Edit Share

CS6910 - Assignment 1

Write your own backpropagation code and keep track of your experiments using wandb.ai

Purvam Jain

Instructions

Problem Statement

Question 1 (2 Marks)

Download the fashion-MNIST dataset and plot 1 sample image for each class as shown in the grid below. Use `from keras.datasets import fashion_mnist` for getting the fashion mnist dataset.

Image from each class

log_images

T-shirt/top

Index 0 ● 9 0

Question 2 (10 Marks)

Implement a feedforward neural network which takes images from the fashion-mnist data as input and outputs a probability distribution over the 10 classes.

Your code should be flexible such that it is easy to change the number of hidden layers and the number of neurons in each hidden layer.

Question 3 (24 Marks)

Implement the backpropagation algorithm with support for the following optimisation functions

- sgd
- momentum based gradient descent
- nesterov accelerated gradient descent
- rmsprop
- adam
- nadam

(12 marks for the backpropagation framework and 2 marks for each of the optimisation algorithms above)

We will check the code for implementation and ease of use (e.g., how easy it is to add a new optimisation algorithm such as Eve). Note that the code should be flexible enough to work with different batch sizes.

Question 4 (10 Marks)

Use the sweep functionality provided by wandb to find the best values for the hyperparameters listed below. Use the standard train/test split of fashion_mnist (use `(X_train, y_train), (X_test, y_test) = fashion_mnist.load_data()`). Keep 10% of the training data aside as validation data for this hyperparameter search. Here are some suggestions for different values to try for hyperparameters. As you can quickly see that this leads to an exponential number of combinations. You will have to think about strategies to do this hyperparameter search efficiently. Check out the options provided by wandb.sweep and write down what strategy you chose and why.

- number of epochs: 5, 10
- number of hidden layers: 3, 4, 5
- size of every hidden layer: 32, 64, 128
- weight decay (L2 regularisation): 0, 0.0005, 0.5
- learning rate: 1e-3, 1 e-4
- optimizer: sgd, momentum, nesterov, rmsprop, adam, nadam
- batch size: 16, 32, 64
- weight initialisation: random, Xavier
- activation functions: sigmoid, tanh, ReLU

wandb will automatically generate the following plots. Paste these plots below using the "Add Panel to Report" feature. Make sure you use meaningful names for each sweep (e.g. hl_3_bs_16_ac_tanh to indicate that there were 3 hidden layers, batch size was 16 and activation function was ReLU) instead of using the default names (whole-sweep, kind-sweep) given by wandb.

Answer:

A naive grid search of the whole hyperparameter space for 9 hyperparameters would be very costly to be done effectively. Instead structuring the the hyperparameter search into 3 steps, forming 3 groups of roughly descending order of importance (determined by prior experience), each comprising one sweep (all individual grid searches) is better:

- Fixing the network architecture and optimizer: Optimizer, Number of layers, Size of each layer
- Optimizing the speed and time characteristic of the network: Learning Rate, Epochs
- Fine tuning the other hyperparameters: L2 regularization, Batch size, Weight initialization method, Activation function

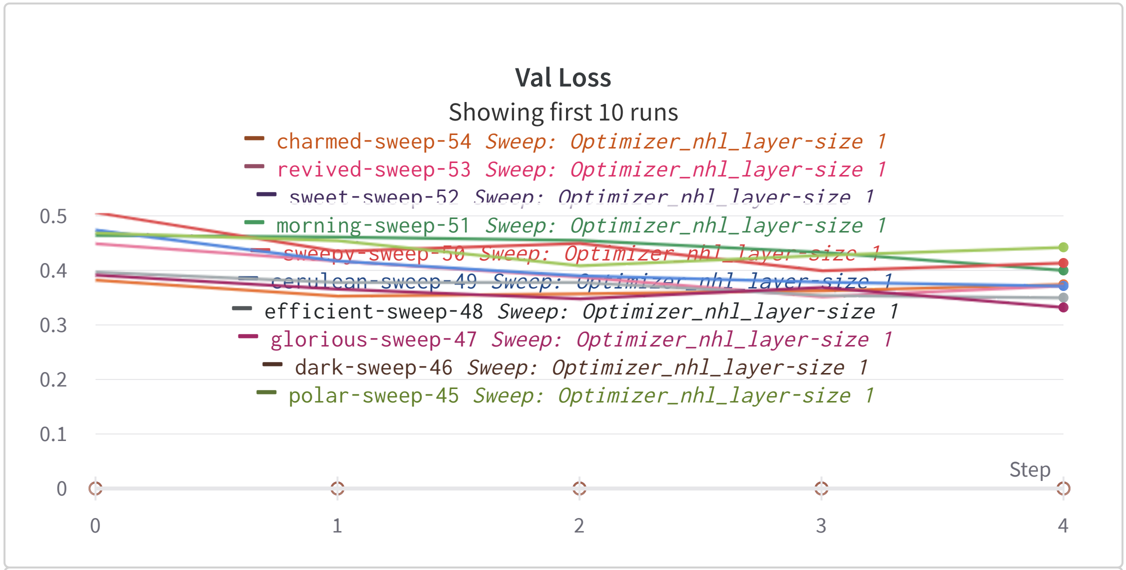
Validation Accuracy is used as the metric to compare the models in each sweep. As can be calculated, the number of models tested with this algorithm is 6x3x3 + 5x2 + 3x3x2x3 = 54 + 6 + 54 = 114 which is way less than the 2x3x3x3x2x6x3x2x3 = 11664 models required for naive grid search.

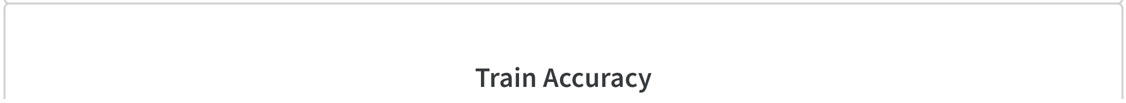
a) Sweep 1 (Network Architecture and Optimizer):

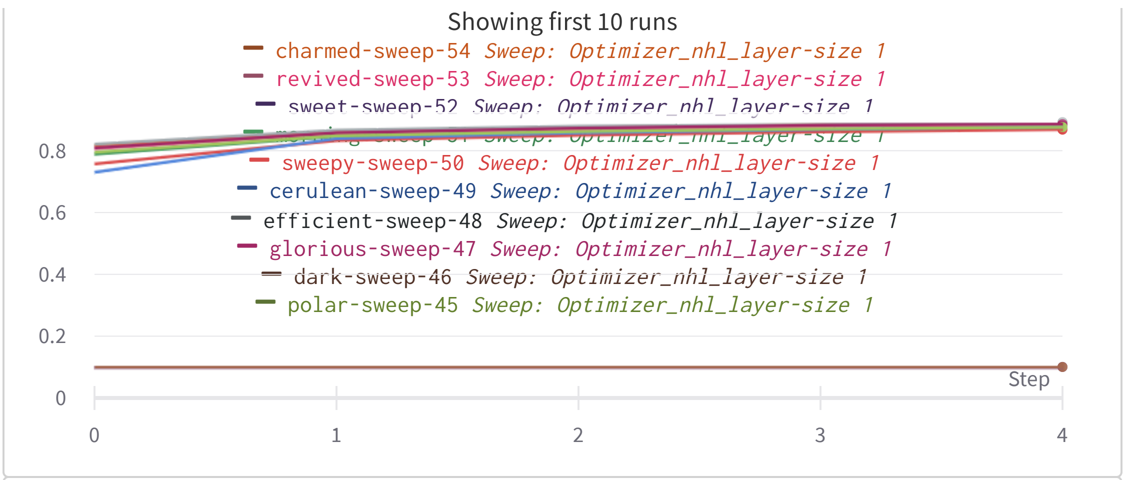
Tested the following hyperparameter values:

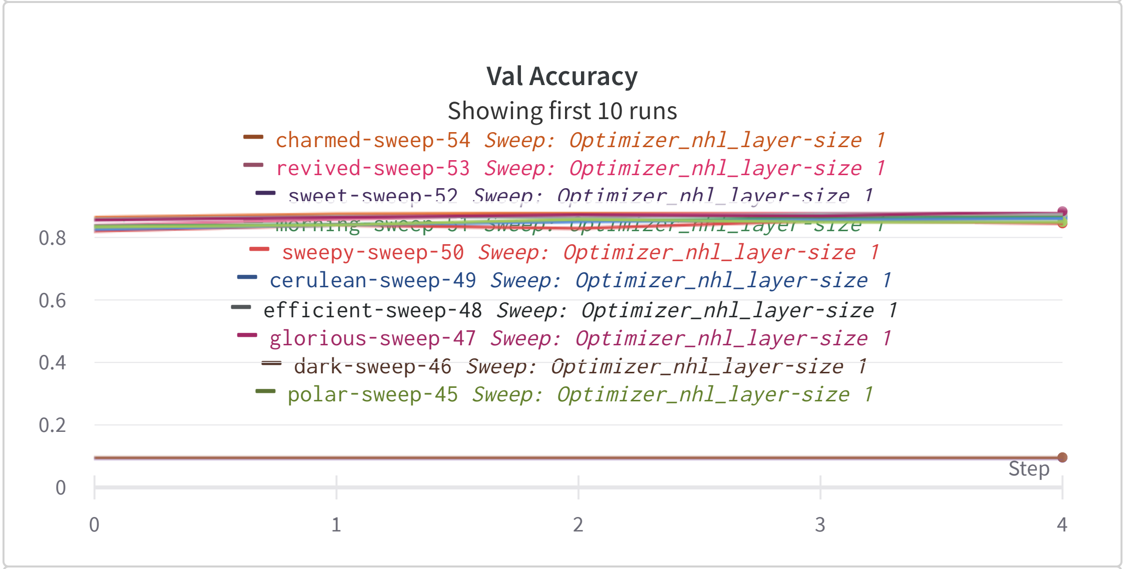
- Optimizer: sgd, momentum, nag, rmsprop, adam, nadam
- Number of Layers: 3, 4, 5
- Layer Size: 32, 64, 128

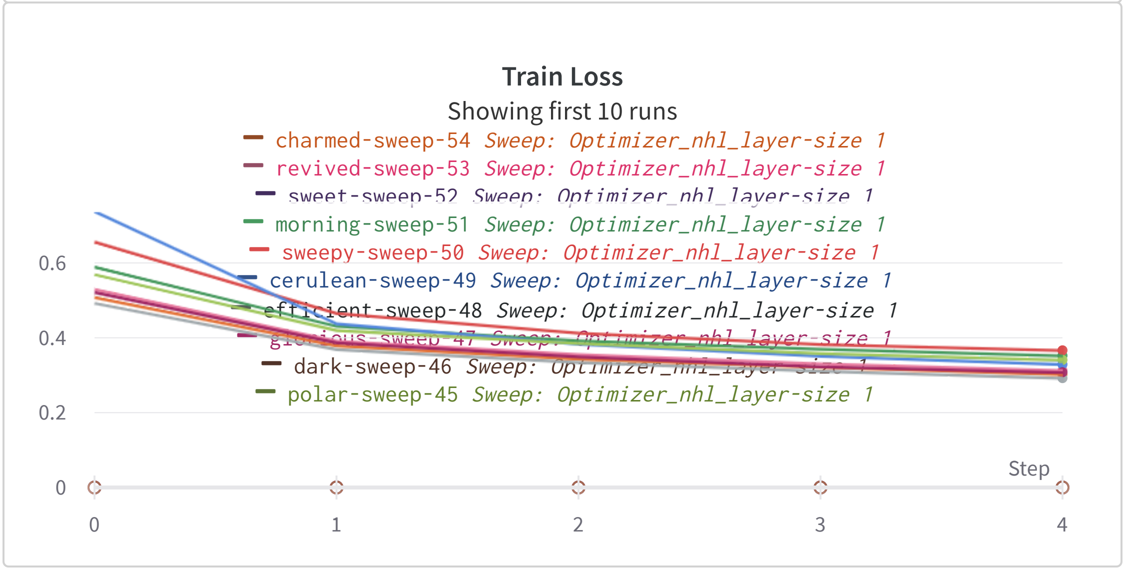
The best configuration was found to be (Adam, 4, 128) which was fixed as the default for ensuing sweeps. Plots which summarise this sweep are shown below.











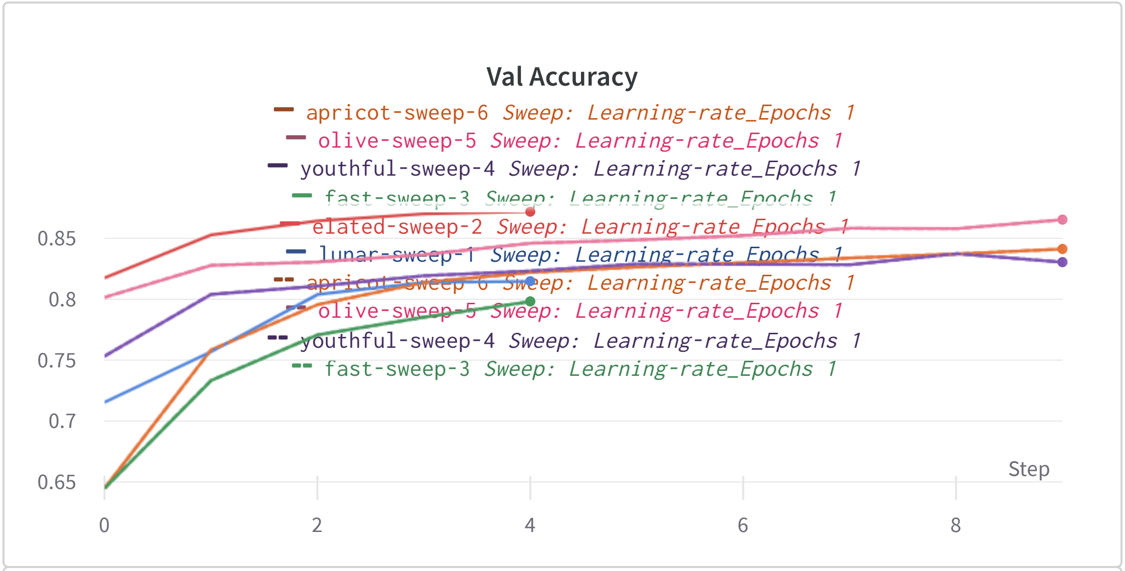
b) Sweep 2 (Speed and Time):

