Final Project

Building a Handwritten Digit Classifier

Course: Handwritten Digit Classification using CNN

1. Project Objective

The goal of this project is to synthesize the knowledge and skills you have acquired throughout this course. You will design, build, train, and evaluate a **Convolutional Neural Network (CNN)** capable of accurately classifying handwritten digits from the famous MNIST dataset. This project will test your understanding of data preparation, CNN architecture, and the practical application of deep learning frameworks like PyTorch.

2. Background

The MNIST dataset is a cornerstone of machine learning and computer vision. It contains **70,000 grayscale images** of handwritten digits (0 through 9), split into a training set of 60,000 images and a testing set of 10,000 images. Each image is a small, 28×28 pixel square. Your challenge is to build a model that can look at one of these images and correctly predict the digit it represents. This project mirrors real-world tasks like optical character recognition (OCR) in postal services and banking.

3. Project Tasks & Requirements

You are to complete the following tasks. Your final submission will be a detailed report that walks through each of these stages.

Task 1: Data Preparation and Augmentation

Before training a model, you must understand and prepare your data.

Load and Explore:

Load the MNIST training and testing datasets.

Visualize:

Display a small batch of images from the dataset along with their corresponding labels. This is a crucial step to ensure the data has loaded correctly.

Preprocessing:

- **Transformation:** Convert the images to PyTorch tensors.
- **Normalization:** Normalize the tensor values. In your report, you must explain why normalization is a critical step for training neural networks and describe the mean and standard deviation values you used for normalization.

Data Augmentation:

- Apply at least two data augmentation techniques (e.g., random rotations, horizontal/vertical shifts, scaling).
- In your report, explain the purpose of data augmentation and how it can help prevent overfitting and improve model generalization.

Task 2: CNN Architecture Design

This is the core design phase of the project. You will define the architecture of your CNN.

Architectural Blueprint:

Design a CNN model. You have creative freedom here, but a typical architecture includes a sequence of convolutional and pooling layers followed by one or more fully connected layers.

Detailed Description:

In your report, you must provide a layer-by-layer description of your model, including:

- **Convolutional Layers:** Specify the number of input/output channels, kernel size, stride, and padding for each.
- **Activation Functions:** State which activation function (e.g., ReLU) you used after each convolutional layer and justify your choice.
- **Pooling Layers:** Describe the type of pooling (e.g., MaxPooling), kernel size, and stride. Explain the role of pooling in a CNN.
- **Fully Connected Layers:** Detail the input and output features for each linear layer.
- **Output Layer:** Specify the final output layer, including its activation function (e.g., LogSoftmax or Softmax), and explain why it is appropriate for this multi-class classification task.

Task 3: Model Training and Evaluation (Week 4 Recap)

With a prepared dataset and a designed model, you will now train and test your network.

Training Configuration:

- **Loss Function:** Select an appropriate loss function (e.g., Cross-Entropy Loss, NLLLoss). Justify your choice in the context of your output layer's activation function.
- **Optimizer:** Choose an optimizer (e.g., Adam, SGD). Explain its role and specify the learning rate you used.
- **Training Loop:** Describe the process of a single training epoch. This description should include how you pass data to the model, calculate the loss, perform backpropagation, and update the model's weights.

Performance Evaluation:

- Train the model on the training dataset for a chosen number of epochs.
- After training, evaluate your model's performance on the unseen test dataset.
- Report the final test accuracy and the final training/validation loss.

Task 4: Analysis and Conclusion

The final step is to analyze your results and reflect on the project.

Results Summary:

Present your final test accuracy. You may also include a confusion matrix to see which digits your model struggled with the most.

Performance Analysis:

Discuss your model's performance. Is the accuracy what you expected? Analyze any interesting findings (e.g., "The model frequently confused 4s and 9s").

Challenges and Improvements:

Describe any challenges you encountered during the project. Propose and explain at least two specific ways you could potentially improve your model's accuracy in future work.

Deliverables

You are required to submit **two items**:

1. **A Written Project Report (PDF Format):** A formal report detailing your work in each of the four tasks outlined above. The report should be well-structured, clearly written, and include all requested explanations and justifications.
2. **Your Complete Source Code:** A fully commented Jupyter Notebook (.ipynb) or Python script (.py) that implements your solution. The code should be clean, runnable, and easy to follow.

Evaluation Criteria

Your project will be evaluated based on the following criteria:

- **Report Clarity and Completeness (40%):** The quality, clarity, and thoroughness of your written report and explanations.
- **CNN Design and Justification (25%):** The thoughtfulness of your model's architecture and the quality of your justification for design choices.
- **Code Implementation (20%):** The correctness, clarity, and organization of your PyTorch code.
- **Final Model Performance (15%):** The final accuracy achieved by your model on the MNIST test set. (Note: Full marks in this category do not require 100% accuracy, but a high-performing, well-trained model.)

Important Note

This project is designed to demonstrate your understanding of the fundamental concepts covered in this course. Focus on clear explanations, thoughtful design choices, and well-documented code. The learning process and your ability to articulate your approach are just as important as the final accuracy numbers.