Adaptive Layer-wise Quantization through System Profiling

Venkata Bharath Reddy Reddem *Savyasach Reddy Enukonda* *Anisha Priyadarshini Mohanty*

Abstract

Modern deep learning systems face significant performance bottlenecks due to the computational and memory demands of large neural networks. Uniform quantization across all layers often leads to inefficient hardware utilization or unacceptable accuracy loss, particularly in heterogeneous or resource constrained environments. This project aims to address this challenge through system aware, adaptive quantization, which dynamically selects which layers to quantize based on runtime profiling.

Our proposed framework conducts layer wise profiling to capture latency, memory footprint, and sensitivity to quantization error, and then performs selective quantization on layers that offer high efficiency gains with minimal impact on accuracy. Unlike traditional post-training quantization (PTQ) approaches, this method introduces a profiling guided quantization policy that adapts to hardware characteristics and workload conditions. The project evaluates system level metrics such as end-to-end inference latency, throughput, memory utilization, and accuracy retention, thus providing insights into optimal quantization strategies for efficient DNN deployment on modern ML systems.

1 Introduction

Deep neural networks (DNNs) have achieved remarkable success in a wide range of domains, including computer vision, natural language processing, and speech recognition. However, these models are increasingly large and computationally expensive, posing significant challenges for deployment on resource constrained and latency sensitive systems. Model quantization represents the parameters and activations with lower bit-widths, and has emerged as an effective technique to reduce computation and memory costs. Despite its promise, uniform quantization across all layers often leads to substantial accuracy degradation, as different layers exhibit varying sensitivities to reduced precision.

Recent advances in post training quantization (PTQ) have made it possible to compress models without retraining, but these methods typically apply a one-size-fits-all quantization strategy, failing to exploit the heterogeneity in layer level computational cost and error sensitivity. This results in suboptimal efficiency-accuracy tradeoffs, particularly in real world deployments where hardware characteristics and model architectures vary widely.

This project addresses this limitation by introducing a profiling guided, layer wise quantization framework. The proposed approach first profiles each layer to measure latency, memory usage, and quantization sensitivity, and then selectively quantize the layers that offer the most computational savings with minimal accuracy loss. By adapting quantization policies based on empirical profiling data, we aim to achieve improved runtime efficiency while maintaining near-baseline accuracy.

2 Motivation

The motivation for this project stems from the observation that DNN layers contribute unequally to overall inference cost and accuracy sensitivity. For example, early convolutional layers are often computationally heavy but robust to quantization noise, whereas deeper layers tend to be more accuracy-critical. Uniform quantization does not capture this variation, leading to inefficient use of hardware resources.

By integrating runtime profiling into the quantization process, we can dynamically identify and quantize layers that yield the most significant performance gains with minimal accuracy impact. This profiling guided approach makes the quantization process adaptive to both model architecture and underlying hardware characteristics, making it more practical for deployment in real-world ML systems. Ultimately, the goal is to design a lightweight, generalizable framework that demonstrates how simple profiling can enable smarter, system aware quantization strategies for modern deep learning workloads.

3 Related Work

Quantization has become a widely adopted method for reducing the computational and memory costs of deep neural networks (DNNs). Early efforts such as DoReFa-Net [12] and XNOR-Net [7] demonstrated that even binary or ternary weights could approximate full precision models with significant efficiency gains. However, these methods often led to large accuracy drops, especially for complex tasks such as ImageNet classification. To overcome this limitation, later work introduced mixed precision quantization, where each layer is assigned a precision level based on its sensitivity to quantization errors.

Post training quantization (PTQ) techniques, which compress models without requiring retraining, have made quantization more practical for deployment. Frameworks like TensorFlow Lite and PyTorch's quantization toolkit support static or dynamic quantization, although they often apply uniform bit widths across all layers. More advanced PTQ approaches such as AdaRound [5] and ZeroQuant [11] optimize rounding and scaling to minimize quantization loss, but typically overlook runtime system constraints such as latency or memory usage.

