CS6910 Assignment - 1

Name: Shrung D N

Roll No: ME19B168

WandB Report Link (make sure it is accessible)**:**
https://wandb.ai/me19b168/ME19B168_CS6910_Assgn1/reports/ME19B168-CS6910-Assignment-1--VmlldzozODIzODQz?accessToken=cogoq8iym13e83dexy5evfm21yy9sv48kvbd28igbsulqoin8bcb23mtpveduco1

GitHub Repository Link (make sure you have added the TAs as collaborators in your private repo)**:** https://github.com/ShrungDN/CS6910

ME19B168 - CS6910 - Assignment 1

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

Shrung D N

Instructions

- The goal of this assignment is twofold: (i) implement and use gradient descent (and its variants) with backpropagation for a classification task (ii) get familiar with Wandb which is a cool tool for running and keeping track of a large number of experiments
- This is a **individual assignment** and no groups are allowed.
- Collaborations and discussions with other students is strictly prohibited.
- You must use Python (NumPy and Pandas) for your implementation.
- You cannot use the following packages from Keras, PyTorch, Tensorflow: optimizers, layers
- If you are using any packages from Keras, PyTorch, Tensorflow then post on Moodle first to check with the instructor.
- You have to generate the report in the same format as shown below using wandb.ai. You can start by cloning this report using the clone option above. Most of the plots that we have asked for below can be (automatically) generated using the APIs provided by `wandb.ai`. You will upload a link to this report on Gradescope.
- You also need to provide a link to your GitHub code as shown

below. Follow good software engineering practices and set up a GitHub repo for the project on Day 1. Please do not write all code on your local machine and push everything to GitHub on the last day. The commits in GitHub should reflect how the code has evolved during the course of the assignment.

- You have to check Moodle regularly for updates regarding the assignment.

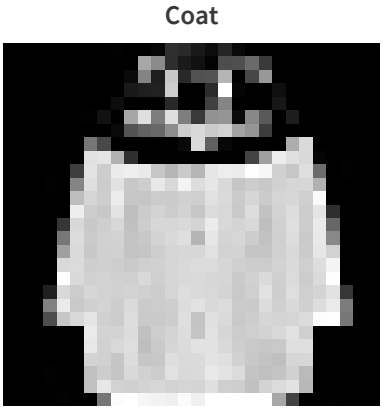
Problem Statement

In this assignment you need to implement a feedforward neural network and write the backpropagation code for training the network. We strongly recommend using numpy for all matrix/vector operations. You are not allowed to use any automatic differentiation packages. This network will be trained and tested using the Fashion-MNIST dataset. Specifically, given an input image (28 x 28 = 784 pixels) from the Fashion-MNIST dataset, the network will be trained to classify the image into 1 of 10 classes.

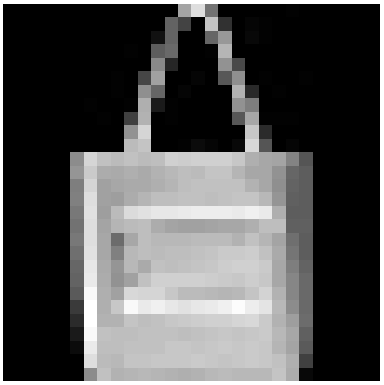
Your code will have to follow the format specified in the `Code Specifications` section.

Question 1 (2 Marks)

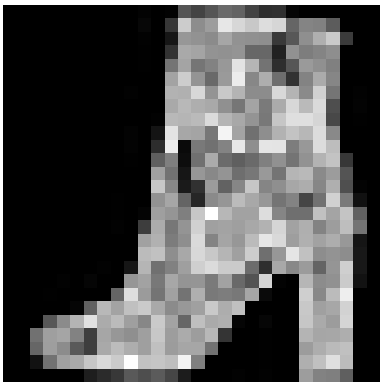
Download the fashion-MNIST dataset and plot 1 sample image for each class as shown in the grid below. Use `from keras.datasets import fashion_mnist` for getting the fashion mnist dataset.



