Edit Share

Assignment 1

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

Pranab Kumar Rout, Pratyush Dash

CS21M045 , CS21M046

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
- We strongly recommend that you work on this assignment in a team of size 2. Both the members of the team are expected to work together (in a subsequent viva both members will be expected to answer questions, explain the code, etc).
- Collaborations and discussions with other groups are 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 can run the code in a jupyter notebook on colab by enabling GPUs.
- 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.

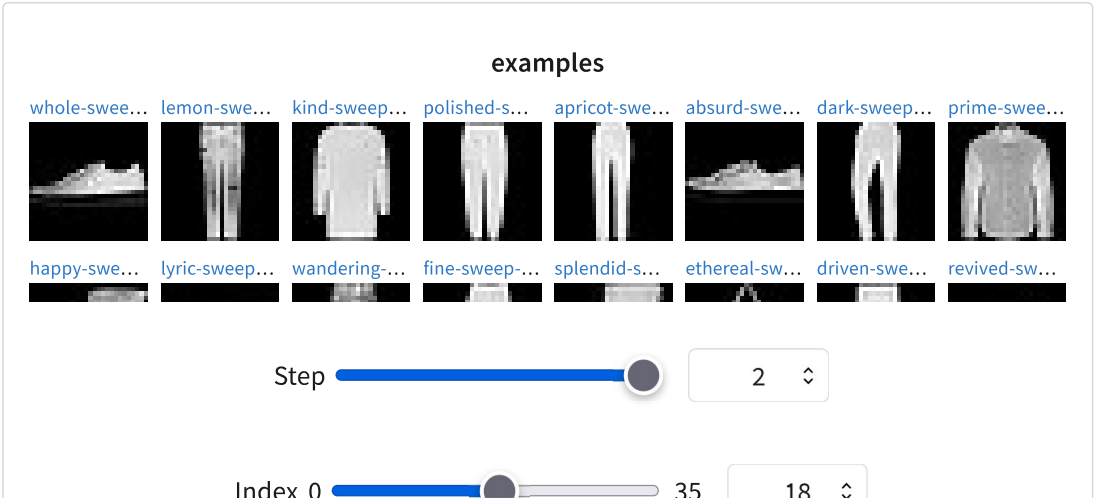
- You also need to provide a link to your github code as shown below. Follow good software engineering practices and set up a github repo for the project on Day 1. Please do not write all code on your local machine and push everything to github on the last day. The commits in github should reflect how 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.

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 so that it is easy to change the number of hidden layers and the number of neurons in each hidden layer.

We will check the code for implementation and ease of use.

Ans: The code is uploaded on GitHub

Question 3 (18 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.

Ans: The code is uploaded on GitHub

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.

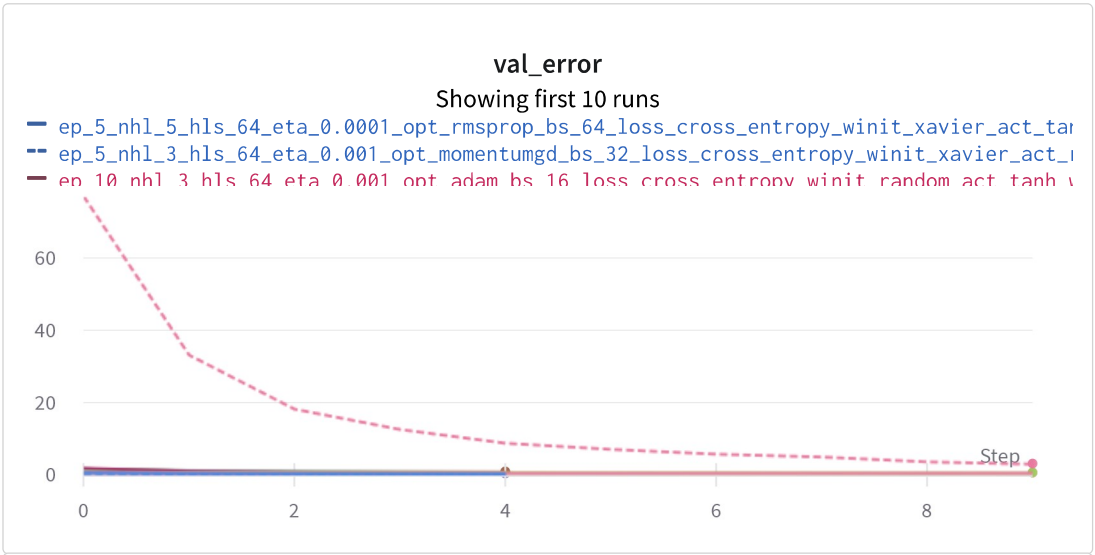
- number of epochs: 5, 10
- number of hidden layers: 3, 4, 5
- size of every hidden layer: 32, 64, 128
- weight decay (L2 regularisation): 0, 0.0005, 0.5
- learning rate: 1e-3, 1 e-4
- optimizer: sgd, momentum, nesterov, rmsprop, adam, nadam
- batch size: 16, 32, 64
- weight initialisation: random, Xavier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