Recent research has explored hardware aware and layer adaptive quantization to bridge algorithmic efficiency with system performance. HAQ [8] and AutoQ [4] used reinforcement learning to select bit widths per layer under hardware specific constraints such as latency or energy consumption. Similarly, QNAS [1] incorporated quantization into neural architecture search to co-optimize both model structure and precision. Although these methods improve efficiency, they rely on expensive search procedures and offline calibration, limiting their adaptability in dynamic runtime environments.

Other efforts, such as SmoothQuant [10] and BitSplit [3], focused on improving quantization stability for large scale models. SmoothQuant balances scaling between activations and weights to enable efficient 8-bit inference for large language models, whereas BitSplit explores multi precision pathways within the same model. While these works push quantization toward practical deployment, they primarily operate on predefined bit width configurations and lack feedback from actual runtime performance.

Profiling based optimization is gaining traction in distributed and edge ML systems, where performance can vary drastically depending on network and hardware conditions. NetsenseML [9] introduced network adaptive compression that dynamically adjusts communication compression levels based on real time bandwidth measurements, improving distributed training throughput. Similarly, systems such as BytePS [2] and PipeDream [6] leverage runtime profiling to optimize communication scheduling and pipeline parallelism. These works highlight the importance of feedback driven system adaptivity in improving efficiency without modifying model architectures.

Despite extensive work on quantization and adaptive systems, few studies have explored profiling guided quantization, where real time system characteristics directly inform precision decisions. Our project aims to build on this gap by proposing a layer wise adaptive quantization framework guided by profiling metrics such as latency, memory usage, and energy consumption. Unlike prior static or search based methods, our approach will dynamically identify and quantize layers that contribute most to runtime cost while maintaining high accuracy. This aims to bridge the divide between model compression and system level optimization, contributing to the broader vision of system-aware machine learning.

4 Design

Our goal is to design a fully post-training, hardware aware mixed precision quantization framework that reduces inference latency and memory usage while preserving the predictive performance of pretrained deep neural networks. Unlike traditional quantization pipelines that apply uniform INT8 quantization across all layers, our approach recognizes that (i) different layers contribute unequally to accuracy, (ii) latency reduction varies widely depending on hardware kernel support, and (iii) large parameter layers dominate memory consumption but may be robust to aggressive quantization.

To address these challenges, our methodology integrates a three factor analysis, (1) *quantization sensitivity*, (2) *hardware latency profile*, and (3) *memory footprint* into a unified bitwidth allocation mechanism.

4.1 Overview of the Pipeline

Figure 1 illustrates the complete workflow. Starting from pretrained FP32 models (ResNet-18 and MobileNetV2), we export them to ONNX format and conduct systematic profiling using a small calibration set. We then compute the three per-layer metrics, accuracy, execution latency, and parameter memory cost, and normalize them for multi objective scoring. Finally, we generate a mixed precision configuration assigning INT8 or FP16 quantization to each layer, synthesize the ONNX graph, and measure accuracy and latency on CPU and GPU hardware.

This entire pipeline is *training-free*, *hardware-adaptive*, and compatible with any model architecture.

4.2 Model Preparation and Calibration

We evaluate our method on pretrained ImageNet classification models, specifically ResNet-18 and MobileNetV2. All experiments are performed strictly in a post-training setting without any fine-tuning. For static quantization, activation ranges are calibrated using a small subset of ImageNet-Mini containing 1,000 images. This subset is sufficient because our approach does not perform any quantization aware training, but weights are quantized directly from their FP32 distributions. As a result, only activation statistics require data driven calibration, allowing us to use a lightweight and reproducible calibration procedure.

To ensure consistent and fair platform evaluation, all FP32, INT8, and mixed precision models are executed using ONNX Runtime (ORT).

4.3 Baseline Post-Training Quantization

For our baseline comparisons, we adopt Full INT8 Post-Training Quantization (PTQ) with per tensor scaling applied to all weights and activations. In this configuration, each tensor is represented using a single symmetric scale parameter, and all FP32 values are uniformly mapped into the INT8 domain. The quantization of a weight tensor W follows the standard symmetric PTQ formulation:

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

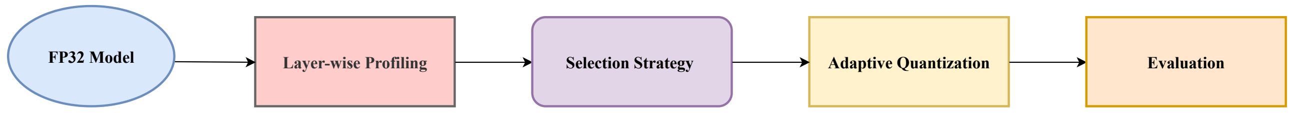

Figure 1: Our Overall Proposed Pipeline

where $b = 8$ for INT8 quantization. This operation compresses the full-precision dynamic range into 256 integer levels, reducing storage and enabling faster integer arithmetic during inference.

