Compressing Large Language Models by Joint Sparsification and Quantization

Jinyang Guo [1 2] **Jianyu Wu** [2] **Zining Wang** [1] **Jiaheng Liu** [1] **Ge Yang** [2] **Yifu Ding** [1] **Ruihao Gong** [3]
Haotong Qin [4] **Xianglong Liu** [1 ✉]

Abstract

In this paper, we introduce a novel model compression technique named Joint Sparsification and Quantization (JSQ), explicitly tailored for large language models (LLMs). Traditional methods employ either sparsification or quantization individually to compress LLMs, leading to performance degradation at high compression ratios. In contrast, our JSQ approach integrates sparsification and quantization cohesively. As sparsification tend to preserve outliers that is harmful to quantization, we introduce a novel sparsity metric to serves as a bridge between the sparsification and quantization. Moreover, it is proven outliers in LLMs have significant impact but harmful to compression. Current solutions are highly coupled with quantization process, which is not helpful to sparsification. To this end, we also introduce a search-based activation editor to automatically eliminate relatively useless outliers. Comprehensive experiments across various datasets and architectures affirm the efficacy of our JSQ framework. Notably, our JSQ achieves 7.96× computation reduction without crashing for the representative model LLaMA. This accomplishment stands in stark contrast to the limitations of most state-of-the-art LLM compression methods, which typically fail under such extreme compression ratios. Our code is released at https://github.com/uanu2002/JSQ.

1. Introduction

Although large language model (LLM) like LLaMA (Touvron et al., 2023a) has achieved increasing attention because of its strong intelligence, it brings a huge computation and storage burden in real-world applications. To address this issue, many model compression techniques are proposed to compress LLMs, including sparsification, quantization, etc.

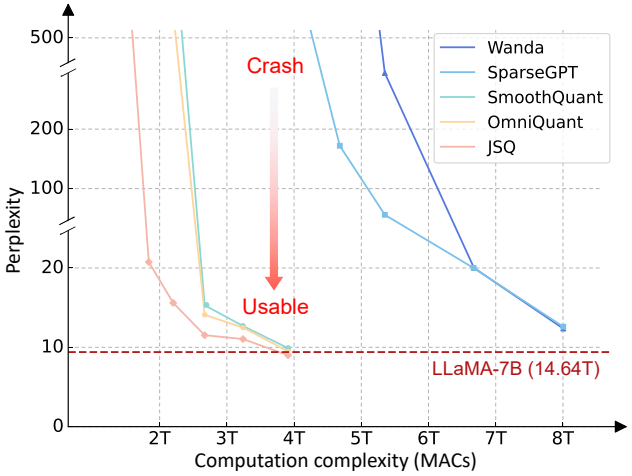

Figure 1. Performance of compressed LLM under different computation complexities.

As LLM is much larger than previous state-of-the-art models, it requires larger compression ratio for practical usage. However, current model compression methods specifically designed for LLM only consider one specific technique to compress the model, suffering from significant performance degradation when compression ratio becomes large. For example, in Fig. 1, the state-of-the-art approach OmniQuant (Shao et al., 2023) crashes under low computation complexities. So a new problem emerges: *Is it possible to maintain LLM performance under high compression ratio?*

To answer this question, in this paper, we propose the Joint Sparsification and Quantization (JSQ) framework to simultaneously consider the sparsification and quantization, which is the first work to jointly utilize these two methods for compressing LLM under high compression ratio.

Simultaneously utilizing these two techniques is non-trivial due to two significant issues. Firstly, sparsification and quantization are in contradiction with each other. Sparsification tends to preserve parameters with large absolute values in LLM (Han et al., 2015), while quantization prefers a small range of parameter values (Wei et al., 2023). Consequently, the preserved parameters in sparsification may degrade the

[1]State Key Laboratory of Complex & Critical Software Environment, Beihang University [2]Institute of Artificial Intelligence, Beihang University [3]SenseTime Research [4]ETH Zurich. Correspondence to: Xianglong Liu <xlliu@buaa.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

performance of quantization. For instance, if we preserve weights with extremely large absolute values, the model after sparsification may perform well. However, the subsequent quantization process will encounter a large value range, hindering promising quantization performance. On the other hand, preserving weights with small values can lead to satisfying quantization performance but poor sparsification results. Therefore, designing a new mechanism to balance the effects of these two techniques is desirable.

To address this issue, we propose a new sparsity metric called Salience with Activation Range (SAR) in our JSQ framework, bridging the mismatch between sparsification and quantization. Specifically, our SAR metric additionally takes activation ranges after sparsification into account in the compression process. We assign lower importance of specific weights if activation range becomes large after removing these weights. As a result, the sparse model after sparsification will have small activation ranges, which is friendly to quantization.

In addition to the contradiction between sparsification and quantization, the second issue is how to deal with outliers. It has been proven that outliers play a crucial role in LLMs (Wei et al., 2023; Xiao et al., 2023). However, these outliers make the LLM difficult to sparsify (Yin et al., 2023) or quantize (Shao et al., 2023) due to their extremely large values. Existing approaches mitigate this problem by using migration (Wei et al., 2023) or clipping (Shao et al., 2023) techniques. But these techniques are highly coupled with the quantization process. If we directly apply these techniques for joint sparsification and quantization, it can only follow the sparsity-clip/migration-quantization pipeline, in which these techniques are not helpful for the sparsity process, causing a sub-optimal problem.

To tackle the second problem, we further propose a search-based activation editor to automatically remove relatively useless outliers before sparsity. We reveal that rather than elegantly incorporating migration or clipping in quantization, directly editing the outliers in activation before sparsity and quantization proves remarkably effective. The core challenge is to find which and how we should edit outliers. Therefore, we propose employing the simulated annealing algorithm to search for an optimal ratio for suppressing the activation, allowing for the smooth removal of relatively useless outliers to enhance sparsification and quantization.

Our contribution are as follows:

► To the best of our knowledge, we propose the first model compression framework JSQ to simultaneously utilize sparsification and quantization to compress large language models under high compression ratios.
► We introduce a new sparsity metric called salience with activation range that bridges the mismatch between sparsification and quantization.
► We propose a new activation editing strategy called the search-based activation editor, which utilizes a simulated annealing algorithm to search for an optimal ratio to truncate the outlier distribution.
► Comprehensive experiments on multiple datasets answer the aforementioned problem: Yes, we can maintain the performance of LLMs under high compression ratios.

2. Related Work

Large language models. The emergence of large language models (LLMs) has become a milestone in the field of natural language processing. For example, (Radford & Narasimhan, 2018) proposed the GPT model to stack multiple transformer decoder blocks. Meta released LLaMA (Touvron et al., 2023a) based on an improved transformer architecture, which is further extended to LLaMA2 (Touvron et al., 2023b). (Zheng et al., 2023) proposed Vicuna, which has an excellent ability to understand long context. (Almazrouei et al., 2023) proposed Falcon, which has state-of-the-art performance on most benchmarks compared to other models of the same size. Although these LLMs achieve significant successes, they suffer from massive parameter sizes and a huge computation burden. In this work, we aim to compress these LLMs for better efficiency.

