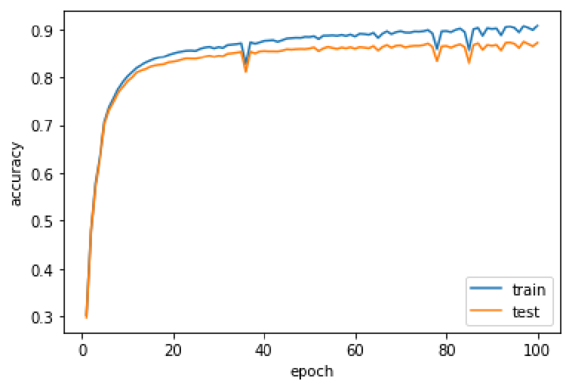
We observed similar performance with the leaky-ReLU and tanh activation functions, with the former yielding an 84.24% accuracy and the latter 83.24%. This result is not very surprising since although each activation function has its pros and cons, all three of them were proven effective by previous research and all are commonly used for training neural networks. ReLU may perform slightly better than tanh because it is less susceptible to vanishing gradients [7]. Leaky-ReLU is expected to perform better than ReLU since it fixes the dying ReLU problem (neurons becoming inactive with negative values) [8].

Figure 2. leaky-ReLU and tanh model performance as function of epoch

(a) leaky-ReLU



(b) Tanh



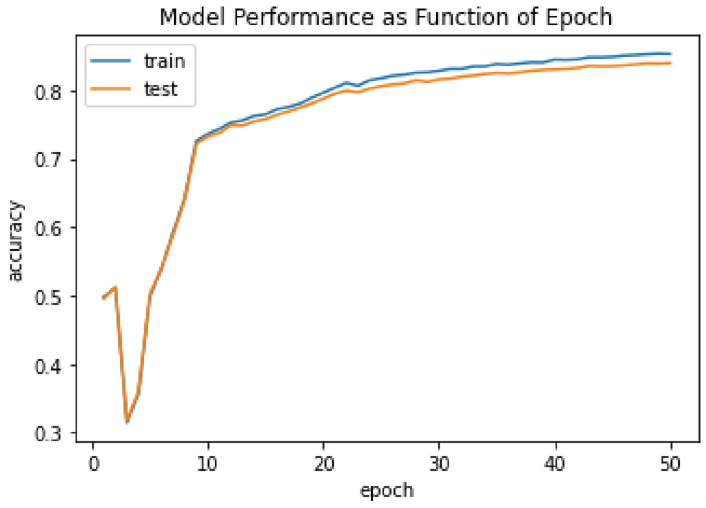
3. Dropout regularization

We next added dropout regularization to a model with 2 hidden layers and ReLU activation function. Specifically, there is a 50% chance that a neuron in the hidden layers could be dropped. At first, we found that the accuracy after adding dropout regularization is 84.22%, which is surprising since we expected an increase in accuracy. Since dropout regularization mainly deals with overfitting, this result may have occurred because we only trained the model for 20 epochs and the overfitting issue is not present yet. Therefore, we then trained the model with 200 epochs. The accuracy increased to 86.83%.

4. Unnormalized images

All previous models have used data that have undergone normalization. To test the effect of normalization, we next trained a model using unnormalized data. The accuracy turned out to be only 10%. Although we do expect a significant decrease in performance, such a low accuracy could also be due to the presence of NaN values during training.

Figure 3. Model performance after dropout regularization



5. CNN with PyTorch

After some hyperparameter tuning, the parameters of the best CNN model are obtained and shown in Table 2. However, these models are only trained with 5 epochs to speed up the process. We then rerun the best model with 30 epochs and the resulting accuracy is 87.03%, which is higher than previous MLP models. It is expected that CNN performs better than MLP on image classification tasks since CNN is more sensitive to spatial correlations, is translation invariant, and detects image features.

Table 2: hyperparameter tuning for CNN.
Note that c1 is the number of output channels in the first convolutional layer and c2 is the number of output channels in the second convolutional layer.

c1	c2	Kernel dimension	accuracy
16	32	2*2	76.48%
16	32	3*3	78.87%
16	32	5*5	75.02%
32	64	3*3	75.28%

6. Best MLP model

We first tested the effect of different normalization methods. Centering the data by subtracting the mean and dividing by the standard deviation yields better results than PCA and whitening (Figure 4). Looking at the result of previous models, we chose a model with one hidden layer and ReLU activation function as a baseline for further testing. We then experimented with the number of hidden units. Table 3 presents the accuracies of four different models with increasing number of hidden units. Each model is trained for 200 training epochs.

As expected, recruiting more hidden units (i.e. increasing width) improves performance. The fact that these models yield higher accuracy than the CNN model above is likely due to the difference in training epochs. Given more time, we would tune the CNN model better. We expect that the best CNN model outperforms the best MLP model.

