

# Assignment 1

Submitted by Argha Boksi, Shubham Mallik Thakur

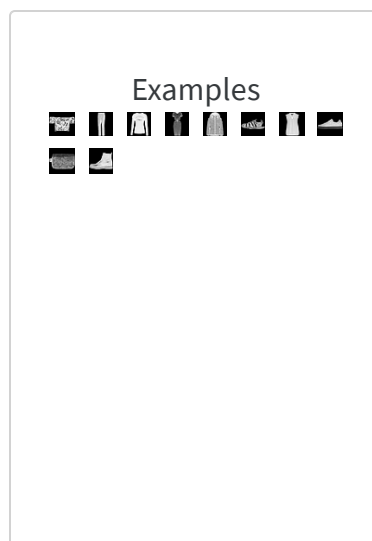
Argha Boksi cs21d407, Shubham Mallik Thakur

Created on February 19 | Last edited on February 25

## ▾ 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 ( $28 \times 28 = 784$  pixels) from the Fashion-MNIST dataset, the network will be trained to classify the image into 1 of 10 classes.

## ▾ Question 1: One sample image for each class



## ▸ Question 2: Implementation of Feedforward Neural Network

Implemented a feedforward neural network which takes a 784 dimensional vector as input and outputs a probability distribution over the 10 classes.

It is easy to change the number of hidden layers and the number of neurons in each hidden layer. In our code we have a list named "layer\_dimensions", which can be used to change the number of hidden layers and the number of neurons in each hidden layer.

Hidden layer activation functions : sigmoid, relu, tanh

Output layer activation function : softmax

Functions : linear\_computation\_of\_a\_layer(), activation\_of\_a\_layer(), forward\_propagation()

## ▸ Question 3: Backpropagation, Different Optimisation Functions

Implemented backpropagation algorithm. Functions : backward\_propagation(), forward\_backward()

Implemented the following optimisation functions :

1) Vanilla GD, Mini-Batch GD and SGD. Functions : mini\_batch\_gradient\_descent(), mini\_batch\_gradient\_descent\_update\_rule().

2) Momentum based gradient descent. Function: momentum()

3) Nesterov accelerated gradient descent. Function : nesterov()

4) RMSprop. Function : rms\_prop()

5) Adam. Function : adam()

6) Nadam. Function : nadam()

The code is flexible enough to work with different batch sizes.

Function : mini\_batch\_generation()

## ▼ Question 4: Sweep Functionality/ Hyperparameter Tuning

Configured the "sweep\_config" object with the following set of hyperparameters values with the goal of maximizing "val\_acc".

