# Assignment 01

Feedforward network with Backpropagation using wandb.ai.

Mitesh Khapra, Priyanka Bedekar, Sushane Parthan

**Link to this report:** https://wandb.ai/priyanka/deep-learning-assignment1/reports/Assignment-01--Vmlldzo1NDIwMDU

# Question 1

1. The Fashion-MNIST dataset was downloaded and 1 sample image per class was plotted.

2. Code for the above is as attached on GitHub as a Colab Jupyter Notebook.


# Question 2

1. We have implemented a Feedforward Neural Network which takes images from the Fashion-MNIST dataset and and outputs a probability distribution over the 10 classes present.

2. Code is flexible, such that it can be trained for a neural network of variable number of hidden layers and number of neurons per hidden layer.

# Question 3

1. Backpropagation algorithm was implemented with support for the following optimisation functions:

- SGD
- Momentum based GD
- Nesterov Accelerated Gradient Descent
- RMSProp
- Adam
- Nadam

2. The code remains flexible, such that batch sizes of any magnitude can be used to tune weights.

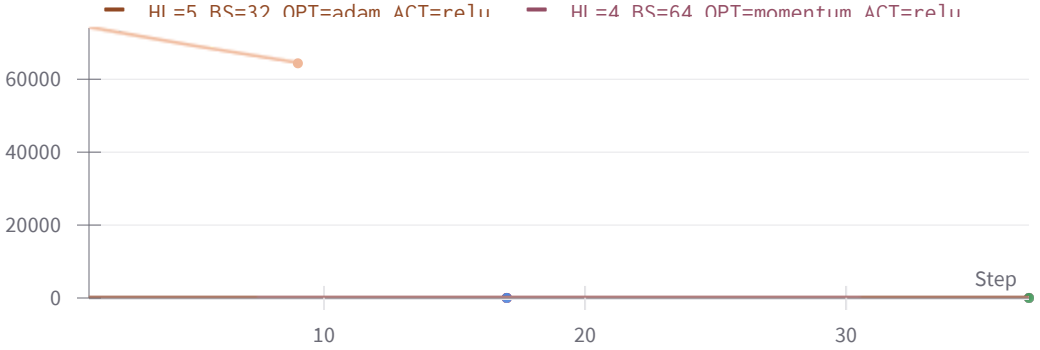
3. Variables such as ß_1, ß_2, running average of gradients and gradient value from previous step are stored as global variables, so that new optimisation algorithms such as Eve can be implemented.

# Question 4

1. Using sweep functionality of wandb, we ran different training optimisers on our neural network. Using the suggested hyperparameter values, to tackle the issue of an exponential number of combinations possible, we used wandb's **random** search method to do a faster selection process.

