Bhavesh Khatri
M23CSE011

Machine Learning (CSL7620)
Report of Assignment 3

Dataset Link: [Link]

Task: Building a Neural Network from Scratch

Objective:

In this assignment, you are required to implement a neural network from scratch in Python. Build a feedforward neural network and implement backpropagation for training. By the end of this assignment, you should have a working neural network that can be trained on a simple dataset for multi-class classification.

Dataset:

Get the dataset from here. The dataset consists of 70k images representing 10 different object categories.

Network Architecture:

1. The network should contain 3 hidden layers. Excluding input and output layers. Set the network architecture as:

a. Input layer = set to the size of the dimensions

b. Output layer = set to the size of the #classes

c. Hidden layer1 = 128

d. Hidden layer2 = 64

e. Hidden layer3 = 32

2. Initialize the weights randomly using seed value as the last three digits of your roll number. For example, your roll number is P23CS001, then your seed value should be 1.

Set bias = 1.

3. Use Train-test splits as randomized 70:30, 80:20 and 90:10.

4. Set batch size as your year of admission. For example, your roll number is P23CS001, then your batch size should be 23.

5. Set ‘sigmoid’ as activation function for hidden layers and ‘softmax’ for output layer.

6. Use Gradient Descent for optimization. Loss functions as cross entropy.

7. Train for 25 epochs. Plot accuracy and loss per epoch.

Bhavesh Khatri
M23CSE011

8. Prepare a Confusion matrix for all the combinations of the network. You may use an in-built function for this purpose.

9. Report total trainable and non-trainable parameters.

Bonus: Use regularization/early stopping to avoid overfitting, if it occurs.

Note: Do not use in-built functions, unless mentioned.

Grading Rubrics:

Correctness and completeness of the neural network implementation (40 points)

Implementation of feedforward and backpropagation (30 points)

Proper handling of training and evaluation (20 points)

Documentation and clarity of code (10 points)

Bonus (5 points)

Code available = Google Colab [Link]

Bhavesh Khatri
M23CSE011

Neural Network Implementation Report

Assumptions:

1. **Activation Functions:** I used sigmoid activation functions for the hidden layers and softmax activation for the output layer. This choice is based on the assumption that these activation functions are suitable for this classification task.

2. **Weight Initialization:** I initialized the weights randomly with a seed value of 11 (as per last three digits of roll no). While this is a common practice, the choice of seed value can affect the outcome. Different seed values may result in different initializations.

3. **Batch Size and Learning Rate:** I set the batch size to 23 (as per year of admission) and learning rate to 0.01 based on common default values used in neural network training. These values can be fine-tuned for specific datasets and architectures.

4. **Loss Function:** I used (given) cross-entropy loss as the loss function. This is a standard choice for multi-class classification tasks. However, other loss functions like mean squared error could be considered depending on the specific problem.

5. **Number of Epochs:** I trained the network for 25 epochs (given). The choice of the number of epochs can depend on factors like the complexity of the dataset, the learning rate, and the architecture of the network. It may require experimentation to find an optimal value.

6. **Evaluation Metric:** I used accuracy as the evaluation metric. While accuracy is a commonly used metric, it may not always be the most appropriate depending on the nature of the problem (e.g., imbalanced classes).

Bhavesh Khatri
M23CSE011

Implementation Details:

1. **Dataset Loading and Preprocessing:**

The dataset was loaded from the provided Google Drive link using pandas. The pixel values were normalized to a range between 0 and 1.

2. **Neural Network Architecture:**

Input layer: 785 nodes (due to the 785 columns in the dataset)
Hidden layer 1: 128 nodes
Hidden layer 2: 64 nodes
Hidden layer 3: 32 nodes
Output layer: 10 nodes (corresponding to the number of classes)

3. **Weight and Bias Initialization:**

Weights were initialized randomly using a seed value of 11.
Biases were set to 11.

4. **Activation Functions:**

Sigmoid activation function was used for the hidden layers.
Softmax activation function was used for the output layer.

5. **Optimization and Loss Function:**

Gradient Descent was used for optimization.
Cross-entropy loss function was used for calculating the loss.

