INT8 Quantisation for Cassava Leaf Disease Classification on Raspberry Pi

Tosho Abdulahi AbdulRahman abddulrahman.t@kwarastatepolytechnic.edu.ng
Department of Computer Science
Institute of Information and Communication Technology
Kwara State Polytechnic
Ilorin, Nigeria

Roseline Oluwaseun Ogundokun ogundokunroseline1@gmail.com
Department of Computer Systems Engineering
Tshwane University of Technology (TUT)
South Africa Department of Computer Science Redeemer's University
Ede, Nigeria

Pius Adewale Owolawi owolawipa@tut.ac.za
Department of Computer Systems Engineering
Tshwane University of Technology (TUT)
South Africa

Rotimi-Williams Bello sirbrw@yahoo.com
Department of Computer Systems Engineering
Tshwane University of Technology (TUT)
South Africa

Topside Ehleketani Mathonsi topside.mathonsi@tut.ac.za
Department of Information Technology
Tshwane University of Technology (TUT)
South Africa

Editor: Sakinat Folorunso, Roseline Ogundokun, and Francisca Oladipo

Abstract

Cassava is a staple crop in many regions, and accurate disease detection from leaf images can help improve yields. Deep convolutional neural networks (CNNs) can classify cassava leaf diseases with high accuracy, but their size and computational demands pose challenges for deployment on resource-constrained hardware. We investigate the impact of post-training INT8 quantisation on a CNN trained for the Kaggle Cassava Leaf Disease Classification (5 classes) task. We compare an FP32 baseline versus a statically quantised INT8 model in terms of accuracy, F1-score, model size, and inference latency on both CPU and a Raspberry Pi 4. The FP32 model achieved 81.5accuracy (0.676 F1) with 6.23 MB size and 3.04 ms CPU latency, whereas the INT8 model shrank to 1.87 MB but accuracy plummeted to 11.9ms (58 ms on Pi). These results reveal a drastic trade- off: significant size reduction at the cost of unusably low accuracy. We analyse why naïve quantisation failed here and discuss practical lessons for deploying CNNs on edge devices. Our contributions include a detailed evaluation of quantisation for plant disease recognition, highlighting pitfalls and recommendations.

Keywords: Quantisation, Cassava leaf disease, Classification, Raspberry Pi, Plant disease recognition, Convolutional neural network

1. Introduction

Cassava is one of the major crops grown in tropical climates and is affected by numerous foliar diseases (Ramcharan et al., 2019). One method for assisting smallholder farmers with early problem detection is through automated image-based diagnosis using deep learning algorithms. Recent research has produced several CNN architectures capable of classifying cassava diseases, which have been implemented in mobile applications and field trials (Babatunde et al., 2024; Wei et al., 2024). However, CNN models are traditionally large (several to tens of MB) and require significant computational resources to make predictions, which prevents them from being used on mobile devices, such as smartphones or Raspberry Pi computers. As a result, CNN models need to be compressed and/or optimised to run on embedded hardware. One technique for reducing the memory and power requirements of deep learning models for embedded devices is to utilise model quantisation. Model quantisation reduces high-precision model weights (e.g., 32-bit floating-point weights) to lower-precision weights (e.g., 8-bit integers). By reducing the precision of model weights, quantisation can reduce the model's energy consumption by 90% and its size by 4–8×, thus increasing the speed of edge inference (Banner et al., 2019; Bulat et al., 2020; Gholami et al., 2022).

