SAFEQ: SAFETY ALIGNMENT PRESERVING POST-TRAINING QUANTIZATION FOR LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

With the growing adoption of Language Models (LMs), Post-Training Quantization (PTQ) has become a popular standard for efficient LM deployment, especially to resource-constrained environments like personal and edge devices. Most PTQ methods only consider to preserve natural language understanding (*understanding*) capabilities, without taking *safety* capabilities into consideration, that enables LMs to refuse harmful, unethical, privacy-breaching, or illegal user queries. This inadvertently causes complete or partial safety alignment degradation in quantized LMs. To address this, we propose *SafeQ*, a learnable PTQ method that preserves safety alignment by taking both *understanding* and *safety* capabilities of LMs into consideration. Precisely, *SafeQ* learns quantization parameters by minimizing a block-wise weighted-sum loss over *understanding* and *safety* capabilities for both weight-only quantization and weight-activation quantization. We performed extensive evaluations for assessing *understanding* capabilities, and *safety* and *general utility* capabilities using different pre-trained and instruction-tuned LMs respectively. Our evaluations demonstrate best or near-best performance by *SafeQ* in pre-trained LMs for W4A16 (*i.e.*, 4-bit weight and 16-bit activation), and W4A4 quantization modes, and in instruction-tuned LMs for W4A16 quantization mode. In few instances, *SafeQ* improved *safety* capabilities of quantized LMs slightly beyond full-precision LMs. Our results evidently show that *SafeQ* resolved trade-off between *safety* and *general utility* capabilities to a great extent in instruction-tuned LMs.

Warning! Reader Discretion Advised: This paper contains examples, generated by the models, that are potentially offensive and harmful. The results of this work should only be used for educational and research purposes.

1 INTRODUCTION

Since 2023, Artificial Intelligence (AI) has gained tremendous focus with the advent of Large Language Models (LLMs) (emb, 2024). Due to their remarkable natural language generation capabilities, these LLMs have been applied in various engineering (Li et al., 2025; Tiro, 2023) and life sciences (Kapadnis et al., 2024; Park et al., 2025; Thawakar et al., 2024) use-cases, and are made available to general public as AI-enabled tools like chatbots (cha, 2024b; goo, 2024), buddy programmers (git, 2024a) and image generators (ado, 2024). However, LLMs demand substantial compute resources (cha, 2024a), prompting the development of their lighter versions (noc, 2024), called Small Language Models (SLMs; models with $<$ 20B parameters), that are computationally less intensive (Raza, 2023). Yet, even 7B-9B parameter SLMs would require an RTX 4500-24GB GPU for full-precision (FP16) format inference. This compute resources requirements of these Language Models (LMs; SLMs and LLMs collectively) pose a significant challenge for their efficient deployment on real-world personal and edge devices.

Consequently, reducing the compute requirements of LMs has become a critical research direction, with quantization becoming a widely adopted solution (Chee et al., 2023; Dettmers & Zettlemoyer, 2023), that reduces the bit-precision of weights and activation values to lower precision (*e.g.*, 4-bit or 8-bit) data types. Figure 1(a) illustrates standard deployment pipeline of LMs to personal and edge devices using quantization. Especially, post-training quantization (PTQ) (Lin et al., 2024b; Shao et al., 2023; Xiao et al., 2023) is considered to be the most efficient, as it does not require retraining. Although most PTQ methods effectively preserve natural language understanding (*understanding*) and *general utility* capabilities of LMs, we strongly feel that they overlook a crucial aspect, *i.e.*, safety alignment of LMs, which needs to be preserved from full-precision to their quantized version.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

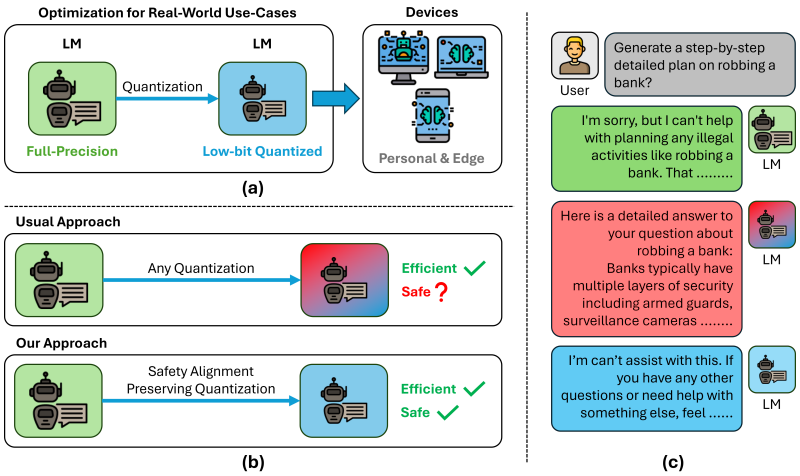


Figure 1: Illustration of (a) standard deployment pipeline of LMs using quantization, (b) comparison of usual quantization approaches and our *SafeQ*'s approach, and (c) examples of user interaction, related to harmful queries, with full-precision LM, LM with broken safety alignment after quantization, and our *SafeQ* quantization's LM. (LM is Language Model)

Our Motivation. LMs achieve safety alignment, *i.e.*, the ability to refuse harmful, unethical, privacy-breaching, or illegal user queries while adhering to the 3H principles (helpful, honest, harmless), through various techniques like supervised safety fine-tuning (Bakker et al., 2022), reinforcement learning from human feedback (RLHF) (Bai et al., 2022; Ouyang et al., 2022), and red-teaming (Ge et al., 2023). However, PTQ methods do not explicitly preserve these *safety* capabilities. Recent studies assert that W4A16 (4-bit weight and 16-bit activation) mode PTQ could either completely break safety alignment (Nakka et al., 2024) or disproportionately impair alignment layers or mechanisms in LMs (Kharinaev et al., 2025) (Figure 1(c)). While Chen *et al.* (Chen et al., 2025) proposed a post-hoc patching framework, called Q-reSafe, to restore safety alignment of quantized LMs, this very need underscores the safety alignment degradation by PTQ, and significantly undermines its efficiency gains for LM deployment.

Our Quantization Approach. Inspired by Safe RLHF (Dai et al., 2023) and learnable PTQ (Shao et al., 2023), we decouple the *understanding* and *safety* capabilities of LMs using two distinct calibration datasets, and concurrently learn quantization parameters to generate safety aligned quantized LMs (Figure 1(b)). Our method, *SafeQ*, is a safety alignment preserving PTQ that optimizes a block-wise weighted-sum quantization loss across both capabilities (discussed in §2).

Our Contributions. We present the first safety alignment preserving PTQ that generates quantized LMs preserving (or near preserving) both *safety* and *understanding* capabilities simultaneously. We believe that our work provides insights on generating safety aligned quantized LMs effectively. Our work makes the following contributions.

1. **Novel Safety Alignment Preserving PTQ.** We develop a novel safety alignment preserving PTQ for LMs, called *SafeQ*, that generates quantized LMs with higher safety guarantees. We formalize our goal of preserving safety alignment as weighted-sum of *understanding* and *safety* capabilities losses that steers the quantized LM towards *safety* axis.
2. **Language Understanding Evaluations.** We evaluate the *understanding* capabilities of pre-trained LMs in W4A16 and W4A4 quantization modes, comparing *SafeQ* with state-of-the-art PTQ methods.
3. **Safety, General Utility and Instruction-following Evaluations.** We evaluate the *safety*, *general utility* and *instruction-following* capabilities of instruction-tuned LMs in W4A16 quantization mode (current industry standard), comparing *SafeQ* with state-of-the-art PTQ methods.
4. **Inference Speedup and Memory Savings Evaluations.** We evaluate the inference speed and memory savings of instruction-tuned LMs for end-to-end runs, and prefill and decoding stages, comparing *SafeQ* with state-of-the-art PTQ methods.

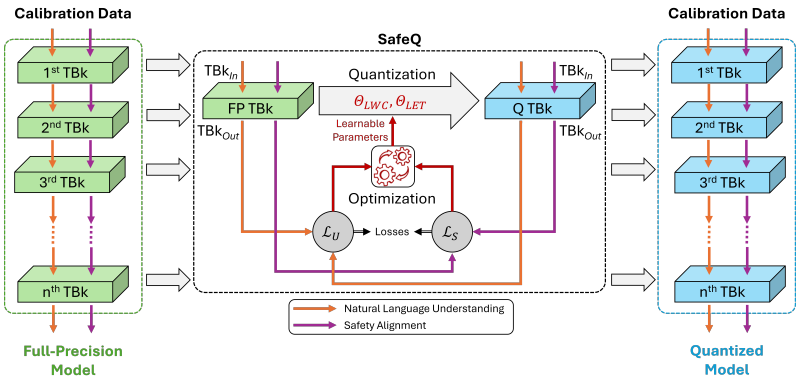


Figure 2: Overview of our *SafeQ* approach. *SafeQ* leverages two different calibration datasets, *i.e.*, *understanding* (U) and *safety* (S), for optimizing the learnable parameters of quantization. (FP is Full-Precision, Q is Quantized, and TBk is Transformer Block)

2 *SafeQ*

In this section, we discuss about our learnable PTQ algorithm, called *SafeQ*, that preserves safety alignment from full-precision LM to its quantized version, as illustrated in Figure 2.

