CS6910 Assignment - 1

Name: Ananthakrishnan A

Roll No: ED22S015

WandB Report Link :
https://wandb.ai/ananthu2014/CS6910_DL_ASS1/reports/CS6910-Assignment-1--VmlldzozNzA2NzAy

GitHub Repository Link :
https://github.com/ananthu2014/CS6910.git

Edit Share

CS6910 - Assignment 1

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

Ananthakrishnan A

(ED22S015)

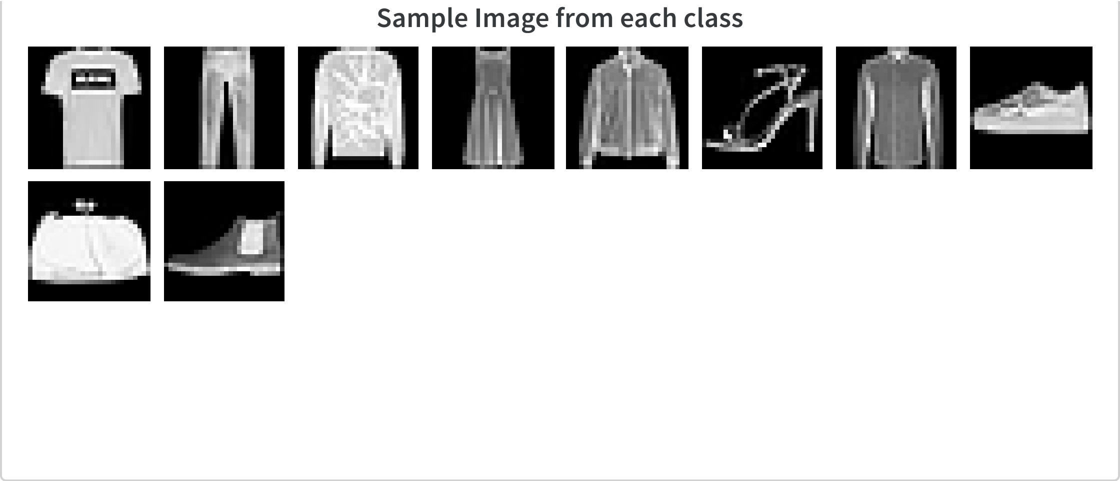
Problem Definition

In this assignment, a feed-forward neural network has been implemented and the backpropagation has been performed for training the network. For all the matrix operations, the Numpy library was used and no automatic differentiation packages were used for the implementation of backpropagation. This network was 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 was trained to classify the image into 1 of 10 classes.

The code follows the format specified in the `Code Specifications` section.

Question 1 (2 Marks)

The FASHION-MNIST dataset was downloaded from Keras. This is a dataset which contains multiple classes, namely trousers, pullovers, dresses, coats, sandals, shirts, sneakers and bags. Given below is the visualization of different classes of the dataset.



Sample Image from each class

Question 2 (10 Marks)

A feed-forward neural network has been implemented, which is flexible enough to change the number of layers and the neurons in each layer. The weights and biases can be initialized using either 'random' initialization or 'Xavier' initialization. In each layer, various activation functions can be used such as the 'Sigmoid' function, 'Identity' function, 'tanh' function, 'ReLu' function etc. The output layer uses the 'soft max' function which gives the probability of occurrence of each class. Given below is the GitHub link of the code which contains the implementation of feed-forward neural network.

https://github.com/ananthu2014/CS6910/blob/master/Assignment%201/feed_forward_NN.ipynb

Question 3 (24 Marks)

The backpropagation algorithm has been implemented in the neural network with support for the given variants of the gradient descent algorithm.

- Vanilla gradient descent

- Stochastic gradient descent
- Momentum-based gradient descent
- Nesterov accelerated gradient descent(NAG)
- RMS-Prop
- Adam(Adaptive Moments)
- Nesterov-adam(NAdam)

The code is flexible to work with each one of the above algorithms and in different batch sizes. A new optimisation algorithm can be easily added without any changes to the existing framework of the code.
Given below is the GitHub link to the code.

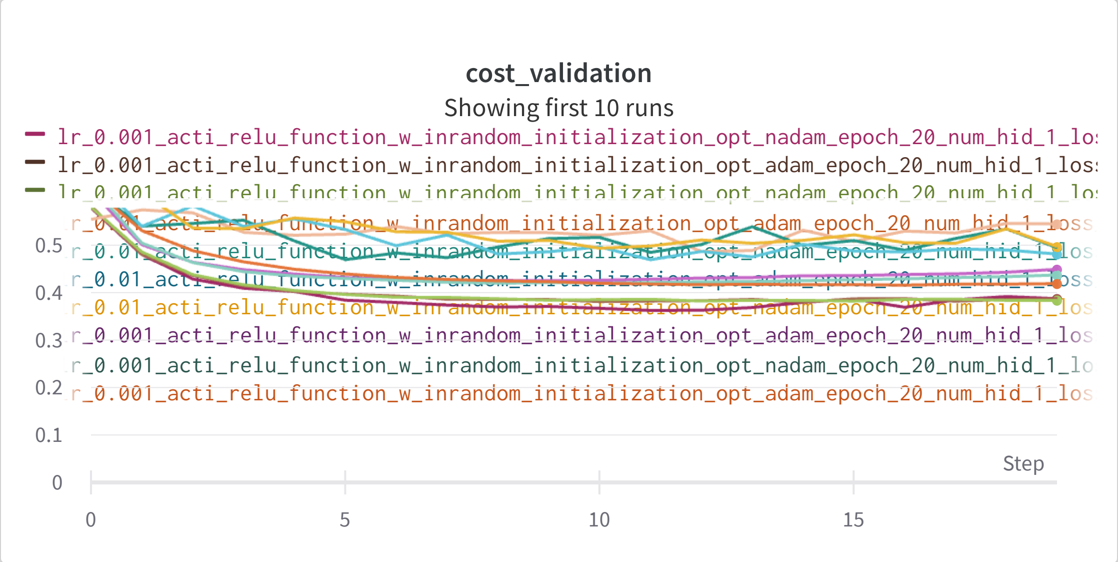
https://github.com/ananthu2014/CS6910/blob/master/Assignment%201/CS6910%20Assignment-1(CODE%20ONLY).ipynb

