

DIAGNOSIS AND MITIGATION OF INT8 QUANTIZATION FAILURE IN MOBILENETV3 FOR PLANT LEAF DISEASE CLASSIFICATION

Sopingi^{1*}, Afu Ichsan Pradana², Kevien Ollyvie Jolanda³, Nauval Nabil Iman⁴

Universitas Duta Bangsa Surakarta^{1,2,3,4}

*Correspondence Email: sopingi@udb.ac.id

ABSTRACT

Edge deployment of deep learning models for plant disease classification requires model compression to fit resource-constrained hardware, yet post-training quantization (PTQ) may introduce accuracy loss that is not always well characterized for compact architectures. This progress report presents interim results from a wider study optimizing a fine-tuned MobileNetV3-Small model for tomato leaf disease classification. After training reached 99.71% validation accuracy on a 10-class PlantVillage subset, standard INT8 static PTQ using ONNX Runtime and TensorFlow Lite produced a severe accuracy collapse (10.4-27.4%) despite systematically testing eleven quantization configurations, including per-channel/per-tensor granularity, calibration methods, operator exclusion, and export settings. Diagnostic analysis revealed a persistent class-wise failure pattern rather than random noise, and an explicit shape-mismatch error confirmed a quantizer defect in per-channel bias requantization for grouped convolutions. As a mitigation, dynamic weight-only and dynamic-range quantization schemes were adopted, restoring laboratory accuracy to 99.4% with 29.3-72.0% size reduction, although field-condition accuracy on secondary open field data dropped to 43.1-43.9%, indicating a substantial domain generalization gap that motivates the next research phase.

KEYWORDS

MobileNetV3; Post-Training Quantization; Plant Disease Classification; ONNX Runtime; Edge Computing



This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International

INTRODUCTION

Indonesia's agricultural sector faces persistent losses caused by plant diseases, and early leaf disease detection remains a critical step in modern crop management. Manual

inspection by agricultural experts is still the dominant approach, but it does not scale well and is inconsistent across observers. Convolutional neural networks (CNNs) have demonstrated strong performance on image-based plant disease classification, including on the widely used PlantVillage dataset (Hughes & Salathé, 2015; Mohanty et al., 2016). However, deploying such models in the field is constrained by the limited computing resources and unreliable internet connectivity typical of rural agricultural areas.

Edge computing, where inference is performed directly on local hardware without dependence on cloud servers, is a practical response to this constraint. Compact CNN architectures such as MobileNetV3 (Howard et al., 2019) were specifically designed for this scenario, and are commonly paired with post-training quantization (PTQ) to further reduce model size and inference latency by converting 32-bit floating point weights and activations to 8-bit integers (Gholami et al., 2022).

The wider research this report belongs to targets a WebAssembly-based edge inference service using WASI-NN and WasmEdge (Wang et al., 2023), benchmarked against conventional deployment through ONNX Runtime (Onnimala & Ruwanpathirana, 2023) and TensorFlow Lite (David et al., 2021). Before that comparison can be carried out, the underlying model optimization pipeline, training, ONNX export, and INT8 quantization, must first produce a reliable quantized model. In the course of this preparatory work, standard static PTQ was found to fail unexpectedly for the MobileNetV3-Small architecture used here, motivating a systematic diagnostic investigation that forms the core contribution of this progress report.

This report is therefore scoped to the model optimization stage only: training on PlantVillage, ONNX export, INT8 quantization (both the static PTQ that failed and the dynamic quantization used as mitigation), and benchmarking against ONNX Runtime and TensorFlow Lite on both laboratory and field-condition test data. Implementation of the WASI-NN web service and its comparison against these baselines on constrained edge hardware, as outlined in the original research proposal, is planned for the next stage of the project.

RESEARCH METHOD

A MobileNetV3-Small model pretrained on ImageNet was fine-tuned on a 10-class tomato subset of the PlantVillage dataset (Hughes & Salathé, 2015; Mohanty et al., 2016), split 85% for training and 15% for validation, using standard augmentation (random crop, horizontal flip, colour jitter) and the AdamW optimizer for 15 epochs. The best checkpoint by validation accuracy was exported to ONNX (opset 17) using the legacy TorchScript-based exporter, which avoided a version-conversion defect present in the newer torch.export-based exporter.

INT8 quantization was first attempted using standard static post-training quantization (PTQ) with ONNX Runtime's `quantize_static` and TensorFlow Lite's calibrated converter, using 500 representative images sampled proportionally across all classes for activation calibration. Because this produced anomalous accuracy loss (reported in the Results section), eleven configurations were systematically tested to isolate the cause, varying: calibration method (MinMax, Percentile), quantization granularity (per-channel, per-tensor), activation data type (QInt8, QUInt8), quantization format (QDQ, QOperator), operator exclusion (Squeeze-and-Excitation convolutions, depthwise convolutions), and the ONNX exporter backend (dynamo-based vs. legacy TorchScript-based).