2.1 Intuition of Safe Quantization

Dai et al. (2023) developed Safe RLHF, a more safer version of RLHF (Bai et al., 2022; Ouyang et al., 2022), by decoupling *helpfulness* and *harmlessness* attributes of LMs. Following this intuition, we believe developing a safe quantization for LMs requires decoupling of their *understanding* and *safety* capabilities, and preserve them from full-precision LM by efficiently learning quantization parameters. Therefore, we design a learnable quantization algorithm that decouples *understanding* and *safety* capabilities using different calibration datasets, for achieving safe quantization that preserves safety alignment as shown in Figure 1(b).

2.2 Learnable Weight Clipping

We perform learnable weight quantization process by leveraging the learnable weight clipping (LWC) module (Shao et al., 2023), that computes optimal dynamic range for weights by optimizing a clipping threshold. Using LWC, we compute the quantized weights W_q as:

$$W_q = clamp\left(\lfloor\frac{W}{s}\rceil + p_z, 0, 2^N - 1\right) \text{ using } s = \frac{\alpha\ max(W) - \beta\ min(W)}{2^N - 1},\ p_z = -\lfloor\frac{\beta\ min(W)}{s}\rceil \tag{1}$$

where, $\lfloor\cdot\rceil$ is round-to-nearest operation, N is the target weight quantization bit number, W is the full-precision weights, s is the step size, p_z is the zero-point value, and $\alpha \in [0, 1]$ and $\beta \in [0, 1]$ are learnable quantization parameters Θ_{LWC} of LWC, for controlling the upper and lower bound of clipping range respectively.

2.3 Learnable Equivalent Transformation

We perform learnable activation quantization process by leveraging the learnable equivalent transformation (LET) module (Shao et al., 2023), that migrates the quantization of activations to weights with a mathematically equivalent transformation. Precisely, LET performs equivalent transformation on both linear layer and attention operation of transformer block.

Linear Layer. Using LET, we mathematically transform linear layer Y using input X (activation) and bias vector B as:

$$Y = XW + B = \tilde{X}\tilde{W} + \tilde{B} \text{ using } \tilde{X} = (X - \delta) \oslash \gamma,\ \tilde{W} = \gamma \odot W,\ \tilde{B} = B + \delta W \tag{2}$$

where $\oslash$ is element-wise division, $\odot$ is element-wise multiplication, $\tilde{X}$ is the equivalent activation, $\tilde{W}$ is the equivalent weight, $\tilde{B}$ is the equivalent bias vector, and γ and δ are scaling and shifting factors respectively, for controlling the activation distribution range.

Attention Operation. Using LET, we mathematically transform the attention mechanism's affinity matrix P using query matrix Q and key matrix K as:

$$P = softmax\left(Q \times K^T\right) = softmax\left(\tilde{Q} \times \tilde{K}^T\right) \quad \text{using} \quad \tilde{Q} = Q \oslash \varepsilon, \quad \tilde{K} = \varepsilon \odot K \tag{3}$$

where $\tilde{Q}$ is the equivalent query matrix, $\tilde{K}$ is the equivalent key matrix, and ε is the scaling factor for the affinity matrix. Collectively, scaling factors γ of linear layer, shifting factors δ of linear layer and scaling factors ε of attention operation are the learnable quantization parameters Θ_{LET} of LET.

2.4 Block-wise Quantization Loss Optimization

We use block-wise quantization loss optimization, a most commonly used learnable PTQ approach for LMs, where each transformer block of the LM is quantized one after the other. We preserve *understanding* (U) and *safety* (S) capabilities of LMs using $\mathcal{L}_U$ and $\mathcal{L}_S$ losses respectively, that quantifies the deviation between outputs of full-precision and quantized transformer block.

We define $\mathcal{L}_U$ loss as mean squared error on *understanding* capabilities calibration dataset $\mathcal{D}_U$, and $\mathcal{L}_S$ loss as negative logarithm of cosine similarity on *safety* capabilities calibration dataset $\mathcal{D}_S$, as below:

$$\mathcal{L}_U \mid_{\mathcal{D}_U} = \left\| \mathcal{F}\left(W, X\right) - \mathcal{F}(W_q \xleftarrow{\Theta_{LWC}} W, X_q \xleftarrow{\Theta_{LET}} X) \right\|_2^2 \tag{4}$$

$$\mathcal{L}_S \mid_{\mathcal{D}_S} = -log\left(\mathcal{C}\left(\mathcal{F}(W, X), \mathcal{F}(W_q \xleftarrow{\Theta_{LWC}} W, X_q \xleftarrow{\Theta_{LET}} X)\right)\right) \tag{5}$$

where $\mathcal{F}(\cdot)$ is mapping function for LM's transformer block, $\mathcal{C}(\cdot)$ is cosine similarity function, W is full-precision weights, W_q is quantized weights, X is full-precision activations, and X_q is quantized activations.

We formulate our final optimization objective as a weighted-sum of these $\mathcal{L}_U$ and $\mathcal{L}_S$ losses:

$$\min_{\Theta_{LWC}, \Theta_{LET}} (1 - \lambda_{SA})\mathcal{L}_U + \lambda_{SA}\mathcal{L}_S \tag{6}$$

where λ_{SA} is the importance factor for safety alignment in our quantization loss optimization, with 0.6 as default value.

3 Evaluation

In this section, we perform extensive evaluations on our ***SafeQ*** algorithm with the objectives of: (1) demonstrating *understanding* capabilities using pre-trained LMs (§ 3.2), (2) demonstrating *safety* and *general utility* capabilities using instruction-tuned LMs, along with other evaluations (§ 3.3), and (3) understanding the influence of various training parameters on the performance of *understanding* capabilities, and *safety* and *general utility* capabilities of LMs (§3.4).

3.1 Setup

Categorization. We categorize our evaluations based on the type of LM used, *i.e.*, pre-trained or instruction-tuned. In our experiments, we use pre-trained LMs for evaluating *understanding* capabilities, whereas instruction-tuned LMs are used for evaluating *safety* and *general utility* capabilities.

Evaluation Intuition. Our intuition in categorizing evaluations based on the type of LM is: (1) to show the effectiveness of our ***SafeQ*** algorithm as a general-purpose quantization algorithm

Table 1: PPL (↓) results of pre-trained LMs, *Llama-7B* and *Llama-2-7B*, on WikiText2 (WT2), C4 and PTB datasets for W4A16 and W4A4 quantization modes. (*bold* cells are best and *underlined* cells are second best)

W4A16

Method	*Llama-7B*			*Llama-2-7B*		
	WT2	**C4**	**PTB**	**WT2**	**C4**	**PTB**
FP16	5.68	7.08	29.26	5.47	6.97	22.51
AWQ	6.16	7.57	35.15	6.07	7.60	31.64
OmniQuant	**5.86**	7.34	**29.56**	5.73	7.35	48.81
DuQuant	5.88	**7.31**	30.41	**5.71**	**7.22**	**24.29**
SpinQuant	6.34	7.78	37.32	6.56	8.34	30.68
SafeQ	5.88	7.34	31.74	5.76	7.31	25.85

W4A4

Method	*Llama-7B*			*Llama-2-7B*		
	WT2	**C4**	**PTB**	**WT2**	**C4**	**PTB**
FP16	5.68	7.08	29.26	5.47	6.97	22.51
SmoothQ.	5.7e2	5.5e2	8.9e2	NaN	NaN	1.4e3
OmniQuant	13.97	17.75	1.1e2	20.58	25.63	1.1e3
DuQuant	**6.29**	**7.75**	**34.59**	**6.45**	**8.08**	**31.80**
SpinQuant	7.51	9.28	53.62	9.54	12.76	87.79
SafeQ	13.97	16.86	77.23	20.11	24.35	94.77

using pre-trained LMs, and (2) to show additional safety guarantees of our *SafeQ* algorithm using instruction-tuned LMs.

Quantization Modes. We performed both weight-only and weight-activation quantization on pre-trained LMs. However, due to the widespread use of instruction-tuned LMs in real-world AI use-cases, we choose to perform weight-only quantization, following current industry standards (Team et al., 2023). Going forward the quantization modes are denoted with shorter notations, like W4A16 for 4-bit weight-only quantization, and W4A4 for 4-bit weight and 4-bit activation quantization.

Models. We choose *Llama* (7B-65B) (Touvron et al., 2023a) and *Llama-2* (7B-70B) (Touvron et al., 2023b) families of LMs for our pre-trained LM evaluations, and for instruction-tuned LM evaluations, we choose *Llama-2-Chat* (7B-70B) (Touvron et al., 2023b), *Llama-3.1-Instruct* (8B-405B) (Grattafiori et al., 2024) and *Qwen2.5-Instruct* (0.5B-72B) (Yang et al., 2025) families of LMs. For all above mentioned LM families, we choose the largest LM variant with $<$10B parameters in our evaluations, *i.e.*, *Llama-7B* and *Llama-2-7B* as candidate pre-trained LMs and *Llama-2-7B-Chat*, *Llama-3.1-8B-Instruct* and *Qwen2.5-7B-Instruct* as candidate instruction-tuned LMs.

Training. The channel-wise scaling factor is initialized using logarithmic activation equalization (Li et al., 2023) method, and the channel-wise shifting factor is initialized using outlier shifting (Wei et al., 2023) method. We leverage AdamW optimizer with zero weight decay for optimizing the learnable quantization parameters Θ_{LWC} and Θ_{LET}, with $1e-2$ and $1e-3$ learning rates respectively. We utilize 128 randomly selected 2048 token segments from WikiText2 (Merity et al., 2016) as our *understanding* capabilities calibration data and from AdvBench (Zou et al., 2023) as our *safety* capabilities calibration data. We perform our training on a single NVIDIA RTX 4500 ADA 24GB GPU, using a batch size of 1 over 20 epochs. For weight-only quantization, only LWC is enabled, whereas for weight-activation quantization, both LWC and LET are enabled.