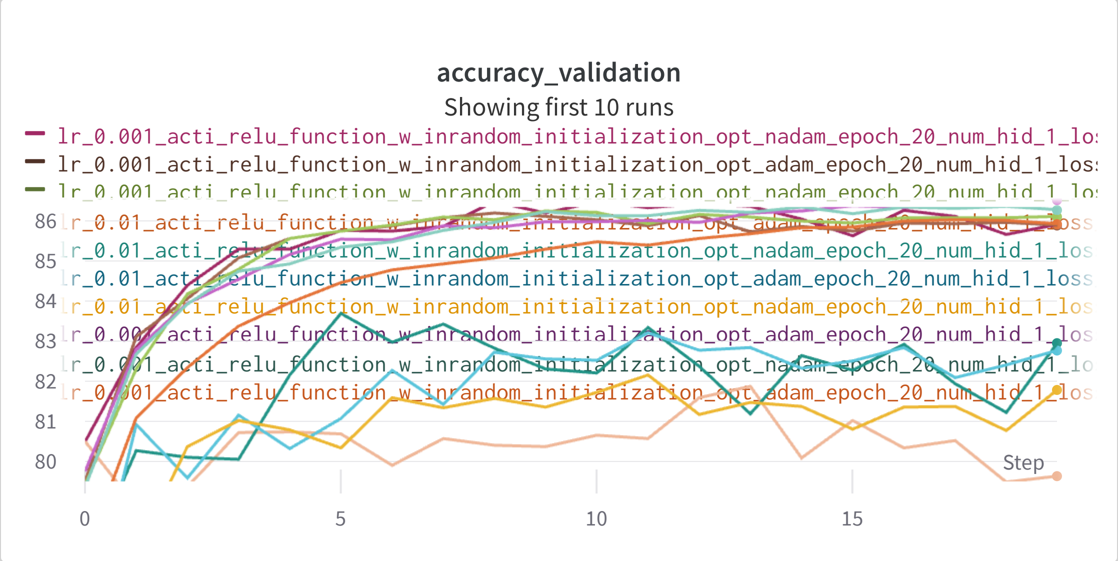
Question 4 (10 Marks)

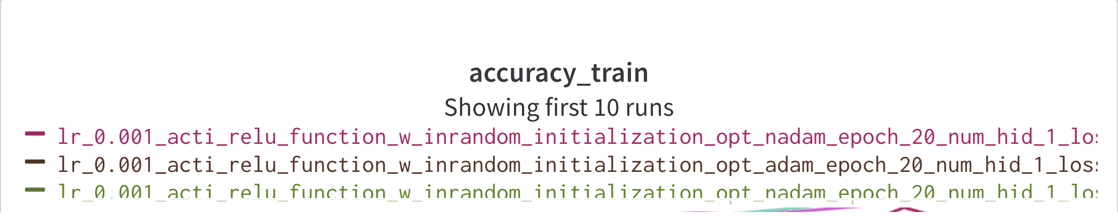
The sweep functionality provided by 'WandB' has been used to find the best values for the hyperparameters listed below. 10% of the training data was kept aside as validation data for this hyperparameter search. Given below was the choices of different hyperparameters in the neural network.

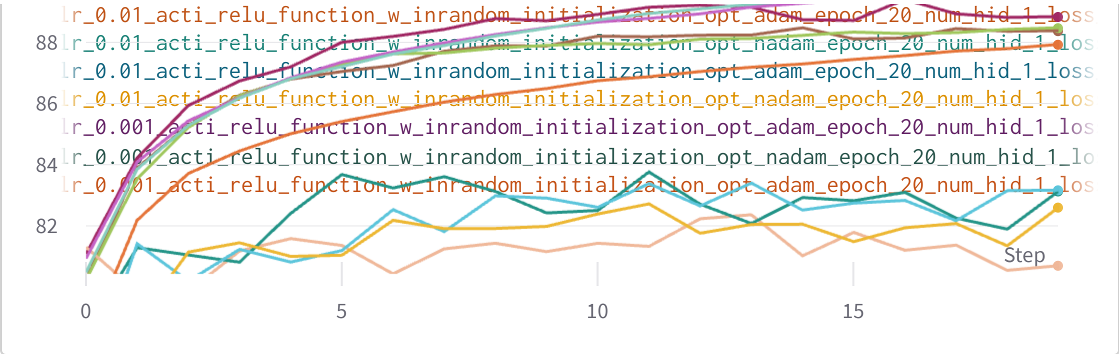
- number of epochs: 10, 15, 20, 40
- Number of hidden layers: 1, 2, 3
- The size of every hidden layer: 32, 64, 128
- weight decay (L2 Regularization): 0, 0.0005, 0.001, 0.01
- learning rate: 1e-1, 1e-2, 1e-3, 1 e-4 ,1e-5
- Optimizer: Stochastic gradient descent, Momentum gradient descent, NAG, Adam, RMS-Prop, Nesterov Adam
- batch size: 2, 4, 8, 16, 32, 64
- Weight initialization: Random, Xavier
- activation functions: Sigmoid, Tanh, ReLU, Identity functions