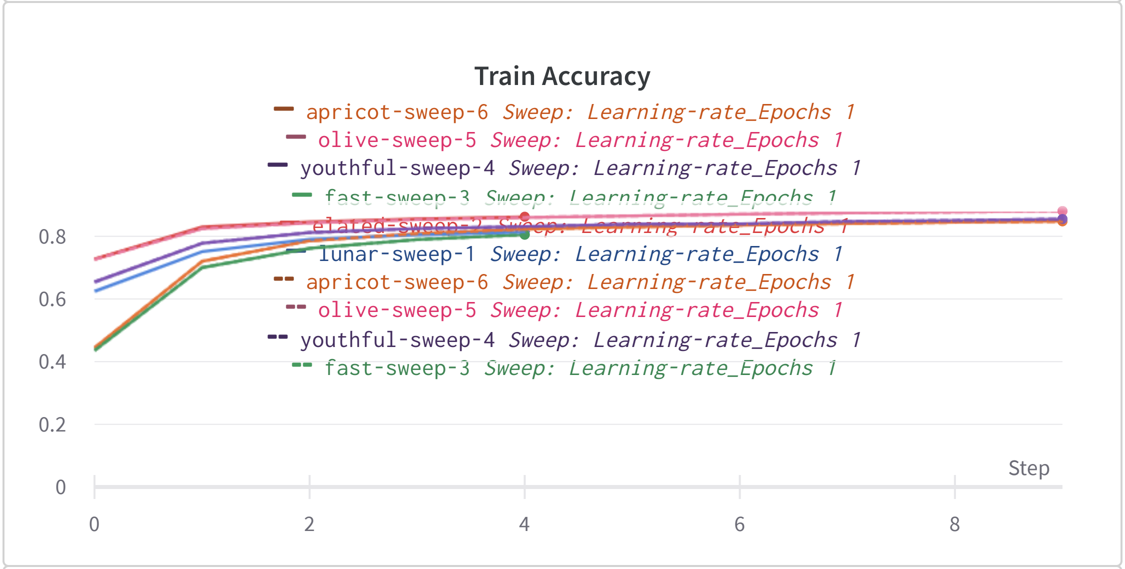
Tested the following hyperparameter values:

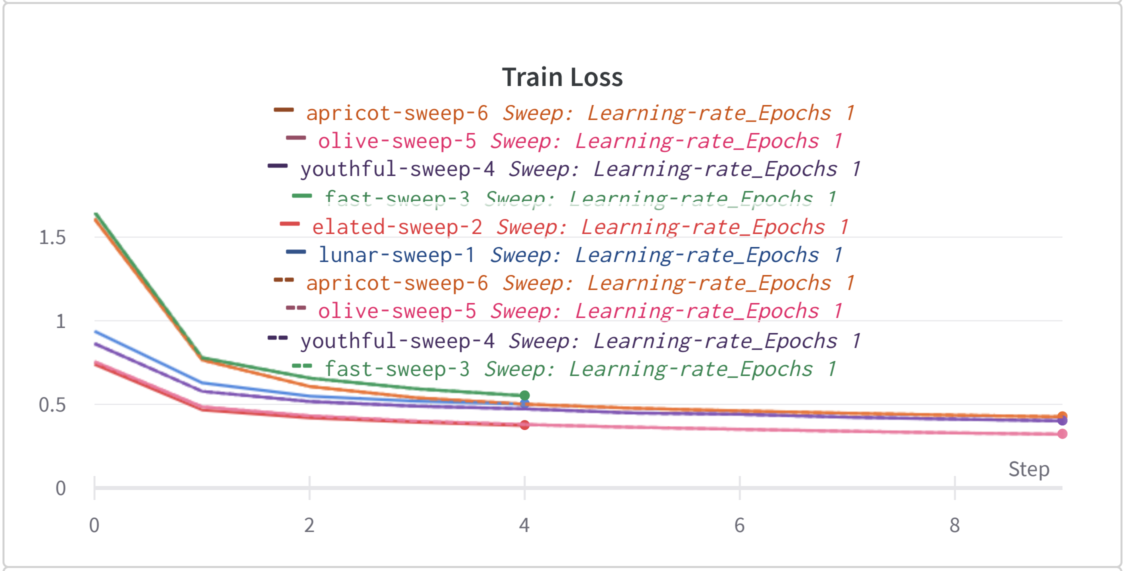
- Learning Rate: 0.0001, 0.001, 0.01

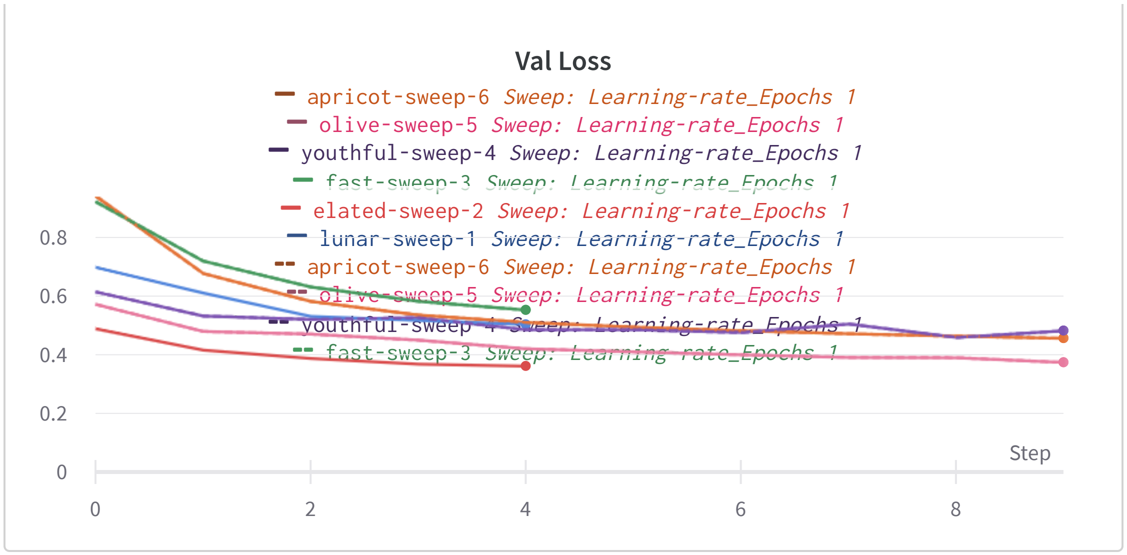
- Epochs: 5, 10

The best configuration was found to be (0.001, 10) which was fixed as the default for the last sweep. Plots which summarise this sweep are shown below.









c) Sweep 3 (Other Hyperparameters):

Tested the following hyperparameter values:

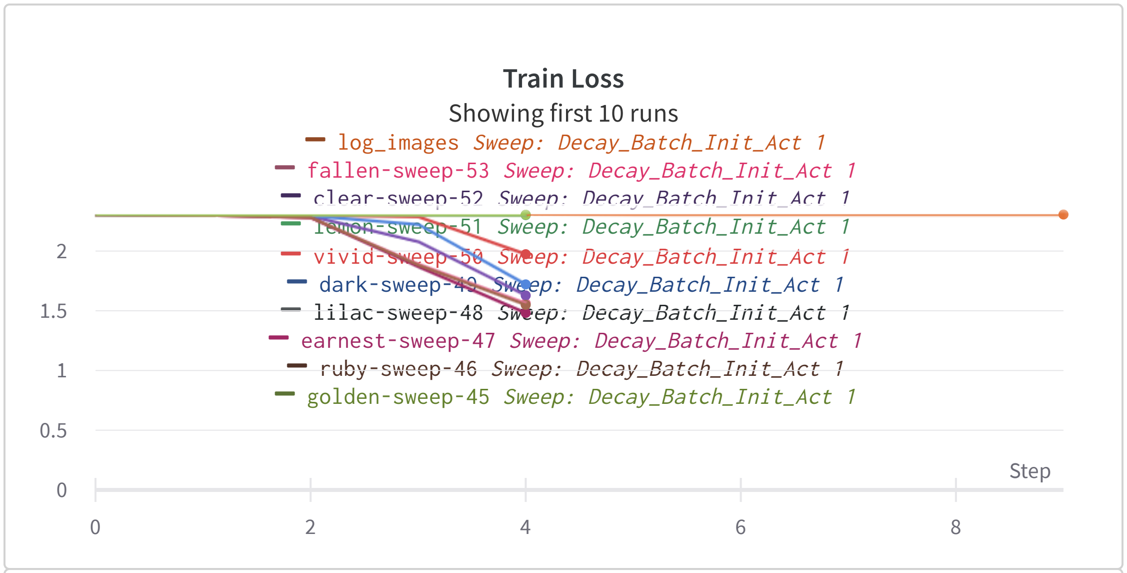
- Weight Decay (L2 regularization): 0, 0.0005, 0.05

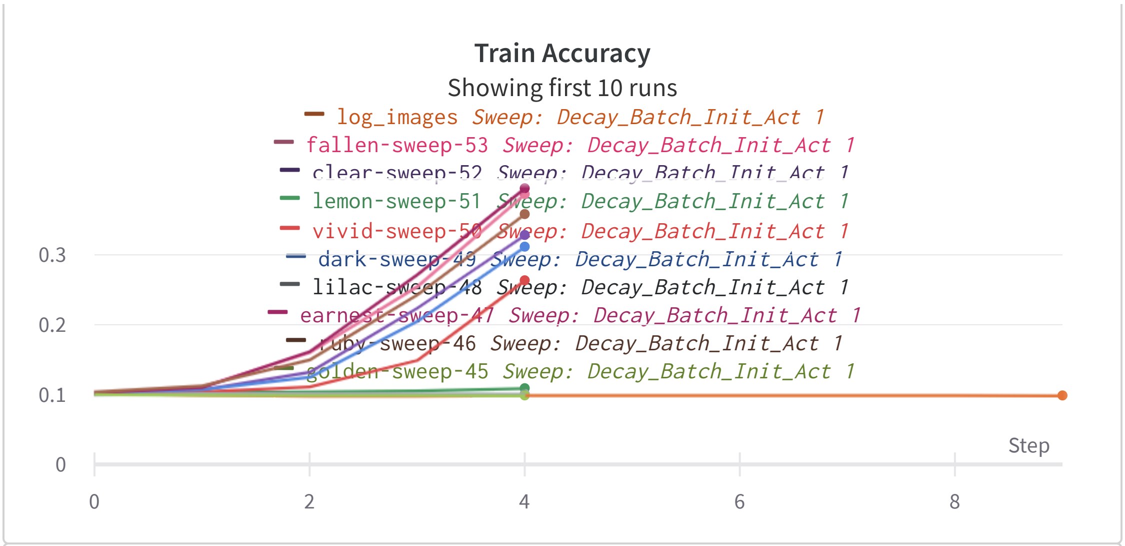
- Batch Size: 16, 32, 64

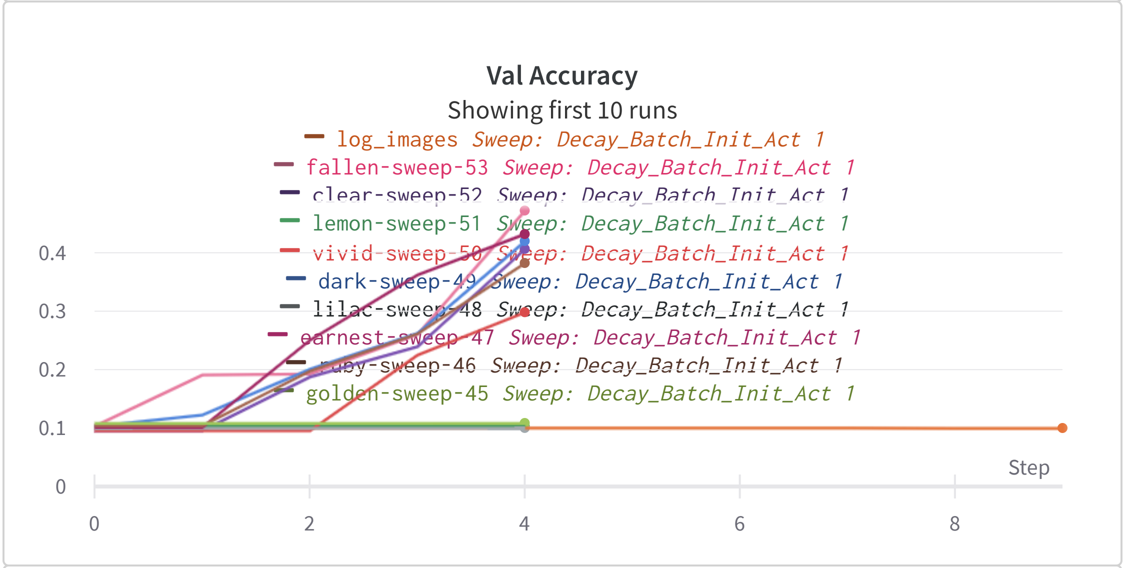
- Weight Initialization Method: random, xavier

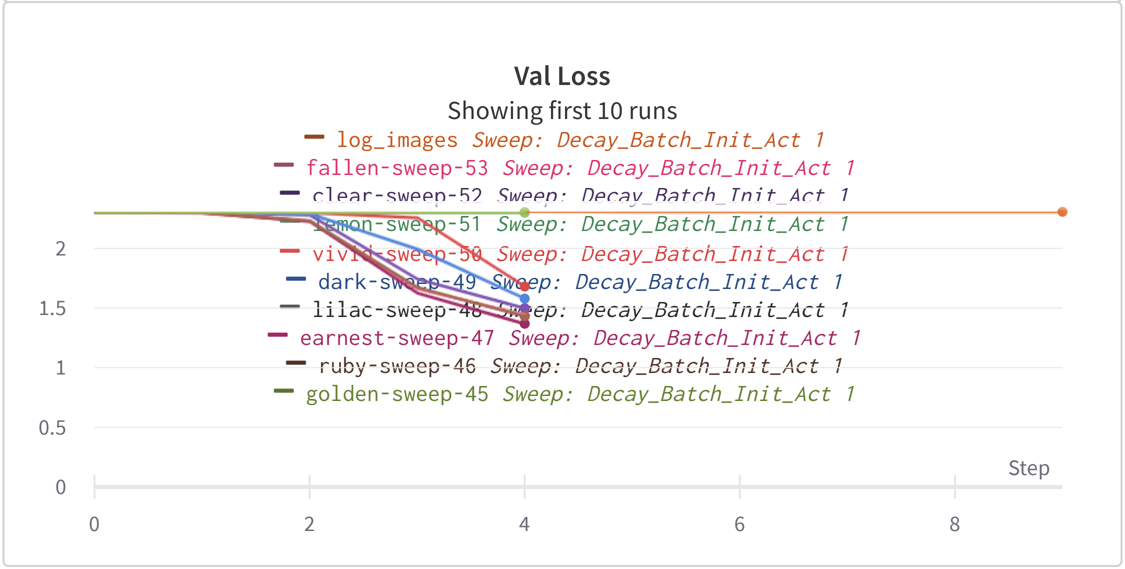
- Activation Function: relu, tanh, sigmoid

The best configuration was found to be (0, 64, xavier, relu) which then gives us the best neural network model for this dataset. Plots which summarise this sweep are shown below.





