AMATH 482/582: HOME WORK 4

XINYUE LYU

Amazing Department, University of Washington, Seattle, WA
xlyu6@uw.edu

Abstract. This report is about using Fully-Connected neural network to classify images from the FashionMNIST dataset. The aim is to improve the neural network model by hyperparameter tuning such as adjusting the number of layers and learning rates. Besides, this report also discusses the impact of different optimizers including RMSProp, Adam and SGD with different learning rates. By analyzing training dynamics and classification accuracy, this assignment highlights the effectiveness of different configurations and optimization strategies in improving neural network performance.

1. Introduction and Overview

People use deep learning models in various areas to extract complex patterns behind data. This method often suffer from unstable training due to issues such as vanishing or exploding gradients, poor weight initialization, and overfitting. In this report, I explore the impact of the number of neurons in each layer, Kaiming (He) Uniform initialization, Batch Normalization, and Dropout on a fully connected neural network's training and test accuracy.

I designed a fully connected neural network model and analyze how different settings affect training convergence, stability, and classification accuracy. By comparing different setups, my aim is to evaluate their individual and combined effects. My findings highlight the trade-offs between stability and generalization, providing insights into best practices for designing deep learning architectures.

2. Theoretical Background

a. **Neural networks and ReLU**: This model might be inspired by how neurons work in our brain. It consists of neurons(or perceptrons) as input and output. Between the input and output layer, there are hidden layers that are linear combinations of neurons in the former layer with some weights and bias.

Putting the different results of the linear combination to an activation function(add non-linearity to the model), we get neurons that make another layer. In this assignment, I used ReLU as the activation function for each layer. This activation function outputs the input if the input is positive, and zero otherwise.

Fully connected neural network(FCN) refers to the situation that every neuron in one layer is connected to every neuron in the next layer.

b. **Cross Entropy loss**: It works as a loss function to calculate the differences between predictions generated from the model and the real data. Cross Entropy loss measures the difference between the predicted probability distribution and the true (or actual) distribution of labels. It is

Date: March 18, 2025.

calculated as:

$$\frac{1}{N}\sum_{i=1}^{N}(y_i \log(p_i) + (1 - y_i)\log(1 - p_i))$$

where N is the number of instances, y_i is the true label for instance i, and p_i is predicted probability by the model.

c. **Gradient Descent**: A gradient decent algorithm helps people find the best performance of a model by minimizing the loss function. In this assignment, I performed Stochastic Gradient Descent(SGD), Root Mean Square Propagation(RMSProp), and Adaptive Moment Estimation(Adam) optimizer with different learning rate (the step size during training).

d. **Overfitting/underfitting**: If a model fits very well to the training data and can not generalize for another unseen data, it is overfitting. If a model is too simple, it is underfitting, resulting in poor performance in training and testing data.

Dropout regularization randomly sets some nodes to zero with probability p during each forward pass. It assists in reducing overfitting.

e. **Kaiming(He) Initialization**: It is a technique for initializing the weights of neural networks. The Kaiming initialization is calculated as a random number with a Gaussian probability distribution with a mean of 0.0 and a standard deviation of $\sqrt{(\frac{2}{n})}$, where n is the number of inputs to the node[1].

f. **Batch Normalization**: This is a technique that normalizes activations within each mini-batch. It adjusts and scales activations using:

$$\hat{x} = \frac{x - \mu}{\sigma}$$

$$y = \gamma\hat{x} + \beta$$

where μ, σ are the mean and standard deviation of the batch and γ, β are learnable parameters.

3. Algorithm Implementation and Development

The primary algorithm used in this assignment is a fully connected neural network (FCN), enhanced with Dropout regularization, Batch Normalization, and Kaiming (He) Uniform initialization to stabilize training and improve accuracy.

For implementation, I use **PyTorch**, providing efficient tensor computations and automatic differentiation. The **torch.nn** module facilitates defining network layers, applying Dropout, and integrating Batch Normalization. Additionally, **Matplotlib** is used for visualizing training progress, while **NumPy** aids in numerical operations.

4. Computational Results

From Figure 1, we can know that as the number of epoch increase, the training loss of my model decreases to around 0.2, and the validation accuracy also increases in general trend. The training loss reduces rapidly for the first half of the picture and slowly decreaces after around 2 epochs. The fluctuation of the validation accuracy might indicate that my model is not stable.

I compared the accuracy with different optimizers and learning rates to the original model with 85.16% test accuracy. From the results showed on Table 1 and the Figure 2, we can easily find that the best optimizer for my model is Adam optimizer with 0.005 learning rate. The model returned 87.53% test accuracy, performing better than the original model using SGD and 0.05 learning rate.