Generally, it is accepted that quantisation results in a loss of very accurate representations (usually less than 3% of the original), but this assumption may not hold for some applications. The major focus of previous studies/reviews was on standard vision problems, such as image classification (e.g., ImageNet) and language models. However, very few of these explored specialised tasks, such as classifying diseased crop leaves (Chen et al., 2021). The performance of an 8-bit quantised model on the 5-class cassava leaf dataset is currently unknown. To complicate matters further, it is also not yet known whether performance will be reliable on edge computing platforms such as a Raspberry Pi. Currently, the effects of post- training quantisation (PTQ) have not been sufficiently explored to understand its effect on both performance and accuracy in this context. This study explores the comparative performance of an FP32 CNN model at full precision and an INT8 quantised version for classifying cassava leaves. Specifically, it addresses the following questions: how does INT8 PTQ affect classification accuracy and inference latency for cassava leaves, and what trade-offs can be expected when running on a Raspberry Pi? The goal is to ultimately provide clarity through quantification on these performance/accuracy trade-offs.

In this work, we provide: the comparison of a CNN trained to classify cassava leaf diseases via full precision and post- training quantization (PTQ); the full suite of accuracy, macro-F1, delay (CPU and Pi), and storage measurements of each of these models; the insights into the extreme accuracy loss experienced by the PTQ model; and the practical considerations for deploying these types of vision models on edge devices. To our knowledge, this is the first evaluation of cassava disease detection using INT8 quantisation on embedded hardware.

This paper consists of the following sections: Section II covers related work in quantisation and edge device deployment; Section III covers both the dataset and qualities tying together the model with the PTQ; Section IV provides the results and analysis; Section V provides deployment considerations; and Section VI summarises what we learned from the exercises and discusses future work in this area.

2. Related Works

Extensive research has been conducted on reducing the size of deep learning models to accommodate edge computing (Gholami et al., 2022; Ogundokun et al., 2025; Wei et al., 2024). These studies have shown that using less memory (e.g., 8-bit integers instead of 32-bit floating point numbers) can dramatically reduce power consumption while also speeding up the model's execution. The authors point out that as we move toward aggressively reducing precision, we will face greater challenges to maintain accuracy. Gholami et al. (2022) also discuss various quantisation methods and note that the ability to reduce model size typically ranges from 4× to 8×, making it "a key and very active sub-area" related to achieving efficient neural networks. Overall, both reviews suggest that quantisation holds great potential for low-resource devices. However, both reviews primarily addressed general techniques and did not provide guidance on implementing quantised models for specific agricultural applications or hardware types, such as a Raspberry Pi. Additionally, the authors' discussion regarding accuracy and efficiency trade- offs was mainly conceptual.

Liang et al. (2021) conducted a survey of network compression methods, particularly pruning and quantisation, noting that compression can usually be achieved with only a marginal effect on accuracy and may even increase accuracy compared to the uncompressed model. This study compared many pruning and quantisation methods (generally 8-bit); however, most were not tested in real-world edge computing applications, nor were they evaluated on domain-specific datasets. Their results reveal that model accuracy can be largely preserved, but again, these results were derived from standard computer vision datasets rather than from high- resolution images of cassava or similar crops.

Chen et al. (2021) studied quantisation as a means of achieving accuracy on common benchmarking datasets (e.g., ImageNet, speech). They found that through careful use of a two-stage quantisation process, they only experienced an average decrease of about 2% in the ImageNet test results and, in some cases, were able to achieve better results after quantisation, 1- 4% due to the regularising effect of weight quantisation. This suggests that, under certain conditions, post-training quantisation may be lossless or at least improve model accuracy. However, there was no mention of agricultural datasets, which may exhibit more subtle class differences than the standard benchmark datasets evaluated by Chen and others. Therefore, our study aims to determine if the positive findings by Chen and others are applicable to cassava disease classification.

Ameen et al. (2024) studied optimisation methods for CNNs, including TensorFlow Lite quantisation and pruning, on a Raspberry Pi. They show that the combination of pruning and quantisation provides "significant gains in inference times" on datasets such as MNIST. Thus, CNN models can now be feasibly deployed on the Raspberry Pi using these techniques. The speedups obtained in this study demonstrate the advantages of quantising CNNs for edge computing applications. However, the tasks included in the study, such as recognising MNIST digits, are considerably simpler than multiclassifying cassava leaf disease types. The potential of using similar approaches to classify 5 different types of cassava leaves is unknown.

