Share Comment Star

CS6910 - Assignment 1

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

Ravish Kumar

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.

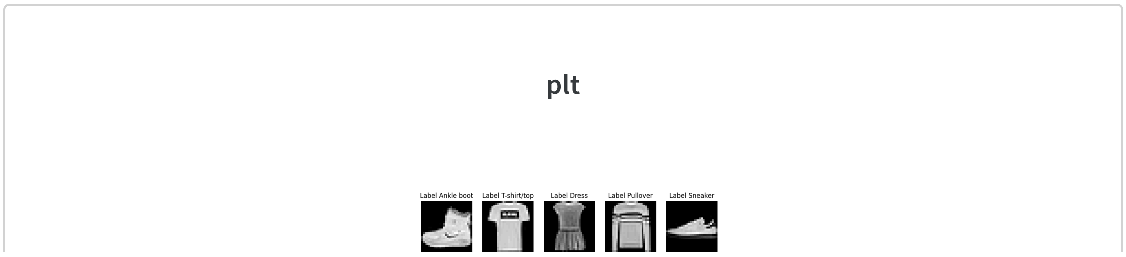
Problem Statement

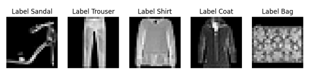
In this assignment you need to implement a feedforward neural network and write the backpropagation code for training the network. We strongly recommend using numpy for all matrix/vector operations. You are not allowed to use any automatic differentiation packages. This network will be trained and tested 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.





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.
github link :
https://github.com/ravishk17/DL_assignment1/blob/main/Q2.py

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.
github link: https://github.com/ravishk17/DL_assignment1

Question 4 (10 Marks)

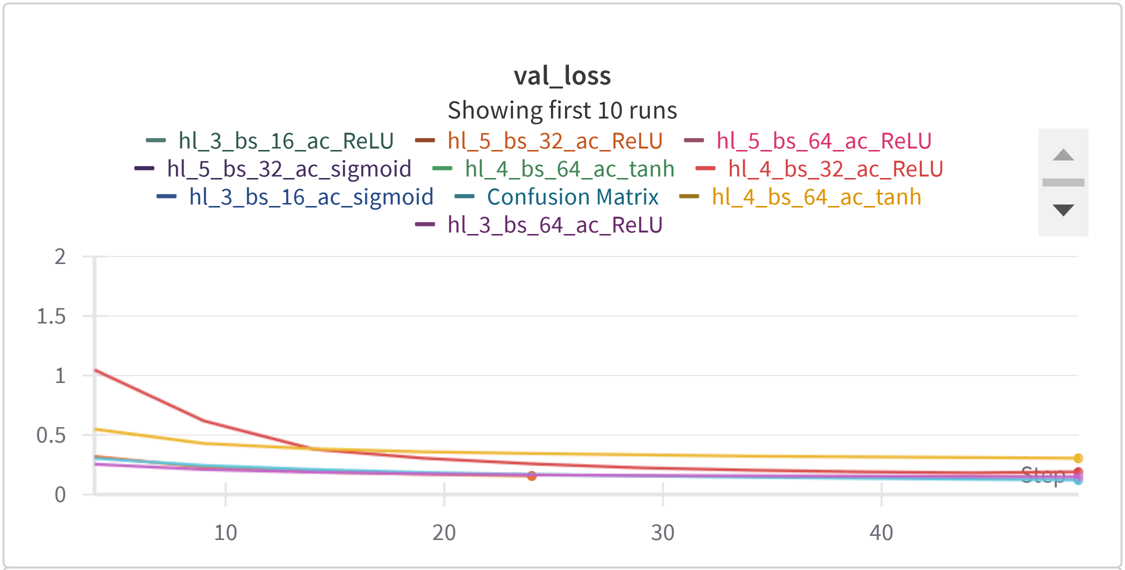
Use the sweep functionality provided by wandb to find the best values for the hyperparameters listed below. Use the standard train/test split of fashion_mnist (use `(X_train, y_train), (X_test, y_test) = fashion_mnist.load_data()`). Keep 10% of the training data aside as validation data for this hyperparameter 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.

