Edit Share

CS6910 Assignment - 1

cs22m062

Name : Pankaj Singh Rawat

Roll No. : CS22M062

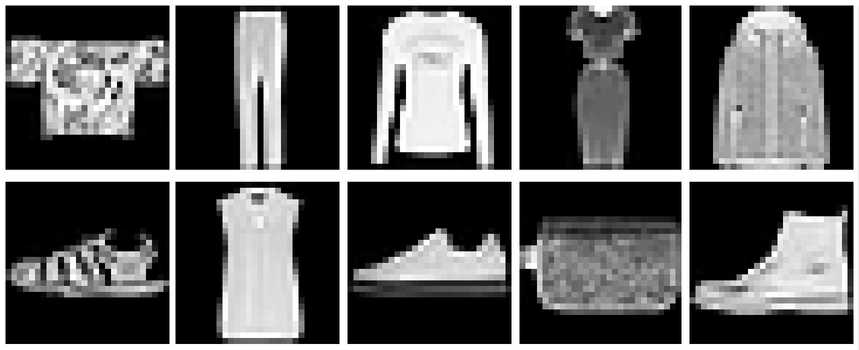
WandB Report Link: https://api.wandb.ai/links/iitmadras/3pav9adf

GitHub Repository Link : https://github.com/pankajrawat9075/CS6910_Assignment_1

Question 1

Printed all the example image for each label.

Dataset visualised



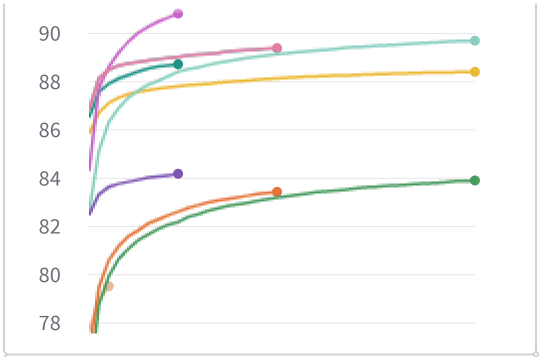
Selected Runs Without Media (10)

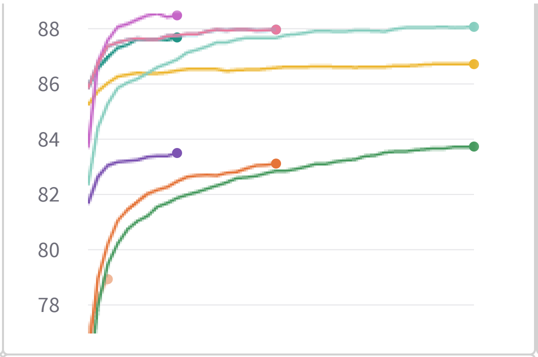
Question 4 :

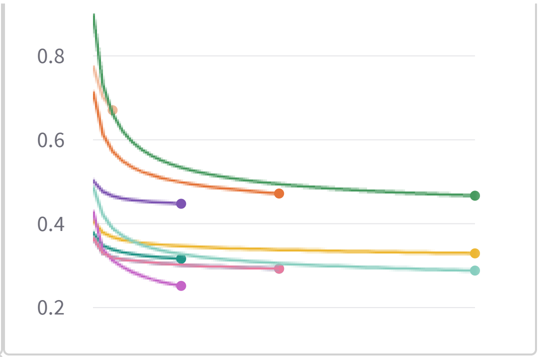
- For sweeps I **didn't go with Grid** option since it would **try out every possible combination of Hyper Parameter** and that would **take a lot of time**.
- Instead I tried **one complete sweeps with Random** setting which allowed me to check random combination of Hyperparameters and **helped me in understanding the overall dependency of my validation error on my Hyperparameters.**
- At last I ran **one sweeps with the Bayes** setting and that **saved a lot of time.** Within few 50 sweeps it was able to get the best validation accuracy sweeps. It uses the **likelihood of my validation error given my hyperparameters** and optimizes based on the results the combination of hyperparameters give.

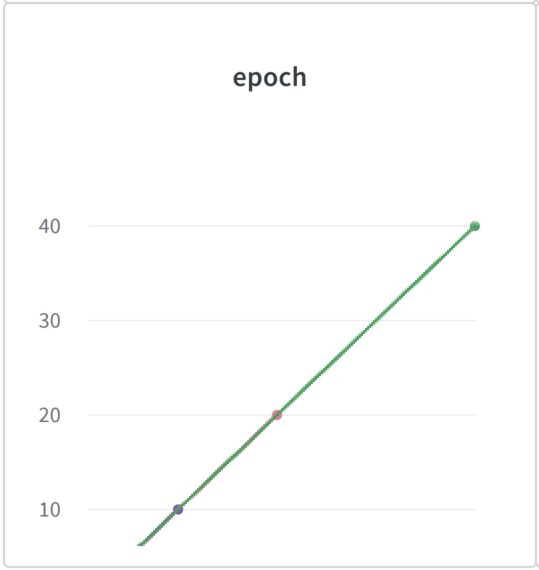
I ran many sweeps but this was the sweep where I incorporated all the techniques learned in class and therefore only displaying results for this sweep

