CSC591
Real-time AI and Machine Learning Systems Final Project Report

Shyamal Gandhi (sgandhi6)

1. What is the original DNN model you chose to use?

The original model used in this project is **LPRNet**. It is a lightweight and high-performance DNN designed specifically for license plate recognition tasks. This model was selected due to its efficiency and performance in real-world applications. The trained model weights were stored in `Final_LPRNet_model.pth` and this served as the baseline model throughout the project to evaluate the impact of different optimizations.

---

2. What optimizations did you apply to the model to improve its accuracy, speed, and space efficiency?

The following optimizations are applied in order to improve the performance of the baseline LPRNet model across key metrics such as accuracy, inference time.

a) Quantization

I applied **PyTorch's 8-bit static quantization** to reduce the model's size and inference time. Static quantization was used because LPRNet does not include linear layers which makes dynamic quantization inapplicable. Static quantization converted model weights and activations from 32-bit floating point to 8-bit integer representations.

- **Advantages**: Reduced model size and inference time.
- **Trade-off**: Accuracy drop due to quantization effects.

b) Pruning

Unstructured pruning was applied with a pruning ratio of 0.5 which targets non-essential weights to reduce computational complexity. This pruning was conducted without retraining or fine-tuning the pruned model.

- **Advantages**: Reduced inference time due to sparsity in the weights.
- **Trade-off**: Moderate accuracy degradation because the pruned model was directly evaluated on the test dataset without fine-tuning.

c) MLC (Machine Learning Compilation)

I utilized **Apache TVM's autoscheduler** to perform compiler-level optimizations. The autoscheduler generated 23 tasks aimed at improving computation and memory efficiency. Key optimizations included:

1. **Operator Fusion**: Combined multiple operations (e.g., convolution + bias add + relu) into a single kernel which helped reduce memory overhead.
2. **Data Layout Transformation (NCHWc)**: Changed the tensor layout to optimize memory locality and improve hardware utilization.
3. **Loop-Level Optimizations**:
    - **Tiling**: Divided loops into smaller chunks for better cache usage.
    - **Vectorization**: Utilized SIMD instructions for parallel processing.
    - **Parallelization**: Distributed computations across multiple CPU cores or GPU threads.
4. **Reduction Optimization**: Accelerated operations like pooling and mean computations by fusing and tiling reductions.

- **Advantages**: Significant speedup with minimal accuracy loss.
- **Trade-off**: Slightly increased compilation complexity.

d) Combined Optimizations

1. **Pruned + Quantized**: Combined pruning and quantization for maximum size reduction and inference speed improvement.
    - **Trade-off**: This resulted in the largest accuracy drop.
2. **Pruned + MLC**: Combined pruning with compiler optimizations and tries to strike a balance between speed and accuracy.
    - **Trade-off**: Moderate accuracy loss due to the pruning component.

---

3. What is the performance (the three metrics) of the original model and the optimized model?

Model Variant	Size (MB)	Test Accuracy (%)	Inference Time (ms)
Original (baseline)	1.91	90.3	212
Quantized	0.5	78.0	27
Pruned	1.91	82.5	152
MLC	1.91	89.0	82
Pruned + Quantized	0.5	62.5	23
Pruned + MLC	1.91	80.0	68

Original Model:

- Achieved the highest accuracy (90.3%) but had the largest size (1.91 MB) and slowest inference time (212 ms).

Quantized Model

- **Size**: Reduced significantly from 1.91 MB to 0.5 MB due to 8-bit integer representation.
- **Accuracy**: Dropped to 78%, as quantization impacts model precision, especially for complex patterns.
- **Inference Time**: Improved drastically to 27 ms, as computations in lower precision (INT8) are faster.

Pruned Model

- **Size**: Remained at 1.91 MB since unstructured pruning doesn't physically reduce the size without model sparsity support.
- **Accuracy**: Moderately reduced to 82.5% due to the removal of weights without fine-tuning.
- **Inference Time**: Improved to 152 ms because fewer computations are required due to pruned weights.

MLC Model

- **Size**: Stayed at 1.91 MB as compiler-level optimizations don't change model weights or structure.
- **Accuracy**: Minimal drop to 89%, as TVM's optimizations focus on computation efficiency, not altering the model's functionality.
- **Inference Time**: Reduced significantly to 82 ms due to advanced loop and operator optimizations.

Pruned + Quantized Model

- **Size**: Reduced to 0.5 MB, combining the size benefits of quantization and pruning.
- **Accuracy**: Dropped significantly to 62.5% due to compounded effects of weight removal and precision loss.
- **Inference Time**: Fastest at 23 ms, leveraging both sparsity (pruning) and low-precision computation (quantization).

Pruned + MLC Model

- **Size**: Stayed at 1.91 MB as MLC does not reduce size and pruning alone doesn't compress the model size.
- **Accuracy**: Dropped to 80%, with pruning being the primary cause of degradation.

- **Inference Time**: Reduced to 68 ms, benefiting from both pruning-induced sparsity and TVM's efficient computation optimizations.

---

4. What lessons have you learned through the project?

1. **Optimization Trade-offs Are Inevitable**:
    - Each optimization technique brings benefits in speed and size but often at the cost of accuracy. Balancing these trade-offs requires aligning optimizations with the intended application and performance requirements.
2. **Static Quantization Works Best for Models Without Linear Layers**:
    - Quantization significantly reduced the model size but highlighted its limitation in accuracy retention, especially for complex tasks.
3. **Pruning Requires Fine-Tuning to Be Effective**:
    - Unstructured pruning alone without retraining results in moderate accuracy degradation. Incorporating retraining or fine-tuning can mitigate this issue.
4. **Compiler-Level Optimizations Offer Significant Gains Without Model Modifications**:
    - Apache TVM's auto-scheduler optimizations demonstrated that compiler-side enhancements could achieve substantial speedup without compromising much on accuracy. This is a promising direction for deploying models in resource-constrained environments.
5. **Combining Techniques Requires Careful Evaluation**:
    - While combining optimizations like pruning and quantization provided extreme benefits in size and speed it led to unacceptable accuracy degradation. It is essential to evaluate whether the trade-offs align with the use case.
6. **Domain Knowledge and Tool Selection Are Key**:
    - Understanding the model architecture and its constraints (e.g., no linear layers in LPRNet) guided the selection of appropriate optimizations, such as static quantization and pruning.

---

5. Github Repository

https://github.com/sgandhi31/LPRNet_Optimization