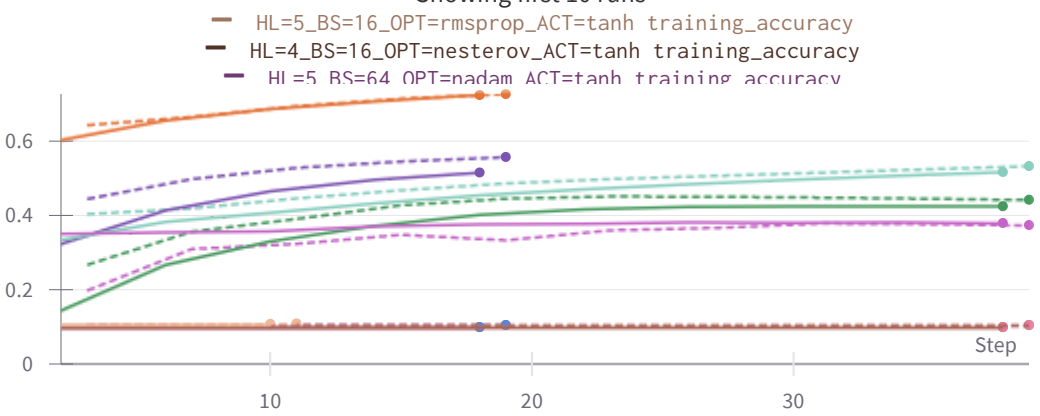
## validation_loss

Showing first 10 runs



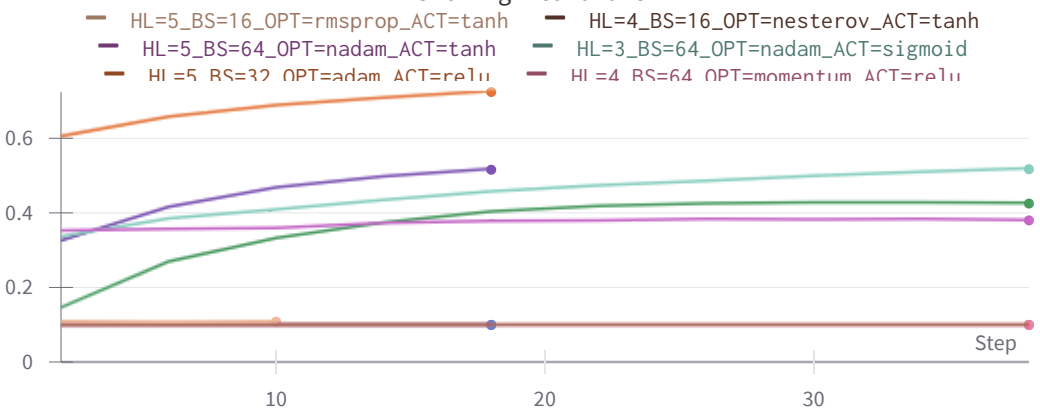

## training_accuracy, validation_accuracy

Showing first 10 runs



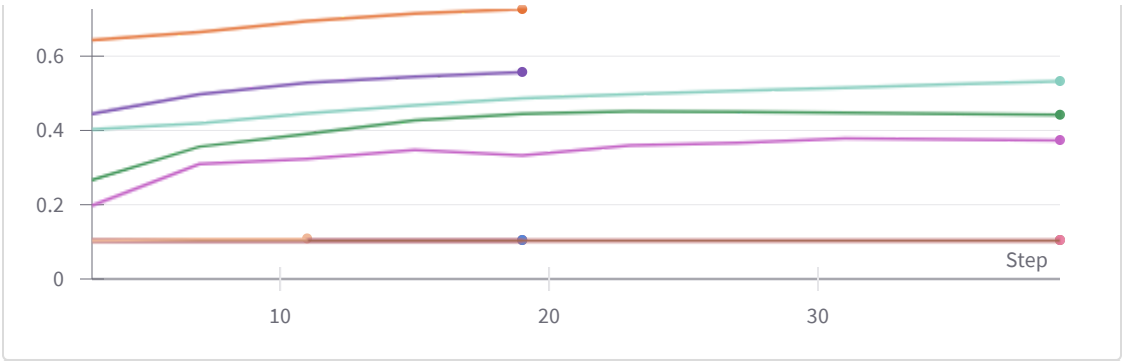

## training_accuracy

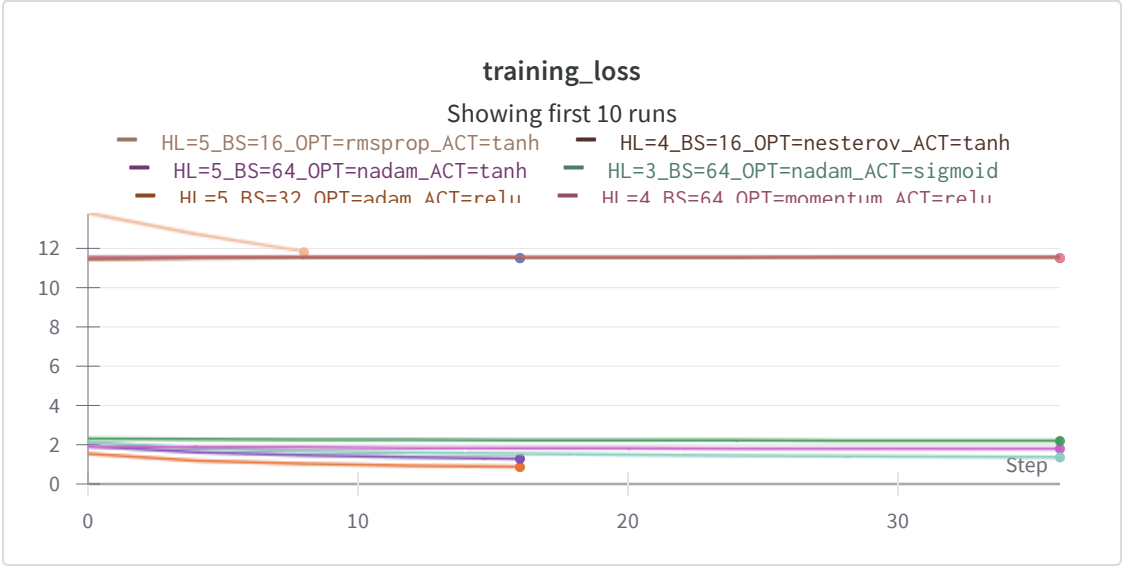