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, 1e-4

optimizer : sgd, momentum, nesterov, rmsprop, adam, nadam

batch size: 16, 32, 64

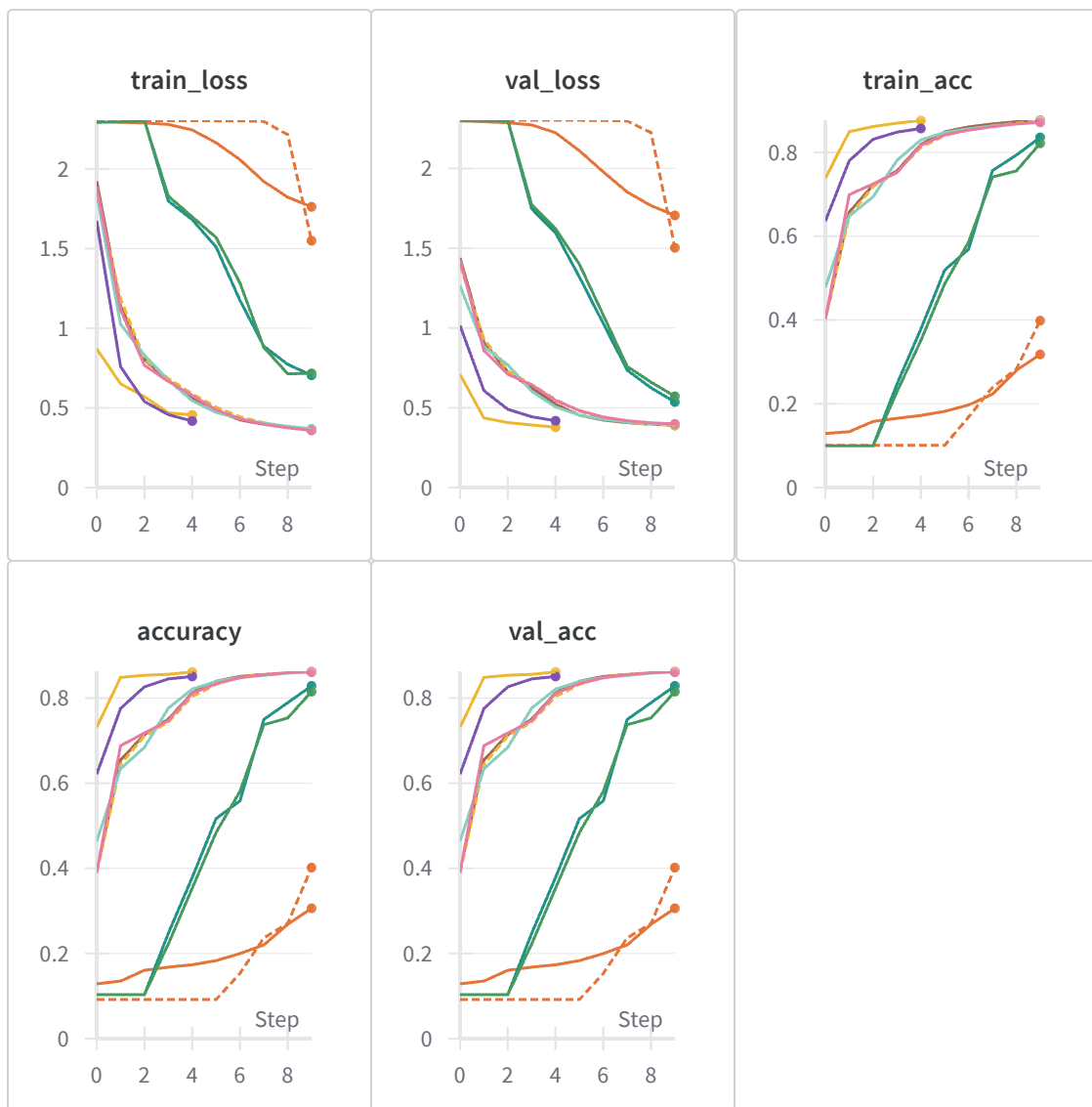
weight initialisation: random, xavier

activation functions: sigmoid, tanh, ReLU

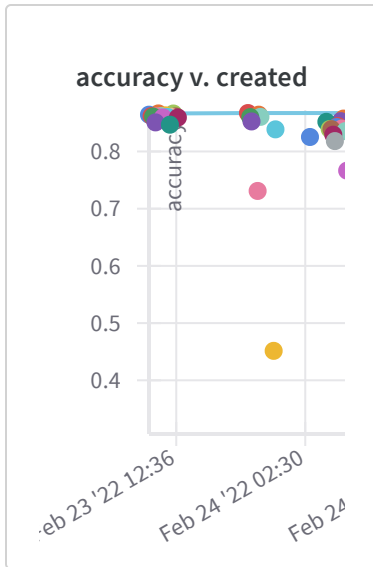
We used the standard train/test split of fashion\_mnist and kept aside 10% of the training data(6,000 examples) as validation data for

hyperparameter search.

The set of possible hyperparameters values lead to exponential number of combinations. So for efficient hyperparameter search we picked "Bayesian Search" method provided by wandb. "Bayesian Sweeps" create a probabilistic model of metric score as a function of the hyperparameters, and choose parameters with high probability of improving the metric(in our case the metric is "val\_acc" and the goal is "maximize").



