Design and Implementation of a Fully Connected Feed-forward Neural Network using TensorFlow.Keras

Student Name

August 5, 2025

Abstract

This report presents the design and implementation of a Fully Connected Feed-forward Neural Network (FCFNN) using TensorFlow.Keras framework. The neural network is applied to the MNIST handwritten digit classification task. The architecture, implementation details, and experimental results are discussed. The complete implementation is available in a Google Colab notebook, the link to which is provided in this report.

1 Introduction

Neural networks have become fundamental tools in machine learning and artificial intelligence. Among various architectures, Fully Connected Feed-forward Neural Networks (FCFNN) are the simplest form, where each neuron in one layer is connected to all neurons in the next layer [1].

The MNIST dataset, consisting of 60,000 training images and 10,000 testing images of handwritten digits (0-9), serves as a benchmark for evaluating the performance of neural networks [2]. In this report, we design, implement, and evaluate an FCFNN for classifying these handwritten digits.

2 Neural Network Architecture

The designed FCFNN consists of the following layers:

1. **Input Layer**: A flattened input of 784 neurons, corresponding to the 28×28 pixel images from the MNIST dataset.
2. **First Hidden Layer**: Comprises 3 neurons with ReLU (Rectified Linear Unit) activation function.
3. **Second Hidden Layer**: Contains 2 neurons with sigmoid activation function.
4. **Output Layer**: Consists of 1 neuron with softmax activation function.

The architecture is implemented using TensorFlow.Keras, which provides a high-level API for building and training neural networks. The model is defined as follows:

- Input shape: (784,)
- Hidden layers: Dense layers with specified activation functions
- Output layer: Dense layer with softmax activation

3 Implementation Details

The implementation involves several key steps:

3.1 Data Preprocessing

The MNIST dataset is loaded and preprocessed before training the neural network:

- Images are reshaped from 28×28 matrices to 784-dimensional vectors.
- Pixel values are normalized to the range [0, 1] by dividing by 255.0.
- Labels are converted to one-hot encoded vectors (e.g., digit 3 becomes [0, 0, 0, 1, 0, 0, 0, 0, 0, 0]).

3.2 Model Compilation

The model is compiled with the following configuration:

- **Optimizer**: Adam, an adaptive learning rate optimization algorithm.
- **Loss Function**: Categorical crossentropy, suitable for multi-class classification.
- **Metrics**: Accuracy, to monitor the classification performance during training.

3.3 Training and Evaluation

The model is trained using the following parameters:

- Epochs: 5 complete passes through the training dataset.
- Batch size: 32 samples per gradient update.
- Validation split: 10% of the training data used for validation.

After training, the model is evaluated on the test set to measure its generalization performance.

4 Results and Analysis

The model's performance is evaluated based on the test accuracy. Additionally, the weights and biases of the first dense layer are extracted to analyze the learned parameters.

The model summary provides information about the architecture, including the number of parameters in each layer and whether they are trainable.

5 Conclusion

This report demonstrated the design and implementation of a Fully Connected Feed-forward Neural Network for the MNIST digit classification task using TensorFlow.Keras. The model architecture, though simple, illustrates the fundamental concepts of neural networks.

The complete implementation, including all code and experimental results, is available in the Google Colab notebook linked below. This allows for reproducibility and further exploration of the implemented neural network.

6 Code Availability

The complete implementation of this neural network is available in the following Google Colab notebook:

`https://colab.research.google.com/drive/1VfwaHZ0aC0vAAJop8G7nUWlqaon21k7u?usp=sharing`

References

[1] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning.* MIT Press.

[2] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.