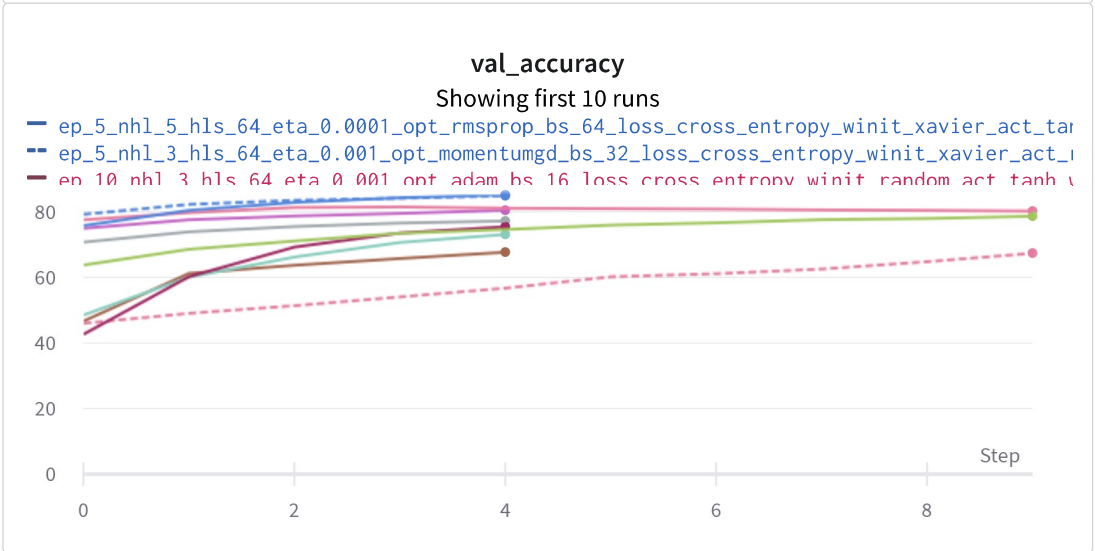
wandb will automatically generate the following plots. Paste these plots below using the "Add Panel to Report" feature. Make sure you use meaningful names for each sweep (e.g. hl_3_bs_16_ac_tanh to indicate that there were 3 hidden layers, batch size was 16 and activation function was ReLU) instead of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

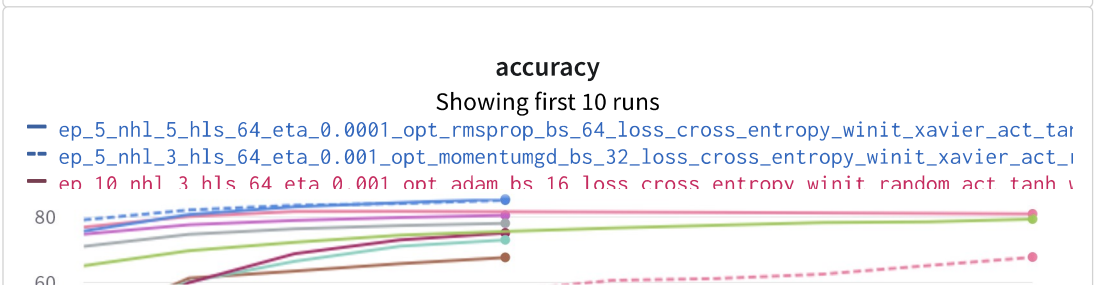
We have three hyperparameter searching strategies provided by

wandb. Grid, Random, and Bayes. The grid does a brute force job and tries all possible combination. Its time taking and also need heavy computation. So inefficient. The Bayes used a normal distribution to model the function and then chooses parameters. But, this strategy does not scale for a large number of hyperparameters.