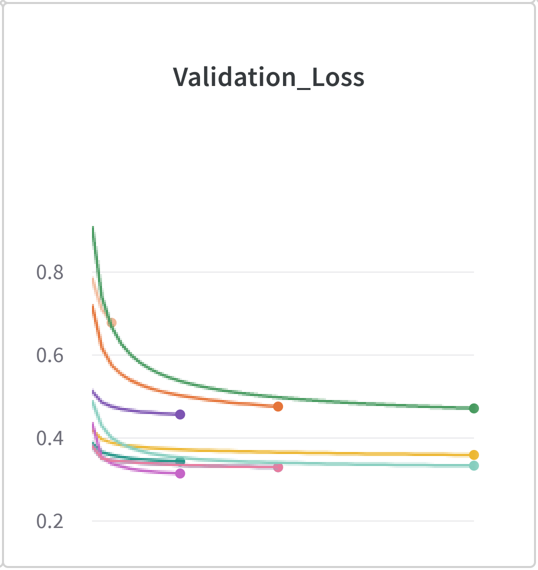
Training_accuracy	val_accuracy	Training_Loss





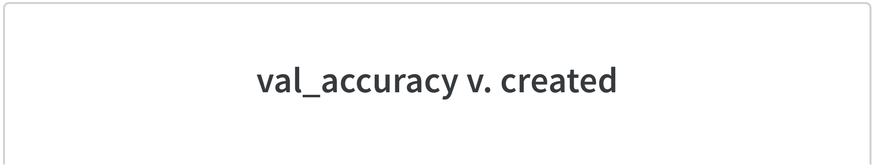


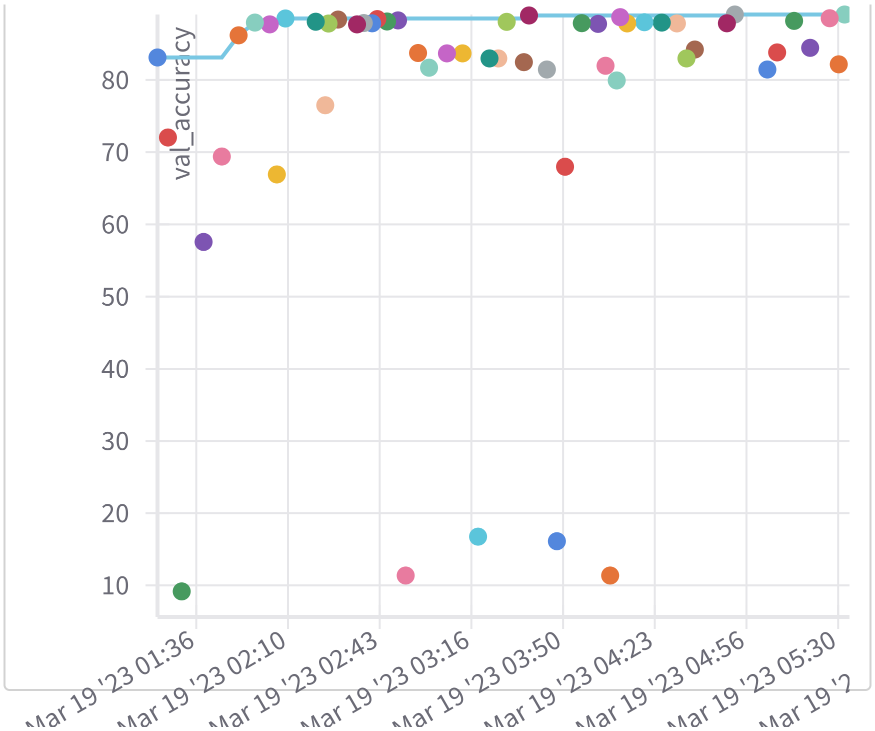




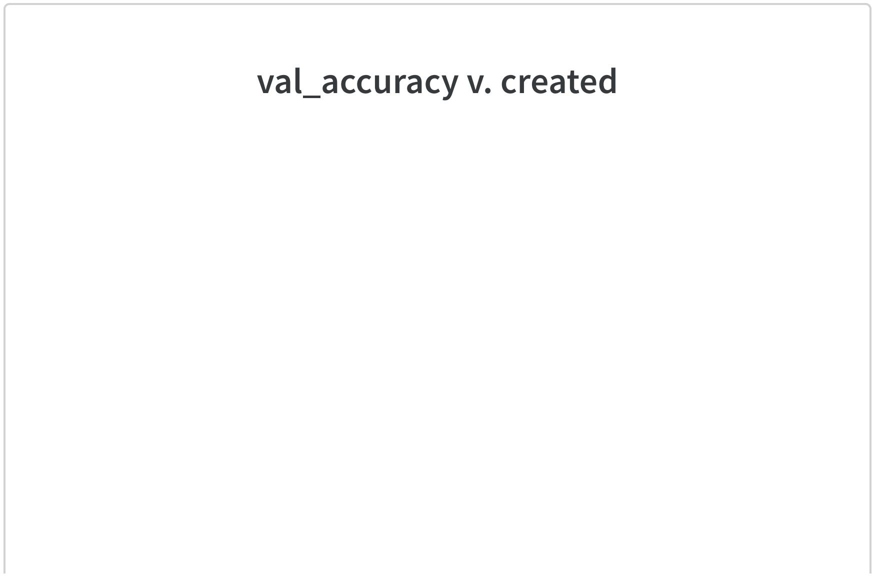
Question 5

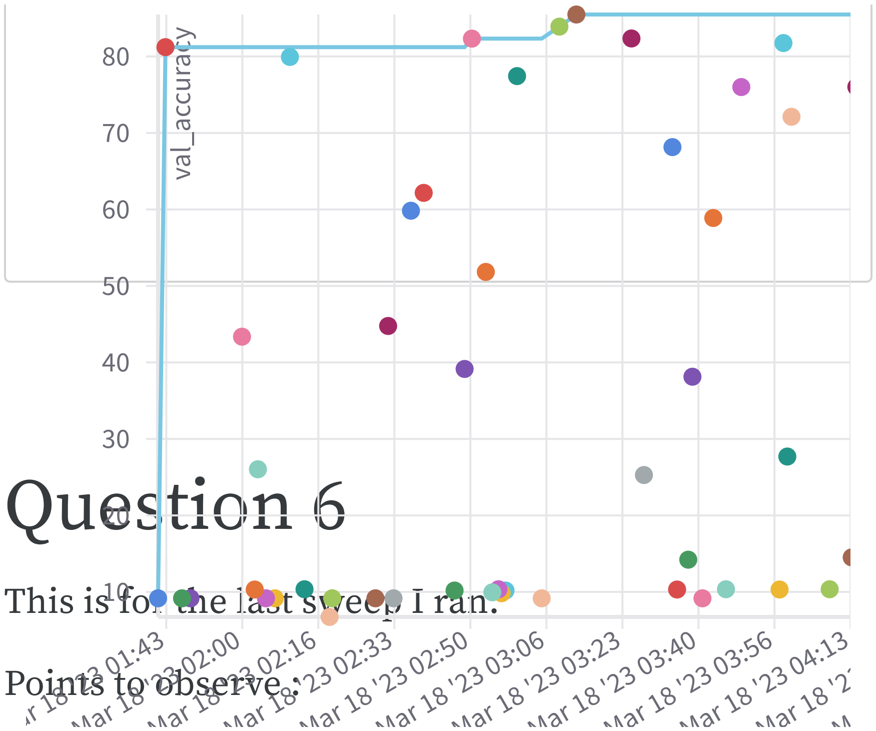
