CS6910 Assignment 1

Rishanth R cs18b044

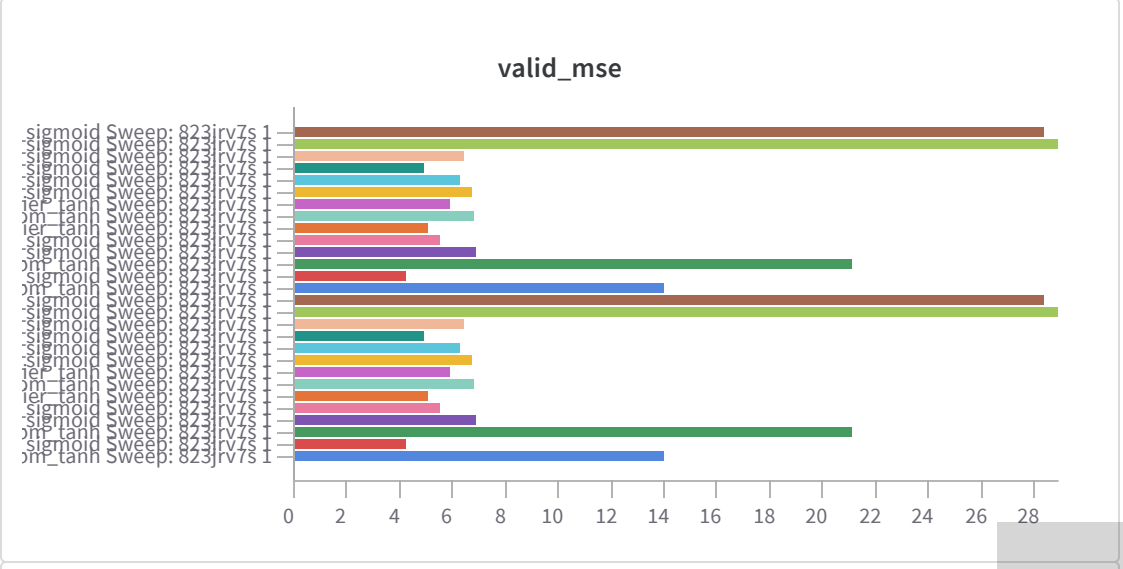
Link to this report:
https://wandb.ai/rishanthrajendhran/cs6910_assignment1/reports/CS6910-Assignment-1--Vmlldzo1Mjc1MTg?accessToken=xfnldls9dxot8klgoaekbuhcoo4w3w4d2fwszd6ocz9o05zzxd129wob2pfmu7gj

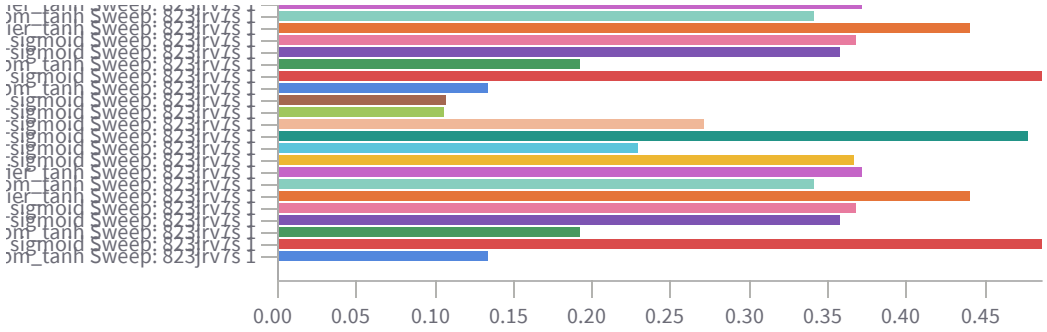
Q2. Plots attached

Q4.

