DA24M021-DA6401-Assignment1

This report is submitted by Siddhant Baranwal (DA24M021) in partial fulfillment of the assignment 1 of DA6401 course.

Siddhant Baranwal da24m021

Created on March 5 | Last edited on March 8

Question 1

- Downloaded the fashion-MNIST dataset and plotted 1 sample image for each class as shown below.

Question 2

- Implemented the feed forward neural network in the code files which is robust enough to handle different number of hidden layers and number of neurons in each layers.

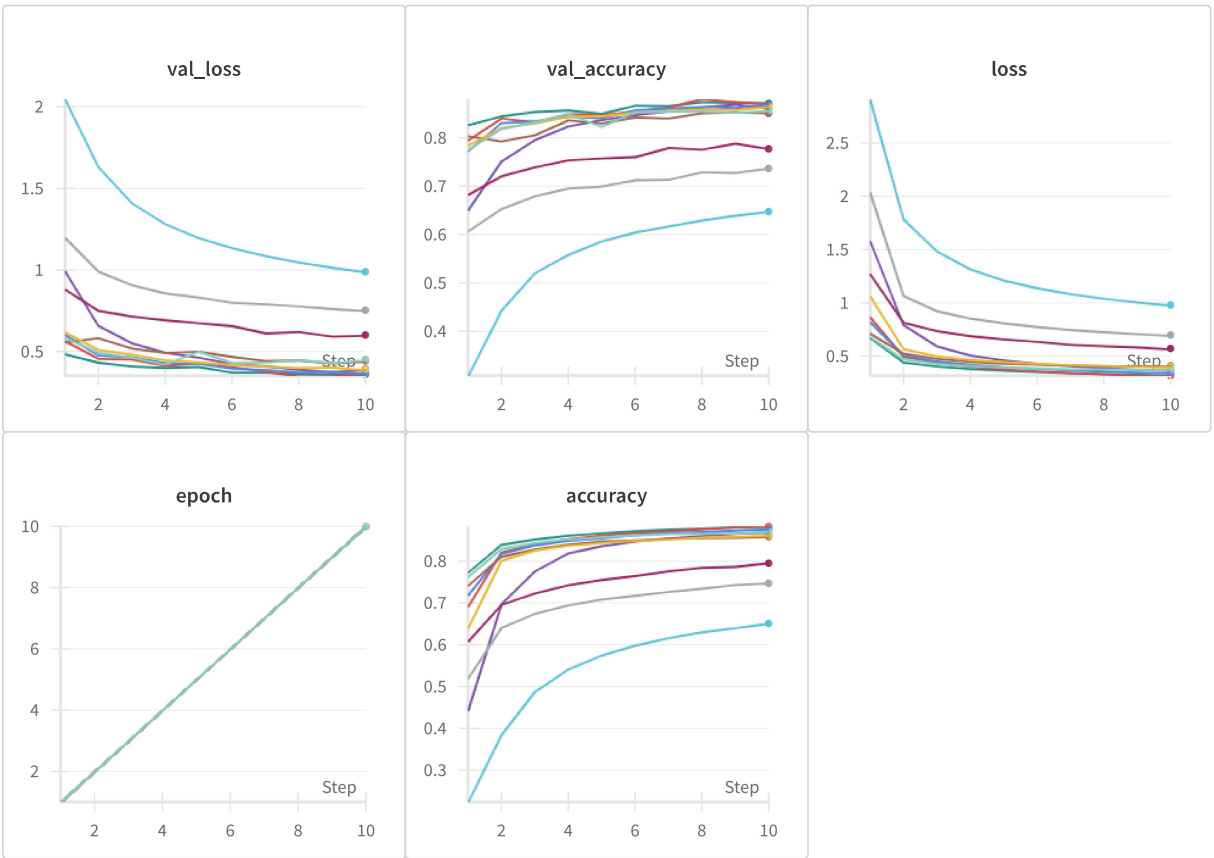
Question 3

- Implemented the backpropagation algorithm in the neural network model with support for each of the asked optimization functions. The code I believe is very modular and can easily handle addition of new activation functions, loss functions, new layer types, weight-initialization techniques and optimization functions.