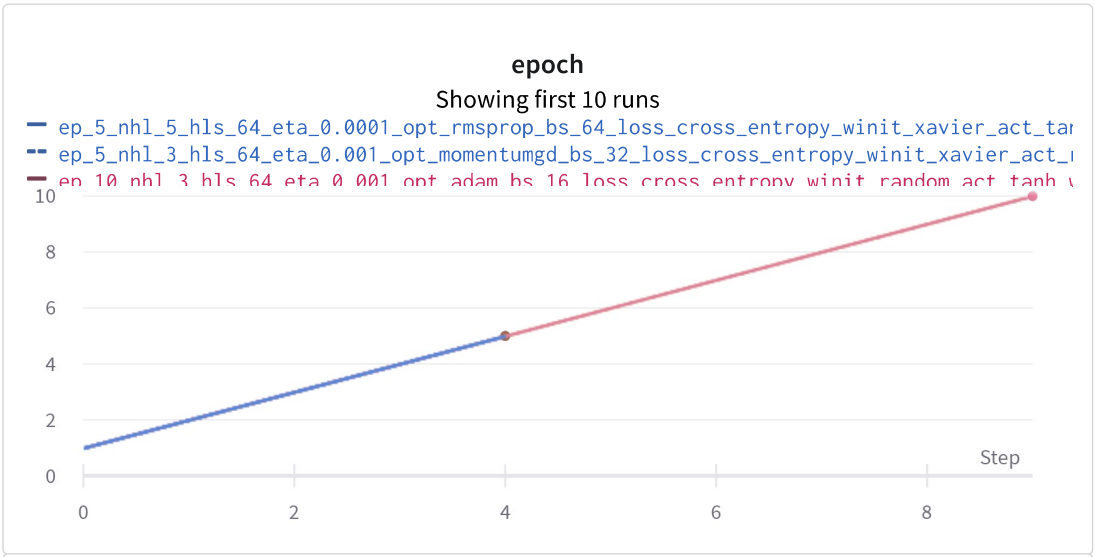
So we use the Random. It takes combination randomly and not all. Its efficient and less time consuming.

