Quantization of ML Models

Code

Code: `github.com/sueszli/byte-sized-gains/`”

Introduction

- Modern and polished design
- Serif font for elegance
- Clean layout for better readability

Key Features

- **Metropolis theme**: Sleek and contemporary
- **Crane color theme**: Professional color scheme
- **16:9 aspect ratio**: Optimized for widescreen displays

Thank You

Questions?

Introduction

In recent years, object detection models (ODMs) and large language models (LLMs) have made unprecedented progress in computer vision and natural language processing, respectively. These deep neural networks have found applications across various domains, from autonomous vehicles and surveillance systems to chatbots and content generation. However, the increasing complexity and size of these models have led to significant computational and energy demands, posing challenges for widespread deployment and raising concerns about their environmental impact.

As the global community grapples with climate change and the urgent need for sustainable technologies, the optimization of ODMs and LLMs has become a critical area of research. Quantization, a technique that reduces the precision of model parameters and activations, has emerged as a promising approach to address these challenges. By converting high-precision floating-point representations to lower-precision formats, quantization can substantially decrease the model's memory footprint and compute usage while compromising predictive performance

ODM Quantization

In the first part of this project we quantize an object detection model.

Challenges

We started this task off by aiming for the stars and comparing the best models we could find on the public "papers with code" leaderboard for the COCO 2017 dataset [1]. Then we ran our own experiments to find the most representative models from each architecture family [2]. We then noticed the DETR family to perform the best, particularly the "facebook/detr-resnet-101-dc5" model, as it generalizes well and was trained on the same dataset we are using, namely COCo-2017.

But after implementing the entire evaluation pipeline for our experiments in PyTorch we realized Torch XLA builds a shared library, `_XLAC.so` that needs to link to the version of Python it was built with (currently 3.10 or 3.11). And in order to ensure that `import _XLAC` can succeed, we had to update the `LD_LIBRARY_PATH` to the lib directory of our Python environment. This was a

LLM Quantization

In the second part of our project, we focused on quantizing a large language model (LLM). We selected the SmolLM-135M model, a smaller variant of LLMs that is more suitable for edge devices and resource-constrained environments.

Methodology

Our methodology involved quantizing this model using AutoGPTQ with three different configurations: int8, int4, and int2, representing increasingly aggressive levels of quantization. For the quantization process, we utilized the GPTQ (Generative Pre-trained Transformer Quantization) method, which is implemented in the AutoGPTQ library. We configured the quantization process with a group size of 64 and used a subset of 100 samples from the LAMBADA dataset as a representative dataset for calibration. This step is crucial for ensuring that the quantized model can accurately represent the distribution of activations in the original model.

Final Thoughts

The experiments we conducted are not representative of data-driven performance analytics research because they fall into several common pitfalls highlighted in the article "Always Measure One Level Deeper"[3]. Firstly, the experiments seem to rely heavily on superficial measurements, such as overall model size and basic precision metrics, without getting into deeper system behaviors or the underlying factors affecting performance. This superficiality can lead to incomplete and potentially misleading conclusions about system performance.

Moreover, there appears to be a confirmation bias in the methodology. The experiments focus on quantization techniques that reduce model size and memory footprint but do not sufficiently explore how these changes impact other critical aspects of performance, such as computational efficiency or energy consumption. This selective reporting can skew results to favor certain outcomes without a comprehensive analysis of all potential impacts. Additionally, the experiments might