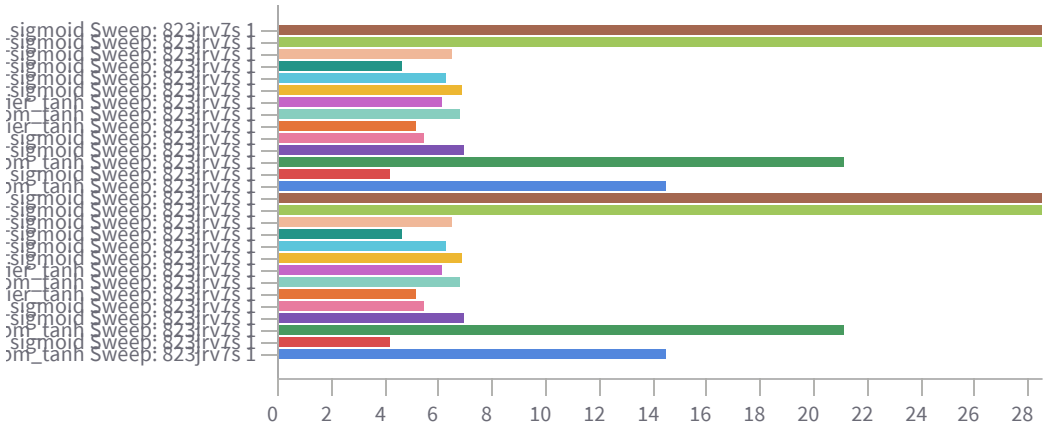
Bayes search strategy was chosen over grid search and random search so as to reduce computational expense and have a better probability of finding the best model.

Note: Entropy losses could not be logged due to an unresolved wandb error "Maximum Recursion Error"; they have been printed as output.

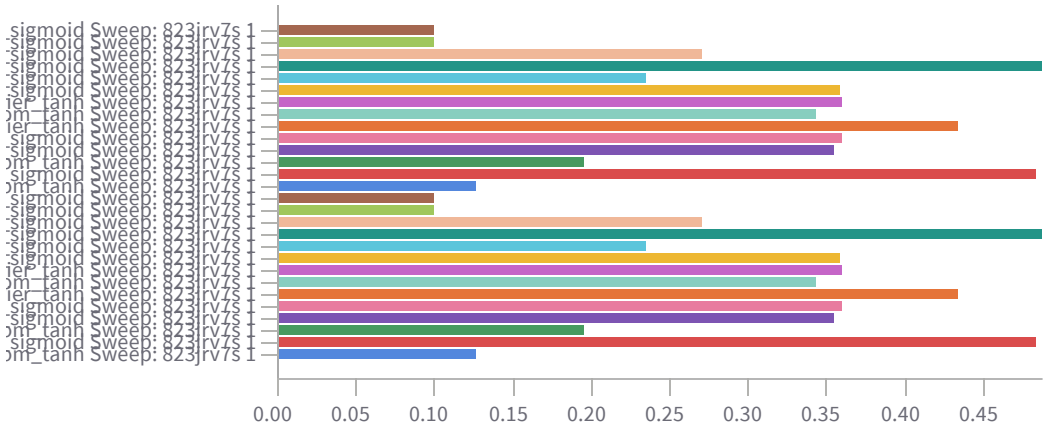




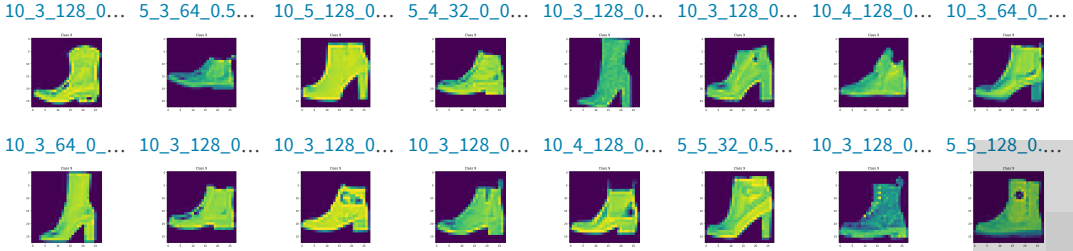
test_mse



test_acc



plot:



Q5. Plot attached

Q6.