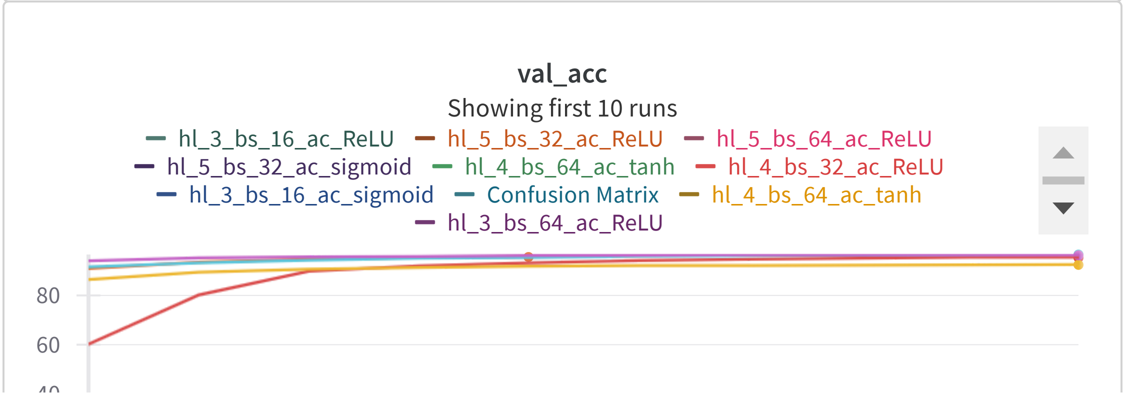
- number of epochs: 5, 10
- number of hidden layers: 3, 4, 5
- size of every hidden layer: 32, 64, 128
- weight decay (L2 regularisation): 0, 0.0005, 0.5
- learning rate: 1e-3, 1 e-4
- optimizer: sgd, momentum, nesterov, rmsprop, adam, nadam
- batch size: 16, 32, 64
- weight initialisation: random, Xavier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

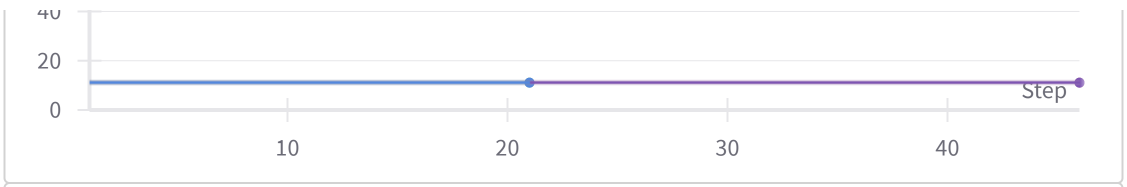
wandb will automatically generate the following plots. Paste these plots below using the "Add Panel to Report" feature. Make sure you use meaningful names for each sweep (e.g. hl_3_bs_16_ac_tanh to indicate that there were 3 hidden layers, batch size was 16 and activation function was ReLU) instead of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

I used bayesian sweep functionality to find the best configuration of hyperparameters for my model which maximizes model's accuracy. Bayesian sweep uses efficient approach informed by previous run. Once it finds a model with high accuracy, next time it changes one or two parameters and then checks the accuracy if it got increased from the previous run. I also tried random sweep, but that gave me multiple runs with accuracy less than 15% and some were above 70%, but on the other hand when I used bayesian, more number of runs yielded better accuracy around 80% and above and very less runs yielded lesser accuracy.

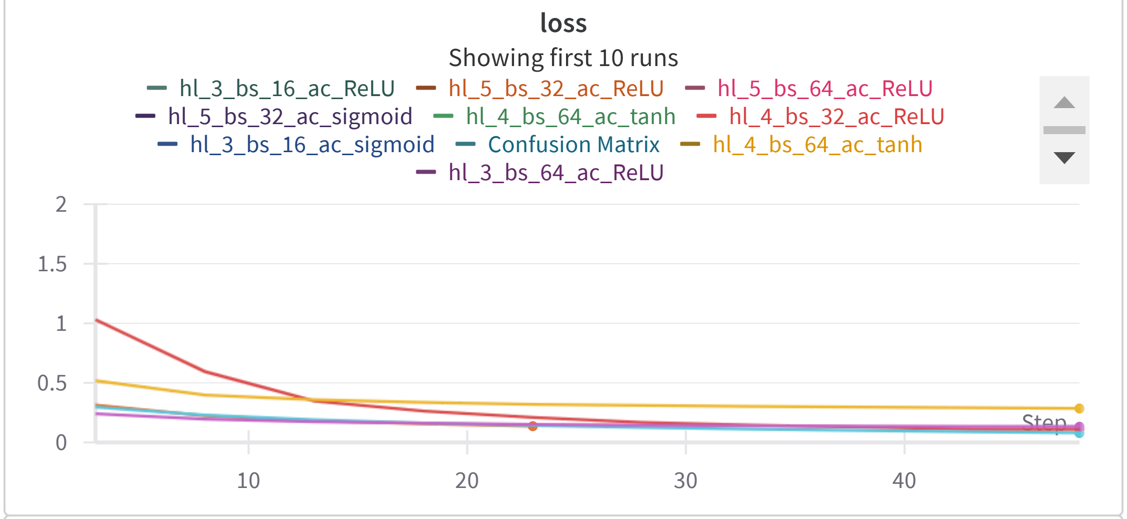
I ran multiple sweeps, which produced 35 different combinations of hyperparameters and below are the accuracy and losses.



