Instruction Manual

Deep Learning - Programming Assignment 1

February 1, 2025

1 Task 1: CNN From Scratch using NumPy

This task requires implementing a Convolutional Neural Network (CNN) from scratch using NumPy and training it for a simple classification problem. The goal is to achieve high accuracy on an 8x8 single-channel handwritten digit dataset.

2 Getting Started

Before starting, ensure you have all required Python packages installed. Run the following command:

```
pip install -r requirements.txt
```

You will be modifying specific files in the given folder. These files are clearly marked, and changes should be made only in designated sections.

The NotImplementedError should also be removed from specific sections. The placeholder should look like this:

```
# ============= Insert Code Here =============
raise NotImplementedError
# ============================================
```

You must replace the `raise NotImplementedError` line with your own implementation.

3 Implementation Details

You need to modify and implement the following 3 files Only:

- **activation.py** - Implements activation functions.

- **layers.py** - Implements different layers of the CNN.
- **loss.py** - Implements the cross-entropy loss function.

Each of these files contains predefined function placeholders that must be completed.

4 Functions to Implement in Each File

4.1 Fully Connected Layer

Class Definition:

- `LinearLayer(in_features, out_features)`
  - **Arguments:**
    - `in_features`: Number of input neurons
    - `out_features`: Number of output neurons
- Implement forward and backward propagation methods.

4.2 Convolutional Layer

Class Definition:

- `ConvolutionLayer(in_channels, out_channels, kernel_size, stride)`
  - **Arguments:**
    - `in_channels`: Number of input channels
    - `out_channels`: Number of output channels
    - `kernel_size`: Size of the kernel (always square)
    - `stride`: Step size for convolution
- Implement forward and backward propagation methods.

4.3 Loss Function

Class Definition:

- `CrossEntropy(eps=1e-8)`
  - No Other required arguments.
  - Implement forward and backward computations.

4.4 Activation Functions

Class Definition:

- `ReLU, Sigmoid, Softmax`
  - No Other required arguments.
  - Implement forward and backward computations.

Provided everything is implemented correctly, you should be able to obtain $\geq 90\%$ accuracy on the test set, and the confusion matrix should resemble the expected structure.

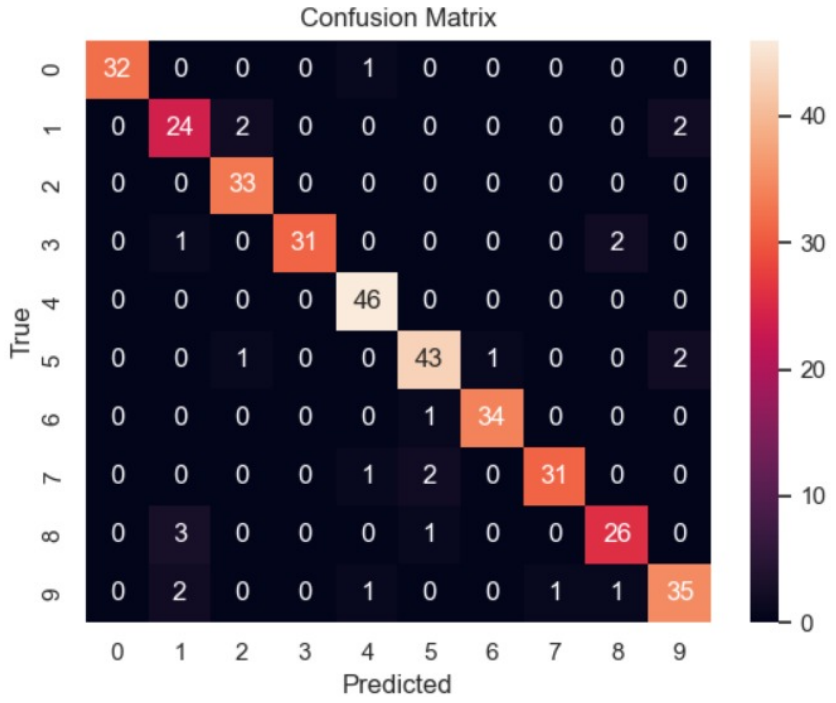

Figure 1: Expected Confusion Matrix

5 Training Pipeline

Once the implementation is complete, running the main script should:

1. Load the dataset and create the train-test split.
2. Train the model.

3. Compute accuracy on the test set.
4. Save training loss, confusion matrix, and prediction plots.

The expected test accuracy should be at least 90%.

6 Testing and Debugging

A total of 18 test cases are provided in the `tests` folder to verify each component separately. Run the following command to execute the tests:

```
python -m pytest --disable-warnings
```

The final test case will pass when all the modules have been implemented and you obtain accuracy greater than 90%.

7 Submission Guidelines

- Ensure all required functions are implemented.
- Do not modify any test files.
- Make sure to submit all your modified files in a single zipped folder for this task.

Important: Plagiarism is strictly prohibited and will be penalized.