CSE-575
Project 3: Classification Using Neural Networks and Deep Learning

Gaurav Kumar-ASU ID-1229081284
gkumar28@asu.edu

Introduction:

This project comprises of two parts. In **Part1** we evaluate the effects of changing kernel sizes and the number of feature maps in convolutional layers on model accuracy and loss. **Part2** of this project entails building a Convolutional Neural Network (CNN) for classifying a subset of the MNIST dataset based on the last 4 digits of ASU ID. The subset contains 4 distinct categories. We will walkthrough each part in details.

Part1Details (*Demo code was provided*):

1. Baseline Model Configuration:

- Convolutional Layer 1: 6 feature maps, 3x3 kernel size
- Convolutional Layer 2: 16 feature maps, 3x3 kernel size

Results:

Test Loss: 0.06806522607803345
Test Accuracy:0.9799000024795532

2. Changing Kernel Size to 5x5:

- Convolutional Layer 1: 6 feature maps, 5x5 kernel size
- Convolutional Layer 2: 16 feature maps, 5x5 kernel size

Results:

Test Loss: 0.0588434636592865
Testing Accuracy: 0.9810000061988831

Observation: We observed an increase in test accuracy in this step.

3. Changing the Number of Feature Maps:

- Convolutional Layer 1: 12 feature maps, 3x3 kernel size
- Convolutional Layer 2: 32 feature maps, 3x3 kernel size

Results:

Test Loss: 0.056993745267391205
Testing Accuracy: 0.9803000092506409

Overall increase in kernel size and feature maps increases accuracy of the model.

Part2 Details:

The CNN is constructed using basic layers like convolution, max pooling, and fully connected layers, all implemented using the NumPy library.

Model Architecture:

The implemented CNN has the following architecture:

- **Convolutional Layer** (**Convolution2D**):
  - Input Channels: 1 (grayscale image)
  - Number of Filters: 6
  - Kernel Size: 5x5
  - Padding: 0
  - Stride: 2
- **Activation Function**: ReLU
- **Max Pooling Layer** (**Maxpooling2D**):
  - Pool Size: 2x2
  - Stride: 2
- **Flatten Layer**: To flatten the feature maps for the fully connected layer
- **Fully Connected Layer**:
  - Input Neurons: 6 x 6 x 6
  - Output Neurons: 32
- **Activation Function**: ReLU
- **Fully Connected Layer**:
  - Input Neurons: 32
  - Output Neurons: 4 (corresponding to 4 classes)
- **Softmax Layer**: For multi-class classification

The model is trained using the backpropagation algorithm.

Methodology:

1. **Data Loading and Preprocessing**:

- The MNIST subset was loaded based on the given ASU ID '1284'.
- Data was shuffled and preprocessed for input into the network.

2. **Training**:
   - The network was trained in mini-batches.
   - The forward pass and backward pass were executed for each training example, weights were updated accordingly.
3. **Evaluation**:
   - Post training, the network was evaluated on both the training set and the testing set to calculate accuracy and loss.

Results:

- After 10 epochs of training, the following results were obtained:
  - **Training Accuracy**: 0.792
  - **Training Loss**: 0.536
  - **Testing Accuracy**: 0.738
  - **Testing Loss**: 0.653

(Note: The loss value was normalized by dividing by the number of training/testing samples.)

Challenges and Troubleshooting:

- **Shape Mismatch**: During the implementation, there was a shape mismatch issue in the **zero_padding** function. This was addressed by ensuring the input data's shape matched the expected input to the function.

Conclusion:

Through this project, a basic Convolutional Neural Network was successfully implemented using numpy. This emphasizes the understanding of the foundational concepts of CNNs, such as convolution, pooling, and backpropagation, without the abstraction provided by higher-level libraries like TensorFlow or PyTorch. The network was able to classify images from the MNIST dataset subset with reasonable accuracy.

Generated plot: