Efficient Vehicle Plate Recognition - LPRNet

Course Project Report

Name : **Yogesh Hasabe** *UnityID* : **yhasabe**

Introduction

Efficient license plate recognition is essential for real-time applications such as traffic monitoring and smart parking systems. In this project, we optimize the LPRNet model to improve its accuracy, speed, and space efficiency. This report documents the original model, applied optimizations, performance outcomes, lessons learned, and repository details.

Original DNN Model

The project employs **LPRNet**, a lightweight and efficient model for license plate recognition.

- **Baseline Metrics**:
    - Accuracy: **90.20%**
    - Inference Speed: **0.21 seconds per 1/1000 images**
    - Model Size: **1816.74 KB**

Model Optimizations

The following optimizations were applied to enhance the model's performance:

1. Quantization

Quantization reduced the precision of model weights and activations to 8-bit integers, thereby decreasing model size and maintaining near-baseline accuracy.

- **Results**:
    - Accuracy: **89.80% (-0.30%)**
    - Speed: **0.33 seconds per 1/1000 (slower inference)**
    - Model Size: **533.57 KB (70.6% reduction)**

2. Pruning

Pruning removed redundant connections to reduce model complexity.

- **Results**:
    - Accuracy: **90.10% (-0.10%)**
    - Speed: **0.32 seconds per 1/1000**
    - Model Size: **1816.74 KB (no change)**

MLC Optimizations

1. Manual Optimization

Manual optimizations focused on custom layer fusion and parallelization techniques using TensorRT.

- **Results**:
    - Accuracy: **89.90% (-0.30%)**
    - Speed: **0.036 seconds per 1/1000 (80.8% improvement)**
    - Model Size: **840.7 KB (53.7% reduction)**

2. AutoTuning Optimization

Automated optimization using tools like NVIDIA NNI explored optimal configurations, further enhancing speed and size.

- **Results**:
    - Accuracy: **89.80% (-0.40%)**
    - Speed: **0.031 seconds per 1/1000 (83.7% improvement)**
    - Model Size: **840.7 KB (53.7% reduction)**

Combined Optimizations

The best results were achieved by combining **pruning** with **AutoTuning**.

- **Results**:
    - Accuracy: **90.00% (+0.30%)**
    - Speed: **0.032 seconds per 1/1000 (84.7% improvement)**
    - Model Size: **850.87 KB (53.2% reduction)**

Performance Metrics Comparison

Optimization	Accuracy (%)	Speed (s/1k imgs)	Model Size (KB)
Baseline Model	90.20	0.21	1816.74
Quantization	89.80	0.33	533.57
Pruning	90.10	0.32	1816.74
Manual Optimization	89.90	0.036	840.7
AutoTuning	89.80	0.031	840.7
Combined (Pruning + AutoTuning)	90.00	0.032	850.87

Lessons Learned

1. **Quantization Trade-offs**: While quantization significantly reduces model size, it can marginally degrade accuracy and sometimes increase inference time.
2. **Importance of Pruning**: Pruning is effective for simplifying models without severely affecting performance.
3. **MLC Optimization Benefits**: Tools like TensorRT and NNI are invaluable for achieving high-speed performance but require thorough experimentation for optimal results.
4. **Combining Methods**: A combination of optimizations often provides the best balance of size, speed, and accuracy.
5. **Onnx file** format is **NOT** compatible with Quantization techniques hence not used.

Repository Details

- **GitHub Repository**: Yogesh31Hasabe/NCSU-CSC_591-RealTime_AI_and_Machine_Learning_Systems-CourseProject-LPRNet
- **Repository Structure**:
    - `./`: Contains all model training and optimization scripts in **.py** & **.ipynb** format.
    - `/data`: **Test** dataset for validation.
    - `/weights`: Saved **models** & **weights** (**baseline** and **optimized**).
    - `README.md`: Detailed instructions for testing and reproducing results.

Instructions for Running the Project

- Clone the repository if running locally & execute the .ipynb files accordingly:

```
Unset
git clone
https://github.com/Yogesh31Hasabe/NCSU-CSC_591-RealTime_AI_and_Ma
chine_Learning_Systems-CourseProject-LPRNet
```

Or Run the following .ipynb files in **Google Colab**

- For Model Optimizations:
    - Quantization: **yhasabe_Course_Project_Model_Optimization_Quantization.ipynb**
    - Pruning: **yhasabe_Course_Project_Model_Optimization_Pruning.ipynb**
- For Model Optimizations:
    - Manual & AutoTuning: **yhasabe_Course_Project_MLC_Optimization_Manual_and_Auto.ipynb**
- For Combined Optimizations: **yhasabe_Course_Project_Combined_Optimizations_Pruning_and_AutoTuning.ipynb**

Conclusion

The project successfully optimized the LPRNet model, achieving a **53.2% reduction in size**, **84.7% improvement in speed**, and maintained **90% accuracy**. These results demonstrate the effectiveness of combining **pruning** and **automated MLC** techniques for **real-world deployment** of lightweight DNNs.