Showing first 10 runs




## validation_accuracy

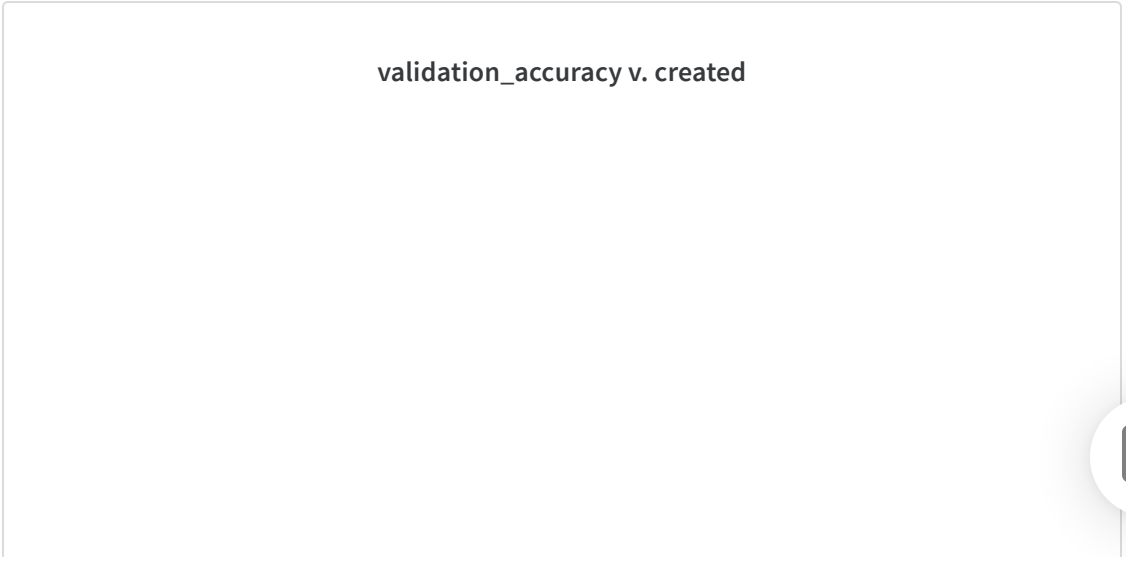
Showing first 10 runs

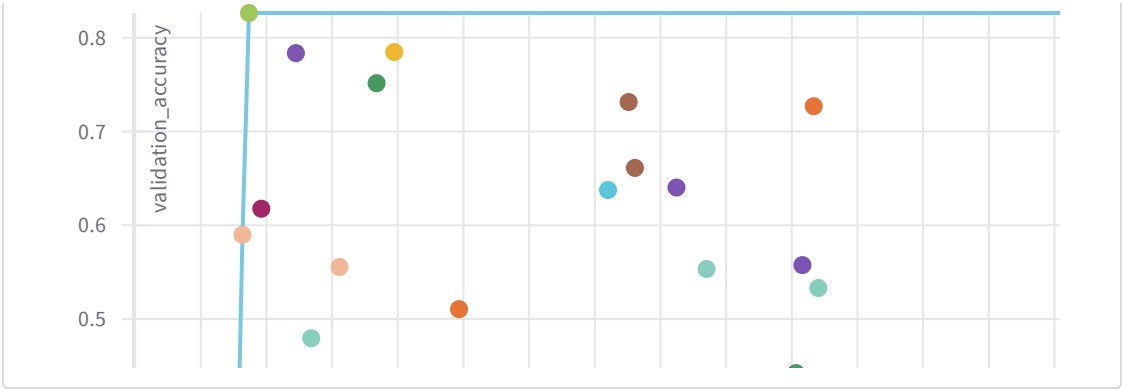



# Question 5

Accuracies of all plots of **Validation Data** and **Training Data** are as shown below.







# Question 6

1. Inferences from Correlation Summary, Parallel Coordinates Plot and Output Plot are as mentioned:

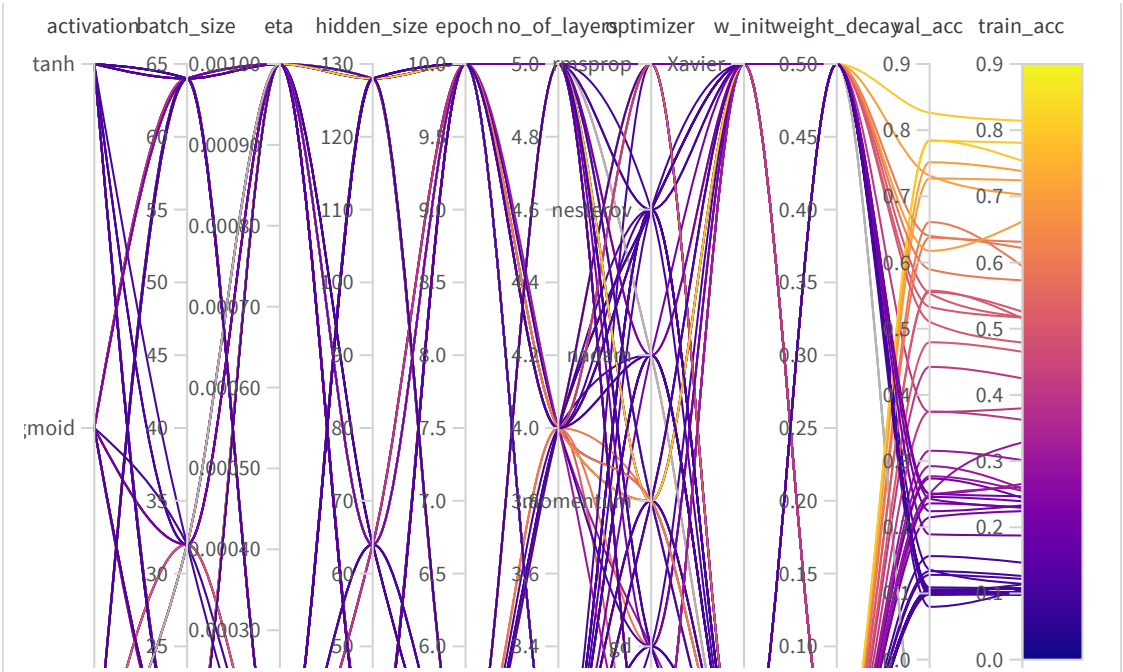
- ReLU as an activation function has the highest correlation with respect to validation and training accuracy.
- ReLU has advantages such as no vanishing gradient, and faster computation, as compared to other activation functions.

- For the given (suggested) values of hyperparameters, Nadam and RMSProp were favourable optimisers since they had a higher positive correlation. In general, RMSProp is a good choice for optimisers with respect to the given dataset since it converges faster.
- Xavier initialisation has higher positive correlation with respect to validation accuracy. In general, studies have also shown that Xavier initialisation is a good choice for random data initialisation because it enables different layers of a neural network to maintain near-identical variances as supposed to standard initialisation functions.
- Number of layers has a positive correlation with validation accuracy, which goes to show that deeper a neural network is, better the results are with test data.
- Hidden layer size has a positive correlation with validation accuracy. Greater the number of neurons in a hidden layer, greater the complexity of the network and  hence the results