Question 4

- Used the sweep functionality for hyperparameters tuning. Did the train-validation splits. Code available :- "hyperparameter_tuning.py"

  NOTE - The curves I believe are a little bumpy because I used a mini-batch instead of full-batch.

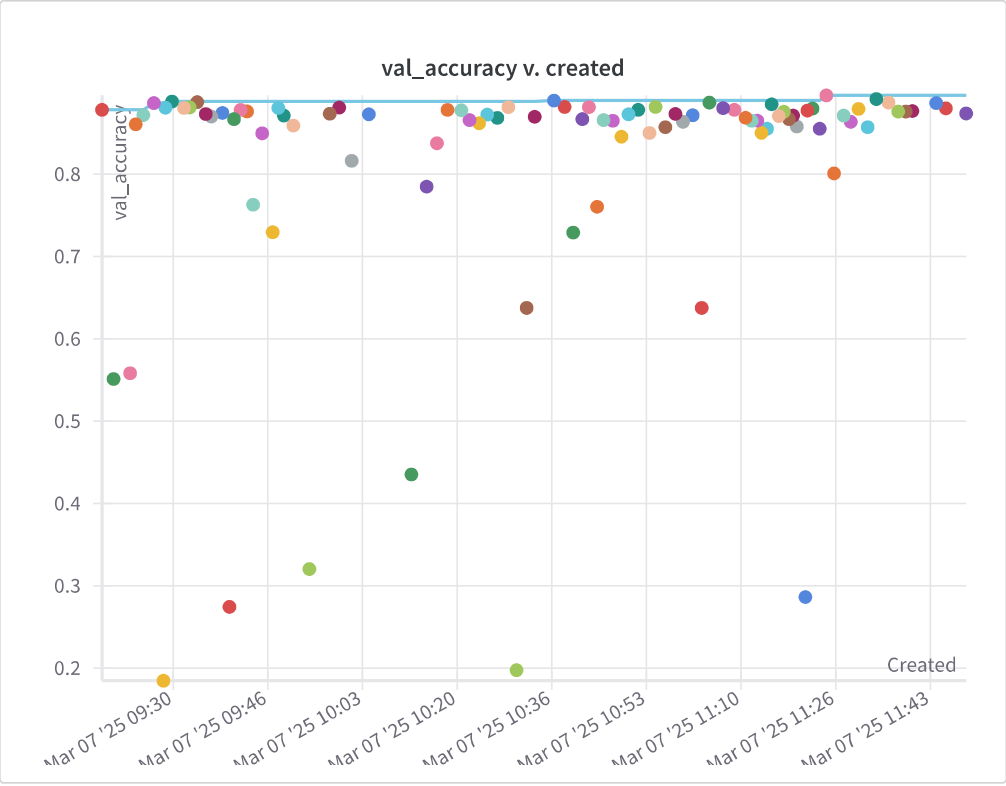


Question 5

Added the val_accuracy plot as asked below.

I used different sweep techniques. I felt Bayesian performed best, the below plot is for "bayes" sweeping.

