Neural Network Architecture for Digit Sum Prediction Using CNN

Pavan Karke

January 2025

Technical Report

1 Introduction

This report details the implementation of a deep learning model designed to process images containing multiple digits and predict their sum. The architecture combines convolutional neural networks (CNN) for digit recognition with a subsequent sum prediction network, creating a two-stage prediction system.

2 Data Processing and Loading

The implementation uses a custom `DigitSumDataset` class inheriting from PyTorch's `Dataset` class. Key features of the data pipeline include:

- Support for multiple data files (`data0.npy`, `data1.npy`, `data2.npy`) allowing for dataset expansion.
- Image preprocessing pipeline using PyTorch transforms:
  - Resizing images to 40×168 pixels.
  - Normalization with mean = 0.5 and std = 0.5.
  - Conversion to PyTorch tensors.
- Grayscale image input handling with a single channel.

3 Model Architecture

3.1 1. Digit Recognition Network (`MNISTDigitModel`)

The first stage employs a scalable CNN architecture with the following characteristics:

- Variable number of convolutional blocks (currently set to 5).
- Each block contains:
  - Two consecutive Conv2D layers with same padding.

- ReLU activation.
- MaxPool2D for dimensionality reduction.
- Dropout for regularization.

- Progressive channel expansion (doubles after each block).
- Final fully connected layers reducing to 40 outputs (4 digits × 10 classes).

3.2 2. Sum Prediction Network (`MNISTSumModel`)

The second stage implements a Multi-Layer Perceptron (MLP) that:

- Takes softmax probabilities from digit recognition.
- Processes through multiple dense layers ($64 \to 128 \to 64 \to 1$).
- Outputs final sum prediction.

3.3 3. Combined Architecture

The two networks are integrated through a `CombinedModel` class that:

- Processes input images through the digit recognition network.
- Feeds probability distributions to the sum prediction network.
- Outputs both digit predictions and final sum.

4 Model Robustness Techniques

Several techniques have been implemented to enhance model robustness:

4.1 Regularization

- Dropout layers (dropout rate: 0.001) to prevent overfitting.
- Multiple dense layers in sum prediction to allow for complex relationships.

4.2 Training Optimization

- Adam optimizer with a conservative learning rate (1×10^{-5}).
- Batch normalization through input normalization.

4.3 Architecture Design

- Skip connections in the form of multiple dense layers.
- Progressive channel expansion in convolutional layers.

5 Performance Analysis

The model's performance on the test dataset shows both strengths and areas for improvement:

5.1 Quantitative Metrics

- Mean Squared Error (MSE): 5.1791.
- Accuracy: 25.93%.

5.2 Sample Predictions Analysis

Examples of model predictions:

- Actual: 27, Predicted: 26.77 (Good approximation).
- Actual: 19, Predicted: 18.93 (Good approximation).
- Actual: 22, Predicted: 17.58 (Larger error).
- Actual: 13, Predicted: 16.57 (Moderate error).

5.3 Performance Summary

- The model shows good approximation capabilities in some cases, with predictions often close to actual values.
- The relatively high MSE (5.1791) indicates significant variance in prediction accuracy.
- The accuracy of 25.93% on exact matches suggests room for improvement in precise predictions.