This plot is for only the last seeps I ran and I also used bayes method for the sweeps. Therefore we see very less models with less accuracy.





This is another plot from another sweep, this helped me understand why some models give less than 65 % accuracy.



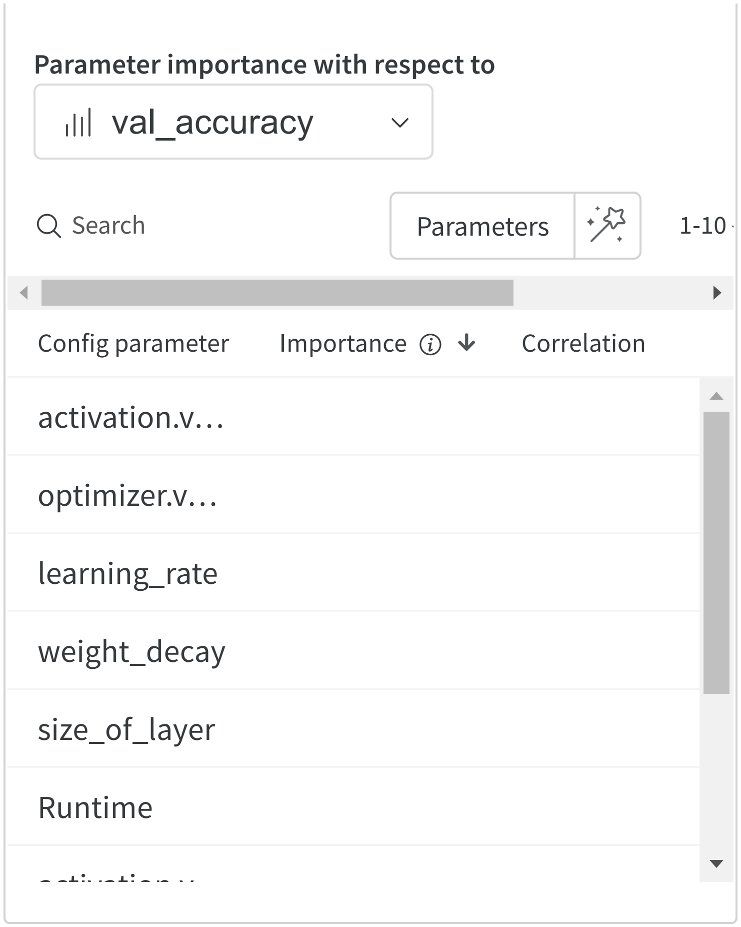


Question 6

This is for the last sweep I ran.

