Homework 4 - Part 5 Report

TensorFlow/Keras - Fashion MNIST Classification

Mai Ngo

DSC 578 Neural Network and Deep Learning - SEC 701

Dr. Noriko Tomuro

October 18, 2023

Table of Contents

First Experiment

To start the experiment, the original Neural Network (NN) with one hidden layer size (1, 128, 10) yields Cross Entropy (CE) loss between the labels and predictions of 0.2375, and accuracy of 91.11% on training set. While on test set, CE loss yields 0.3241 with associated accuracy of 88.33%. Given this, my initial objective with this experiment is to improve the NN and achieve closer CE loss and accuracy differences between train and test sets, originally 2.78%.

For the first experiment, create a NN with two hidden Dense layers size (1, 128, 32, 10). I applied four activation functions RELU, SELU, SOFTMAX, and SWISH on the new hidden layer size 32. Using RELU activation function, I got pretty much the same output as the original model, 90.99% accuracy on train set and 87.21% for test set. Noticeably, when I re-ran the model several times, on average the second model yields better performance than the original mode.

In addition, SOFTMAX yields not as similar results, distinct losses increase, and accuracies decrease. On the other hand, both SELU and SWISH activation functions give similar performance as RELU. At this point so far, I conclude that two hidden layers NN both using RELU activation function performs best with an average of 2.05% accuracy difference based on several model ran. Therefore, I will use this model moving forward with the experiment.

2 Hidden Layers NN – 2nd hidden layer activation function.	Accuracy -Train	Accuracy - Test	Difference
RELU	**90.99%**	**87.21%**	**3.78%**
SELU	91.12%	88.27%	2.85%
SOFT MAX	88.93%	86.08%	2.85%
SWISH	91.16%	88.38%	2.78%

Table presents accuracy scores using different activation functions on the 2nd hidden layer.

Learning Rate

Initially I was thinking choosing a different optimizer algorithm. However, after several tries, I decided to stick with the original Adam algorithm, just intuitive decision. The article also mentions that Adam algorithm is computationally efficient and has little memory requirement so I thought it would be safer as well since I am testing parameters sequentially many times.

For learning rate, I tried with five different values. I started out with learning rate value of 0.03 then notice accuracies drop significantly. Now, I have a new additional expectation is to keep my NN's performance consistent; allowing somewhat 5% and 0.3 increase/decrease in accuracy and CE loss, respectively but no more than that. At learning rate of 0.01, the results start to be acceptable. Then I tried with smaller eta values, considering the differences in train and test set performance. Eventually, I decided to stay with learning rate of 0.01. The results stick to my objective, smaller accuracy difference of 1.64% | 87.37% for train and 85.73% for test set. Furthermore, I also noticed the smaller the learning rate, accuracy difference got bigger but smaller CE losses. Another reason I chose learning rate of 0.01 because if CE loss difference between train and test sets is not that significant compared to other eta values.

After this step, my current NN has size (1, 128, 32, 10), RELU activation function on the second hidden layer and learning rate of 0.01.

Learning Rate	Accuracy -Train	Accuracy - Test	Accuracy - Difference	Loss -Train	Loss -Test	Loss -Difference
0.03	71.77%	71.00%	1.77%	0.7354	0.7929	0.0575
0.01	**87.37%**	**85.73%**	**1.64%**	**0.3588**	**0.4355**	**0.0767**
0.0075	88.53%	86.54%	1.99%	0.3204	0.3971	0.0767
0.001	90.96%	88.44%	2.52%	0.2424	0.3413	0.0989
0.00075	95.05%	89.04%	6.01%	0.1311	0.4269	0.2958
0.0005	93.94%	89.42%	4.52%	0.1626	0.3457	0.1831
0.00025	95.57%	89.24%	6.33%	0.1166	0.3881	0.2715

Table presents train and test set performance on different learning rate.