## ▸ Question 5 : Accuracy on the Validation Set across Trained Models



## ▸ Question 6: Observations and Inferences


Some observations based on the experiments.

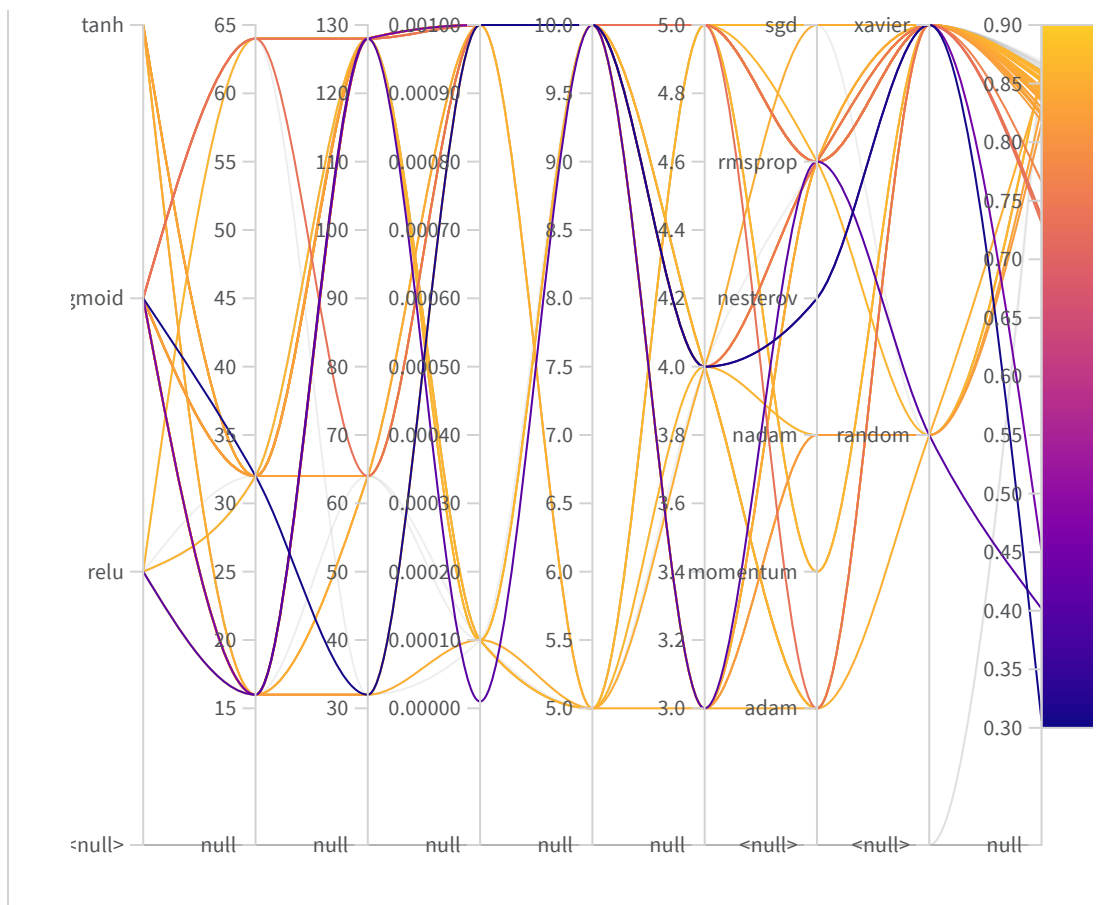
- 1) Best validation accuracies are reported by adaptive learning algorithms such as rms\_prop, adam and nadam.
- 2) Runs with high validation accuracies are usually associated with high learning rate( $1e-3$ ). Low learning rates provide smaller updates and may take more time to train.
- 3) Increasing Batch size increases the validation accuracy.
- 4) As we increase the number of hidden layers and hidden layer size, i.e, if we increase the model complexity, we see an improvement in

validation accuracies while training. This might be due to model's increased ability to learn complex features present in the data.

