Deep Learning Research: Analysis of Post-Training Integer Quantization with 8-Bit Precision Using Min-Max Scheme on MNIST for Digit Detection

Abstract

The growing demand for deploying deep learning models on resource-limited platforms like edge devices and mobile phones has fueled interest in efficient optimization techniques. Quantization, which reduces the precision of model representations, stands out as a powerful approach. This paper focuses on post-training integer quantization to 8-bit precision, specifically examining the Min-Max scheme.

Post-training quantization offers simplicity and speed by quantizing a fully trained model, avoiding extensive retraining. The choice of 8-bit precision balances model size reduction with accuracy preservation. Representing weights and activations with 8 bits instead of 32-bit floats significantly cuts memory usage and accelerates inference.

This paper aims to review existing literature on 8-bit post-training integer quantization with a focus on Min-Max, explain the theoretical basis and practical aspects of Min-Max quantization, and recommend suitable datasets for evaluating its effectiveness. Subsequent sections will cover background on model quantization, an in-depth look at the 8-bit Min-Max scheme, trade-offs, a literature review, and dataset selection guidance. The need to optimize models for resource-constrained environments and the practicality of post-training quantization underscore the importance of this investigation.

Background on Model Quantization

The remarkable advancements in deep learning have led to the development of increasingly complex models capable of achieving state-of-the-art performance across a wide array of tasks. However, this increasing complexity often translates to high computational and memory requirements, particularly for large language models (LLMs) and intricate network architectures.[4] These substantial demands can pose significant challenges for deploying such models on resource-limited devices, thereby restricting their applicability in real-time scenarios and edge computing environments.[1] Consequently, the field has witnessed a growing focus on model optimization techniques aimed at reducing these resource demands while preserving the model's predictive capabilities.

Quantization stands out as a leading model optimization technique, involving the reduction of the precision of the numerical representations within a neural network.[1] This encompasses a range of methods, broadly categorized into post-training quantization (PTQ), quantization-aware training (QAT), dynamic quantization, and mixed-precision quantization. Post-training quantization is applied to a model after it has been fully trained, converting its weights and activations to lower-precision formats without any further training.[3] Quantization-aware training, in contrast, integrates the quantization process into the training phase itself, allowing the model to learn and adapt to the effects of reduced precision.[1] Dynamic quantization involves quantizing weights and sometimes activations on the fly during inference, often based on the observed range of values.[6] Finally, mixed-precision quantization utilizes different levels of precision for various parts of the model, aiming to optimize efficiency while maintaining accuracy.[3]

Post-training quantization offers several compelling advantages. It is generally simpler and faster to implement compared to quantization-aware training, as it operates on an already trained model.[1] This eliminates the need for extensive retraining, saving significant computational resources and time.[3] As such, PTQ is particularly well-suited for the rapid deployment of pre-trained models in resource-constrained environments.[3] However, a potential drawback of post-training quantization is the possibility of a notable accuracy loss, especially in models that are highly sensitive to precision or when very low bit-widths are employed.[1] To mitigate this, careful calibration of the quantization parameters is often necessary to minimize the degradation in performance.[1]

The availability of diverse quantization techniques provides a spectrum of trade-offs between implementation complexity, computational overhead, and the degree of optimization achieved. Post-training quantization distinguishes itself through its efficiency in scenarios where retraining is either impractical or undesirable. Its ability to directly compress pre-trained models makes it a valuable tool for quickly adapting state-of-the-art models for deployment on a wider range of hardware platforms.

Deep Dive into 8-Bit Integer Quantization With Min-Max Scheme

The adoption of 8-bit precision for integer quantization in deep learning offers a significant set of benefits that contribute to its popularity in model optimization. One of the most prominent advantages is the substantial reduction in model size. Representing each weight and activation with 8 bits instead of 32 bits can lead to a compression ratio of up to 75%, which is crucial for deployment on devices with limited storage capacity.[1] Beyond size reduction, 8-bit integer arithmetic is significantly faster on many hardware platforms, including CPUs, GPUs, and specialized accelerators, compared to floating-point operations, leading to accelerated inference speeds.[1] This lower computational demand also translates to reduced energy consumption, a critical factor for battery-powered mobile and edge devices.[1] Furthermore, many modern hardware architectures are specifically optimized for 8-bit computations, allowing quantized models to leverage these capabilities for enhanced performance.[1]