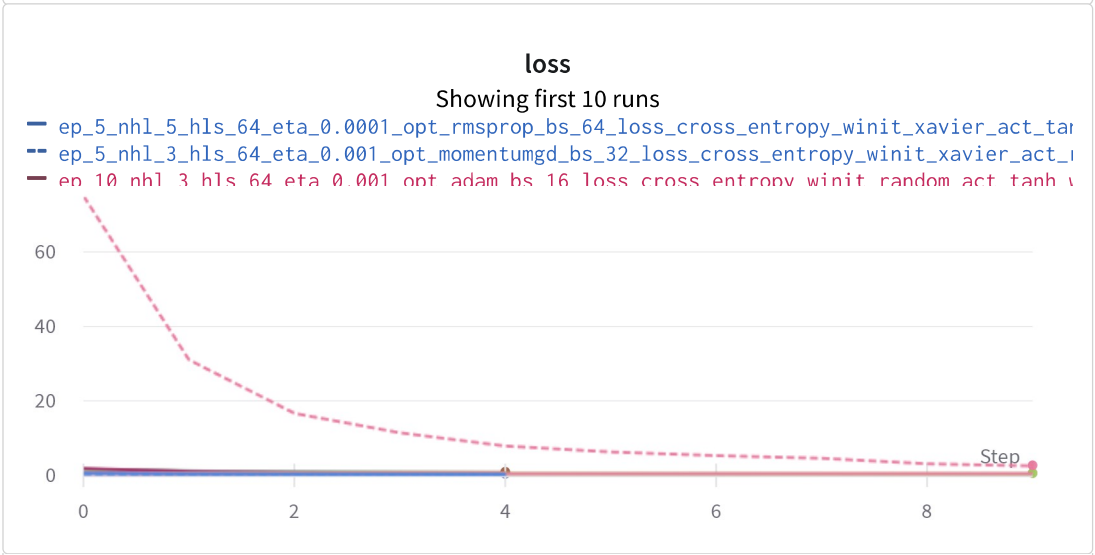


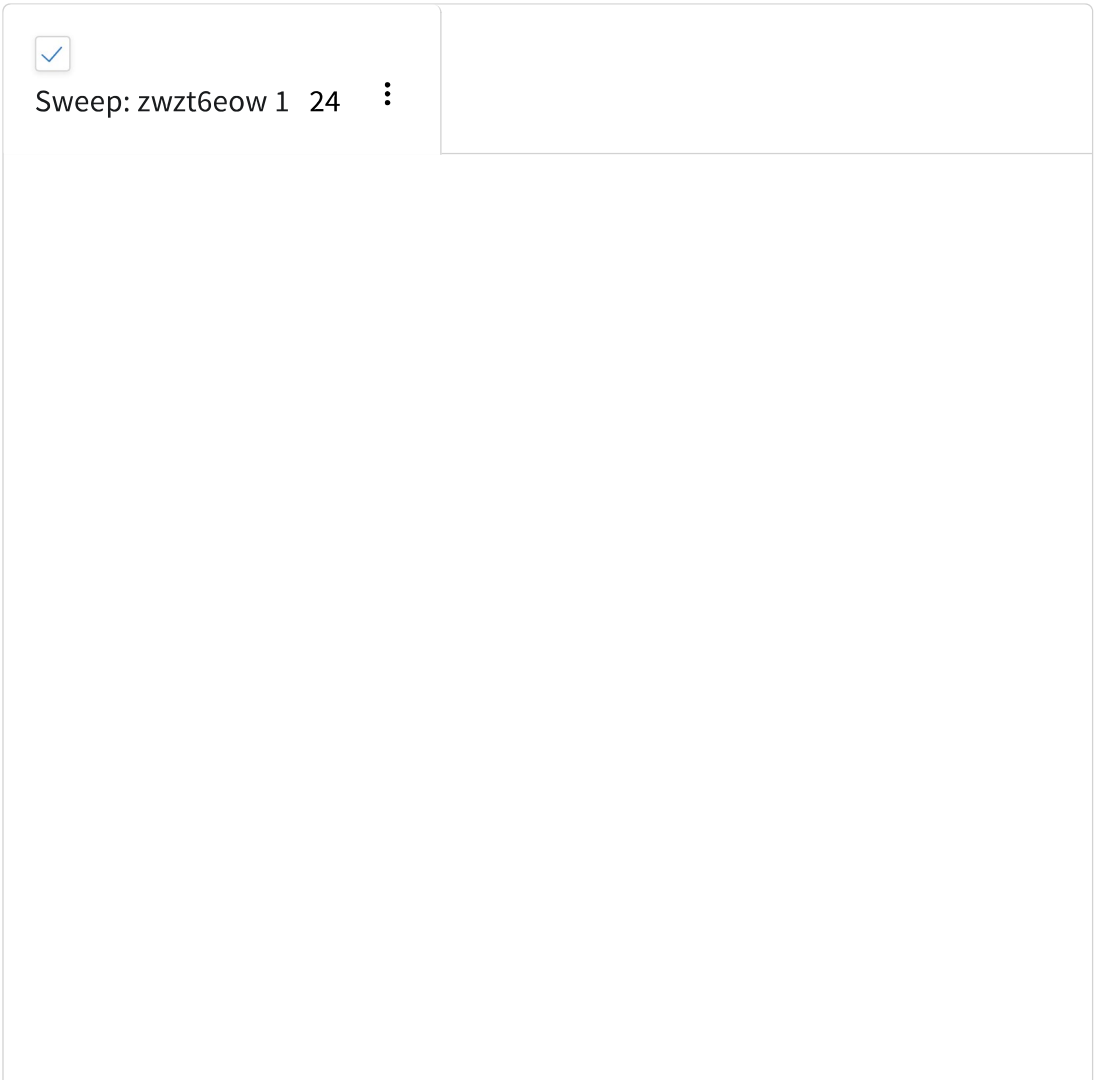





40
20
0
Step
0
2
4
6
8


epoch
Showing first 10 runs
ep_5_nhl_5_hls_64_eta_0.0001_opt_rmsprop_bs_64_loss_cross_entropy_winit_xavier_act_tar
ep_5_nhl_3_hls_64_eta_0.001_opt_momentumgd_bs_32_loss_cross_entropy_winit_xavier_act_
ep_10_nhl_3_hls_64_eta_0.001_opt_adam_bs_16_loss_cross_entropy_winit_random_act_tanh_
10
8
6
4
2
0
Step
0
2
4
6
8