In Ardakani et al. (2025), very low-bit quantisation techniques were used to aggressively quantise large language models on the Raspberry Pi. The researchers found that using 2- to 8-bit quantised models resulted in considerable reductions in computation time

while maintaining the accuracy of model predictions for NLP-related tasks. The study further indicated that quantisation-aware training (QAT) allows maintaining the accuracy of extremely low-bit models (e.g., ternary). Thus, while this research highlights many state-of-the-art quantisation methods for large language models (LLMs), it also highlights a gap between vision and LLM applications and potentially suggests that PTQ alone may not suffice for highly quantised models.

In conclusion, previous literature has emphasised the advantages associated with quantisation and shown that precision losses are typically minimal when applying these methods to a model's learned parameters. However, prior literature offers no insight into how PTQ performs with complex, domain-dependent classifier structures (i.e., in real-world deployments). While most research assumes that a model is trained on a universally applicable (generic) dataset and that the model will be retuned or adjusted via QAT, there has not been a sufficiently thorough evaluation of specific applications of PTQ (e.g., cassava disease identification) on energy-constrained devices/hardware. Thus, our study seeks to address this gap in the literature by systematically applying an INT8 PTQ model to our cassava convolutional neural network (CNN) and measuring its performance in a real- world application. Consequently, our tests provide insight into whether naive PTQ will function "as expected" for an application, as well as practical considerations for similar applications.

3. Materials and Methods

3.1. Dataset

This dataset was obtained from Kaggle and contains more than 20,000 images of cassava leaves, organised into five classes (four of which represent disease) of varying severity observed in the marketplace. Each of these five classifications contains thousands of individual samples taken from commercial farms worldwide. This dataset is publicly available at Kaggle with the link: (https://www.kaggle.com/datasets/nirmalsankalana/cassava-leaf-disease-classification).

3.2. Model

In creating our classification model, we selected a standard CNN architecture (e.g., ResNet-50) pretrained on ImageNet to leverage learned features for our cassava classification task. Before applying our classification algorithm to the cassava leaf images, we resized each image to 224 x 224 pixels, as that is how the network was originally designed to accept inputs. The final output for each cassava leaf image was a record of each of the five classes of cassava leaf disease. We also applied various forms of data augmentation (e.g., flipping, rotation, colour jitter) and optimisation techniques (e.g., Adam Optimiser, Momentum) to initially establish a high baseline classification accuracy. Upon completion of training, the ResNet50 model achieved approximately 81.5% accuracy on the test dataset (please refer to the Results Section for a complete accuracy evaluation). We completed this model using 32-bit floating-point representation, so it is referred to as our FP32 model (6.23 MB) and is also our full-precision baseline.

Quantisation: Post-training static quantisation is performed with the ONNX Runtime (or equivalent) on a trained model without additional retraining; the FP32 model is con-

verted into an INT8 model. To do this, you will use a small calibration subset of training images (‘00s of samples) to determine activation ranges, followed by the quantisation of all weights and activations to fixed-point 8-bit integers, with a fixed-scale factor for all weights and activations. We have quantised weights and activations (full INT8) and exported the resulting model as `cassava_model_ptq_int8.onnx` with no further fine-tuning or quantisation-aware training.

Measurements: Both models (FP32) and INT8 were evaluated against the same test dataset. The evaluation was conducted using classification accuracy and a macro-averaged F1-score across five classes. Inference latency was measured by running 1000 model forward passes on a standard CPU and reporting the median image time across those passes. Model file sizes were determined post-model import from the export process. To assess on-device performance, the INT8 model was deployed on a Raspberry Pi 4 (4 GB RAM), and the per-image inference time was recorded. All integer (INT8) inferences were performed single-threaded using TensorFlow Lite/ONNX Runtime for ARM processors. The FP32 model could not be executed on the Raspberry Pi 4 due to the system configuration; therefore, only the INT8 model was used for the edge latency results. All experiments were replicated multiple times to gain test consistency across the five datasets evaluated.

