# CS 243 Final Project Midterm Report: Performance and Cost Evaluations of Quantization Aware Training

**Michal Kurek** [1] **Natnael Teshome** [1] **Zev Minsky-Primus** [1]

## Abstract

This paper examines the tradeoffs in model quantization for Deep Neural Networks (DNNs), proposing a partial quantization strategy that targets the final layers to maintain accuracy while reducing model size. Our hypothesis—that quantizing early or middle layers, such as attention mechanisms, incurs greater accuracy loss than final layers—guides our empirical analysis on both small and large models. Preliminary results affirm this approach, suggesting a viable pathway for deploying efficient DNNs in resource-limited environments. Challenges such as extended training times and storage constraints are addressed through distributed training.

## Introduction

Deep neural networks(DNNs) have been used extensively for multiple tasks, showing great performance. The increasing size and complexity of these models poses challenges in resource-constrained environments. These resources can be compute, storage, or communication, becoming a bottleneck in the training process.

Model quantization has emerged as a potent solution to this problem, enabling reductions in storage and computational overhead by using lower bit-width representations for weights and activations. Full model quantization can offer substantial size reductions—for example, when converting from 32-bit floating point numbers to 8-bit integers. Yet, the adoption of full quantization is not without consequence, as it may lead to unacceptable accuracy loss, thus posing a dilemma for practitioners who must balance model efficiency with performance.

This project asseses the tradeoff between full model quantized aware training and non-quantized training with an emphasis on the hypothesis that quantizing the final layers of DNNs can offer a compromise, retaining the bulk of the model's accuracy while still achieving meaningful reductions in size. The motivation for this focus arises from findings that indicate quantizing critical components, such as the attention mechanisms within transformers, often results in significant performance degradation. Conversely, prior work has shown that the last layers of neural networks, often fully connected layers, can be quantized with a more modest impact on the overall accuracy (1).

By drawing on these insights, we posit that a strategic approach to quantization, targeting the final layers of a network, can yield a favorable balance. This approach is particularly relevant in use-cases where model size dictates feasibility, such as in mobile applications, or where model update frequency is limited by network bandwidth, such as in distributed training scenarios. Additionally, by maintaining higher precision in the initial and middle layers, we preserve the delicate structures that are critical for complex pattern recognition and data representation, which are often the first to suffer under aggressive quantization strategies.

In this paper, we provide a systematic study of partial quantization. We do empirical analysis of the impact on accuracy, size, and network bandwidth of the resulting models after quantizing the last layers of a chosen DNN model. We aim to provide a mathematical formula of the accuracy reduction and the size reduction as a function of the number of layers quantized, thus enabling practitioners to systematically determine the number of layers to quantize for their specific resource-constrained environment.

## Design

Our project's testing infrastructure is currently based on EC2 G2 instances (g2.8xlarge), which are equipped with four high-performance NVIDIA GRID GPUs. These instances provide the necessary computational power for model fine-tuning, which we aim to perform in a distributed manner.

Development has been taking place locally in a Jupyter notebook, mainly due to the modest scale of our initial toy models. We've encountered limitations with Google Colab's frequent timeouts and want to conserve our AWS credits for

*Equal contribution [1]Harvard University. Correspondence to: Michal Kurek <mkurek@college.harvard.edu>, Natnael Teshome <nteshome@college.harvard.edu>, Zev Minsky-Primus <nteshome@college.harvard.edu>.

the intensive training phase of larger models.

Given the constraints of our resources—both in terms of AWS credits and time—we have scaled down our initial experiments. We are now focusing on ResNet and the ImageNet dataset. For context, training ResNet from scratch on the ImageNet dataset took around 3.5 days on a GPU cluster back in 2019, involving roughly 23 million parameters. In contrast, GPT-1 has 117 million parameters. Since our training process involves testing different quantization levels of quantization for a given model, we need to run many similar experiments, which could compound our time.