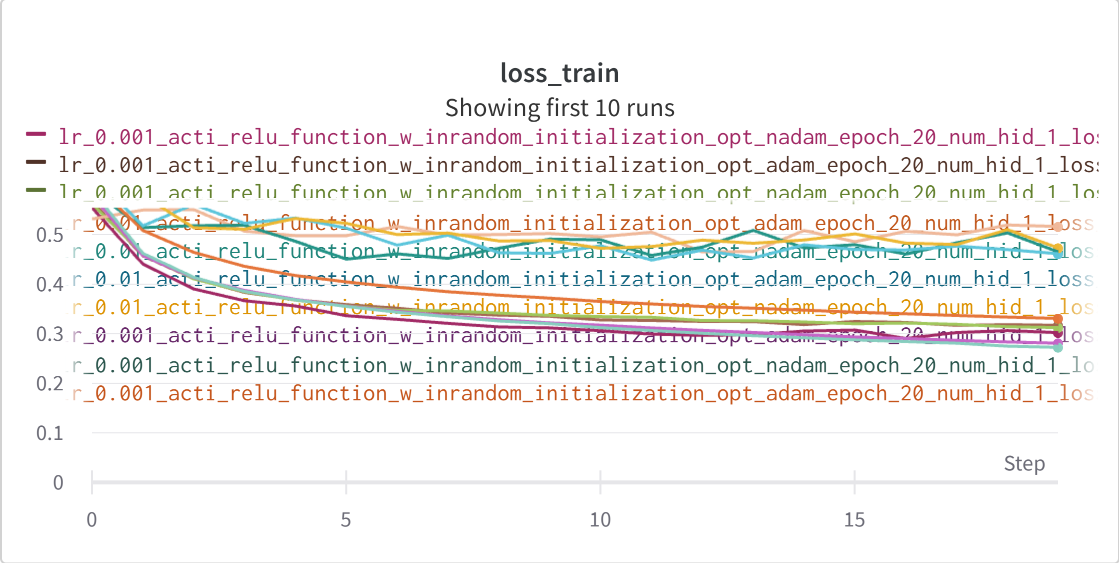
- loss functions: Mean-squared-error, Categorical-cross-entropy functions
- Momentum: 0.8,0.9
- These are the plots generated by wandb for train accuracy, validation accuracy, train cost, validation cost and number of epochs. The rest of the plots are given in the subsequent sections. The names of each sweep are given in such a way that it gives the details of all the hyperparameters used in that particular run.

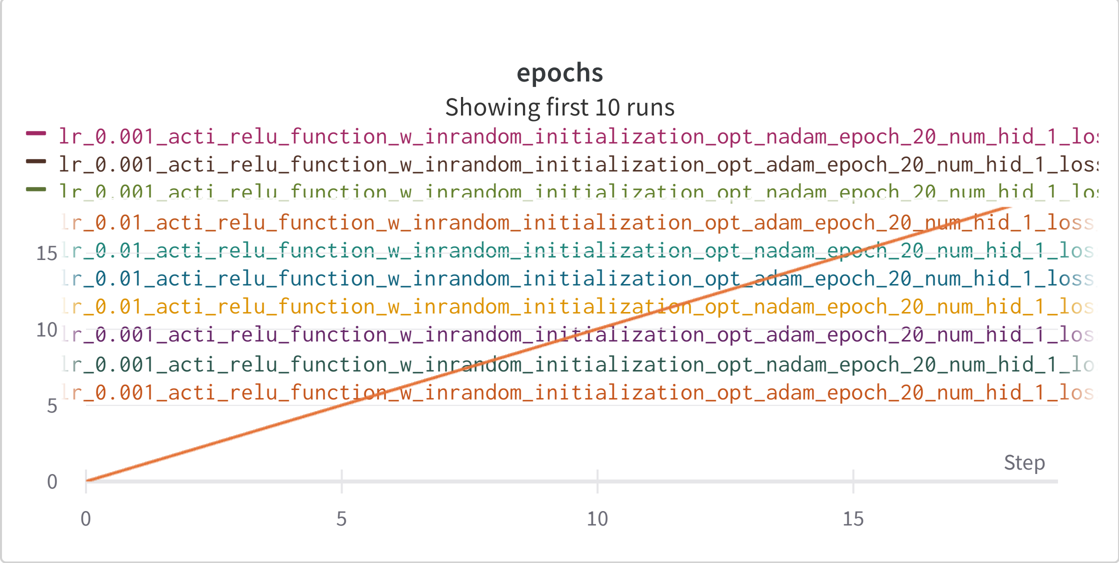












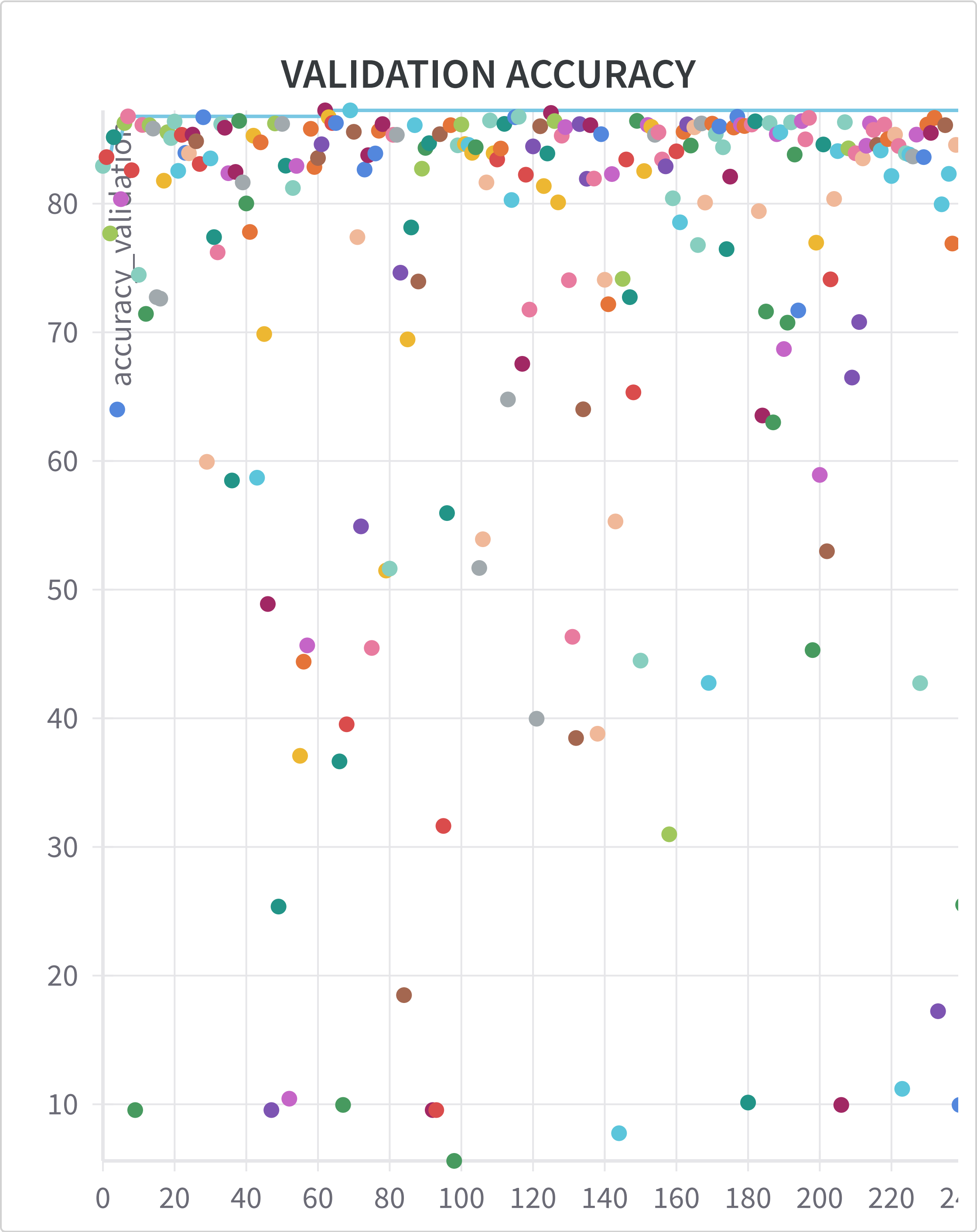
Question 5 (5 marks)

