Handwritten Digit Classification Using CNN and MNIST Dataset

Sora Owada

Mar 20, 2025

1. Describe

I am building a Convolutional Neural Network (CNN) that classifies handwritten digits (0–9) using the MNIST dataset. The project uses Python and TensorFlow to preprocess data, train a CNN model, and evaluate its performance. Currently, the model architecture and training process are implemented and functioning correctly in a Python script.

2. Examples

• **Input:** A grayscale image of a digit from the MNIST dataset (28x28 pixels).
Start Condition: The image is loaded, normalized, and reshaped into the input format of the CNN.
Output: The trained CNN model predicts a digit label (e.g., 3, 7, 0).

3. Meet the Objectives

• **Apply neural network architecture:**
I use TensorFlow's Keras API to define and train a CNN with multiple layers such as Conv2D, MaxPooling2D, and Dense layers.
• **Preprocess data:**
The program normalizes pixel values and reshapes the dataset into formats suitable for the CNN model.
• **Train and evaluate:**
I train the model using the training set and evaluate accuracy using the test set, tracking loss and performance.
• **Visualize performance:**
The code includes functions to display accuracy/loss curves and evaluate classification metrics.

4. Resources, Skills, Models, Tools, etc.

• **TensorFlow/Keras:**
Used to construct and train the CNN model. Offers high-level abstractions for layers and training.
• **MNIST dataset (via tensorflow.keras.datasets):**
Standard benchmark for digit recognition. Easily loadable for training/testing.
• **Matplotlib:** Used to visualize training/validation accuracy and loss over epochs.
• **Jupyter Notebook/Python Scripts:**
Tools for rapid development and debugging.
• **Online Tutorials (e.g., TensorFlow CNN tutorial):**
Helped in setting up and debugging model architecture.

5. Your Approach

Workflow Design:
1. **Input:** Load MNIST dataset using tf.keras.datasets.mnist.load_data().
2. **Preprocessing:**
• Normalize pixel values to range [0, 1].
• Reshape input images to (28, 28, 1).
• One-hot encode the labels for training.
3. **Model Architecture:**
• Conv2D -> MaxPooling2D -> Conv2D -> MaxPooling2D -> Flatten -> Dense -> Dense(10).
4. **Training:**
• Compile the model using Adam optimizer and sparse categorical cross-entropy.
• Train using .fit() for a set number of epochs with validation split.
5. **Evaluation:**
• Evaluate performance using .evaluate() on the test set.
• Plot training history (loss/accuracy).
6. **Output:** Predicted digit label with accuracy and loss reported.

6. Feedback

It would be helpful to receive feedback on:
• Whether the model complexity is appropriate for this task.
• Any best practices for structuring the CNN to improve accuracy.
• Suggestions on hyperparameter tuning or training visualization enhancements.
• Ideas for expanding the project (e.g., real-time digit prediction from user-drawn input).

7. References

MNIST Dataset: https://www.kaggle.com/datasets/hojjatk/mnist-dataset
GitHub: https://github.com/gursky1/MNIST-Tensorflow-2