The Min-Max quantization scheme is a fundamental technique within post-training quantization. It operates by first determining the range of values present in a floating-point tensor, whether it be a weight or an activation tensor.[3] This range is defined by identifying the minimum value (min_val) and the maximum value (max_val) within the tensor.[7] Once the range is established, the next step involves mapping this floating-point range to a target integer range, typically [-128, 127] for signed 8-bit integers or for unsigned integers.[7] This mapping is achieved through the calculation of a scaling factor (scale), which linearly scales the floating-point values to fit within the integer range. The scaling factor is calculated using the formula: scale = (max_val - min_val) / (Q_max - Q_min), where Q_max and Q_min represent the maximum and minimum values of the target integer range, respectively.[7]

In addition to the scaling factor, the Min-Max scheme often employs a zero-point (zero_point), which is an integer offset that ensures the floating-point value of 0 is accurately represented by an integer in the quantized range.[7] This is particularly important for signed integer representations. The zero-point is calculated using the formula: zero_point = round(Q_min - min_val / scale).[7]

The process of quantization involves converting a floating-point value to its integer representation using the calculated scale and zero-point. The formula for quantization is: quantized_value = round(floating_point_value / scale + zero_point).[9] Conversely, to obtain an approximate floating-point value from its quantized integer representation, the process of dequantization is applied using the formula: dequantized_value = (quantized_value - zero_point) * scale.[9]

Variations of the Min-Max quantization scheme exist to accommodate different requirements and data characteristics. Symmetric quantization uses a symmetric range around zero for both the floating-point and integer values, often with the zero-point set to 0.[9] This approach

simplifies the quantization process but may not be optimal for data distributions that are not centered around zero. Asymmetric quantization, on the other hand, allows for a non-symmetric floating-point range and utilizes a non-zero zero-point, which can be beneficial for activations that are inherently non-negative.[3] Furthermore, quantization can be applied either per-tensor, where a single scale and zero-point are used for the entire tensor, or per-channel, where each channel within the tensor has its own set of quantization parameters.[9]

The Min-Max scheme provides a direct and intuitive method for mapping floating-point numbers to integers. However, its effectiveness is intrinsically linked to the distribution of the data within the determined minimum and maximum values. The presence of outliers, which are extreme values far from the majority of the data, can significantly skew the calculated scaling factor. This can result in a reduced quantization resolution for the bulk of the values, potentially leading to a loss of accuracy.[3] The choice between symmetric and asymmetric quantization, as well as the granularity of quantization (per-tensor or per-channel), offers avenues to adapt the Min-Max scheme to specific data distributions and model architectures, ultimately influencing the accuracy of the quantized model.

Trade-offs Between Model Size And Performance

While 8-bit quantization using the Min-Max scheme aims to minimize the loss of accuracy, some level of degradation is possible.[1] This is particularly true if the dynamic range of the weights or activations is not effectively managed, or if the data contains significant outliers that disproportionately influence the quantization parameters.[3] The extent of this accuracy loss can vary depending on several factors, including the specific architecture of the deep learning model, the nature of the task it is designed to perform, and the inherent characteristics of the dataset used for evaluation.[3] However, research has indicated that for many models and tasks, 8-bit post-training quantization can achieve performance that is very close to that of the original floating-point model.[4]

To mitigate potential accuracy degradation associated with 8-bit Min-Max quantization, several strategies can be employed. Calibration plays a crucial role, involving the use of a representative dataset to accurately estimate the dynamic range of the activations within the model.[1] This ensures that the quantization parameters are derived from a realistic distribution of values encountered during inference. Another important aspect is outlier handling. Techniques such as clipping extreme values or employing specialized quantization schemes specifically designed for outliers can help to prevent these values from unduly affecting the overall quantization range and resolution.[3] Bias correction, which involves adjusting the bias terms in the quantized model, can also compensate for some of the errors introduced by the quantization process.[4] In some cases, using mixed-precision quantization, where more sensitive layers of the model are kept at a higher precision while others are quantized to 8 bits, can provide a better balance between efficiency and accuracy.[3] Finally, if the accuracy loss from post-training quantization is unacceptable, quantization-aware training can be considered, where the model is fine-tuned while simulating the effects of quantization, allowing it to learn parameters that are more robust to lower precision.[1]

Ultimately, achieving an optimal balance between the reduction in model size and the preservation of performance often necessitates a careful consideration of these trade-offs, along with meticulous tuning and evaluation. The specific strategies chosen to mitigate accuracy loss will depend on the unique characteristics of the model, the demands of the application, and the level of accuracy that is deemed acceptable.