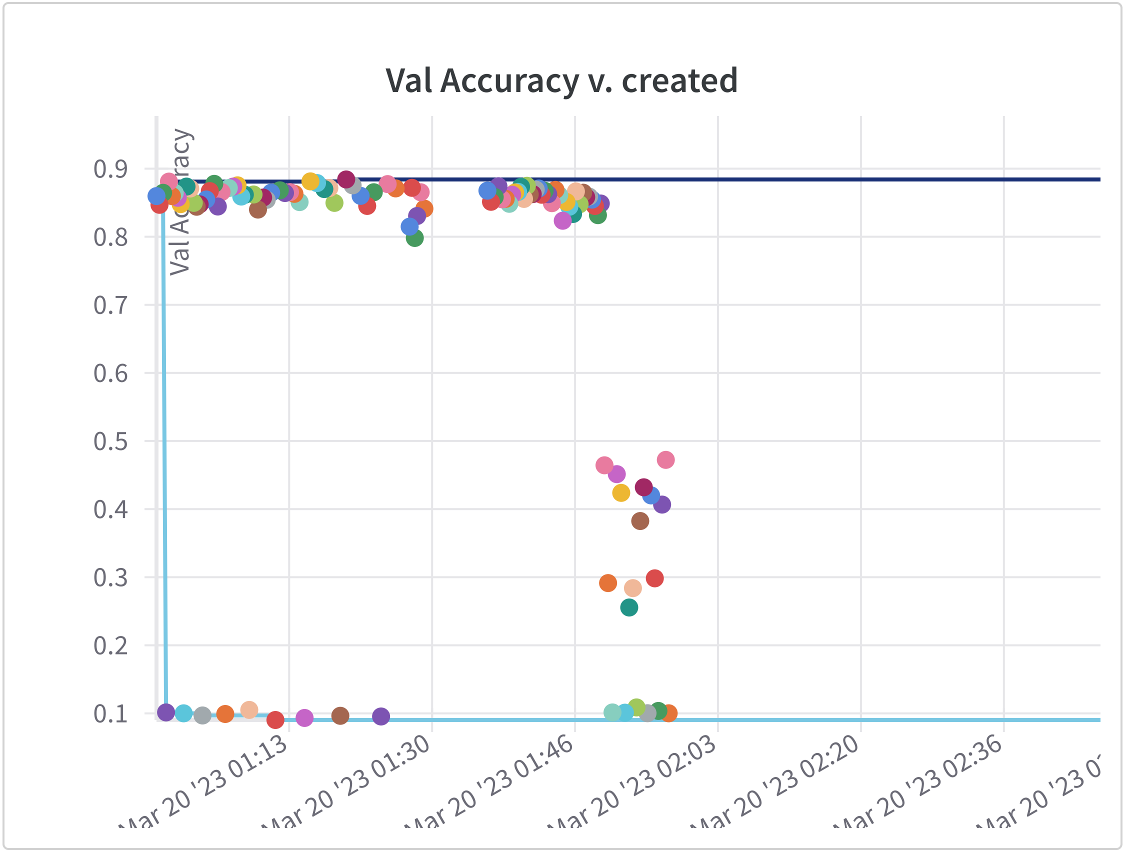




Question 5 (5 marks)

We would like to see the best accuracy on the validation set across all the models that you train.

wandb automatically generates this plot which summarises the test accuracy of all the models that you tested. Please paste this plot below using the "Add Panel to Report" feature



Question 6 (20 Marks)

Based on the different experiments that you have run we want you to make some inferences about which configurations worked and which did not.

Here again, wandb automatically generates a "Parallel co-ordinates plot" and a "correlation summary" as shown below. Learn about a "Parallel co-ordinates plot" and how to read it.

By looking at the plots that you get, write down some interesting observations (simple bullet points but should be insightful). You can also refer to the plot in Question 5 while writing these insights. For example, in the above sample plot there are many configurations which give less than 65% accuracy. I would like to zoom into those and see what is happening.

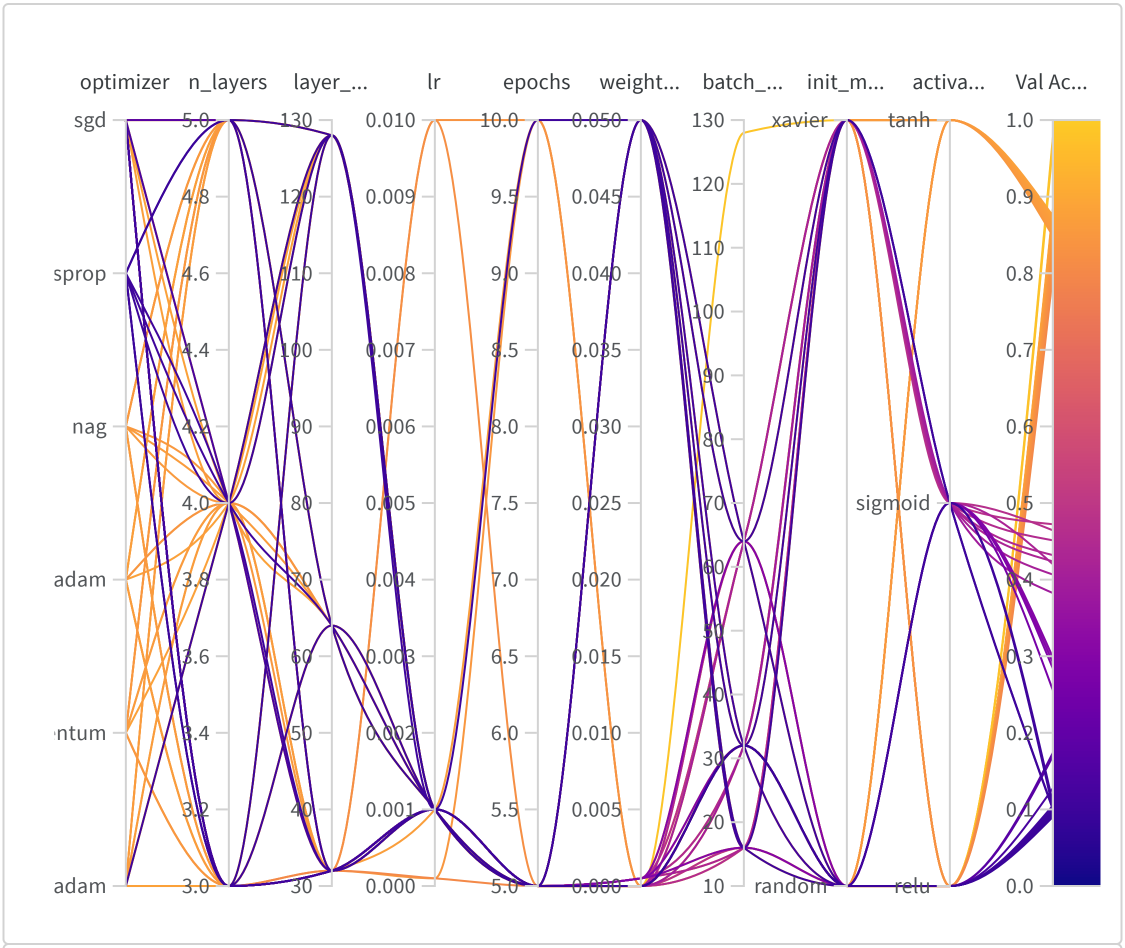
I would also like to see a recommendation for what configuration to use to get close to 95% accuracy.

