CS6910: Assignment-1

Key Objective: Implementation of Feed Forward Neural Networks from Scratch and familiarisation with wandb.

Submitted by:
1. EE20S006 Snehal Singh Tomar
2. EE20D006 Ashish Kumar

Snehal Singh Tomar

Note to the evaluators:

During the course of writing this report, we experimented with a new set of hyper-parameters for Problem-8. These hyper-parameters ended up giving us the the best accuracy so far on the validation dataset. We were already done with most part of the report until then and could not afford to re-do the whole thing because of time constraints(we are already way-past the deadline). However, we have still kept the plot in our section for Problem-8. We have also made a recommendation for the most optimal set of hyper-paramters there. It would be great if the same could be considered as the final accuracy metric of our model during evaluation. The programs for Problem-8 can be found in the directory, "best-accuracy/" on our Github repository.

Problem 1

Training Data Specimen

Thus, we have displayed one image from each class of the fashion_mnist dataset along with its true label.

Problem 2

The program for Forward Propagation over the Neural Network has been implemented within out self-defined header, "myDLkit2.py" located in assignment-1-v-2/ on our submitted github repository.

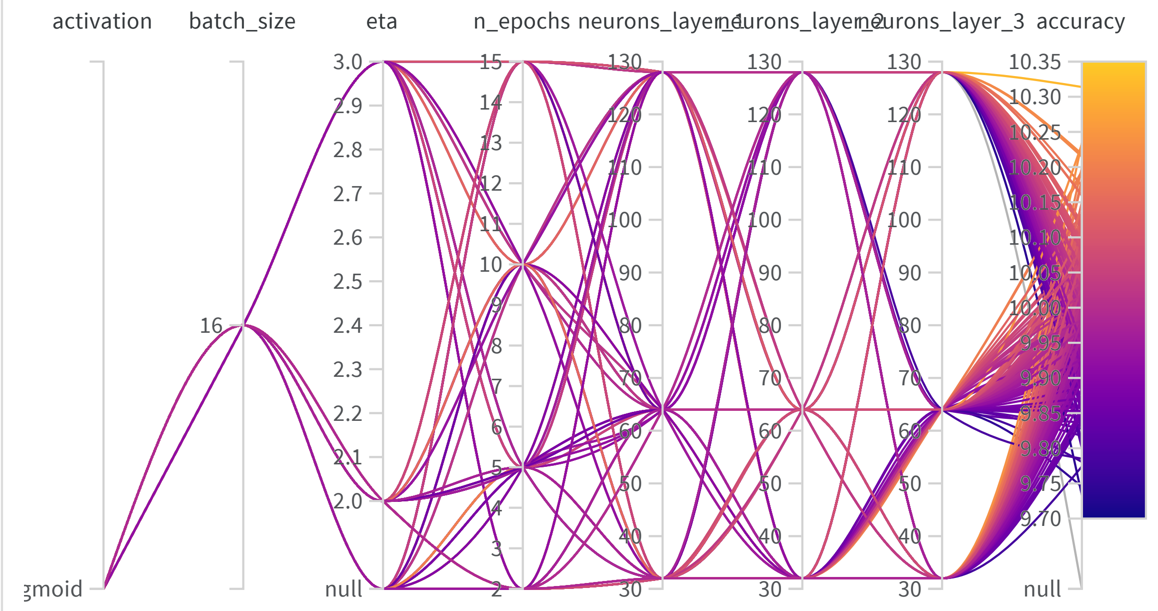
Overall path on github: CS6910-Assignment-1/assinment-1-v-2/myDLkit2.py/class feed_*fwd*_nn\forward_prop(function)

Problem 3

This problem required us to implement various optimizers for training our Neural Network using several (objective function, hyper parameter) combinations to determine the optimal set of hyper parameters for training the Neural Network.

i. SGD:

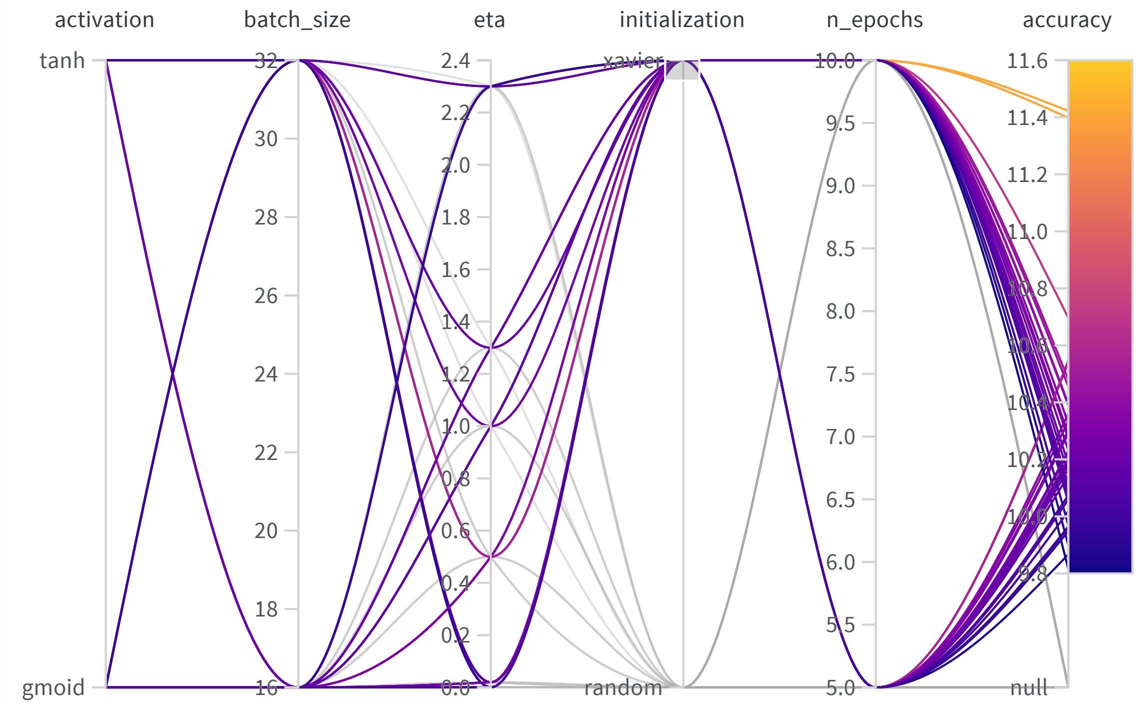
The following plot represents a summary of different sweeps corresponding to each set of hyper-paramets:



We trained the model over various sweeps and found in 'fast-sweep-273' we found that eta = 3e-2, number of epochs =10, and neuron configuration of [32, 32, 128] for the 3 hidden layers works best for the optimizer when sigmoid activation used. Similarly, we trained the model for other activation functions and batch sizes as well.

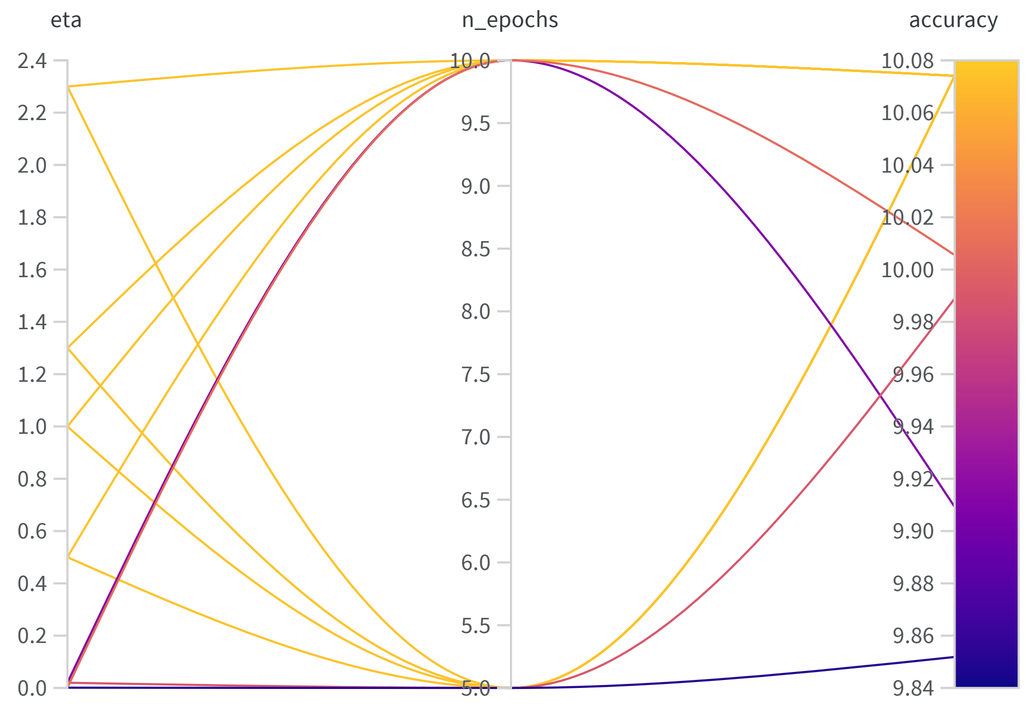
ii. Momentum Based Gradient Descent:

Analysis of the learning process using this optimizer for different hyper-parameter combinations is presented as below:



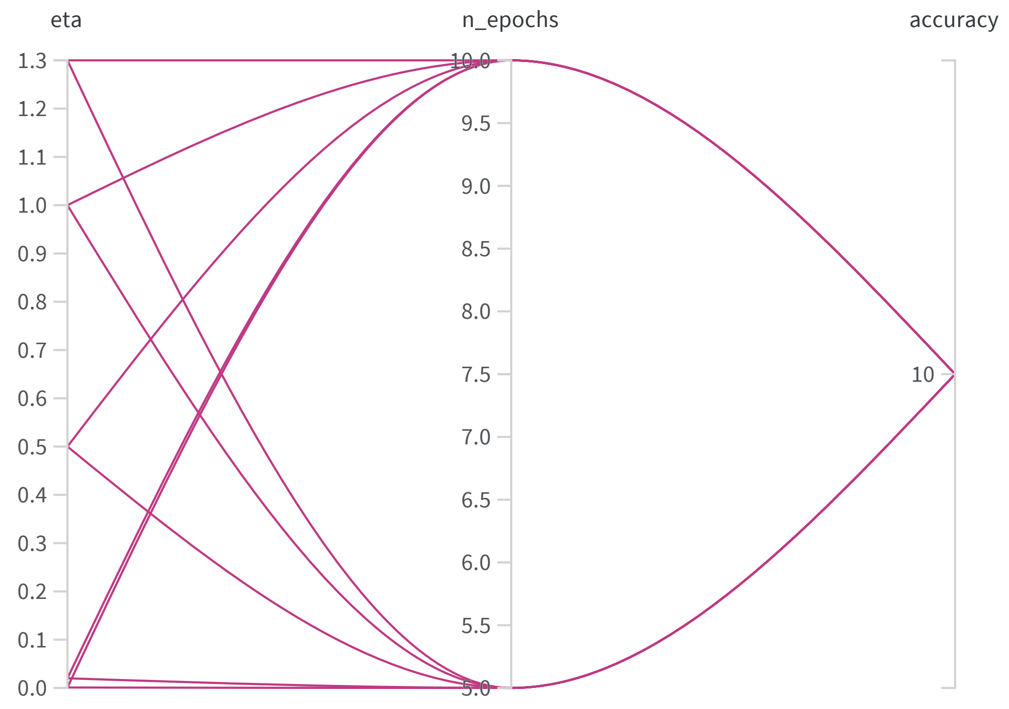
We trained the model over various sweeps and found in 'grateful-sweep-58' we found that eta = 0.5, number of epochs =10, batch size =16, tanh activation and Xavier Initialization of the Weights and Biases works best for the optimizer when sigmoid activation is used. Similarly, we trained the model for other activation functions and batch sizes as well.