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

2.  For achieving 95 percent accuracy, we would recommend the following:

- Number of hidden layers = 5
- Number of neurons in each hidden layer = 128
- Activation Function = ReLU with Softmax as output function
- Data initialisation = Xavier
- Optimiser = RMSProp, Nadam
- Number of epochs > 10
- Learning Rate = 0.0001
- Weight Decay = 0.0005




Parameter importance with respect to validation_acc…

Search

Parameters

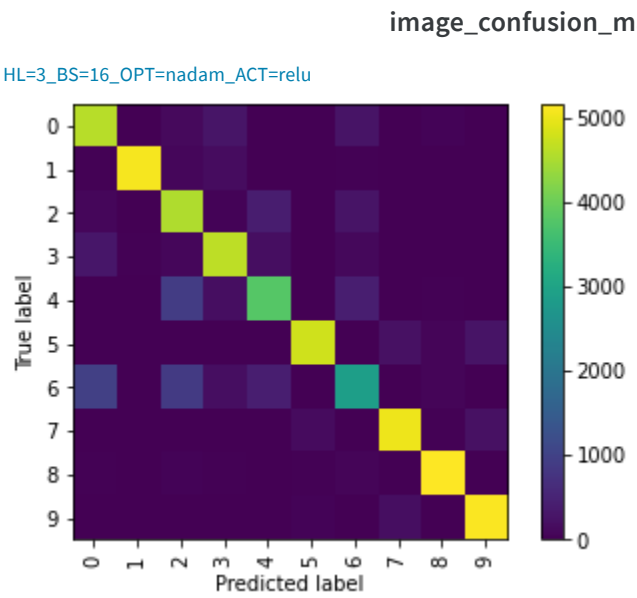
Rows per page 10

1-10 of 19

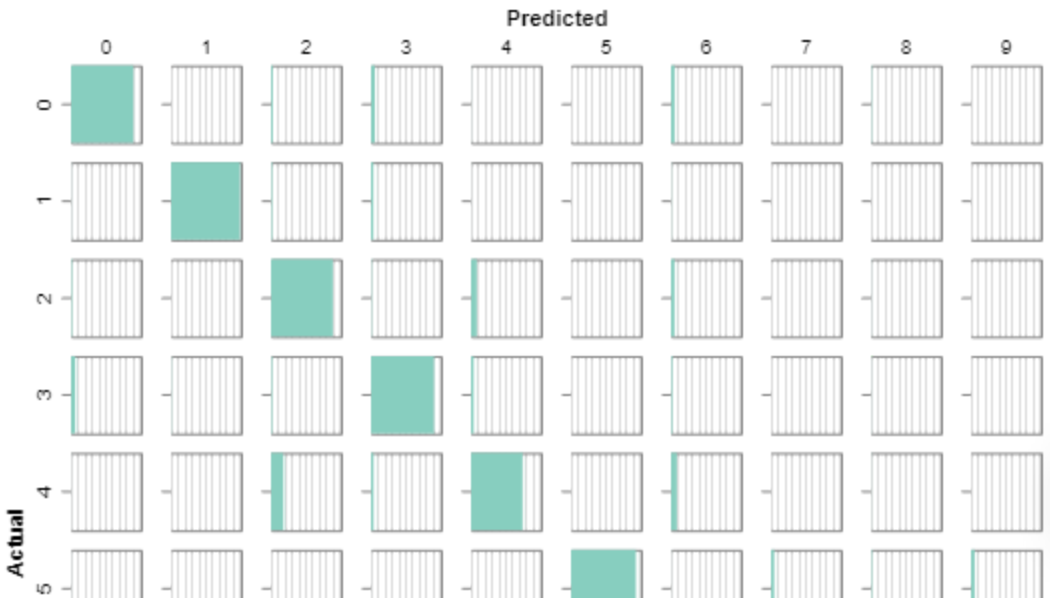
| Config parameter | Importance | Correlation |
|---|---|---|
| activation.value_relu | | |
| activation.value_tanh | | |
| eta | | |
| no_of_layers | | |
| w_init.value_Xavier | | |
| activation.value_sig… | | |
| w_init.value_Random | | |
| optimizer.value_adam | | |

