DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)
Shahbad Daulatpur, Bawana Road, Delhi 110042

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CO328 : DEEP LEARNING

LAB FILE

Submitted To:
Prof. Anil Singh Parihar
Department of Computer Science
And Engineering

Submitted By:
Ayush Gupta
2K22/CO/127
3rd Year (SEM - VI)

EXPERIMENT 1

EXPERIMENT OBJECTIVE

To implement a Fully Connected Neural Network (FCNN) for classifying handwritten digits from the MNIST Dataset using NumPy.

DATA PREPROCESSING

Loading the MNIST Dataset

- The dataset is loaded from binary files containing images and labels.
- The images are 28x28 grayscale images, reshaped into a (784,) vector.
- The labels are converted into one-hot encoded vectors.

Data Augmentation

- **Random Rotation**: Images are rotated within a range of -15° to 15° with a 50% probability.
- **Horizontal Flip**: Images have a 50% chance of being flipped horizontally.

Splitting the Dataset

- The dataset is divided into training, validation, and test sets.
- The training set is further split into 80% training and 20% validation.

NEURAL NETWORK IMPLEMENTATION

Architecture

- **Input Layer**: 784 neurons (28x28 pixels flattened)
- **Hidden Layer 1**: 256 neurons, ReLU activation
- **Hidden Layer 2**: 128 neurons, ReLU activation
- **Output Layer**: 10 neurons (digits 0-9), softmax activation

Weight Initialization

- Weights are initialized using He initialization.
- Biases are initialized to zeros.

Activation Functions

- **ReLU (Rectified Linear Unit)**: Used in hidden layers.
- **Softmax**: Applied to the output layer for probability distribution.

Regularization

- **Dropout**: Randomly drops activations during training to prevent overfitting.
- **Gradient Clipping**: Limits gradient values to avoid exploding gradients.

TRAINING CONFIGURATION

Training the Model

- **Loss Function**: Cross-entropy loss is used.
- **Optimizer**: The model updates weights using backpropagation and gradient descent.
- **Learning Rate**: 0.005 (with decay over time)
- **Drop Out**: 0.2
- **Epochs**: Trained for 5000 epochs.
- **Batch Processing**: Mini-batch gradient descent is implemented.
- **Best Model Selection**: Saves weights of the best-performing model (lowest validation loss).
- **Training Time**: 3h 53m 19s

Model Checkpointing

- The best model weights (based on validation loss) are saved periodically to **best_weights.npy**.

TRAINING AND VALIDATION RESULTS

Key Performance Metrics from Training Output

Epoch	Training Loss	Validation Loss	Accuracy (%)
0	2.2431	2.2489	23.38%
51	0.9854	1.0010	67.47%
213	0.6239	0.6056	79.03%
506	0.3754	0.3681	87.50%
1006	0.1675	0.1687	95.06%
1570	0.0801	0.0769	97.97%
2891	0.0152	0.0136	99.82%
3316	0.0118	0.0102	99.78%

4654	0.0029	0.0027	99.95%
4934	0.0022	0.0018	99.96%

Evaluation Results

- After training, the best model weights are loaded and tested on unseen test data.
- **Final Test Accuracy**: ~96.88%
- **Final Test Loss**: ~0.165

MODEL SAVING AND LOADING

- **Saving Weights**: The best model weights (lowest validation loss) are saved to disk.
- **Loading Weights**: Enables reloading the best weights for inference or further training.

RESULTS AND CONCLUSIONS

- The model achieves high accuracy using a simple fully connected architecture.
- Data augmentation and regularization significantly improve generalization.
- The saved best weights allow for consistent reproducibility of results.