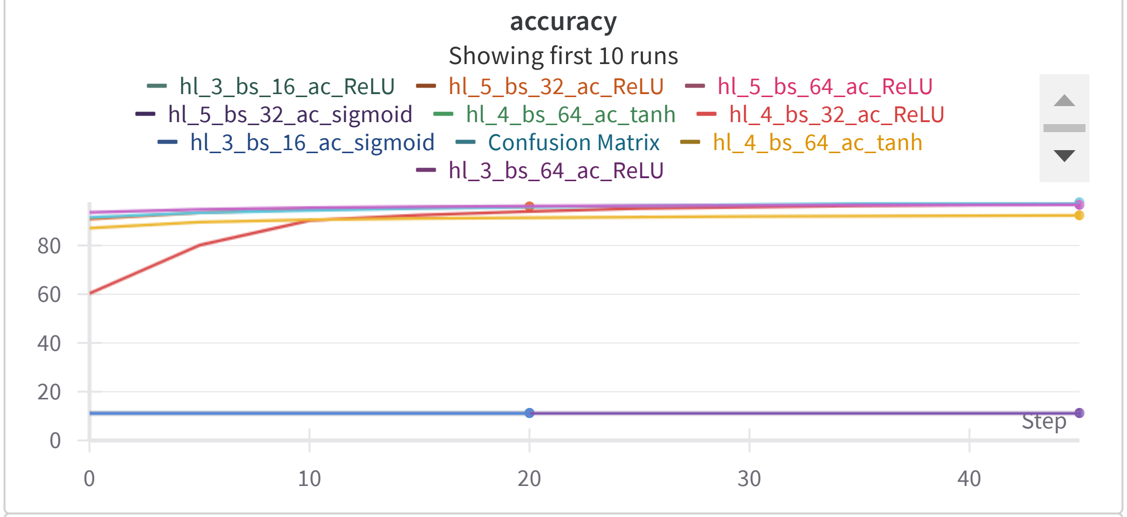




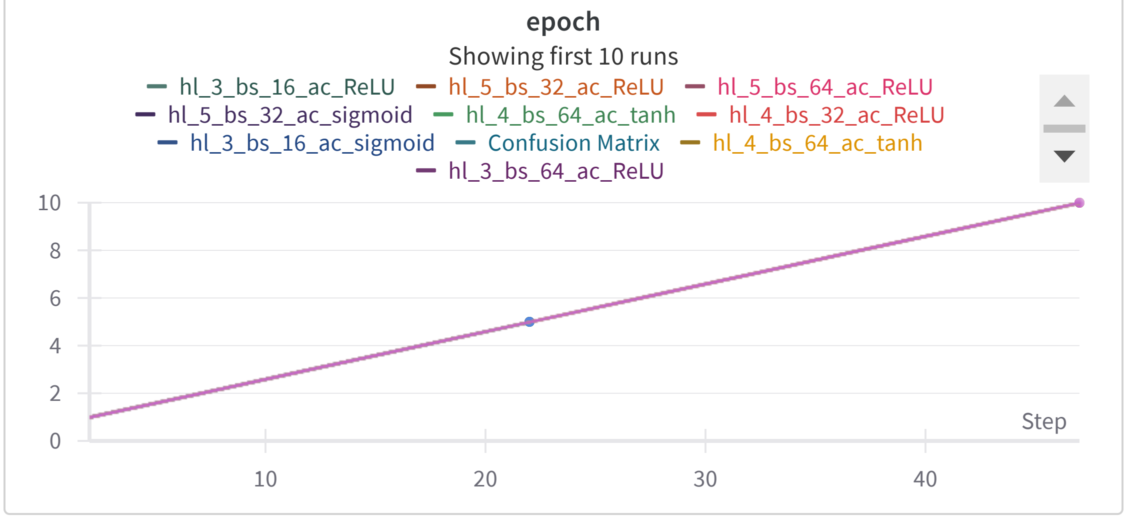
40
20
0
Step
10
20
30
40



loss
Showing first 10 runs
hl_3_bs_16_ac_ReLU
hl_5_bs_32_ac_ReLU
hl_5_bs_64_ac_ReLU
hl_5_bs_32_ac_sigmoid
hl_4_bs_64_ac_tanh
hl_4_bs_32_ac_ReLU
hl_3_bs_16_ac_sigmoid
Confusion Matrix
hl_4_bs_64_ac_tanh
hl_3_bs_64_ac_ReLU
2
1.5
1
0.5
0
Step
10
20
30
40