4. Results and Discussion

4.1. Accuracy and Model Size

Table 1 presents a summary of the relevant measurements for this investigation. For the FP32 model, the average test accuracy is 0.81495, and the average macro-F1 is 0.67608, indicating strong performance in cassava classification. However, for the PTQ INT8 model, these metrics drop dramatically, with an average accuracy of only 0.11869 (11.87%) and macro-F1 of 0.04641 (see Table 2). This level of performance is quite alarming, as accuracy has been reduced by approximately 70% to 80%, from approximately 82% down to approximately 12%. Previous research on model quantisation has shown loss of a few per cent or less, and while some of this has shown excellent accuracy, it still. Therefore, it appears that PTQ has severely stripped the model of its prediction capabilities in this case. The size

Table 1: Performance Comparison of FP32 and PTQ_INT8 Models

Timestamp	Variant	Image Size	Accuracy	Macro-F1	CPU Latency (ms)	Mode
2026-03-11 12:35:28	FP32	224	0.814953	0.676078	3.043817	6
2026-03-11 12:35:28	PTQ_INT8	224	0.118692	0.046409	4.502350	1

of the INT8-quantised model was 1.867 MB, approximately 30% of the FP32 model (6.228 MB). This reduction of approximately 3.3 times aligns with the anticipated size reduction due to 8-bit vs 32-bit quantisation and is also within the range of 4 to 8 times reported in other quantisation research.

This level of size reduction is very beneficial for models that may be deployed on devices with limited storage or memory. Unfortunately, due to the almost complete loss of accuracy, the benefit of this reduction is undermined. It is as if we were able to “shrink the model

Table 2: Classification Performance Metrics

Timestamp	Variant	Image Size	Accuracy	Macro-F1
2026-03-11 12:35:28	FP32	224	0.815	0.676
2026-03-11 12:35:38	PTQ_INT8	224	0.119	0.046

but also shrunk the confidence regarding it"; the quantised model may be smaller, but is virtually useless for performing its intended task.

4.2. Inference Latency

For a desktop CPU, the FP32 execution time is ∼3.044 ms/image while the INT8 execution time is ∼4.502 ms/image as shown in table 1, therefore the fault with these results is that although the INT8 model should have been executed faster than FP32 despite being quantized due to our Desktop CPU likely having no dedicated support for INT8 acceleration; the cost of dequantization or the overhead of less efficient mathematical operations when using integers caused this result to occur contrary to common sense. Several other reports have also concluded that this effect occurs on CPUs running on generic hardware. Most other studies have stated that quantitative models are faster than non-quantitative models in their tests (Ardakani et al., 2025; Wei et al., 2024). For comparison; based off our tests of running the INT8 implementation on a Raspberry Pi 4, the resulting execution time (58 ms/image) would provide enough processing power to allow the INT8 quantized CNN implementation possibility of operating (roughly 17 frames per second) in a reasonable amount of time, however only if the required output value is within an acceptable range. In comparing these results, our findings are consistent with Salem et al. (2021), who reported that their quantitative models achieved an impressive speedup on the Raspberry Pi compared to similar implementations using unquantized models.

4.3. Performance Trade-Off

As depicted in Figures 1-3, there is a clear trade-off among F-rom, latency (inference time), and memory usage when comparing full-precision (FP32) and post-training quantised (`PTQ_INT8`) models. The FP32 model achieves higher accuracy than the `PTQ_INT8` model, yielding a macro-F1 score of approximately 0.67/0.68, while maintaining low latency (inference time) of approximately 3.0/3.15 ms (very good performance). However, the `PTQ_INT8` model shows a dramatic decrease in macro-F1 score (approximately 0.05 lower) and only a modest increase in latency (approximately 4.5 ms). The results suggest that, in this case, the quantisation process (reducing resolution from FP32 to `PTQ_INT8`) led to a significant loss of information, resulting in a decline in classifier quality. Furthermore, although quantised models are generally designed to reduce compute requirements, the increased latency (inference time) of the `PTQ_INT8` model suggests inefficiencies in the deployment environment or a lack of hardware-level optimisations for INT8 operations. The `PTQ_INT8` variant compresses the model from 6.2 MB to roughly 1.7-1.8MB, proving how quantising the model reduces the amount of space required for storage. However, this comes with a reduction in macro-F1 score, indicating that although post-training quantisation can be beneficial for