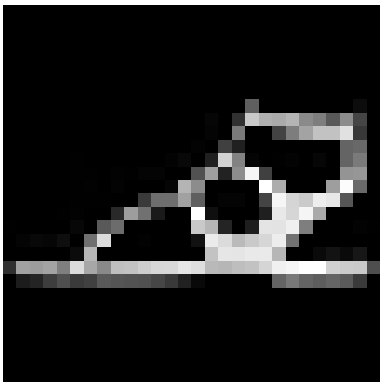
Bag



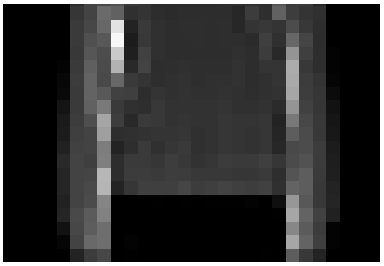
Ankle Boot



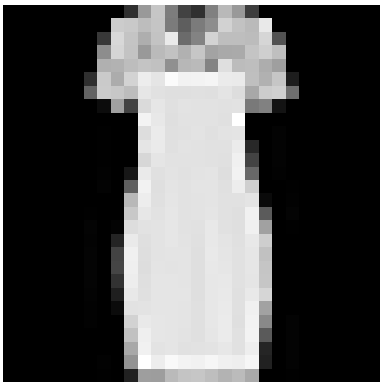
Sandal



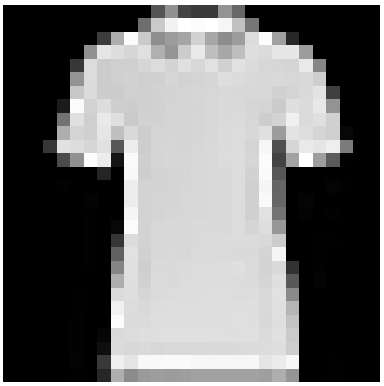
Pullover



Dress

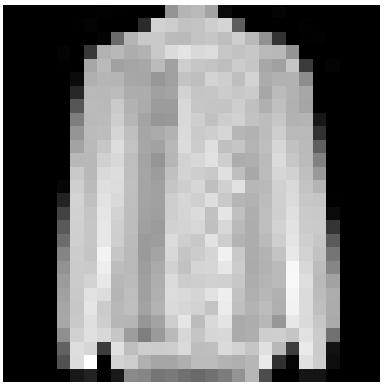


T-Shirt

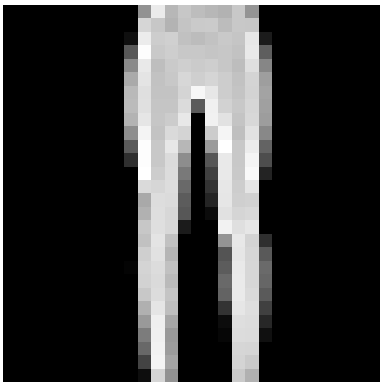


Sneaker

Shirt



Trouser



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.

Answer:

Another optimization algorithm can be added by modifying the "train.py" file. To do so, another "elif" statement needs to be added. The forward and backward propagation functions provide the derivative at a particular weights configuration. This can be used to add optimization algorithms easily.

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 total of 4 sweeps were done with 4 different configurations. This is because a grid search of the various combinations would lead to a lot of computational time and resources. Moreover, most of the runs in this method would have very less accuracies. Therefore, I chose the strategy of breaking down the sweep into multiple sweeps, and filtering out only the good configurations after each sweep.

Sweep 1:

- I ran a configuration which approximately matches the one given in the question. I used the "sweep table" tool on WandB to filter out sweeps that resulted in validation accuracy of 85% or above.
- I then grouped these filtered runs on the basis of different parameters.
- For weight initialization,I observed that Xavier initialization resulted in higher average validation accuracy than random initialization. Hence I discarded random initialization for future sweeps.
- For activation function, I observed that tanh and ReLu had much higher average validation accuracy than identity and sigmoid. Hence, I discarded identity and Sigmoid for the next sweep.
- Weight decay did not have a clear margin and hence, both the

values of 0 and 0.1 were used in the next sweep.

- For optimizer, 'nag' and 'sgd' performed very bad and hence were discared.
- In this sweep config, not a lot of importance was given to the shape of the neural network as the motive was to filter out the other parameters based on average validation accuracy.

Sweep 2:

- In this sweep, more importance was given to the shape of the neural network and optimizers.
- By comparing in a similar way, by using WandB's sweep table, I concluded that a neural network with 5 hidden layers, each consisting 128 neurons performed the best.
- The above neural network architecture worked best with adam and nadam optimizers.

Sweep 3:

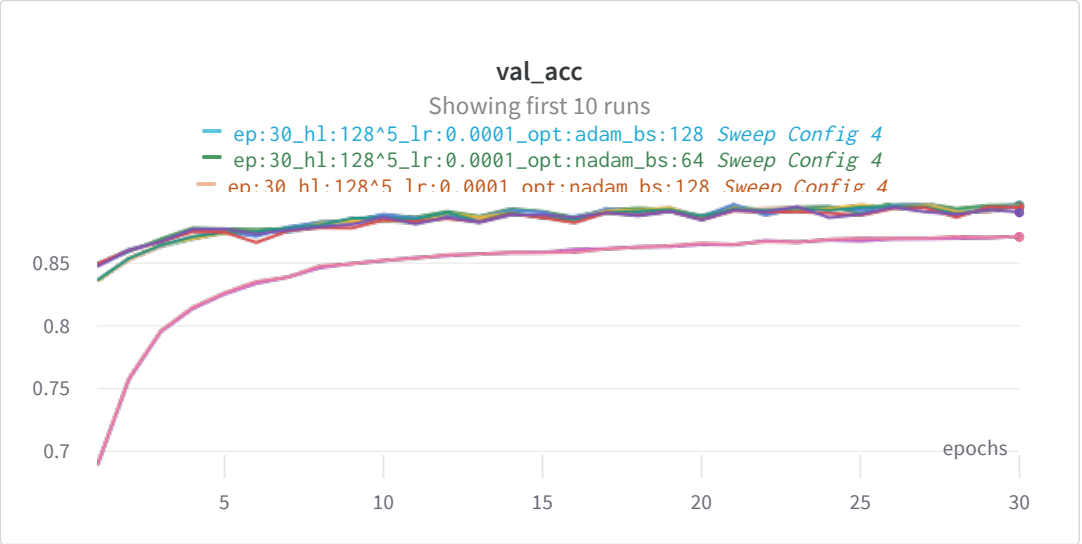
- As adam and nadam optimizers performed the best, I decided to tune 'beta1' and 'beta2' parameters along with weight decay and learning rate.
- I chose the best model from this sweep to perform one last sweep.

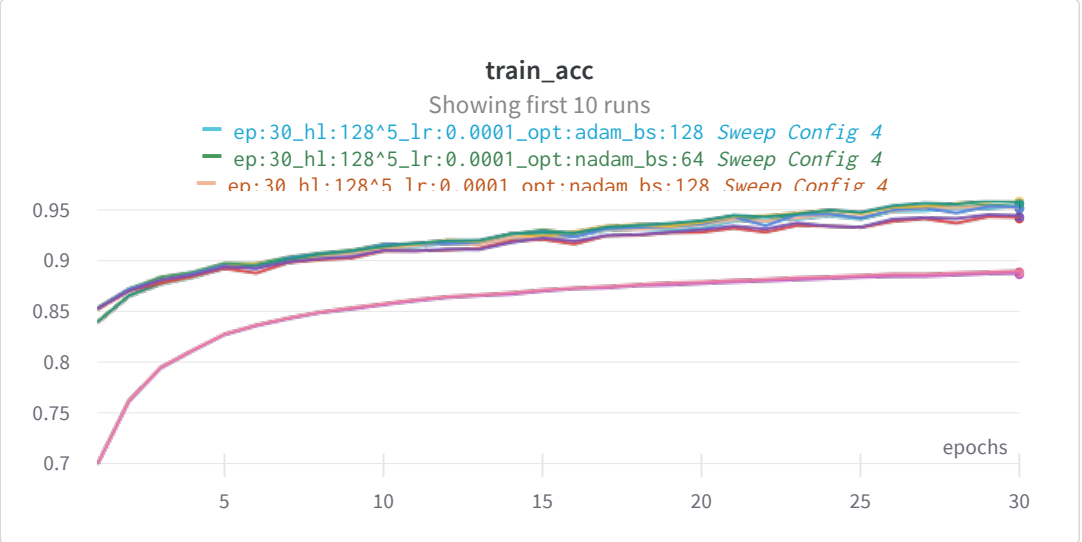
Sweep 4:

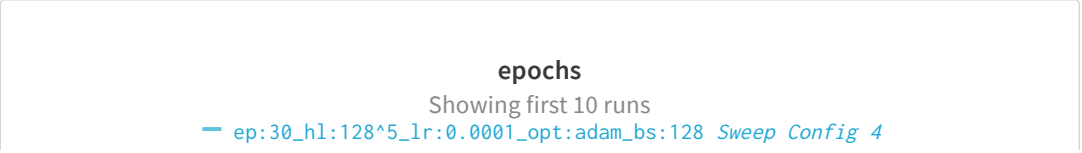
- In all the above cases, I chose a batch size of 64 as it did not lead to overflow errors and performed faster compared to batch sizes of 16 or 32.
- I ran a sweep with different batch sizes of 64 and 128 along with a few alternatives to learning rate to make up for the increased batch size. I also included another option for weight decay to give the model more space to improve.
- I chose the best model from this.