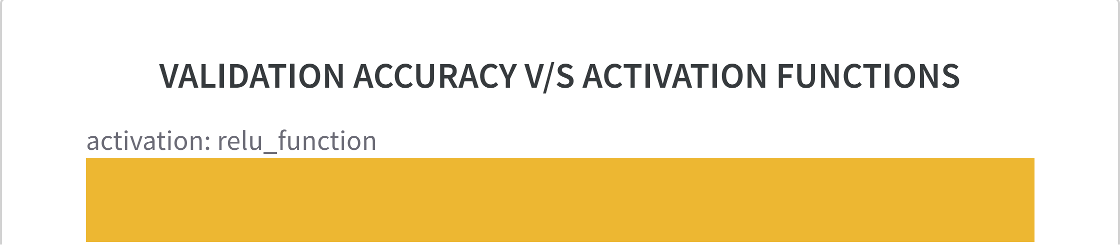
Given below are three plots based on the results of different Sweeps:

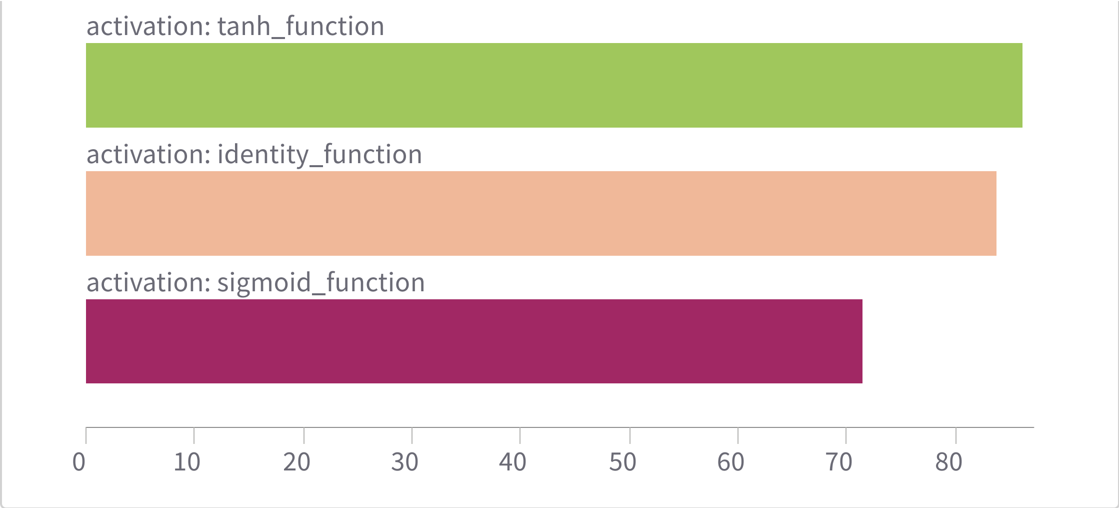
1) VALIDATION ACCURACY VS INDEX

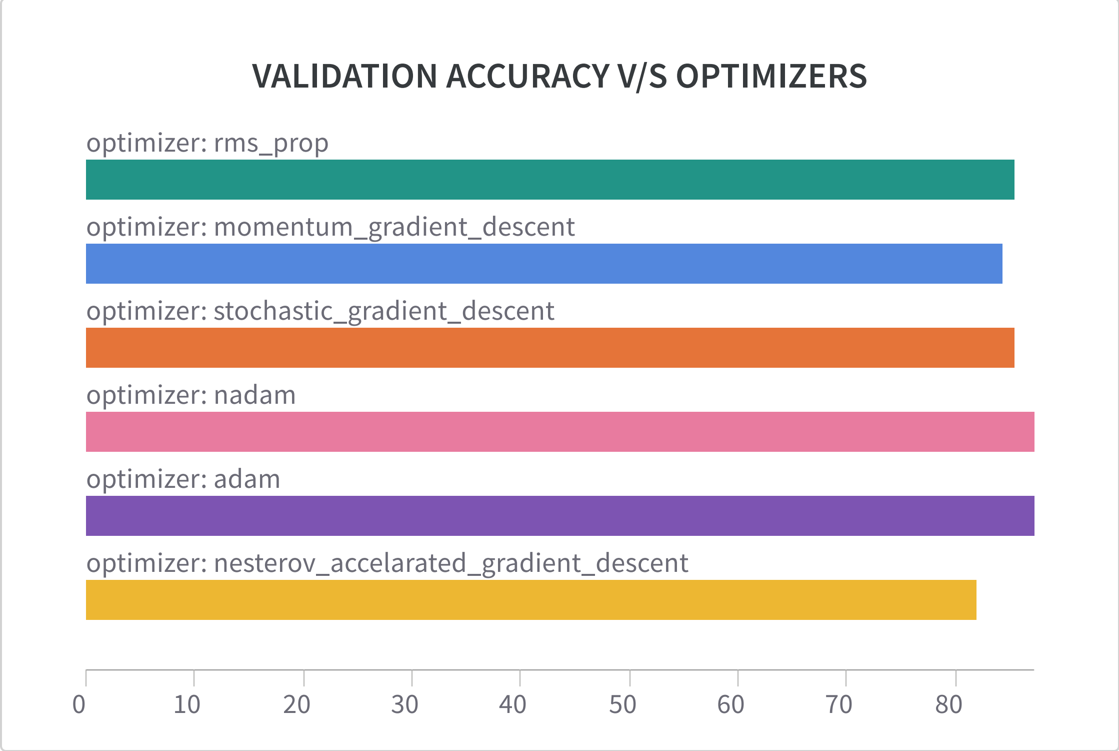
2) VALIDATION ACCURACY VS OPTIMIZER

3) VALIDATION ACCURACY VS ACTIVATION FUNCTIONS


VALIDATION ACCURACY


VALIDATION ACCURACY V/S ACTIVATION FUNCTIONS





OBSERVATIONS:

The highest validation accuracy obtained was found to be 87.25% and the test accuracy for the same was found to be 86.1%. The values of different hyperparameters for the same were as follows:-

- learning rate=0.001
- epochs=20
- batch size=8
- momentum =0.9
- neurons=64

- number of hidden layers=1
- Activation function =ReLu function
- Loss = Cross entropy function
- Optimizer=Adam
- Regularization parameter=0

I have tried various methods to find the validation accuracy values. It was found that normalized data gave more accurate results than UN-normalised data. I have divided the input data by 255 for normalization.

When L2 Regularization was used, it was found that the overfitting decreased considerably where the difference between train accuracy and validation accuracy decreased and were almost the same.

I have tried to add noise to the data by adding Gaussian noise with mean=0 and variance=0.01 and the results obtained were similar to what I got when L2 regularization was added.

STRATEGY USED FOR PERFORMING SWEEP:-

There are 3 different sweep methods for performing sweep: 'RANDOM', 'BAYESIAN', and 'GRID'. The random method takes random combinations of various hyper-parameters used and performs a sweep. The grid method takes all combinations of all the values of hyper-parameters given. The Bayes method involves using probabilistic methods to search the hyper-parameter space in order to find the optimal set of hyper-parameters that result in the best performance of the model.

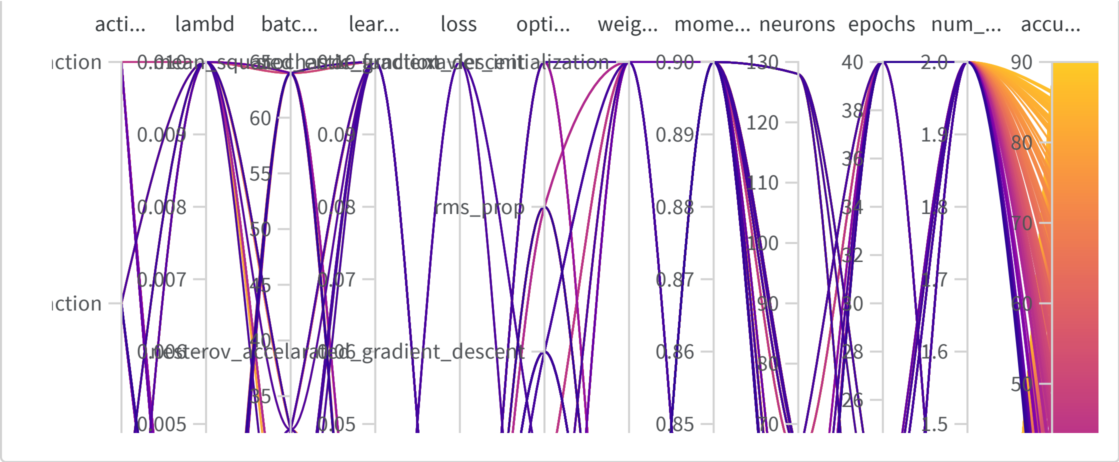
Initially, I used a Random search since the hyper-parameters were high in number and sorted the parameters which gave the highest validation accuracy. Then, I performed subsequent Bayes and Grid sweeps to find the best validation accuracy by reducing the sampling size of a number of hyper-parameters in the subsequent sweeps. As a consequence, a total number of 479 sweeps have been performed.

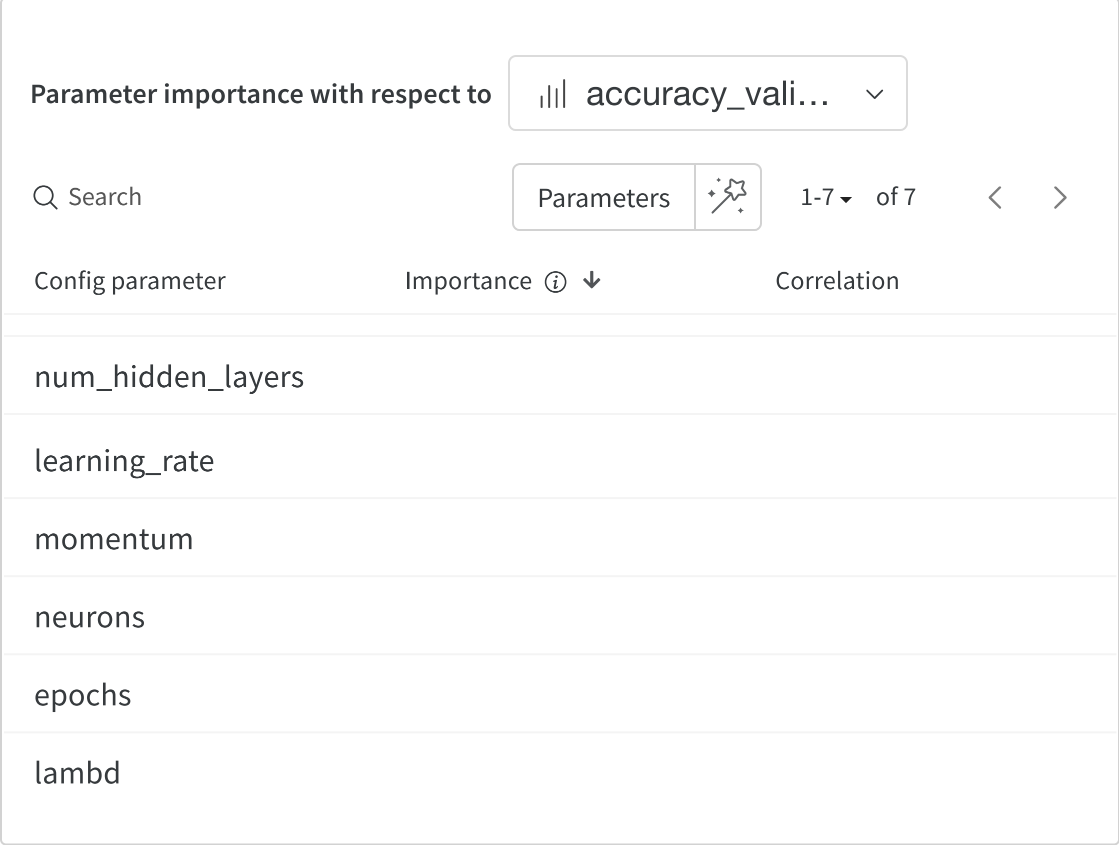
Question 6 (20 Marks)