Best validation accuracy achieved:- 0.8893



Question 6

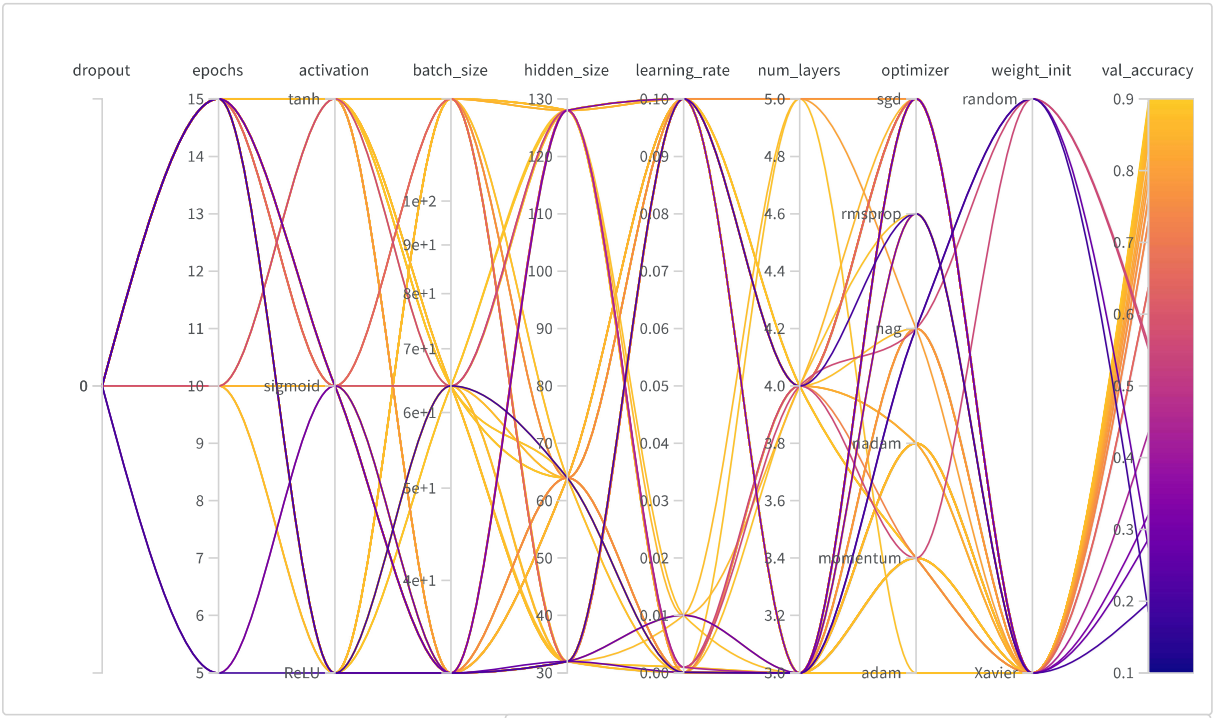
I am getting lower accuracy (< 65%) for

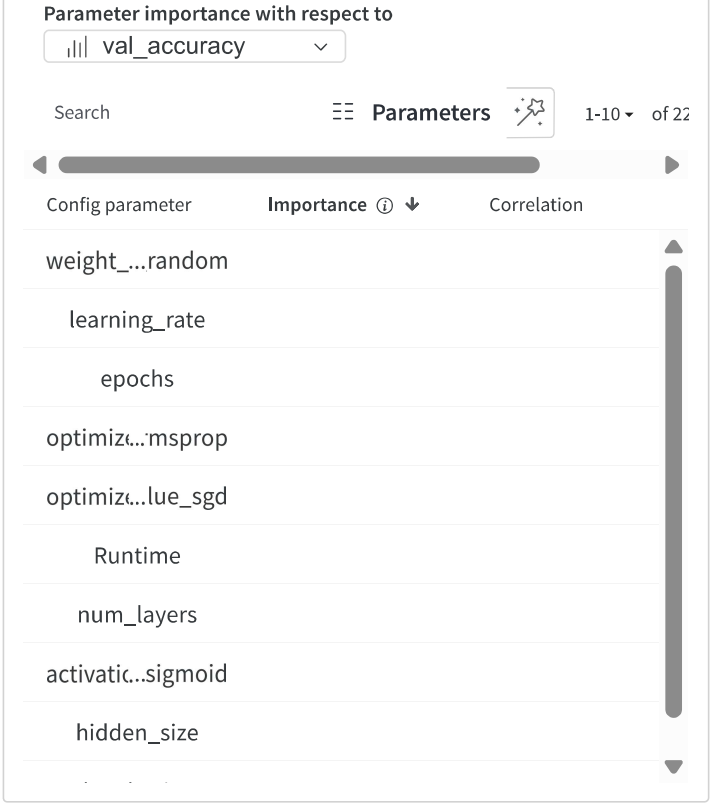
- random weight initialization when compared to Xavier initialization
- rmsprop optimizer may be due to killing of learning rate
- 3 and 4 hidden layer counts, 5 hidden layer is performing better as it may be fitting better

For getting higher accuracy(>95%), we may try

- More number of layers, maybe the model if underfitting

- Better weight initialization techniques, or other regularization methods
- As it is image data, CNN model should also be explored