Literature Review Synthesis

The academic literature contains a wealth of research on post-training quantization, exploring various methodologies, theoretical underpinnings, and empirical evaluations across different deep learning models and tasks.[3] Several papers have specifically investigated integer quantization with 8-bit precision, highlighting its potential for achieving significant model compression and inference acceleration.[1] Within this body of work, the Min-Max quantization scheme has been examined as a straightforward and widely applicable technique.[3]

Research in post-training quantization encompasses a broad range of topics. Zhang et al. [4] introduced a generalized post-training neural network quantization method called GPFQ, based on a greedy path-following mechanism, and demonstrated its effectiveness in quantizing common architectures to low bit-widths with minimal accuracy loss on ImageNet. Xiao et al. [5] proposed SmoothQuant, a training-free solution for 8-bit weight and activation quantization in large language models, addressing the challenge of activation outliers. The use of 8-bit floating-point (FP8) formats as a potentially more efficient alternative to INT8 for post-training quantization has also been explored, as presented in the work by de Amorim et al..[14]

The literature underscores the widespread acceptance of INT8 quantization as a method that offers a good compromise between model efficiency and accuracy.[1] Studies have quantified the substantial memory savings and speed gains achievable through 8-bit representation, often with only a minor impact on the model's predictive performance.[1]

The Min-Max quantization scheme, as a range-based linear quantization technique, has been the subject of specific investigation. It involves mapping the minimum and maximum values of a tensor to the limits of the integer range.[7] The importance of correctly handling the zero value within the floating-point range during quantization has been noted in research, particularly in the context of symmetric quantization.[12] Comparisons between Min-Max quantization and other methods, such as those based on Kullback-Leibler divergence for determining quantization thresholds, have also been conducted.[16]

The trade-offs between model size and performance are a recurring theme in the literature on post-training quantization. Studies consistently analyze the relationship between the bit-width used for quantization, the resulting model size, the impact on inference speed, and

the potential degradation in accuracy.[17] The choice of the quantization technique itself, whether it be post-training or quantization-aware training, as well as the specific strategies employed to handle challenges like outlier activations, significantly influence this balance.[3] The body of literature reflects an ongoing effort to refine post-training quantization techniques, particularly to address the challenges of maintaining accuracy when using lower bit-widths and when quantizing complex models such as transformers and large language models. The emergence of alternative numerical formats like FP8 suggests a dynamic field continuously seeking improved methods for efficient deep learning model deployment.

Dataset Selection And Characteristics

The selection of appropriate datasets is crucial for a thorough and meaningful evaluation of quantization techniques. Datasets used for this purpose should ideally be representative of the intended application domain of the models being quantized.[19] To facilitate comparisons with existing research, it is also beneficial to utilize datasets that are commonly employed in the literature for evaluating quantization methods.[21] For assessing the robustness of quantization, datasets that allow for the evaluation of performance under conditions such as distribution shifts or the presence of noisy data can be particularly valuable.[20] The size of the dataset should be sufficient to yield statistically significant results, while also being manageable for the experimental setup.[4] For post-training quantization, a smaller, yet representative, subset of the training or validation data is often used as a calibration dataset to estimate the dynamic range of activations.[3]

Several datasets are frequently used in the literature for evaluating the performance of quantized deep learning models. In the domain of image classification, CIFAR-10 is a widely adopted dataset consisting of 60,000 32x32 color images categorized into 10 classes.[21] ImageNet, a much larger dataset, contains millions of images spanning 1000 classes and is often used as a benchmark for more complex models.[21] MNIST, a dataset of handwritten digits, is commonly used for introductory tasks and evaluating fundamental quantization effects.[21] These datasets vary significantly in their size, complexity, and the nature of the tasks they are designed for. ImageNet, with its vast number of images and classes, presents a more challenging evaluation scenario compared to CIFAR-10 or MNIST. Similarly, the GLUE benchmark encompasses a range of tasks that test different aspects of language understanding.

