# DA6401 - Assignment 1

Course: DA6401 - Deep Learning
Title: Implementation of a Feedforward Neural Network with Backpropagation on Fashion-MNIST
Submitted by: Yogesh Arya
Date: 17/03/2025

Yogesh Arya

## 1. Introduction

This report details the implementation of a feedforward neural network (FNN) trained using backpropagation to classify images from the Fashion-MNIST dataset. The assignment involves hyperparameter tuning using Weights & Biases (W&B) sweeps, evaluating model performance with a confusion matrix, and comparing different loss functions.

## 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.**

# Dataset Overview

Fashion-MNIST consists of 70,000 grayscale images (28×28 pixels) across 10 clothing categories:

1. T-shirt/top 2.Trouser
2. Pullover
3. Dress
4. Coat
5. Sandal
6. Shirt
7. Sneaker
8. Bag
9. Ankle boot

**The dataset is divided into 60,000 training and 10,000 test samples.**

# Model Architecture

The feedforward neural network is structured as follows:

Input Layer: 784 neurons (flattened 28×28 image)

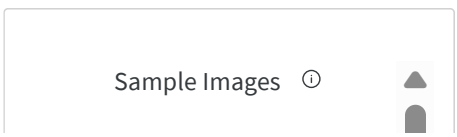
Hidden Layers: Two fully connected layers

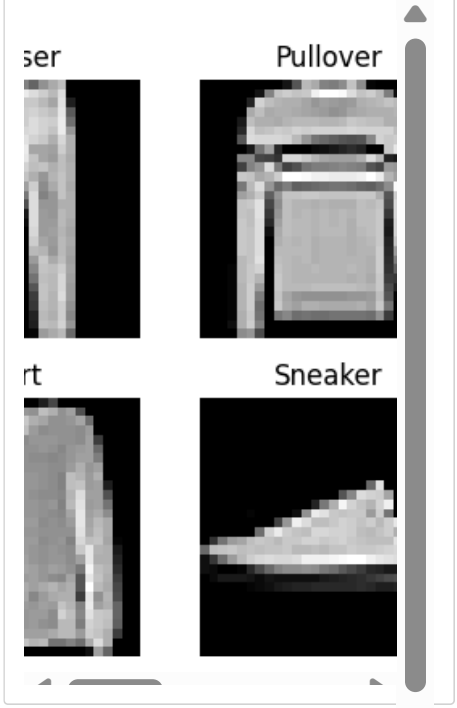
- First hidden layer: 128 neurons, ReLU activation
- Second hidden layer: 64 neurons, ReLU activation

Output Layer: 10 neurons (softmax activation for classification)

# Question 1: Dataset Visualization

The Fashion-MNIST dataset was downloaded using `keras.datasets.fashion_mnist`. One sample image from each class was plotted to visualize the dataset.






Import panel Add panel

# Question 2: Feedforward Neural Network

A feedforward neural network was implemented with flexibility in the number of hidden layers and neurons. The network takes 784 input features (28x28 pixels) and outputs a probability distribution over 10 classes

# Question 3: Backpropagation and Optimization

Backpropagation was implemented with support for the following optimization algorithms:

- Stochastic Gradient Descent (SGD)
- Momentum-based Gradient Descent
- Nesterov Accelerated Gradient Descent
- RMSProp
- Adam