Regularization

Moving on to regularization, this step took me a while to find the most optimal set of parameters. I started with using both L1 and L2 regularizations given default value of 0.01, the NN performance significantly drop. Then tried again with each regularization individually and still got the same results. Then I did some research on functionality of each regularization:

- L1 Regularization (Lasso) suppresses features (pixels) of each image potentially to be exactly zero if possible, basically doing feature selection which could end up predict 'shirt' and 'coat' as the same category. This is not ideal.
- L2 Regularization (Ridge) suppresses features (pixels) of each image as much as it can but not to be exactly zero. Given features (pixels) reasonable weight distribution. Which is more appropriate in this image classification task.

Applying L2 penalty on the layer's kernel with both values of 0.01 and 0.0001. The performance was not what I want ideally, training accuracy was around 70%. Added penalty on both layer's bias and output, results still not improve. Then I added dropout on both hidden layers with initial value of 0.3, tried seven or eight more times with different L2/dropout values, results remain the same. At this point, I traced back to learning rate. Eventually the best of combination that I move forward with are:

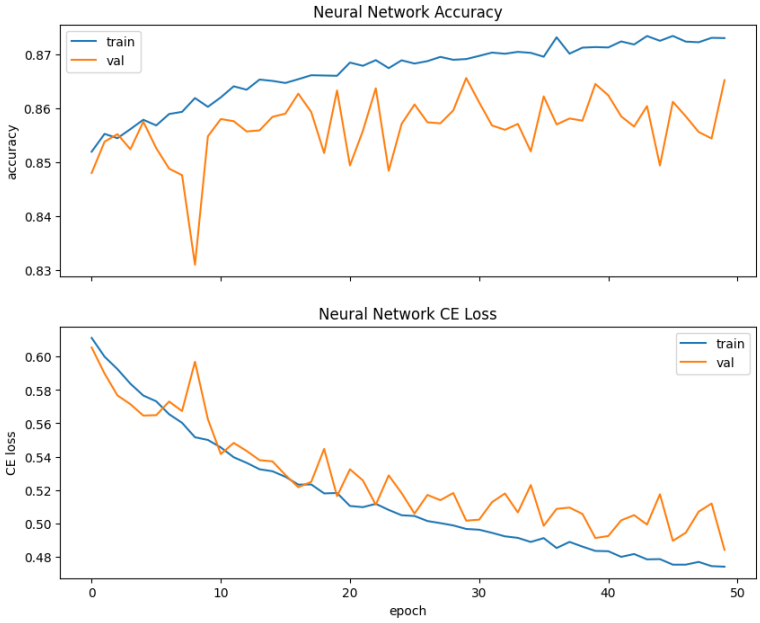
- L2 penalty on the layer's kernel/bias/output of 0.01
- Dropout rate of the first hidden layer of 0.3 and second hidden layer of 0.2
- Learning rate of 0.00025

This combination of parameters yields training accuracy of 83.23% - CE loss of 0.6960, evaluation accuracy of 84.37% and CE loss of 0.6478. Very small differences which I expect to not have underfitting/overfitting issue. I also want to comment further that if only using regularization L2, accuracies on average comes around 73%.

Dropout and Visualization

At this stage I have dropout implemented on both hidden layers. Follow the instruction, I only retain drop out on the second one, tried with both drop out value of 0.2 and 0.3. The final model using dropout of 0.2. The performance of both train and test sets improved, still stick to my initial objective of having small differences between training and test set performance: 85.04% and 85.09% accuracy, 0.6238 and 0.609 CE loss on train and test sets, respectively.

As for visualization, I re-fit the final NN using 50 epochs. The plot shows performance on both train and test data are pretty similar. Looking at the y-axis, the range of Accuracy and CE loss are not wide, which satisfies my expectation in this experiment; accuracy goes up and CE loss goes down. As for best performance indicators, I decided to stick with classification accuracy and CE loss since both provide straightforward interpretation. Given running NN time after time with different set of parameters, these two performance evaluation indicators give me the best guidance to intuitively choosing next step.



Throughout this experiment, my expectation after implement an additional parameter completely took a different turn each time. At least, I cannot sequentially/keep tuning new parameter without adjusting the previous ones. I understand there is no benchmark in what is deemed as best set of parameters, this assignment has given the experience in how combination of parameters can explicitly change the performance of a NN model. At the same, gaining intuition in data modeling.