accuracy
Showing first 10 runs
hl_3_bs_16_ac_ReLU
hl_5_bs_32_ac_ReLU
hl_5_bs_64_ac_ReLU
hl_5_bs_32_ac_sigmoid
hl_4_bs_64_ac_tanh
hl_4_bs_32_ac_ReLU
hl_3_bs_16_ac_sigmoid
Confusion Matrix
hl_4_bs_64_ac_tanh
hl_3_bs_64_ac_ReLU
80
60
40
20
0
Step
0
10
20
30
40

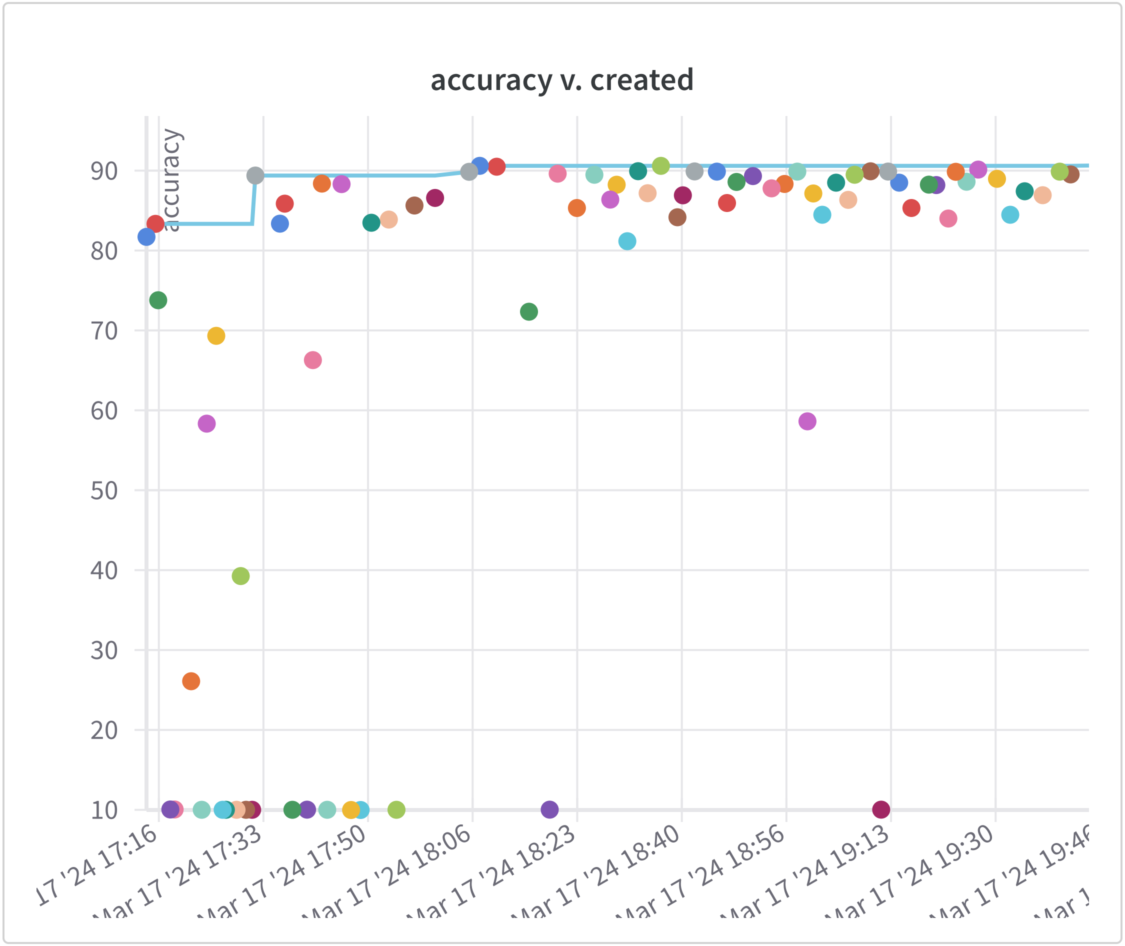


epoch
Showing first 10 runs
hl_3_bs_16_ac_ReLU
hl_5_bs_32_ac_ReLU
hl_5_bs_64_ac_ReLU
hl_5_bs_32_ac_sigmoid
hl_4_bs_64_ac_tanh
hl_4_bs_32_ac_ReLU
hl_3_bs_16_ac_sigmoid
Confusion Matrix
hl_4_bs_64_ac_tanh
hl_3_bs_64_ac_ReLU
10
8
6
4
2
0
Step
10
20
30
40