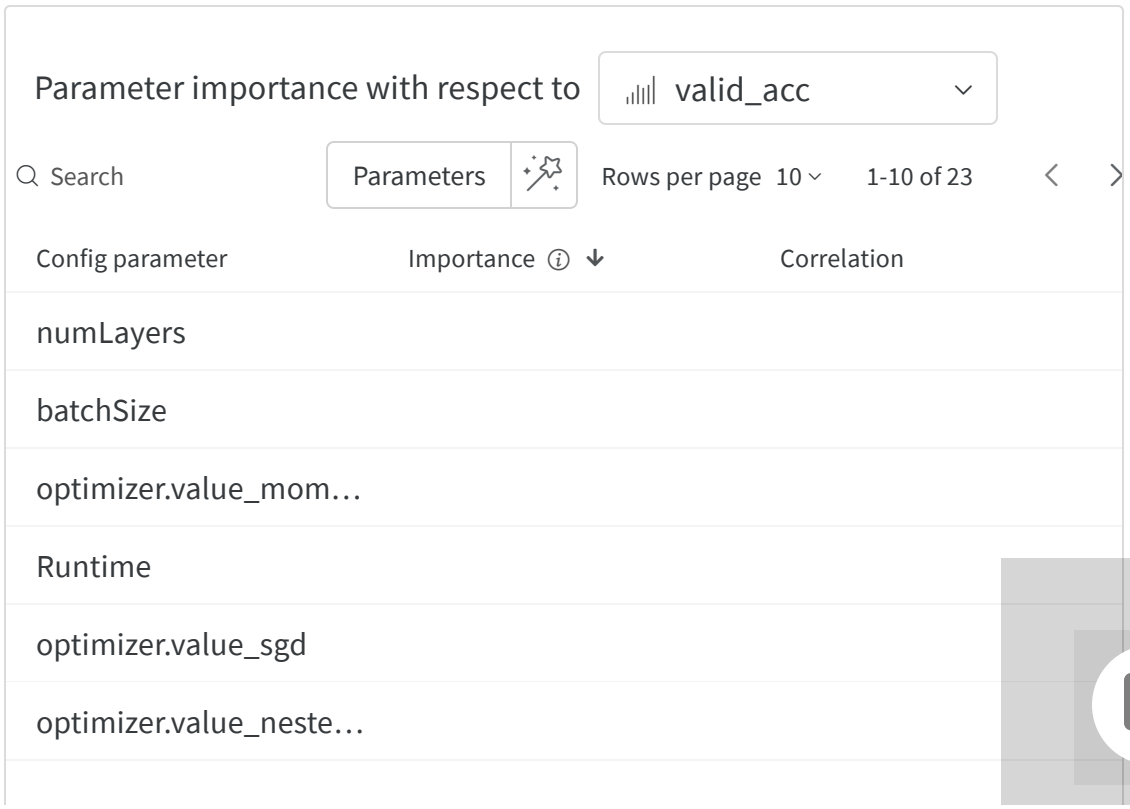
We can make the following observations about accuracy and the various hyperparamters:

(i) Lower the number of layers in the neural network, greater was the validation accuracy probably because lesser layers allow more generalisation

(ii) Greater the batchSize, greater the accuracy which also seems reasonable given that stochastic g.d. is an approximation of the actual gradients => Greater the batch size, better the approximation

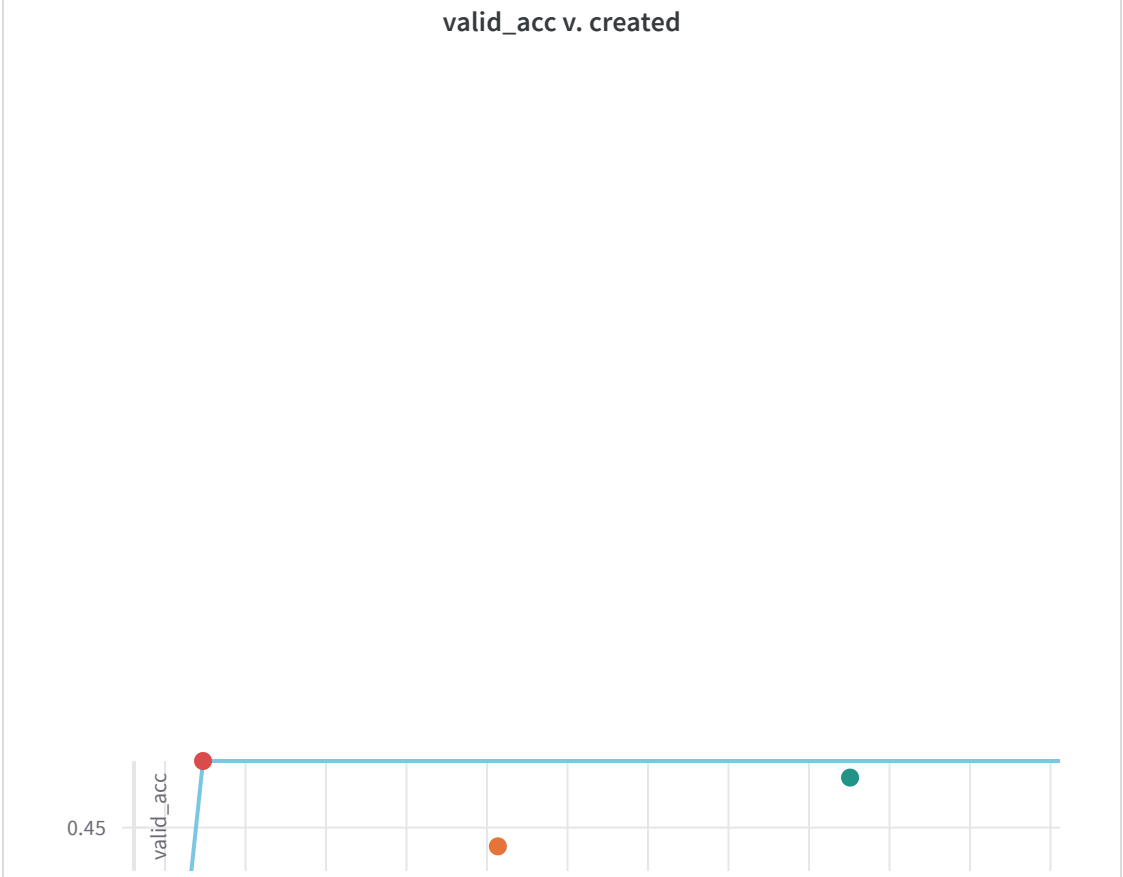
(iii) Greater the learningRate and number of epochs, greater is the accuracy which matches with intuition since more iterations would mean we are moving one step closer to the minima

