LLMEasyQuant - An Easy to Use Toolkit for LLM Quantization

Dong Liu[1] dliu328@wisc.edu
Meng Jiang[2] mjiang2@nd.edu
Kaiser Pister[1] kaiser@cs.wisc.edu

[1]University of Wisconsin-Madison,
[2]University of Notre Dame

Abstract

Currently, there are many quantization methods appeared for LLM quantization, yet few are user-friendly and easy to be deployed locally. Packages like TensorRT[NVI24] and Quanto[Fac24] have many underlying structures and self-invoking internal functions, which are not conducive to developers' personalized development and learning for deployment. Therefore, we develop LLMEasyQuant, it is a package aiming to for easy quantization deployment which is user-friendly and suitable for beginners' learning.

The code of LLMEasyQuant can be found at code link[1].

1 Introduction

Quantization is the process of mapping a large set of input values to a smaller set of output values, often integers. It is a key technique in digital signal processing where continuous signals are mapped to discrete digital values, and it reduces the data's precision to make storage and computation more efficient while attempting to retain essential information.

Traditional quantization methods like uniform and scalar quantization, emphasizing their utility in diverse applications such as telecommunications and consumer electronics.

The quantization process can be described by:

$$Q(x) = \text{clamp}\left(\left\lfloor \frac{x - \min(X)}{\Delta} + 0.5 \right\rfloor + z, -128, 127\right)$$

where:

- x is a floating-point value from the dataset X,

[1]https://github.com/NoakLiu/LLMEasyQuant

- Δ (scale) is calculated as $\frac{\max(X)-\min(X)}{255}$,
- z (zero point) is -128 or calculated to shift the scale,
- clamp function ensures values are kept within the $[-128, 127]$ range.

2 Methodology

LLMEasyQuant is a toolkit designed to simplify the process of quantizing large language models (LLMs). To achieve an easy quantization method, we developed LLMEasyQuant as an easy-to-use toolkit for straightforward quantization.

- absmax: scale $= \frac{127}{\max(|X|)}$
- zeropoint: Shifting the tensor values based on a computed zero-point
- smoothquant: A smoothing technique to the quantization process.[XLS+24]
- symquant: Symmetric scaling based on the absolute maximum value.[YAZ+22]
- zeroquant: Adjust the numeric range of input data so that zero values in the original data can be represented exactly in the quantized format. [YAZ+22]
- simquant: A quantization technique by KV Cache Quant. [HKM+24]

3 Implementation

3.1 ZeroQuant: Zero-point Quantization

3.1.1 Definition and Concept

Zero-point quantization, or ZeroQuant [YAZ+22], focuses on adjusting the numeric range of data so that zero values are represented exactly by zero in the quantized format. This technique is particularly useful in optimizing quantization schemes where the preservation of zero values is crucial, such as in sparse datasets used in machine learning and signal processing.

3.1.2 Mathematical Formulation

The quantization process using ZeroQuant can be described by the following formula:

$$Q(x) = \text{clamp}\left(\left\lfloor \frac{x - \min(X)}{\Delta} + z \right\rfloor, -128, 127\right)$$

where:

- x is the value to be quantized,
- $\min(X)$ is the minimum value in the dataset,

- Δ is the quantization step size, calculated as $\frac{\max(X)-\min(X)}{255}$,
- z is the zero-point, adjusted so that $\min(X)$ maps to 0,
- The output is clamped to the range $[-128, 127]$, typical of 8-bit signed integers.

3.1.3 Applications

ZeroQuant is commonly used in areas where data sparsity is significant, such as in the storage and transmission of high-dimensional but sparse datasets. In neural networks, employing ZeroQuant can significantly reduce the computational costs and power consumption during inference on edge devices.

3.2 Symmetric 8-bit Quantization

3.2.1 Concept

Symmetric 8-bit quantization [FFBL18] involves mapping both positive and negative values of a dataset uniformly around zero, optimizing the quantization process for symmetric data distributions.

3.2.2 Mathematical Formulation

The symmetric quantization can be described with:

$$Q(x) = \text{clamp}\left(\text{round}\left(\frac{x}{s}\right), -128, 127\right), \quad s = \frac{\max(|X|)}{127.5}$$

where s is the scale factor calculated based on the maximum absolute value of the data, ensuring all quantized values fall within the 8-bit integer range of -128 to 127.

3.2.3 Applications

This quantization method is extensively used in hardware implementations of neural networks, where maintaining data symmetry simplifies the computational requirements and enhances performance consistency.

3.3 Layer-by-Layer Quantization

3.3.1 Concept

Layer-by-layer quantization [TOW+23] is a technique applied in deep learning where each layer of a neural network is quantized independently to maintain accuracy while reducing the overall model size and computational demand.