Points to observe :

- **Activation function** plays most important role for improving models accuracy. Many of the times the weights values have become nan and therefore the learning is stopped. Some activation function are able to handle the 'nan' situation while some fail
- In this sweeps we can see that the **learning** rate is given less importance. This was not the case generally. When I ran the earlier sweeps, I saw great importance for learning rate as we will se further.
- **Optimizer** also is crucial for models accuracy. This is mainly because the optimizer decides the direction and magnitude to move in the error direction. While some optimizer do that well and some fail.



Parameter importance with respect to

val_accuracy

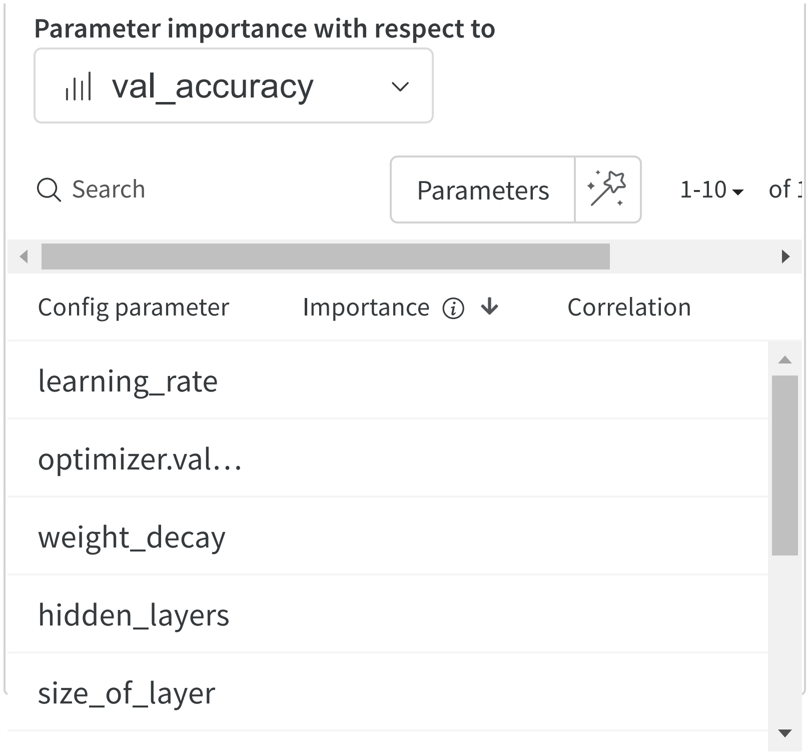
Search | Parameters | 1-10

Config parameter	Importance	Correlation
activation.v...		
optimizer.v...		
learning_rate		
weight_decay		
size_of_layer		
Runtime		

This is for the earlier sweeps I ran.

Points to observe

- **Learning rate** do play important role as we can see for these sweeps.
- Again **optimizer** also play important role and it is evident.


Parameter importance with respect to

val_accuracy

Search

Parameters

1-10 of

Config parameter	Importance	Correlation
learning_rate		
optimizer.val…		
weight_decay		
hidden_layers		
size_of_layer		