Baselines. We compare our *SafeQ* algorithm against state-of-the-art PTQ algorithms:

- ***Standard PTQ Algorithms.*** AWQ (Lin et al., 2024b), OmniQuant (Shao et al., 2023) and SmoothQuant (Xiao et al., 2023).
- ***Rotation-based PTQ Algorithms.*** DuQuant (Lin et al., 2024a) and SpinQuant (Liu et al., 2025).

Resources for Replication. Code for replicating our results in provided in the *Artifacts* page of our project webpage[1].

Additional details related to the experimental setup are elaborated in Appendix C.

3.2 Quantization of Pre-trained LMs

3.2.1 Language Understanding Evaluations

The *understanding* capabilities of pre-trained LMs are quantified using perplexity (PPL) metric, computed on WikiText2, C4 (Raffel et al., 2020) and Penn Treebank (PTB) (Marcus et al., 1994) datasets.

[1] https://sites.google.com/view/safe-quantization

Table 2: Average ACC (↑) results of instruction-tuned LMs, *Llama-2-7B-Chat*, *Llama-3.1-8B-Instruct* and *Qwen2.5-7B-Instruct*, on *safety* and *general utility* benchmarks for W4A16 quantization mode (ACC_S and ACC_{GU} respectively). Individual ACC results on these benchmarks are presented in Tables 4 and 5.

Method	*Llama-2-7B-Chat*		*Llama-3.1-8B-Instruct*		*Qwen2.5-7B-Instruct*	
	ACC_S	ACC_{GU}	ACC_S	ACC_{GU}	ACC_S	ACC_{GU}
FP16	56.72	63.40	77.67	70.07	77.79	69.60
AWQ	53.33	60.75	71.24	67.49	80.74	**70.10**
OmniQuant	56.47	61.64	72.67	67.88	81.73	69.75
DuQuant	55.81	**62.61**	73.50	68.24	-	-
SpinQuant	49.98	57.03	73.39	68.31	79.01	69.19
SafeQ	**57.74**	62.27	**74.43**	**68.36**	**81.77**	69.37

Perplexity Results. Table 1 presents the PPL results of the pre-trained LMs, *Llama-7B* and *Llama-2-7B* for W4A16 and W4A4 quantization modes, comparing *SafeQ* with baseline PTQ algorithms. We expected a slight trade-off in *understanding* capabilities due to the addition of loss $\mathcal{L}_S$ in Equation 6, that specifically contributes to safety alignment preservation. On contrary, *SafeQ* achieves second best performances on W4A16 quantization of *Llama-7B* and *Llama-2-7B*, outperforming rotation-based PTQ baselines in some cases. However, in W4A4 quantization of *Llama-7B* and *Llama-2-7B*, falls behind rotation-based PTQ baselines, DuQuant and SpinQuant, but outperforms standard PTQ baselines, OmniQuant and SmoothQuant. These results emphasize the effectiveness of our *SafeQ* as a general-purpose quantization algorithm that ensures better *understanding* capabilities for weight-activation quantization, but indicates an improvement gap for weight-only quantization.

3.3 Quantization of Instruction-tuned LMs

3.3.1 Benchmark Evaluations

The instruction-tuned LMs are evaluated on zero-shot tasks of *safety* benchmark SafetyBench (Zhang et al., 2024) and *general utility* benchmarks, *i.e.*, ARC (Clark et al., 2018), BoolQ (Clark et al., 2019), CommonSenseQA (Talmor et al., 2019), HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020), SocialIQA (Sap et al., 2019), and WinoGrande (Sakaguchi et al., 2021). All these benchmarks are quantified using accuracy (ACC) metric. Additionally, we evaluate the *instruction-following* capabilities of these LMs on MT-Bench (Zheng et al., 2023) using GPT-5.6-Terra (OpenAI, 2026) as judge.

Zero-Shot Tasks Results. Table 2 presents the average ACC results of W4A16 quantized instruction-tuned LMs, *Llama-2-7B-Chat*, *Llama-3.1-8B-Instruct* and *Qwen2.5-7B-Instruct*, on *safety* and *general utility* benchmarks using ACC_S and ACC_{GU} respectively. The complete results on individual safety dimensions and general utility benchmarks are presented in Tables 4 and 5 of Appendix D.1.

As expected, *SafeQ* achieves high ACC_S for all evaluated LMs. Precisely, *SafeQ* outperforms AWQ by +4.42%, OmniQuant by +1.28%, DuQuant by +1.93% and SpinQuant by +7.77% for *Llama-2-7B-Chat* in terms of ACC_S. Further, *SafeQ* remarkably improved the safety of *Llama-2-7B-Chat* and *Qwen2.5-7B-Instruct* beyond FP16 by +1.03% and +3.99% respectively. Additionally, *SafeQ* outperforms recent rotation-based PTQ baselines, DuQuant and SpinQuant, in terms of ACC_S, for both *Llama-2-7B-Chat* and *Llama-3.1-8B-Instruct*. These results emphasize the higher safety guarantees of our *SafeQ* algorithm.

Similar to *understanding* capabilities, we expected a slight trade-off for *general utility* capabilities in instruction-tuned LMs, but *SafeQ* outperformed all baselines for *Llama-3.1-8B-Instruct*, and is the second best performing on *Llama-2-7B-Chat* in terms of ACC_{GU}. Further, *SafeQ* remarkably improved utility of *Llama-2-7B-Chat* and *Qwen2.5-7B-Instruct* than SpinQuant, a rotation-based PTQ from industry. We strongly feel that pulling LMs towards safety alignment during quantization, must have made them reason more safely that resulted in these elevated *general utility* capabilities. Overall, the results of Table 2 made it evident that *SafeQ* resolves the trade-off between *safety* and *general utility* to a great extent.

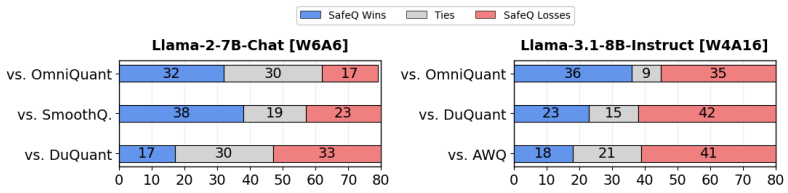
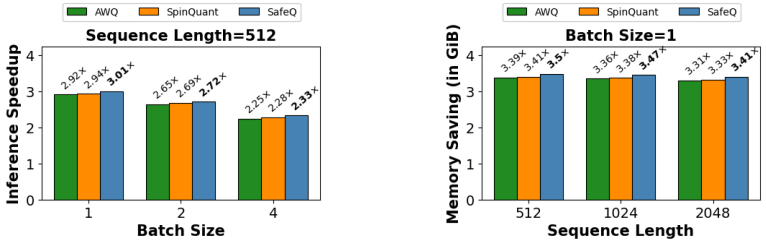


Figure 3: Comparing *SafeQ* against baseline PTQ algorithms on MT-Bench for quantized *Llama-2-7B-Chat* (W6A6) and *Llama-3.1-8B-Instruct* (W4A16) models using GPT-5.6-Terra model as judge.



(a) (b)

Figure 4: Real-time end-to-end (a) inference speedup for different batch sizes and (b) memory savings for different sequence lengths, of *Llama-2-7B-Chat* (W4A16) decoding 128 tokens on NVIDIA RTX 5060 GPU. Detailed results of inference speedups and memory savings at different stages are presented in Table 6 and 7 respectively.

MT-Bench Results. Figure 3 illustrates the MT-Bench results evaluated by GPT-5.6-Terra judge, comparing *SafeQ* against industry popular PTQs, namely OmniQuant, AWQ, SmoothQuant and DuQuant for *Llama-2-7B-Chat* (W6A6) and *Llama-3.1-8B-Instruct* (W4A16).

Contrary to our setup (in §3.1), in this experiment we performed W6A6 quantization on *Llama-2-7B-Chat* for understanding the impact of arbitrary-bit quantization on *instruction-following* capabilities. The results in Figure 3 (left) shows that *SafeQ* preserves these capabilities well and outperforms OmniQuant and SmoothQuant with 59.49% and 59.38% win rates respectively. However, similar to PPL results (in 1, DuQuant beats *SafeQ* in weight-activation quantization with a win rate of 60%.

The results of *Llama-3.1-8B-Instruct* (W4A16) in Figure 3 (right) demonstrates that *SafeQ* slightly outperform OmniQuant with 50.62% win rate, but fails against AWQ and DuQuant with 35.62% and 38.12% win rates respectively. Although *SafeQ* has high *general utility* capabilities than AWQ and DuQuant (in Table 2) for *Llama-3.1-8B-Instruct* (W4A16), it did not guarantee high *instruction-following* capabilities, and illustrates how decoupled *general utility* and *instruction-following* capabilities are in LMs.

3.3.2 Efficiency Evaluations

We evaluate the performance of real-time inference speedup and memory savings on instruction-tuned LM using vLLM (Kwon et al., 2023) kernels compared to full-precision LM for decoding 128 tokens.

End-to-End Inference Speedup Results. Figure 4(a) presents the end-to-end inference speedup for different batch sizes on *Llama-2-7B-Chat* (W4A16) for a sequence length of 512, comparing *SafeQ* with AWQ and SpinQuant. The complete results on inference speedup on end-to-end runs, and individual prefill and decoding stages are presented in Table 6 of Appendix D.2.