While the provided snippets do not offer a consolidated table of 8-bit Min-Max quantization performance on these specific datasets, individual research papers within the literature report findings. For instance, articles discuss the performance of 8-bit quantized models on CIFAR-10 [27] and ImageNet [30], often comparing the accuracy and efficiency against their floating-point counterparts and other quantization techniques.

The choice of dataset is a critical factor in evaluating quantization techniques. The scale and complexity of datasets like ImageNet provide a rigorous test for the robustness of quantization methods, while smaller datasets like MNIST allow for more rapid experimentation and analysis of the fundamental impacts of quantization, which is also the backbone dataset

of this research paper.

TensorFlow Implementation of 8-Bit Min-Max Quantization

TensorFlow, a widely used deep learning framework, provides a comprehensive set of tools and APIs for implementing post-training quantization.[1] The TensorFlow Lite Converter (tf.lite.TFLiteConverter) is a central component in this process, enabling the conversion of trained TensorFlow models into an optimized TensorFlow Lite format, which can include quantization.[1]

Several relevant TensorFlow functions and tools facilitate post-training quantization. The tf.lite.Optimize.DEFAULT flag, when used during conversion, applies a set of default post-training optimizations, which often includes quantizing model weights to 8-bit precision.[34] For achieving full integer quantization, where both weights and activations are quantized to 8-bit integers, the converter.representative_dataset attribute is used to provide a calibration dataset.[34] This dataset allows the converter to estimate the dynamic range of the tensors within the model. To ensure that the converted model uses only integer operations, the converter.target_spec.supported_ops attribute can be set to ``.[34] While the focus here is on post-training quantization. A typical workflow for performing post-training integer quantization in TensorFlow involves the following steps: First, a pre-trained TensorFlow model is loaded. Next, a TFLiteConverter object is created from this loaded model. The optimization flag is then set to tf.lite.Optimize.DEFAULT to enable quantization. If full integer quantization is desired, a function that yields representative data samples is defined and assigned to the converter.representative_dataset attribute.[34] The model is then converted to the TensorFlow Lite format using the converter.convert() method. Finally, the resulting quantized model is saved for deployment.[19]

TensorFlow provides a relatively streamlined process for post-training quantization, with the TFLiteConverter serving as the central tool. The default quantization settings in TensorFlow implicitly utilize the principles of the Min-Max scheme for determining the quantization parameters. Understanding the various options available within the TFLiteConverter, particularly the use of a representative dataset for full integer quantization, is essential for effectively implementing 8-bit Min-Max quantization for deep learning models.

Empirical Analysis And Data Collection Plan

To empirically evaluate the impact of 8-bit post-training integer quantization using the Min-Max scheme, a systematic methodology will be employed. A suitable baseline floating-point model architecture will be selected.[2] Corresponding representative datasets will be chosen for evaluation, for example, MNIST for image classification[1] .

The implementation of 8-bit post-training integer quantization will be carried out using the TensorFlow framework, leveraging the tf.lite.TFLiteConverter with the appropriate settings to ensure full integer quantization. If full integer quantization is used, a calibration dataset, which is a small representative subset of the training or validation data, will be prepared.[3] Both the original floating-point model and the quantized model will then be evaluated on the chosen datasets to assess their performance.[19]

Several key metrics will be used to evaluate the effectiveness of the quantization process. Model size will be measured by the storage space occupied by the model on disk, typically in megabytes (KB).[1] Inference speed will be assessed by measuring the latency, which is the time taken to perform a single inference (e.g., in milliseconds per inference), and/or the throughput, which is the number of samples that can be processed per unit of time (e.g., samples per second) on a designated target hardware platform.[1] Finally, the accuracy of the model on the specific task will be evaluated using standard metrics such as top-1 accuracy for image classification.

The process of data collection will involve running inference with both the original floating-point model and the 8-bit quantized model on the selected datasets. The model size will be directly obtained from the saved model files. Inference speed will be measured by timing the execution of a sufficient number of inference passes on the target hardware. Accuracy will be calculated based on the model's predictions compared to the ground truth labels in the evaluation datasets.

A key component of this empirical analysis will be a direct comparison of the performance metrics between the floating-point model and its 8-bit quantized counterpart. This will allow for a quantitative assessment of the trade-offs in terms of accuracy, speed, and model size resulting from the quantization process.