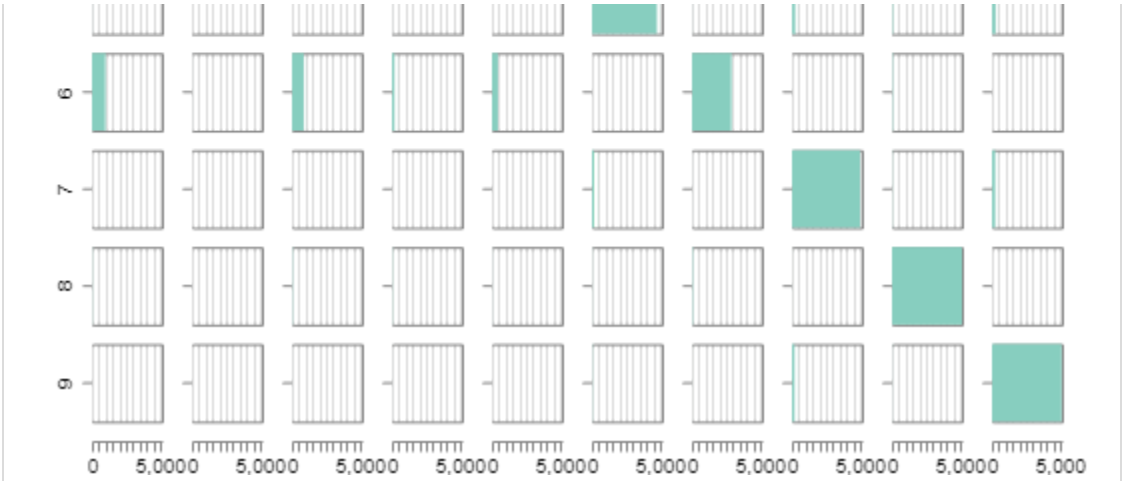
# Question 7

1. For the best model identified above, Training accuracy = 0.7833, Validation accuracy = 0.8057.




Selected Runs Without Media (10)






**conf_mat_table**

HL=3_BS=16_OPT=nadam_ACT=relu

| Actual | Predicted | nPredictions |
|---|---|---|
| 0 | 0 | 4598 |
| 0 | 1 | 1 |
| 0 | 2 | 133 |
| 0 | 3 | 290 |
| 0 | 4 | 16 |
| 0 | 5 | 1 |
| 0 | 6 | 273 |
| 0 | 7 | 0 |

# Question 9

https://github.com/SushaneP/CS6910-Deep-Learning.git

# Question 10

1. Recommendation for MNIST Dataset:

(a) Activation = ReLU, Optimiser = RMSProp, Initialisation = Xavier, Number of Layers = 5, Hidden Size = 128, Learning Rate = 0.0001, Weight Decay = 0.0005

Validation Accuracy = 0.7847

(b) Activation = ReLU, Optimiser = Nadam, Initialisation = Xavier, Number of Layers = 5, Hidden Size = 128, Learning Rate = 0.001, Weight Decay = 0.5.

Validation Accuracy = 0.8057

(c) (a) Activation = ReLU, Optimiser = Momentum, Initialisation = Xavier, Number of Layers = 5, Hidden Size = 128, Learning Rate = 0.0001, Weight Decay = 0.5

Validation Accuracy = 0.8264

# Self Declaration

CS20M050: (50% contribution)

- Implementing SGD, Momentum, Nesterov Optimisers.
- Implementing functions forward propagation and plotting Confusion Matrix.
- Setting up the sweep in wandb
- Analysing parallel co-ordinates plot and writing inferences.

CS20M003: (50% contribution)

- Implementing RMSProp, Adam, Nadam
- Implementing functions for Backpropagation for variable batch sizes, and Loss and Accuracy.
- Setting up the sweep in wandb.
- Analysing the parallel coordinates plot and writing inferences.

We, **Priyanka Bedekar** and **Sushane Parthan**, swear on our honour that the above declaration is correct.

# Note:

1. Number of extra hours taken = 118 hours (04 days and 22 hours).
2. Number of hours remaining = 242 hours.

Created with ♡ on Weights & Biases.

https://wandb.ai/priyanka/deep-learning-assignment1/reports/Assignment-01--Vmlldzo1NDIwMDU