Key observations and inferences

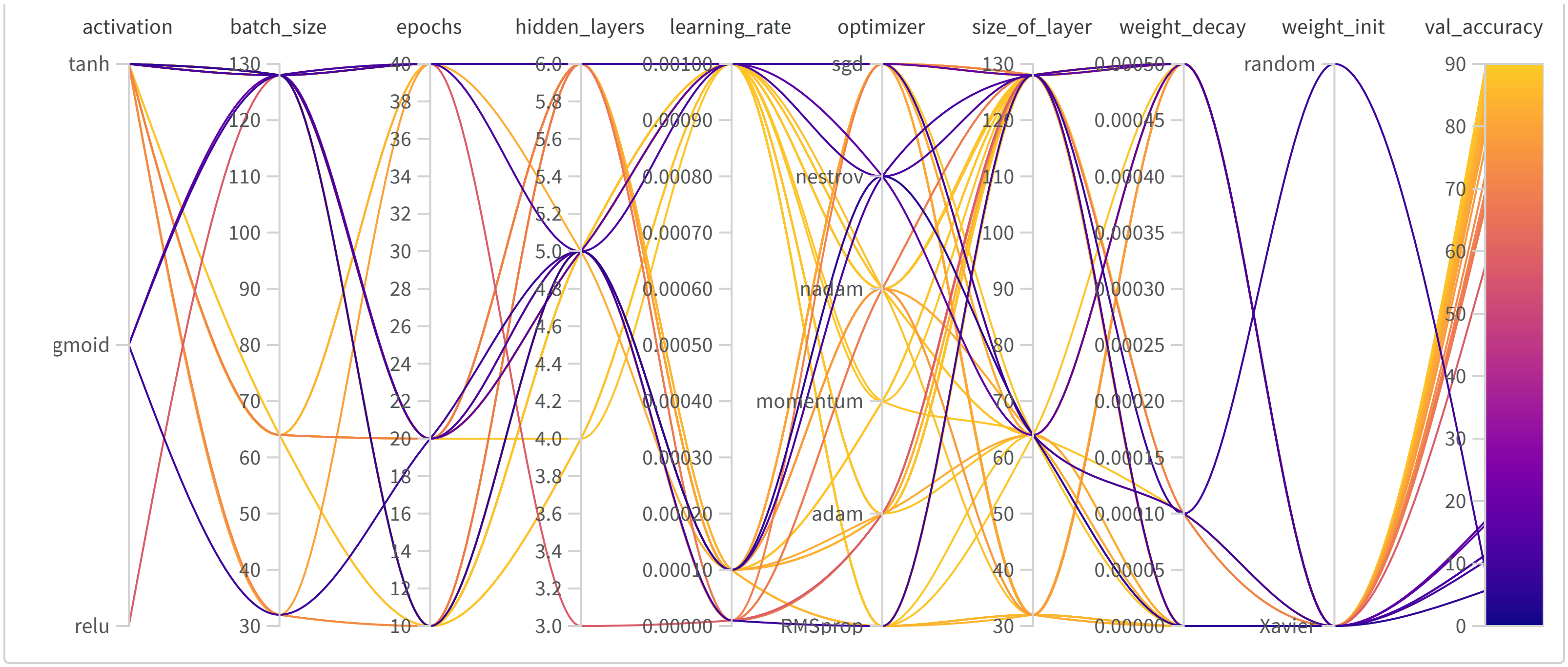
1. We notice that the **Nesterov** accelerated gradient optimizer does not perform well. In the case of Nesterov, the maximum validation accuracy obtained is 16.117%.
2. **Sigmoid** activation function doesn't give good accuracy for any model. It fails to give over 20% accuracy in any case. It is evident that sigmoid suffers from vanishing gradients problem.
3. **tanh** activation function is working very well and we see most of the above 85% accuracy. In earlier sweeps **relu** function was also giving accuracy over 85%.
4. In very few rare cases **random** weight initialization works well and generally in produces less accuracy. **Xavier** method is working well in many cases and produces over 88% accuracies also.

5. **Nadam** and **adam** optimizers take less epochs about 20 to get the same accuracy. SGD takes more epochs to get the same accuracy.
6. **Momentum** works for only 1 case where it give 88% accuracy otherwise it performs poorly in other cases.
7. weight decay of 0.0005 seems to be the best for adam, nadam and RMSprop and gives accuracies over 85%. This is because it restricts the weights to be large allowing the model to generalize better for the unseen validation data.
8. For the best accuracies, learning rate of 0.001 seems to be the best. Most high accuracy model use this rate.

Recommendations for 95%

1. Initialization recommend be Xavier, tanh function as optimizer(relu also performs well) and 64 or 128 neurons in each hidden layer. For each hidden layer we may have different neurons.
2. The optimizer should be RMSProp, Adam, or Nadam. It is important to have good learning rate and batch size that is suitable for the optimizer.
3. To avoid overfitting we should use a weight decay of 0.001 or 0.005.
4. Epochs should be taken care of and methods like early stopping to not allow model to overfit on training data.
5. Using techniques like dropout, adding noise, data augmentation to help the model generalize better on unseen test data.

Parallel Coordinates Plot for Hyperparameter search (using cross-entropy loss):



Question 7

The Best models hyperparameters are

learning_rate = 0.001

size_of_layer = 128

epochs = 40

activation_func = 'tanh'

optimizer = 'RMSprop'

batch_size = 128

weight_init = 'Xavier'