A diagnostic script was developed to inspect model behaviour beyond aggregate accuracy: per-class prediction distribution, per-class recall, and the most common

misclassification target for each class. This tool was applied after every configuration change to determine whether the failure pattern was random noise or systematic.

As a mitigation, two alternative quantization schemes were adopted that avoid static activation calibration: dynamic weight-only quantization restricted to the classifier's Gemm/MatMul layers for the ONNX Runtime model, and dynamic-range quantization (which quantizes convolutional weights but computes activations at runtime without a fixed calibration range) for the TensorFlow Lite model, produced via the onnx2tf conversion toolchain.

All models (FP32 baseline, static PTQ, and the two mitigated INT8 variants) were benchmarked on the same CPU-only environment using two independent test sets: 500 laboratory-condition images from the PlantVillage validation split, and 360 field-condition images from a secondary open dataset on Roboflow Universe (h1-tyt8n, 2024, CC BY 4.0 license), covering 9 of the 10 trained classes. Reported metrics are top-1 accuracy, mean inference latency, and model file size; all inference runs used identical preprocessing (224×224 resize, ImageNet mean/standard-deviation normalization) to keep the comparison fair across runtimes.

RESULT AND DISCUSSION

Table 1 summarizes the final benchmarking results for all evaluated model variants on both the laboratory-condition (PlantVillage) and field-condition (Roboflow secondary) test sets. The FP32 baseline achieved 99.4% top-1 accuracy, closely matching the 99.71% validation accuracy obtained during training, confirming that the ONNX export pipeline and test-tensor preprocessing were implemented correctly.

Table 1. Accuracy, model size, and latency of the FP32 baseline versus the failed static PTQ reference and the two mitigated INT8 variants

Model Variant	Quantization Scheme	Lab Acc. (%)	Field Acc. (%)	Size (MB)	Mean Latency (ms)
FP32 baseline	None	99.4	-	5.85	1.34
ONNX Runtime INT8 (failed reference)	Static PTQ, per-channel, QUInt8, Percentile	27.4	-	4.13*	1.54
ONNX Runtime INT8 (mitigated)	Dynamic weight-only (Gemm/MatMul only)	99.4	43.1	4.13	1.38
TensorFlow Lite INT8 (mitigated)	Dynamic-range (onnx2tf, includes Conv)	99.4	43.9	1.63	1.28

*Model size for the static PTQ reference is listed for completeness; this variant is not recommended for deployment due to the accuracy collapse discussed below.

The static PTQ reference in Table 1 reflects the best top-1 accuracy obtained across eleven systematically tested quantization configurations (calibration method, granularity, activation type, format, and operator exclusion); every configuration produced accuracy between 10.4% and 27.4%, far below the 1-3 percentage-point degradation typically reported for INT8 PTQ in the literature (Gholami et al., 2022). The per-class diagnostic tool revealed that this was not random noise: in every configuration, the same four of ten classes were never predicted at all (0% recall), regardless of which nodes were excluded from quantization. Table 2 reports this pattern for the best-performing static configuration.

Table 2. Per-class recall for the best static PTQ configuration (per-channel, QUInt8, Percentile calibration, QDQ format), illustrating the systematic four-class collapse.

Class	Recall (%)
Bacterial spot	30.0
Early blight	26.0
Late blight	38.0
Leaf mold	18.0
Septoria leaf spot	88.0
Spider mites (two-spotted)	0.0
Target spot	0.0
Yellow leaf curl virus	0.0
Mosaic virus	0.0
Healthy	70.0

Root-cause analysis ruled out several plausible explanations one at a time. Excluding the Squeeze-and-Excitation gating convolutions, and separately excluding all depthwise convolutions, from quantization left the collapse pattern unchanged. Forcing the legacy TorchScript-based ONNX exporter (avoiding a known version-conversion defect in the newer dynamo-based exporter) also had no effect. The decisive evidence came from attempting the QOperator quantization format instead of QDQ: this raised an explicit NOT_IMPLEMENTED/shape-mismatch error inside ONNX Runtime's quantize_bias_static routine, with mismatched array shapes (1152,) versus (72,), during per-channel bias requantization for a grouped convolution. This confirms a defect in the installed ONNX Runtime quantizer's handling of per-channel bias scaling for this layer topology, consistent with similar unresolved reports in the ONNX Runtime and related-tooling issue trackers for MobileNetV3-style architectures.