Network sparsification. Network sparisification aims to remove unimportant weights to accelerate the model (Yang et al., 2024; Guo et al., 2023a;b; 2020b;a;c; 2022b; 2021). (LeCun et al., 1989) proposed the first network sparsification method OBD, using the second derivative information of Taylor series as the weight importance. Based on OBD, (Hassibi et al., 1993) achieves better performance by analyzing the recursion relation for calculating the inverse Hessian matrix. Recently, many network sparsification approaches were proposed to reduce the computational and storage burden of LLMs. For example, (Sun et al., 2023) proposed an unstructural sparsity method called Wanda based on the weight magnitudes and the corresponding input activations. By taking activations into account, Wanda is able to effectively remove less important weights and induce sparsity in pretrained LLMs. LLM-Pruner (Ma et al., 2023) prunes coupled structures in unison by analyzing structure dependency. SparseGPT (Frantar & Alistarh, 2023a) incrementally prunes each column of the weight matrix and update the remaining weights in those rows as compensation. All of these approaches do not jointly consider quantization. In contrast, our JPQ approach truncates the distribution in the sparsification process for better quantization.

Network quantization. Many methods were also proposed for quantizing deep neural networks (Lv et al., 2024; Qin et al., 2023a;b;c). For example, Quant-Noise (Fan et al., 2021) quantizes some random parts of weights during each

forward, and updates most of the weights using unbiased gradients. (Guo et al., 2022a) proposed a data-free quantization method by taking advantage of approximate Hessian information. Recently, with the popularity of LLMs, some quantization methods especially designed for LLMs have also been proposed (Wei et al., 2023; Huang et al., 2024; Qin et al., 2024). For example, GPTQ (Frantar et al., 2023) improves OBQ (Frantar & Alistarh, 2022) so that it can run on LLMs with massive parameter scale. SmoothQuant (Xiao et al., 2023) addresses the outlier issue by smoothly transferring the difficulty of quantizing activations to weights. AWQ (Lin et al., 2023) greatly reduces the quantization error by protecting few salient parameters. OminiQuant (Shao et al., 2023) introduces an omnidirectionally calibrated quantization technique for LLMs by using learnable weight clipping and learnable equivalent transformation. However, these methods only consider how to quantize the model without considering the sparsity. So, they often crash under higher compression ratio.

Joint optimization. There are also joint optimization approaches to simultaneously consider sparsification and quantization (Wang et al., 2020b). For example, (Ullrich et al., 2017) utilizes soft weight-sharing to simultaneously perform sparsification and quantization in a single step. (Wang et al., 2020a) significantly reduces the time consumption of architecture searching by employing a quantization-aware accuracy predictor. (Wang et al., 2021) determines the optimal trade-off between sparsity and quantization through solving a joint gradient-based optimization problem. (Frantar & Alistarh, 2022) introduces a new compression framework that considers both sparsity and quantization in a unified setting. These methods are not designed for LLMs. Therefore, they do not consider how to deal with outliers in LLM in the joint sparsification and quantization scheme.

3. Methodology

3.1. Overview

The overview of our Joint Sparsification and Quantization (JSQ) framework is shown in Fig. 2. Given a pretrained LLM, we first use our activation editor to remove the relatively useless outliers in activation. After editing the activation, we use our Salience with Activation Range (SAR) metric to sparsify and quantize the weights and activation in a layer-by-layer fashion. We iteratively perform this process and use the simulated annealing search to find an optimal editing vector. After searching, we use the best editing vector to compress the LLM and obtain the compact model.

3.2. Salience with Activation Range

Motivation. As introduced in Sec. 1, sparsity and quantization are contradictory. Sparsification tends to preserve parameters with a large absolute value, while quantization prefers a small parameter range. To mitigate this problem, we propose a new sparsity metric called Salience with Activation Range (SAR) to consider the subsequent quantization process during sparsification.

Salience with activation outlier. Fig. 2 illustrates our SAR metric. Formally, let us consider a linear layer in LLM. Denote the input activation as $\mathbf{X} \in \mathbb{R}^{N \times c_{in}}$ and the weight as $\mathbf{W} \in \mathbb{R}^{c_{out} \times c_{in}}$, where N, c_{in}, c_{out} are number of tokens, input dimensions, and output dimensions, respectively. The output of this layer $\mathbf{Y}$ can be written as follows:

$$\begin{aligned} \mathbf{Y} &= \mathbf{X}\mathbf{W}^{\mathbf{T}} \\ \text{where } \mathbf{Y}_{ij} &= \Sigma_{k=1}^{c_{in}} \mathbf{X}_{ik} \cdot \mathbf{W}_{jk}. \end{aligned} \tag{1}$$

Here, the subscript $\cdot_{ij}$ denotes the element at ith row and jth column in the weight or input matrix.

When compressing LLM, we aim to sparsify the weight matrix of each layer and also quantize both weights and activation subsequently for larger compression ratio. As outliers play an important role in LLM, Wanda (Sun et al., 2023) designs a criterion with the consideration of activation, which can be written as follows:

$$\mathbf{I}_{ij} = \|\mathbf{X}_j\|_2 \cdot |\mathbf{W}_{ij}|, \tag{2}$$

where $\|\mathbf{X}_j\|_2$ is the Frobenius norm of the jth channel aggregated along the token dimension. $|\cdot|$ denotes the absolute value. In this way, we can calculate the importance $\mathbf{I} \in \mathbb{R}^{c_{out} \times c_{in}}$ for each individual weight and sparsify them based on the importance.

This metric tends to preserve outliers in LLM for more information, resulting in an output activation with large values. However, it does not consider the quantization process. Intuitively, quantization prefers the activation with a small range, causing a contradiction with this sparsity metric. To mitigate this problem, we design our SAR metric by considering both activation outliers and the active range. As outliers often stay in specific channels, we are allowed to perform per-channel outlier suppression, which often leads to a smaller per-token activation range as well (Xiao et al., 2023; Li et al., 2023). Therefore, our goal is to minimize the activation range of each output channel in our SAR metric.

Auxiliary salience. To achieve the aforementioned goal, we need to design a metric that can model the effect of each weight on the output activation. The intuitive idea is to measure the activation range of each channel after removing each individual weight and incorporate this measurement into the weight salience. From Eq. 1 and Fig. 2, we observe the jth output channel is only related to jth row in $\mathbf{W}$ (i.e., jth column in $\mathbf{W}^T$). Based on this observation, we first

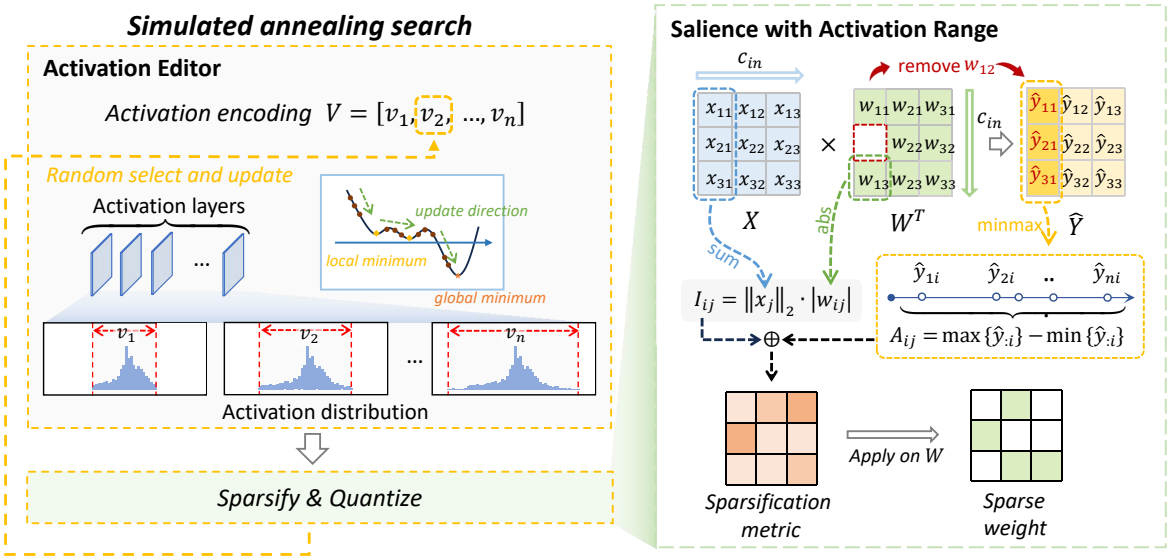

Figure 2. Overview of our JSQ approach: We first edit the activation to remove relatively useless outliers. Then, we sparsify and quantize the model based on salience with the activation range metric. We employ simulated anneal search to find the best editing vector.

define an auxiliary salience $\mathbf{A} \in \mathbb{R}^{c_{out} \times c_{in}}$ as follows:

$$\begin{aligned} \mathbf{A}_{ij} &= \max(\hat{\mathbf{Y}}_{:i}) - \min(\hat{\mathbf{Y}}_{:i}), \\ \text{where } \hat{\mathbf{Y}} &= \mathbf{X} \cdot (\Phi(\mathbf{W}; i; j))^T. \end{aligned} \tag{3}$$

Here, $\Phi(\mathbf{W}; i; j))$ denotes an auxiliary weight matrix when setting the element at ith row and jth column as 0 in $\mathbf{W}$. $\max(\cdot)$ and $\min(\cdot)$ denotes the maximum and minimum values of the vector, respectively. $\hat{\mathbf{Y}}_{:i}$ is the ith column of the matrix $\hat{\mathbf{Y}}$. Each element in auxiliary matrix $\mathbf{A}$ reflect a simple characteristic: how will the activation range changes if we sparsify this element in weight. In this way, we can use the range salience to measure the effect of each weight on the subsequent quantization process.

Salience with activation range. By jointly consider the salience with activation outlier and auxiliary salience, our salience with activation range metric $\mathbf{S} \in \mathbb{R}^{c_{out} \times c_{in}}$ can be defined as:

$$\mathbf{S}_{ij} = \mathbf{I}_{ij} + \lambda \mathbf{A}_{ij}, \tag{4}$$

where λ is the factor to trade-off different terms. In our implementation, we empirically set λ as 2.

The SAR metric in Eq. 4 reflects an intuitive design concept. If sparsifying a weight will result in a large activation range (i.e., large $\mathbf{A}_{ij}$), we should assign higher salience on this weight to preserve it, or vice versa. By using our SAR metric, we can achieve a better trade-off between preserving outliers for more information and minimizing activation range for better quantization. In other words, the side-effect of preserving outliers on quantization will be alleviated. The activation range after sparsification can be seen in Fig. 5, from which we have reduction on activation range after considering the auxiliary salience.

3.3. Search-based Activation Editor

Motivation. It is proven that outliers are important in LLMs. To deal with outliers, many compression approaches design migration (Wei et al., 2023) or clipping (Shao et al., 2023) strategy, which is elegantly coupled with quantization. However, we reveal that it is surprisingly effective to directly edit the activation before compression, which can further strengthen the balance between outliers and compression. Fig. 3 demonstrates this finding. We measure the perplexity of compressed LLaMA-7B on WikiText under different editing strength. Specifically, we first suppress the outliers in activation and then compress the model using sparsity and quantization. As we start to remove the outliers in activation, the performance of compressed LLM first increase. We argue that when we start the suppression, the benefit from sparsification and quantization will surpass the performance drop caused by the outlier removal. As we continue to increase the editing strength, the critical outliers will start to be edited as well, and thus the drawback of activation editing will surpass its benefit from sparsification and quantization. So it is important to find a proper activation editor to suppress the outliers. To solve this problem, we propose our search-based activation editor (SAE).

Activation editing. Formally, let us denote the activation of one layer to be edited in LLM as $\mathbf{F}$, the activation editing can be written as:

$$\begin{aligned} \mathbf{F}_{edit} &= \text{clamp}(\mathbf{F}, LB, HB), \\ \text{where } LB &= r \cdot \max(\mathbf{F}) + (1 - r) \cdot \min(\mathbf{F}), \\ \text{and } HB &= (1 - r) \cdot max(\mathbf{F}) + r \cdot \min(\mathbf{F}). \end{aligned} \tag{5}$$

$\text{clamp}()$ denotes clamp operation on $\mathbf{F}$ with the lower

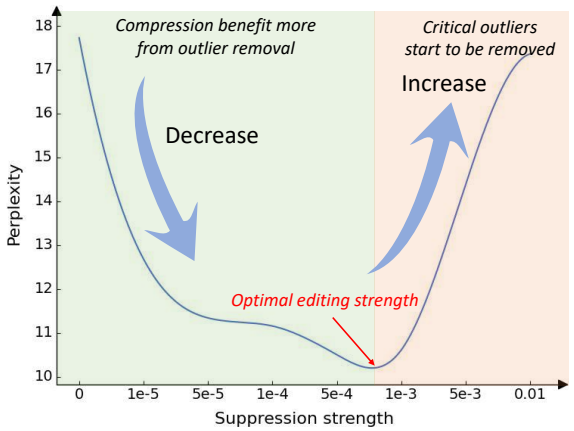

Figure 3. Perplexity under different activation editing strengths.

bound LB and higher bound HB. $r \in [0, 1]$ is the editing strength. Eq. 5 indicates we clamp the activation range to $[r, 1-r]$ of the original one to eliminate the outliers.

Activation encoding. It is non-trivial to determine the editing strength r as there is a large amount of possible combinations for the entire network. So we propose to search for the optimal one. To enable search, we encode the editing strength into a vector. For example, suppose we have L activation layers to be edited, we design a set of editing strength choices for each layer $R = \{r_1, r_2, \ldots, r_n\}$, where n is the number of predefined choices to provide a good starting point for searching. Therefore, we can form an editing strength of the whole network into a L-dimensional editing vector $V = [v_1, v_2, \ldots, v_L]$ and each element v_l can be chosen from R.

Simulated annealing search. After the activation encoding, we aim to search for the optimal editing vector V. Formally, the overall goal can be written as follows:

$$V^* = \arg\max_V \mathcal{P}(\mathrm{COM}(LLM; V)). \tag{6}$$

V is the editing vector for whole network. $\mathrm{COM}(LLM; V)$ denotes compressing the LLM using the editing vector V. $\mathcal{P}$ is the performance of the model. In this equation, we aim to search for a editing vector for the optimal LLM performance after compression.

To achieve this, we first randomly initialize our editing vector. Then, we utilize the simulated annealing algorithm to automatically search for the optimal editing vector. Specifically, we edit the activation of each layer based on the editing vector, and use our SAR metric to sparsify and quantize the edited model. After that, we test the compressed model on the calibration data. We randomly change one element in the editing vector V and iteratively perform this process to search for the best editing vector V^*. In this way, we can smoothly edit the relatively useless outliers for subsequent sparsification and quantization process.

Algorithm 1 Our JSQ workflow for compressing LLMs

Input: Pretrained model LLM; Calibration data $\mathcal{D}$; Initial and final temperature T_i, T_f.
Initialize current temperature $T = T_i$, editing vector V.
while $T > T_f$ **do**
  //Activation editing and sparsification&quantization
  Forward pass on $\mathcal{D}$ to obtain activations.
  Edit LLM activation based on V.
  Sparsify and quantization by SAR metric.
  //Simulated annealing search
  Test performance P of the compressed model on $\mathcal{D}$.
  if P is better than P_{best} **then**
    Update editing vector V and $P_{best} = P$.
  else
    Update editing vector V and $P_{best} = P$ with the probability $1 - exp[(P - P_{best})/T]$.
  end if
  Reduce temperature T based on schedule.
end while
Output: Compressed large language model.

By seamlessly incorporating salience with activation range and search-based activation editor, we construct our JSQ framework, and the workflow can be seen in Algorithm 1.

4. Experiments

4.1. Datasets and Models

We evaluate JSQ framework on most widely used LLM model families including: LLaMA (Touvron et al., 2023a), LLaMA-2 (Touvron et al., 2023b), and ChatGLM3 (Du et al., 2021). We follow LLaMA's protocol to perform zero-shot classification evaluation on commonly used datasets including PIQA (Bisk et al., 2020), BoolQ (Clark et al., 2019), MMLU (Hendrycks et al., 2020), HellaSwag (Zellers et al., 2019), Arc-easy (Clark et al., 2018), Arc-challenge (Clark et al., 2018), and WinoGrande (Sakaguchi et al., 2021). Following previous works on LLM compression (Sun et al., 2023; Xiao et al., 2023), we also evaluate the perplexity on WikiText2 (Merity et al., 2016).

4.2. Implementation Details

For fair comparison, we follow (Sun et al., 2023) to utilize uniform sparsity for all linear layers. We set the input length as 2,048. The editing strength choice R is set as $\{0, 4e^{-5}, 5e^{-5}, 6e^{-5}, 7e^{-5}\}$. To calculate the SAR metric, we use 128 samples from C4 (Raffel et al., 2020) as the calibration data. In the simulated annealing search, we set the initial and the final temperature as 300 and 10, respectively.

We adjust the sparsity level of each layer and the bit-width of parameters to achieve compressed model with different

Table 1. Zero-shot performance. Avg. is calculated among seven classification datasets. Red: the best result. Blue: the second-best result.

Method	#MACs	Model Size	WikiText↓	PIQA	BoolQ	MMLU	HellaSwag	Arc-e	Arc-c	WinoGrande	Avg.
LLaMA-7B (Touvron et al., 2023a)	14.64T	12.6G	9.42	79.16	74.95	33.0	76.20	72.81	44.71	70.01	64.41
Wanda (Sun et al., 2023)	3.24T	1.8G	260364.00	50.98	48.41	22.9	26.46	26.85	26.88	50.28	36.11
LLM-Pruner (Ma et al., 2023)	3.24T	1.8G	2535837.61	49.35	46.91	23.5	26.26	26.77	28.92	49.96	35.95
SparseGPT (Frantar & Alistarh, 2023b)	3.24T	1.8G	5789.33	51.41	37.98	23.2	27.15	26.01	25.09	48.70	34.22
SmoothQuant (Xiao et al., 2023)	3.24T	4.7G	12.70	76.93	72.60	28.4	71.56	70.16	41.72	66.06	61.06
OmniQuant (Shao et al., 2023)	3.24T	4.7G	12.49	77.24	73.27	28.5	71.33	69.10	42.11	68.10	61.38
Wanda+SmoothQuant	3.24T	3.5G	12.34	77.97	72.69	28.1	71.84	68.27	41.89	68.43	61.31
SparseGPT+SmoothQuant	3.24T	3.5G	11.98	77.20	73.24	30.2	70.91	68.48	41.21	69.30	61.51
JSQ (Ours)	3.24T	3.5G	11.04	78.72	74.58	30.0	72.33	69.30	43.42	69.58	62.56
LLaMA-13B (Touvron et al., 2023a)	26.52T	24.3G	8.21	80.14	77.89	42.1	79.09	74.75	47.70	72.77	67.78
Wanda (Sun et al., 2023)	4.09T	3.4G	172946.83	50.33	37.83	25.7	26.47	26.81	26.54	49.49	34.74
LLM-Pruner (Ma et al., 2023)	4.09T	3.4G	934523.91	49.67	38.10	25.0	25.52	25.97	29.10	49.96	34.76
SparseGPT (Frantar & Alistarh, 2023b)	4.09T	3.4G	1265.67	51.36	38.07	23.2	28.02	27.82	23.46	49.72	34.52
SmoothQuant (Xiao et al., 2023)	4.09T	9.1G	11.65	77.80	70.98	31.9	76.97	68.52	41.72	69.61	62.50
OmniQuant (Shao et al., 2023)	4.09T	9.1G	10.27	78.80	75.43	36.4	75.31	70.11	43.63	70.32	64.29
Wanda+SmoothQuant	4.09T	6.8G	10.39	78.45	75.66	31.5	75.54	71.21	44.71	69.61	63.81
SparseGPT+SmoothQuant	4.09T	6.8G	10.49	78.40	76.61	32.8	75.45	69.53	44.11	67.96	63.55
JSQ (Ours)	4.09T	6.8G	9.58	80.05	78.16	38.2	75.41	72.93	46.67	72.24	66.24
LLaMA-33B (Touvron et al., 2023a)	33.12T	60.6G	6.24	82.21	82.72	54.4	82.62	78.91	52.90	75.77	72.79
Wanda (Sun et al., 2023)	4.77T	8.5G	313764.56	50.00	42.78	22.8	26.20	26.56	27.39	49.64	35.05
LLM-Pruner (Ma et al., 2023)	4.77T	8.5G	3533383.20	48.97	39.63	24.2	25.94	26.18	26.96	49.01	34.41
SparseGPT (Frantar & Alistarh, 2023b)	4.77T	8.5G	461.82	52.34	57.25	22.9	29.10	28.70	22.18	47.99	37.21
SmoothQuant (Xiao et al., 2023)	4.77T	22.7G	11.05	77.04	76.24	36.8	76.86	72.56	50.17	72.45	66.02
OmniQuant (Shao et al., 2023)	4.77T	22.7G	9.41	79.05	78.20	45.1	78.54	73.37	50.82	75.22	68.61
Wanda+SmoothQuant	4.77T	17.0G	8.00	77.97	80.98	38.2	79.35	74.71	51.11	74.19	68.07
SparseGPT+SmoothQuant	4.77T	17.0G	8.05	77.75	82.78	43.7	79.20	74.37	50.43	74.66	68.98
JSQ (Ours)	4.77T	17.0G	7.68	79.20	81.92	49.3	80.76	76.35	51.92	77.51	70.27
LLaMA2-7B (Touvron et al., 2023b)	14.64T	12.6G	8.79	79.11	77.71	41.6	76.01	74.49	46.25	69.06	66.32
Wanda (Sun et al., 2023)	3.24T	1.8G	100584.54	48.97	37.83	25.5	26.31	26.26	26.96	49.96	34.54
SmoothQuant (Xiao et al., 2023)	3.24T	4.7G	12.22	76.12	72.32	35.2	72.86	70.63	43.69	64.56	62.20
Wanda+SmoothQuant	3.24T	3.5G	10.74	78.35	75.44	34.2	72.91	69.53	43.17	67.17	62.97
JSQ (Ours)	3.24T	3.5G	10.64	79.09	76.10	34.6	73.28	71.23	45.56	69.03	64.13
LLaMA2-13B (Touvron et al., 2023b)	26.52T	24.3G	7.90	80.52	80.61	52.1	79.37	77.48	49.06	72.30	70.21
Wanda (Sun et al., 2023)	4.09T	3.4G	201702.58	49.29	37.83	22.9	25.81	27.06	26.79	51.70	34.48
SmoothQuant (Xiao et al., 2023)	4.09T	9.1G	9.32	77.97	76.42	46.4	75.85	73.78	45.14	67.64	66.17
Wanda+SmoothQuant	4.09T	6.8G	9.79	78.54	78.91	46.7	76.64	73.23	47.61	70.09	67.38
JSQ (Ours)	4.09T	6.8G	8.58	79.25	79.61	48.5	76.32	75.51	46.25	71.84	68.18
ChatGLM3-6B (Du et al., 2021)	12.24T	11.6G	10.12	81.45	86.45	62.1	78.01	79.59	53.58	72.61	73.40
Wanda (Sun et al., 2023)	2.20T	1.6G	49249.03	51.58	49.20	23.0	26.18	25.00	25.94	49.33	35.75
SmoothQuant (Xiao et al., 2023)	2.20T	4.4G	15.23	73.12	75.54	48.7	51.30	60.82	36.60	53.83	57.13
Wanda+SmoothQuant	2.20T	3.3G	12.01	78.67	83.43	57.7	72.42	73.86	46.59	68.35	68.72
JSQ (Ours)	2.20T	3.3G	11.83	79.89	83.89	57.9	73.34	75.10	47.02	69.74	69.55

Table 2. Structured 2:4 and 4:8 sparsity performance on WikiText.

Method	#MACs	Model Size	WikiText↓
LLaMA-7B	14.64T	12.6G	9.42
Unstructured	3.03T	3.2G	11.45
Structured 2:4	3.03T	3.2G	13.54
Structured 4:8	3.03T	3.2G	12.15
LLaMA2-7B	14.64T	12.6G	8.79
Unstructured	3.03T	3.2G	10.89
Structured 2:4	3.03T	3.2G	13.24
Structured 4:8	3.03T	3.2G	12.07

Table 3. Ablation study for different components on WikiText.

Method	#MACs	Model Size	WikiText↓
JSQ	3.24T	3.5G	11.04
-SAR	3.24T	3.5G	11.75
-Editing	3.24T	3.5G	11.62
-Search	3.24T	3.5G	11.29

complexity. For quantization methods like SmoothQuant and OmniQuant, we quantize the full-precision model to different bit-width. For sparsity methods like Wanda, LLM-Pruner, and SparseGPT, we adjust sparsity ratio of the model to achieve similar #MACs. For joint optimization methods like Wanda+SmoothQuant, SparseGPT+SmoothQuant, and our JSQ, we jointly adjust bit-width and sparsity ratio to achieve target #MACs. We report the results under this setting because most of baseline methods will crash under lower computation complexities. So it is less meaningful to compare with them under lower complexities.

4.3. Experimental Results

We compare our JSQ framework with several state-of-the-art LLM compression approaches, including sparsity approaches: LLM-Pruner (Ma et al., 2023), Wanda (Sun et al., 2023), SparseGPT (Frantar & Alistarh, 2022), and quantization approaches: SmoothQuant (Xiao et al., 2023), OmniQuant (Shao et al., 2023), as well as their combination: Wanda+SmoothQuant and SparseGPT+SmoothQuant. For Wanda+SmoothQuant or SparseGPT+SmoothQuant, we

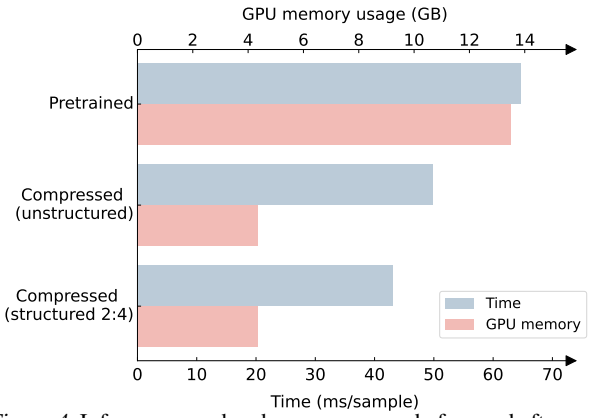

Figure 4. Inference speed and memory usage before and after compression.

Table 4. Performance of JSQ using different subsequent quantization approaches.

Method	#MACs	Model Size	WikiText↓
JSQ	3.24T	3.5G	11.04
JSQ (GPTQ)	3.24T	3.5G	11.06

first use the corresponding sparsity approach to sparsify the model and then quantize the sparse model using the corresponding quantization methods. For #MACs and Model Size, we report the theoretical value for these metrics.

Zero-shot performance. Table 1 shows the zero-shot performance of the compressed model. From Table 1, we have the following observations: (1) Under the same computation, quantization is better than the sparsity approaches. Therefore, quantization can act as a more useful model compression methods for large language models. (2) Under high compression ratio, sparsity methods like Wanda and SparseGPT collapse under most of cases, indicating sparsity only compression paradigm may not suitable for LLM under high compression ratio. (3) The performance of combining different compression methods (e.g., sparsity and quantization) is better than only using one specific compression approach in the most of cases. So it is useful to simultaneously use different compression methods for efficient LLM application. (4) Our JSQ framework outperforms state-of-the-art LLM compression approaches including Wanda and SmoothQuant, and their combination. For example, our JSQ can surpass SmoothQuant by 22.04% on HellaSwag when compressing ChatGLM.

Structured 2:4 and 4:8 sparsity. In Table 1, we report the performance of our JSQ when we use unstructured sparsity in the sparsification process. In Table 2, we also report the performance when using structured 2:4 and 4:8 sparsities in the sparsification process. Compared with unstructured sparsity, we can have better practical acceleration performance using 2:4 and 4:8 sparsities with limited performance drop.

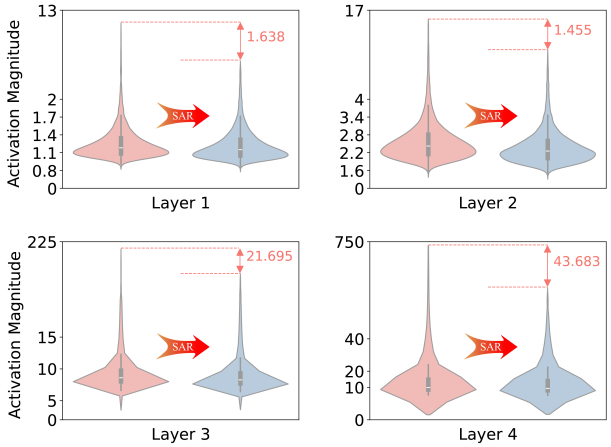

Figure 5. Visualization of activation range after sparsification with and without considering activation range.

Table 5. Performance of JSQ with and without finetuning.

Method	#MACs	Model Size	WikiText↓
LLaMA-7B	14.64T	12.6G	9.42
JSQ	3.24T	3.5G	11.04
JSQ (finetune)	3.24T	3.5G	10.30

Inference speed. We demonstrate the acceleration and the GPU memory reduction of the compressed model in Fig. 4. From Fig. 4, our JSQ framework can bring 23% inference acceleration for compressed model with unstructured sparsity, and 33% acceleration for compressed model with structured 2:4 sparsity. In terms of GPU memory, the compressed model can achieve 68% reduction.

4.4. Ablation Study

In this section, we use LLaMA-7B on WikiText as an example to perform extensive ablation studies.

Effect of salience with activation range. To investigate the effect of the SAR metric, we remove the auxiliary salience $\mathbf{A}$ in Eq. 4, and the result is denoted as "-SAR" in Table 3. Our SAR metric can bring 0.71 performance improvement on WikiText.

Effect of activation editing. We also report the performance without editing activation. In this case, we only use the SAR metric in Eq. 4 to compress the LLM, and the result is shown as "-Editing" in Table 3. Our search-based activation editor brings 0.58 performance improvement on WikiText for LLM compression.

Effect of simulated annealing search. We also report the performance without using the simulated annealing search

Table 6. Hyperparameter analysis of λ.

Value of λ	1	2	3	4	5
WikiText↓	11.15	11.04	14.22	147.83	136.64

Table 7. Some examples of generations obtained with the original model and the compressed model.

Model	Origin Model	Compressed Model
LLaMA	It is a beautiful day. My wife is looking good. She has on a pretty dress. Her smile is as big as the sun. We go to the beach and have a picnic. We play in the water and eat a big juicy hamburger. We have a great day together...	It is a beautiful day. I can't help but to get out of the car and walk along the beach. I see a couple fishing, and a group of kids playing soccer on the sand. I'm looking at the sky, and it's just like the sky in a Thomas Kinkade painting...
LLaMA2	The moon is so beautiful tonight. I am outside and look up and I see it and my heart starts to beat faster. It is an amazing sight and I am in awe. It's as if the moon knows what is going on in my life and it is beckoning me to it...	The moon is so beautiful tonight. Everybody in the neighbourhood is taking photos of the moon. It seems to be the day for it. I even saw people taking photos of the moon on a bridge. All this reminds me of a beautiful time a few years ago...
ChatGLM3	Write a poem about life. Life is a strange journey, filled with ups and downs, twists and turns. It's not always easy, but it's worth the ride. Some days are sunny, some days are stormy, but it's how we weather the storm that matters...	Write a poem about life. My life like a poem, a story of growth. The pain and the joy, the laughter and the tears. Each chapter of my story contains many words, together they form a picture of a life well lived. As the years go by, I write more lines...

for activation editing. In this case, we empirically set the editing strength r for each layer as $5e^{-5}$, and the result is denoted as "-Search" in Table 3. The simulated annealing search for activation editing can bring 0.25 performance improvement on WikiText.

4.5. Algorithm Analysis

In this section, we also use LLaMA-7B on WikiText to conduct the experiments on algorithm analysis.

Visualization on activation range. In Fig. 5, we visualize the activation range after sparsification with and without considering activation range. The height indicates the activation range, while the width suggests the distribution density at different value. We observe our SAR metric can effectively reduce the activation range of the sparse model, which is more beneficial to the subsequent quantization process.

Analysis on different quantization approaches. We also report the performance when using different subsequent quantization approach, and the results are shown in Table 4. JSQ (GPTQ) uses GPTQ (Frantar et al., 2023) as the subsequent quantization approach. JSQ and JSQ (GPTQ) can achieve comparable performance, demonstrating the generalization ability of JSQ.

Impact of finetuning. Although we do not use finetuning technique in our JSQ for faster compression speed, we can further improve the LLM compression performance by additionally utilizing the finetuning. In Table 5, we report the performance after using C4 datasets (Raffel et al., 2020) to finetune our compressed model. Our JSQ can achieve further improvement using the finetuning technique.

Analysis on λ in Eq. 4. We conduct the experiment to investigate the performance when using different values of λ, and the result is shown in Table 6. The sweet range of λ is from 1 to 3. This is because the importance $\mathbf{I}$ and auxiliary salience $\lambda\mathbf{A}$ in Eq. 4 are in similar magnitude when $1 < \lambda < 3$. When $\lambda > 3$, $\lambda\mathbf{A}$ will much larger than $\mathbf{I}$, causing degraded performance.

Table 8. Performance using different numbers of calibration data.

#Calibration data	8	16	32	64	128
WikiText↓	11.14	11.13	11.06	11.05	11.04

Impact of calibration data. In Table 8, we investigate the impact of the amount of calibration data. We find our JSQ framework can achieve best performance using 128 calibration data. So we use 128 calibration data by default.

Generalization on different complexities. To demonstrate the generalization ability of our JSQ framework, we report the performance of compressed LLMs under different complexities. From Fig. 1, we observe our JSQ framework outperforms other model compression methods under different #MACs, especially under large compression ratio. We hypothesize that this is because if we only compress LLM from one dimension (e.g., sparsity), parameters containing critical information will be removed. However, if we jointly use multiple compression dimension, we can avoid removing significant parameter in each dimension to achieve high compression ratio. So it is beneficial to simultaneously use sparsification and quantization for LLM compression.

Case study. We provide some examples of sentences generated by the original and compressed models in Table 7. The sentences generated by the compressed model are comparable to original model in terms of fluency, relevance, and informativeness regarding the given topic. Nevertheless, during our experiments, we observe the compressed model occasionally generate sentences that are meaningless or contain repetitive tokens.

5. Conclusion

In this paper, we proposed Joint Sparsification and Quantization (JSQ) framework to compress LLM under high compression ratio, which is the first work to simultaneously utilize sparsification and quantization techniques for compression LLMs. We first propose a new sparsity metric called salience with activation range to bridge the mismatch between sparsification and quantization. We also reveal it is surprisingly effective to directly edit activations for outlier elimination. So we proposed search-based activation editor to automatically remove relatively useless outliers. Extensive experiments on multiple datasets demonstrate we can maintain LLM performance under high compression ratio.

One limitation of our JSQ is we need to predefine the editing strength R, which requires prior knowledge. We will investigate how to automatically design them in future work.

Acknowledgement

This work was supported by the National Science and Technology Major Project (2021ZD0110503), the National Natural Science Foundation of China (No. 62306025, No. 92367204), and Beijing Municipal Science and Technology Project (Nos. Z231100010323002).

Impact Statement

This work aims to compress the large language models for better efficiency in practical usage. So it does not have ethical concerns. It can reduce the computation burden of large language models. So it can help more people in society to use the model, which has positive societal impact.

References

Almazrouei, E., Alobeidli, H., Alshamsi, A., Cappelli, A., Cojocaru, R., Debbah, M., Goffinet, E., Heslow, D., Launay, J., Malartic, Q., Noune, B., Pannier, B., and Penedo, G. Falcon-40B: an open large language model with state-of-the-art performance. 2023.

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

Clark, C., Lee, K., Chang, M.-W., Kwiatkowski, T., Collins, M., and Toutanova, K. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

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

Fan, A., Stock, P., Graham, B., Grave, E., Gribonval, R., Jegou, H., and Joulin, A. Training with quantization noise for extreme model compression, 2021.

Frantar, E. and Alistarh, D. Optimal brain compression: A framework for accurate post-training quantization and pruning. *Advances in Neural Information Processing Systems*, 35:4475–4488, 2022.

Frantar, E. and Alistarh, D. Sparsegpt: Massive language models can be accurately pruned in one-shot. *ArXiv*, abs/2301.00774, 2023a. URL https://api.semanticscholar.org/CorpusID:255372747.

Frantar, E. and Alistarh, D. Sparsegpt: Massive language models can be accurately pruned in one-shot. 2023b.

Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. Gptq: Accurate post-training quantization for generative pre-trained transformers, 2023.

Guo, C., Qiu, Y., Leng, J., Gao, X., Zhang, C., Liu, Y., Yang, F., Zhu, Y., and Guo, M. SQuant: On-the-fly data-free quantization via diagonal hessian approximation. In *International Conference on Learning Representations*, 2022a. URL https://openreview.net/forum?id=JXhROKNZzOc.

Guo, J., Ouyang, W., and Xu, D. Channel pruning guided by classification loss and feature importance. *AAAI*, 2020a.

Guo, J., Ouyang, W., and Xu, D. Multi-dimensional pruning: A unified framework for model compression. In *CVPR*, 2020b.

Guo, J., Zhang, W., Ouyang, W., and Xu, D. Model compression using progressive channel pruning. *IEEE Transactions on Circuits and Systems for Video Technology*, 2020c.

Guo, J., Liu, J., and Xu, D. Jointpruning: Pruning networks along multiple dimensions for efficient point cloud processing. *IEEE Transactions on Circuits and Systems for Video Technology*, 2021.

Guo, J., Liu, J., and Xu, D. 3d-pruning: A model compression framework for efficient 3d action recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(12):8717–8729, 2022b.

Guo, J., Xu, D., and Lu, G. Cbanet: Towards complexity and bitrate adaptive deep image compression using a single network. *IEEE Transactions on Image Processing*, 2023a.

Guo, J., Xu, D., and Ouyang, W. Multidimensional pruning and its extension: A unified framework for model compression. *IEEE Transactions on Neural Networks and Learning Systems*, 2023b.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

Hassibi, B., Stork, D. G., and Wolff, G. J. Optimal brain surgeon and general network pruning. *IEEE International Conference on Neural Networks*, pp. 293–299 vol.1, 1993. URL https://api.semanticscholar.org/CorpusID:61815367.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Huang, W., Liu, Y., Qin, H., Li, Y., Zhang, S., Liu, X., Magno, M., and Qi, X. Billm: Pushing the limit of post-training quantization for llms. In *International Conference on Machine Learning*, 2024.

LeCun, Y., Denker, J. S., and Solla, S. A. Optimal brain damage. In *Neural Information Processing Systems*, 1989. URL https://api.semanticscholar.org/CorpusID:7785881.

Li, Q., Zhang, Y., Li, L., Yao, P., Zhang, B., Chu, X., Sun, Y., Du, L., and Xie, Y. Fptq: Fine-grained post-training quantization for large language models. *arXiv preprint arXiv:2308.15987*, 2023.

Lin, J., Tang, J., Tang, H., Yang, S., Dang, X., Gan, C., and Han, S. Awq: Activation-aware weight quantization for llm compression and acceleration, 2023.

Lv, C., Chen, H., Guo, J., Ding, Y., and Liu, X. Ptq4sam: Post-training quantization for segment anything. *CVPR*, 2024.

Ma, X., Fang, G., and Wang, X. Llm-pruner: On the structural pruning of large language models. *ArXiv*, abs/2305.11627, 2023. URL https://api.semanticscholar.org/CorpusID:258823276.

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

Qin, H., Ding, Y., Zhang, X., Wang, J., Liu, X., and Lu, J. Diverse sample generation: Pushing the limit of generative data-free quantization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023a.

Qin, H., Zhang, M., Ding, Y., Li, A., Cai, Z., Liu, Z., Yu, F., and Liu, X. Bibench: Benchmarking and analyzing network binarization. In *International Conference on Machine Learning*, pp. 28351–28388. PMLR, 2023b.

Qin, H., Zhang, X., Gong, R., Ding, Y., Xu, Y., and Liu, X. Distribution-sensitive information retention for accurate binary neural network. *International Journal of Computer Vision*, 131(1):26–47, 2023c.

Qin, H., Ma, X., Zheng, X., Li, X., Zhang, Y., Liu, S., Luo, J., Liu, X., and Magno, M. Accurate lora-finetuning quantization of llms via information retention. In *International Conference on Machine Learning*, 2024.

Radford, A. and Narasimhan, K. Improving language understanding by generative pre-training. 2018. URL https://api.semanticscholar.org/CorpusID:49313245.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

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

Shao, W., Chen, M., Zhang, Z., Xu, P., Zhao, L., Li, Z., Zhang, K., Gao, P., Qiao, Y., and Luo, P. Omniquant: Omnidirectionally calibrated quantization for large language models, 2023.

Sun, M., Liu, Z., Bair, A., and Kolter, J. Z. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. Llama: Open and efficient foundation language models, 2023a.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

Ullrich, K., Meeds, E., and Welling, M. Soft weight-sharing for neural network compression, 2017.

Wang, T., Wang, K., Cai, H., Lin, J., Liu, Z., and Han, S. Apq: Joint search for network architecture, pruning and quantization policy, 2020a.

Wang, Y., Lu, Y., and Blankevoort, T. Differentiable joint pruning and quantization for hardware efficiency. In *European Conference on Computer Vision*, pp. 259–277. Springer, 2020b.

Wang, Y., Lu, Y., and Blankevoort, T. Differentiable joint pruning and quantization for hardware efficiency, 2021.

Wei, X., Zhang, Y., Li, Y., Zhang, X., Gong, R., Guo, J., and Liu, X. Outlier suppression+: Accurate quantization of large language models by equivalent and optimal shifting and scaling. *arXiv preprint arXiv:2304.09145*, 2023.

Xiao, G., Lin, J., Seznec, M., Wu, H., Demouth, J., and Han, S. Smoothquant: Accurate and efficient post-training quantization for large language models, 2023.

Yang, G., Zhang, C., Gao, L., Guo, Y., and Guo, J. Domain adaptive channel pruning. *Electronics*, 13(5), 2024. ISSN 2079-9292. doi: 10.3390/electronics13050887. URL https://www.mdpi.com/2079-9292/13/5/887.

Yin, L., Wu, Y., Zhang, Z., Hsieh, C.-Y., Wang, Y., Jia, Y., Pechenizkiy, M., Liang, Y., Wang, Z., and Liu, S. Outlier weighed layerwise sparsity (owl): A missing secret sauce for pruning llms to high sparsity, 2023.

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

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., and Stoica, I. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023.

A. Quantitative Results on Different Complexity

The quantitative results on different complexities are shown in Table 9. We use LLaMA-7B as an example to conduct the experiment. We find our JSQ can significantly surpass other baseline methods under low computation complexities. For example, when #MAC=1.84T, the state-of-the-art quantization approach OmniQuant (Shao et al., 2023) crashes. Even the combination method Wanda+SmoothQuant or SparseGPT+SmoothQuant obtain over 100 perplexity on WikiText. In contrast, our JSQ can achieve 21.53 perplexity under this setting, making a notable improvement.

We also notice under low computation, it is beneficial to combine sparsity and quantization methods to compress the model, which further demonstrates the importance of this combination under high compression ratio.

Table 9. Results under different computation complexities. The average is calculated among seven classification datasets. Red: the best result. Blue: the second-best result.

Method	#MACs	Model Size	WikiText↓	PIQA	BoolQ	MMLU	HellaSwag	Arc-e	Arc-c	WinoGrande	Avg.
LLaMA-7B (Touvron et al., 2023a)	14.64T	12.6G	9.42	79.16	74.95	33.0	76.20	72.81	44.71	70.01	64.41
Wanda (Sun et al., 2023)	1.84T	0.4G	2770390.71	49.73	37.83	24.7	26.18	25.21	29.01	49.09	34.54
LLM-Pruner (Ma et al., 2023)	1.84T	0.4G	1853892.00	49.35	45.32	24.7	26.22	25.08	30.63	50.99	36.04
SparseGPT (Frantar & Alistarh, 2023b)	1.84T	0.4G	95290.93	50.22	38.53	23.0	26.03	26.14	27.9	50.43	34.61
SmoothQuant (Xiao et al., 2023)	1.84T	2.4G	4165426.51	48.69	47.49	25.5	25.92	25.93	28.24	54.14	36.56
OmniQuant (Shao et al., 2023)	1.84T	2.4G	120871.22	50.18	48.35	25.8	25.74	25.92	29.17	54.32	37.07
Wanda+SmoothQuant	1.84T	1.4G	403.69	60.12	56.39	23.0	35.28	37.25	24.23	52.09	41.19
SparseGPT+SmoothQuant	1.84T	1.4G	109.11	62.08	59.60	23.0	42.09	43.73	26.28	53.51	44.33
JSQ (Ours)	1.84T	1.4G	21.53	72.70	67.85	26.9	60.44	60.17	31.44	61.74	54.46
LLaMA-7B (Touvron et al., 2023a)	14.64T	12.6G	9.42	79.16	74.95	33.0	76.20	72.81	44.71	70.01	64.41
Wanda (Sun et al., 2023)	2.20T	0.8G	810596.71	49.29	47.03	24.7	25.94	26.6	26.02	50.99	35.80
LLM-Pruner (Ma et al., 2023)	2.20T	0.8G	1460384.52	50.27	41.07	24.5	26.37	25.00	29.27	48.38	34.98
SparseGPT (Frantar & Alistarh, 2023b)	2.20T	0.8G	95653.57	49.51	37.71	27.0	25.70	27.48	26.71	50.36	34.92
SmoothQuant (Xiao et al., 2023)	2.20T	3.2G	260035.17	48.91	38.72	23.5	26.83	27.15	24.15	50.59	34.26
OmniQuant (Shao et al., 2023)	2.20T	3.2G	89752.17	49.21	38.79	25.2	27.72	27.43	26.12	50.57	35.01
Wanda+SmoothQuant	2.20T	2.1G	20.43	72.52	68.65	24.9	61.47	56.99	34.22	60.77	54.22
SparseGPT+SmoothQuant	2.20T	2.1G	19.39	73.67	67.98	27.1	60.95	57.24	36.35	62.90	55.17
JSQ (Ours)	2.20T	2.1G	15.65	75.45	69.77	27.3	63.96	65.58	36.98	66.53	57.94
LLaMA-7B (Touvron et al., 2023a)	14.64T	12.6G	9.42	79.16	74.95	33.0	76.20	72.81	44.71	70.01	64.41
Wanda (Sun et al., 2023)	2.67T	1.2G	734849.55	49.24	37.83	22.9	26.29	26.43	26.45	51.70	34.41
LLM-Pruner (Ma et al., 2023)	2.67T	1.2G	1167339.08	51.03	45.11	24.3	26.13	25.88	27.65	52.01	36.02
SparseGPT (Frantar & Alistarh, 2023b)	2.67T	1.2G	24074.31	49.67	37.83	24.4	25.95	26.35	26.54	48.86	34.23
SmoothQuant (Xiao et al., 2023)	2.67T	3.9G	15.30	73.61	66.67	26.7	66.72	63.38	39.16	63.38	57.09
OmniQuant (Shao et al., 2023)	2.67T	3.9G	14.12	74.48	67.64	27.1	66.87	63.89	41.02	63.29	57.76
Wanda+SmoothQuant	2.67T	3.3G	11.92	76.93	73.64	28.4	71.72	68.94	41.55	65.75	60.99
SparseGPT+SmoothQuant	2.67T	3.3G	11.81	77.80	74.16	28.9	72.31	69.57	41.98	65.75	61.50
JSQ (Ours)	2.67T	3.3G	11.53	78.47	74.41	28.9	73.07	68.51	42.88	70.36	62.37
LLaMA-7B (Touvron et al., 2023a)	14.64T	12.6G	9.42	79.16	74.95	33.0	76.20	72.81	44.71	70.01	64.41
Wanda (Sun et al., 2023)	3.91T	2.4G	76782.09	50.16	62.26	23.0	26.53	27.31	27.22	50.83	38.19
LLM-Pruner (Ma et al., 2023)	3.91T	2.4G	89197.57	50.11	42.39	24.7	25.86	25.88	28.07	50.99	35.43
SparseGPT (Frantar & Alistarh, 2023b)	3.91T	2.4G	697.06	52.94	39.69	23.4	29.42	29.42	22.35	49.96	35.31
SmoothQuant (Xiao et al., 2023)	3.91T	5.5G	9.93	79.16	74.80	30.9	75.02	73.65	44.71	70.01	64.04
OmniQuant (Shao et al., 2023)	3.91T	5.5G	9.51	79.52	75.10	31.2	74.91	73.17	44.83	70.68	64.20
Wanda+SmoothQuant	3.91T	4.6G	10.03	79.00	75.87	31.9	75.12	72.52	44.80	69.46	64.10
SparseGPT+SmoothQuant	3.91T	4.6G	9.89	78.45	76.15	31.2	75.47	70.75	44.45	69.53	63.71
JSQ (Ours)	3.91T	4.6G	9.02	79.72	76.19	32.4	75.80	75.89	45.06	71.13	65.44

B. More Sentence Examples

In Table 10, we provide more examples generated by the original and compressed models. We find the sentences generated by the compressed model are comparable to original model in terms of fluency, relevance, and informativeness regarding the given topic.

Table 10. More examples of generations obtained with the original model and the compressed model.

Model	Origin Model	Compressed Model
LLaMA	With the development of science and technology, the demand of people's life is also constantly improving, and is inseparable from the sustainable development of science and technology. Therefore, the new equipment must be scientific and high-quality. For the company, the energy conservation equipment is also very important, therefore, the choice of machine will pay more attention to the environment to promote the sustainable development of the company...	With the development of science and technology, it is getting more and more important to deal with big data. With the use of computers, big data is processed better and more efficiently than before, though what we can do is really limited. Among big data processing techniques, there is one technique which is important enough to be mentioned but not so famous as the other related techniques, called the big data preprocessing...
LLaMA	The extinction of the dinosaurs can be traced back long. At the end of the Devonian Period, some 350 million years ago, the "Great Dying" occurred—an extinction event wiping out 70 percent of all animal species. The Devonian Period saw the first appearance of sharks and land plants. A vast marine predator called Dunkleosteus ruled the shallow sea. With its armored hide, jaws of crocodile-like teeth, and a bite of 7,000 pounds-per-square...	The extinction of the dinosaurs can be traced back long. While they've been part of myth and legend in our society's imagination for millennia, scientific evidence of dinosaurs really took off in the 19th century. The first dinosaur was identified in 1841, which was named Megalosaurus. It sparked a lot of interest, and more discoveries took place. Moving into the late 19th century, theories started to evolve to explain what caused the dinosaurs to become extinct...
LLaMA2	The world is made of atoms. Most of these atoms in our galaxy are the same. It is the atoms that make up what we see. The sky, the land, the food we eat, the air that we breathe. All of the life on earth is made from the same elements. Elements are the basic building blocks of life. When atoms smash together it releases energy which generates heat and light. In a nuclear reactor, the energy is released with the goal of producing heat. Heat can be used to create energy, create electricity, and heat your home in the winter...	The world is made of atoms. According to modern physics, nothing can be smaller than an atom. And yet, the universe is made of atoms. If you look closely enough at any material you'll see it's not a single atom, but a collection of atoms arranged in one of many possible ways, or a crystal structure. The crystal structures that atoms can form are largely based on three-dimensional shapes such as the cube, the hexagon, or the octahedron – but there are hundreds more...
LLaMA2	In 2008, Beijing hosted the Olympic Games. The magnificent National Stadium, also known as Bird's Nest, was the centerpiece of the entire event. It was an instant visual icon, standing proudly amongst Beijing's other landmark buildings. Bird's Nest was the meeting place for the athletes from around the world. Their unity resonated across the entire world. In Beijing and across China, they united around a love for Chinese food, Chinese culture and Chinese sport. And today, we're all united again...	In 2008, Beijing hosted the Olympic Games. As in all Olympic cities, a large-scale cultural program was set up. A large sports complex, called the Bird's Nest, was built. Today the "Bird's Nest" is a symbol of Beijing's urban cultural life. But is it really a symbol of a new cultural district? Or is this a place of public and entertainment? When approaching the National Stadium, we see a huge square that covers 20.5 hectares of urban space. There are four roads leading from all directions...