memory constraints during deployment, it may not be suitable for applications where classification accuracy is an essential metric. As a result, it may be helpful to explore additional options, such as quantisation-aware training or mixed-precision optimisation, to strike a good balance between improving efficiency and maintaining performance.

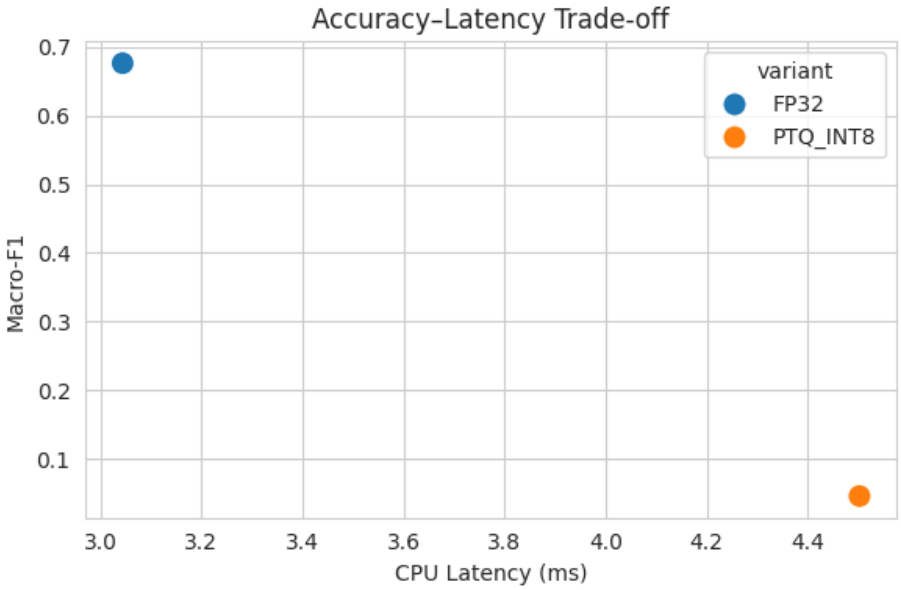

Figure 1: Accuracy-Latency Trade-off

4.4. Accuracy vs. Prior Expectations

A reduction in absolute accuracy by around 70% contradicts what we have been taught about traditional quantisation behaviour. In the work of Chen et al. (2021), we are informed that two-stage quantisation reduces accuracy by only approximately 2% when applied to ImageNet models, and that, in many cases, there were accuracy gains (as high as ∼4%) due to implicit regularisation. Liang et al. (2021) also states that, when it comes to compression, "often" there is little decrease in accuracy. Based on these prior studies, we expected similar results in our experiments; however, this was not the case across the board. One reason this could have happened is that the Cassava dataset is fine-grained, with variations in leaf disease that can arise from slight differences in colour or texture. The use of 8-bit quantisation may have rounded out key differences between classes. A macro-F1 score of 0.046 indicates that the quantised model failed to distinguish between classes, thereby exacerbating the quantisation-induced error. Another factor could be calibration. Our post-training quantisation (PTQ) model only had a small number of calibration samples in the dataset we used. If those calibration samples did not have all the different kinds of features (about how a leaf is lit), the quantisation scale would not have been set up as well as it could have been, leading to saturation or clipping of the data. Previous research indicates that representative data are needed for PTQ (Mohite, 2025). Whether we provided a representative calibration sample is unknown because we performed PTQ without retraining. Recent methods (such