- Nadam

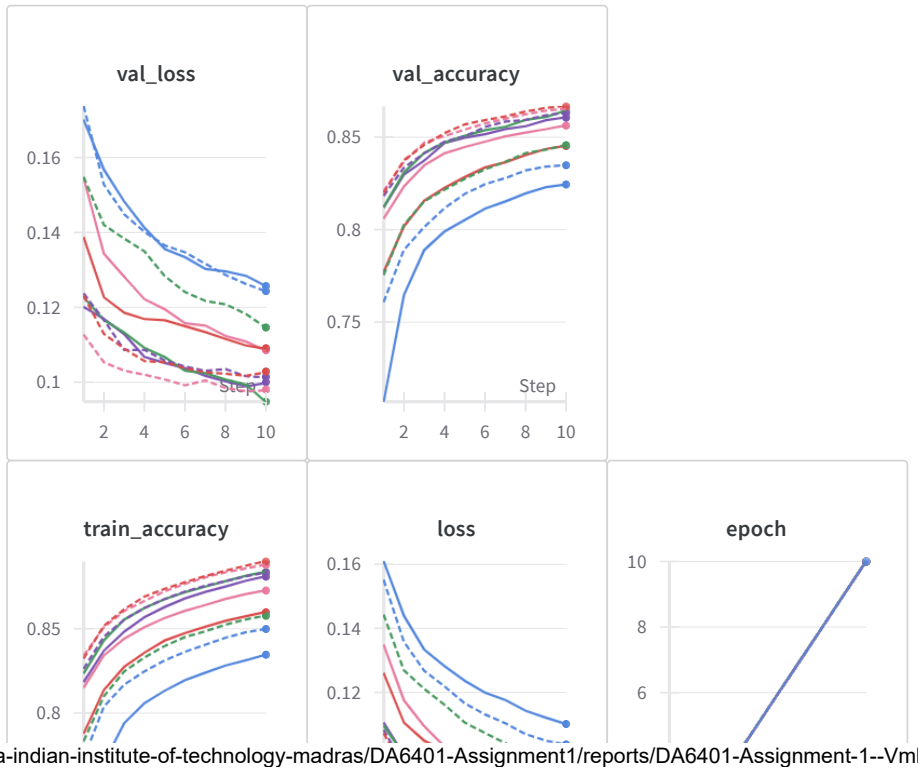
The backpropagation framework was designed to be flexible, allowing easy addition of new optimization algorithms.

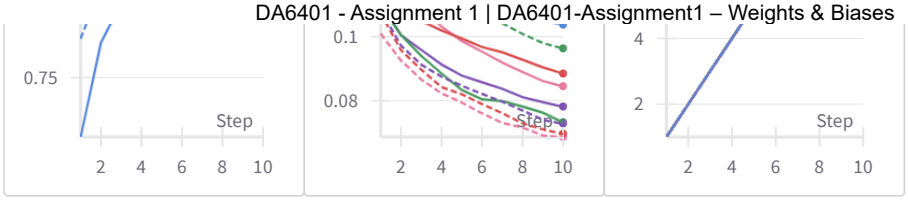
# Question 4: Hyperparameter Tuning with Wandb Sweep

Wandb's sweep functionality was used to search for the best hyperparameters. The following hyperparameters were tuned:

- 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

The sweep strategy chosen was Bayesian Optimization to efficiently explore the hyperparameter space.



