Final Project Report: License Plate Character Recognition with Hardware Acceleration

1. Project Overview

This project focuses on recognizing characters from vehicle license plates using a convolutional neural network (CNN) combined with hardware acceleration for performance optimization. The project integrates software-based character detection and classification with Verilog-based hardware simulation of 2D convolution for efficient image processing.

2. Objectives

- Develop a character recognition system using deep learning (CNN).
- Build and train a dataset from real-world license plate characters.
- Implement 2D convolution in Verilog to accelerate CNN operations.
- Validate results with test images and assess speed/accuracy improvements.
- Combine theory and hardware-software co-design to demonstrate embedded AI applications.

3. Development Workflow

1. **Dataset Creation**: Custom character images were generated using Python and OpenCV and saved in organized folders per class (0–9, A–Z).
2. **CNN Model Training**: train_char_cnn.py trains a simple CNN model using the synthetic dataset.
3. **Character Prediction**: plate_reader.py uses OpenCV to extract characters from a real license plate image and predicts characters using the trained CNN model.
4. **Profiling**: profile_cnn.py and associated .prof and .txt files log performance.
5. **Hardware Integration**: The 2D convolution layer is implemented in Verilog using conv2d.sv, tested with conv2d_tb.sv and conv2d_sliding_tb.sv.

4. Software Components and File Descriptions

- generate_char_dataset.py: Generates character dataset by rendering characters using OpenCV. Each character is stored in char_dataset/<char>/.
- train_char_cnn.py: Defines and trains a CNN using PyTorch to classify the character images.
- model.pth: The saved PyTorch model weights after training.
- plate_reader.py: Takes an image of a license plate, segments it into characters, and classifies them using the trained CNN model.
- profile_cnn.py: Measures the execution time of CNN inference and logs profiling data.

- profile_results.prof and profile_summary.txt: Output files containing the performance profile of the model.
- smoke_check.py and sanity_check.py: Basic test scripts to verify model loading and prediction accuracy.
- Detected Characters.jpg: Sample output showing recognized characters.

5. Hardware Design (Verilog)

- conv2d.sv: Implements 2D convolution for image feature extraction.
- conv2d_tb.sv: Standard testbench to validate the correctness of conv2d.sv with static input.
- conv2d_sliding_tb.sv: Testbench that simulates sliding-window operation across a feature map.
- run_hw.bat: Batch script to compile and simulate Verilog code using Icarus Verilog.

6. Dataset Creation and Training

- Each character (A-Z, 0-9) is rendered as a grayscale image using OpenCV.
- Images are saved in subfolders under char_dataset/.
- Training is performed using a CNN defined in train_char_cnn.py, which outputs model.pth.

CNN Architecture (Simplified):

- Conv2D → ReLU → MaxPool → Flatten → Dense → Softmax
- Trained using cross-entropy loss and Adam optimizer.

7. Profiling and Results

- Profiling done with cProfile on CNN inference (profile_cnn.py).
- profile_results.prof and profile_summary.txt give insights into function-wise execution time.
- Final test image (Detected Characters.jpg) confirms model accuracy and correct segmentation.

8. Reproducibility Instructions

Prerequisites:

- **Software**: Python 3.8+, PyTorch, OpenCV, Matplotlib, NumPy
- **Hardware Tools**: Icarus Verilog, GTKWave (optional for waveform viewing)

Setup and Execution:

```
# Step 1: Install dependencies
pip install torch torchvision opencv-python matplotlib numpy

# Step 2: Generate training dataset
python generate_char_dataset.py
```

```
# Step 3: Train the CNN model
python train_char_cnn.py
# This creates 'model.pth'

# Step 4: Run plate reader script
python plate_reader.py
# This reads the plate image, segments characters, and prints predictions.

# Step 5: Profile the CNN inference
python -m cProfile -o profile_results.prof profile_cnn.py
python -m pstats profile_results.prof
# This outputs function call time summary

# Step 6: Simulate Verilog hardware
# (Make sure Icarus Verilog is installed)
run_hw.bat
# This compiles conv2d and runs testbenches
```

9. Conclusion and Future Work

This project successfully demonstrates a complete character recognition pipeline with real-world license plate images, synthetic dataset training, and hardware acceleration using Verilog.

Future Improvements:

- Improve CNN architecture for higher accuracy.
- Extend Verilog modules to support full CNN pipeline.
- Use real-time video input for continuous license plate recognition.
- Port Verilog modules to FPGA for on-board acceleration.