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



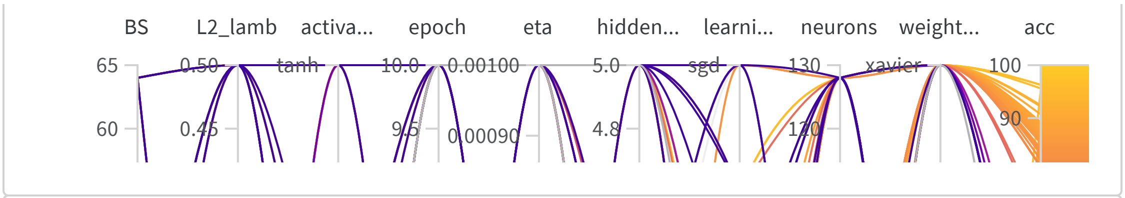
Question 6 (20 Marks)

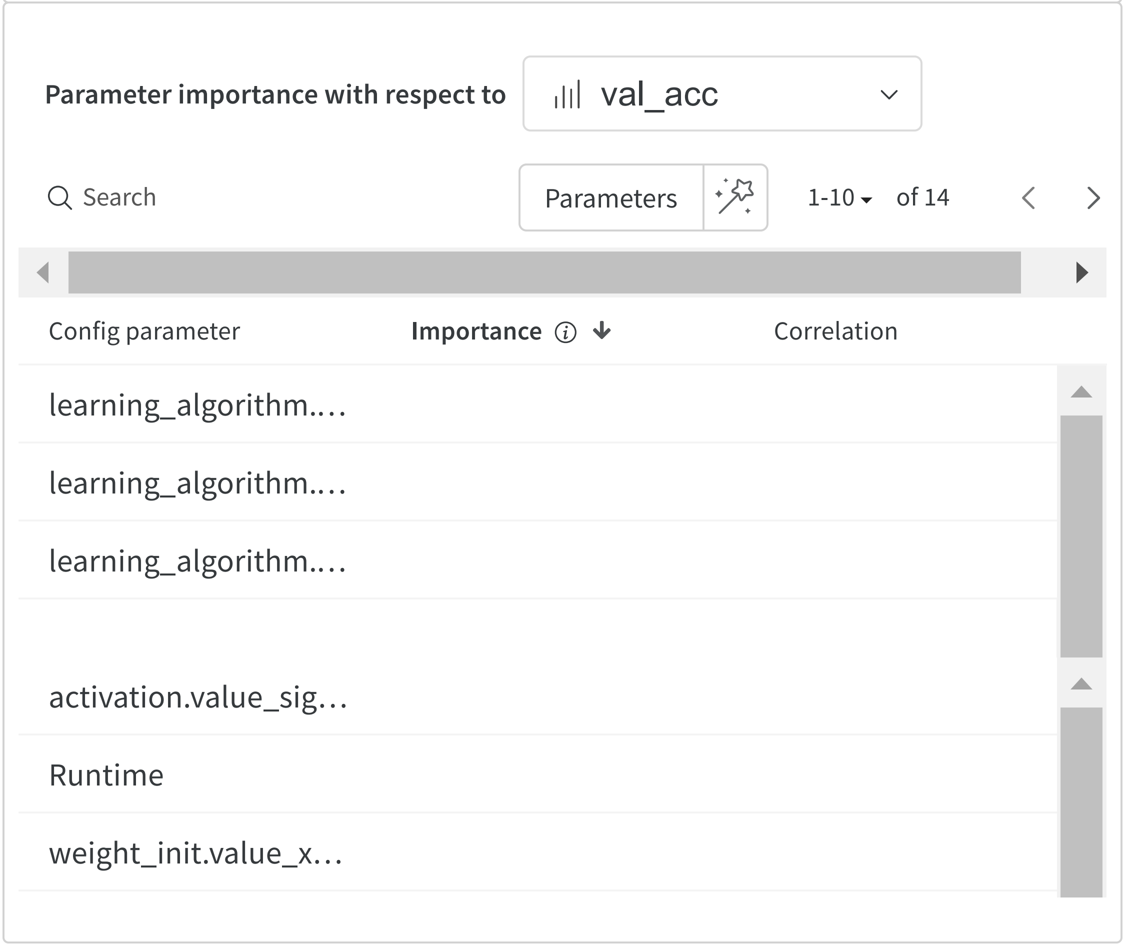
Parallel coordinates plots provide a visual representation of how hyperparameter values relate to model metrics, specifically accuracy in this case. By examining these plots, I can identify which combinations of parameters lead to better accuracy performance. For instance, when using optimizers like momentum and sgd, along with a squared-error loss function and tanh activation, the resulting accuracy tends to be low ($\leq$ 30%).On the other hand, I expect that a combination of Nadam or Adam optimizers, cross-entropy loss, ReLU activation, no regularization, and a learning rate around 0.001 would yield an accuracy greater than 90%.