Because the QDQ format silently produced corrupted results instead of raising this error, the failure went undetected by graph validation and would not have been caught without the per-class diagnostic tooling developed for this study. This distinction, between static PTQ (which requires activation calibration and triggers the defect) and dynamic quantization (which does not perform static activation calibration and therefore avoids it), is what motivated the mitigation strategy shown in Table 1: dynamic weight-only quantization for ONNX Runtime, and dynamic-range quantization for TensorFlow Lite, both of which fully restored laboratory accuracy to 99.4%.

The two mitigated variants differ substantially in the compression they achieve. Because the ONNX Runtime dynamic scheme was restricted to the classifier's Gemm/MatMul layers to avoid the defective Conv quantization path, its convolutional backbone, which accounts for most of MobileNetV3's parameters and FLOPs, remains in FP32, yielding only a 29.3% size reduction and negligible latency improvement over the FP32 baseline. The TensorFlow Lite dynamic-range scheme, in contrast, quantizes convolutional weights directly without static activation calibration, achieving a 72.0% size reduction that is much closer to the compression levels reported in prior PTQ literature (Gholami et al., 2022), while also avoiding the collapse observed in this study's static PTQ experiments.

The most consequential finding for the wider research goal is the accuracy gap between laboratory and field conditions. Both mitigated INT8 models retained the full 99.4% accuracy on PlantVillage but dropped to 43.1-43.9% on the secondary field dataset, a drop of more than fifty percentage points that is consistent across two independently implemented runtimes and quantization schemes. This consistency suggests the drop reflects genuine domain shift rather than a quantization artifact, and is in line with prior findings that PlantVillage models can exploit background and imaging biases specific to its controlled acquisition conditions (Noyan, 2022) and generalize poorly to field imagery

with natural lighting and cluttered backgrounds (Xu et al., 2024). This directly answers the fourth research question posed in the original proposal and indicates that model generalization, not quantization efficiency, is likely to be the primary obstacle to reliable field deployment.

Table 3 breaks this comparison down by class using the mitigated ONNX Runtime model. On the laboratory test set, per-class recall was 94.0-100.0% for every class, with no collapse pattern of any kind, confirming that the dynamic weight-only mitigation fully resolved the static PTQ defect described above. On the field-condition test set, however, per-class recall was highly uneven, ranging from 0.0% (Bacterial spot) to 97.5% (Late blight).

Table 3. Per-class recall of the mitigated ONNX Runtime model (dynamic weight-only INT8) on laboratory versus field-condition test data

Class	Lab Recall (%)	Field Recall (%)
Bacterial spot	100.0	0.0
Early blight	94.0	17.5
Late blight	100.0	97.5
Leaf mold	100.0	7.5
Septoria leaf spot	100.0	55.0
Spider mites (two-spotted)	100.0	55.0
Yellow leaf curl virus	100.0	55.0
Mosaic virus	100.0	70.0
Healthy	100.0	30.0

Table 3 note: Target spot is excluded because the field source dataset does not cover this class (see Research Method).

Unlike the static PTQ failure, this degradation is not a uniform collapse but a systematic bias toward a single class: across the misclassified field images, the model predicted Late blight far more often than any other class, accounting for the majority of errors in eight of the nine evaluable classes (for example, 32 of 33 Early blight errors and all 18 Yellow leaf curl virus errors were misread as Late blight). Late blight itself retained 97.5% field recall, the highest of any class. This pattern suggests the model learned to associate dark, irregular lesion patterns broadly characteristic of Late blight with diseased tissue in general under the more varied lighting and background conditions of field imagery, rather than the finer-grained visual cues that separate diseases under the controlled backgrounds of PlantVillage. This refines the generalization gap reported above from a general accuracy drop into a specific, addressable failure mode, and suggests that targeted data augmentation or field-representative fine-tuning aimed at the confusions in Table 3, rather than blanket regularization, may be the more productive direction for future generalization work.

CONCLUSION

This progress report established the training and quantization pipeline for a MobileNetV3-Small tomato leaf disease classifier as the first stage of a wider study on WASI-NN based edge inference. Standard static INT8 post-training quantization was found to be unreliable for this architecture on the available tooling, producing a reproducible, systematic four-class accuracy collapse that was traced through eleven controlled experiments to a confirmed defect in ONNX Runtime's per-channel bias requantization for grouped convolutions. Dynamic weight-only and dynamic-range quantization were adopted as mitigations, fully restoring laboratory accuracy while achieving 29.3% and 72.0% size reduction, respectively. Evaluation on a secondary field-

condition dataset revealed a substantial and consistent accuracy drop, from 99.4% to 43.1-43.9%, indicating that domain generalization, rather than quantization efficiency, is the more pressing challenge for reliable field deployment. The next stage of this research will implement the WASI-NN web service in Rust on WasmEdge, benchmark it against the ONNX Runtime and TensorFlow Lite baselines reported here on constrained edge hardware, and explore quantization-aware training as a longer-term route to a fully INT8 convolutional backbone without the accuracy collapse observed in this study.