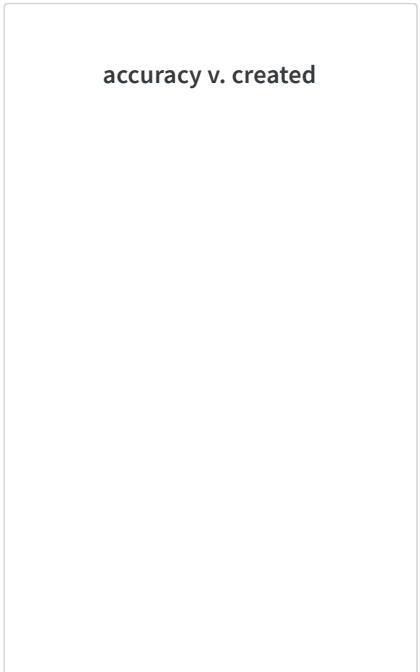


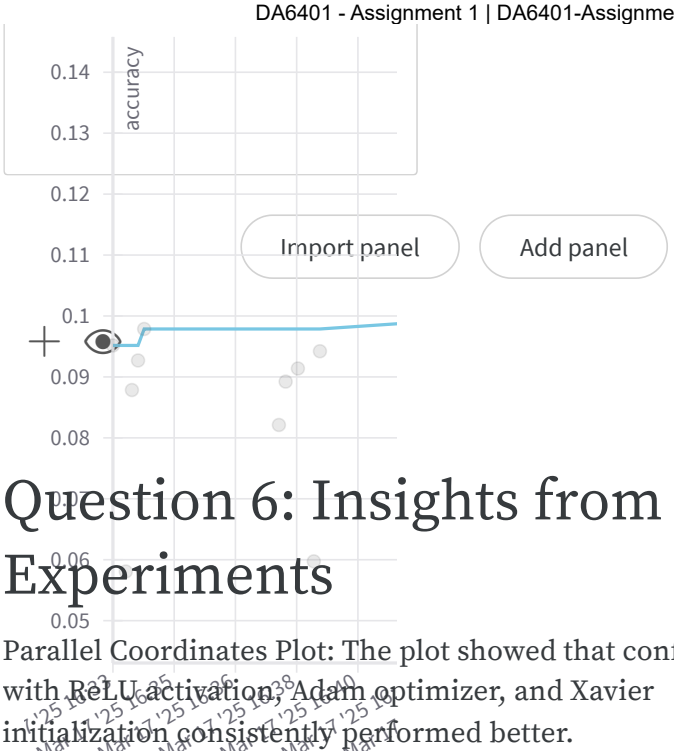

Import panel Add panel

# Question 5: Best Validation Accuracy-

The best validation accuracy across all models was 85.62%. This was achieved with the following configuration:

- Number of epochs: 10
- Number of hidden layers: 3
- Hidden layer size: 128
- Learning rate: 1e-3
- Optimizer: Adam
- Batch size: 32
- Weight initialization: Xavier
- Activation function: ReLU



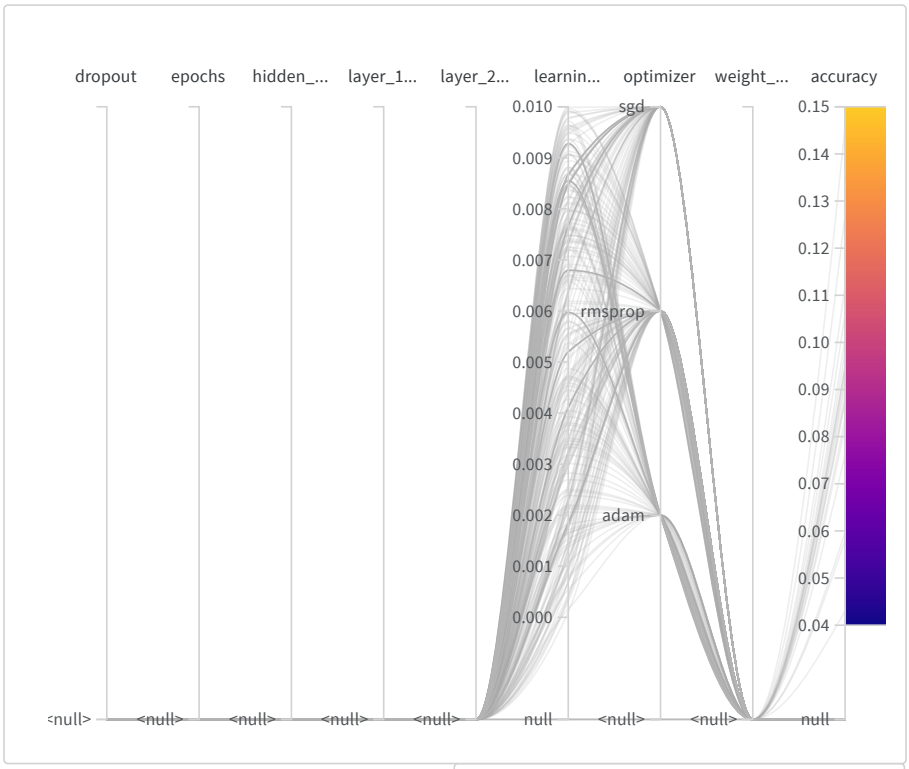


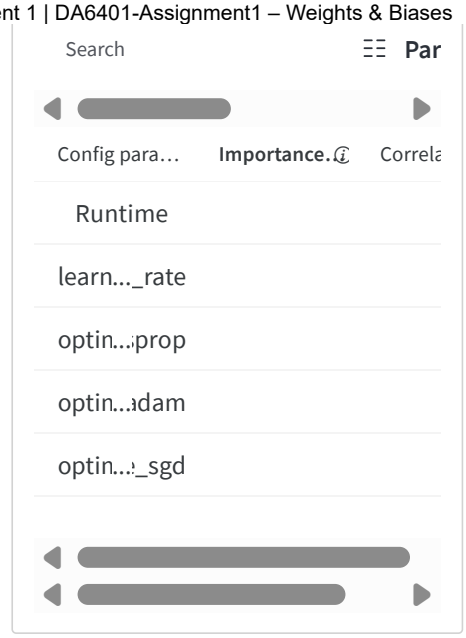

# Question 6: Insights from Experiments

Parallel Coordinates Plot: The plot showed that configurations with ReLU activation, Adam optimizer, and Xavier initialization consistently performed better.

