Sensitivity-Guided Mixed-Precision Quantization with Structured Pruning for Tiny Vision Models

Prithvi Raj Singh[1] and Satyendra Raj Singh[2]

[1]McNeese State University, Lake Charles, LA, USA
[2]Louisiana Tech University, Ruston, LA, USA
[1]Email: psingh8@mcneese.edu

October 25, 2025

Abstract

Deploying deep neural networks on ultra-low-power microcontrollers remains challenging due to stringent memory and computational constraints. While uniform quantization and pruning techniques can reduce model size, they often fail to exploit per-layer redundancy and result in suboptimal accuracy-efficiency trade-offs. This work presents **SG-MPQP** (Sensitivity-Guided Mixed-Precision Quantization with Structured Pruning), a practical framework that jointly optimizes neural network compression for TinyML vision applications. Our approach combines magnitude-based structured channel pruning with a lightweight layer sensitivity metric that guides per-layer bitwidth assignment (8/6/4 bits) without expensive neural architecture search. The sensitivity score, defined as $S_{l,b} = \Delta A_{l,b}/\text{MemorySaved}(b)$, quantifies the accuracy impact per unit of memory saved when quantizing layer l to bitwidth b. A greedy allocation algorithm then assigns lower bitwidths to less sensitive layers while respecting hardware memory budgets. We validate our method on CIFAR-10 and TinyImageNet using MobileNetV2 and ResNet-20 architectures deployed on ARM Cortex-M4 microcontrollers (Arduino Nano 33 BLE Sense @ 80MHz).Our framework requires no hardware-in-the-loop search, making it accessible for rapid TinyML deployment.

Keywords: TinyML, mixed-precision quantization, structured pruning, model compression, edge AI, ARM Cortex-M4, sensitivity analysis, quantization-aware training

1 Introduction

Motivation: TinyML enables intelligent processing on battery-powered edge devices, but conventional deep learning models exceed the memory (<256KB RAM) and computational budgets of microcontrollers. Uniform compression strategies ignore layer-wise sensitivity, leading to unnecessary accuracy degradation or missed optimization opportunities.

Hypothesis: We hypothesize that combining structured pruning with sensitivity-guided mixed-precision quantization can achieve superior compression ratios while maintaining accuracy, as different layers exhibit varying tolerance to quantization and pruning.

Approach: Our four-stage pipeline: (1) trains baseline models (MobileNetV2/ResNet-20); (2) applies structured channel pruning using magnitude-based importance; (3) performs layer-wise sensitivity analysis to determine optimal bitwidth allocation; (4) fine-tunes with quantization-aware training (QAT) and deploys on ARM Cortex-M4 hardware.

Planned Key Contributions:

- A unified compression framework combining structured pruning and mixed-precision quantization without neural architecture search.

- A lightweight sensitivity metric for efficient per-layer bitwidth assignment.
- Comprehensive evaluation on real TinyML hardware with validated memory and latency improvements.
- Reproducibility document to assist in adoptation of our work and further research in applications of TinyML.

2 Methodology

2.1 Structured Channel Pruning

We adopt magnitude-based pruning via TensorFlow Model Optimization Toolkit. For each convolutional layer l with weights $W_l \in \mathbb{R}^{C_{out} \times C_{in} \times k \times k}$, channel importance is computed as $I_c = \|W_{l,c}\|_1$, and channels with lowest I_c are removed until target sparsity s_l is achieved.

2.2 Layer Sensitivity Analysis

For each layer l and candidate bitwidth $b \in \{8, 6, 4\}$, we quantize only that layer and measure validation accuracy drop $\Delta A_{l,b}$. The sensitivity score is defined as:

$$S_{l,b} = \frac{\Delta A_{l,b}}{\text{MemorySaved}(b)} \tag{1}$$

This metric prioritizes layers where aggressive quantization causes minimal accuracy impact per unit of memory saved.

2.3 Mixed-Precision Assignment

Given total memory budget B_{max}, we solve the optimization:

$$\min_{b_l} \sum_l S_{l,b_l} \quad \text{s.t.} \quad \sum_l \text{Bits}(b_l) \leq B_{max} \tag{2}$$

A greedy allocator assigns lower bitwidths to layers with smallest $S_{l,b}$ values, enabling efficient compression while preserving accuracy.

2.4 Quantization-Aware Training and Deployment

After bitwidth assignment, we perform QAT using TensorFlow Model Optimization Toolkit to recover accuracy. Integer-only TFLite models are deployed on ARM Cortex-M4 via TensorFlow Lite Micro, with latency measured using hardware cycle counters.

3 Planned Experiment

Setup: CIFAR-10 (60K images, 32×32×3) and TinyImageNet (200 classes, 64×64×3); MobileNetV2 ($\alpha = 0.35$) and ResNet-20; ARM Cortex-M4 @ 80MHz (Arduino Nano 33 BLE Sense); TensorFlow, Tensorflow model optimization toolkit, TensorFlow Lite Micro.

4 Final Remarks

We aim to present SG-MPQP, a practical framework combining structured pruning and sensitivity-guided mixed-precision quantization for TinyML vision model. We expect our approach will achieve substantial compression and speedup on real ARM Cortex-M4 hardware without expensive neural architecture search. The lightweight sensitivity metric enables efficient bitwidth assignment, making the framework accessible

for rapid edge AI deployment. Future work will explore sub-4-bit quantization and dynamic precision adjustment for further optimization.

Sample of how we will present results:

Method	**Acc. (%)**	**Size (KB)**	**Lat. (ms)**
FP32 Baseline	-	-	-
8-bit QAT	-	-	-
Pruning Only	-	-	-
Mixed-Precision	-	-	-
SG-MPQP (Ours)	**-**	**-**	**-**

Table 1: We intent to present our findings on CIFAR-10 and TinyImageNet with MobileNetV2 and ResNet-20 architectures deployed on ARM Cortex-M4 microcontrollers as shown in the table.

Reproducibility: We will provide a detailed reproducibility document including code, hyperparameters, and deployment instructions to facilitate adoption and further research in TinyML applications once the work is complete and published.