Following are the hyperparameter configuration for my best model:

- optimizer - adam
- activation - ReLU
- eta - 0 .001
- number of neurons in hidden layer - 128
- number of hidden layer - 3
- weight initialization - xavier
- epochs -10
- batch size - 16

In order to achieve close to 95% accuracy, I would suggest few changes in the configuration like increasing epochs, number of hidden layers and tuning the learning rate





Parameter importance with respect to val_acc

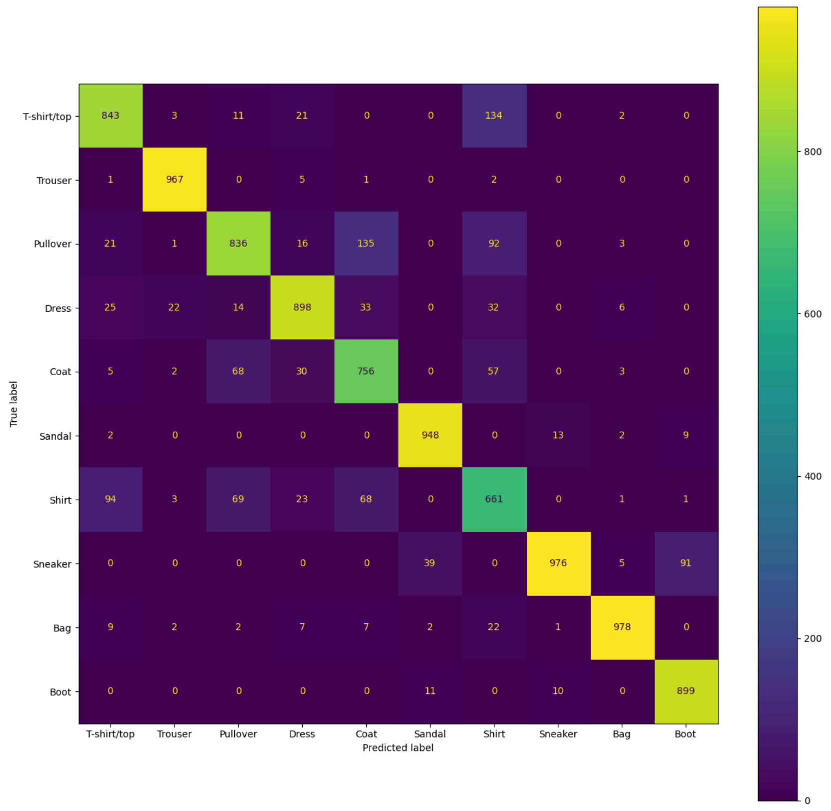
Config parameter	Importance	Correlation
learning_algorithm.…		
learning_algorithm.…		
learning_algorithm.…		
activation.value_sig…		
Runtime		
weight_init.value_x…		

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)