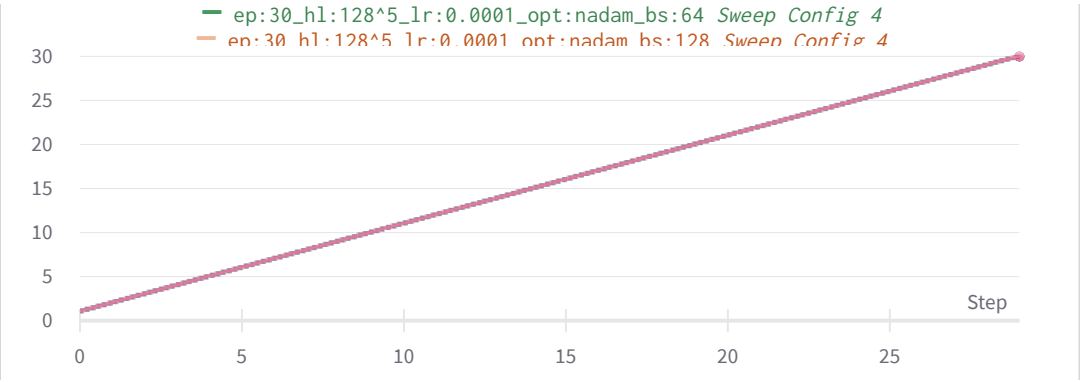
In all these sweeps, I had split the data randomly into train and validation sets. I then tuned the random seed of my dataset splitter to find a good validation accuracy. The results of this tuned model is

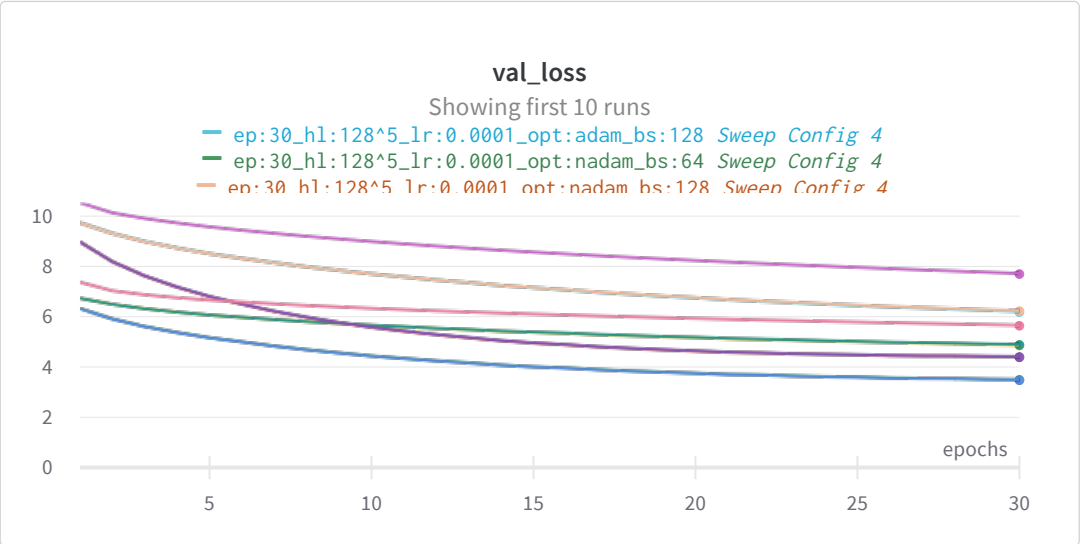
summarized in the following sections. The results of each sweep can be viewed separately by checking/unchecking the respective boxes.