Figure 4. Comparing different data regularization methods

(a) zero-center (b) PCA (c) whitening

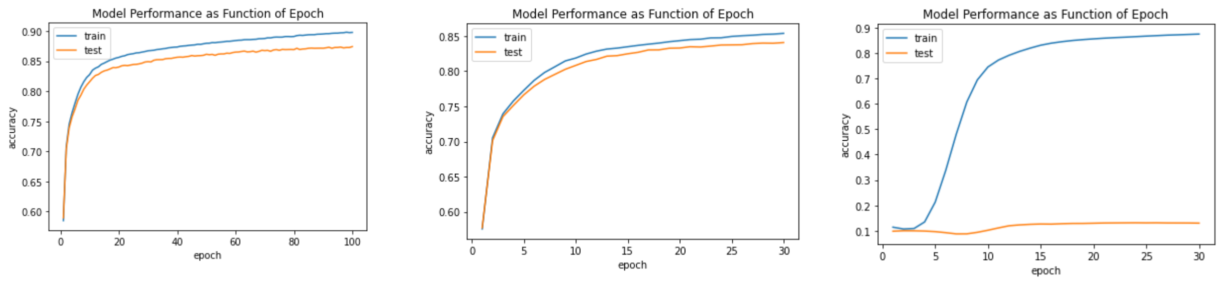


Table 3. Effect of increasing hidden units on prediction accuracy

# hidden units	64	128	192	256
accuracy	86.38%	88.61%	89.00%	89.06%

Discussion

Multilayer Perceptron models (MLP) have been widely used in image classification in machine learning tasks [9]. In this project, we trained multiple variations of MLP on the Fashion-MNIST dataset [1]. In our implementation of the MLP model, there is freedom of choice in the number of hidden layers, hidden units, epochs, batch sizes, and different types of activation functions. There were also hyperparameters to tune such as the learning rate in the Gradient Descent algorithm when updating the weights during training. We explored how variations made from different choices affected model accuracies. We also compared our best-performing MLP with CNN created by existing libraries.

We found that normalized data performed much better than unnormalized data. We also observed that model accuracies increased as the number of hidden layers increased. Similarly, model accuracies were positively correlated with the number of hidden units in each layer. Furthermore, when we increased the number of epochs, the model accuracies increased accordingly. It is worth noting that we did not spot overfitting within 100 epochs, although model accuracies may decrease after a certain large number of epochs due to overfitting. Activation function-wise, leaky ReLu performed slightly better than ReLu and tanh. This could be due to the leaky ReLu allowing errors to propagate through the network. Last but not least, we saw a great improvement in accuracy after applying the dropout technique, as expected.

Although there were some differences in results, we noticed that our model predictions did not respond markedly to our manipulations of hyperparameters. In future work, we would like to investigate what may be some of the causes. Future work may also include more hyperparameter tuning, such as finding the optimal p during dropout and using pooling techniques in CNN.

Statement of Contribution

Shelley focused on data extraction, data pre-processing, and writing the dataset and discussion section of the writeup. Yiwei ran all experiments and wrote the result section of the writeup. Mingze implemented the multilayer perceptron model with gradient descent and wrote the abstract and introduction section of the manuscript.
All three sacrificed sleep, health and hope for a bright future. (Fig. 5 in Appendix a.).

References

[1] Zalandoresearch. (n.d.). *Zalandoresearch/fashion-mnist: A mnist-like fashion product database. benchmark*. GitHub. Retrieved April 4, 2022, from https://github.com/zalandoresearch/fashion-mnist

[2] Tanveer, M. S., Karim Khan, M. U., & Kyung, C.-M. (2021). Fine-tuning darts for Image Classification. 2020 25th International Conference on Pattern Recognition (ICPR). https://doi.org/10.1109/icpr48806.2021.9412221

[3] Elsken, Thomas; Metzen, Jan Hendrik; Hutter, Frank (August 8, 2019). "Neural Architecture Search: A Survey". Journal of Machine Learning Research. 20 (55): 1–21. arXiv:1808.05377. Bibcode:2018arXiv180805377E – via jmlr.org

[4] Zoph, Barret; Vasudevan, Vijay; Shlens, Jonathon; Le, Quoc V. (2017-07-21). "Learning Transferable Architectures for Scalable Image Recognition". arXiv:1707.07012 [cs.CV].

[5] Brownlee, Jason. "Deep Learning CNN for Fashion-Mnist Clothing Classification." Deep Learning CNN for Fashion-MNIST Clothing Classification, Jason Brownlee, 27 Aug. 2020, https://machinelearningmastery.com/how-to-develop-a-cnn-from-scratch-for-fashion-mnist-clothing-classification/.

[6] CS231N convolutional neural networks for visual recognition. (n.d.). Retrieved April 4, 2022, from https://cs231n.github.io/neural-networks-2/#datapre

[7] Brownlee, J. (2021, January 21). *How to choose an activation function for deep learning*. Machine Learning Mastery. Retrieved April 4, 2022, from https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/

[8] Kumawat, D. (n.d.). 7 types of activation functions in neural network. Analytics Steps. Retrieved April 4, 2022, from https://www.analyticssteps.com/blogs/7-types-activation-functions-neural-network

[9] Khalid, Irfan Alghani. "Multilayer Perceptron for Image Classification." Medium, Irfan Alghani Khalid, 10 May 2021, https://towardsdatascience.com/multilayer-perceptron-for-image-classification-5c1f25738935.

Appendix A

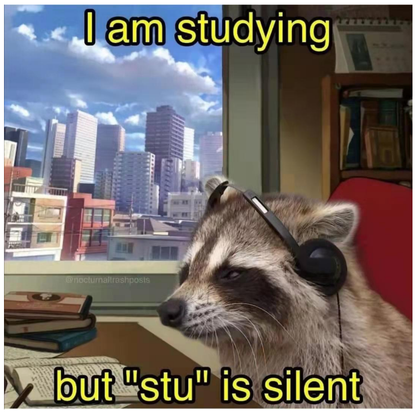


Figure 5.