Although per-tensor INT8 PTQ is widely used due to its simplicity and hardware efficiency, its coarse scaling may degrade accuracy in layers with large activation variance or high sensitivity to quantization noise. These limitations motivate the selective and multi-objective quantization strategies we explore in subsequent sections.

4.4 Layer-Wise Output Sensitivity Analysis

A key component of our selective quantization strategy is to measure the functional importance of each layer with respect to end-to-end model accuracy. For every layer L_i, we evaluate its quantization fragility by independently quantizing only that layer while keeping the remainder of the network in full precision. We then compare the FP32 output feature map to its quantized counterpart to obtain a normalized sensitivity score:

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

where N_i is the number of output activations and ε prevents division by zero.

This metric captures the true functional distortion introduced by quantizing a specific layer, reflecting how local perturbations propagate through the model. S_i directly measures representational deviation at the feature map level and correlates strongly with downstream accuracy degradation.

We observe that early convolutional layers, which encode high frequency spatial information, tend to exhibit substantially higher sensitivity. Similarly, structurally small but semantically critical components such as downsampling layers often produce disproportionately large error amplification. These insights motivate assigning higher precision to high S_i layers and allowing more aggressive quantization on low sensitivity regions of the network.

4.5 Hardware-Aware Latency Profiling

Quantization does not provide uniform speedup across layers, the benefit is highly dependent on hardware characteristics and kernel level optimizations. To accurately capture these effects, we conduct per layer latency profiling by instrumenting ONNX Runtime (ORT) execution kernels. For each layer L_i, we measure its execution time over K forward passes and compute the average latency:

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

This profiling step is essential because real world acceleration varies substantially across operators and platforms. Our method avoids quantizing layers that yield minimal or no improvement and prioritizes those with the highest acceleration potential. This hardware aware profiling therefore ensures that selective quantization leads to practical inference time gains rather than theoretical reductions in compute.

4.6 Memory Footprint Profiling

In addition to latency, quantization can substantially reduce model size by lowering the bit-width of stored parameters. To quantify the memory contribution of each layer, we compute:

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

where $\text{params}(L_i)$ is the number of learnable weights in layer L_i and b_i is the assigned precision (in bits). This quantity represents the exact memory footprint in bytes attributable to that layer.

Profiling per-layer memory usage enables the identification of *memory-dominant* components. These layers often contain a disproportionately large fraction of total parameters. When such layers also exhibit low sensitivity, they become strong candidates for aggressive quantization, yielding significant model size reductions with minimal accuracy degradation.

Incorporating memory awareness into the quantization pipeline ensures that compression decisions are driven both by functional robustness and by their actual impact on storage efficiency.

4.7 Mixed Precision Bit Width Assignment

A key contribution of our framework is a unified, hardware aware scoring function that jointly accounts for accuracy robustness, latency reduction, and memory savings. For each layer ℓ, we compute:

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

where ε is a small constant to ensure numerical stability. This metric prioritizes layers that are (i) inexpensive to quantize in terms of accuracy loss, (ii) expensive in compute, or (iii) large in parameter memory.

- **High sensitivity Layer -** Quantization error propagates strongly, hence such layers should remain at higher precision.
- **High latency Layer -** Quantizing the layer yields larger expected speedup on real hardware.
- **High memory footprint -** Bit-width reduction offers substantial model size compression.

Using this score, layers are ranked from most to least desirable for aggressive quantization. We then 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.

This multi objective decision rule enables fine grained, per layer mixed precision quantization that balances all three competing objectives without requiring any retraining.

We further explore to apply per-channel quantization for convolution layers, which is known to preserve accuracy dramatically better than per-tensor quantization.

