Adaptive Layer-wise Quantization through System Profiling

Team Members
Venkata Bharath Reddy Reddem (vreddem@uci.edu)
Savyasach Reddy Enukonda (senukond@uci.edu)
Anisha Priyadarshini Mohanty (apmohant@uci.edu)

Introduction

- Deep Neural Networks (DNNs) power modern AI applications in vision, language, and speech.
- Growing model sizes cause **high computation and memory costs** during inference.
- **Deploying large models** on edge or latency-sensitive systems remains difficult.
- **Quantization** reduces precision (e.g FP32 → INT8) to speed up inference and lower memory use.
- **Uniform quantization** across all layers leads to **inefficient hardware utilization** and **accuracy degradation**.

Previous Works

- Early quantization such as DoReFa-Net, XNOR-Net reduced model precision to binary or ternary weights, which resulted in large accuracy loss.
- **PTQ** made deployment easier by **quantizing pre-trained models without retraining** (**TensorFlow Lite** and **PyTorch Quantization Toolkit**).
- Most PTQ frameworks still **apply uniform bit-widths**, ignoring layer sensitivity.
- **HAQ** and **AutoQ** introduced hardware-aware quantization using RL to choose per-layer bit-widths, but relied on **offline profiling** and **costly search**.
- **QNAS** extended this idea via quantization-aware architecture search, yet remained **computationally expensive** and lacked runtime flexibility.
- **SmoothQuant** and **ZeroQuant** improved stability for large models but used **fixed 8-bit configurations** with no adaptive feedback.

Problem with Uniform Quantization

- Traditional quantization uses the **same bit-width for all layers**, but different layers have **different sensitivities** which causes **accuracy degradation**.
- Current widely used PTQ methods fail to utilize hardware efficiently, especially in **heterogeneous systems** (CPU, GPU, NPU, edge).
- **Early convolution layers** are highly **sensitive** to precision changes, while **deeper layers** tend to be more **robust**.
- **Downsampling blocks** are small but **critical** and can degrade quickly when quantized too aggressively.
- **Large, memory-heavy layers** are often **resilient** to lower precision and can be quantized more **aggressively** without major accuracy loss.

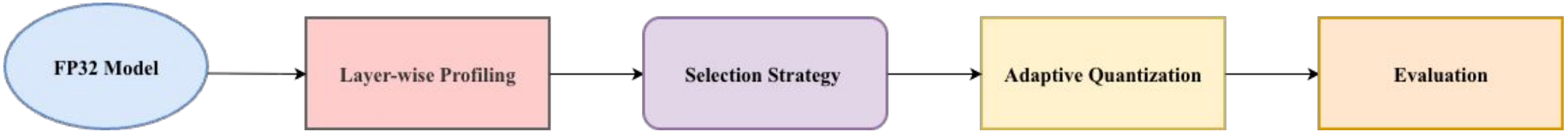
Why Adaptive Quantization?

- Modern DNNs (ResNet, Transformers) are **large and compute-heavy**, creating latency and memory bottlenecks.
- Deploying models on **edge or heterogeneous hardware** (CPU, GPU, NPU) demands **efficient inference** without losing accuracy.
- **Uniform quantization** ignores layer diversity, either wastes resources or reduces accuracy.
- **Hardware characteristics differ** as a one-size-fits-all bit-width is inefficient.
- Need a **profiling-guided, system-aware strategy** to adapt precision per layer for optimal efficiency.

Our Proposed Approach

- Perform **layer-wise profiling** to measure:
  - Latency
  - Memory footprint
  - Sensitivity to quantization
- Use profiling feedback to **selectively quantize layers**.
- **Dynamically adjust bit-widths** for each layer based on profiling results and hardware characteristics.
- Achieve a **balanced latency–accuracy trade off** and improved **system-level efficiency** (throughput, memory usage, and runtime).
- Enables **system-aware precision control**, adapting quantization to both the model and the underlying hardware.