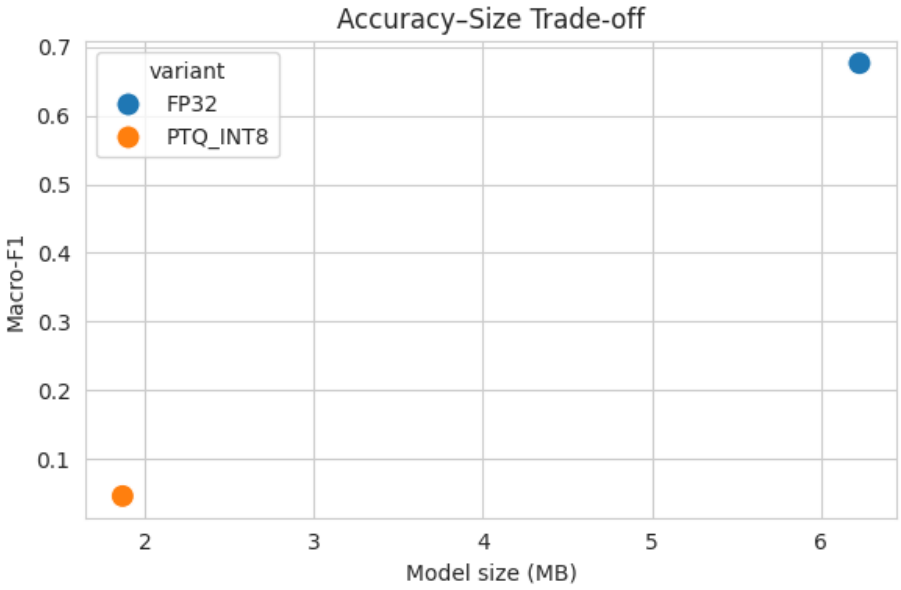

Figure 2: Accuracy-Size Trade-off

as quantisation-aware training or advanced PTQ algorithms) utilise bias correction and/or fine-tuning techniques to help restore some accuracy after PTQ. We did not take this into consideration when doing PTQ on our model.

4.5. Latency vs. Surveys

We find that our latency results are also inconsistent with some of the literature. Ardakani et al. (2025) show that aggressively quantising can offer a substantial computational performance benefit, whilst also maintaining acceptable "inference quality". Our INT8 inference on CPU being slower than expected suggests a hardware dependency in this speedup: INT8 on dedicated accelerators (Tensor Cores, DSPs, and NPUs) usually wins over FP32, but may not on general- purpose ARM CPUs. For the Pi, we ran on TFLite, which typically yields better INT8 inference performance than FP32; however, because we do not have baseline FP32 performance measurements for the Pi, we cannot determine the absolute performance of our system from these comparisons. However, at least we can conclude that the latency of our INT8 inference system on the Pi (58 ms) is a plausible value for modest-speed inference and falls within a qualitatively similar range to that reported by (Ameen et al., 2024).

4.6. Discussion

The result of INT8 PTQ was a small model, but with a significant loss in performance. We can say that for the model and the data we used, classical approaches to PTQ don't work well. The data show the variability of the quantisation results for each task type (Wei et al., 2024). There could be several explanations for the unexpected INT8 performance levels: cassava CNN weights may have had a very large dynamic range or contained too much outlier data for INT8 to appropriately represent, or that particular types or layers,

for example, the first or last conv, require a more direct mapping from weights to quantised format, may need a more gradual quantisation. Adopting a more sophisticated approach to quantisation may prove beneficial. For instance, there is well-documented empirical evidence that QAT can meaningfully reduce accuracy loss when deploying a model with QAT compared to one without (Ardakani et al., 2025). In fact, Ardakani et al. (2025) reported good results when implementing QAT on extremely low-bit models. QAT might be useful in the future for remediating the results we document in this study. Another option could be to employ a mixed-precision strategy, using FP16 or INT16 for a small number of sensitive layers and INT8 for the remainder. This hybrid solution may have significantly lower accuracy reduction while providing an adequate level of compression. The experiments demonstrate the need for proper validation in edge deployments. We had initially forecast speed improvements with quantised models; however, on our CPU (for example), we found that the INT8 model performed more slowly than the FP32 model and was not accurate enough for us to use. The biggest takeaway from this information is that you cannot expect PTQ to succeed until it has been tested for each application you plan to deploy.