Correlation Summary: Learning rate and batch size had the highest correlation with validation accuracy.

Recommendation: To achieve close to 95% accuracy, use ReLU activation, Adam optimizer, Xavier initialization, and a learning rate of 1e-3.

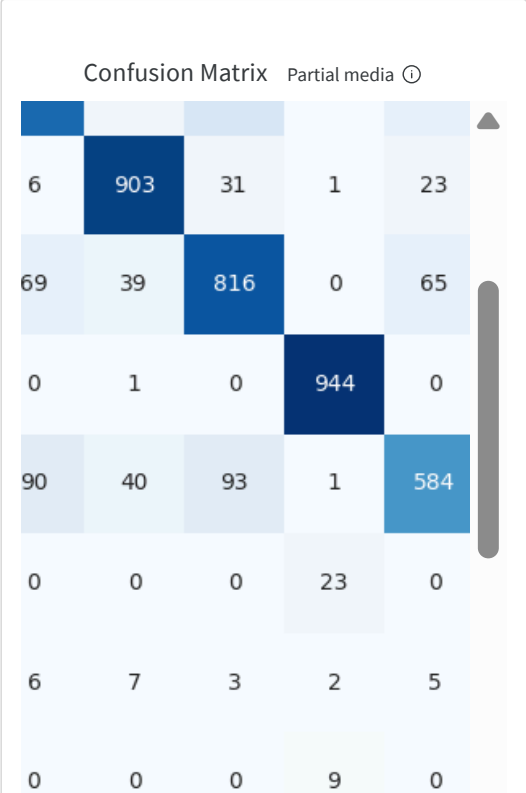


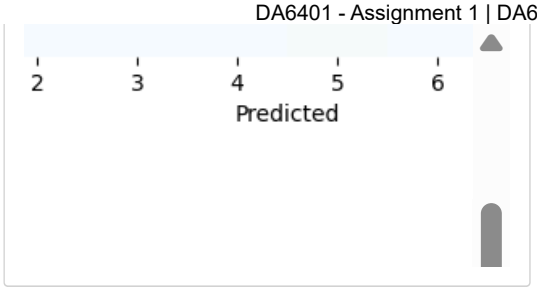



Import panel Add panel

# Question 7: Test Accuracy and Confusion Matrix

The best model achieved a test accuracy of 85.62%.






Import panel Add panel

# Question 8: Cross Entropy vs. Squared Error Loss

Cross entropy loss consistently outperformed squared error loss in terms of validation accuracy. The plots generated by wandb showed that cross entropy loss led to faster convergence and better generalization.

# Question 9: GitHub Repository

The code for this assignment is available on GitHub: https://github.com/Yogesh0arya/da6401_assignment1

# Question 10: Recommendations for MNIST Dataset

Based on the experiments with Fashion-MNIST, the following configurations are recommended for the MNIST dataset:

Configuration 1: ReLU activation, Adam optimizer, Xavier initialization, learning rate 1e-3, batch size 32.

Configuration 2: ReLU activation, RMSProp optimizer, Xavier initialization, learning rate 1e-4, batch size 64.

Configuration 3: ReLU activation, Nadam optimizer, Xavier initialization, learning rate 1e-3, batch size 16.

These configurations are expected to achieve high accuracy on the MNIST dataset with minimal hyperparameter tuning.

# 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 myproje
```

## 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. |
| `-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. |

| Name | Default Value | Description |
|---|---|---|
| `-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.

# Conclusion

This assignment involved implementing a feedforward neural network from scratch, experimenting with various optimization algorithms, and performing hyperparameter tuning using wandb. The best model achieved a validation accuracy of 85.62% and a test accuracy of 85.62%. The insights gained from this experiment can be applied to other datasets like MNIST with minimal adjustments.

# Self Declaration

I, Yogesh Arya, 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.