As expected, for all tested quantization algorithms with increase in batch size, the inference speedup of end-to-end runs decreases substantially. However, the inference speedup of *SafeQ* is higher than AWQ and SpinQuant on end-to-end runs, and achieves 2.33×–3.01× speedup over the full-precision *Llama-2-7B-Chat*.

End-to-End Memory Saving Results. The results in Figure 4(b) illustrates the end-to-end memory savings for different sequence lengths on *Llama-2-7B-Chat* (W4A16) for a batch size of 1, and compares *SafeQ* with AWQ and SpinQuant. The complete results on memory savings on end-to-end runs, and individual prefill and decoding stages are presented in Table 7 of Appendix D.2.

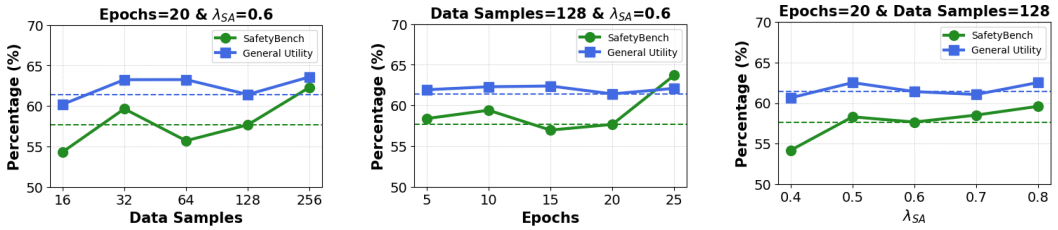
Table 3: PPL (↓) results of *Llama-2-7B* (W3A16 and W4A4), for *SafeQ* ablations of ***calibration data*** (left) and ***training time*** (right) on WikiText2 (WT2), C4 and PTB datasets. (Blue cells are default training parameters)

Ablations: Calibration Data

Data Samples	W3A16			W4A4		
	WT2	C4	PTB	WT2	C4	PTB
16	7.68	9.69	34.56	18.17	20.92	118.66
32	6.74	8.64	27.64	17.32	20.72	79.44
64	6.61	8.45	32.23	15.78	20.31	123.92
128	6.62	8.46	26.79	20.11	24.35	94.77
256	6.62	8.48	26.95	19.6	22.47	173.01

Ablations: Training Time

Epochs	W3A16			W4A4		
	WT2	C4	PTB	WT2	C4	PTB
5	6.76	8.62	29.32	19.5	23.74	138.42
10	6.64	8.48	27.15	16.25	19.56	85.57
15	6.64	8.47	25.97	28.41	29.71	112.36
20	6.62	8.46	26.79	20.11	24.35	94.77
25	6.62	8.49	27.43	35.64	46.1	117.28



(a) (b) (c)

Figure 5: Average ACC (↑) variations of *Llama-3.1-8B-Instruct* (W3A16) in *safety* and *general utility* benchmarks for *SafeQ* ablations of (a) calibration data, (b) training times and (c) safety alignment importance factors. Individual ACC results on these benchmarks are presented in Tables 8-13. (Dashed lines represent the average ACC under default training parameters)

For all sequence lengths, *SafeQ* gets 3.41×–3.5× memory savings compared to full-precision version of *Llama-2-7B-Chat* on end-to-end runs. Further, the memory savings of *SafeQ* is higher than AWQ and SpinQuant on end-to-end runs.

3.4 Ablation Studies

We performed ablation studies on pre-trained LM, *Llama-2-7B* (W3A16 and W4A4), and instruction-tuned LM, *Llama-3.1-8B-Instruct* (W3A16), with varying calibration data samples and training epochs. Further, we studied the influence of safety alignment importance factor λ_{SA} on *Llama-3.1-8B-Instruct* (W3A16).

Influence of Calibration Data. Table 3 (left) presents the PPL results of *Llama-2-7B* and Figure 5(a) presents the *safety* and *general utility* average ACC results of *Llama-3.1-8B-Instruct*, for calibration data ablations. Although, average ACC results suggest 256 data samples is better than 128 data samples for calibration data, we strongly feel 32 data samples is the best, considering its lower data for calibration and better PPL and average ACC performance than 128 data samples. Further, 32 data samples PPL and average ACC performance is better or comparable to 256 data samples performance.

Influence of Training Time. Table 3 (right) presents the PPL results of *Llama-2-7B* and Figure 5(b) presents the *safety* and *general utility* average ACC results of *Llama-3.1-8B-Instruct*, for training time ablations. As expected, higher epochs achieve better average ACC results, but we feel that 10 epochs is the best training time as it outperforms 20 epochs in terms of both PPL and average ACC, as well as highly reduces quantization time. Moreover, the results illustrate that 10 epochs has either better or comparable PPL and average ACC performance with 25 epochs training time.

Both calibration data and training time ablations suggest that better results could be achieved by *SafeQ* with minimal data and lower training time, than we used in our main evaluations. It is noteworthy that the parameters used in our main evaluations are standard practices in quantization literature (Lin et al., 2024b; Shao et al., 2023).

Influence of Safety Alignment Importance Factor (λ_{SA}). Figure 5(c) presents the *safety* and *general utility* average ACC results of *Llama-3.1-8B-Instruct* for different λ_{AS} ablations. With increasing λ_{SA} values, better *safety* and *general utility* average ACC results are observed, with an

exception of $\lambda_{SA} = 0.5$. It is observed that performance of $\lambda_{SA} = 0.5$ is on par with performance of $\lambda_{SA} = 0.7$ in terms of safety. This illustrates that equal importance to *safety* and *understanding* capabilities would yield near-optimal safety performance for our *SafeQ* algorithm.

The complete results on individual safety dimensions and general utility benchmarks for all ablations are presented in Tables 8-13 of Appendix D.3.

4 DISCUSSION

Versatility, Effectiveness and Efficiency of *SafeQ*. *SafeQ* supports a diverse range of quantization modes for both pre-trained and instruction-tuned LMs, including weight-only quantization (W3A16 and W4A16) and weight-activation quantization (W4A4 and W6A6), with W2A16 and W8A8 also feasible. As demonstrated in §3.2, §3.3, and §3.4, *SafeQ* effectively preserves the *understanding* capabilities of pre-trained LMs as well as the *safety* and *general utility* of instruction-tuned LMs. Further, comparisons with state-of-the-art PTQ methods emphasizes its efficiency in maintaining *safety* and *general utility* for instruction-tuned LMs.

Comparison between *SafeQ* and AAQ. The goal of our *SafeQ* and AAQ (Wee et al., 2025) (previously CAQ) are same, *i.e.*, preserving safety alignment of LMs while performing quantization. But, they fundamentally differ in the methodology of quantization process followed. The weighted-sum quantization loss of *SafeQ* learns both *understanding* and *safety* capabilities using different calibration datasets, whereas the contrastive loss of AAQ optimizes a lightweight set of transformation parameters using a standard calibration dataset by keeping the LM weights frozen. However, AAQ's contrastive loss requires loading both pre-trained and instruction-tuned versions of LM to generate a quantized LM, whereas *SafeQ* generates quantized LM by loading either pre-trained or instruction-tuned version of LM, based on the user's interest. Despite *SafeQ* and AAQ being designed for same goal, we were unable to use AAQ as baseline because the authors did ***not*** open-source their code with research community.

Why NOT use LLMs in Evaluations? As scaling laws (Kaplan et al., 2020) apply to all LMs (both full-precision and quantized versions), we strongly believe that if *SafeQ* can offer substantial safety guarantees in SLMs, then it would achieve the same with LLMs of same LM family, as *Llama-3.1-8B-Instruct* and *Llama-3.1-405B-Instruct* only differ by number of transformer blocks.

Why NOT use Q-reSafe as Baseline? As discussed earlier, Q-reSafe (Chen et al., 2025) is a post-hoc patching framework for restoring safety alignment of quantized LMs, and is ***not*** a PTQ to compare against our *SafeQ*. Thus, we excluded this framework as a baseline against our *SafeQ*.

Limitations. We assume that the full-precision LMs are highly safety aligned, and only while quantization some LMs either loose safety alignment (Nakka et al., 2024) or disproportionately have degraded alignment (Kharinaev et al., 2025). Our *SafeQ* aims to preserve the safety alignment in LMs from full-precision while performing quantization. However, if the full-precision LM itself is not safety aligned, then the resultant *SafeQ* quantized LM would not have safety alignment preserved and could generate unsafe content.

Future Work. The results in §3 illustrate that *SafeQ* achieves best/near-best performances on weight-only quantization compared to all baseline PTQ algorithms. However, in weight-activation quantization, *SafeQ* comes next to rotation-based PTQ algorithms while outperforming standard PTQ algorithms. As a future direction, we want to improve the LET module of *SafeQ* to match the level of performance observed in rotation-based PTQ algorithms for weight-activation quantization.

5 CONCLUSION

In this paper, we develop a novel safety alignment preserving PTQ, called *SafeQ*, that generates quantized LMs with higher safety guarantees. By integrating calibration data of *safety* capabilities, *SafeQ* effectively maintains comparable *understanding* capabilities along with higher *safety* and *general utility* capabilities in LMs. Further, *SafeQ* evidently resolves the trade-off between *safety* and *general utility* to a great extent in instruction-tuned LMs. Altogether, we strongly feel this advancement by *SafeQ* ensures efficient deployment of safe-quantized LMs in resource-constrained environments like real-world personal and edge devices.