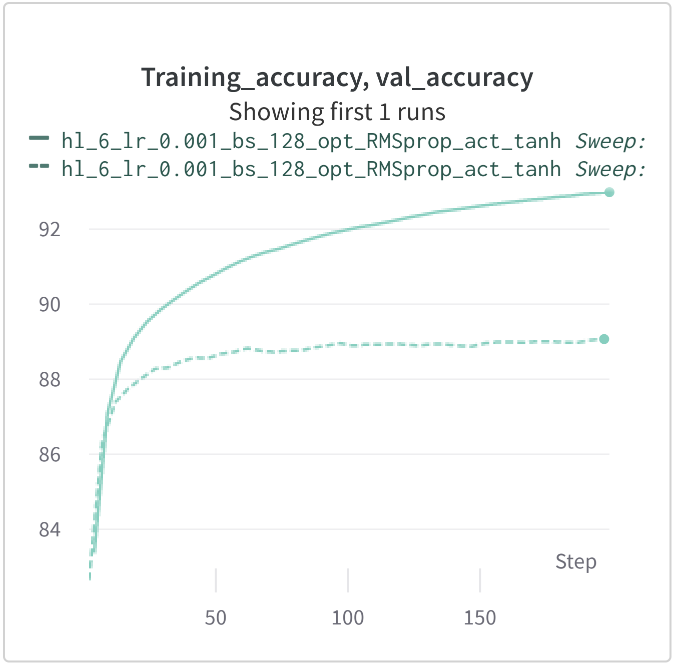
weight_decay = 0

hidden_layers = 6

Training accuracy = 0.8906666666666667

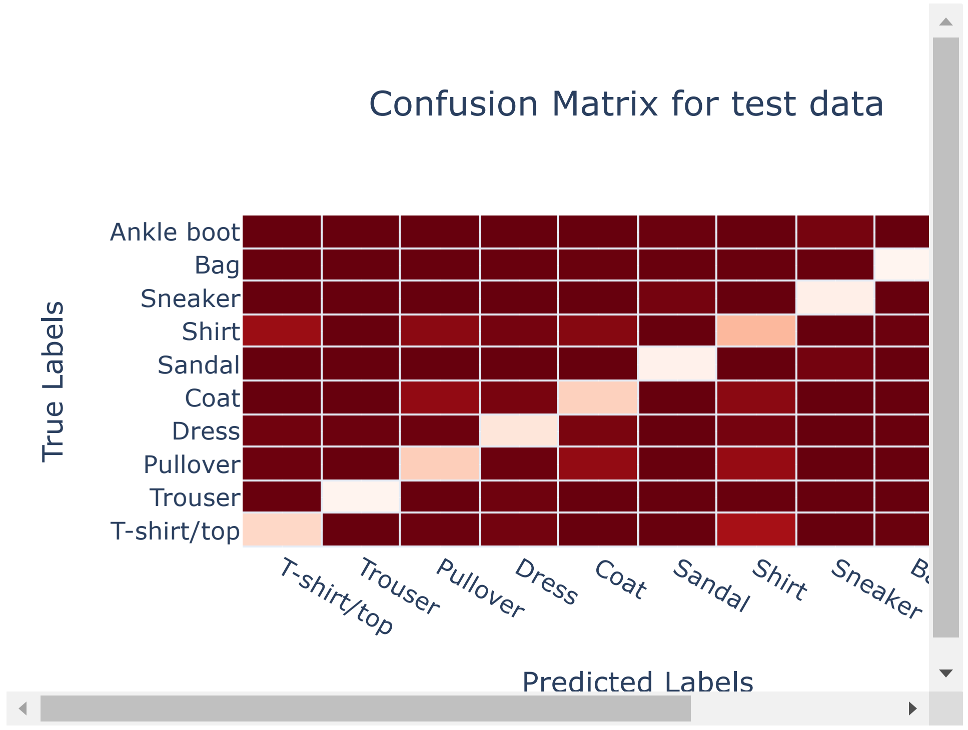
Test accuracy = 0.8856

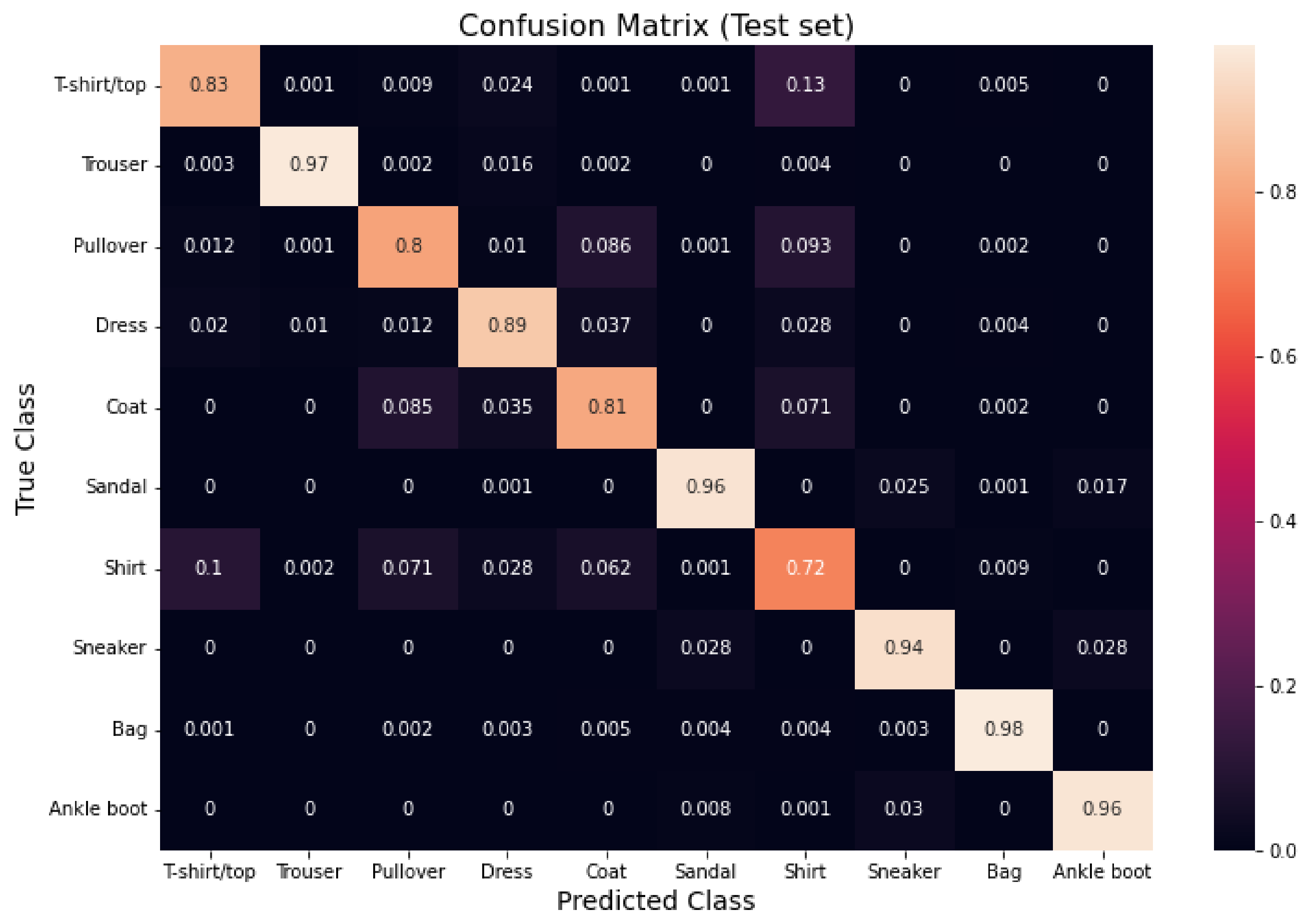
Plots for the best model



Confusion matrix