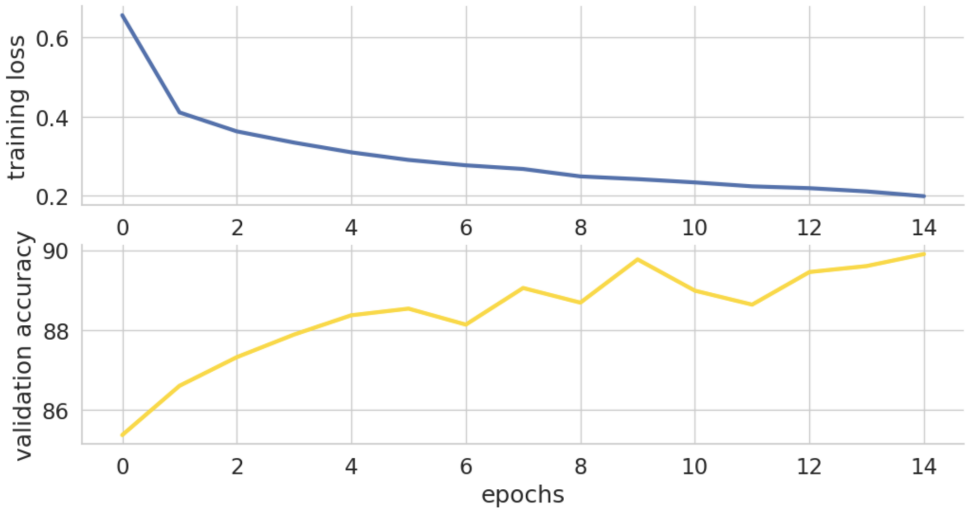

FIGURE 1. This figure shows how the training loss(Cross Entropy loss) and validation accuracy of my model change as the number of epochs increases.

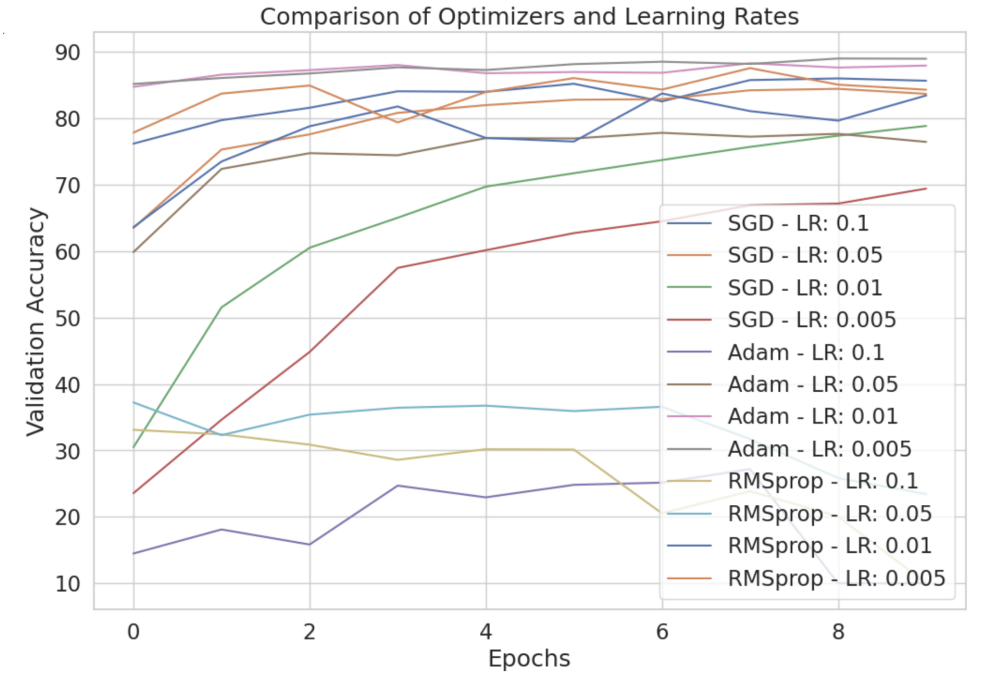

FIGURE 2. This figure shows how validation accuracy changes as the number of epoch increases for different optimizers and different learning rates.

Optimizer	0.1 LearningRate	0.05 LearningRate	0.01 LearningRate	0.005 LearningRate
SGD	86.39	83.85	79.04	70.75
Adam	19.67	77.94	87.35	87.53
RMSProp	10.4	11.92	81.05	85.44

TABLE 1. This table shows the test accuracy(in percentage) for three optimizer with four different learning rate, aiming to find the most suitable one for my FCN model.

Most lines are increasing as the number of epoch increase, but several of them are decreasing and remain very low values, which are caused by the learning rate. Large learning rate can cause underfitting since it prevent the model from learning data patterns.