Design



- Run systematic **layer-wise profiling** and compute three core per-layer metrics:
  - Output sensitivity
  - Execution latency
  - Parameter memory cost
- Calculate a **multi-objective score** combining the three metrics.
- Assign the amount of quantization (INT8/FP16) per layer.
- Evaluate **accuracy** and **latency** on CPU and GPU.
- Entire process is **training-free**, **hardware-adaptive**, and **architecture-agnostic**.

Baseline Quantization

Full INT8 PTQ (Per-Tensor)

- INT8 for all weights + activations
- Single symmetric scale per tensor
- Standard PTQ formula:

$$\hat{W} = \text{round}\left(\frac{W}{s}\right)s, \qquad s = \frac{\max(|W|)}{2^{b-1}-1}$$

Limitations

- May degrade accuracy in high-variance layers
- Cannot adapt to hardware-specific behavior

Sensitivity Analysis

Layer-wise Sensitivity Computation

$$S_i = \frac{\|Y_i^{FP32} - Y_i^{quant}\|_2}{N_i + \varepsilon}$$

Interpretation

- **High Si** → fragile layer → keep higher bit-width
- **Low Si** → safe to quantize

Observed patterns

- Early convolutional layers are the most **sensitive**.
- Downsampling layers show **unexpected fragility**.
- Later blocks are comparatively **more robust**.

Hardware Latency Profiling and Memory Profiling

Per-Layer Latency Measurement

$$T_i = \frac{1}{K} \sum_{k=1}^{K} \text{latency}(L_i), \quad K = 100$$

Per-Layer Memory Cost

$$M_i = \frac{\text{params}(L_i) \cdot b_i}{8}$$

Multi-Objective Scoring

Unified Layer Score

$$\text{score}_\ell = \frac{\text{latency}_\ell + \text{memory}_\ell}{\text{sensitivity}_\ell + \varepsilon}$$

- Rank layers using memory cost, latency benefit, and sensitivity to assign mixed precision.
- Assign the amount of quantization according to their relative positions in this ranking, where top 50% of the layers with INT8 and next 35% of the layers with FP16 quantization.

Experimental Setup

Evaluation Protocol

- 1000 ImageNet validation samples
- ONNX Runtime
- Batch size = 1
- Metrics
  - Accuracy
  - Latency (ms/img)
  - Model Size (MB)

Results (1)

Table 1: ResNet-18 – CPU Performance Comparison

Method	Acc (%)	Lat (ms)	Size (MB)
FP32 Baseline	69.6	82	45.7
INT8 Full PTQ	66.4	41	11.2
Latency-aware	69.2	43	17.8
Latency+Memory	68.7	43	17.1
Latency+Memory+Sensitivity	69.4	43	17.2
Mixed-Precision	69.4	41	14.8

Results (2)

Table 2: MobileNetV2 – CPU Performance Comparison

Method	Acc (%)	Lat (ms)	Size (MB)
FP32 Baseline	71.8	38.2	13.6
INT8 Full PTQ	70.9	14.7	3.4
Latency-aware	71.2	18.4	4.9
Latency+Memory	70.7	18.7	4.3
Latency+Memory+Sensitivity	71.5	18.7	4.5
Mixed-Precision	71.6	15.8	3.9

Conclusion and Next Steps

- Full INT8 provides **strong compression** but often **reduces accuracy**.
- Selective quantization strategies **recover accuracy.**
- Mixed-precision gives best trade-off
  - Comparable latency and size
  - Near-FP32 accuracy

- Extend experiments beyond CPU and measure real INT8 + FP16 acceleration.
- Benchmark using ONNX Runtime (CUDA EP) and TensorRT engines.
- Validate our selective quantization gains under GPU kernel scheduling.

Future Work

- Add **power** and **energy profiling** for more comprehensive optimization.
- Extend the method to **transformer models** and **LLMs**.
- Explore **reinforcement learning** for **automated bit-width selection**.
- Ablation on multi-objective score calculation.
- Adapt this technique to explore per-channel quantization.
- Explore other techniques to assign the type of quantization for each layer.