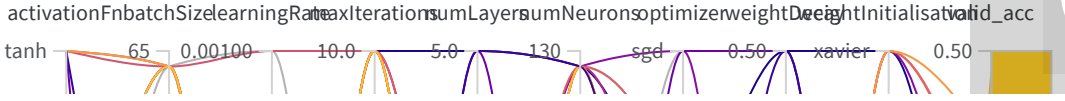
(iv) Momentum based gradient descent did better than stochastic gradient descent as expected



learningRate

numNeurons





Q7.

Best model identified:

Configuration:

ActivationFn : sigmoid

batchSize: 64

beta1: 0.9

beta2: 0.999

eps: 1.000e-8

gamma: 0.9

learningRate: 0.0001

maxIterations: 10

numClasses: 10

numLayers: 3

numNeurons: 128

optimizer: momentum

validSize: 6000

weightDecay: 0

weightInitialisation: random

Accuracies obtained:

"test_acc": 0.4836,

"test_loss": 56111.169788829146,

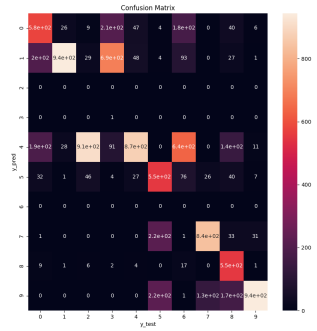
"test_mse": 4.1725,

"valid_acc": 0.4861666666666667,

"valid_loss": 33064.48118814381,

"valid_mse": 4.258666666666667

plot

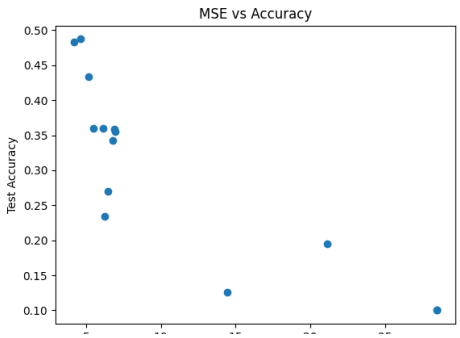


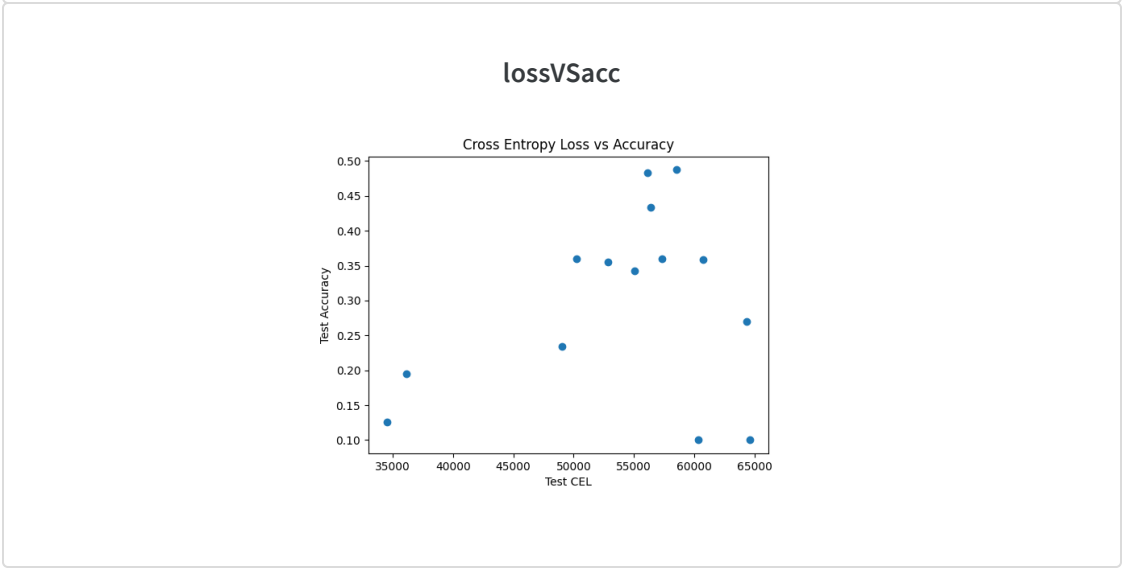
Q8.