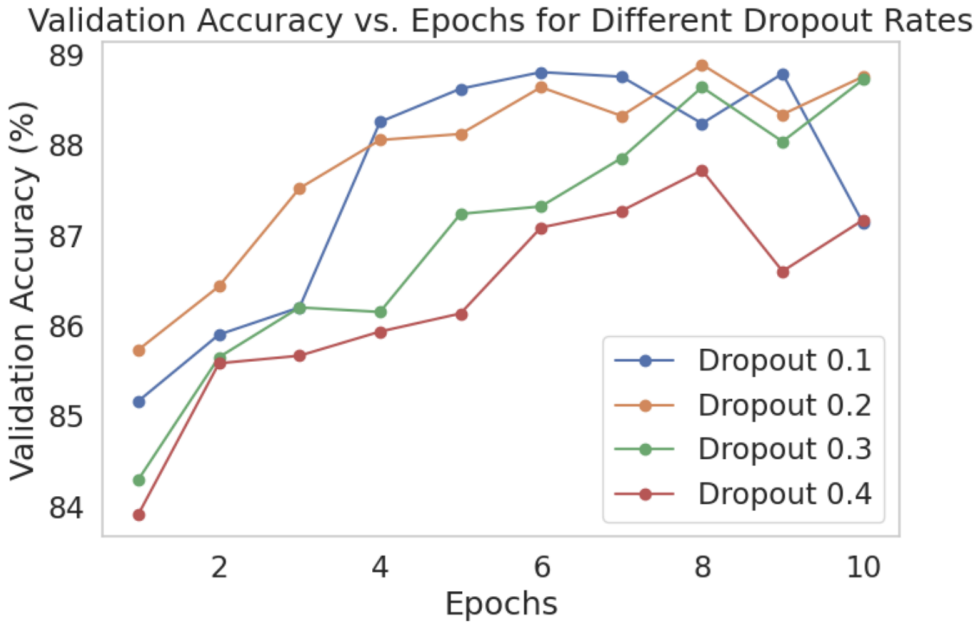

FIGURE 3. This figure shows how validation accuracy changes with different dropout probabilities.

To prevent overfitting and underfitting, I add dropout regularization to my model with several probabilities, $0.1, 0.2, 0.3, 0.4$. The higher the probability, the lower the chance of overfitting. From the accuracy results on Figure 3, I find that the model will perform better with a lower dropout rate such as 0.2. As the rate increases, my model tends to be underfitting with a lower accuracy. The best test accuracy with dropout rate 0.2 is 87.51%. The dropout regulation doesn't change the accuracy of my model too much.

Then, I explored how Kaiming(He) Uniform Initialization. The test accuracy is 86.92%, a little lower but not too much from the result without the initialization. The validation accuracy generally increases while the training loss decreases as the number of epoch increases, shown on Figure 4. The times of training time that are above 16s for each number of epoch reduce after adding the initialization to the weights.

Lastly, the test accuracy of the model with Batch normalization is 87.57, greater than 86.92%. It improves my model. My model only has three layers with each layer containing around 200 neurons. The reason why the effects of the initialization and normalization are not obvious may be the simplicity of my model.

5. Summary and Conclusions

In this assignment, I performed herperparameters tuning to find the best parameters for my Fully-Connected Neral Network model. The model aims to classify 28×28 images of FashionMNIST dataset. The best optimizer for my model is Adam with learning rate 0.005. The test accuracy generally increases as I adding 0.2 dropout rate, Kaiming initialization, and batch normalization

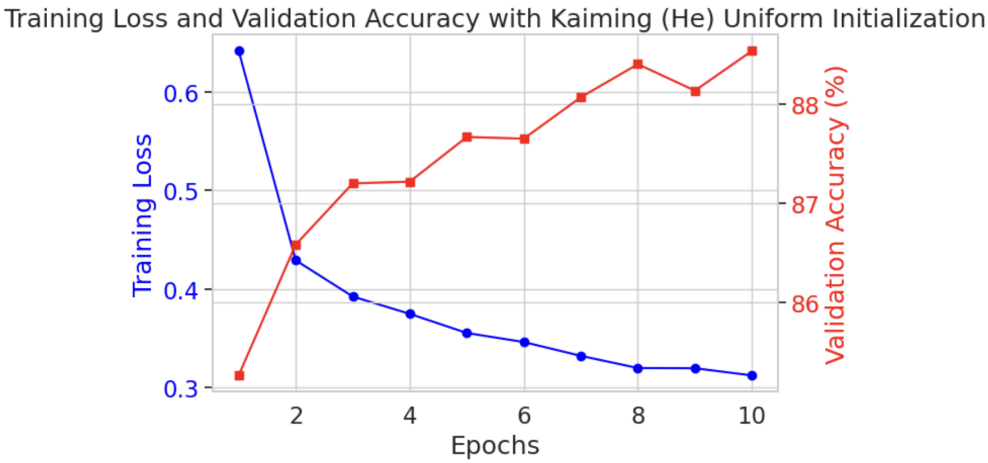

FIGURE 4. This figure shows validation accuracy and training loss after performing Kaiming Uniform initialization to weights of the neural network.

to the model. For a better test accuracy, I need to change hyperparameters such as increasing the numbers of neurons and adjusting the dropout rate.

ACKNOWLEDGEMENTS

The author is thankful to Dr.Frank for instructing the concepts of neural network, gradient descent, and the coding techniques along with the textbook[2].

REFERENCES

[1] GreeksforGreeks. Kaiming initialization in deep learning, 27 Dec, 2023.
[2] J. N. Kutz. *Data-driven modeling & scientific computation: methods for complex systems & big data*. OUP Oxford, 2013.