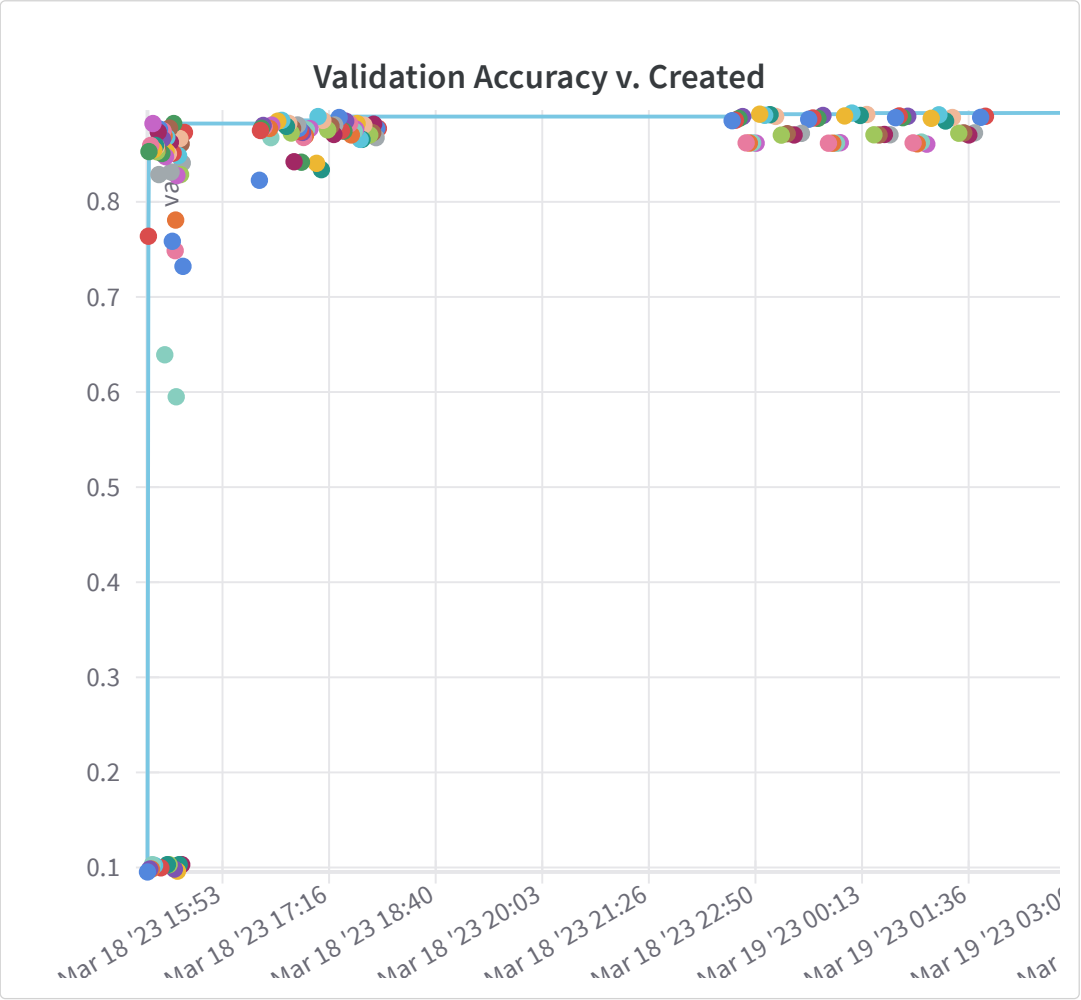

ep:30_hl:128^5_lr:0.0001_opt:nadam_bs:64 Sweep Config 4
ep:30_hl:128^5_lr:0.0001_opt:nadam_bs:128 Sweep Config 4
30
25
20
15
10
5
0
Step
0
5
10
15
20
25


val_loss
Showing first 10 runs
ep:30_hl:128^5_lr:0.0001_opt:adam_bs:128 Sweep Config 4
ep:30_hl:128^5_lr:0.0001_opt:nadam_bs:64 Sweep Config 4
ep:30_hl:128^5_lr:0.0001_opt:nadam_bs:128 Sweep Config 4
10
8
6
4
2
0
epochs
5
10
15
20
25
30

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.

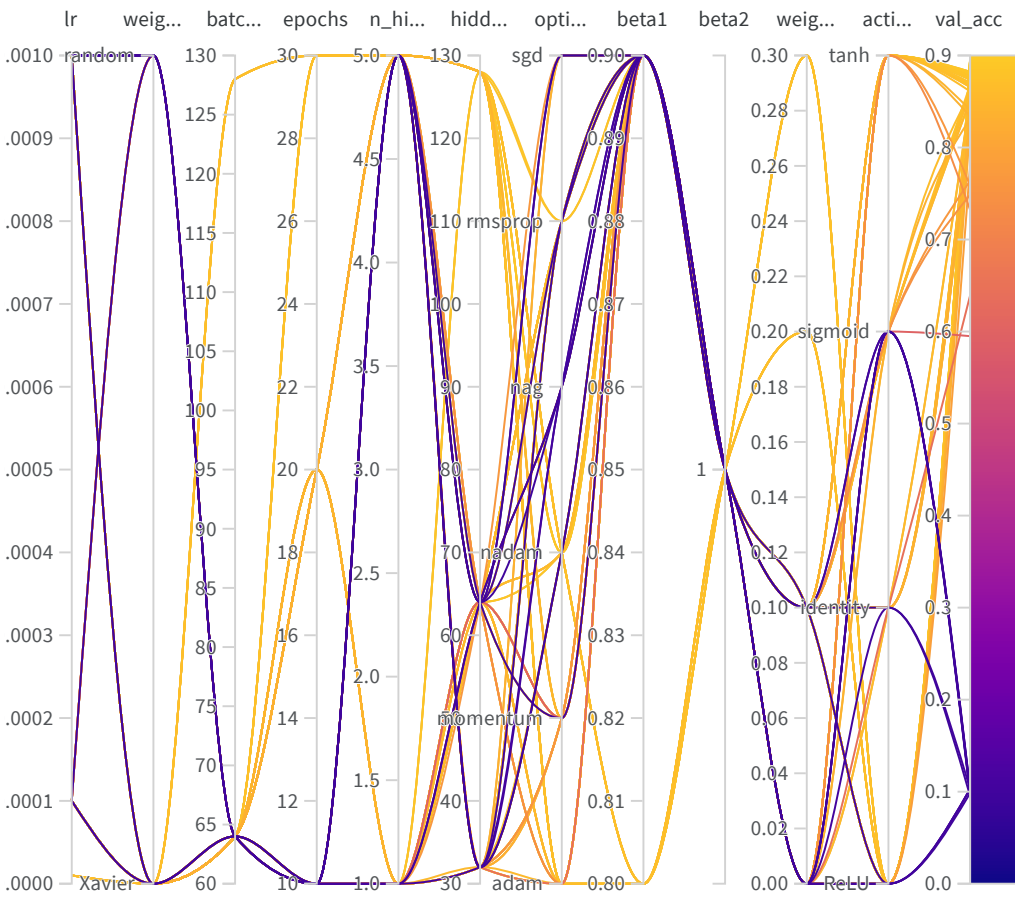
By looking at the plots that you get, write down some interesting observations (simple bullet points but should be insightful). You can also refer to the plot in Question 5 while writing these insights. For example, in the above sample plot there are many configurations which give less than 65% accuracy. 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: From the following plots and also from the trend I observed by examining in the sweep table, some observations I have made are:

- 'random' weight initialization typically results in lower validation accuracy.
- 'identity' and 'sigmoid' activations perform worse than 'tanh' and 'ReLU'.
- 'learning rate' is a very important hyperparameter to tune.
- shape of the neural network is important and it performs differently when paired with different learning rates and activation functions (deep neural networks perform better with larger learning rate and ReLU activation, while shallow neural networks work better with smaller learning rate and other activation functions as well).
- The above issue is related to the problem of vanishing gradients.

Note: (When all the sweep configurations are checked, it can be observed that (for instance), beta1 value of 0.9 has resulted in very poor performance. This is because sweep configuration 1 only took a value of 0.9 for beta1 and it resulted in many bad accuracies. To check the variation of beta1, the sweep where beta1 was tuned, i.e.



Parameter importance with respect to val_acc

Search

Parameters

1-10 of 20

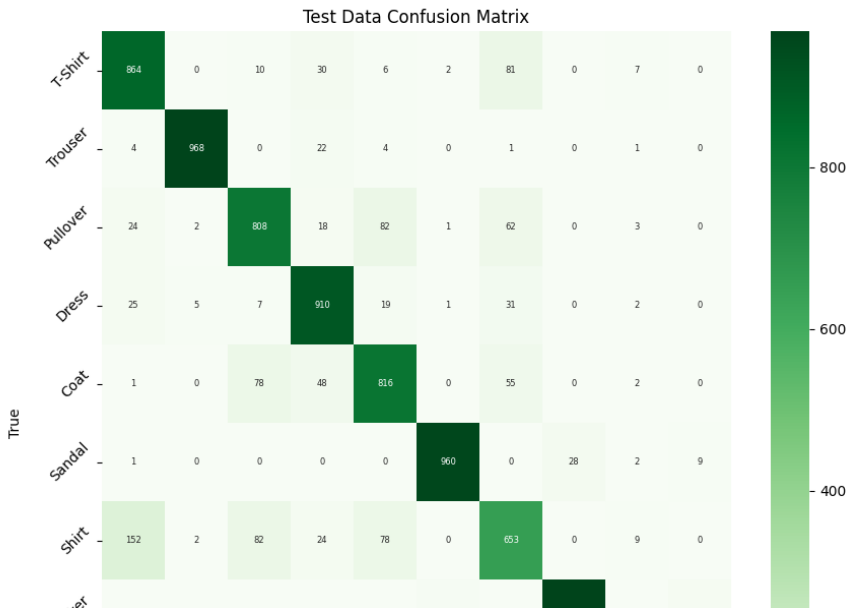
Config parameter	Importance	Correlation
n_hidden		
optimizer.value_adam		
weight_initialization.v…		
optimizer.value_nag		
activation.value_sigm…		
weight_initialization.v…		
lr		

activation.value_ReLU		
activation.value_tanh		
hidden_size		

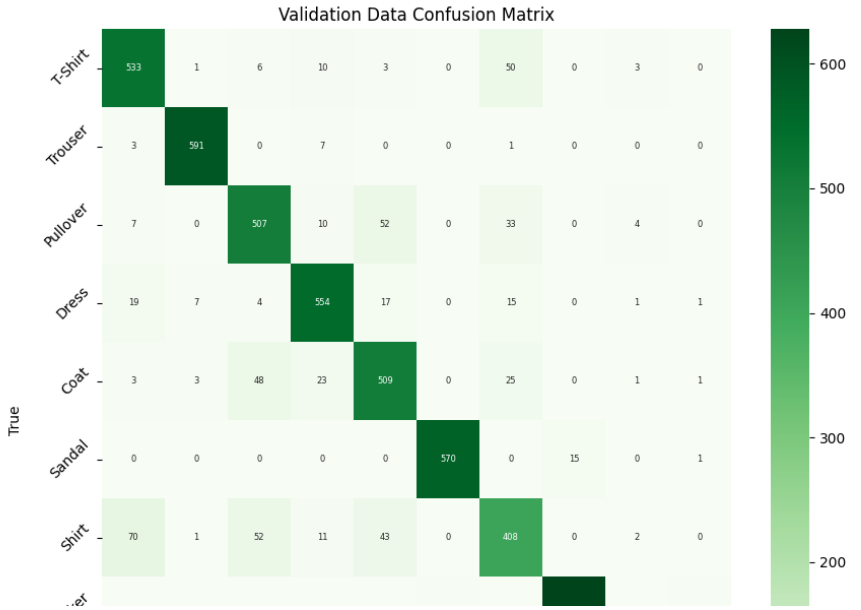
Question 7 (10 Marks)