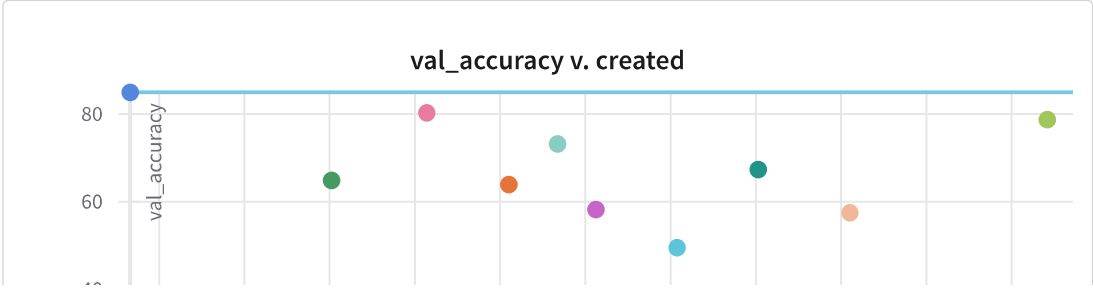

loss
Showing first 10 runs
ep_5_nhl_5_hls_64_eta_0.0001_opt_rmsprop_bs_64_loss_cross_entropy_winit_xavier_act_tar
ep_5_nhl_3_hls_64_eta_0.001_opt_momentumgd_bs_32_loss_cross_entropy_winit_xavier_act_
ep_10_nhl_3_hls_64_eta_0.001_opt_adam_bs_16_loss_cross_entropy_winit_random_act_tanh_
60
40
20
0
Step
0
2
4
6
8

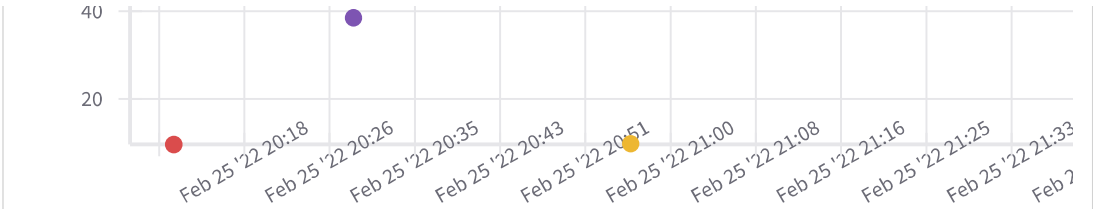

Sweep: zwzt6eow 1 24

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





Sweep: zwzt6eow 1 27

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.

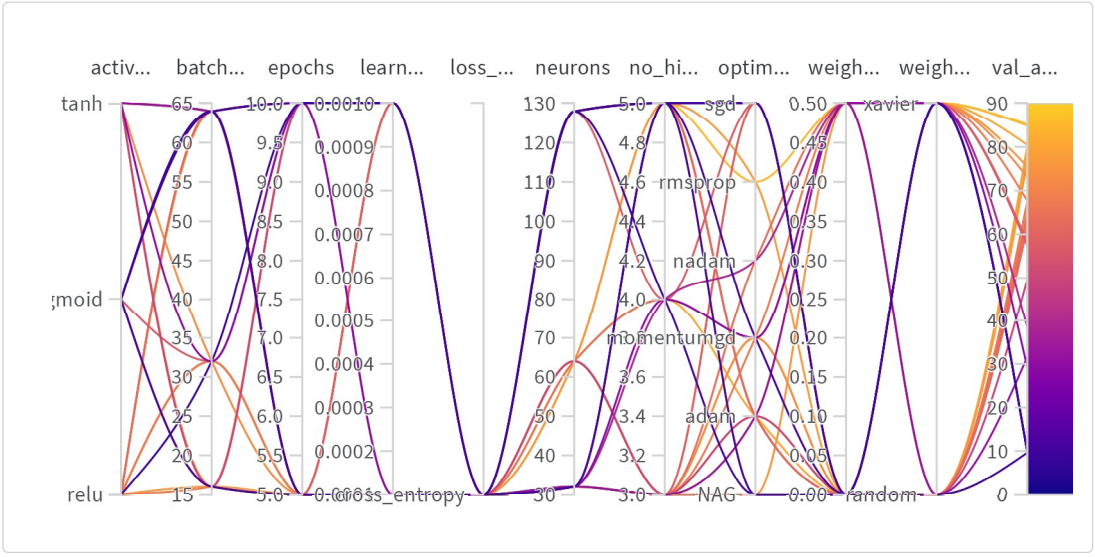
Here again, wandb automatically generates a "Parallel co-ordinates plot" and a "correlation summary" as shown below. Learn about a "Parallel co-ordinates plot" and how to read it.