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

- Adam and NAdam were found to be the best optimizer with the former giving 87.30% and the latter giving 87.25% validation accuracy.
- The ReLu function was the better-performing activation function whereas the tanh function also performed well.
- As the learning rate decreases below a limit, the final validation accuracy was found to be less. The same was the case when a very low learning rate was used but the former gave a rather fluctuating graph of validation accuracy v/s epochs.
- As the learning rate decreases below a limit, the final validation accuracy was found to be less. The same was the case when a very low learning rate was used but the former gave a rather fluctuating graph of validation accuracy v/s epochs.
- As the learning rate decreases below a limit, the final validation accuracy was found to be less. The same was the case when a very low learning rate was used but the former gave a rather fluctuating graph of validation accuracy v/s epochs.
- Batch size and the number of hidden layers were found to be the most important parameters from the 'Parameter importance and correlation summary' graph.

PARALLEL COORDINATES PLOT





Parameter importance with respect to accuracy_vali… ⌄

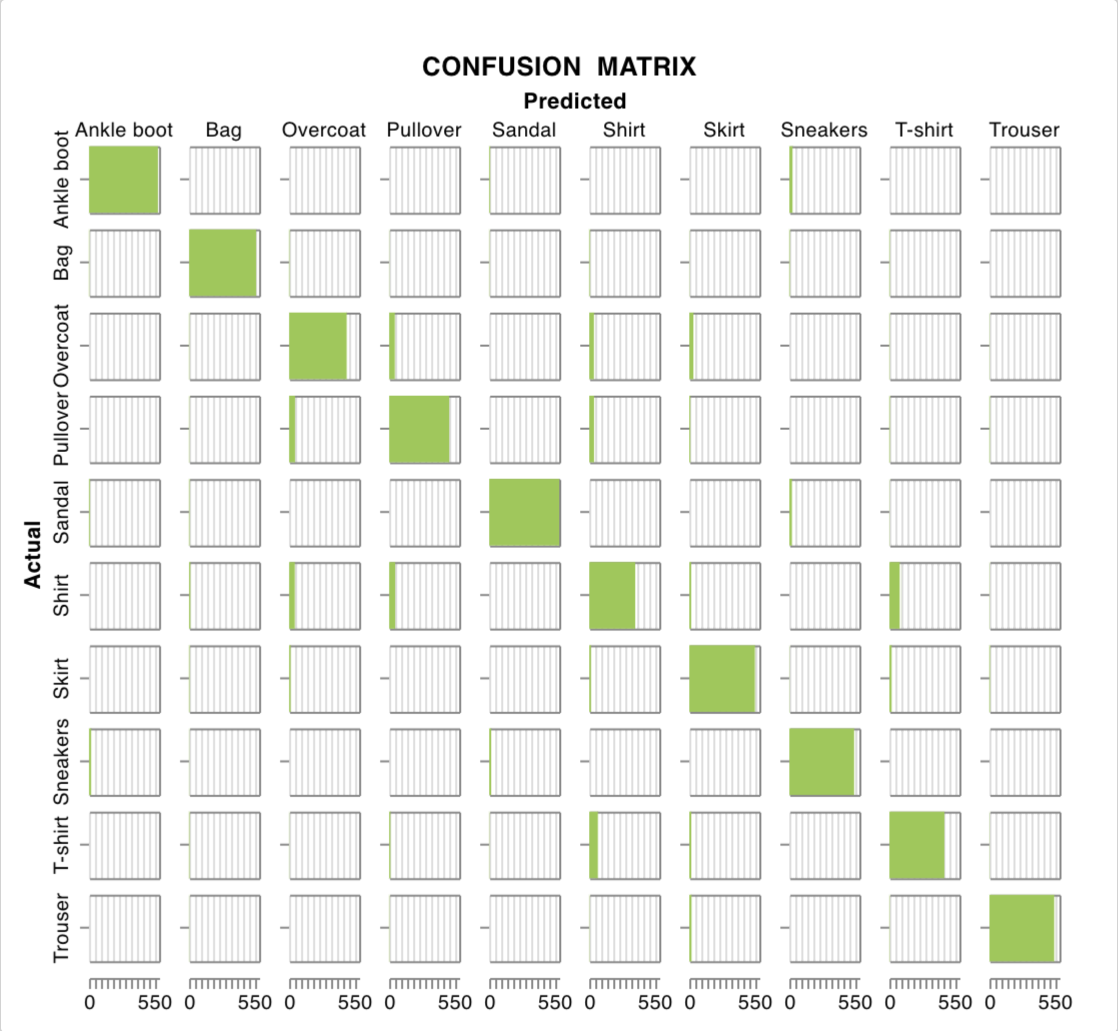
Search Parameters 1-7 ▾ of 7 ‹ ›

Config parameter	Importance ⓘ ↓	Correlation
num_hidden_layers		
learning_rate		
momentum		
neurons		
epochs		
lambd		

Question 7 (10 Marks)

The confusion matrix for the highest accuracy case is plotted below. In the vertical axis, the Actual classes are given and the predicted classes are given in the horizontal axis. Given below is the link to the code which includes the plot of conusion matrix for test and validation data:-