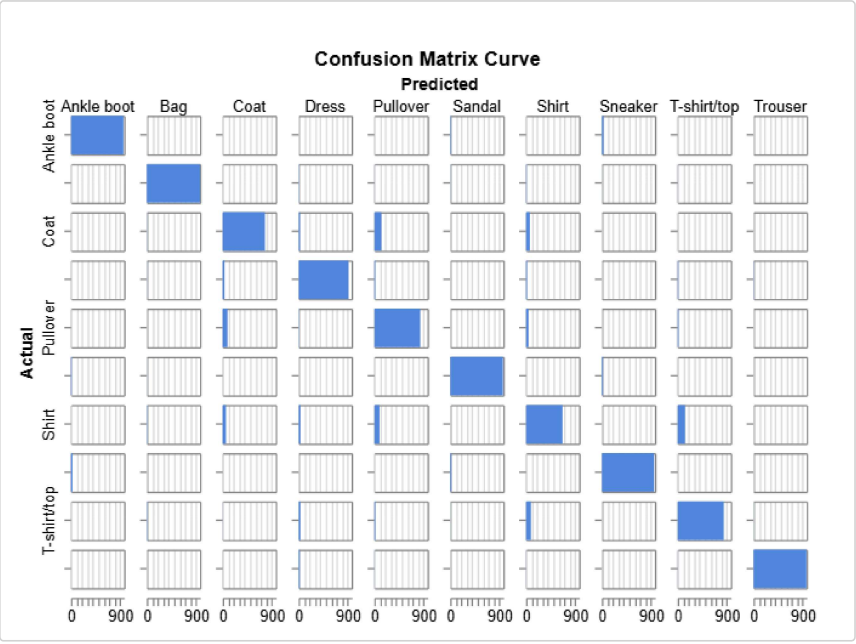


- Ran multiple sweeps on my model using different methods provided by wandb - random, grid and bayes. Out of which bayes gave the best performance.
- Some observations from the above plot and table:-
  - Xavier weight initialization method gave a big improvement over random initialization
  - High batch sizes also helped as they give better approximation of true gradient direction
  - High number of neuron also helped increasing validation accuracy, which is expected as the model gets trained longer
  - Large learning rate also affects validation accuracy which indicates that initialization may be far from minima or may be the loss surface is gentle
  - Optimizers adam and nadam also performed better than nesterov and rmsprop
  - High layer count also affects validation accuracy as they may provide more complex function to b fitted over the data
  - Activation tanh generally performed better than sigmoid and relu