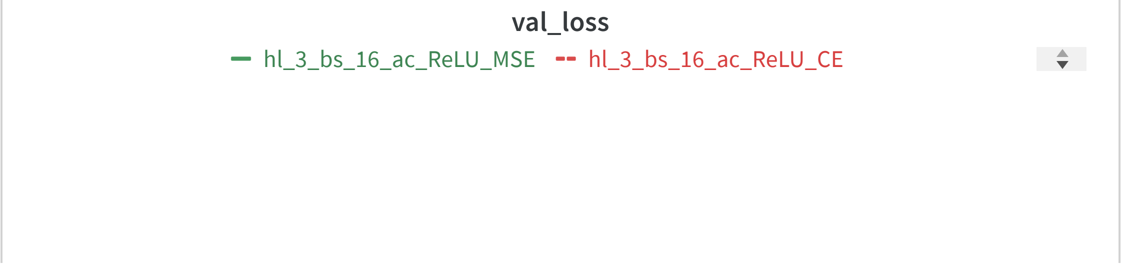
Accuracy: 86.16667%

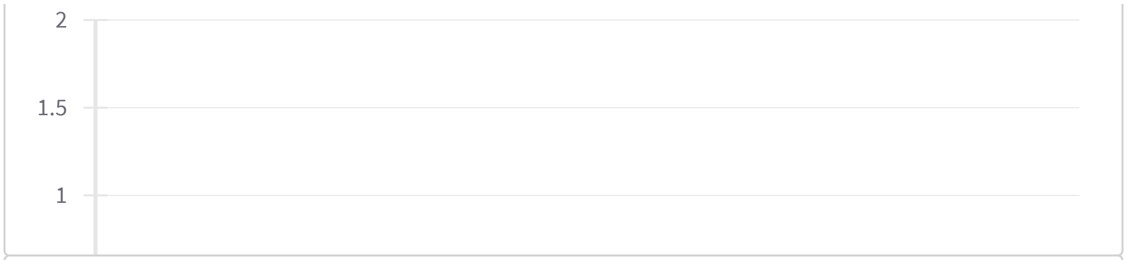
confusion_matrix



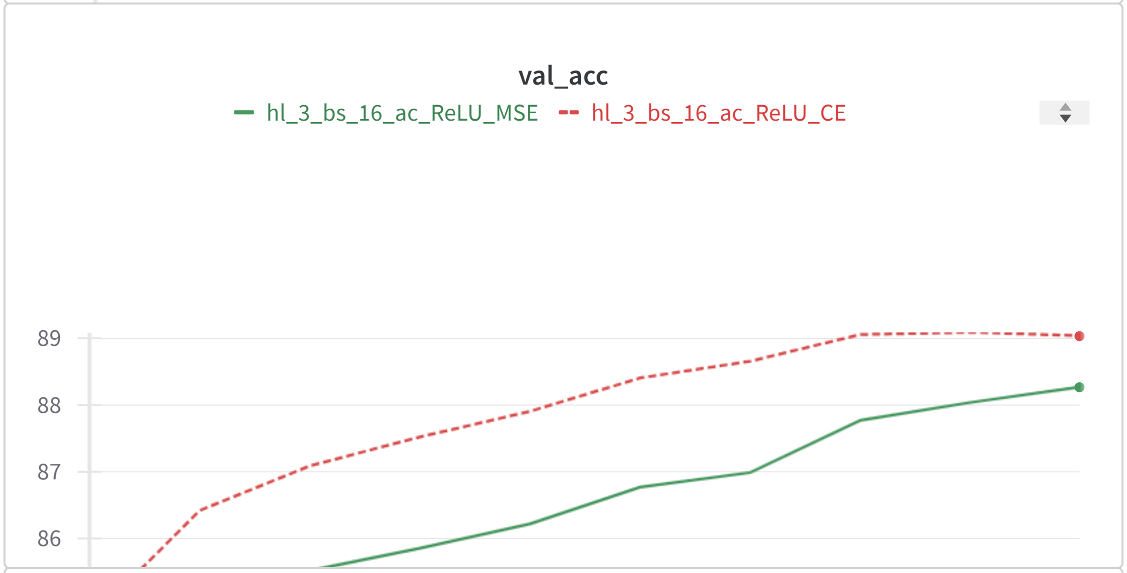
Question 8 (5 Marks)

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.