iii. Nesterov Accelerated Gradient Descent (NAG):



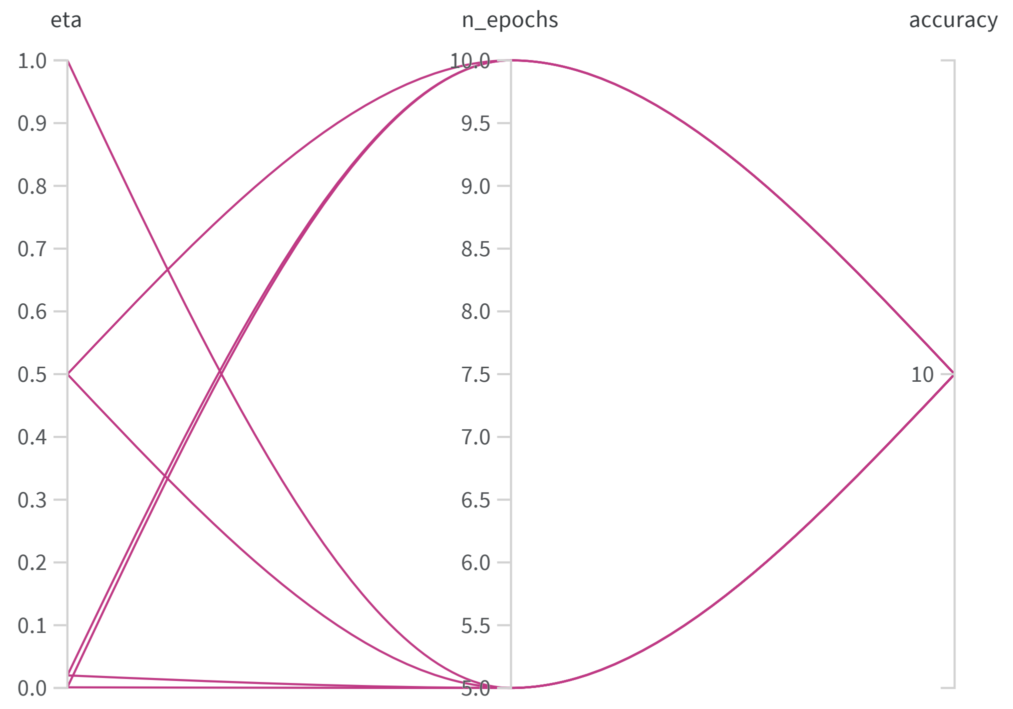
We trained the model over various sweeps and found in 'radiant-sweep-56' we found that eta = 0.5, number of epochs =10, works best for the optimizer when sigmoid activation is used. Similarly, we trained the model for other activation functions and batch sizes as well.

iv. Adam:



We trained the model over various sweeps and found in 'dauntless-sweep-6' we found that eta = 0.5, number of epochs =10, works best for the optimizer when sigmoid activation is used. Similarly, we trained the model for other activation functions and batch sizes as well.

v. Rmsprop:



We trained the model over various sweeps and found in 'spring-sweep-2' we found that eta = 0.5, number of epochs =10, works best for the optimizer when sigmoid activation is used. Although the model behaved almost same for other in]iterations as well, we trained the model for other activation functions and batch sizes as well.

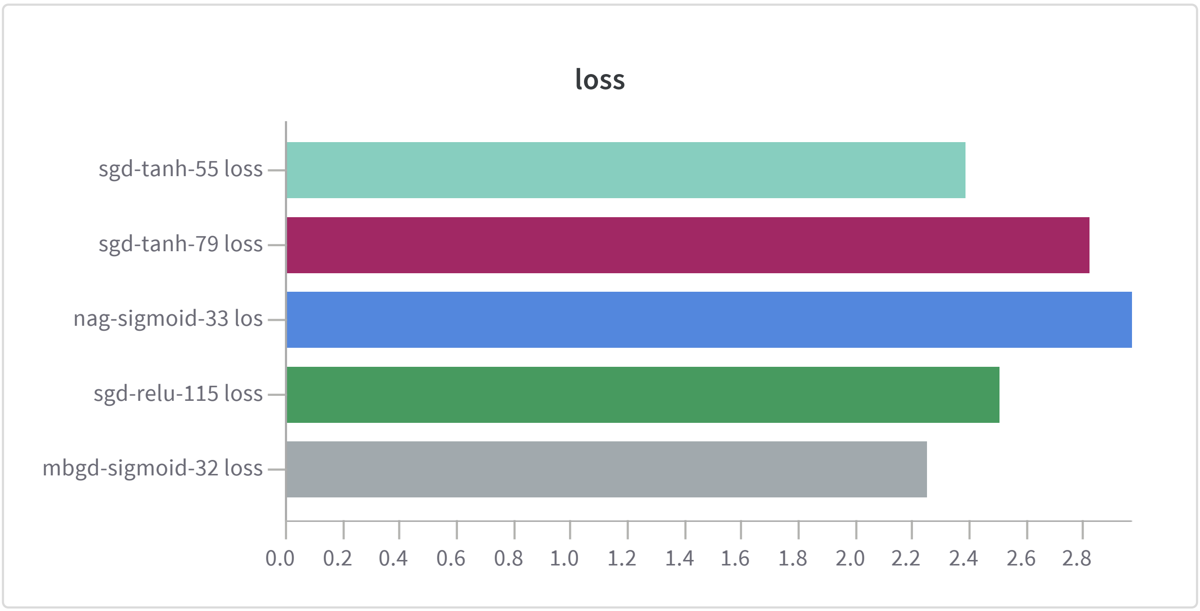
vi. Nadam:

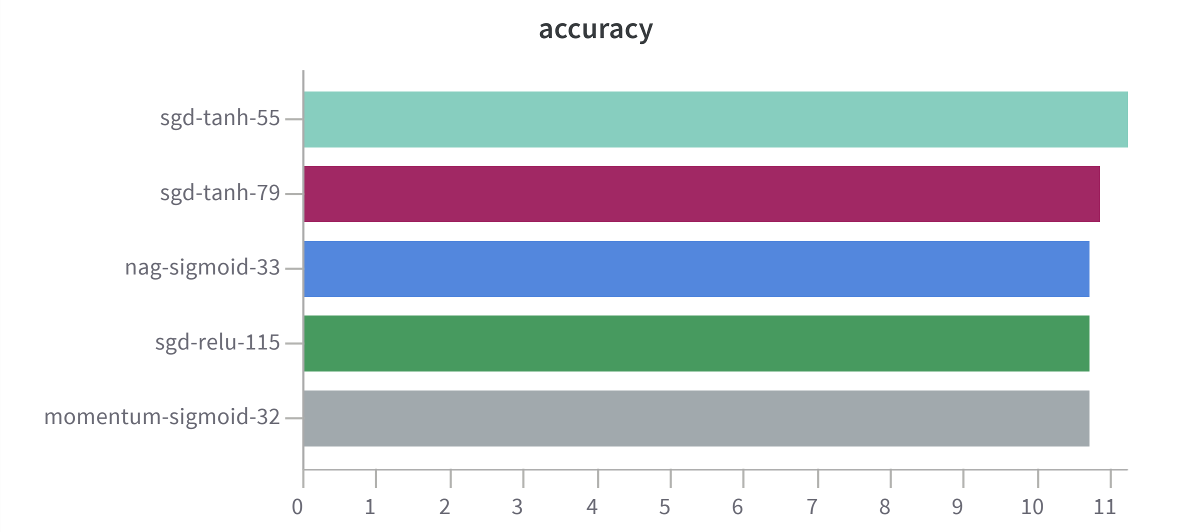
The implementation of this optimizer can be found on our Github repository. Please find it at the following path:

Problem 4

In this problem, we dive into the subject of finding the most optimal set of hyper-parameters   for performing the task of classification of images in the fasion-mnist dataset using our Neural Network.

We carried out a variety of wandb sweeps on our validation data-set for hyper-parameter search. Results of the same are summarized by the plots below. The plots indicate variation in the accuracy(percentage) and loss over different sweeps (using different permutations of hyper-parameters) taken over the validation data set.



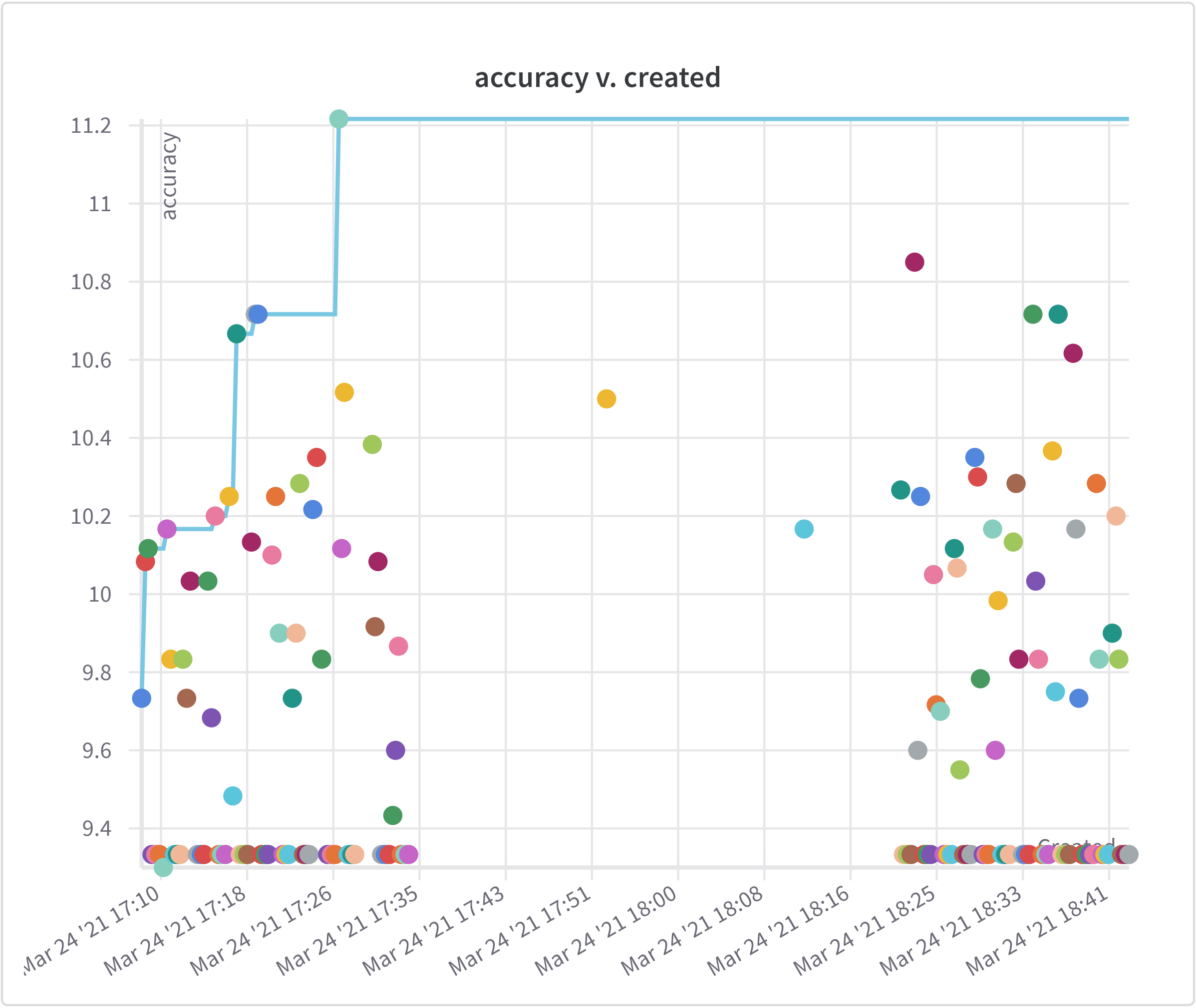


The naming convention followed for naming these sweeps is as follows:

sweep-name = {"optimizer"-"activation"-"sweepID"}

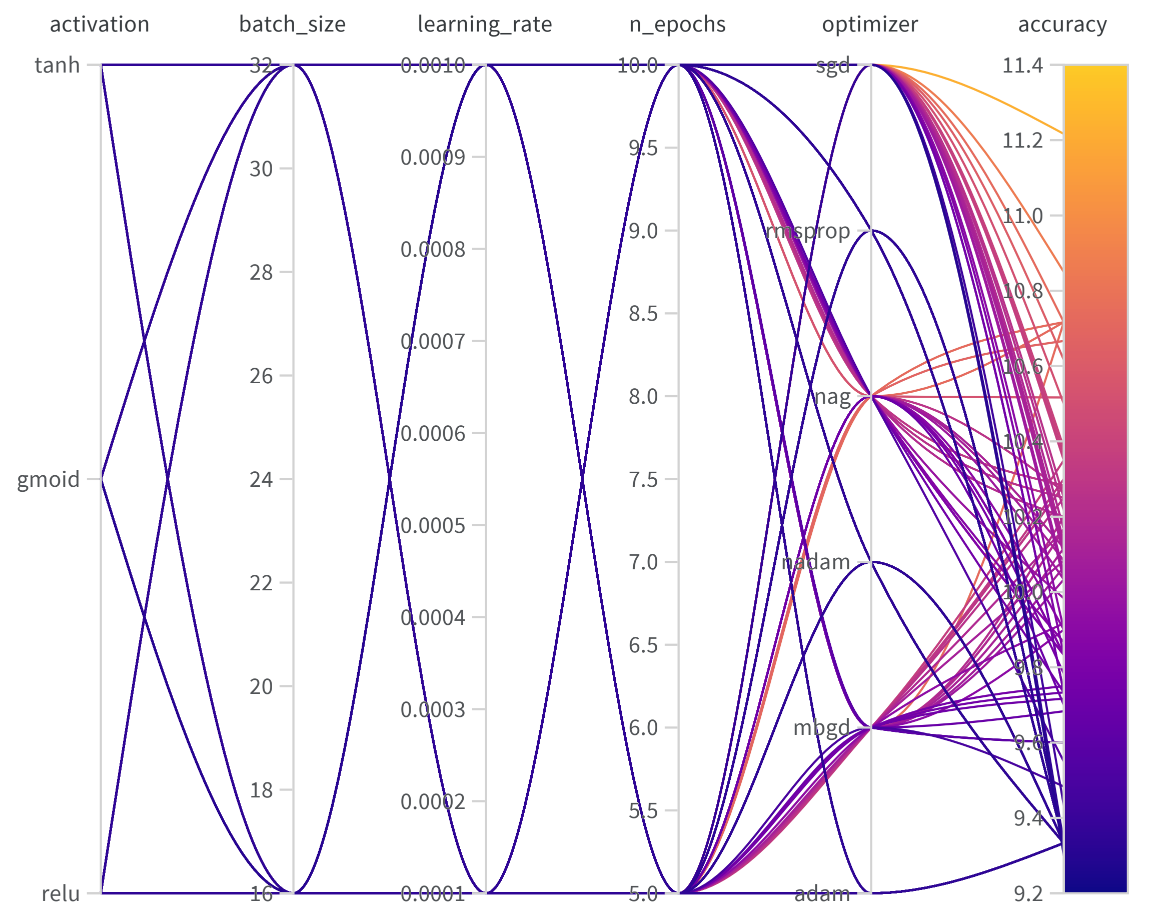
Problem 5

The accuracy versus sweep curve is appended as below:



Problem 6

The parallel axis plot and corrrelation summary for our hyper parameter search over the validation data set is as follows:



Parallel Axis Plot for finding the best set of hyper-parameters. The sweep "sgd-tanh-55" returned best accuracy. The sweep "sgd-tanh-55"entails the following hyper-parameters:

1. Optimizer: Mini Batch Gradient Descent
2. Activation: tanh
3. learning rate: 0.001
4. batch size: 16
5. Number of Epochs: 10
6. Number of Hidden Layers: 3
7. Number of Neurons in each layer: [64 32 32]
8. Initialization: Xavier

Parameter importance with respect to accuracy

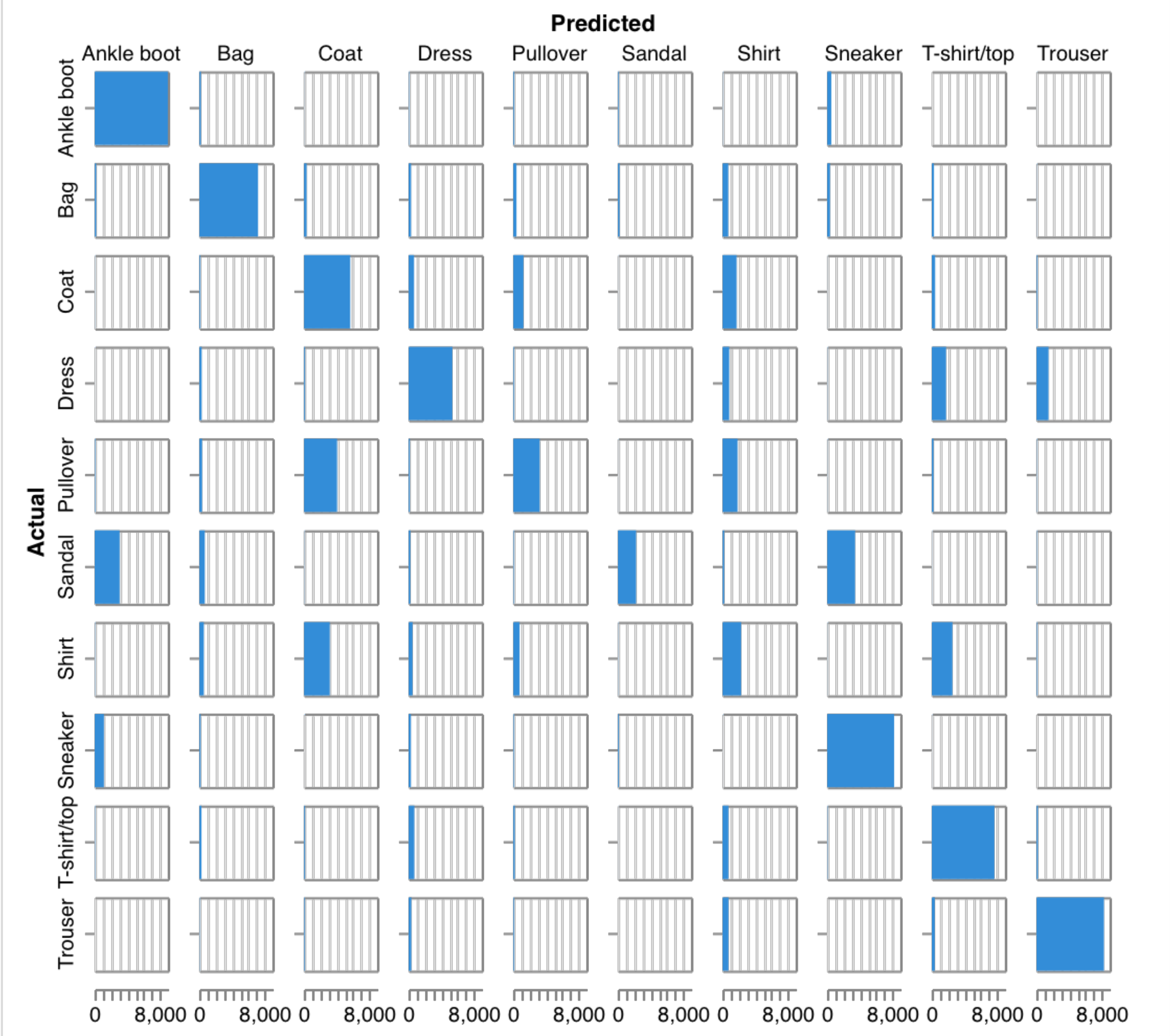
Parameters Rows per page 10 1-10 of 17

Config parameter	Importance ↓	Correlation
activation.value_tanh		
learning_rate		
n_epochs		
optimizer.value_sgd		
Runtime		
optimizer.value_mbgd		
batch_size		
activation.value_relu		
activation.value_sigm…		
optimizer.value_nag		

The above Plot indicates that learning rate is the hyper-parameter which has maximum correlation with accuracy for our model

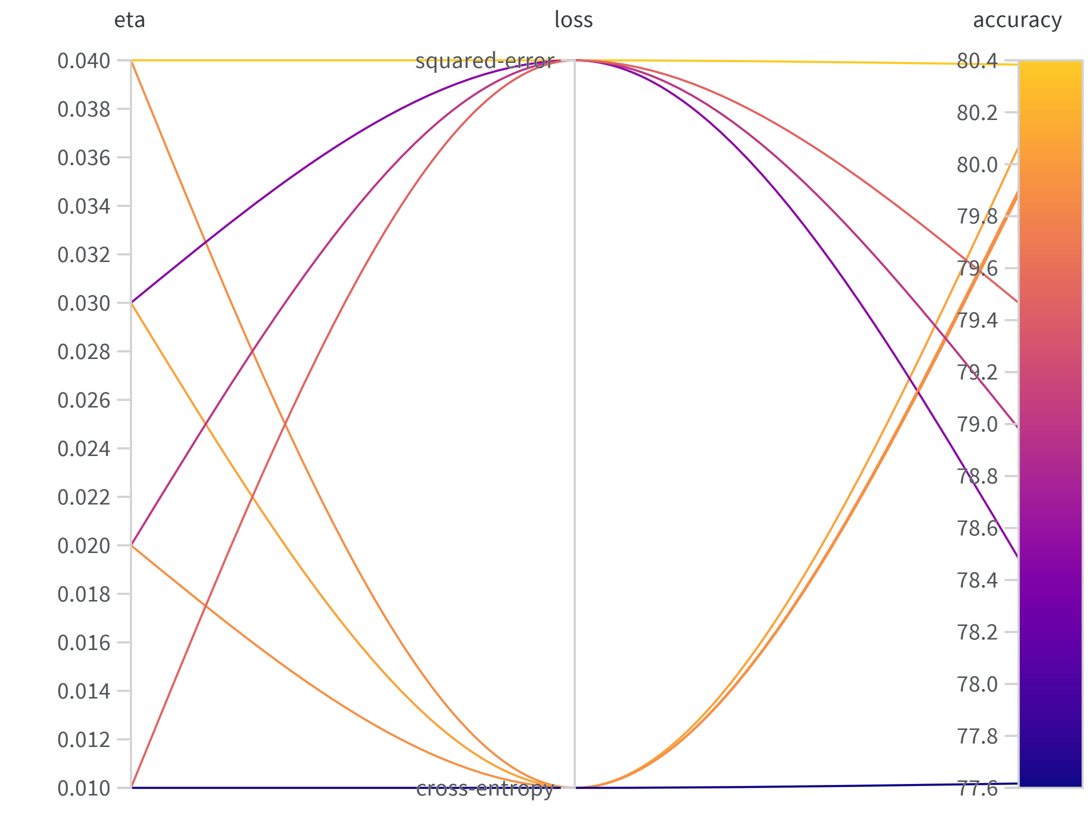
Problem 7

The confusion matrix obtained upon assessing the performance of our trained network on the test data-set is presented as below:



Problem 8

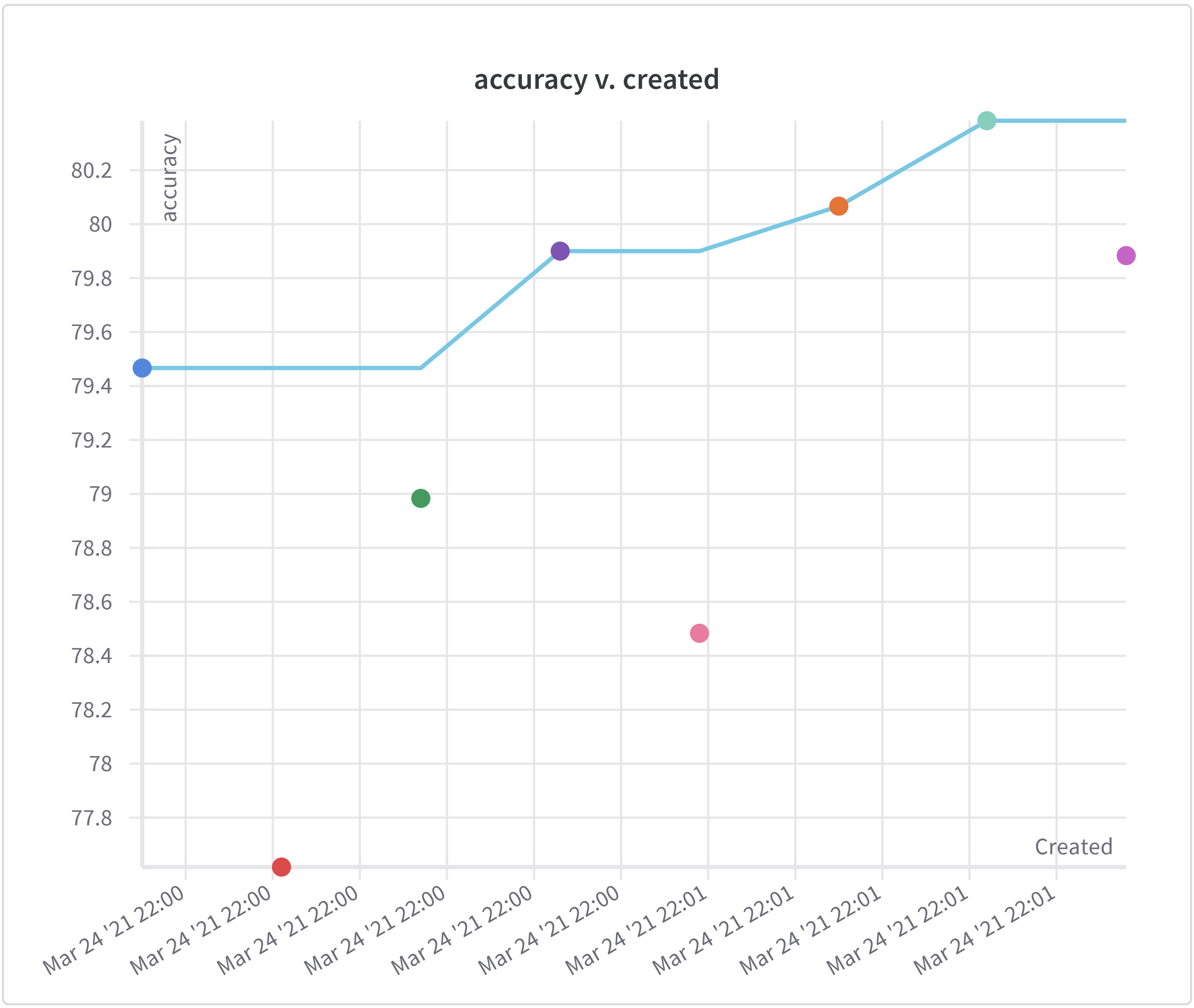
We tried both squared-error loss and cross-entropy loss during validation. Here's the Parallel Axis plot representing our findings:

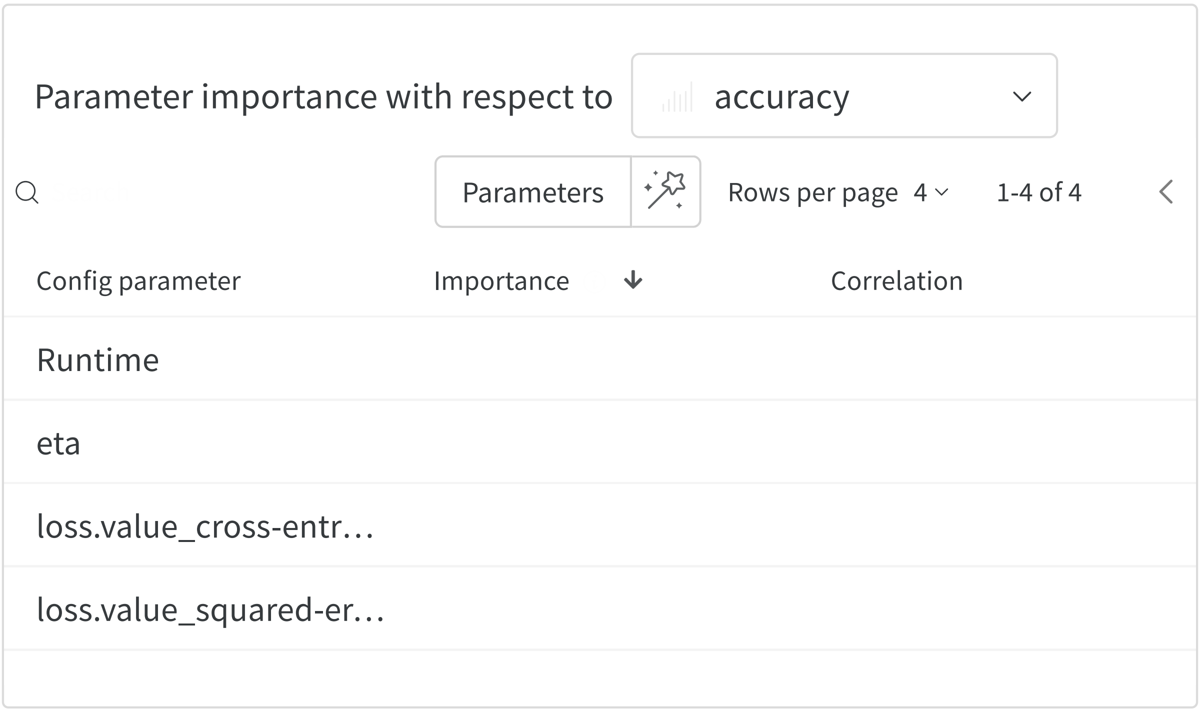


The above plot gives us several interesting insights:

1. Firstly, this set of hyper-parameters seems to work rather well for us! We have reported this at the beginning of our report
2. Theoretically, the loss metric shouldn't have much of an effect the convergence if a model like ours and although our best accuracy came with "squared-error loss" for our model; We would like to keep the opinion that cross-entropy performs better for classification problems.

The following are summary plots for this set of Hyper-Parameters:


accuracy v. created
accuracy
80.2
80
79.8
79.6
79.4
79.2
79
78.8
78.6
78.4
78.2
78
77.8
Created
Mar 24 '21 22:00
Mar 24 '21 22:00
Mar 24 '21 22:00
Mar 24 '21 22:00
Mar 24 '21 22:00
Mar 24 '21 22:00
Mar 24 '21 22:01
Mar 24 '21 22:01
Mar 24 '21 22:01
Mar 24 '21 22:01
Mar 24 '21 22:01



Config parameter	Importance	Correlation
Runtime		
eta		
loss.value_cross-entr…		
loss.value_squared-er…		

Thus, the most optimal set of Hyper-Parameters for our model are:

1. Optimizer: Mini Batch Gradient Descent
2. Activation: tanh
3. learning rate: 0.04
4. batch size: 16
5. Number of Epochs: 5
6. Number of Hidden Layers: 3
7. Number of Neurons in each layer: [64 32 32]
8. Initialization: Xavier

This gives us an accuracy of 80.383 %. We would like to recommend learning rate = 0.06 with the same set of other hyper-parameters for close to 95% accuracy, provided problems lik vanishing-gradients do not occur.

Problem 9

Here's the link to our Github Repository for this assignment:

https://github.com/snehalstomar/CS6910-Asignment-1

Problem 10

MNIST is the digit classification data consisting of handwritten digits from 0 to 9. Since in the image, only a few pixels are of impotence and rest image regions are not of importance, we need to use CNN to reduce the parameters.

Based on our experience with the assignment, we point out the important hyperparameters that play a crucial role for the performance of a model

1. Learning rate: It shows how fast a model converges and learns a very close transform function from input to output result. it should not be stuck in the local valley. Based on our experiments, we observe that learning rate plays an important role. It is difficult to tune the learning rate. While choosing a high learning rate, we can miss minima, on the other hand considering very low learning rate it takes too long to find the optimal function. For our case we found a learning rate 0.04 to be good.
2. Optimizer: Based on our experiments, we figured out that selection of optimizer plays a critical role in learning the optimal solution. Optimizer is responsible for updating the weights and biased values in the principled manner. Proper finding the appropriate weights and bias decides the accuracy of the model. For all experiments, sgd worked well producing maximum accuracy as compared to other optimizers. Based on our experiments, we found sgd to be working

for us. However it doesn't guarantee to work the same optimizer for other classification problems.

3. Activation function of hidden layers: We also observe that selection of activation functions in the hidden layers affect the accuracy of the system. For some cases, we found that sigmoid to be problematic as gradients failed to update resulting saturation in the model. For our case, tanh seems to be a better choice.
4. Loss: Based on logistic loss function and (mean) squared error, we conclude that choice of loss function also has an effect on the model.
5. Our model has accuracy of 80.383%.

we are implementing it using fully connected network, hence considering all of the pixel values i.e. for a 28 * 28 images, a total of 784 pixels are used while only certain pixels have relevant information required for the image classification. Hence we have unnecessary weights and bias to learn. This issue can be resolved using CNN followed by fully connected layers. CNN aims to learn the useful features required for image classification and reduce the size before feeding into fully connected networks.

Self Declaration

Although, it is hard to distinguish contribution of individual team-mates in a quantitative manner; We would like to present our individual contributions in the following manner:

1. Teammate 1: EE20S006 Snehal Singh Tomar(Overall Contribution 52%)

Specific Contributions:

i. Implemented the Neural Network Class, forwardprop and backprop functions (myDLkit2.py)

ii. Implemented the SGD optimizer

iii. Performed tasks related to wandb integration and preparation of report

iv. Performed Hyper Parameter Tuning.

2. Teammate 2: EE20D006 Ashish Kumar(Overall Contribution 48%)

Specific Contributions:

i. Implemented the following optimizers: mbgd, nag, rmsprop, adam, nadam using/in myDLkit2.py

ii. Performed pre-processing and splitting of data(myDataExtractor.py)

iii. Prepared the solution to Problem 10.

iv. Performed Hyper Parameter Tuning.

We, Snehal and Ashish, swear on our honour that the above declaration is correct.

https://wandb.ai/snehalstomar/cs6910-assignment-1/reports/CS6910-Assignment-1--Vmlldzo1NTAwODA