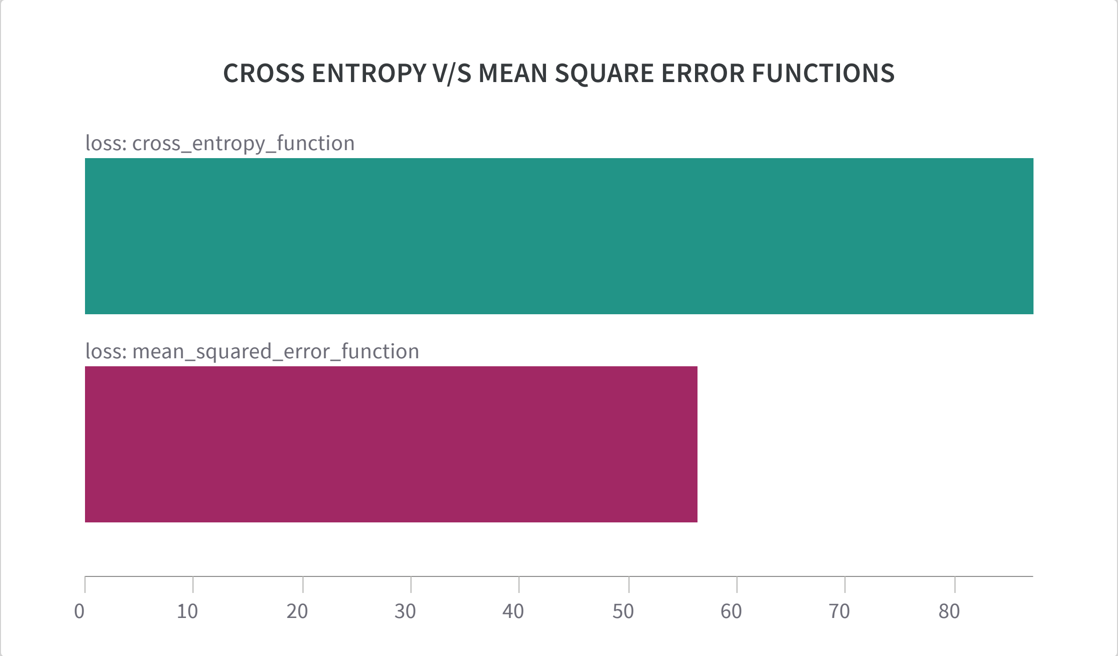
https://github.com/ananthu2014/CS6910/blob/master/Assignment%201/CS6910%20Assignment-1-partial%20sweep%20results.ipynb



Question 8 (5 Marks)

I have used the "Cross-entropy" loss function for the previous sweeps in which I got the highest accuracy. Now, here, I compared the validation accuracy given by the "Mean-Squared-Error" loss function. It was found that categorical cross entropy gave the highest accuracy of 87.25% whereas Mean squared error gave the validation accuracy of 56.417% for the same hyper parameter configuration which gave the highest accuracy for the former. This owes to the fact that the cross-entropy function gives a higher penalty in cases of misclassification and the error will tend to be higher so that the minimization will also be higher, whereas in case of the mean square error function, even when misclassification occurs, the error will be less because the difference between y_pred and y will max be only 1 (0-1)**2 and reflects in lesser minimization of error as well. Therefore, in the case of classification problems, the categorical cross-entropy loss function performs well and is widely used in feed-forward neural networks.

Another thing I observed while using the mean-squared error loss was that the graph of accuracy and loss v/s epochs were fluctuating continuously between higher and lower values.



Question 9 (10 Marks)

Given below is the link to GitHub:-

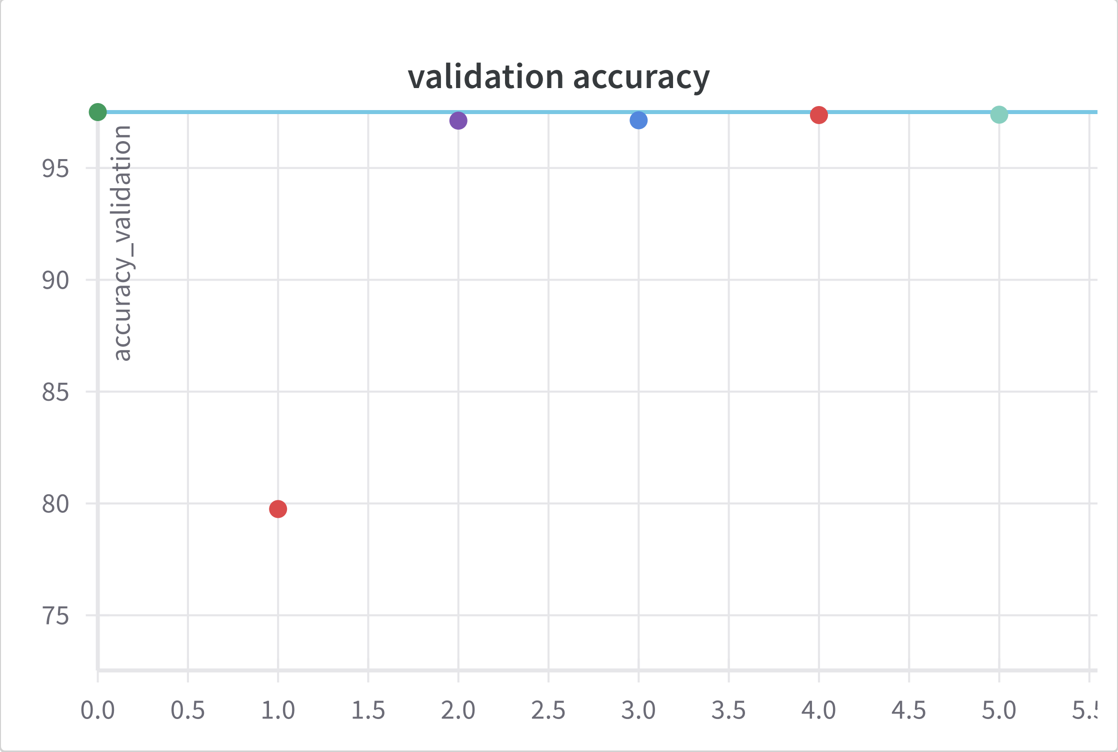
Example: https://github.com/ananthu2014/CS6910.git