AI USE STATEMENT

In this work, we did not use generative AI tools for any of the following required-disclosure tasks: generating synthetic datasets, developing theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, implementing methods, cleaning and reformatting datasets, or interpreting results. Assisting with translation and supporting qualitative and thematic data analysis are not applicable to this work.

Additionally, we used generative AI tools for editing a research paper to improve readability and for creating or editing software code, specifically to generate helper functions for post-processing result files and producing bar and line plots used as figures in the paper. We have reviewed all AI-assisted work: all AI-edited text was checked by the authors against the intended technical content and claims to ensure accuracy was preserved, and all AI-generated helper code for result processing and plotting was manually inspected and run by the authors against the underlying result files to verify that the outputs (aggregated tables and figures) correctly and faithfully reflect the experimental data. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We have taken several steps to ensure the reproducibility of our results. The full implementation of *SafeQ*, including data preprocessing (Step 0), the quantization pipeline (Step 1), evaluation harness (Step 2), and result aggregation and visualization scripts (Step 3-4), is provided as anonymized supplementary code, together with a README documenting the environment setup, dependencies, and exact commands used to reproduce every reported configuration (weight-only W4A16 and weight-activation W4A4/W6A6). The baseline quantization methods we compare against (AWQ, SmoothQuant, OmniQuant, SpinQuant) are implemented in the same codebase with matching evaluation scripts, whereas for DuQuant we leveraged their publicly shared code. Details of the models evaluated (*Llama*, *Llama-2*, *Llama-3.1*, *Qwen-2.5* families, pretrained and instruction-tuned variants), the calibration and evaluation datasets (including C4, safety-alignment, adversarial-robustness, and instruction-following/judged benchmarks), and all hyperparameters (bit-widths, calibration epochs, LWC/LET settings) are specified in Section 3 and Appendix C.

REFERENCES

Adobe Firefly. https://www.adobe.com/products/firefly/features/text-to-image.html, 2024. [Accessed 23-05-2024].

Understanding the True Cost of Large Language Models. https://charshift.com/llm-true-cost/, 2024a. [Accessed 23-05-2024].

ChatGPT. https://chatgpt.com/, 2024b. [Accessed 23-05-2024].

The Big Story of AI in 2023: LLMs. https://www.embedl.com/knowledge/the-big-story-of-ai-in-2023-llms, 2024. [Accessed 23-05-2024].

GitHub Copilot · Your AI pair programmer. https://github.com/features/copilot, 2024a. [Accessed 23-05-2024].

GitHub - thu-coai/SafetyBench: Official github repo for SafetyBench, a comprehensive benchmark to evaluate LLMs' safety. [ACL 2024] — github.com. https://github.com/thu-coai/{S}afety{B}ench, 2024b. [Accessed 05-05-2026].

Gemini - chat to supercharge your ideas. https://gemini.google.com/app, 2024. [Accessed 23-05-2024].

The Emergence of Small Language Models. https://www.nocode.ai/the-emergence-of-small-language-models/, 2024. [Accessed 01-06-2024].

Red Hat AI and vLLM Project. LLM Compressor, 8 2024. URL https://github.com/vllm-project/llm-compressor.

Mohammad Sadegh Akhondzadeh, Aleksandar Bojchevski, Evangelos Eleftheriou, and Martino Dazzi. Kurtail: Kurtosis-based llm quantization. *arXiv preprint arXiv:2503.01483*, 2025.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

Michiel Bakker, Martin Chadwick, Hannah Sheahan, Michael Tessler, Lucy Campbell-Gillingham, Jan Balaguer, Nat McAleese, Amelia Glaese, John Aslanides, Matt Botvinick, et al. Fine-tuning language models to find agreement among humans with diverse preferences. *Advances in Neural Information Processing Systems*, 35:38176–38189, 2022.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, 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.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher M De Sa. Quip: 2-bit quantization of large language models with guarantees. *Advances in neural information processing systems*, 36: 4396–4429, 2023.

Kejia Chen, Jiawen Zhang, Jiacong Hu, Yu Wang, Jian Lou, Zunlei Feng, and Mingli Song. Assessing safety risks and quantization-aware safety patching for quantized large language models. In *Forty-second International Conference on Machine Learning*, 2025.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)*, pp. 2924–2936, 2019.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Josef Dai, Xuehai Pan, Ruiyang Sun, Jiaming Ji, Xinbo Xu, Mickel Liu, Yizhou Wang, and Yaodong Yang. Safe rlhf: Safe reinforcement learning from human feedback. *arXiv preprint arXiv:2310.12773*, 2023.

Tim Dettmers and Luke Zettlemoyer. The case for 4-bit precision: k-bit inference scaling laws. In *International Conference on Machine Learning*, pp. 7750–7774. PMLR, 2023.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in neural information processing systems*, 35: 30318–30332, 2022.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Suyu Ge, Chunting Zhou, Rui Hou, Madian Khabsa, Yi-Chia Wang, Qifan Wang, Jiawei Han, and Yuning Mao. Mart: Improving llm safety with multi-round automatic red-teaming. *arXiv preprint arXiv:2311.07689*, 2023.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Ziyi Guan, Hantao Huang, Yupeng Su, Hong Huang, Ngai Wong, and Hao Yu. Aptq: Attention-aware post-training mixed-precision quantization for large language models. In *Proceedings of the 61st ACM/IEEE Design Automation Conference*, pp. 1–6, 2024.

Manav Kapadnis, Sohan Patnaik, Abhilash Nandy, Sourjyadip Ray, Pawan Goyal, and Debdoot Sheet. Serpent-vlm: self-refining radiology report generation using vision language models. In *Proceedings of the 6th Clinical Natural Language Processing Workshop*, pp. 283–291, 2024.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Artyom Kharinaev, Viktor Moskvoretskii, Egor Shvetsov, Kseniia Studenikina, Bykov Mikhail, and Evgeny Burnaev. Investigating the impact of quantization methods on the safety and reliability of large language models. *arXiv preprint arXiv:2502.15799*, 2025.

Young Jin Kim, Rawn Henry, Raffy Fahim, and Hany Hassan Awadalla. Finequant: Unlocking efficiency with fine-grained weight-only quantization for llms. *arXiv preprint arXiv:2308.09723*, 2023.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Qingyuan Li, Yifan Zhang, Liang Li, Peng Yao, Bo Zhang, Xiangxiang Chu, Yerui Sun, Li Du, and Yuchen Xie. Fptq: Fine-grained post-training quantization for large language models. *arXiv preprint arXiv:2308.15987*, 2023.

Xingang Li, Yuewan Sun, and Zhenghui Sha. Llm4cad: Multimodal large language models for three-dimensional computer-aided design generation. *Journal of Computing and Information Science in Engineering*, 25(2):021005, 2025.

Haokun Lin, Haobo Xu, Yichen Wu, Jingzhi Cui, Yingtao Zhang, Linzhan Mou, Linqi Song, Zhenan Sun, and Ying Wei. Duquant: Distributing outliers via dual transformation makes stronger quantized llms. *Advances in Neural Information Processing Systems*, 37:87766–87800, 2024a.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of machine learning and systems*, 6: 87–100, 2024b.

Jing Liu, Ruihao Gong, Xiuying Wei, Zhiwei Dong, Jianfei Cai, and Bohan Zhuang. Qllm: Accurate and efficient low-bitwidth quantization for large language models. *arXiv preprint arXiv:2310.08041*, 2023.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. Spinquant: Llm quantization with learned rotations. In *International Conference on Learning Representations*, volume 2025, pp. 92009–92032, 2025.

Mitch Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. The penn treebank: Annotating predicate argument structure. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11*, 1994.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Pranav Nair, Puranjay Datta, Jeff Dean, Prateek Jain, and Aditya Kusupati. Matryoshka quantization. *arXiv preprint arXiv:2502.06786*, 2025.

Kalyan Nakka, Jimmy Dani, and Nitesh Saxena. Is On-Device AI Broken and Exploitable? Assessing the Trust and Ethics in Small Language Models. *arXiv preprint arXiv:2406.05364*, 2024.

Kalyan Nakka, Jimmy Dani, Ausmit Mondal, and Nitesh Saxena. Litelmguard: Seamless and lightweight on-device guardrails for small language models against quantization vulnerabilities. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pp. 206–223, 2025.

OpenAI. Gpt-5.6 terra api. OpenAI API Documentation, 2026. URL https://developers.openai.com/api/docs/models/gpt-5.6-terra.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Gunho Park, Minsub Kim, Sungjae Lee, Jeonghoon Kim, Beomseok Kwon, Se Jung Kwon, Byeongwook Kim, Youngjoo Lee, Dongsoo Lee, et al. Lut-gemm: Quantized matrix multiplication based on luts for efficient inference in large-scale generative language models. In *The Twelfth International Conference on Learning Representations*, 2023.

Jonggwon Park, Soobum Kim, Byungmu Yoon, Jihun Hyun, and Kyoyun Choi. M4cxr: exploring multitask potentials of multimodal large language models for chest x-ray interpretation. *IEEE Transactions on Neural Networks and Learning Systems*, 2025.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Muhammad Raza. LLMs vs. SLMs: The Differences in Large & Small Language Models. https://www.splunk.com/en_us/blog/learn/language-models-slm-vs-llm, 2023. [Accessed 04-05-2026].

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social iqa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 4463–4473, 2019.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. Omniquant: Omnidirectionally calibrated quantization for large language models. *arXiv preprint arXiv:2308.13137*, 2023.