3.3.2 Quantization Procedure

The process involves:

- Calculating the quantization parameters for each layer separately.
- Adjusting each layer's weights and biases according to the quantization rules defined for symmetric or asymmetric quantization methods.
- Optionally recalibrating layer parameters during or after the quantization to optimize performance.

3.3.3 Implementation Example

Here is an example of applying layer-by-layer quantization:

```
# Pseudocode for Layer-by-Layer Quantization
for layer in model.layers:
    scale = compute_scale(layer.weights)
    quantized_weights = quantize(layer.weights, scale)
    layer.weights = quantized_weights
```

3.3.4 Applications

Particularly beneficial in deploying deep neural networks on mobile or embedded devices where memory and processing power are limited.

3.4 Symmetric 8-bit and Layer-by-Layer Quantization

3.4.1 Layer-by-Layer ZeroQuant Function

Applies quantization per layer in a model:

- Encode input and compute unquantized outputs.
- For each layer i:
  - Freeze all but layer i.
  - Update i-th layer's parameters by quantization.
- Compute loss and update using:

$$\text{loss} = \text{MSE}(\text{teacher\_outputs}[i], \text{quantized\_outputs}[i])$$

- Optimize with gradient descent.

3.5 SimQuant: A Novel Approach by KV Cache Quant

3.5.1 Definition and Concept

SimQuant [HKM+24], developed by KV Cache Quant, represents a groundbreaking advance in the field of data quantization. This technique is designed to optimize the efficiency of data representation while maintaining high accuracy, particularly in environments where computational resources and storage are limited. Unlike traditional quantization methods that apply fixed parameters across various datasets, SimQuant introduces a dynamic adjustment mechanism that tailors the quantization process to the specific statistical properties of the dataset in use.

3.5.2 Mathematical Formulation

The core of the SimQuant technique is based on the formula:

$$Q(x) = \text{round}\left(\frac{x - \min(X)}{\Delta}\right) + Z$$

where:

- x is the data point being quantized,
- $\min(X)$ is the minimum value in the dataset,
- Δ represents the quantization interval, which is adaptively calculated,
- Z is the zero-point adjustment to center the quantization range.

3.5.3 Algorithmic Steps

- **Data Analysis**: Initially, SimQuant analyzes the statistical distribution of the dataset to determine optimal quantization parameters.
- **Parameter Adjustment**: It then adjusts Δ and Z dynamically during the quantization process to minimize information loss.
- **Quantization Application**: Finally, the data is quantized using the calculated parameters, ensuring that the most critical information is retained with minimal resource usage.

3.5.4 Quantization Process

- Calculate range values:

$$\text{vals}_{\min} = \min(X_{\text{channel}}), \quad \text{vals}_{\max} = \max(X_{\text{channel}})$$

- Compute scale s and zero point z:

$$s = \frac{2^{\text{bits}} - 1}{\text{vals}_{\max} - \text{vals}_{\min}}, \quad z = -\text{vals}_{\min} \cdot s$$

- Apply quantization:

$$X_{\text{quant}} = \text{clamp}(\lfloor X \cdot s + z + 0.5 \rfloor, 0, 2^{\text{bits}} - 1)$$

3.5.5 Dequantization Process

$$X_{\text{dequant}} = \frac{X_{\text{quant}} - z}{s}$$

3.6 Applications

SimQuant is particularly effective in environments where storage and processing resources are at a premium, such as in embedded systems, mobile devices, and cloud-based machine learning platforms. Its ability to adaptively quantize data makes it suitable for real-time applications that require efficient data processing on-the-fly.

3.7 SmoothQuant: Smoothing Quantization Process

3.7.1 Definition and Concept

SmoothQuant [XLS+24] involves a sophisticated quantization approach that applies a smoothing algorithm to data before quantization. This method is particularly effective in applications where preserving the relational dynamics between different features of data is crucial.

3.7.2 Mathematical Formulation

The quantization and smoothing process of the 'SmoothQuantMatrix' can be mathematically described as follows:

$$\text{smoothed\_X} = X \cdot s, \quad \text{dequantized\_X} = \frac{\text{smoothed\_X}}{s}$$

where the scale factor s is calculated using the activity scales act_scales and the weight scales of the features:

$$s = \left(\frac{\text{act\_scales}^{\alpha}}{\text{weight\_scales}^{1-\alpha}} \right)$$

Here, α is a parameter that determines the balance between the activity and the weight scales, affecting the smoothness of the quantization.

3.7.3 Implementation Details

The 'SmoothQuantMatrix' class computes the smoothing scales based on the provided activity scales and the inherent data characteristics, dynamically adjusting each feature's scale. This customization allows the quantization process to be more adaptive and sensitive to the underlying data distribution.

3.7.4 Applications

SmoothQuant is particularly beneficial in neural network training and inference, where it can lead to more stable and robust models by mitigating the impact of quantization noise. It's also used in image processing and audio signal processing to maintain quality while reducing data bandwidth.

4 result

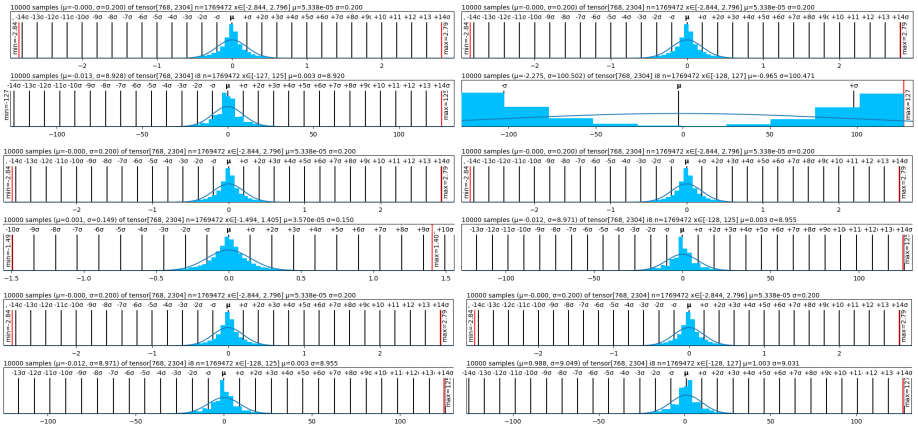

Figure 1: Quantized Weights Distribution

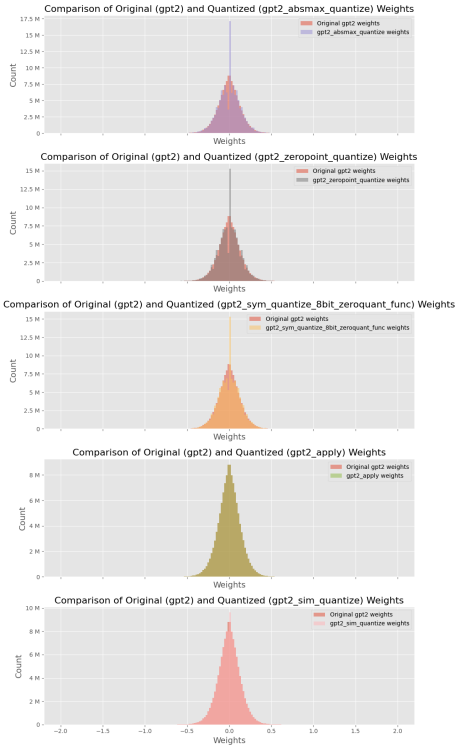

Figure 2: Performance Comparison after Quantization on GPT

Models	**Perplexity (ppl)**
GPT-2	4.01
GPT-2 INT8	6.83
GPT-2 AbsMax Quantize	9.32
GPT-2 ZeroPoint Quantize	8.93
GPT-2 Smooth Quant Apply	6.31
GPT-2 Sim Quantize	7.16
GPT-2 Sym Quantize 8bit	7.01
GPT-2 Sym Quantize 8bit ZeroQuant Func	7.37

Table 1: Perplexity Analysis of Quantization Models

5 Discussions

5.1 Simplification of Quantization Processes

LLMEasyQuant provides a user-friendly interface that simplifies the application of quantization techniques, making it accessible to both novices and experienced users without requiring deep technical knowledge of the underlying algorithms.

5.2 Customization and Flexibility

Despite its simplicity, the package offers extensive customization options that allow users to tailor the quantization process to their specific needs, balancing efficiency and performance according to the model's deployment context.

5.3 Efficiency in Deployment

Optimized for performance, LLMEasyQuant helps reduce the computational load and memory usage of models, facilitating their deployment on devices with limited resources such as mobile phones and embedded systems.

References

[Fac24] Hugging Face. Optimum-quanto. `https://github.com/huggingface/optimum-quanto`, 2024.

[FFBL18] Julian Faraone, Nicholas Fraser, Michaela Blott, and Philip H. W. Leong. Syq: Learning symmetric quantization for efficient deep neural networks, 2018.

[HKM+24] Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. Kvquant: Towards 10 million context length llm inference with kv cache quantization, 2024.

[NVI24] NVIDIA. Tensorrt-llm. `https://github.com/NVIDIA/TensorRT-LLM`, 2024.

[TOW+23] Chen Tang, Kai Ouyang, Zhi Wang, Yifei Zhu, Yaowei Wang, Wen Ji, and Wenwu Zhu. Mixed-precision neural network quantization via learned layer-wise importance, 2023.

[XLS+24] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models, 2024.

[YAZ+22] Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers, 2022.