5) Tanh seems to perform slightly better when compared with ReLU and sigmoid in terms of observed validation accuracies.

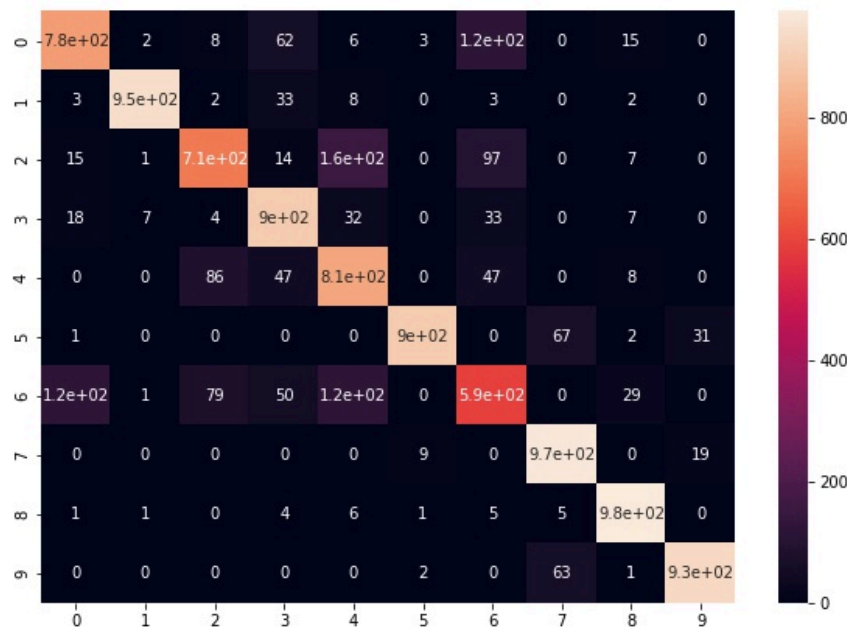
For getting around 95% accuracy, we suggest trying learning rates larger than  $1e-3$ , high hidden layer size( $\sim 128$ ) and number( $\sim 4$ ), and using rms prop for training for around 10 epochs.

Parameter importance with respect to		accuracy	
Q Search	Parameters		1-10 of 16
Config parameter	Importance ⓘ ↓	Correlation	
Runtime			
batch_size			
hidden_layer_size			
number_of_hidden_l...			
activation.value_relu			
learning_rate			
weight_initialization....			
weight_initialization....			
activation.value_sig...			
activation.value_tanh			



## ▸ Question 7: Confusion Matrix and Test Accuracy

The confusion matrix below is plotted using sklearn and seaborn libraries. It is obtained from the predicted and actual labels on test dataset using rms prop algorithm with 3 hidden layers each of size 128 with learning rate 0.001 and batch size 32. The accuracy on test set is 0.8526