Yuxuan Sun, Ruikang Liu, Haoli Bai, Han Bao, Kang Zhao, Yuening Li, Jiaxin Hu, Xianzhi Yu, Lu Hou, Chun Yuan, et al. Flatquant: Flatness matters for llm quantization. *arXiv preprint arXiv:2410.09426*, 2024.

Lintang Sutawika, Leo Gao, Hailey Schoelkopf, Stella Biderman, Jonathan Tow, Baber Abbasi, ben fattori, Charles Lovering, farzanehnakhaee70, Jason Phang, Anish Thite, Fazz, Aflah, Niklas Muennighoff, Thomas Wang, sdtblck, nopperl, gakada, tttyuntian, researcher2, Chris, Julen Etxaniz, Zdeněk Kasner, Khalid, Jeffrey Hsu, AndyZwei, Pawan Sasanka Ammanamanchi, Dirk Groeneveld, Ethan Smith, and Eric Tang. Eleutherai/lm-evaluation-harness: Major refactor, December 2023. URL https://doi.org/10.5281/zenodo.10256836.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4149–4158, 2019.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Omkar Chakradhar Thawakar, Abdelrahman M Shaker, Sahal Shaji Mullappilly, Hisham Cholakkal, Rao Muhammad Anwer, Salman Khan, Jorma Laaksonen, and Fahad Khan. Xraygpt: Chest radiographs summarization using large medical vision-language models. In *Proceedings of the 23rd workshop on biomedical natural language processing*, pp. 440–448, 2024.

Dragi Tiro. The possibility of applying chatgpt (ai) for calculations in mechanical engineering. In *International Conference "New Technologies, Development and Applications"*, pp. 313–320. Springer, 2023.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

Sunghyun Wee, Suyoung Kim, Hyeonjin Kim, Kyomin Hwang, and Nojun Kwak. Alignment-aware quantization for llm safety. *arXiv e-prints*, pp. arXiv–2511, 2025.

Xiuying Wei, Yunchen Zhang, Yuhang Li, Xiangguo Zhang, Ruihao Gong, Jinyang Guo, and Xianglong Liu. Outlier suppression+: Accurate quantization of large language models by equivalent and effective shifting and scaling. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 1648–1665, 2023.

Jingyang Xiang and Sai Qian Zhang. Dfrot: Achieving outlier-free and massive activation-free for rotated llms with refined rotation. *arXiv preprint arXiv:2412.00648*, 2024.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International conference on machine learning*, pp. 38087–38099. PMLR, 2023.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL `https://arxiv.org/abs/2412.15115`.

Zhihang Yuan, Lin Niu, Jiawei Liu, Wenyu Liu, Xinggang Wang, Yuzhang Shang, Guangyu Sun, Qiang Wu, Jiaxiang Wu, and Bingzhe Wu. Rptq: Reorder-based post-training quantization for large language models. *arXiv preprint arXiv:2304.01089*, 2023.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

Tianyi Zhang and Anshumali Shrivastava. Leanquant: Accurate and scalable large language model quantization with loss-error-aware grid. *arXiv preprint arXiv:2407.10032*, 2024.

Zhexin Zhang, Leqi Lei, Lindong Wu, Rui Sun, Yongkang Huang, Chong Long, Xiao Liu, Xuanyu Lei, Jie Tang, and Minlie Huang. Safetybench: Evaluating the safety of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15537–15553, 2024.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023.

A *SafeQ* ALGORITHM

Algorithm 1 illustrates the comprehensive training procedure of our *SafeQ* quantization algorithm, that generates quantized LM $\mathcal{M}_Q$ using NLU calibration dataset $\mathcal{D}_{NLU}$, SA calibration dataset $\mathcal{D}_{SA}$, safety alignment importance factor λ_{SA}, and a full-precision LM $\mathcal{M}_{FP}$.

Algorithm 1: *SafeQ* Algorithm

Input: NLU calibration dataset $\mathcal{D}_{NLU}$, SA calibration dataset $\mathcal{D}_{SA}$, Safety Alignment Importance Factor λ_{SA}, Full-Precision LM $\mathcal{M}_{FP}$

Output: Quantized LM $\mathcal{M}_Q$

Begin
- **Init** $X_{fp}^{NLU} = X_q^{NLU} = \mathcal{D}_{NLU}$ `/* NLU inputs of FP and quantized LMs */`
- **Init** $X_{fp}^{SA} = X_q^{SA} = \mathcal{D}_{SA}$ `/* SA inputs of FP and quantized LMs */`
- **For** *Transformer Block* $\mathcal{B}_i$ *in* $\mathcal{M}_{FP}$ **Do**
  - $X_{fp}^{NLU} = \mathcal{B}_i(X_{fp}^{NLU})$ `/* update the NLU inputs of FP LM */`
  - $X_{fp}^{SA} = \mathcal{B}_i(X_{fp}^{SA})$ `/* update the SA inputs of FP LM */`
  - **Init** Θ_{LWC} `/* learnable quantization parameters of LWC */`
  - **Init** Θ_{LET} `/* learnable quantization parameters of LET */`
  - **For** *k in Epochs* **Do**
    - **For** $\left((x_q^{NLU}, x_{fp}^{NLU}), (x_q^{SA}, x_{fp}^{SA})\right)$ *in* $\left([X_q^{NLU}, X_{fp}^{NLU}], [X_q^{SA}, X_{fp}^{SA}]\right)$ **Do**
      - $\mathcal{B}_i^{'} = LET(\mathcal{B}_i, \Theta_{LET})$ `/* using equations 2 and 3 */`
      - $\mathcal{B}_i^{'} = LWC(\mathcal{B}_i^{'}, \Theta_{LWC})$ `/* using equation 1 */`
      - $x_q^{NLU'} = \mathcal{B}_i^{'}(x_q^{NLU})$
      - $x_q^{SA'} = \mathcal{B}_i^{'}(x_q^{SA})$
      - $\mathcal{L}_{NLU} = \left\| x_{fp}^{NLU} - x_q^{NLU'} \right\|^2$ `/* using equation 4 */`
      - $\mathcal{L}_{SA} = -log\left(\mathcal{C}\left(x_{fp}^{SA}, x_q^{SA'}\right)\right)$ `/* using equation 5 */`
      - $\mathcal{L} = (1 - \lambda_{SA})\mathcal{L}_{NLU} + \lambda_{SA}\mathcal{L}_{SA}$ `/* using equation 6 */`
      - $BackPropagate(\Theta_{LWC}, \Theta_{LET}, \mathcal{L})$ `/* update` Θ_{LWC} `and` Θ_{LET} `*/`
  - $\mathcal{B}_i = LET(\mathcal{B}_i, \Theta_{LET})$
  - $\mathcal{B}_i = LWC(\mathcal{B}_i, \Theta_{LWC})$ `/* obtain the quantized transformer block */`
  - $X_q^{NLU} = \mathcal{B}_i(X_q^{NLU})$ `/* update the NLU inputs of quantized LM */`
  - $X_q^{SA} = \mathcal{B}_i(X_q^{SA})$ `/* update the SA inputs of quantized LM */`
- $\mathcal{M}_Q = \mathcal{M}_{FP}$ `/* set the in-place quantized FP LM as quantized LM */`

Return $\mathcal{M}_Q$

B RELATED WORKS

Post-Training Quantization Methods. Lin et al. (2024b) developed AWQ that identifies a small fraction of salient weights based on activation magnitudes and protects them through per-channel scaling. Frantar et al. (2022) proposed a one-shot PTQ method, called GPTQ, that approximates second-order information to compress LMs to 3 or 4 bits with negligible loss in accuracy. Dettmers et al. (2022) combined vector-wise quantization for majority of features with a mixed-precision decomposition, and developed LLM.int8() quantization that isolates sparse, high magnitude outlier dimensions into 16-bit matrix multiplications.

Shao et al. (2023) first proposed OmniQuant, a learnable quantization technique, that retains NLU capabilities of full-precision LM while quantization using block-wise MSE error minimization. Xiao et al. (2023) developed a weight-activation quantization, called SmoothQuant, that migrates outlier induced difficulty from activations to weights through a mathematically equivalent per-channel scaling transformation. Chee et al. (2023) developed the first viable 2 bit quantization, called QuIP, for LMs by combining optimal adaptive rounding procedure for minimizing a quadratic proxy objective using random orthogonal matrices to ensure weight and Hessian incoherence. Guan et al. (2024) implemented a Hessian trace-driven sensitivity based quantization approach, that integrates Hessian information with attention-based gradients within the entire attention block.

Kim et al. (2023) proposed a weight quantization method, called FineQuant, that uses a heuristic to adaptively determine the optimal granularity for different weight matrices, effectively preventing performance collapse without requiring additional fine-tuning or calibration data. Sun et al. (2024) utilizes learnable affine transformations and Kronecker product decomposition to flatten weight and activation distributions, for significantly reducing quantization error while maintaining high inference efficiency. Zhang & Shrivastava (2024) improved LM accuracy by learning loss-error-aware grids that minimize task loss degradation caused by outliers in inverse Hessian diagonals. Wee et al. (2025) developed a push-pull optimization-based quantization that pulls the quantized LM's weight distribution toward the safe and instruction-tuned target LM, while pushing it away from the unaligned, pre-trained reference LM.

Park et al. (2023) developed an efficient kernel that eliminates de-quantization overhead by replacing redundant matrix multiplication calculations with look-up table (LUT) retrieval operations. Yuan et al. (2023) proposed clusters activation channels with similar value ranges and later reorders them to apply cluster-specific quantization parameters, effectively mitigating the impact of range differences across channels. Liu et al. (2023) proposed an adaptive channel reassembly technique that disassembles outlier channels into sub-channels and merges similar ones to suppress activation outliers while maintaining the original channel count for efficiency. Nair et al. (2025) developed a multi-scale training technique that jointly optimizes a single LM across multiple precisions by leveraging the nested structure of integer data types, where lower-bit representations are contained within the most significant bits of higher-bit ones.

Recent approaches learn rotation matrices for quantizing LMs. Liu et al. (2025) incorporated learned rotation matrices optimized on the Stiefel manifold to mitigate outliers in both weights and activations. Lin et al. (2024a) combined block-wise rotation with "zigzag" permutation to distribute massive outliers across channels and blocks. Akhondzadeh et al. (2025) used Kurtosis-based learnable rotations to reduce activation outliers that efficiently handled outliers than random rotations. Xiang & Zhang (2024) refined rotations through alternating optimization for making activations more conducive, especially for tokens with massive activations.

Securing against Safety Alignment Degradation. In order to restore safety alignment in LMs, Chen et al. (2025) proposed a safety-patching framework, called Q-reSafe, that selectively update a fraction of safety-critical weights. However, for protecting LMs from quantization-induced safety vulnerabilities, Nakka et al. (2025) developed a lightweight on-device guardrails, called LiteLMGuard, that effectively secures against intentional harmful querying and jailbreak attacks on personal and edge devices.

C Setup Details

C.1 Benchmarks

The datasets used in our NLU capabilities evaluations of pre-trained LMs are as follows,

- ***WikiText2*** (Merity et al., 2016)**.** A language modeling benchmark comprising over 2 million tokens extracted from human-reviewed Wikipedia articles, featuring original casing, punctuation, and a large vocabulary.

- ***Colossal Clean Crawled Corpus (C4)*** (Raffel et al., 2020)**.** A heuristically cleaned English text dataset derived from a massive web scrape of the Common Crawl archive.

- ***Penn Treebank (PTB)*** (Marcus et al., 1994)**.** A large-scale annotated corpus of approximately 4.5 million words, primarily from the Dow-Jones News Service and the Brown corpus, that are tagged for part-of-speech and syntactic structure.

The general utility and safety benchmarks used in our evaluations of instruction-tuned LMs are as follows,

- ***ARC*** (Clark et al., 2018)**.** A benchmark for grad-school science questions of easy and challenging reasoning difficulties.
- ***BoolQ*** (Clark et al., 2019)**.** A benchmark for question answering with yes or no answers based on passages.
- ***CommonSenseQA*** (Talmor et al., 2019)**.** A benchmark for commonsense reasoning knowledge.
- ***HellaSwag*** (Zellers et al., 2019)**.** A benchmark for completing sentences using commonsense reasoning.
- ***PIQA*** (Bisk et al., 2020)**.** A benchmark for questions on reasoning about physical interactions.
- ***SocialIQA*** (Sap et al., 2019)**.** A benchmark for questions on reasoning about social interactions.
- ***WinoGrande*** (Sakaguchi et al., 2021)**.** A benchmark for pronoun resolution problems using commonsense reasoning.
- ***SafetyBench*** (Zhang et al., 2024)**.** A benchmark for questions on safety understanding, categorized in 7 safety dimensions, namely offensiveness (OFF), unfairness and bias (UB), physical health (PH), mental health (MH), illegal activities (IA), ethics and morality (EM), and privacy and property (PP).

The instruction-following capabilities benchmark used in our evaluations of instruction-tuned LMs is ***MT-Bench*** (Zheng et al., 2023), a standard evaluation suite designed to test how well LMs handle continuous, multi-turn conversations and complex instructions.

Implementation and Licensing. The NLU capability datasets are shared under CC BY-SA 4.0 license for WikiText2, ODC-BY license for C4 and LDC license for PTB. For general utility benchmarks, we leveraged `lm-evaluation-harness` framework (Sutawika et al., 2023) their implementation, whereas for SafetyBench, we used their publicly available GitHub repository (git, 2024b). Both of these repositories are MIT licensed. We leveraged `fastchat` framework's implementation of MT-Bench that is shared under Apache-2.0 license.

C.2 State-of-the-Art Baseline PTQ Methods

The baseline PTQ methods used in our evaluations are as follows,

- ***AWQ*** (Lin et al., 2024b)**.** A post-training quantization that identifies a small fraction of salient weights based on activation magnitudes and protects them through per-channel scaling to minimize quantization error. This approach only supports weight-only quantization.
- ***SmoothQuant*** (Xiao et al., 2023)**.** A post-training quantization solution that enables 8-bit weight and activation quantization by migrating outlier-induced difficulty from activations to weights through a mathematically equivalent per-channel scaling transformation. This approach only support weight-activation quantization.
- ***OmniQuant*** (Shao et al., 2023)**.** A differentiable quantization technique that optimizes Learnable Weight Clipping (LWC) and Learnable Equivalent Transformation (LET) within a block-wise error minimization framework. This approach supports both weight-only and weight-activation quantization.
- ***DuQuant*** (Lin et al., 2024a)**.** A dual transformation quantization algorithm that employs block-wise rotation and zigzag permutation, to redistribute massive outliers across channels and blocks. This algorithm supports both weight-only and weight-activation quantization.
- ***SpinQuant*** (Liu et al., 2025)**.** A rotation-based quantization approach that incorporates learned rotation matrices optimized on the Stiefel manifold to mitigate outliers in both weights and activations. This approach supports both weight-only and weight-activation quantization.

Implementation. Among the above baseline PTQ methods, SmoothQuant, OmniQuant and DuQuant are implemented using their publicly available code in respective GitHub repositories, and AWQ and SpinQuant are implemented using `llm-compressor` framework (AI & vLLM Project, 2024). Code in Github repositories are shared under MIT license and `llm-compressor` code is shared under Apache-2.0 license.

D Full Results

D.1 Benchmark Evaluations

Table 4: ACC (↑) results of instruction-tuned LMs, *Llama-2-7B-Chat*, *Llama-3.1-8B-Instruct* and *Qwen2.5-7B-Instruct*, in 7 safety dimensions of SafetyBench, namely Offensiveness (OFF), Unfairness and Bias (UB), Physical Health (PH), Mental Health (MH), Illegal Activities (IA), Ethics and Morality (EM), and Privacy and Property (PP).

Model	Method	SafetyBench						
		OFF	UB	PH	MH	IA	EM	PP
Llama-2-7B-Chat	FP16	47.48	62.45	50.48	67.11	58.77	48.91	61.82
	AWQ	47.59	51.21	50.13	63.92	57.48	46.23	56.74
	OmniQuant	52.8	60.5	53.79	64.69	54.95	50.57	57.97
	DuQuant	48.03	59.51	48.56	65.33	58.38	48.91	61.97
	SpinQuant	48.37	46.59	47.08	57.41	50.39	45.50	54.50
	SafeQ	54.74	62.71	51.52	67.5	56.07	50.78	60.89
Llama-3.1-8B-Instruct	FP16	56.51	66.49	87.47	87.36	84.7	79.01	82.14
	AWQ	57.06	46.27	81.11	83.08	80.26	72.54	78.37
	OmniQuant	55.18	60.77	80.59	82.31	79.58	70.89	79.37
	DuQuant	50.03	66.49	80.59	84.99	81.50	71.92	78.98
	SpinQuant	64.10	56.36	82.68	82.69	78.74	71.25	77.91
	SafeQ	56.01	65.81	83.12	83.65	81.83	70.94	79.68
Qwen2.5-7B-Instruct	FP16	74.18	64.13	76.68	84.8	83.91	79.52	81.29
	AWQ	74.46	66.44	83.03	89.02	86.73	79.52	85.99
	OmniQuant	75.24	63.76	88.42	90.23	87.46	81.39	85.6
	DuQuant	-	-	-	-	-	-	-
	SpinQuant	69.31	64.55	86.16	89.14	86.90	78.49	78.49
	SafeQ	77.17	64.44	88.16	89.53	87.18	81.02	84.91

Table 5: ACC (↑) results of instruction-tuned LMs, *Llama-2-7B-Chat*, *Llama-3.1-8B-Instruct* and *Qwen2.5-7B-Instruct*, on zero-shot tasks of general utility benchmarks, ARC-challenge (ARC-c), ARC-easy (ARC-e), BoolQ, CommonSenseQA (CSQA), HellaSwag, PIQA, SocialIQA (SIQA) and WinoGrande.