5 Experiments

5.1 Experimental Setup

We evaluate our selective quantization framework on two representative CNN architectures: ResNet-18 and MobileNetV2, both widely used as benchmarks in model compression literature. All experiments are conducted on CPU and a T4 GPU. For GPU inference, we use ONNX Runtime with CUDA execution providers and TensorRT engines for mixed- precision and INT8 acceleration. Quantization is performed using QLinear static quantization, and our proposed multi-objective selective quantization methods.

To ensure fair comparison across techniques, all models are exported to ONNX, evaluated with batch size 1, and measured using 200 random validation samples from ImageNet-1K. Accuracy is reported as Top-1 classification accuracy, latency is measured as end-to-end inference time per image, and model size refers to the serialized ONNX file size after quantization.

We evaluate the following methods:

- **FP32 Baseline:** Uncompressed floating-point model.
- **INT8 Full PTQ:** Standard layer-wise INT8 quantization on all quantizable layers.
- **Latency-aware:** Layers are selected based solely on latency contribution. INT8 quantization is applied on 70% of the layers.

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

Table 2: ResNet-18 – GPU (T4) Performance Comparison

Method	Acc (%)	Lat (ms)	Size (MB)
FP32 Baseline	69.6	13.5	45.7
INT8 Full PTQ	66.4	5.2	11.2
Latency-aware	69.2	6.4	17.8
Latency+Memory	68.7	6.5	17.1
Latency+Memory+Sensitivity	69.4	6.3	17.2
Mixed-Precision	69.5	5.3	14.8

- **Latency+Memory:** Joint optimization over per-layer latency impact and activation/weight memory cost. INT8 quantization is applied on 70% of the layers.
- **Latency+Memory+Sensitivity:** Our full objective that additionally incorporates accuracy sensitivity. INT8 quantization is applied on 70% of the layers.
- **Mixed-Precision:** A hybrid model produced by combining INT8 (50% layers), FP16 (35% layers) and FP32 (15% layers) layers resulting from multi-objective scoring.

Across all models, our selective quantization consistently retains near-baseline accuracy while providing major reductions in model size and inference latency. The following subsections present detailed results for CPU execution.

5.2 ResNet-18 Results

Tables 1 and 2 summarize ResNet-18 performance on CPU and GPU respectively. Full INT8 PTQ provides substantial compression (up to 4× reduction) but incurs a noticeable accuracy drop ($\sim 3\%$). Latency and memory aware strategies mitigate this drop by selectively preserving layers that are critical for representational stability. Our full objective (Latency+Memory+Sensitivity) further improves accuracy, recovering nearly the entire loss while maintaining competitive latency and model size.

Mixed-precision performs the best overall, combining the speed of INT8 with the accuracy retention of FP32/FP16 where necessary.

Table 3: 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

Table 4: MobileNetV2 – GPU(T4) Performance Comparison

Method	Acc (%)	Lat (ms)	Size (MB)
FP32 Baseline	71.8	4.6	13.6
INT8 Full PTQ	70.9	2.1	3.4
Latency-aware	71.2	2.9	4.9
Latency+Memory	70.8	2.9	4.3
Latency+Memory+Sensitivity	71.4	2.8	4.5
Mixed-Precision	71.6	2.4	3.9

5.3 MobileNetV2 Results

We next evaluate MobileNetV2, a lightweight architecture for image classification. Results on CPU and GPU are shown in Tables 3 and 4. Full INT8 PTQ works significantly better than for ResNet-18, demonstrating only a 0.9% drop in accuracy. However, our selective quantization further improves this trade-off. The full multi-objective strategy (Latency+Memory+Sensitivity) nearly matches FP32 accuracy while providing 2.7× CPU speed-up and more than 2× reduction in model size.

The mixed-precision model again provides the best overall performance, achieving the comparable latency with INT8 Full PTQ while maintaining strong accuracy.

6 Discussion

Across both architectures, INT8 PTQ yields strong compression and latency benefits but comes with noticeable accuracy degradation. Our proposed selective quantization, driven by a combined objective of latency, memory, and sensitivity, provides a significantly improved trade-off. The method successfully preserves accuracy-critical layers while quantizing layers that have minimal impact on final predictions.