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.

Observations

- The RMSprop optimizer and the Relu & activation function worked quite well with accuracy on a bit higher side. But tanh activation with Rmsprop gave the hoghest validation accuracy
- Relu activation performed well because the gradient is always a constant which helps it to reduce the impact of vanishing gradients. In case of greater number of layers this is helpful
- We got the highest validation accuracy of 85.504 for the following configuration -- Number of epochs: 5, Number of hidden layers: 5, Size of each hidden layer: 64, Learning rate: 0.0001, Optimizer: rmsprop, Batch size: 64, Weight Initialization: Xavier, Activation: Tanh, Loss: cross_entropy
- L2 decay gave us better results as compared to normal
- Xavier initialization was also can be seen as better than random initialization
- Since the outputs are probability values, using cross_entropy is a better choice.



Sweep: zwzt6eow 1 27

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
The best model in our experiment gives the accuracy of 85.2% and has the configuration as -

Number of epochs: 5

Number of hidden layers: 5

Size of each hidden layer: 64

Learning rate: 0.0001

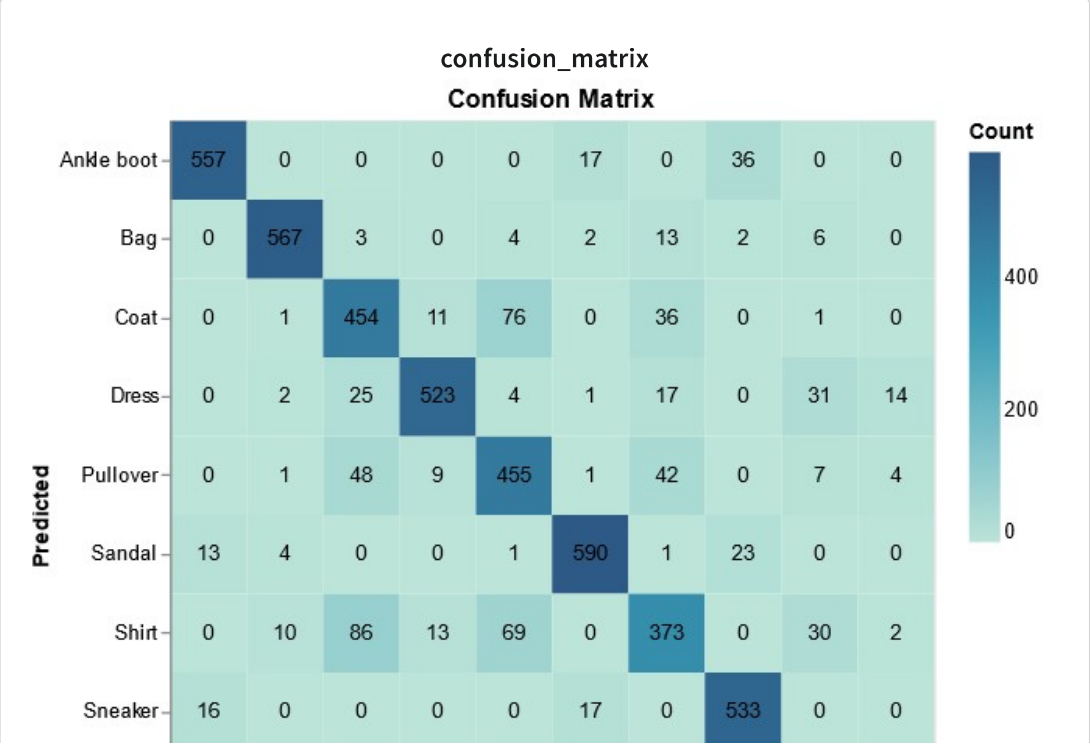
Optimizer: rmsprop

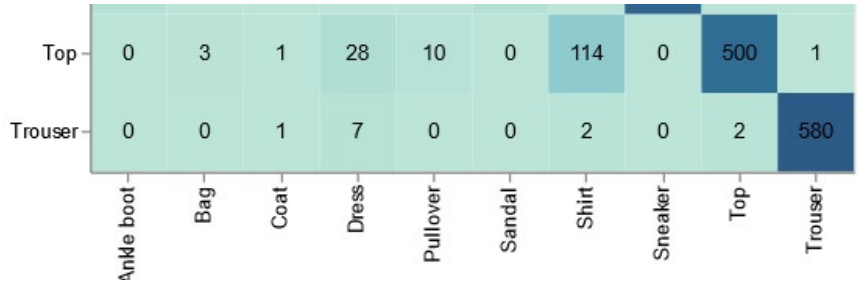
Batch size: 64

Weight Initialization: xavier

Activation: tanh

Loss: cross_entropy





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.A

ANSWER:

Yes, as its a classification problem, the use of MSE doesn't make sense as we are using non linear functions like sigmoid and Tanh. We have used the cross entropy loss. The MSE was giving out poor results on same same hyperparameter configuration.

Question 9 (10 Marks)

Paste a link to your github code for this assignment

Link: https://github.com/pratyush-dash/Deep-Learning-Assignment-1/

Wandb Report: https://wandb.ai/pandp/Ass1_sweep/reports/Assignment-1--VmlldzoxNjE0OTU4

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.

Answers:

Based on the experiments, we have concluded few things. The proper Weight initialization helps in converging faster. We got very good results in xavier initialization. The scaling of data sets helped in the training the model quite well. We could get better accuracy because of that. Activation functions play a key role here. We do not need to have very high number of layers and neurons. Descent number of that would also work good. For learning rate we should not take too high or too less, so 0.001 is chosen. Also for higher value of weight decay we are getting big loss. So a minimal amount of L2 weight decay would be good and it will help against overfiting. The Shirt and pullovers category were quite confusing to distuinguish but other were good.

The following are the best three configuration we run

Configuration 1

- Number of epochs: 5

Number of hidden layers: 5

Size of each hidden layer: 64

Learning rate: 0.0001

Optimizer: rmsprop

Batch size: 64

Weight Initialization: xavier

Activation: tanh

Loss: cross_entropy

MNIST Validation Accuracy: 85.504

Configuration 2

Number of epochs: 5

Number of hidden layers: 3

Size of each hidden layer: 64

Learning rate: 0.001

Optimizer: momentum GD

Batch size: 32

Weight Initialization: xavier

Activation: relu

Loss: cross_entropy

MNIST Validation Accuracy: 85.137

Configuration 3

Number of epochs: 5

Number of hidden layers: 4

Size of each hidden layer: 64

Learning rate: 0.0001

Optimizer: adam

Batch size: 64

Weight Initialization: xavier

Activation: relu

Loss: cross_entropy

MNIST Validation Accuracy: 80.524

Self Declaration

CS21M045: (50% contribution)

- Implementing Class structure for Layer and network
- implemented basic backpropagation
- implementing sgd, NAG, momentum GD, Batch GD
- setting up the sweep in wandb

CS21M046: (50% contribution)

- implementing RMSprop, Adam, Nadam
- setting up the sweep in wandb
- Running the sweeps
- plotting the confusion matrix- plotting the confusion matrix

We, Pranab Kumar Rout and Pratyush Dash, swear on our honour that the above declaration is correct.

https://wandb.ai/pandp/Ass1_sweep/reports/Assignment-1--VmlldzoxNjE0OTU4