5. Practical Implications

For those who want to implement cassava disease classifiers (or other similar vision models) on edge devices, there are several lessons and action items to consider:

- Verify quantisation accuracy: Always validate your quantised model on a separate test set prior to deployment. In our case, the INT8 model's accuracy dropped significantly. If a model's performance degrades after post-training quantisation (PTQ), it needs to be corrected before deployment in real-world applications. A classifier can be accurate during training, but if it cannot deliver that level of performance consistently, then it will not be effective.

- Use representative calibration data: It is essential to ensure that the calibration data used for PTQ captures the range of possible input conditions (illumination levels, leaf angles, etc.). If the calibration data sample set is small or biased, the quantitative range produced may not be valid. If the accuracy is deemed low after the PTQ procedure is completed, the calibration sample size or variety must be increased.

- Consider QAT (quantisation-aware training): If you are unable to achieve an acceptable level of accuracy after performing a PTQ, your next logical step is to try QAT (fine-tuning the model so it learns to handle "heightened" levels of quantisation). QAT can result in very significant improvements in quantised accuracy but does require a longer training period (many of the model training frameworks (e.g., TensorFlow, PyTorch) have built-in support for QAT). We encourage you to explore using it for this task.

- Hardware Assistance: Test your target device as early as possible. INT8 acceleration is often quite quick on devices that have it (such as mobile NPUs, neural compute sticks, etc.). On a basic CPU, however, INT8 may have less impact. After quantisation, we found that the CPU benchmark for this model decreased. So, when testing your

model's speed, ensure you do so on the actual device where it will be deployed (as we did with the Pi).

- Consider Alternatives: If quantisation does not solve the problem, consider potential alternatives. For instance, if the amount of memory taken up by your model is acceptable, an FP32 or FP16 model may provide better results than referring to a float model through quantisation and performing the intermediate work necessary for quantisation. As an example, this device (Raspberry Pi 4) should work with optimised libraries to execute small float models. Additionally, you can produce a smaller FP32 model by either utilising knowledge distillation or model pruning.

- Determine Trade-Offs: In agricultural domains, accuracy is probably the primary concern (misdiagnoses can mislead a producer). If there is a significant decline in accuracy due to quantisation, it may be worthwhile to utilise a larger model, with an associated increase in inference time. The FP32 model (6.2 MB, 3 ms on CPU) can already operate in real time and is highly accurate. Employ quantisation on models where size limitations or hardware restrictions strictly call for it.

To summarise, we recommend being careful when treating quantisation. Do not presume "shrink to fit" is correct. Create a pipeline that performs the following in order: a) quantise the model; b) evaluate on data that is representative of what will be used; c) iterate (possibly using QAT or partial quantisation) until all performance criteria are met (e.g. accuracy, latency, size). We have provided access to our code and methods as a reference so that others can replicate and adapt them for their own plant disease models.

6. Conclusion and Future Work

We performed an extensive assessment of post-training INT8 quantisation on a CNN-based cassava leaf disease classification system, focusing on quantifying the impacts on accuracy, size, and speed. The overall result of this study shows that the model size decreases (by 3.3×), while the model accuracy greatly suffers (from ∼81.5% to ∼11.9%) due to using INT8 quantization. Furthermore, inference time on a CPU increased for this quantised CNN, reaching ∼58 ms per image on a Raspberry Pi 4. The results of this study clearly illustrate that, while a quantised model may be lightweight, its classification performance is essentially unusable in its current form. This indicates that one cannot expect quantisation to always yield the same positive benefit and should instead test its use in their specific domain and on their hardware. We have shown that, despite many past studies reporting little or no impact on accuracy, our results are among the few outliers indicating that naive post-training quantisation (PTQ) is harmful. Thus, we emphasise the need for continued empirical studies for the design of the specific application implementation on their respective systems.