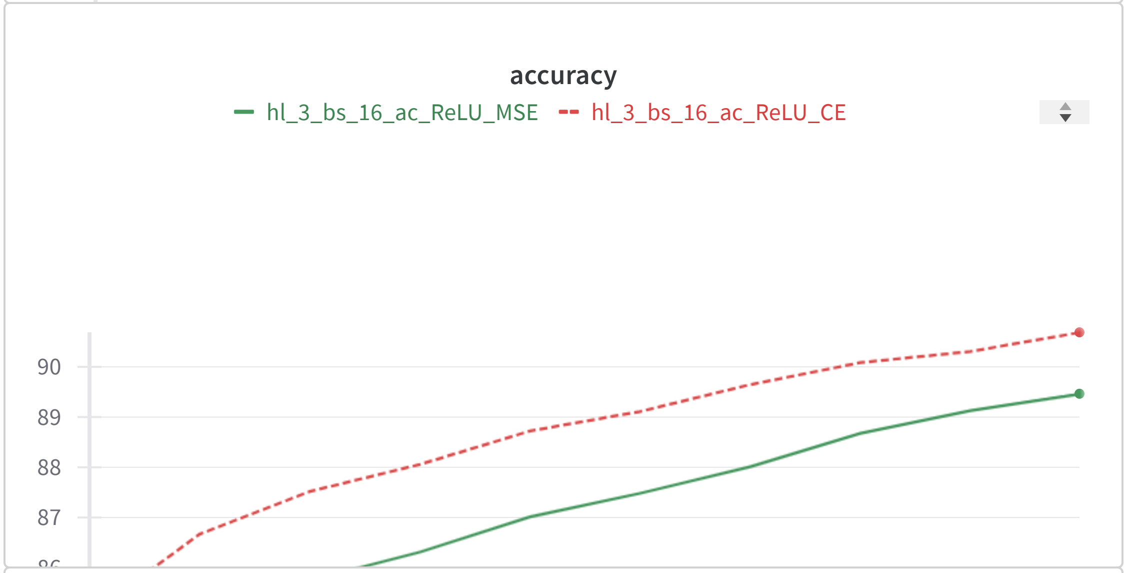


2
1.5
1

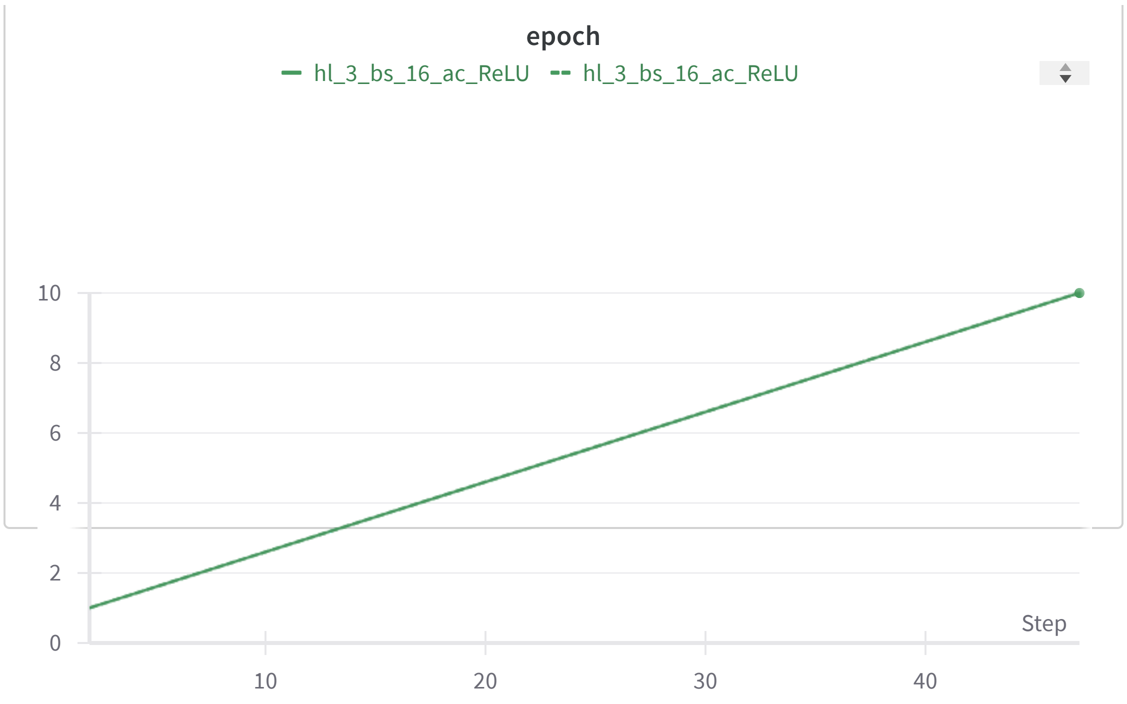


val_acc
hl_3_bs_16_ac_ReLU_MSE
hl_3_bs_16_ac_ReLU_CE
89
88
87
86

loss
hl_3_bs_16_ac_ReLU_MSE
hl_3_bs_16_ac_ReLU_CE
2
1.5
1



accuracy
hl_3_bs_16_ac_ReLU_MSE
hl_3_bs_16_ac_ReLU_CE
90
89
88
87



Question 9 (10 Marks)

Paste a link to your github code for this assignment

Example: https://github.com/<user-id>/cs6910_assignment1;

- 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

github link: https://github.com/ravishk17/DL_assignment1

Question 10 (10 Marks)

Batch size - 64 ; L2_lamb - 0.0005; activation : tanh; epoch - 10; eta - 0.001; hidden layers - 4; learning algo - adam; neurons - 128; loss - cross-entropy; weight init - xavier; accuracy - 96.61667

Batch size - 64 ; L2_lamb - 0.5; activation : ReLU; epoch - 10; eta - 0.001; hidden layers - 3; learning algo - rmsprop; neurons - 128; loss - cross-entropy; weight init - xavier; accuracy - 96.28333
Batch size - 32; L2_lamb - 0.0005; activation - ReLU; epoch - 10; eta - 0.001; hidden layers - 4; learning algo - adam; neurons - 32; loss - cross entropy; weight init - random; accuracy - 95.28333

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 myprojectn
```

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.

Name	Default Value	Description
`-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, Ravish Kumar CS23M055, 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.