As the project evolves, we are also considering BERTmini as a more feasible alternative for our experiments. Despite its smaller size, BERTmini is still representative of the scale of large language models and can provide insights into the impact of quantization. Our focus will primarily be on fine-tuning pre-trained models rather than training from scratch. Quantization-aware training introduces specific modifications that allow for the model's weights to be representative even after quantization, thereby converting a model with pre-trained weights into a quantization-aware one efficiently.

## Implementation

Over the past month, our primary focus has been on establishing our AWS infrastructure. This included identifying suitable EC2 instances with GPU support and learning how to efficiently train models using TensorFlow's GPU capabilities. Coming from a systems background rather than machine learning, this required significant effort.

Our codebase, including testing (Jupyter) notebooks and this midway report, is available on our shared GitHub repository at https://github.com/zevbo/CS243-final-project/. We opted for TensorFlow over PyTorch due to our greater familiarity with the former and its robust and extensible API that facilitates model quantization. This API supports not only whole-model quantization but also layer-specific and custom quantization approaches.

Additionally, we plan to exploit TensorFlow's distributed features for the model fine-tuning process. For our final quantized models and inference testing, we've chosen TensorFlow Lite (TFLite) as our preferred backend. Despite some intricacies associated with TFLite, we believe it aligns well with our objectives.

Our benchmarks will primarily examine the relationship between model accuracy, training time, and size. While we anticipate that smaller models—resulting from more extensive or aggressive quantization—may show reduced performance, this isn't always a given. Previous findings about models like VGG16 have piqued our curiosity about how large language models and bigger CNNs will respond to targeted layer quantization. If feasible, we also aim to monitor network bandwidth and overheads during the fine-tuning process across EC2 instances. We're interested in determining the extent to which quantization might reduce these factors.

## Evaluation

Our initial experiments involved a series of toy models—simple neural networks we designed ourselves—to streamline the targeting of specific layers for quantization. The primary objectives were to validate our TensorFlow setup's capability to train with GPUs and to assess our ability to quantize models, both through quantized-aware training and post-training quantization. We confirmed that both approaches are feasible within TensorFlow's default 8-bit integer quantization. Although TensorFlow offers a QuantizationWrapper class for customization, we plan to invest more research into implementing 4-bit quantization.

The results were promising. Full quantization of a model typically resulted in a fourfold size reduction and a slight accuracy drop—up to 1 percent in our small network trials. However, we anticipate a greater accuracy loss when quantizing more complex structures, such as the attention layers in Large Language Models (LLMs).

Our experiments also sought to determine the impact of quantizing different layers. As hypothesized, the quantization of initial layers led to a more significant accuracy decrease compared to the final layers. This effect could be attributed to both the larger quantity of weights in the early layers and the potential for early-stage quantization errors to propagate through subsequent layers. Interestingly, when quantizing a specific layer in TensorFlow, the model summary indicated that the preceding layer was also quantized. We believe this is due to the need for the activation function to match the quantization of the subsequent layer's input.

Further investigation is required to understand whether this automatic quantization affects only the activation function or if it extends to the weights, raising the question of why such a change wouldn't cascade back through all preceding layers.

Moving forward, our research will focus on applying these quantization strategies to larger models, which presents a unique set of challenges detailed in the subsequent section.

## Challenges

Our main challenge is identifying a model that is appropriate for our comprehensive empirical analysis. We want to find a model that satisfies the following requirements:

1. The model has a pre-trained model weight as it is better to start with pre-trained weights to converge faster.
2. The implementation of this model to be able to identify the layers to quantize
3. A large enough and complicated model with different distinct parts to experiment with quantizing the different types of layers.

Once we choose the appropriate model and write a script for local training, we plan to do the actual training in a distributed setting using the AWS clusters. Hence, this is another major step in our project.
Another challenge is if training our model takes a very long time, we could potentially run out of our AWS credit.

## References

[1] FAN, C. Quantized transformer.