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)

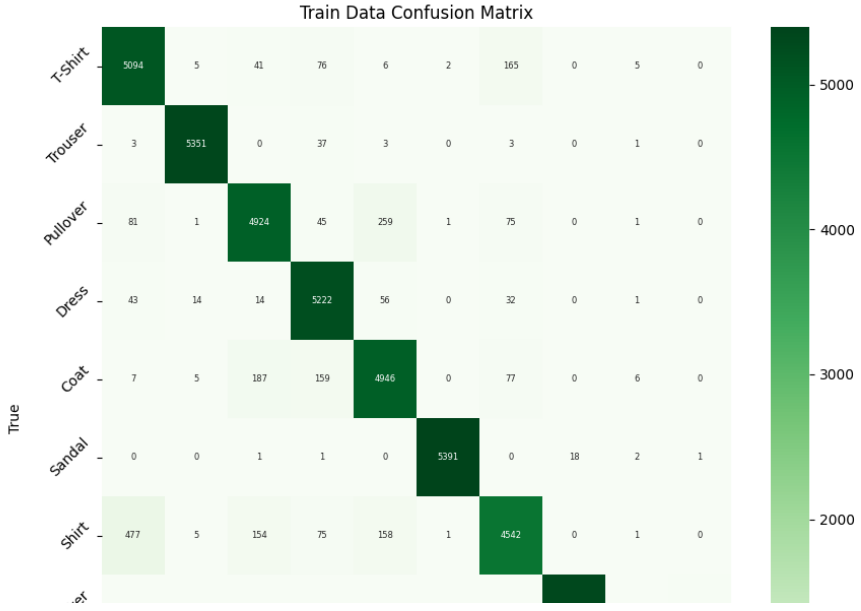
CM_test



CM_val



CM_train



Final Test Accuracy

final_model_CE

0.8851

Final Validation Accuracy

final_model_CE

0.8965

Final Train Accuracy

final_model_CE

0.9523

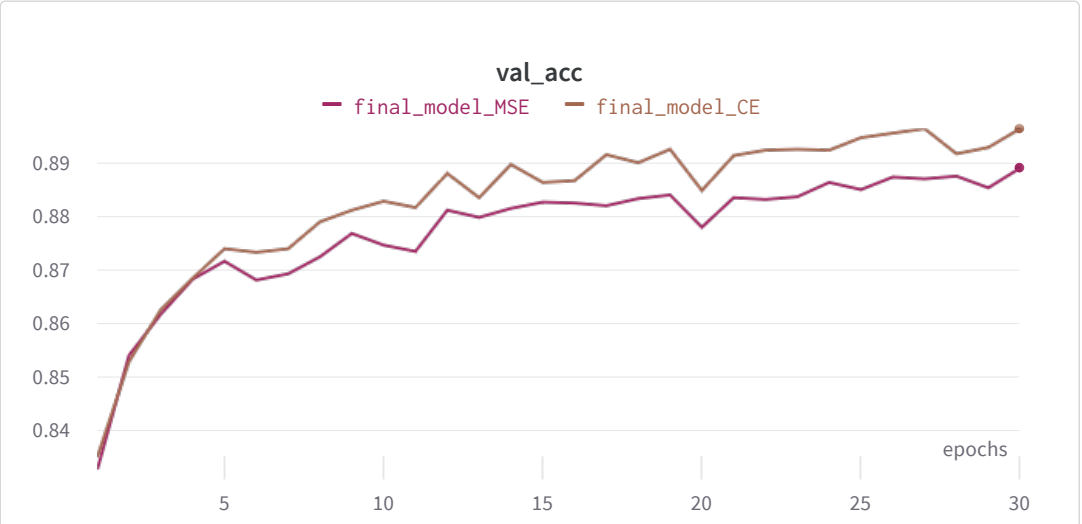
(0.9523, 0.9523)

