Norm Tweaking: High-Performance Low-Bit Quantization of Large Language Models

Liang Li, Qingyuan Li, Bo Zhang, Xiangxiang Chu

Meituan
{liliang58,liqingyuan02,zhangbo97,chuxiangxiang}@meituan.com

Abstract

As the size of large language models (LLMs) continues to grow, model compression without sacrificing accuracy has become a crucial challenge for deployment. While some quantization methods, such as GPTQ, have made progress in achieving acceptable 4-bit weight-only quantization, attempts at lower-bit quantization often result in severe performance degradation. In this paper, we introduce a technique called norm tweaking, which can be used as a plugin in current PTQ methods to achieve high precision while being cost-efficient. Our approach is inspired by the observation that rectifying the quantized activation distribution to match its float counterpart can readily restore accuracy for LLMs. To achieve this, we carefully design a tweaking strategy that includes calibration data generation and channel-wise distance constraint to update the weights of normalization layers for better generalization. We conduct extensive experiments on various datasets using several open-sourced LLMs. Our method demonstrates significant improvements in both weight-only quantization and joint quantization of weights and activations, surpassing existing PTQ methods. On GLM-130B and OPT-66B, our method even achieves the same level of accuracy at 2-bit quantization as their float ones. Our simple and effective approach makes it more practical for real-world applications.

Introduction

Recently, OpenAI's ChatGPT (OpenAI 2023b) has demonstrated outstanding performance on text generation, sparking a research frenzy in large language models (LLMs). Some of the most famous LLMs include GPT series like GPT-3 (Brown et al. 2020), GPT-4 (OpenAI 2023a), and PaLM (Chowdhery et al. 2022), Ernie (Zhang et al. 2019). Open-sourced ones like GLM (Du et al. 2021), BLOOM (Laurençon et al. 2022), OPT (Zhang et al. 2022) and LLaMa series (Touvron et al. 2023) have remarkably accelerated the development of the community. In essence, LLMs are *generative models* that are trained on excessively large amounts of text data that mimics how humans use language, and they exhibit superior zero-shot performance in a large range of natural language processing (NLP) tasks, including language translation, sentiment analysis, text classification, and question answering, etc. They are increasingly being used in applications such as chatbots, language understanding, and speech recognition systems.

Copyright © 2024, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

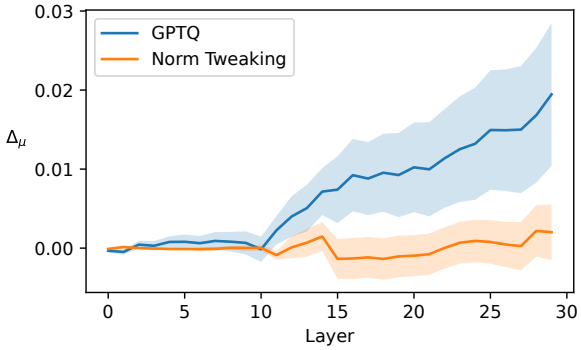

Figure 1: Activation distribution of *norm tweaking* is closer to its float counterpart compared with GPTQ. A batch size of 128 is used to compute the mean difference Δ_μ.

Nevertheless, due to the large scale (normally tens of billions or even trillions of parameters) of large language models, it causes large resource consumption even for deployment. Taking GPT-3 as an example, it has 175 billion parameters and uses FP16 for inference, occupying approximately 350 GB of GPU memory, which means at least 8 NVIDIA A100 GPUs are needed to support the deployment of a single model. Therefore, it is more than necessary to reduce the cost.

Model quantization, as a classic method of model compression, can effectively reduce the memory consumption of LLMs. For example, when using 4-bit quantization, GPT-3 can be deployed on 2 A100 GPUs due to one-fourth of memory reduction. GPTQ (Frantar et al. 2022) is currently the most prominent low-bit weight-only quantization method, which can compress some LLMs to 4-bit while maintaining acceptable precision degradation. Smoothquant (Xiao et al. 2023) could achieve 8-bit quantization for both weights and activations, by equivalently transferring the multiplication factors in weights and activations. However, these methods suffer from significant accuracy loss when applied to lower-bit quantization, such as 2-bit weight-only quantization using GPTQ or W4A8(4-bit for weights and 8-bit

for activation) quantization using SmoothQuant. According to ZeroQuant-V2 (Yao et al. 2023), LLaMa-65B with GPTQ 2-bit quantization, the accuracy on the LAMBADA dataset (Paperno et al. 2016) decreased from 79% to 57%, for which reason it proposes a quantization-aware training method based on low-rank compensation. However, it not only requires additional training costs but also introduces additional parameters, which is not a viable choice for efficient deployment.

To improve the lower-bit performance of quantized models, we first draw an intuition that LLMs have sufficient noise resilience, such that it calls a tender solution for precision recovery. It is demonstrated in Prompt Quantization (Xu et al. 2023) that for a compressed LLM, providing an appropriate prompt can yield high-precision generation without updating parameters. ZeroQuantV2 (Yao et al. 2023) indicates that the larger parameter a model has, the less degradation will the quantization have. Next, we explore why LLMs behave poorly on lower-bit quantization from a numerical perspective. We observe that the distribution of the quantized model's output tensor deviates significantly from that of the original float model, and it accumulates layer by layer to become uncontrollable, see Figure 1. Therefore a question is raised: *could we improve the performance of the quantized model by simply matching its activation distribution to that of the float model?*

To achieve this goal, we propose a method called Norm-Tweaking to enhance the quantized model by slightly adjusting the parameters of the LayerNorm layer to tweak the quantized distribution. This method can be widely applied to a variety of quantization methods, achieving significant accuracy improvement with only minimal additional computational cost. Our method is evaluated on various models and datasets, and the results indicate that Norm-Tweaking consistently improves the performance of GPTQ and SmoothQuant on different large language models. For LLaMa models, Norm-Tweaking demonstrates a general performance enhancement over GPTQ on diverse datasets, with a notable accuracy improvement of approximately 10% on the LAMBADA dataset. Moreover, during subjective evaluations of quantized models, we observe that Norm-Tweaking excels in preserving the general semantic ability of extremely low-bit quantized models. In a nutshell, our contribution is *three-fold*,

1. **Firstly**, we discover that large language models in general are *robust* against weight distortion, merely slight partial weight adjustment could recover its accuracy even in extreme low-bit regime. It is unnecessary to adopt heavy quantization-aware training or other sophisticated techniques.
2. **Secondly**, we carefully devise an LLM tweaking strategy composed of three parts **(1)** adjusting only the parameters of `LayerNorm` layers while freezing other weights, which can be applied to nearly all LLMs since it is pervasively used; **(2)** *constrained data generation* enlightened by LLM-QAT (Liu et al. 2023) to obtain the required calibration dataset, which effectively reduces the dependence on specific datasets during model quantization and fine-tuning process; **(3)** a *channel-wise tweaking loss* to specifically minimize the difference of the activation distribution of the quantized model to that of its float counterpart.
3. **Last but not least**, our technique is simple and effective with minimal resource consumption which can be used as a plugin in other PTQ methods. Extensive experiments demonstrate that our proposed norm-tweaking method achieves high-performance quantization for general LLMs, surpassing algorithms such as GPTQ.

Related Work

LLM Optimization. As most LLMs are based on Transformer (Vaswani et al. 2017), which is a typical memory-intensive architecture. The inference bottleneck lies more in the GPU's memory bandwidth, hence reducing its memory access can significantly improve the inference speed. FlashAttention (Dao et al. 2022), DeepSpeed (Aminabadi et al. 2022), and FlexGen (Sheng et al. 2023) propose optimized transformer implementations or efficient memory management to improve the throughput of LLMs. Others achieve this goal through model pruning, such as LoSparse (Li et al. 2023), SparseGPT (Frantar and Alistarh 2023), and LLM-Pruner (Ma, Fang, and Wang 2023). MiniMoE (Zhang et al. 2023) obtains smaller models with high performance through distillation.

Post-training Quantization. *Weight-only* quantization schemes like GPTQ (Frantar et al. 2022) compresses and stores weight parameters, and decompresses them to FP16 for inference during calculation. This approach can effectively reduce the proportion of memory access time during inference while maintaining model accuracy. LLM.int8() (Dettmers et al. 2022) proposes to use float calculation or to adjust the multiplication operations of `LayerNorm` to reduce quantization loss. Smoothquant (Xiao et al. 2023) proposes a method to reduce the activation ranges by equivalently transferring the multiplication factors in weights and activations. GPTQ (Frantar et al. 2022) reconstruct weights based on the method in OBS (Hassibi, Stork, and Wolff 1993) via Hessian matrix to reduce quantization error. GPTQ has been widely applied in many scenarios where some LLMs could achieve high precision at 4-bit quantization. RPTQ (Yuan et al. 2023) and AWQ (Lin et al. 2023) further improve this method.

Quantization-aware Training. Another method to improve the performance of the quantized models is quantization-aware training (QAT), which is to fine-tune the quantized models to match the original float models. QAT is widely studied in convolutional networks, but it encounters significant setbacks in large language model quantization. As the training process of LLMs consumes a huge amount of text data (usually in the order of trillions of tokens), how to efficiently fine-tune the quantized LLMs while maintaining their general knowledge and generalization ability remains an open question. To name a few attempts, LLM-QAT (Liu et al. 2023) requires the update the whole parameters of the LLMs on a set of at least 100k sampled data. ZeroQuantV2 (Yao et al. 2023) introduces a Low Rank Com-

pensation to achieve parameter-efficient fine-tuning, but this approach neither eliminates the need for a large amount of training data nor avoids the introduction of additional parameters.

Method

Motivation

Based on the observation shown in Figure 1, the difference between the output tensors of each layer in the quantized model and its floating counterpart accumulates, while the output of the quantized model gradually deviates from the quantization-friendly zero-mean distribution. This is somewhat expected since `LayerNorm` magnifies the outlier (Xiao et al. 2023) and no measure is taken to deal with this effect. Hence when we iteratively update the quantized weights of each layer using GPTQ, it inevitably disrupts the zero-mean distribution of the current layer and increases the deviation.

To this end, we aim to improve the quantized model's performance by adjusting its output distribution to approach that of its float counterpart. Complete fine-tuning of the quantized model through QAT is a direct approach, but due to the large number of parameters in the LLM model and the enormous amount of required training data, QAT becomes an impractical option. In order to achieve high performance the quantized model within the time constraint, we are driven to improve current PTQ methods. As `LayerNorm` is highly effective in managing the distribution, we choose to adjust this layer to achieve the goal. It is also economical to update its weight considering the small number of parameters. Furthermore, nearly all mainstream LLMs use `LayerNorm` or similar operators, so that the method can be applied universally to a variety of large language models. Therefore, our core objective can be summarized as adjusting the parameters of `LayerNorm` to make the output distribution of the quantized model approach that of the float model, which can be expressed formally as,

$$arg\min_{W_{ln}} L_{dist}(T(X), \hat{T}(X)) \tag{1}$$

where $T(X|W_{attn}, W_{mlp}, W_{ln})$ denotes a Transformer block, including the Attention module, MLP module, LayerNorm layer, and activation functions, and $\hat{T}(X)$ represents its quantized version. $L_{dist}(\cdot)$ denotes the distribution loss function between the quantized and float models. Our goal is then to design a strategy to optimize $\hat{W}_{ln}$ to minimize $L_{dist}(\cdot)$, while keeping $\hat{W}_{attn}$ and $\hat{W}_{mlp}$ frozen.

Norm Tweaking

Motivated by the above analysis, we propose a PTQ method for LLMs, called Norm-Tweaking, to quickly restore models' performance by slightly tweaking LayerNorm layers of the quantized model. Norm tweaking serves as a plugin to be easily embedded into other quantization methods. Here, we take GPTQ as an example and present a *weight-only* post-quantization algorithm pipeline, as shown in Algorithm 1. **Firstly**, we use the LLM model to generate a set of text data as for calibration (explained in detail in the section on Calibration Dataset Generation), instead of directly sampling from real datasets. **Next**, we iteratively process each transformer layer, quantizing and updating the weights of the Linear layers, just like GPTQ. **Finally**, we compute a channel-wise loss based on the difference between the distribution of quantized output and float output. We then use stochastic gradient descent to update the parameters of `LayerNorm` in this layer, forcing the activation distribution of the quantized model to mimic that of the float model. During this process, the rest parameters of the current layer such as `Linear` are frozen and do not participate in the weight update.

Algorithm 1: Norm-Tweaking

Input: Pre-trained LLM model
Output: Quantized LLM model

Generate calibration dataset ($n\_samples$ = 128, $token\_length$ = 2048) using pre-trained LLM model
for each layer-l in the Transformer structure (L layers total) **do**
  if l = 0 **then**
    use calibration data as input
  else
    use last output $qOut_{l-1}$ as input
  end if
  Calculate the float output $fOut_l$
  Quantize the weights of layer l
  Freeze all Linear's weights in layer l
  for each it for total $Iters$ **do**
    Calculate the float output $qOut_l$
    Calculate L_{dist} between $fOut_l$ and $qOut_l$
    Backward and update LayerNorms' parameters
  end for
end for
Get the high-performance quantized LLMs

Although only the parameters of `LayerNorm` are updated, our process is distinct from parameter-efficient fine-tuning strategies. It should be noted that the parameters of the `LayerNorm` layer are very sensitive and excessive tuning can seriously damage the quantized models' performance (see Table 6). We slightly update the LayerNorm with a relaxed constraint, whose goal is to make the quantized models' distribution approaching that of float ones. This is precisely why we definite our method as a *tweaking*, rather than a fine-tuning.

At a glimpse, we carefully design the entire tweaking procedure to achieve our goal. For example, we use a very small number of iterations during tuning, typically only one iteration on the calibration text is required. We also adopt a small learning rate and design a step scheduler to assign different learning rates for the subsequent layers. In addition, our calibration data generation and the design of the distribution loss function harmoniously resonate with our tweaking principle.

Calibration Data Generation

A crucial problem that matters in the generalization ability of the quantized model is the appropriate choice of calibra-

tion data. We found that different calibration datasets substantially affect the performance of the quantized model. It usually performs well on the calibration dataset, but it generally suppresses the performance on other datasets. LLM-QAT (Liu et al. 2023) demonstrated that training the quantized model with a specific dataset further damages LLMs' generalization ability. Therefore, we adopt a data generation scheme following LLM-QAT that utilizes the generated data of the model itself for calibration instead of a specific real dataset. The benefit is that thus-generated data can effectively activate the neurons of the LLM which facilitates model quantization. It also enjoys rich semantic information stored in the model and it is less biased towards a specific dataset which is more generalizable.

Our generation process is a variant of that of LLM-QAT. Firstly, a random token is taken from a list of given languages and then a two-stage pattern proposed by LLM-QAT is employed where the picked token is fed as the input prompt to let LLMs generate subsequent tokens. We enhance this data generation process by enforcing a restriction on the first random token. We observe a significant disparity in terms of proportions between the language categories in the *training corpus* and *tokenization vocabulary*. As shown in Table 1, taking BLOOM as an example, it is trained on a total of 1.61 TB of text, with the top five language types accounting for over 75% of the corpus. If we consider the related corpus (e.g. `zht` as a traditional version of `zhs`) and derivative ones (e.g. programming languages) of these five language types, the proportion exceeds 90%. In contrast, there are 250680 tokens in the tokenization vocabulary, whose total number of tokens corresponding to these five languages only accounts for 17%. Therefore, the first token of input directly affects the language type of the generated text. If we randomly select from the entire vocabulary, we cannot get appropriate calibration data that matches the training corpus. To this very purpose, we restrict the first random token to be selected only from the language categories in the list of top languages that have the highest proportion, which turns out to effectively improve the generalization of the quantized model on different datasets (Table 8).

Language	en	zhs	fr	es	pt
Corpus(MB)	485.0	261.0	208.2	175.1	79.3
Vocab	7943	380	15483	6999	8669

Table 1: Text size and token count for the top 5 languages.

Channel-wise Distribution Loss

To guide the direction of parameter updates, it is crucial to design a corresponding loss function. In this context, we aim to minimize the difference between the activation distribution of the quantized model and its original float model. **Firstly**, as the activation distribution of LLMs exhibits significant differences along the channel dimension, with some channels displaying extreme values (referred to as *outliers*) (Xiao et al. 2023), it poses great challenges for the quantization process. In order to preserve the differences between channels while tweaking the model parameters and to retain the original model capacity as much as possible, we enforce a channel-wise constraint. **Secondly**, a strict alignment of the *point-wise* activation values between quantized and float models may result in overfitting to calibration data, thereby compromising the generalization performance across different datasets. Therefore, we adopt a more relaxed alignment strategy by directly aligning the mean and variance between each channel, instead of strictly aligning the targets at the point-wise level. As a result, we introduce a *channel-wise distribution loss* function, as shown below:

$$L_{dist} = \frac{1}{C}\sum_{c=1}^{C}(\left\|\mu_f^c - \mu_q^c\right\|_2 + \left\|(\sigma_f^c)^2 - (\sigma_q^c)^2\right\|_2) \quad (2)$$

where C is the number of channels, μ and σ represent the mean and variance of each channel in tensor T, the subscript $_f$ and $_q$ indicates the float and quantized model.

Furthermore, current algorithms like GPTQ iteratively quantize LLMs layer by layer, whose deviation of intermediate activation distributions gradually accumulates, resulting in large errors in the final layers. Thus, we apply a layer-level scheduler to adjust the learning rate of each layer during the tweaking process where we simply adopt a step increase to allocate different learning rates on different layers.

$$lr_i = lr_0 * (1 + scale * (i/L)) \quad (3)$$

Experiments

Settings

We tested our method on LLMs of different sizes and types, including GLM (Du et al. 2021), BLOOM (Laurençon et al. 2022), OPT (Zhang et al. 2022) and LLaMa series (Touvron et al. 2023). Our Norm-Tweaking results presented in the paper, unless otherwise noted, are obtained using weight-only quantization based on the GPTQ algorithm. Considering the kernel support for deployment frameworks, such as FasterTransformer (NVIDIA 2023), we use symmetric per-channel quantization. In the tweaking process, we choose the Adam optimizer (Kingma and Ba 2015) to update the LayerNorm parameters of LLMs or the RMSNorm (Zhang and Sennrich 2019) parameters of LLaMA. The learning rate needs to be carefully set. A large learning rate would damage the final results. In our experiments, we typically use a grid search to obtain the optimal learning rate, with an initial value set at 1e-5.

Our primary experimental evaluations are performed on the LAMBADA dataset (Paperno et al. 2016), which is renowned for its high demand for the understanding ability of natural language. This dataset necessitates a comprehensive understanding of the entire text to provide precise answers. To further substantiate the generalization of our method on different datasets, we employed Benchmark Harness (Gao et al. 2021) to conduct tests on a broader spectrum of datasets, encompassing HellaSwag (Zellers et al. 2019), PIQA (Bisk et al. 2020), WinoGrande (Sakaguchi et al. 2021), OpenBookQA (Mihaylov et al. 2018), and some

Model	FP16	W4		W2	
		GPTQ	Norm-Tweaking	GPTQ	Norm-Tweaking
BLOOM-7b1 (Laurençon et al. 2022)	57.6751	55.0615	**57.4811** (2.4196↑)	33.4714	**37.4539** (3.9825↑)
BLOOM-176b	67.7081	67.1842	**67.6887** (0.5045↑)	63.0507	**65.6317** (2.581↑)
LLaMa-7b (Touvron et al. 2023)	73.5106	71.8999	**72.4820** (0.5387↑)	11.8766	**21.3856** (9.509↑)
LLaMa-65b	79.0996	78.0516	**79.2354** (1.1838↑)	57.1512	**67.4753** (10.3241↑)
GLM-130b (Du et al. 2021)	69.4159	**69.2218**	69.1964 (0.0254↓)	67.6499	**69.4293** (1.7794↑)
OPT-66b (Zhang et al. 2022)	73.2971	73.0060	**73.8405** (0.8345↑)	71.3953	**73.4912** (2.0959↑)

Table 2: The quantized accuracy results of LLMs on the LAMBADA dataset. W4/2: 4/2-bit weights-only quantization.

datasets from the General Language Understanding Evaluation (GLUE) benchmark. We also use WikiText-2 (Merity et al. 2016), PTB (Marcus et al. 1994), C4 (Raffel et al. 2020) in Table 5, to provide some demonstrations of text generated by quantized LLMs, which helps to more intuitively visualize the performance recovery of Norm-Tweaking. Following the settings in GPTQ, we used a calibration dataset size with $n\_samples$=128, with the maximum sequence length $token\_length$=2048.

Tweaking Cost

We demonstrate that Norm-Tweaking incurs extremely low costs. Taking BLOOM (Laurençon et al. 2022) as an example, given the hidden dimension as h, each transformer block generally has 4 Linear layers, with a total parameter count of about $12h^2 + 9h$, while `LayerNorm` has two layers, with a parameter count of $4h$. The hidden dimension h is typically very large (for example, 14336 for BLOOM-176B), so the parameter quantity of the `Linear` layer is much larger than that of the `LayerNorm` layer (on the order of $10^7 \sim 10^9$). In addition, to avoid overfitting on specific calibration data, we only perform one iteration on each sample of text. Therefore, the proposed Norm-Tweaking method has minimal resource consumption and extra time.

Model	BLOOM-7B	LLaMA-7B	OPT-13B
GPTQ	19.6	15.5	27
GPTQ+NT	22.8	27.3	46.6

Table 3: Quantization runtime measured in minutes for GPTQ and Norm-Tweaking on various LLMs.

Table 3 shows the time cost taken to quantize LLMs using GPTQ and Norm-Tweaking. All experiments were conducted on a single NVIDIA A100 GPU. The additional time cost of Norm-Tweaking is less than the time cost of GPTQ itself, and our method still remains within the category of post-quantization. For BLOOM-7B, the time cost increase accounts for only 16%.

Results on LAMBADA

As shown in Table 2, our proposed model quantization method is applied to LLMs at different scales, including BLOOM, LLaMa, GLM, and OPT, where the accuracy of each quantized model is evaluated on the LAMBADA dataset and is compared comprehensively with GPTQ. In addition, we also conduct experiments on 2-bit weight-only quantization with a fine-grained quantization with a group of 64. Our Norm-Tweaking post-quantization method generally outperforms the GPTQ algorithm in terms of model accuracy. In 2-bit quantization, the GPTQ algorithm caused significant accuracy loss for most models, making a large part of results are not acceptable. However, our proposed quantization method is able to achieve accuracy performance close to the floating-point model even on the GLM-130B and OPT-66B models, and it outperforms GPTQ by nearly 10% on LLaMa.

Method	Mode	BLOOM-7B	OPT-13B
w/o PTQ	FP16	57.6751	69.0860
RTN	W4A16	48.3602	62.7402
RTN+NT	W4A16	**51.5622**	**64.7584**
SmoothQuant	W4A8	53.9492	68.6590
SmoothQuant+NT	W4A8	**54.5896**	**69.7264**

Table 4: Norm-Tweaking (NT) on various LLM PTQ methods. Note for OPT-13B, W5A8 is used for SmoothQuant and SmoothQuant+NT.

Comparison with RTN and SmoothQuant

We integrate Norm-Tweaking into two commonly used post-quantization methods, *round-to-nearest* (RTN) (Yao et al. 2022; Dettmers et al. 2022) and SmoothQuant (Xiao et al. 2023), to verify its general effectiveness across different algorithms. Several LLMs are quantized in different modes and evaluated on the LAMBADA dataset, results are shown in Table 4. Specifically, we apply 4-bit weight-only quantization to RTN, and W4A8 (4-bit for weight and 8-bit for activation) quantization to the SmoothQuant. Note OPT-13b is severely compromised when using SmoothQuant W4A8 quantization, resulting in an accuracy of 0. The results demonstrate the universality of Norm-Tweaking, as it provides stable performance improvements for different quantization methods, including RTN, GPTQ, and SmoothQuant, as well as for different quantization modes, including weight-only and both weight and activation.

Benchmark Harness

We benchmark the 2-bit quantized LLaMa-7b and LLaMa-65b on the few-shot evaluation framework LM Evaluation Harness (Gao et al. 2021) in Table 7. In short of space and time, we report the result on a selected list of tasks. Our proposed method generally outperforms GPTQ 2-bit results,

Model	BLOOM -176B
FP16	Beijing is the capital of China. Beijing is also one of the world's oldest cities. It is also one of the most modern and beautiful cities of China. As of 2016, it had a population of over 20 million. Beijing is considered the most beautiful city in China. It is known for its many beautiful and ancient buildings and sites. These include the Forbidden City, the Temple of Heaven, and the Great Wall of China.
GPTQ (4-bit)	Beijing is the capital of China. There are 13.7 million people in the city of Beijing. The city is divided into 16 districts, and the most important districts are Dongcheng, *Xicheng,Haidian*, *Chaoyang,Fengtai*, Shijingshan and Daxing. The city is located in the *north-east* of China. *The city is located in the north-east of China, close to the Pacific coast. The city is located on the left bank of the Beijing River.*
Norm-Tweaking (4-bit)	Beijing is the capital of China, and the city is one of the most popular tourist destinations in the world. You can find here a lot of interesting places. You can visit the Forbidden City. It is located in the center of the city and is considered the most significant attraction. Also you can visit the Temple of Heaven. This place is located a few kilometers from the center of the city. You can admire the beauty of the Beijing city from the observation deck of the World Trade Center.
Model	LLaMa-65B
FP16	Beijing is the capital of China, and also one of the largest cities in the world. It is a modern city that has successfully managed to retain its ancient Chinese roots and the essence of its culture. A city of over 21 million people, Beijing is a major hub for international business, and as such attracts a large number of travelers.
GPTQ (2-bit)	Beijing is the capital of China, and has a rich history *datin* back to 5th in 1910s. Peking was the old capital in *1910s* and renamed as Beijing in *1913*, and became capital in *1972*. Beijing is an interesting city, with *the Forbidden City in the Forbidden City*, which is a world heritage site.
Norm-Tweaking (2-bit)	Beijing is the capital of China. The country has a population of around 1.3 billion Chinese people. The country is one of the leading exporters in the world, and also one of the leading importers of the world. China is one of the leading manufacturers of the world. China is a large country, and is one of the largest countries in the world.

Table 5: Example of 4-bit quantized BLOOM-176B and 2-bit quantized LLaMa-65B text generation on the specified prompt "Beijing is the capital of China". The text in italic is either grammatically wrong or counterfactual.

Iters	1	2	5
Acc	**57.4811**	55.7539	52.1056
Iters	10	20	50
Acc	46.8465	32.3307	11.3332

Table 6: Effect of tweaking iterations.

with some even better than FP16 accuracy. This again proves the robustness of our method and strong generalizability to a wide range of datasets. We discuss the performance variations among datasets in the appendix.

Subjective Evaluation

Subjective evaluation of the generated results is a common and effective method for evaluating the performance of language models such as LLM. In Table 5, the FP16 mode of LLaMa-65B and BLOOM-176B, as well as quantized model with GPTQ and Norm-Tweaking are evaluated through the lens of human evaluation on generated results. With the same input prompt, it can be seen that different models give significantly different results, especially the GPTQ low-bit quantization model, which is subject to obvious errors. These errors mainly manifest either grammatical errors (e.g. misspelled words or incorrect use of punctuation or spaces), logical errors in the language (e.g. repeated statements), and factual errors (e.g. birth date). Nevertheless, adopting the quantization method proposed in this paper, the quantized model obtained under the same settings does not have these obvious errors in the output results, suggesting the robustness of our quantization method.

Ablation

Tweaking Iterations. We investigate the effect of the number of iterations for Norm-Tweaking and report the results of BLOOM-7B tested on LAMBADA dataset in Table 6. It turns out that increasing the iteration numbers during the tweaking process significantly damages the model's accuracy performance. This is as expected since the parameters of `LayerNorm` are highly sensitive, where excessive iterations can easily lead to the collapse of model performance. This is also why we recommend tweaking instead of tuning, which also clearly distinguishes us from those QAT methods such as LLM-QAT.

Model (Precision)	HellaSwag	PIQA	WinoGrande	OpenBookQA	RTE	MRPC	QNLI	BOOLQ	CB	COPA	WIC
LLaMa-7b (FP16)	56.44	78.35	67.09	28.00	53.07	68.38	49.57	73.15	33.93	84.00	50.00
w/ GPTQ (2-bit)	30.73	58.49	48.54	13.20	**53.43**	49.75	**51.53**	52.02	37.50	68.00	49.53
w/ Norm-Tweak (2-bit)	**34.03**	**61.81**	**52.17**	**15.80**	51.26	**54.66**	50.61	**56.91**	**48.21**	**68.00**	**51.41**
LLaMa-65b (FP16)	63.97	81.66	77.19	36.40	71.48	68.38	54.00	82.32	64.29	91.00	58.46
w/ GPTQ (2-bit)	45.99	72.20	60.77	23.20	60.65	64.95	**52.35**	66.33	**39.29**	82.00	49.84
w/ Norm-Tweak (2-bit)	**52.15**	**74.04**	**67.24**	**26.80**	**61.37**	**68.38**	49.60	**76.15**	30.36	**93.0**	**50.00**
BLOOM-176b (FP16)	55.91	78.78	70.32	32.20	62.09	34.80	51.38	69.85	71.43	87.00	48.43
w/ GPTQ (2-bit)	50.04	75.73	68.67	27.40	57.40	**54.66**	49.86	66.64	46.43	81.00	**50.00**
w/ Norm-Tweak (2-bit)	**54.64**	**78.51**	**71.51**	**32.00**	**58.84**	35.29	**51.42**	**71.74**	**48.21**	**87.00**	48.90
OPT-66b (FP16)	56.45	78.62	68.82	30.40	59.93	34.07	52.24	69.72	39.29	86.00	50.00
w/ GPTQ (2-bit)	49.72	75.35	**65.90**	25.80	54.51	45.34	**53.08**	64.68	41.07	**86.00**	50.47
w/ Norm-Tweak (2-bit)	**49.81**	**75.41**	64.25	**26.80**	**54.51**	**68.38**	49.53	**69.88**	**41.07**	85.00	**50.47**

Table 7: The quantized accuracy results of LLMs on the LM Evaluation Harness benchmark.

Calibration Data	WikiText2	PTB	C4
WikiText2	**12.16**	21.17	18.28
PTB	12.51	**20.72**	18.42
C4	12.28	20.97	**18.16**
Random	13.25	22.82	19.60
GenData V1	12.43	21.25	18.34
GenData V2	**12.32**	**20.95**	**18.28**

Table 8: Effects of different calibration datasets. V1 follows LLM-QAT, and V2 is our improved version.

Model	L_{MSE}	L_{KL}	L_{Dist}
BLOOM-7b	55.8704	56.2779	**57.4811**
LLaMa-7b	72.3850	71.7446	**72.4820**
OPT-13b	68.3291	68.2709	**68.7173**

Table 9: Comparison of different loss functions.

Calibration Data. Table 8 shows how the choice of calibration dataset significantly affects the performance of quantized models on different datasets. We use three real datasets WikiText2 (Merity et al. 2016), PTB (Marcus et al. 1994), and C4 (Raffel et al. 2020), as well as random data and generated data, as calibration sets to quantize the BLOOM-7B model using GPTQ. And we give the perplexity (PPL) on WikiText2, PTB, and C4 respectively, with lower PPL indicating better performance. The first three rows show the strong correlation between GPTQ and the calibration dataset, that is, a LLM calibrated on a certain dataset performs better on that dataset, but correspondingly worse on other datasets.

To avoid the dependence on real data, we randomly sample data from Gaussian distribution with the same mean and variance of the real data for calibration. However, the performance of the quantized model was extremely poor. We guess that this is because random data is without actual semantic meaning, which cannot produce positive activations for LLMs when being used as a calibration dataset. We exploit the LLM itself to generate calibration data. It can produce meaningful text and effectively activate the model. The results show that using generated data for calibration can improve the performance of the quantized model, and it does not show dependence on specific data. Using the language scope restriction proposed in this paper can further improve the quality of generated data.

Loss Function. To showcase the importance of our proposed channel-wise distribution loss L_{Dist}, we compare it with several different loss functions like mean square error L_{MSE} and Kullback-Leibler Divergence loss L_{KD} (Hinton, Vinyals, and Dean 2015), the result is shown in Table 9 where the proposed L_{Dist} works best in all cases. This result echos our analysis that channel-wise treatment is necessary (better than L_{KL}) to deal with outliers while pointwise alignment (L_{MSE}) harms the performance. As a collaborative result of multiple components in Norm-Tweaking, the difference of quantized activation distribution to its float counterpart is largely narrowed, as shown in Figure 1. This observation fairly answers our original question that minimizing the activation distribution of LLMs between two precisions readily renders high performance, even for extremely low-bit quantization.

Conclusion

In conclusion, we have proposed a novel quantization compression method for large-scale language models (LLM) that surpasses existing state-of-the-art methods such as GPTQ and SmoothQuant. Our method is characterized by generating generalizable calibration data and tweaking the normalization layer with channel-wise distribution loss, enabling us to quickly achieve high-precision model quantization in a low-cost manner. Notably, we have explored LLM model compression at the 2-bit range, marking state-of-the-art performance. Our approach delivers a promising solution for reducing the computational and storage costs associated with LLMs while maintaining their high performance.

Acknowledgements

This work was supported by National Key R&D Program of China (No. 2022ZD0118700).

References

Aminabadi, R. Y.; Rajbhandari, S.; Zhang, M.; Awan, A. A.; Li, C.; Li, D.; Zheng, E.; Rasley, J.; Smith, S.; Ruwase, O.; and He, Y. 2022. DeepSpeed Inference: Enabling Efficient Inference of Transformer Models at Unprecedented Scale. arXiv:2207.00032.

Bisk, Y.; Zellers, R.; Gao, J.; Choi, Y.; et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 7432–7439.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. In *Conference on Neural Information Processing Systems (NeurIPS)*.

Chowdhery, A.; Narang, S.; Devlin, J.; Bosma, M.; Mishra, G.; Roberts, A.; Barham, P.; Chung, H. W.; Sutton, C.; Gehrmann, S.; et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Dao, T.; Fu, D. Y.; Ermon, S.; Rudra, A.; and Ré, C. 2022. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. *arXiv preprint arXiv:2205.14135*.

Dettmers, T.; Lewis, M.; Belkada, Y.; and Zettlemoyer, L. 2022. LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale. *arXiv preprint arXiv:2208.07339*.

Du, Z.; Qian, Y.; Liu, X.; Ding, M.; Qiu, J.; Yang, Z.; and Tang, J. 2021. Glm: General language model pretraining with autoregressive blank infilling. *arXiv preprint arXiv:2103.10360*.

Frantar, E.; and Alistarh, D. 2023. SparseGPT: Massive Language Models Can Be Accurately Pruned in One-Shot. arXiv:2301.00774.

Frantar, E.; Ashkboos, S.; Hoefler, T.; and Alistarh, D. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*.

Gao, L.; Tow, J.; Biderman, S.; Black, S.; DiPofi, A.; Foster, C.; Golding, L.; Hsu, J.; McDonell, K.; Muennighoff, N.; Phang, J.; Reynolds, L.; Tang, E.; Thite, A.; Wang, B.; Wang, K.; and Zou, A. 2021. A framework for few-shot language model evaluation.

Hassibi, B.; Stork, D. G.; and Wolff, G. J. 1993. Optimal brain surgeon and general network pruning. In *IEEE International Conference on Neural Networks*.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the Knowledge in a Neural Network. In *NIPS Deep Learning and Representation Learning Workshop*.

Kingma, D. P.; and Ba, J. 2015. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*.

Laurençon, H.; Saulnier, L.; Wang, T.; Akiki, C.; del Moral, A. V.; Le Scao, T.; Von Werra, L.; Mou, C.; Ponferrada, E. G.; Nguyen, H.; et al. 2022. The BigScience Corpus: A 1.6 TB Composite Multilingual Dataset.

Li, Y.; Yu, Y.; Zhang, Q.; Liang, C.; He, P.; Chen, W.; and Zhao, T. 2023. LoSparse: Structured Compression of Large Language Models based on Low-Rank and Sparse Approximation. In Krause, A.; Brunskill, E.; Cho, K.; Engelhardt, B.; Sabato, S.; and Scarlett, J., eds., *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, 20336–20350. PMLR.

Lin, J.; Tang, J.; Tang, H.; Yang, S.; Dang, X.; and Han, S. 2023. AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration. arXiv:2306.00978.

Liu, Z.; Oguz, B.; Zhao, C.; Chang, E.; Stock, P.; Mehdad, Y.; Shi, Y.; Krishnamoorthi, R.; and Chandra, V. 2023. LLM-QAT: Data-Free Quantization Aware Training for Large Language Models. *arXiv preprint arXiv:2305.17888*.

Ma, X.; Fang, G.; and Wang, X. 2023. LLM-Pruner: On the Structural Pruning of Large Language Models. arXiv:2305.11627.

Marcus, M.; Kim, G.; Marcinkiewicz, M. A.; MacIntyre, R.; Bies, A.; Ferguson, M.; Katz, K.; and Schasberger, B. 1994. The Penn treebank: Annotating predicate argument structure. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.

Merity, S.; Xiong, C.; Bradbury, J.; and Socher, R. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Mihaylov, T.; Clark, P.; Khot, T.; and Sabharwal, A. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*.

NVIDIA. 2023. FasterTransformer.

OpenAI. 2023a. GPT-4 Technical Report. arXiv:2303.08774.

OpenAI. 2023b. Introducing ChatGPT.

Paperno, D.; Kruszewski, G.; Lazaridou, A.; Pham, Q. N.; Bernardi, R.; Pezzelle, S.; Baroni, M.; Boleda, G.; and Fernández, R. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. *arXiv preprint arXiv:1606.06031*.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(140): 1–67.

Sakaguchi, K.; Bras, R. L.; Bhagavatula, C.; and Choi, Y. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9): 99–106.

Sheng, Y.; Zheng, L.; Yuan, B.; Li, Z.; Ryabinin, M.; Fu, D. Y.; Xie, Z.; Chen, B.; Barrett, C.; Gonzalez, J. E.; Liang, P.; Ré, C.; Stoica, I.; and Zhang, C. 2023. FlexGen: High-Throughput Generative Inference of Large Language Models with a Single GPU. arXiv:2303.06865.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Conference on Neural Information Processing Systems (NeurIPS)*.

Xiao, G.; Lin, J.; Seznec, M.; Wu, H.; Demouth, J.; and Han, S. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, 38087–38099. PMLR.

Xu, Z.; Liu, Z.; Chen, B.; Tang, Y.; Wang, J.; Zhou, K.; Hu, X.; and Shrivastava, A. 2023. Compress, Then Prompt: Improving Accuracy-Efficiency Trade-off of LLM Inference with Transferable Prompt. *arXiv preprint arXiv:2305.11186*.

Yao, Z.; Aminabadi, R. Y.; Zhang, M.; Wu, X.; Li, C.; and He, Y. 2022. ZeroQuant: Efficient and Affordable Post-Training Quantization for Large-Scale Transformers. *arXiv preprint arXiv:2206.01861*.

Yao, Z.; Wu, X.; Li, C.; Youn, S.; and He, Y. 2023. ZeroQuant-V2: Exploring Post-training Quantization in LLMs from Comprehensive Study to Low Rank Compensation. arXiv:2303.08302.

Yuan, Z.; Niu, L.; Liu, J.; Liu, W.; Wang, X.; Shang, Y.; Sun, G.; Wu, Q.; Wu, J.; and Wu, B. 2023. RPTQ: Reorder-based Post-training Quantization for Large Language Models. arXiv:2304.01089.

Zellers, R.; Holtzman, A.; Bisk, Y.; Farhadi, A.; and Choi, Y. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Zhang, B.; and Sennrich, R. 2019. Root Mean Square Layer Normalization. arXiv:1910.07467.

Zhang, C.; Yang, Y.; Liu, J.; Wang, J.; Xian, Y.; Wang, B.; and Song, D. 2023. Lifting the Curse of Capacity Gap in Distilling Language Models. arXiv:2305.12129.

Zhang, S.; Roller, S.; Goyal, N.; Artetxe, M.; Chen, M.; Chen, S.; Dewan, C.; Diab, M.; Li, X.; Lin, X. V.; et al. 2022. OPT: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Zhang, Z.; Han, X.; Liu, Z.; Jiang, X.; Sun, M.; and Liu, Q. 2019. ERNIE: Enhanced language representation with informative entities. *arXiv preprint arXiv:1905.07129*.