confusion_matrix_hover

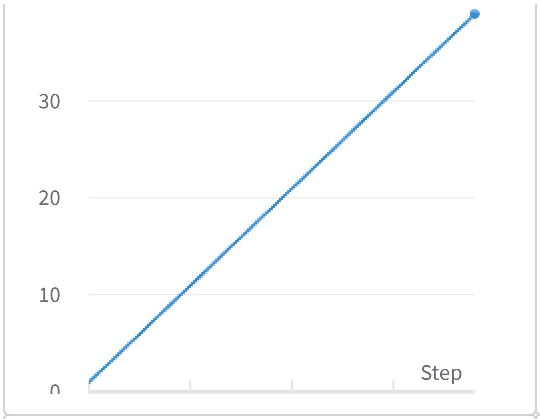


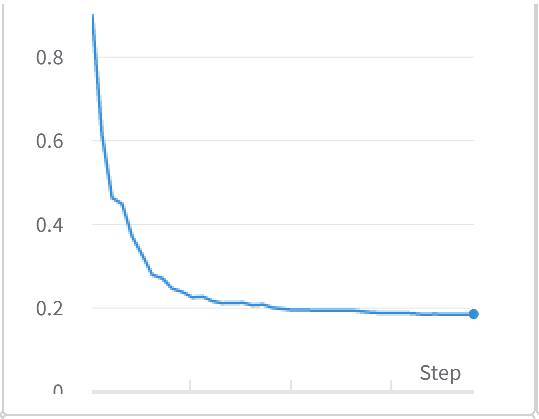


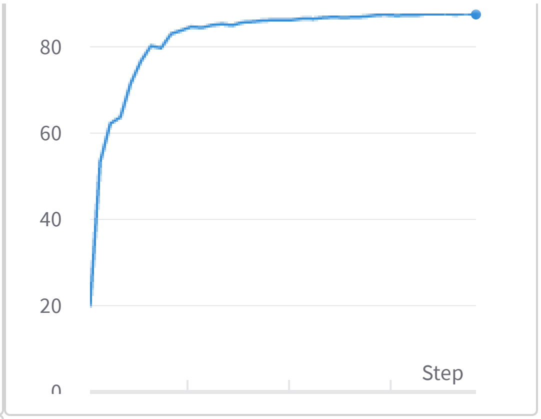
Question 8

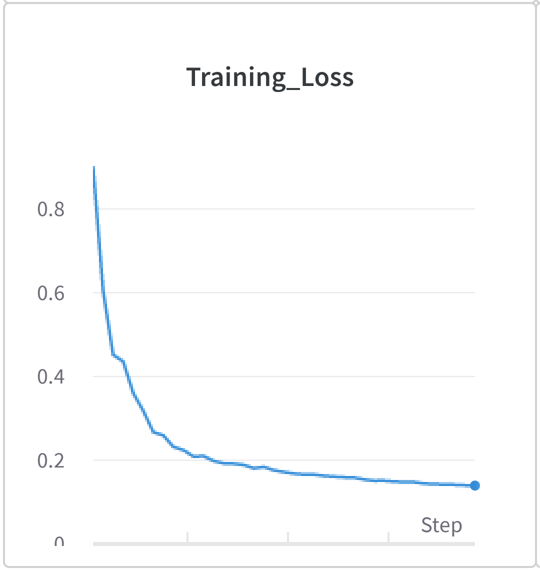
The following Plots are for MSE loss with the best Hyperparameters.

