CNN Implementation for MNIST Digit Recognition- A Brief Report

Rahul Bhandari
University of South Dakota
Student ID: 101145564
rahul.bhandari@coyotes.usd.edu

Abhaya Rawal
University of South Dakota
Student ID: 101145746
abhaya.rawal@coyotes.usd.edu

Praveen Raghubanshi
University of South Dakota
Student ID: 101141342
praveen.raghubanshi@coyotes.usd.edu

May 1, 2024

1 Introduction

This report elaborates on the implementation of a Convolutional Neural Network (CNN) for the MNIST dataset, specifically focusing on the architecture, data preparation, training, and evaluation processes.

2 Dataset and Preprocessing

The MNIST dataset is a collection of 70,000 grayscale images of handwritten digits (0-9), each 28x28 pixels. For this project, we use a preprocessed version available at `https://archive.ics.uci.edu/ml/dataset` which simplifies the integration and grading process.

2.1 Data Loading

Data is loaded directly from the UCI repository using a custom function tailored to handle the dataset structure.

2.2 Preprocessing

Preprocessing steps include:

- Normalization: Pixel values are scaled from [0, 255] to [0, 1] to aid in neural network performance.
- Reshaping: Images are reshaped from a 1D array of 784 elements to a 2D array of 28x28 pixels to maintain spatial hierarchy for convolution operations.

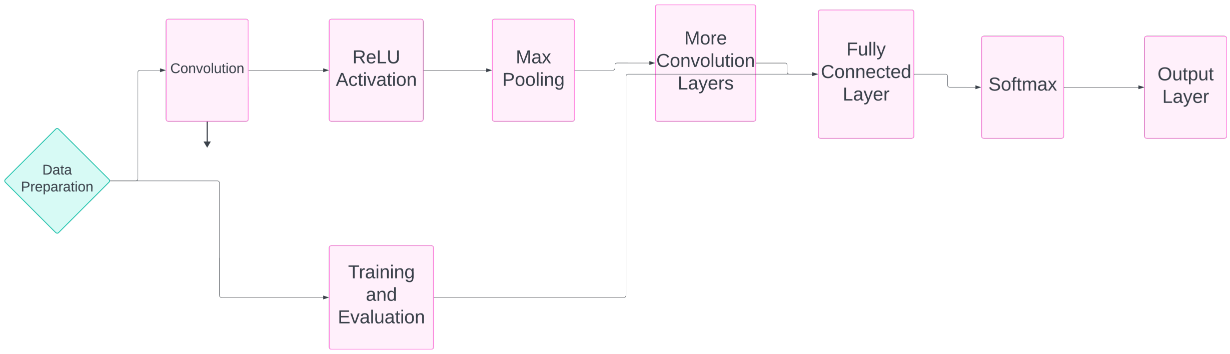

Figure 1: Architectural Diagram of CNN

3 Convolutional Neural Network Architecture

Our CNN comprises 3 to 5 convolutional layers, each followed by a max pooling layer. Below is the detailed architecture:

3.1 Layer Configuration

- Convolutional Layer 1: Applies 32 filters of 3x3 with ReLU activation, producing a feature map of size 26x26 (due to the absence of padding).
- Max Pooling Layer 1: Reduces feature map size to 13x13 using a 2x2 pooling filter.
- Further layers continue this pattern, increasing the number of filters while reducing feature map dimensions.

3.2 Mathematical Model of Convolution

The convolutional layer transforms the input volume into an output volume through a set of learnable filters. Each filter is small spatially (along width and height), but extends through the full depth of the input volume. For each region of the input, the dot product between the filter and the input is computed. Mathematically, this can be described as:

$$Z = \sum_{i=1}^{F_H} \sum_{j=1}^{F_W} I_{i,j} \cdot K_{i,j} \tag{1}$$

Where I is the input, K is the kernel, F_H and F_W are the height and width of the filter, respectively.

3.3 Architectural Diagram

4 Max Pooling

Each convolutional layer is followed by a max pooling layer, which reduces the spatial size of the representation, decreases the number of parameters, and the amount of computation in the network, and helps prevent overfitting. The operation performed by a max pooling layer is:

$$P_{x,y} = \max(I_{sub}) \tag{2}$$

Where I_{sub} is a subsection of the input matrix and P is the output matrix from the pooling layer.

5 Fully Connected Layer and Softmax

After the final pooling layer, the feature map is flattened into a vector and fed into a fully connected layer. This layer connects every neuron in one layer to every neuron in the next layer, facilitating the final classification.

$$S_i = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{3}$$

Where S represents the softmax output probabilities, and z_i are the inputs to the softmax function from the last fully connected layer.

6 Training and Evaluation

Training involves multiple epochs where the model learns to minimize a loss function (categorical crossentropy) and improve its accuracy. The performance is validated using K-fold cross-validation and visualized through loss and accuracy curves. A confusion matrix is used post-training to analyze the model's performance across different classes.

7 Conclusion

This comprehensive approach to training a CNN on the MNIST dataset illustrates the effectiveness of deep learning techniques in recognizing handwritten digits, offering robust performance and insights into the model's predictive capabilities.