CNN Classification Model Design Report

CNN Optimization, Quantization, Analysis for CIFAR-10

MINWOO CHOI

Quantization Strategy

I employed **Post-Training Static Quantization (PTQ)** to optimize the model. There are three quantization configuration options.

Configuration strategy	Activations Observer	Weights Observer	Backend engine
MinMax	MinMax Observer	Per-channel Minmax	CPU x86
Histogram	Histogram Observer	Per-channel Minmax	CPU x86
fp16 (GPU only)	x	x	GPU only
fp32 (baseline)	x	x	CPU / GPU

The int8 quantization configurations were categorized into two strategies according to type of Activaiton Observer. I fixed weight Observer and differentiated activation observer part with minmax observer and histogram observer. I also investigated FP16 (Half-Precision) as an alternative approach to reduce model size.

Performance Table

The table below summarizes the performance of the FP32 baseline versus various quantized configurations. (*means running time on GPU)

Model	Test Accuracy	Inference Time	Size
FP32 (Baseline)	82.8%	27.037s, (3.267s)*	5.9 MB
FP32 (Overfit Candidate)	83.1%	27.075s	5.9 MB
Int8 (MinMax Quantized)	82.5%	17.768s (66%)	1.6 MB (27%)
Int8 (Histogram Quantized)	82.5%	18.059s (67%)	1.6 MB (27%)
FP16 (only GPU)	82.8%	None, (2.560s)*	3 MB (50.8%)

Comprehensive Evaluation of Quantization

1. Methodology: Quantizer Selection (for int8 quantization)

- **Weights:** Adopted **channel-wise minmax observers**. This preserves unique channel characteristics, whereas layer-wise quantization would significantly degrade smaller weights.
- **Activations:** Adopted **layer-wise minmax observers** VS **Histogram observers** to account for Batch Normalization distributions.

2. Performance Analysis

- **Robustness:** The model shows strong robustness to quantization noise. (82.8% vs 82.5).
- **Accuracy:** Tested on 10,000 samples, accuracy dropped by only **0.3%**.
- **Model Size:** Reduced to **27%** of the original size. (Note: It does not reach a perfect 25% reduction due to the overhead of additional quantization parameters required to record distributions).
- **Inference Speed:** Increased by **1.5x** (Inference time reduced to approx. 66–67%).
    - *Reason:* INT8 operations consume less memory bandwidth and are computationally faster.
    - The reduction in inference time was not more dramatic because, while the model's computation speed increased, the latency for **loading image data from memory remains constant (I/O bottleneck)**.

3. GPU & FP16 Considerations

- **FP16 Feasibility:** For GPU-based inference/training, FP16 is a viable alternative. Converting the FP32 model using .half() resulted in negligible quantization error.
- **Current Code Limitation:** If we use float16 based train/inference, the ToTensor() function requires modification to transfer INT8 data directly to the GPU. (Note: CPUs usually do not offer native FP16 operations).
- **Bottleneck Analysis:** FP16 and FP32 inference time shows the **memory transfer time** (CPU to GPU) occupies the largest part of the inference time. Sometimes FP32 inference time is shorter than FP16. Even parameter computation is faster on FP16, there exist current code limitation(upper) and memory loading problem.