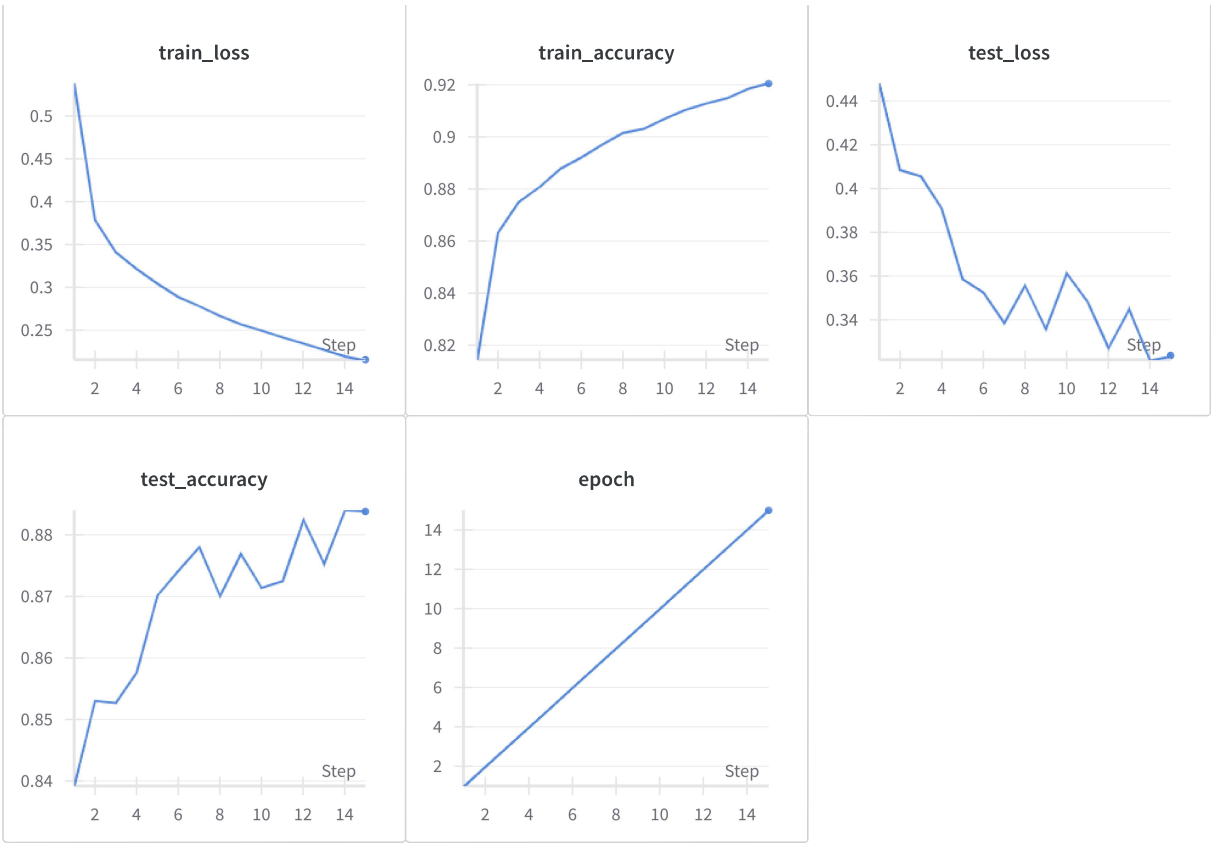
Question 7

The best fitted model gave a training accuracy of 0.92053 and test accuracy of 0.8838

I ran it for 15 epochs.



- Best model parameters over the run are plotted below.



- Sample test data predictions:

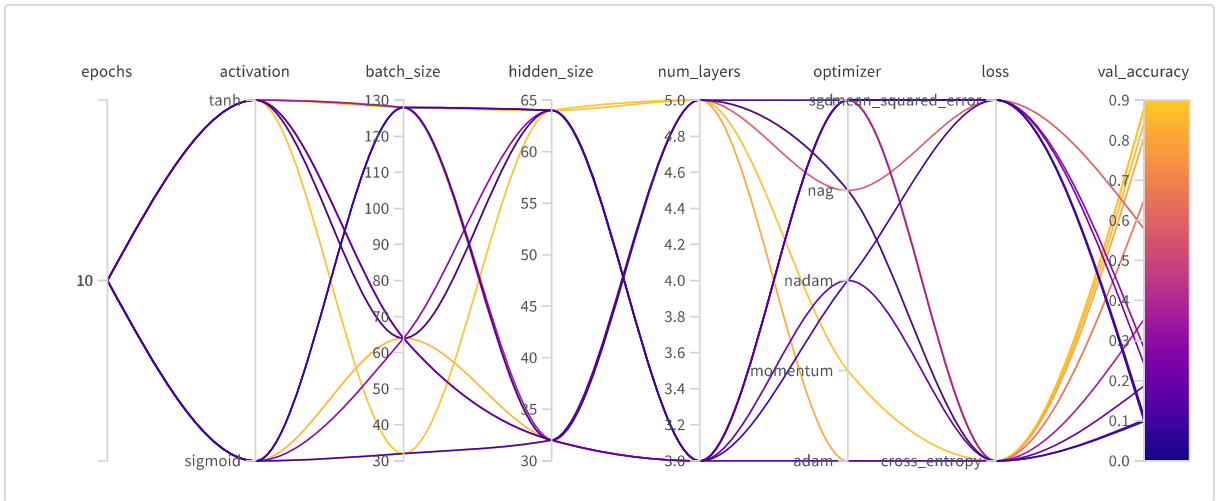
```
runs.summary["Prediction Samples"]
```