REFERENCES

- Córdova Cárdenas, R., Amor Martín, D. and Gutiérrez Martín, Á. (2025) ‘Edge AI in practice: A survey and deployment framework for neural networks on embedded systems’, *Electronics*, 14(24), 4877. <https://doi.org/10.3390/electronics14244877>
- David, R., Duke, J., Jain, A., Reddi, V.J., Jeffries, N., Li, J., Kreeger, N., Nappier, I., Natraj, M., Regev, S., Rhodes, R., Wang, T. and Warden, P. (2021) ‘TensorFlow Lite Micro: Embedded machine learning for TinyML systems’, *Proceedings of Machine Learning and Systems*, 3, pp. 800–811.
- Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M.W. and Keutzer, K. (2022) ‘A survey of quantization methods for efficient neural network inference’, in *Low-Power Computer Vision*. Chapman and Hall/CRC, pp. 291–326. <https://doi.org/10.1201/9781003162810-13>
- h1-yty8n (2024) *Tomato leaf diseases dataset* [Data set]. Roboflow Universe. Available at: <https://universe.roboflow.com/h1-yty8n/tomato-leaf-diseases-dslxl>
- Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Le, Q.V. and Adam, H. (2019) ‘Searching for MobileNetV3’, *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1314–1324. <https://doi.org/10.1109/ICCV.2019.00140>
- Hughes, D. and Salathé, M. (2015) ‘An open access repository of images on plant health to enable the development of mobile disease diagnostics’, *arXiv*. Available at: <https://arxiv.org/abs/1511.08060>
- Mohanty, S.P., Hughes, D.P. and Salathé, M. (2016) ‘Using deep learning for image-based plant disease detection’, *Frontiers in Plant Science*, 7, 1419. <https://doi.org/10.3389/fpls.2016.01419>
- Noyan, M.A. (2022) ‘Uncovering bias in the PlantVillage dataset’, *arXiv*. Available at: <https://arxiv.org/abs/2206.04374>
- Onnimala, P. and Ruwanpathirana, G.P. (2023) ‘Evaluation of ONNX Runtime performance for neural network inference on edge devices’, *International Journal of Advanced Computer Science and Applications*, 14(5), pp. 321–329. <https://doi.org/10.14569/IJACSA.2023.0140536>
- Ratul, I.J., Zhou, Y. and Yang, K. (2025) ‘Accelerating deep learning inference: A comparative analysis of modern acceleration frameworks’, *Electronics*, 14(15), 2977. <https://doi.org/10.3390/electronics14152977>
- Wang, X., Ying, H., Su, Q. et al. (2023) ‘WasmEdge: A high-performance WebAssembly runtime for edge AI’, *Proceedings of the IEEE/ACM International Conference on Utility and Cloud Computing*, pp. 1–8. <https://doi.org/10.1145/3603166.3632135>
- Xu, M., Park, J.E., Lee, J., Yang, J. and Yoon, S. (2024) ‘Plant disease recognition datasets in the age of deep learning: Challenges and opportunities’, *Frontiers in Plant Science*, 15, 1452551. <https://doi.org/10.3389/fpls.2024.1452551>

Author's short biography

Authors Name:

Name : Sopingi, M. Kom

Affiliation : Universitas Duta Bangsa Surakarta

Phone Number : 085648490678

Email : sopingi@udb.ac.id

Lecturer in the Information Systems study program, Faculty of Computer Science, Universitas Duta Bangsa Surakarta. Research interests include web service architecture, applied machine learning, and edge computing.

Authors Name:

Name : Dr. Afu Ichsan Pradana, M. Kom

Affiliation : Universitas Duta Bangsa Surakarta

Phone Number : 085786200077

Email : afu_ichsan@udb.ac.id

Lecturer in the Informatics Engineering study program, Faculty of Computer Science, Universitas Duta Bangsa Surakarta. Research interests include software engineering and computer vision.

Authors Name:

Name : Kevien Ollyvie Jolanda

Affiliation : Universitas Duta Bangsa Surakarta

Phone Number : 087885258300

Email : kevien.oj@gmail.com

Undergraduate student in the Informatics Engineering study program, Universitas Duta Bangsa Surakarta, contributing to dataset preparation and model benchmarking for this research.

Authors Name:

Name : Nauval Nabil Iman

Affiliation : Universitas Duta Bangsa Surakarta

Phone Number : 089665696749

Email : nauvalnabil100@gmail.com

Undergraduate student in the Information Systems study program, Universitas Duta Bangsa Surakarta, contributing to dataset preparation and model benchmarking for this research.