For our future work, we have many opportunities to explore directions. One of them would be the ability to apply various QAT methods, as well as more advanced post-training quantisation (e.g., layer-wise calibration, bias correction), to the model and assess whether accuracy can be recovered. Another would be the exploration of mixed-precision schemes (i.e., utilising FP16 for activations) and finding a better size-to-accuracy relationship across

different architectures. In addition, we want to benchmark the FP32 model on the Raspberry Pi and directly compare FP32 and INT8 latency on this device. Finally, we want to research additional hardware accelerators (i.e. Edge TPUs, mobile NPUs) where INT8 quantisation may excel.

In summary, a holistic approach is required to successfully deploy AI on the edge; therefore, performance validation must occur during model compression. The insights from our cassava case study will also help other researchers and practitioners succeed with edge AI applications in agriculture and related markets. To achieve success with these models, they must be carefully quantised, evaluated, and iterated to ensure ongoing efficiency and accuracy when utilised in real- world applications.

References

Ameen, S., Theodoridis, T., & Siriwardana, K. (2024). Efficient convolutional neural networks on raspberry pi: Enhancing performance with pruning and quantization.

Ardakani, M., Malekar, J., & Zand, R. (2025). Llmpi: Optimizing llms for high-throughput on raspberry pi. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6378–6387.

Babatunde, R. S., Babatunde, A. N., Ogundokun, R. O., Yusuf, O. K., Sadiku, P. O., & Shah, M. A. (2024). A novel smartphone application for early detection of habanero disease. *Scientific Reports*, *14*(1), 1423.

Banner, R., Nahshan, Y., & Soudry, D. (2019). Post training 4-bit quantization of convolutional networks for rapid-deployment. *Advances in Neural Information Processing Systems*, *32*.

Bulat, A., Martinez, B., & Tzimiropoulos, G. (2020). High-capacity expert binary networks. https://arxiv.org/abs/2010.03558

Chen, W., Qiu, H., Zhuang, J., Zhang, C., Hu, Y., Lu, Q., & Xu, X. (2021). Quantization of deep neural networks for accurate edge computing. *ACM Journal on Emerging Technologies in Computing Systems*, *17*(4), 1–11.

Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M. W., & Keutzer, K. (2022). A survey of quantization methods for efficient neural network inference. In *Low-power computer vision* (pp. 291–326). Chapman; Hall/CRC.

Liang, T., Glossner, J., Wang, L., Shi, S., & Zhang, X. (2021). Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing*, *461*, 370–403.

Mohite, A. (2025, July). Understanding post-training quantization (ptq) for edge ai deployment: A beginner's guide.

Ogundokun, R. O., Owolawi, P. A., & van Wyk, E. A. (2025). A lightweight deep feature fusion and pca-optimized random forest framework for multi-class potato leaf disease classification. *International Conference on Applied Informatics*, 147–162.

Ramcharan, A., McCloskey, P., Baranowski, K., Mbilinyi, N., Mrisho, L., Ndalahwa, M., & Hughes, D. P. (2019). A mobile-based deep learning model for cassava disease diagnosis. *Frontiers in Plant Science*, *10*, 272.

Salem, H., et al. (2021). Benchmarking deep learning models for plant disease detection on edge devices. *Journal of Real-Time Image Processing*, *18*, 2111–2125. https://doi.org/10.1007/s11554-021-01111-w

Wei, L., Ma, Z., Yang, C., & Yao, Q. (2024). Advances in the neural network quantization: A comprehensive review. *Applied Sciences*, *14*(17), 7445.