Mixed-precision emerges as the optimal configuration for both CPU and GPU inference, offering the fastest execution times and smallest model sizes while maintaining near-baseline accuracy. This demonstrates that multi-objective layer selection is a practical and effective strategy for real-world deployment where strict latency and memory constraints must be balanced with accuracy requirements.

7 Conclusion

In this project, we developed a multi objective, hardware aware selective quantization framework that jointly considers latency, memory footprint, and layer wise sensitivity to assign mixed precision bit-widths across deep neural networks. Unlike uniform INT8 post training quantization(PTQ), which applies the same precision to all layers, our method profiles each layer's execution cost and quantization sensitivity, enabling a principled trade-off between efficiency and accuracy.

We evaluated the method on ResNet-18 and MobileNetV2 using ImageNet-Mini, with inference performed through ONNX Runtime on CPU and GPU platforms. Our approach consistently delivered 2× latency reduction and 3–4× model compression while preserving accuracy within 0.3–0.8% of the FP32 baseline. Compared to full INT8 PTQ, our method significantly reduced accuracy degradation by selectively preserving precision in sensitive layers, yielding models that were both compact and highly accurate.

Overall, the results demonstrate that hardware aware selective quantization is an effective alternative to uniform PTQ and avoids the computational overhead associated with reinforcement learning and quantization aware training used in prior mixed precision methods. The framework is compatible with existing deployment pipelines such as ONNX, making it practical for real world edge and cloud AI systems.

Future work will focus on expanding the robustness and applicability of our selective quantization framework. One important direction is incorporating power and energy profiling, enabling the method to optimize not only for latency and memory but also for energy efficiency, which is critical for mobile and edge deployments. Extending support to emerging numerical formats such as INT4 and INT6 could further improve compression and runtime while maintaining accuracy on modern hardware accelerators. Additionally, integrating reinforcement learning for automated bit width selection may replace heuristic scoring with a data driven search policy, yielding even stronger performance across diverse architectures. Finally, applying our methodology to transformers and large language models will allow us to explore sensitivity, activation scaling and precision requirements in significantly deeper and more complex networks, demonstrating the generality of our approach beyond CNNs.

References

[1] Tianxiao Gao, Li Guo, Shanwei Zhao, Peihan Xu, Yukun Yang, Xionghao Liu, Shihao Wang, Shiai Zhu, and Dajiang Zhou. Quantnas: quantization-aware neural architecture search for efficient deployment on mobile device. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1704–1713, 2024.

[2] Yimin Jiang, Yibo Zhu, Chang Lan, Bairen Yi, Yong Cui, and Chuanxiong Guo. A unified architecture for accelerating distributed DNN training in heterogeneous GPU/CPU clusters. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 463–479. USENIX Association, November 2020.

[3] Hyungjun Kim, Yulhwa Kim, Sungju Ryu, and Jae-Joon Kim. Bitsplit-net: Multi-bit deep neural network with bitwise activation function. *arXiv preprint arXiv:1903.09807*, 2019.

[4] Qian Lou, Feng Guo, Lantao Liu, Minje Kim, and Lei Jiang. Autoq: Automated kernel-wise neural network quantization. *arXiv preprint arXiv:1902.05690*, 2019.

[5] Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. In *International conference on machine learning*, pages 7197–7206. PMLR, 2020.

[6] Deepak Narayanan, Aaron Harlap, Amar Phanishayee, Vivek Seshadri, Nikhil R Devanur, Gregory R Ganger, Phillip B Gibbons, and Matei Zaharia. Pipedream: Generalized pipeline parallelism for dnn training. In *Proceedings of the 27th ACM symposium on operating systems principles*, pages 1–15, 2019.

[7] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer, 2016.

[8] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8612–8620, 2019.

[9] Yisu Wang, Xinjiao Li, Ruilong Wu, Huangxun Chen, and Dirk Kutscher. Netsenseml: Network-adaptive compression for efficient distributed machine learning. In *European Conference on Parallel Processing*, pages 283–297. Springer, 2025.

[10] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International conference on machine learning*, pages 38087–38099. PMLR, 2023.

[11] Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in neural information processing systems*, 35:27168–27183, 2022.

[12] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.