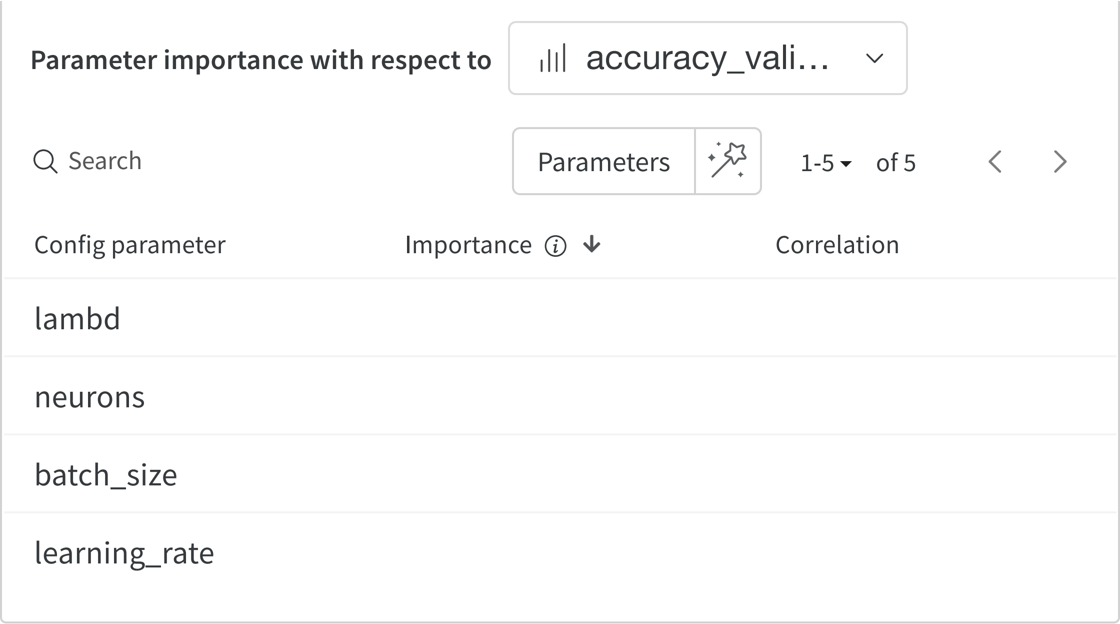
https://github.com/ananthu2014/CS6910/blob/master/Assignment%201/CS6910%20Assignment-1.ipynb

- The training and test data have been split properly and randomly using random methods and the weights are also initialized similarly.

Question 10 (10 Marks)

Based on the learning on the FASHION-MNIST dataset, experimentation was done on the MNIST dataset. It was found that the highest validation accuracy obtained was 97.5% which was very high compared to the 87.25% validation accuracy obtained on the FASHION-MNIST dataset. The validation accuracy was obtained as given below:-




Parameter importance with respect to accuracy_vali…

Config parameter	Importance	Correlation
lambd		
neurons		
batch_size		
learning_rate		

From the parameter importance and correlation graph, it was found that the regularization parameter, number of neurons in the hidden layer, learning rate, batch size etc were highly influential in the validation accuracy of the MNIST dataset. Based on the sweeps, the three hyperparameter configurations that I would recommend are:-

- Learning rate=0.001
- Regularization parameter=0.001
- Epochs=20
- Optimizer=Adam
- Batch size=8
- Number of hidden layers=1
- Neurons=64
- Beta1=0.9
- Beta2=0.99
- Loss=cross entropy function
- Activation=relu function
- Validation accuracy=97.5%

- Learning rate=0.001
- Regularization parameter=0.001
- Epochs=20
- Optimizer=Adam
- Batch size=16
- Number of hidden layers=1
- Neurons=64
- Beta1=0.9
- Beta2=0.99
- Loss=cross entropy function
- Activation=relu function
- Validation accuracy=97.38%

- Learning rate=0.001
- Regularization parameter=0.001
- Epochs=10
- Optimizer=NAdam
- Batch size=8
- Number of hidden layers=1
- Neurons=64
- Beta1=0.9
- Beta2=0.99
- Loss=cross entropy function
- Activation=relu function
- Validation accuracy=97.36%

Code Specifications

All the codes that were used to run have been added to the GitHub repository.

A python script called `train.py` has been provided in the root directory of the given GitHub repository that accepts the following command line arguments with the specified values:-

```
python train.py  --wandb_project CS6910_DL_ASS1
```

Arguments to be supported

Name	Default Value	Description
`--wandb_project`	myprojectname	Project name used to track experim
`--dataset`	fashion_mnist	choices: ["mnist", "fashion_mnist"]
`--epochs`	20	Number of epochs to train neural n
`--batch_size`	8	Batch size used to train neural netw
`--loss`	cross_entropy_function	choices: ["mean_squared_error_fu
`--optimizer`	adam	choices: ["vanilla_gradient_descent,"stocha "nesterov_accelarated_gradient_d
`--learning_rate`	0.001	Learning rate used to optimize moc
`--momentum`	0.9	Momentum used by momentum an
`--beta`	0.9	Beta used by rmsprop optimizer
`--beta1`	0.9	Beta1 used by adam and nadam op
`--beta2`	0.99	Beta2 used by adam and nadam op
`--epsilon`	1e-8	Epsilon used by optimizers.
`--lambd`	0	Regularization parameter
`--weight_init`	random_initialization	choices: ["random_initialization", "
`--num_hidden_layers`	1	Number of hidden layers used in fe
`--neurons`	64	Number of hidden neurons in a fee
`--activation`	relu_function	choices: ["identity_function", "sigm

The default hyperparameters are set to the values that gave the best validation accuracy for the FASHION-MNIST dataset.

(NB): The 'WandB entity' parameter is not given since the kernel was getting crashed continuously when it was given.

Self Declaration

I, Ananthakrishnan A(ED22S015), 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.

References :

- CS6910-DEEP LEARNING: NPTEL
- YOUTUBE(UNDERSTANDING VARIOUS PRINCIPLES)
- CS6910-PPT

https://wandb.ai/ananthu2014/CS6910_DL_ASS1/reports/CS6910-Assignment-1--VmlldzozNzA2NzAy