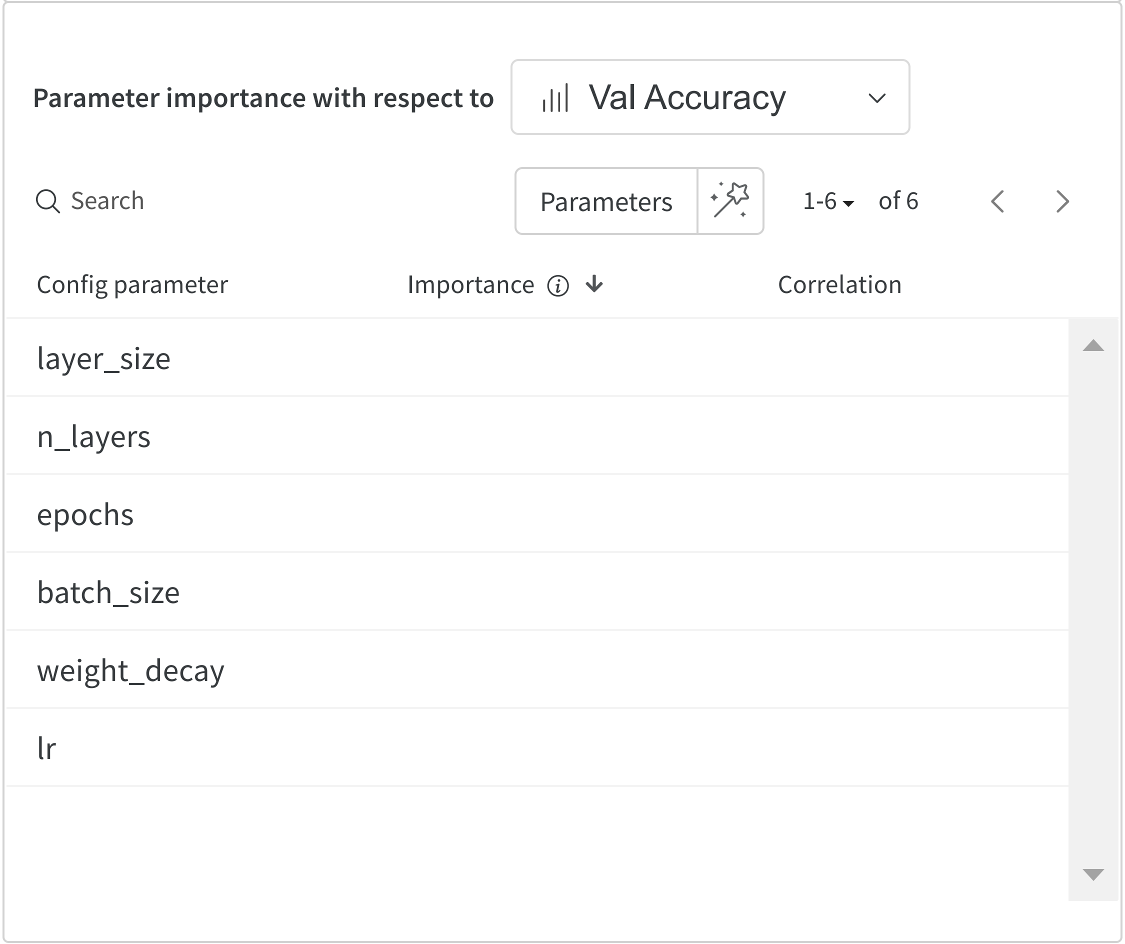
Answer

- RMSProp and Sigmoid seem to have the worst performance among all optimizers, possible cause could be wrong set of hyperparameters for this particular dataset.
- n_layers*layer_size determines the complexity of a model. In our case as was found from subsequent sweeps in stages, 4 hidden layers with 128 neurons seem to perform the best.
- Ignoring the lines originating from RMSProp , we find that all learning rates and epoch variations are performing withing a small margin of accuracy around 88%.
- Batch_size doesn't have much effect which can be confirmed from its correlation factor close to zero.
- Weight decay plays an important role here, where the lack of it seems to give the best performance. Whenever weight decay is introduced we see a decrease in accuracy. It is possible that over higher epochs weight decay might perform better, but in the current scenario since our validation and training accuracies are quite close, there's no overfitting.

To get accuracy close to 95%:

- Train the model for more epochs, ideally model should have more layers along with regularization for better feature generation.
- Instead of training ANNs we can use CNN architecture which are more suited for image data.
- We can provide some additional features along with network output.





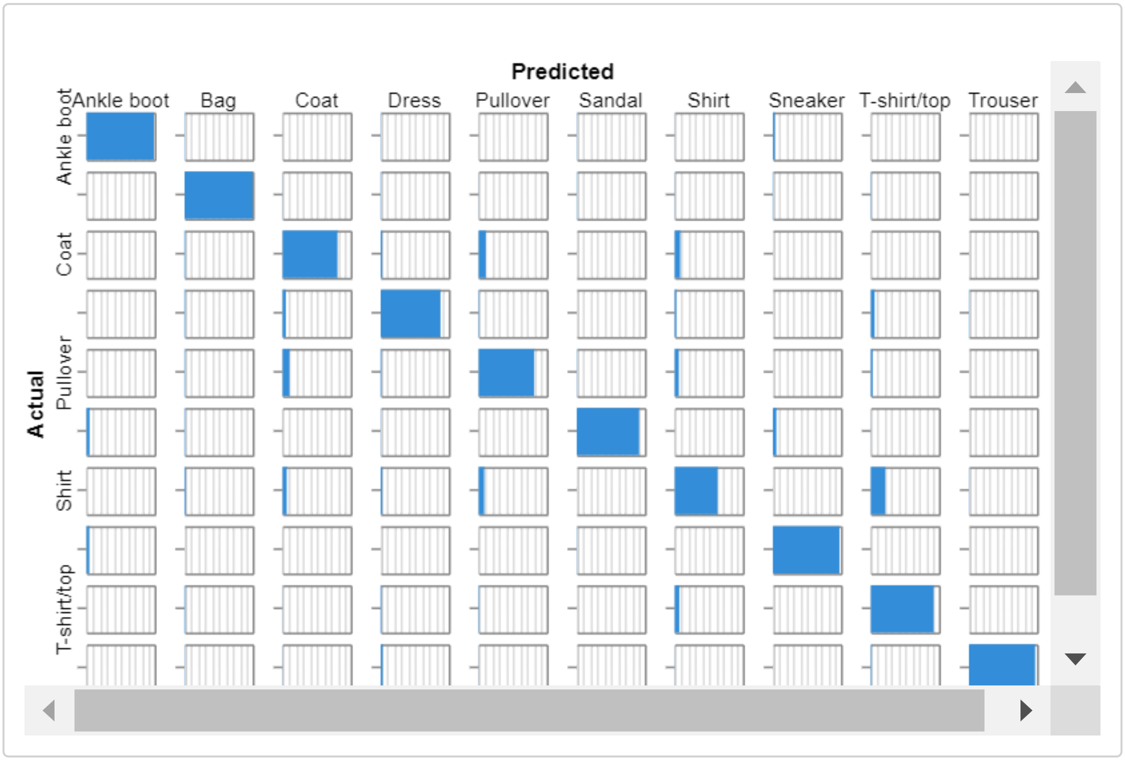
Parameter importance with respect to Val Accuracy