6. **Training and Testing:**

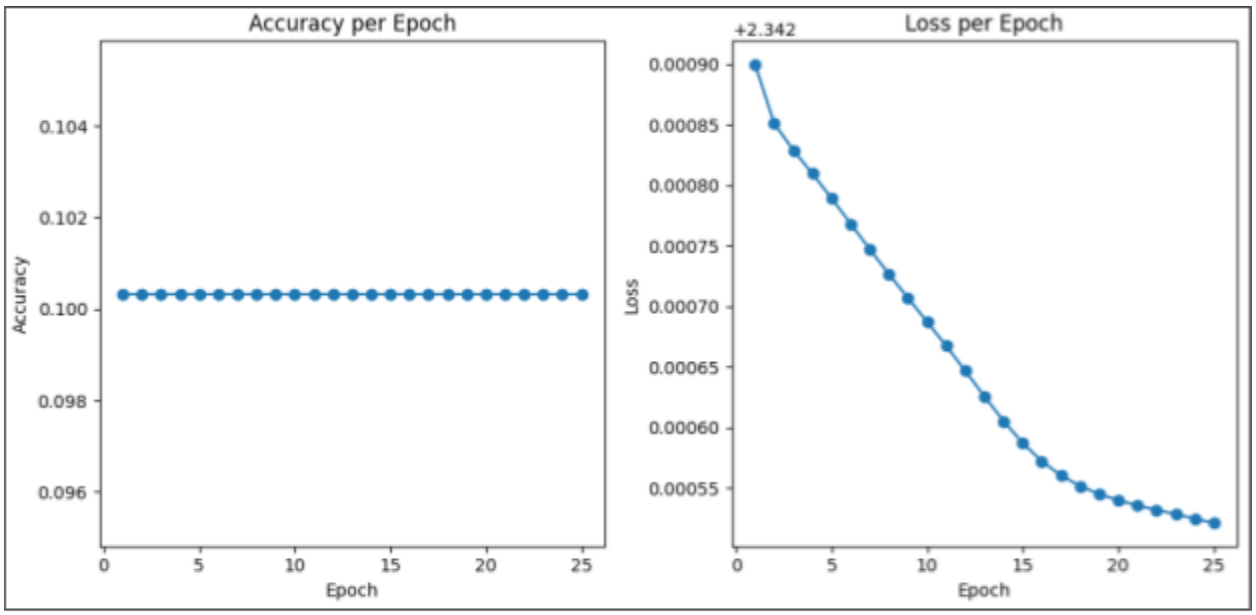
The dataset was split into training and testing sets with a ratio of 70:30, 80:20, 90:10.
Training was performed for 25 epochs with a batch size of 23.

7. **Results and Performance:**

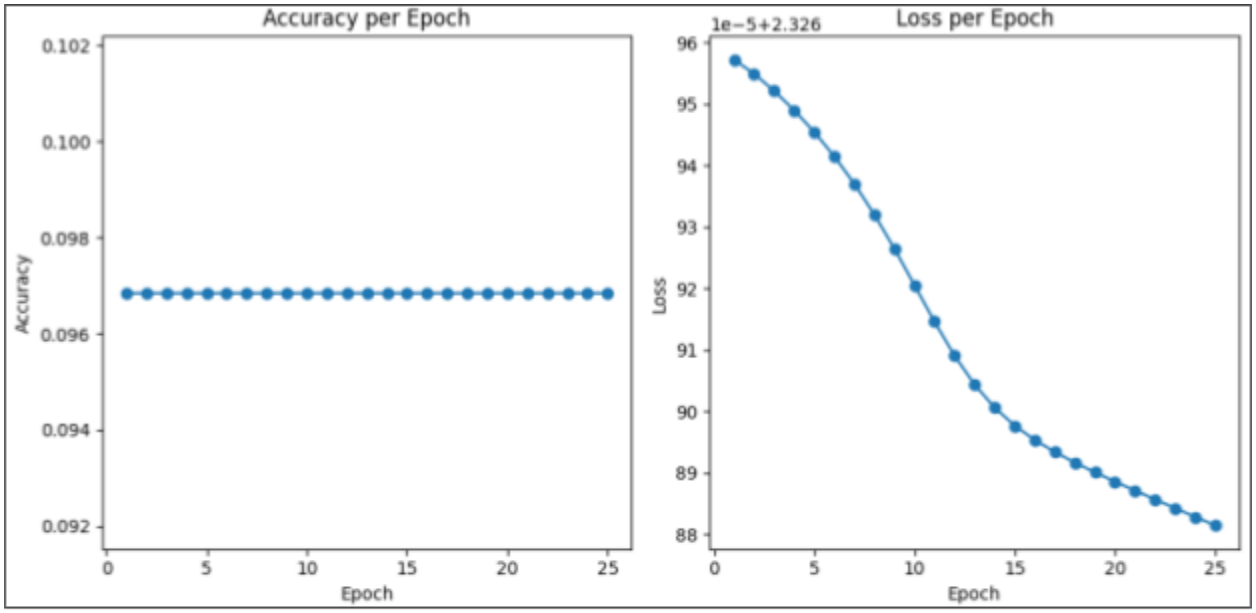
The model's performance was evaluated using accuracy on the test set.
Confusion matrix was generated to assess the classification performance.