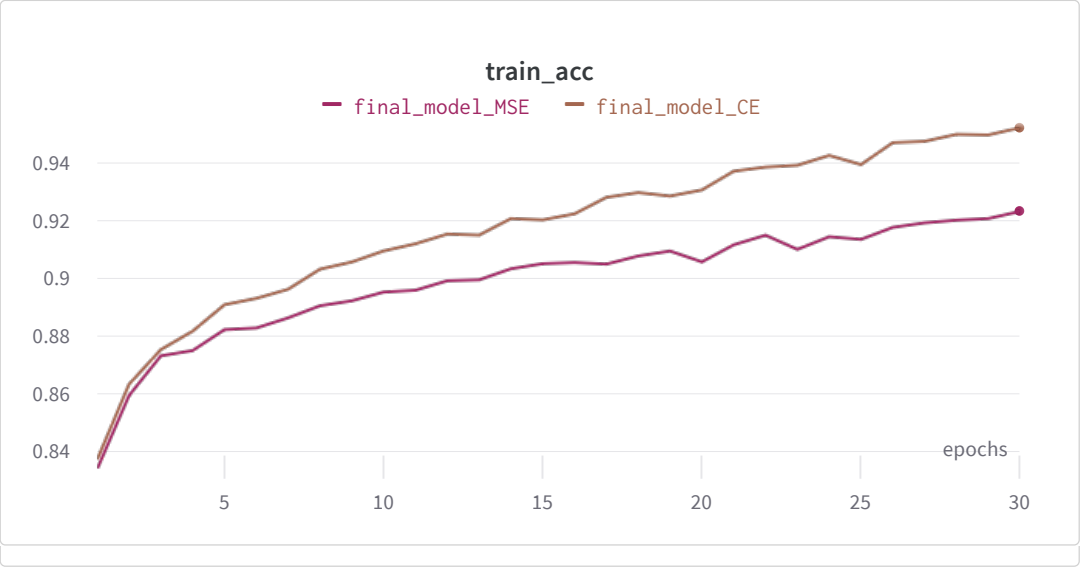
Question 8 (5 Marks)

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.

Answer: It can be observed that MSE results in lower validation and much lower test accuracy as compared to CE loss. The variation in this case with a neural network of shape 128^5 is low, because the a weight decay of 0.3 is used to reduce overfitting. This causes the weights to be low and hence the output of the network would be close to the range of 0-1.

However, if a shallower network is used with lower weight decay, the MSE model can be expected to perform much worse than the CE model.





Question 9 (10 Marks)

Paste a link to your github code for this assignment

Example: https://github.com/ShrungDN/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.

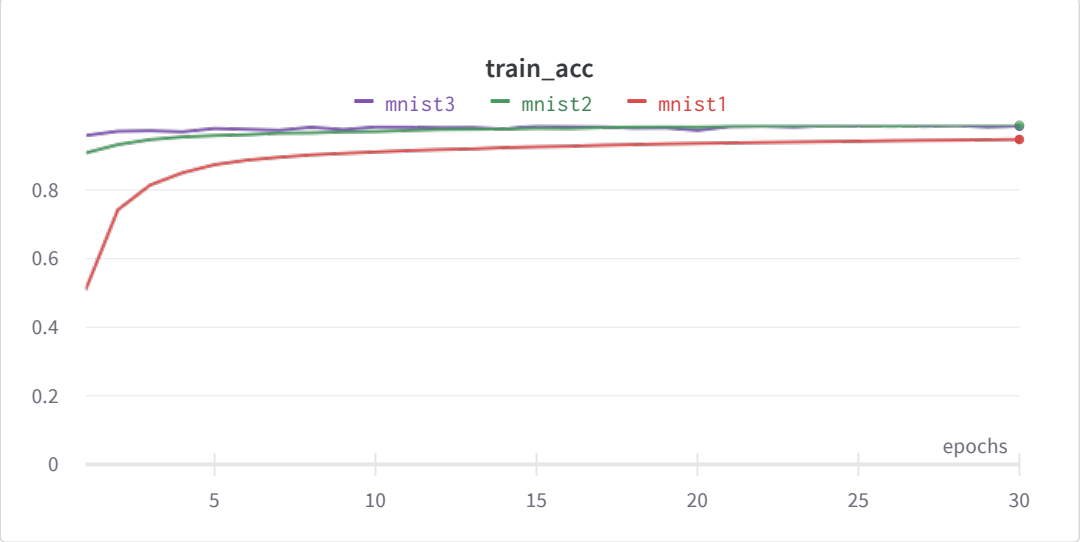
Answer:
Based on my earlier analysis, I would choose the best model that I found from the first three sweeps. In the parameter importance plot, when only sweep 4 is selected, it shows that the correlation the learning rate is very important. Hence, I would choose to train the model with 3 different learning rates and choose the model that gives the best result.

mnist1: lr = 0.00001

mnist2: lr = 0.0001

mnist3: lr = 0.001

It can be observed that the initial growth in accuracy largely depends on the learning rate. However, with the right optimizers, the final accuracy is around the same.



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
`-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.

Name	Default Value	Description

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, Shrung D N, ME19B168, 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.