Search Parameters 1-6 of 6

Config parameter	Importance	Correlation
layer_size		
n_layers		
epochs		
batch_size		
weight_decay		
lr		

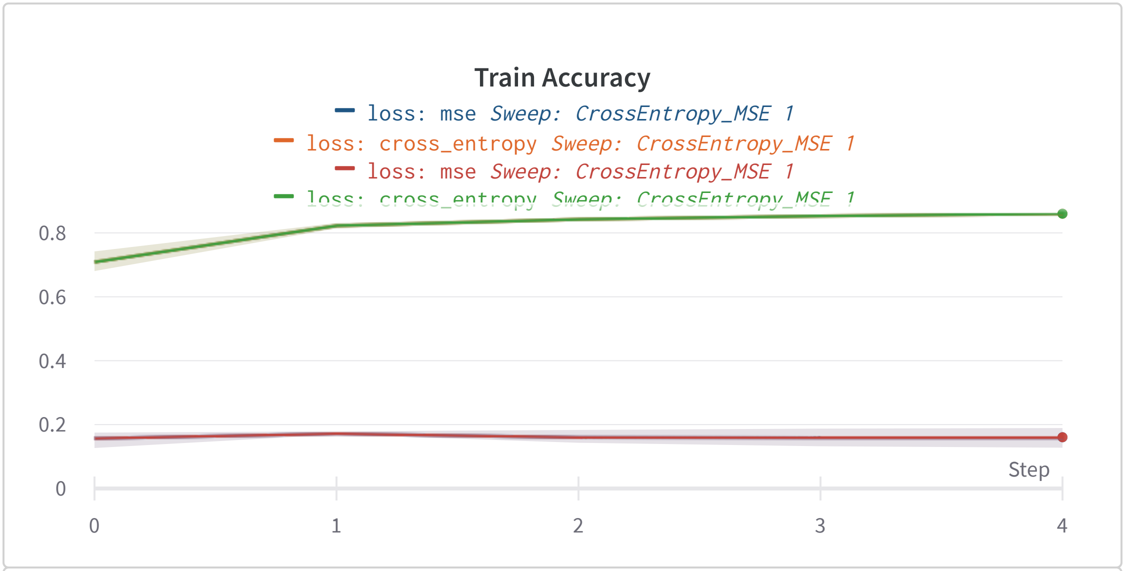
Question 7 (10 Marks)

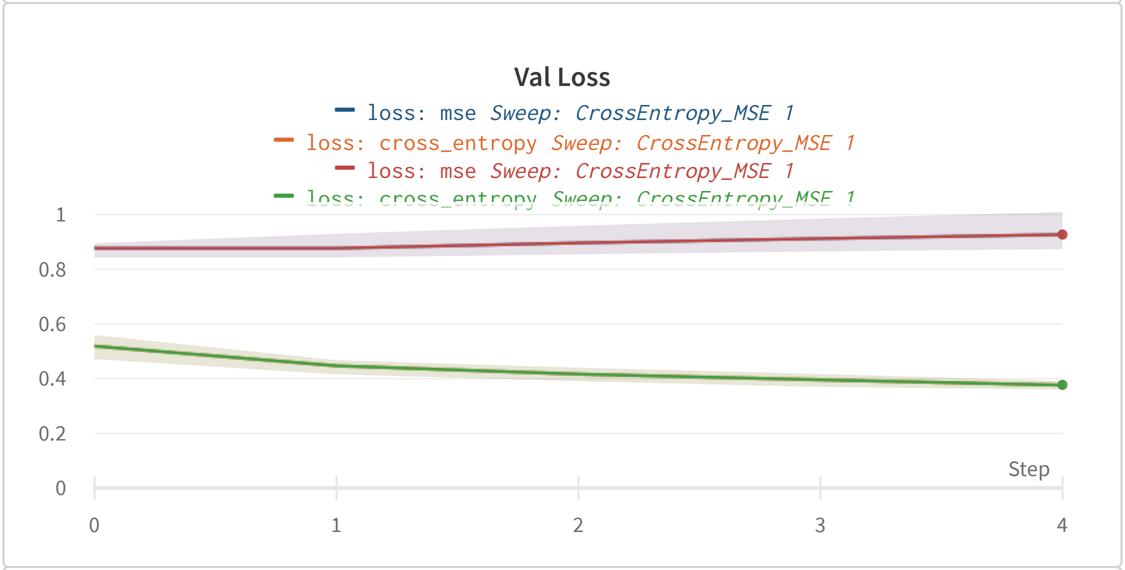
For the best model identified above, report the accuracy on the test set of fashion_mnist and plot the confusion matrix as shown below. More marks for creativity (less marks for producing the plot shown below as it is)
Here, we not only see True Positive and False Negatives but also, the misclassified labels which can give us more insight into which classes our model is confident in performance and where does it lack in performance. AS we can see our model is confused between classes as "Shirt" and "T-Shirt/Top".