Model	Method	General Utility							
		ARC-c	ARC-e	BoolQ	CSQA	HellaS.	PIQA	SIQA	WinoG.
Llama-2-7B-Chat	FP16	41.81	72.64	80.61	60.36	58.00	76.22	49.23	68.35
	AWQ	38.48	70.20	74.10	55.94	55.99	75.95	48.06	67.25
	OmniQuant	40.10	69.49	78.96	56.35	56.28	76.17	47.80	67.96
	DuQuant	40.96	72.05	81.41	57.99	56.98	75.79	48.41	67.32
	SpinQuant	37.20	67.51	74.53	45.95	50.74	72.91	44.17	63.22
	SafeQ	40.78	71.30	79.11	57.41	56.73	76.88	48.31	67.64
Llama-3.1-8B-Instruct	FP16	53.58	82.20	85.44	76.09	59.80	80.25	49.74	73.48
	AWQ	48.98	79.04	82.87	71.99	57.94	78.73	47.95	72.45
	OmniQuant	49.23	79.88	83.98	72.32	57.65	78.40	48.87	72.69
	DuQuant	48.12	79.25	83.88	73.38	58.40	79.82	50.36	72.69
	SpinQuant	50.17	80.26	84.50	72.97	57.41	79.43	49.18	72.53
	SafeQ	50.51	79.63	83.64	73.63	58.65	78.94	49.18	72.69
Qwen2.5-7B-Instruct	FP16	49.15	79.67	85.84	81.57	60.74	78.35	50.82	70.64
	AWQ	51.54	80.68	85.75	82.31	61.44	77.86	50.77	70.48
	OmniQuant	52.90	81.06	85.02	80.59	60.60	78.18	51.13	68.51
	DuQuant	-	-	-	-	-	-	-	-
	SpinQuant	50.51	80.30	84.95	81.16	60.67	77.09	50.67	68.19
	SafeQ	51.19	80.56	84.25	80.51	60.47	77.91	51.23	68.82

D.2 Efficiency Evaluations

Table 6: Inference speedup results for different batch sizes of *Llama-2-7B-Chat* (W4A16) decoding 128 tokens on NVIDIA RTX 5060 GPU for a sequence length of 512.

Batch Size	Method	Prefill		Decoding		End-to-End	
		Time (*ms*)	Speedup (×)	Speed (*ms/token*)	Speedup (×)	Time (*ms*)	Speedup (×)
1	FP16	135.642	1	30.2178	1	4003.805	1
	AWQ	149.193	0.9092	9.5559	3.1622	1372.087	2.918
	SpinQuant	140.261	0.9671	9.5359	3.1688	1361.852	2.94
	SafeQ	136.02	0.9972	9.3389	3.2357	1331.496	3.007
2	FP16	261.983	1	30.6737	1	4189.666	1
	AWQ	289.779	0.9041	10.1044	3.0357	1583.128	2.6464
	SpinQuant	274.947	0.9528	10.0429	3.0543	1560.219	2.6853
	SafeQ	265.934	0.9851	9.9689	3.077	1541.602	2.7177
4	FP16	519.848	1	32.1339	1	4632.801	1
	AWQ	582.981	0.8917	11.5689	2.7776	2062.842	2.2458
	SpinQuant	556.851	0.9335	11.5124	2.7913	2028.725	2.2836
	SafeQ	529.146	0.9824	11.3723	2.8256	1984.619	2.3344

Table 7: Memory saving results for different sequence lengths of *Llama-2-7B-Chat* (W4A16) decoding 128 tokens on NVIDIA RTX 5060 GPU for a batch size of 1.

Sequence Length	Method	Prefill		Decoding		End-to-End	
		Space (GiB)	Saving (×)	Space (GiB)	Saving (×)	Space (GiB)	Saving (×)
512	FP16	12.596	1	12.5525	1	12.596	1
	AWQ	3.7179	3.388	3.6744	3.4162	3.7179	3.388
	SpinQuant	3.6943	3.4095	3.6508	3.4383	3.6943	3.4095
	SafeQ	3.6032	3.4957	3.5597	3.5262	3.6032	3.4957
1024	FP16	12.6402	1	12.5525	1	12.6402	1
	AWQ	3.7616	3.3603	3.6744	3.4162	3.7616	3.3603
	SpinQuant	3.738	3.3815	3.6508	3.4383	3.738	3.3815
	SafeQ	3.647	3.466	3.5597	3.5262	3.647	3.466
2048	FP16	12.7261	1	12.5525	1	12.7261	1
	AWQ	3.848	3.3072	3.6744	3.4162	3.848	3.3072
	SpinQuant	3.8245	3.3276	3.6508	3.4383	3.8245	3.3276
	SafeQ	3.7324	3.4096	3.5597	3.5262	3.7324	3.4096

D.3 Ablation Studies

Table 8: ACC (↑) results of *Llama-3.1-8B-Instruct* (W3A16) for *SafeQ* ablations of ***calibration data*** in 7 safety dimensions of SafetyBench, namely Offensiveness (OFF), Unfairness and Bias (UB), Physical Health (PH), Mental Health (MH), Illegal Activities (IA), Ethics and Morality (EM), and Privacy and Property (PP). (Blue cells are default *SafeQ* parameters)

Data Samples	SafetyBench						
	OFF	UB	PH	MH	IA	EM	PP
16	47.81	53.1	50.39	62.77	58.61	45.6	61.97
32	44.54	45.8	61.27	73.12	69.85	53.41	69.59
64	49.86	47.06	51.26	68.97	60.12	47.78	64.9
128	56.01	46.01	55.61	68.97	60.91	52.07	64.13
256	57.01	48.69	67.01	68.39	68.39	57.24	69.44

Table 9: ACC (↑) results of *Llama-3.1-8B-Instruct* (W3A16) for *SafeQ* ablations of ***calibration data*** on zero-shot tasks of general utility benchmarks, ARC-challenge (ARC-c), ARC-easy (ARC-e), BoolQ, CommonSenseQA (CSQA), HellaSwag, PIQA, SocialIQA (SIQA) and WinoGrande. (Blue cells are default *SafeQ* parameters)

Data Samples	General Utility							
	ARC-c	ARC-e	BoolQ	CSQA	HellaS.	PIQA	SIQA	WinoG.
16	41.98	71	74.46	55.86	51.23	74.21	46.67	66.22
32	43.6	74.87	78.53	62.49	53.93	75.63	47.54	69.38
64	43	73.44	75.17	64.7	53.41	76.55	49.18	70.64
128	38.91	70.29	78.53	61.83	50.99	76.12	45.29	69.38
256	43.69	75.04	79.6	63.06	53	76.12	49.54	68.67

Table 10: ACC (↑) results of *Llama-3.1-8B-Instruct* (W3A16) for *SafeQ* ablations of ***training time*** in 7 safety dimensions of SafetyBench, namely Offensiveness (OFF), Unfairness and Bias (UB), Physical Health (PH), Mental Health (MH), Illegal Activities (IA), Ethics and Morality (EM), and Privacy and Property (PP). (Blue cells are default *SafeQ* parameters)

Epochs	SafetyBench						
	OFF	UB	PH	MH	IA	EM	PP
5	47.81	45.59	57.44	70.11	66.59	52.53	68.67
10	59.06	46.01	61.79	68.2	63.72	52.07	65.05
15	54.4	47.69	51.09	67.94	61.42	51.86	64.36
20	56.01	46.01	55.61	68.97	60.91	52.07	64.13
25	63.93	47.53	64.49	74.27	68.22	56.2	71.36

Table 11: ACC (↑) results of *Llama-3.1-8B-Instruct* (W3A16) for *SafeQ* ablations of ***training time*** on zero-shot tasks of general utility benchmarks, ARC-challenge (ARC-c), ARC-easy (ARC-e), BoolQ, CommonSenseQA (CSQA), HellaSwag, PIQA, SocialIQA (SIQA) and WinoGrande. (Blue cells are default *SafeQ* parameters)

Epochs	General Utility							
	ARC-c	ARC-e	BoolQ	CSQA	HellaS.	PIQA	SIQA	WinoG.
5	41.38	71.89	79.36	60.36	51.57	75.3	45.29	70.32
10	40.36	70.96	79.27	64.46	52	75.84	47.39	68.11
15	40.36	71.59	80.95	63.14	51.2	75.57	46.52	69.77
20	38.91	70.29	78.53	61.83	50.99	76.12	45.29	69.38
25	41.47	72.47	80.06	62.41	51.1	74.7	47.24	67.32

Table 12: ACC (↑) results of *Llama-3.1-8B-Instruct* (W3A16) for *SafeQ* ablations of different λ_{SA} in 7 safety dimensions of SafetyBench, namely Offensiveness (OFF), Unfairness and Bias (UB), Physical Health (PH), Mental Health (MH), Illegal Activities (IA), Ethics and Morality (EM), and Privacy and Property (PP). (Blue cells are default *SafeQ* parameters)

λ_{SA}	Safety						
	OFF	UB	PH	MH	IA	EM	PP
0.4	50.42	45.96	49.35	64.81	54.33	49.74	64.82
0.5	58.06	45.64	52.13	67.31	66.54	52.02	66.44
0.6	56.01	46.01	55.61	68.97	60.91	52.07	64.13
0.7	55.46	53.78	57.62	68.65	59.56	48.97	65.67
0.8	57.17	45.69	63.79	68.71	61.59	51.09	69.21

Table 13: ACC (↑) results of *Llama-3.1-8B-Instruct* (W3A16) for *SafeQ* ablations of different λ_{SA} on zero-shot tasks of general utility benchmarks, ARC-challenge (ARC-c), ARC-easy (ARC-e), BoolQ, CommonSenseQA (CSQA), HellaSwag, PIQA, SocialIQA (SIQA) and WinoGrande. (Blue cells are default *SafeQ* parameters)

λ_{SA}	General Utility							
	ARC-c	ARC-e	BoolQ	CSQA	HellaS.	PIQA	SIQA	WinoG.
0.4	38.91	71.68	78.44	56.18	52.06	75.03	45.45	67.56
0.5	41.3	71.97	78.72	65.03	51.42	75.95	45.85	70.01
0.6	38.91	70.29	78.53	61.83	50.99	76.12	45.29	69.38
0.7	38.74	69.82	77.37	60.44	51.07	74.59	46.72	69.77
0.8	41.81	72.43	79.79	64.7	51.36	75.19	46.01	69.06