Theoretical Analysis And Interpretation

The empirical results obtained from the data collection process will be subjected to a thorough analysis. The performance of the 8-bit quantized model will be directly compared to that of the original floating-point model across all measured metrics, including accuracy, inference speed, and model size. This comparison will be interpreted in the context of the existing literature on post-training quantization and the specific characteristics of the Min-Max quantization scheme.

Any observed degradation in accuracy will be carefully analyzed to identify potential underlying reasons. This may involve considering the distribution of weights and activations in the original model, the impact of the Min-Max quantization process on this distribution, and the presence of any outliers that may have disproportionately affected the quantization parameters. The findings will be related back to the theoretical principles of the Min-Max scheme and the specific properties of the datasets and model architectures used in the evaluation.

The improvements in inference speed and the reduction in model size achieved through 8-bit quantization will be quantified and discussed. These gains will be contextualized within the broader goals of model optimization for efficient deployment on resource-constrained devices. The analysis will also consider the potential impact of the target hardware platform on the observed speedups, as the efficiency of integer arithmetic can vary across different hardware architectures.

By analyzing the empirical results in conjunction with the theoretical understanding of the Min-Max quantization scheme and the findings reported in the literature, this section will aim

to provide a comprehensive interpretation of the effectiveness and limitations of 8-bit post-training integer quantization for deep learning projects.

Conclusion And Future Directions

This paper has provided a comprehensive exploration of post-training integer quantization with 8-bit precision, focusing on the Min-Max quantization scheme. The literature review highlighted the increasing importance of model optimization techniques for efficient deployment, with quantization emerging as a key approach. The 8-bit precision offers a compelling balance between model size reduction and performance preservation, making it a widely adopted choice in the field. The Min-Max quantization scheme, while straightforward to implement, necessitates careful consideration of the dynamic range of the model's parameters to minimize potential accuracy loss.

The planned empirical analysis will provide valuable insights into the practical effectiveness of this quantization technique on representative deep learning models and datasets. By measuring the impact on model size, inference speed, and accuracy, a quantitative assessment of the trade-offs involved can be made.

The theoretical analysis of these empirical results, interpreted in the context of existing literature, will further enhance the understanding of the strengths and limitations of the 8-bit Min-Max quantization scheme.

Future research could explore more advanced post-training quantization techniques that may offer improved accuracy, particularly at lower bit-widths. Investigating the impact of different calibration methods on the performance of Min-Max quantized models could also yield valuable insights. Furthermore, applying this quantization scheme to a wider range of modern model architectures, including transformers and large language models, and evaluating its effectiveness across diverse tasks, would contribute to a more comprehensive understanding of its applicability and limitations.

Works cited - Bibliography

1. How to 8-bit quantize large models using bits and bytes - AI Accelerator Institute. https://www.aiacceleratorinstitute.com/how-to-8-bit-quantize-large-models-using-bits-and-bytes/
2. What Is int8 Quantization and Why Is It Popular for Deep Neural Networks? - MathWorks. https://www.mathworks.com/company/technical-articles/what-is-int8-quantization-and-why-is-it-popular-for-deep-neural-networks.html
3. Advances in the Neural Network Quantization: A Comprehensive Review - MDPI, https://www.mdpi.com/2076-3417/14/17/7445
4. Post-training Quantization for Neural Networks with Provable Guarantees | SIAM Journal on Mathematics of Data Science, https://epubs.siam.org/doi/10.1137/22M1511709
5. [2211.10438] SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models - arXiv, https://arxiv.org/abs/2211.10438

6. About the int8 training question - quantization - PyTorch Forums, https://discuss.pytorch.org/t/about-the-int8-training-question/169907