From the plots we can see that as the MSE increases, the accuracy decreases but there is no such regular pattern observed in the case of cross entropy loss.

Even though this is a classification problem where a cross entropy loss would seem more suited, the same is not being reflected in these plots which might be because we have only done a limited number of runs in our sweep.

mseVSacc





Q9.

Github Link:
https://github.com/RishanthRajendhran/cs6910_assignment1

Q10.

We note that even the best accuracy is low which is suspected to primarily be because of poor pre processing of the input. Currently we are only doing the standard normalisation of the input data. Applying data augmentation techniques and denoising the input using auto encoders can help improve the accuracy.

Another important point to be noted is that we aren't taking advantage of the spatial arrangement of the input data since we are flattening the input data and treating it as 1D data. If we made use of the 2D structure of the input and the relative distances between the different features in the grid, we could get better accuracy. That is a much more meaningful thing to do when working with images.

Based on the sweep performed, the top 3 configurations would be:

1. Configuration:

   ActivationFn : sigmoid

batchSize: 64

beta1: 0.9

beta2: 0.999

eps: 1.000e-8

gamma: 0.9

learningRate: 0.0001

maxIterations: 10

numClasses: 10

numLayers: 3

numNeurons: 128

optimizer: momentum

validSize: 6000

weightDecay: 0

weightInitialisation: random

Accuracies obtained:

"test_acc": 0.4836,

"test_loss": 56111.169788829146,

"test_mse": 4.1725,

"valid_acc": 0.4861666666666667,

"valid_loss": 33064.48118814381,

"valid_mse": 4.258666666666667,

2. Configuration:

activationFn: sigmoid

batchSize: 64

beta1: 0.9

beta2: 0.999

eps: 1.000e-8

gamma: 0.9

learningRate: 0.0001

maxIterations: 10

numClasses: 10

numLayers: 3

numNeurons: 128

optimizer: momentum

validSize: 6000

weightDecay: 0.0005

weightInitialisation: random

Accuracies obtained:

"test_acc": 0.4874,

"test_loss": 58512.9887310171,

"test_mse": 4.65,

"valid_acc": 0.4771666666666667,

"valid_loss": 35113.18684126332,

"valid_mse": 4.9085,

3. Configuration:

activationFn: tanh

batchSize: 64

beta1: 0.9

beta2: 0.999

eps: 1.000e-8

gamma: 0.9

learningRate: 0.0001

maxIterations: 10

numClasses: 10

numLayers: 3

numNeurons: 128

optimizer: momentum

validSize: 6000

weightDecay: 0

weightInitialisation: xavier

Accuracies obtained:

"test_acc": 0.4337,

"test_loss": 56420.62842148082,

"test_mse": 5.1717,

"valid_acc": 0.43983333333333335,

"valid_loss": 33672.38020987896,

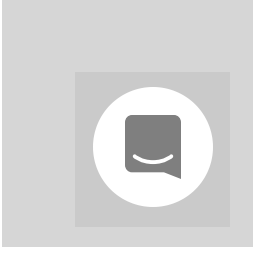
"valid_mse": 5.0615

Interestingly momentum gradient descent was the optimiser scheme used in all the three instances.

Self declaration:

Contributions by Rishanth .R (CS18B044):

Q1-4



Code for Q7,Q8

Contribution by Rahul Chaurasia (CS20M002):

Q5-8 - Analysis

Joint Contribution:

Q9 and Q10

Created with ❤️ on Weights & Biases.

https://wandb.ai/rishanthrajendhran/cs6910_assignment1/reports/CS6910-Assignment-1--Vmlldzo1Mjc1MTg