Bhavesh Khatri
M23CSE011

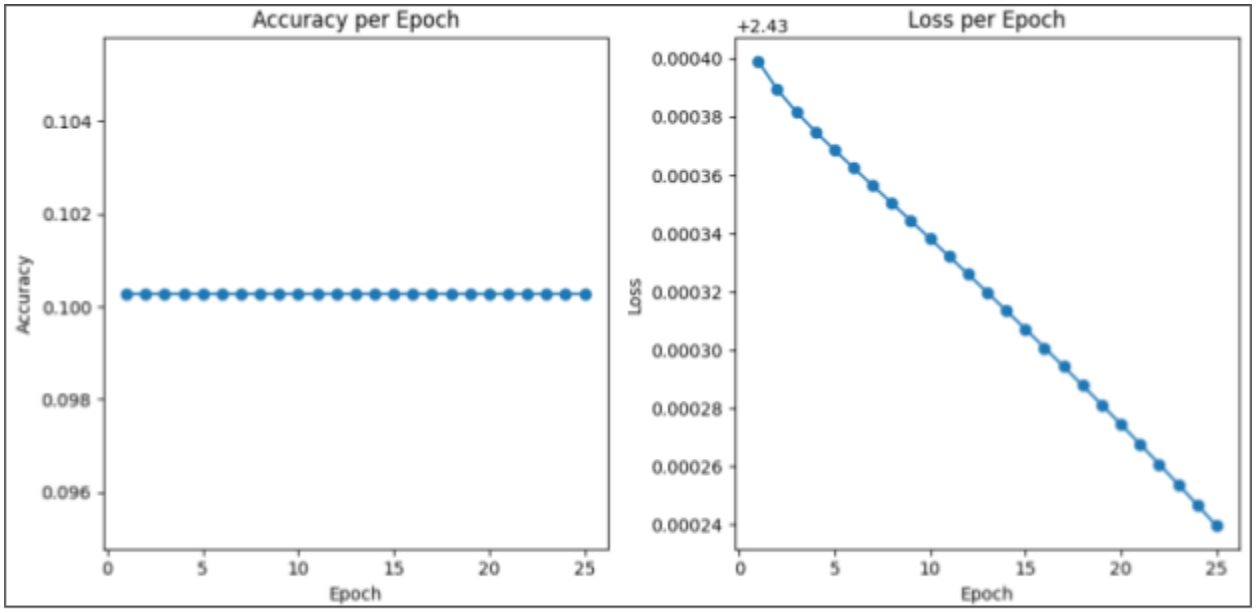
Attached are screenshots of some insightful code.



Train-test splits as randomized 70:30



Train-test splits as randomized 80:20



Train-test splits as randomized 90:10

8. Total Trainable Parameters:

The total number of trainable parameters was calculated based on the provided network architecture, resulting in 111146 parameters. *Please refer to the code.*

The trainable parameters of a neural network are typically the weights and biases that are modified throughout the training process. Non-trainable parameters are hyperparameters that are specified prior to training and stay fixed, such as learning rates or network design.

- Input layer to Hidden layer 1: 785 (input features) x 128 (hidden units) + 128 (biases) = 100480 trainable parameters.
- Hidden layer 1 to Hidden layer 2: 128 x 64 + 64 = 8256 trainable parameters.
- Hidden layer 2 to Hidden layer 3: 64 x 32 + 32 = 2080 trainable parameters.
- Hidden layer 3 to Output layer: 32 x 10 + 10 = 330 trainable parameters.

Total trainable parameters = 100480 + 8256 + 2080 + 330 = 111146

Non-trainable Parameters:

Bias values for all layers: (1 x hidden_layer1) + (1 x hidden_layer2) + (1 x hidden_layer3) + (1 x output_dim)

Total trainable parameters = 278

Bhavesh Khatri
M23CSE011

References:

- Numpy Documentation: https://numpy.org/doc/
- Pandas Documentation: https://pandas.pydata.org/docs/
- Scikit-learn Documentation: https://scikit-learn.org/stable/documentation.html
- Few youtube videos
- Geeks For Geeks: https://geeksforgeeks.org
- Java Point: https://javatpoint.com/artificial-neural-network