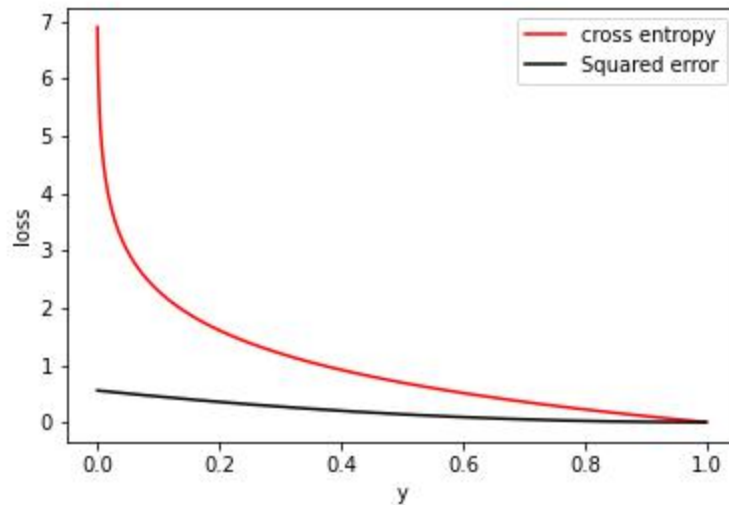


## ▼ Question 8: Squared Error vs Cross Entropy Loss

Consider a an output vector of probabilities such that its first entry is 'y', a number between (0,1) and the other 9 entries be  $(1-y)/9$ . This ensures that the probabilities sum up to one. Now using a target vector of  $[1,0,0,0,0,0,0,0,0,0]$ , i.e, vector indicating that label 0 is the true label, by varying 'y', we will calculate both cross entropy loss and squared error loss for this example and see how they behave.

We observe that when the true label and predicted label are close, both squared error and cross entropy are close to zero. But as we move away from the true label, we observe that squared error loss penalizes the difference way less than cross entropy loss. Hence, we conclude that in multi-class classification problems, such as this fashion-MNIST classification, cross entropy is much better than squared error loss.





## ▼ Question 9 : Github link

<https://github.com/arghaboksi/CS6910-Deep-Learning>

## ▼ Question 10 : MNIST Recommendations

- 1) We observed adaptive learning algorithms perform better than others, hence we would like to recommend using the same for MNIST.
- 2) Using high learning rate( $\sim 1e-3$ ) is recommended based on results from Fashion-MNIST.
- 3) Keep hidden layer size high(64-128) as they seem to report high validation accuracies on Fashion-MNIST.

Given a budget of running only 3 hyper parameter configs, we would like to test

- 1) (#Hidden Layers = 3, Batch size = 32, Hidden layer size = 128, epochs = 10, optimizer = rms prop, activation = sigmoid, lr =  $1e-3$ ):

reported val accuracy 0.8547

2) (#Hidden Layers = 4, Batch size = 32, Hidden layer size = 128, epochs = 10, optimizer = adam, activation = sigmoid, lr = 1e-3):  
reported val accuracy 0.8525

3) (#Hidden Layers = 5, Batch size = 16, Hidden layer size = 64, epochs = 10, optimizer = rms prop, activation = sigmoid): reported val accuracy 0.8427

## ▸ Self Declaration

Argha Boksi (CS21D407): (50% contribution)

- 1) Pre-processed and Standardized the dataset
- 2) Implemented Random and Xavier weight initialisation functions
- 3) Implemented Forward Propagation and the associated functions
- 4) Implemented forward\_backward() function which processes a minibatch and outputs gradients and loss.
- 5) Implemented mini\_batch\_generation() and mini\_batch\_gradient\_descent() functions
- 6) Configured wandb sweep and agent for hyperparameter search
- 7) Included the required plots in the report and wrote inferences

Shubham Mallik Thakur(MA20M022) : (50% contribution)

- 1) Implemented Backpropagation
- 2) Implemented Adam, Momentum, Nesterov, RMS Prop, Nadam

- 3) Made plots for confusion matrix, squared error vs cross entropy.
- 4) Included the required plots in the report and wrote inferences.
- 5) Implemented helper functions for train-split, prediction accuracy, losses.

We, Argha Boksi and Shubham Mallik Thakur, swear on our honour that the above declaration is correct.

Date: 25-02-2022

Created with  on Weights & Biases.

[https://wandb.ai/shubham-argha/ma20m022\\_cs21d407/reports/Assignment-1--VmIldzoxNTgyNjMw?accessToken=p0uqhb1md1thrc3hq4c3c3sehum7bg5ln0h29loqp5njvn1tm79uc1o2inbmtsib](https://wandb.ai/shubham-argha/ma20m022_cs21d407/reports/Assignment-1--VmIldzoxNTgyNjMw?accessToken=p0uqhb1md1thrc3hq4c3c3sehum7bg5ln0h29loqp5njvn1tm79uc1o2inbmtsib)

---

Made with Weights & Biases. [Sign up](#) or [log in](#) to create reports like this one.