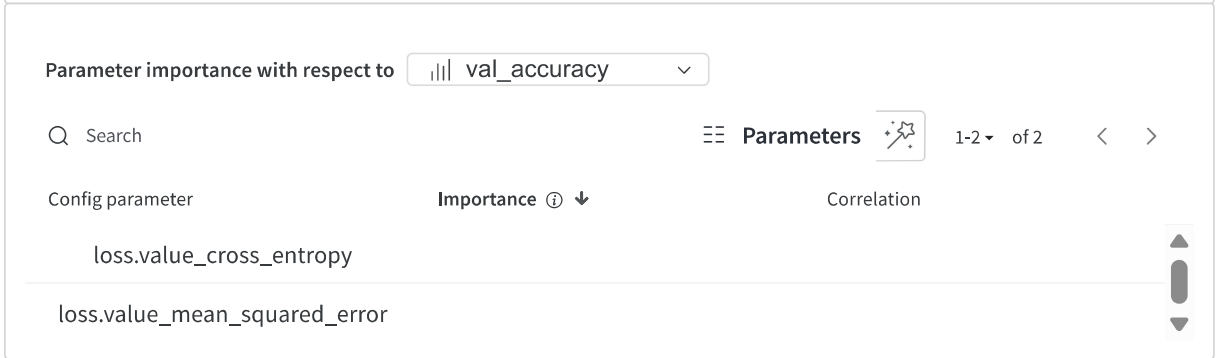
	Image	True Label	Predicted Label
1		Trouser	Trouser
2		Dress	Bag
3		Trouser	Trouser
4		Sandal	Sandal
5		T-shirt/top	T-shirt/top
6		T-shirt/top	T-shirt/top
7		Bag	Bag
8		Coat	Coat
9		Sandal	Sandal
		T-shirt/top	T-shirt/top

Question 8

- Cross-entropy loss is better suited for multi-class classification when compared to mean-squared-error loss because:
  - Cross entropy loss reflects the probabilistic nature of the problem while mse don't
  - Cross entropy loss produces larger gradients for incorrect classification, thus more effective learning
  - Cross entropy loss treats the outputs and true labels as probability distribution and measures the difference between them.

- The plots below also shows that cross-entropy generally performs better than mean-squared-loss over random sweep.





Question 9

Here is the Github repo link:-

https://github.com/Siddhant-DA24M021/DA6401-assignment-01-Siddhant-DA24M021.git

Question 10

I ran the experiment on mnist dataset:-

Configuration 1:

python train.py --dataset mnist --epochs 15 --batch_size 128 --hidden_size 64 --num_layers 3 --activation tanh --optimizer adam --learning_rate 0.001 --loss cross_entropy --weight_init Xavier

Reasoning:- I want to try this configuration because this is the configuration which gave me highest validation accuracy on the fashion mnist dataset. So, this should be the obvious choice to try first.

Run summary:

epoch: 15

test_accuracy: 0.9756

test_loss: 0.09337

train_accuracy: 0.99552

train_loss: 0.01519

Configuration 2:

python train.py --dataset mnist --epochs 15 --batch_size 128 --hidden_size 64 --num_layers 5 --activation tanh --optimizer rmsprop --learning_rate 0.01 --loss cross_entropy --weight_init Xavier

Reasoning:- I want to try this configuration because this is has a deeper network with more features so that it can learn more nuanced characteristics of data, with rmsprop optimizer and higher learning rate.

Run summary:

epoch: 15

test_accuracy: 0.9556

test_loss: 0.17793

train_accuracy: 0.9655

train_loss: 0.13206

Configuration 3:

python train.py --dataset mnist --epochs 15 --batch_size 128 --hidden_size 128 --num_layers 2 --activation ReLU --optimizer adam --learning_rate 0.003 --loss cross_entropy --weight_init Xavier

Reasoning:- I want to try this configuration because higher learning rate (0.003) with the Adam optimizer to see if it accelerates convergence or improves final accuracy compared to the baseline configuration. A higher learning rate can sometimes allow the model to escape local minima and potentially find a slightly better solution faster

Run summary:

epoch: 15

test_accuracy: 0.9761

test_loss: 0.11858

train_accuracy: 0.99412

train_loss: 0.01751

Self Declaration:-

I, Siddhant Baranwal (DA24M021), 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.