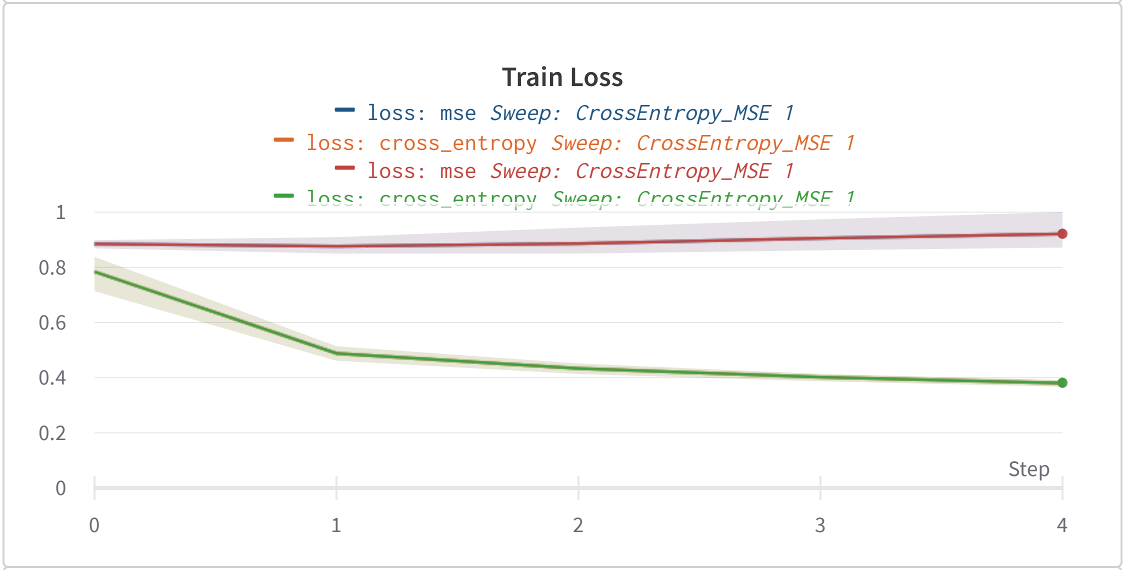


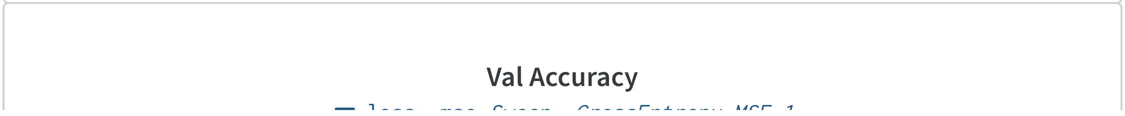
Question 8 (5 Marks)

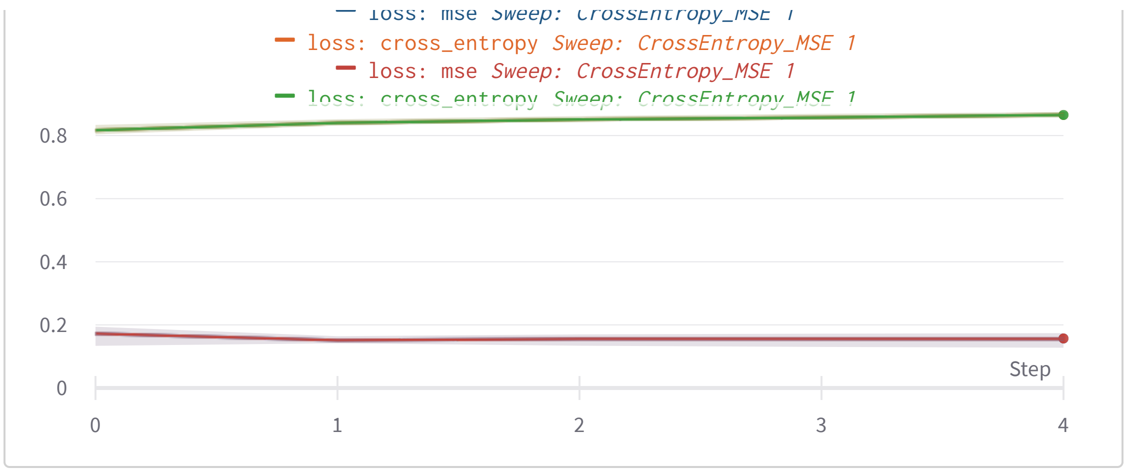
In all the models above you would have used cross entropy loss. Now compare the cross entropy loss with the squared error loss. I would again like to see some automatically generated plots or your own plots to convince me whether one is better than the other.











Question 9 (10 Marks)

Paste a link to your github code for this assignment

Example: https://github.com/wigglytuff-tu/CS6910;

- We will check for coding style, clarity in using functions and a `README` file with clear instructions on training and evaluating the model (the 10 marks will be based on this)
- We will also run a plagiarism check to ensure that the code is not copied (0 marks in the assignment if we find that the code is plagiarized)
- We will also check if the training and test data has been split properly and randomly. You will get 0 marks on the assignment if we find any cheating (e.g., adding test data to training data) to get higher accuracy

Question 10 (10 Marks)

Based on your learnings above, give me 3 recommendations for what would work for the MNIST dataset (not Fashion-MNIST). Just to be clear, I am asking you to take your learnings based on extensive experimentation with one dataset and see if these learnings help on another dataset. If I give you a budget of running only 3 hyperparameter configurations as opposed to the large number of experiments you have run above then which 3 would you use and why. Report the accuracies that you obtain using these 3 configurations.

Since we have already determined the ideal set of hyperparameters for FashionMNIST as : number of hidden layer = 4, size = 128, activation = RELU, optimizer = adam, lr=0.001, weight_decay = 0, epochs = 5. It leaves us with the option to vary the batch_size, ideally a larger batch size should be a better approximation of dataset so we will use the three as params:

- batch size = 32 : Train = 0.9826, Val = 0.9682
- batch size = 64 : Train = 0.9841, Val = 0.9729
- batch size = 32 : Train = 0.9839, Val = 0.9773

Code Specifications

Please ensure you add all the code used to run your experiments in the GitHub repository.

You must also provide a python script called `train.py` in the root directory of your GitHub repository that accepts the following command line arguments with the specified values -

We will check your code for implementation and ease of use. We will also verify your code works by running the following command and checking wandb logs generated -

```
python train.py --wandb_entity myname --wandb_project myprojectname
```

Arguments to be supported

Name	Default Value	Description
`-wp`, `--wandb_project`	myprojectname	Project name used to track experiments in Weights & Biases dashboard

Name	Default Value	Description
`-we`, `--wandb_entity`	myname	Wandb Entity used to track experiments in the Weights & Biases dashboard.
`-d`, `--dataset`	fashion_mnist	choices: ["mnist", "fashion_mnist"]
`-e`, `--epochs`	1	Number of epochs to train neural network.
`-b`, `--batch_size`	4	Batch size used to train neural network.
`-l`, `--loss`	cross_entropy	choices: ["mean_squared_error", "cross_entropy"]
`-o`, `--optimizer`	sgd	choices: ["sgd", "momentum", "nag", "rmsprop", "adam", "nadam"]
`-lr`, `--learning_rate`	0.1	Learning rate used to optimize model parameters
`-m`, `--momentum`	0.5	Momentum used by momentum and nag optimizers.
`-beta`, `--beta`	0.5	Beta used by rmsprop optimizer
`-beta1`, `--beta1`	0.5	Beta1 used by adam and nadam optimizers.
`-beta2`, `--beta2`	0.5	Beta2 used by adam and nadam optimizers.
`-eps`, `--epsilon`	0.000001	Epsilon used by optimizers.
`-w_d`, `--weight_decay`	.0	Weight decay used by optimizers.
`-w_i`, `--weight_init`	random	choices: ["random", "Xavier"]
`-nhl`, `--num_layers`	1	Number of hidden layers used in feedforward neural network.
`-sz`, `--hidden_size`	4	Number of hidden neurons in a feedforward layer.
`-a`, `--activation`	sigmoid	choices: ["identity", "sigmoid", "tanh", "ReLU"]

Please set the default hyperparameters to the values that give you your best validation accuracy. (Hint: Refer to the Wandb sweeps conducted.)

You may also add additional arguments with appropriate default values.

Self Declaration

I, Purvam Jain, 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.