7. Post-training Quantization for Neural Networks with Provable Guarantees - arXiv, https://arxiv.org/abs/2201.11113
8. LiteRT 8-bit quantization specification | Google AI Edge - Gemini API, accessed April 20, 2025, https://ai.google.dev/edge/litert/models/quantization_spec
9. Introduction to Weight Quantization | Towards Data Science, https://towardsdatascience.com/introduction-to-weight-quantization-2494701b9c0c/
10. Quantization - Hugging Face, accessed April 20, 2025, https://huggingface.co/docs/optimum/concept_guides/quantization
11. Weight quantization using min max calibration - TensorRT - NVIDIA Developer Forums, https://forums.developer.nvidia.com/t/weight-quantization-using-min-max-calibration/278330
12. Post training 4-bit quantization of convolutional networks for rapid-deployment - OpenReview,, https://openreview.net/pdf?id=Syel64HxLS
13. Efficient Post-training Quantization with FP8 Formats - MLSys Proceedings, https://proceedings.mlsys.org/paper_files/paper/2024/file/dea9b4b6f55ae611c54065d6fc750755-Paper-Conference.pdf
14. Quantization - Neural Network Distiller - Intel Labs, https://intellabs.github.io/distiller/quantization.html
15. Quantization noise in low bit quantization and iterative adaptation to quantization noise in quantizable neural networks - ResearchGate, https://www.researchgate.net/publication/357177732_Quantization_noise_in_low_bit_quantization_and_iterative_adaptation_to_quantization_noise_in_quantizable_neural_networks
16. Understanding Model Quantization in Large Language Models | DigitalOcean, https://www.digitalocean.com/community/tutorials/model-quantization-large-language-models
17. Comparing Different Quantization Methods: Speed Versus Quality Tradeoffs - RunPod Blog, https://blog.runpod.io/comparing-different-quantization-methods-speed-versus-quality-tradeoffs/
18. Post-training Quantization — PyTorch Lightning 2.5.1 documentation, https://lightning.ai/docs/pytorch/stable/advanced/post_training_quantization.html
19. Benchmarking the Reliability of Post-training Quantization: a Particular Focus on Worst-case Performance - OpenReview, https://openreview.net/pdf?id=iKBdmOVJ5T
20. Machine Learning Datasets - Papers With Code, https://paperswithcode.com/datasets?mod=images
21. Machine Learning Datasets - Papers With Code, https://paperswithcode.com/datasets?task=image-classification

22. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding,https://www.researchgate.net/publication/334116956_GLUE_A_Multi-Task_Benchmark_and_Analysis_Platform_for_Natural_Language_Understanding

23. A Comprehensive Evaluation of Quantization Strategies for Large Language Models - arXiv,, https://arxiv.org/html/2402.16775v1
24. Quantize NLP models with Post-Training Quantization in NNCF, https://docs.openvino.ai/2023.3/notebooks/105-language-quantize-bert-with-output.html
25. Accurate Post Training Quantization With Small Calibration Sets - Proceedings of Machine Learning Research, http://proceedings.mlr.press/v139/hubara21a/hubara21a.pdf
26. Post-training Quantization — PyTorch Lightning 1.9.4 documentation, https://lightning.ai/docs/pytorch/1.9.4/advanced/post_training_quantization.html
27. Dynamic Activation Step Size for Post-Training Quantization - Practical-DL Workshop, https://practical-dl.github.io/2022/long_paper/07.pdf
28. README.md - TimS-ml/Handwritten-Quantization - GitHub, https://github.com/TimS-ml/Handwritten-Quantization/blob/master/README.md
29. Post Training Quantization - MindSpore Lite, https://www.mindspore.cn/lite/docs/en/r1.10/use/post_training_quantization.html
30. PTQ4ViT: Post-Training Quantization for Vision Transformers with Twin Uniform Quantization - arXiv, https://arxiv.org/html/2111.12293v3
31. Clipping-Based Post Training 8-Bit Quantization of Convolution Neural Networks for Object Detection - MDPI, https://www.mdpi.com/2076-3417/12/23/12405
32. Post-training quantization | Google AI Edge - Gemini API, https://ai.google.dev/edge/litert/models/post_training_quantization
33. tf.quantization.fake_quant_with_min_max_vars | TensorFlow v2.16.1, https://www.tensorflow.org/api_docs/python/tf/quantization/fake_quant_with_min_max_vars
34. Post-Training Quantization in PyTorch using the Model Compression Toolkit (MCT) - GitHub, https://github.com/sony/model_optimization/blob/main/tutorials/notebooks/mct_features_notebooks/pytorch/example_pytorch_post_training_quantization.ipynb
35. Quantization of Image Classification Models - OpenVINO™ documentation, https://docs.openvino.ai/2025/notebooks/image-classification-quantization-with-output.html