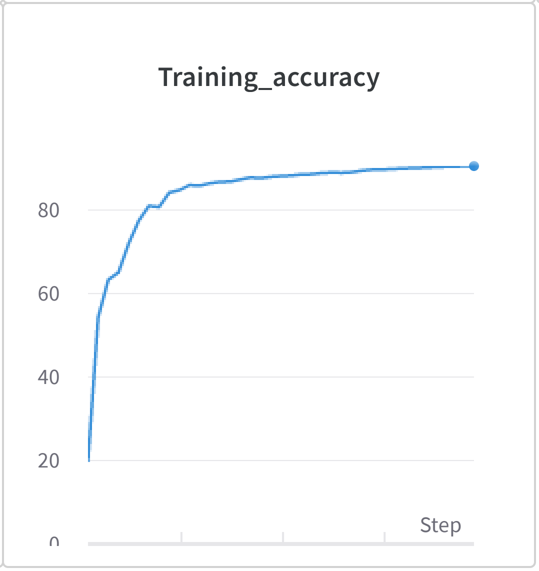
epoch	Validation_Loss	val_accuracy



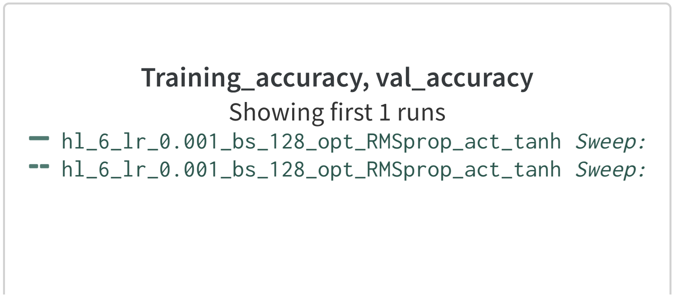


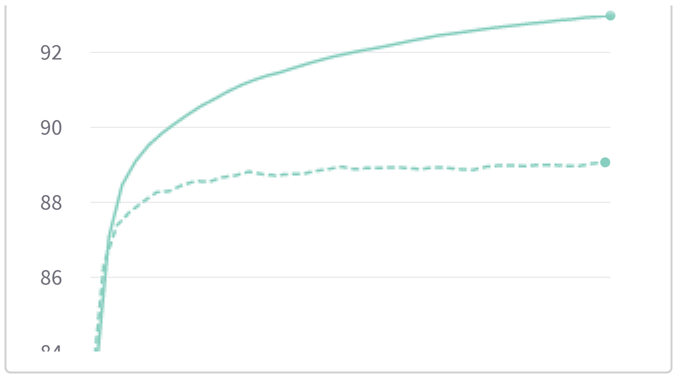






Plot for the cross entropy loss with best hyperparameters





As we can see that the validation accuracy with square loss is less. This is in general true as we I tried to run this for many configurations of hyperparameters but it always shows less accuracy with respect to the cross entropy loss.

Some reasons why I think cross entropy loss seems to be better than squared loss:

- In classification tasks, it is typical to encounter imbalanced datasets, where some classes are represented by significantly fewer samples than others. This imbalance can affect the performance of different loss functions. For instance, the mean squared error (MSE) loss can be overwhelmed by the more prevalent classes, whereas cross-entropy loss can better manage class imbalance by assigning more weight to underrepresented classes.
- Using MSE loss in a classification task is not appropriate because it does not take into account the fact that the outputs are probabilities that should sum up to 1. It also penalizes large errors more heavily than small errors, which can lead to unstable training.

Question 9

GitHub Repository Link : https://github.com/pankajrawat9075/CS6910_Assignment_1

Question 10

Since MNIST is a much simpler dataset, and a very similar image classification task with the same number of classes, the configurations of hyperparameters that worked well for Fashion-MNIST is expected to work well for MNIST too.

Hyperparameter config 1:

learning_rate = 0.001

size_of_layer = 128

epochs = 40

activation_func = 'tanh'

optimizer = 'RMSprop'

batch_size = 128

weight_init = 'Xavier'

weight_decay = 0

hidden_layers = 6

Training accuracy = 0.9992592592592593

Test accuracy = 0.9762

Hyperparameter config 2:

learning_rate = 0.001

size_of_layer = 128

epochs = 20

activation_func = 'tanh'

optimizer = 'nadam'

batch_size = 128

weight_init = 'Xavier'

weight_decay = 0.005

hidden_layers = 6

Training accuracy = 0.9938

Test accuracy = 0.9799

Hyperparameter config 3:

learning_rate = 0.001

size_of_layer = 64

epochs = 10

activation_func = 'tanh'

optimizer = 'adam'

batch_size = 128

weight_init = 'Xavier'

weight_decay = 0.005

hidden_layers = 5

Training accuracy = 0.9824

Test accuracy = 0.9681

Self Declaration

I, Name Pankaj Singh Rawat, swear on my honour that I have written the code and the report by myself and have not copied it from the internet or other students.