Deep Learning Assignment 1 Report

Name: P. Varun Reddy

Class: IT-3, Sem-VI

Roll No: 160122737192

Date: 03 March 2025

Github link: https://github.com/varunreddy70/DL-Assignment1-MNIST-FNN/

1. Introduction

This report presents the implementation and analysis of a Feedforward Neural Network (FNN) for handwritten digit classification using the MNIST dataset. The primary objectives include:

- Constructing a flexible neural network with adjustable parameters.
- Training the model with various optimizers and hyperparameters.
- Evaluating the impact of different loss functions.
- Analyzing and comparing results for model selection.

2. Dataset Overview

The MNIST dataset consists of:

- 60,000 training images and 10,000 test images.
- Images are grayscale (28x28 pixels), representing digits (0-9).
- Labels are categorical (0-9), and data is normalized to the range [0,1].

3. Model Architecture

A feedforward neural network with the following architecture was implemented:

- Input Layer: Flattening the 28x28 image into 784 features.
- Hidden Layers: Configurable (3, 4, or 5 layers) with ReLU activation.
- Neurons per Layer: 32, 64, or 128 neurons.
- Output Layer: Softmax activation for classification.
- Weight Initialization: Xavier (Glorot) or Random Normal.

4. Training and Hyperparameter Tuning

4.1 Optimizers Used:

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

4.2 Hyperparameter Variations:

- Epochs: 5, 10
- Batch Size: 32, 64
- Learning Rate: 0.001
- Weight Decay (L2 Regularization):0.0005
- Activation Functions: ReLU

5. Model Evaluation & Results

5.1 Performance Metrics:

- Validation Accuracy: Tracked across all models.
- Confusion Matrix: Generated for the best model.
- Loss Comparison: Cross-Entropy vs. Squared Error Loss.

5.2 Best Model Performance:

- Configuration: 3 Hidden Layers, 128 Neurons per Layer, Xavier Initialization, Adam Optimizer.
- Test Accuracy: 97.67%
- Loss: 0.0934 (Cross-Entropy Loss)

5.3 Loss Function Comparison:

Loss Function	Accuracy	Loss
Cross-Entropy Loss	97.67%	0.0934
Squared Error Loss	97.30%	0.0046

Cross-entropy loss is the standard choice for classification tasks because it directly optimizes probability distributions by minimizing the difference between predicted and actual class probabilities. It is particularly effective when dealing with categorical labels since it helps the model assign higher confidence to the correct class.

On the other hand, squared error loss, commonly used in regression tasks, treats classification as a continuous value prediction problem. While it still works for classification, it does not handle probability distributions as effectively as cross-entropy. The results confirm that squared error loss led to slightly lower accuracy and a much smaller loss value, which does not always indicate better performance in a classification context.

Overall, cross-entropy is preferred for classification due to its strong probabilistic foundation, whereas squared error loss is more suitable for continuous target values in regression models.

6. Results and Observations

The model was trained with different configurations, and the test accuracy was recorded.

Hidden Layers	Learning Rate	Batch Size	Optimizer	Accuracy
[64, 128, 256]	0.001	32	Adam	97.67%
[64, 128, 256]	0.001	32	SGD	92.34%
[64, 128, 256]	0.001	64	RMSprop	95.12%

From the results, the model performed best with the **Adam optimizer**, a **learning rate of 0.001**, and a **batch size of 32**, achieving **97.67% accuracy**. Using **SGD** with the same configuration (learning rate of 0.001 and batch size of 32) resulted in significantly lower performance, with an accuracy of **92.34%**. **RMSprop** performed better than SGD, achieving **95.12% accuracy** with a batch size of 64, but it was still slightly less effective than Adam. Overall, Adam consistently provided the best performance across all configurations.

7. Conclusion and Recommendations

7.1 Key Findings:

- Adam and RMSprop optimizers provided the best performance.
- More neurons and layers improved accuracy but also risked overfitting.
- Cross-Entropy Loss was more effective than Squared Error Loss for classification.

7.2 Best Recommendations for MNIST:

1. Use 3-5 hidden layers with 128 neurons per layer.
2. Use Adam optimizer with a learning rate of 0.001.
3. Use Cross-Entropy Loss for classification.
4. Use a batch size of 32 for optimal performance.

8. References

- MNIST Dataset Description
- TensorFlow & Keras Documentation
- Deep Learning Course Material
- PyTorch documentation