EXAQ: Exponent Aware Quantization For LLMs Acceleration

Moran Shkolnik [†°*] **Maxim Fishman** [†*] **Brian Chmiel** [†]
Hilla Ben-Yaacov [†] **Ron Banner** [†] **Kfir Yehuda Levy** [°]

[†]Habana Labs – An Intel company, Caesarea, Israel,
[°]Department of Electrical Engineering - Technion, Haifa, Israel

{mshkolnik, mfishman, bchmiel, hbyaacov, rbanner}@habana.ai
kfiryehud@gmail.com

Abstract

Quantization has established itself as the primary approach for decreasing the computational and storage expenses associated with Large Language Models (LLMs) inference. The majority of current research emphasizes quantizing weights and activations to enable low-bit general-matrix-multiply (GEMM) operations, with the remaining non-linear operations executed at higher precision. In our study, we discovered that following the application of these techniques, the primary bottleneck in LLMs inference lies in the softmax layer. The softmax operation comprises three phases: exponent calculation, accumulation, and normalization, Our work focuses on optimizing the first two phases. We propose an analytical approach to determine the optimal clipping value for the input to the softmax function, enabling sub-4-bit quantization for LLMs inference. This method accelerates the calculations of both e^x and $\sum(e^x)$ with minimal to no accuracy degradation. For example, in LLaMA1-30B, we achieve baseline performance with 2-bit quantization on the well-known "Physical Interaction: Question Answering" (PIQA) dataset evaluation. This ultra-low bit quantization allows, for the first time, an acceleration of approximately 4x in the accumulation phase. The combination of accelerating both e^x and $\sum(e^x)$ results in a 36.9% acceleration in the softmax operation. A reference implementation[2] is provided.

1 Introduction

In recent years, the landscape of natural language processing (NLP) has been transformed by large language models (LLMs), showcasing unparalleled capabilities in contextual understanding and common sense reasoning. These capabilities are particularly evident as models are scaled up, driving research efforts towards further enlarging model dimensions [5, 23]. However, the substantial size of modern LLMs imposes considerable computational demands, making them resource-intensive in terms of training, fine-tuning, and inference processes. Consequently, there has been a surge in efforts to alleviate memory consumption and computational requirements. Among the promising approaches is quantization, a technique that involves representing parts of the model with lower bit widths, thereby reducing resource usage without compromising performance.

The foundation of LLMs lies in the attention mechanism [24], which encompasses intensive general-matrix-multiply (GEMM) operations, coupled with non-linear operations like softmax. Consequently,

[*]Equal contribution

[2]https://github.com/Anonymous1252022/EXAQ

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

prior quantization studies have primarily focused on reducing GEMM operations to 8 [18] or 4 [6, 9] bits with minimal to no degradation, showcasing significant advancements in this area. Additionally, modern hardware accelerators like Gaudi2 [1] and H100 [2] support accelerated 8-bit (FP8) GEMMs, further emphasizing advancements in this area of quantization and diminishing the computational burden of GEMM operations, thus alleviating them from being the primary computational load.

Once the bottleneck of GEMMs has been alleviated, attention has shifted toward reducing the computational demands of the softmax operation, which can account for more than 30% of the total inference time. Efforts to accelerate softmax operations within the attention mechanism have predominantly revolved around quantizing the entire dynamic range of softmax inputs to 16 or 8 bits [13]. However, this approach underscores the necessity for novel methods to enhance the efficiency of the softmax layer in terms of runtime, bandwidth, and memory usage, while maintaining accuracy. Our research indicates that current softmax acceleration via quantization is sub-optimal, suggesting that substantial performance improvements can be attained by tailoring the quantization optimization process to the specific properties of the softmax layer.

The softmax operation comprises three main parts: (1) Exponent calculation: This involves taking the exponent of each input element. (2) Accumulation: The exponentiated values are then summed together. (3) Normalization: Each exponentiated value is divided by the sum to obtain the final softmax probabilities. In this work, we are able to accelerate both steps (1) and (2) by quantizing, for the first time, the input to the exponent to below 4 bits.

First, we analyze the quantization error in the context of the exponential operation, comparing e^X to e^{X_q}, where X_q represents the quantized version of the input X. Subsequently, we introduce a pioneering approach to input quantization, coined "exponent-aware quantization" (EXAQ). This methodology presents an analytical model that strategically focuses on minimizing the quantization error after the exponent operation, directly targeting the exponential attributes of the softmax function. Lastly, we leverage the low-bit characteristics and propose a technique to unite the summation phase through a lookup table (LUT) operation, facilitating acceleration by approximately 4x. When we combine EXAQ with the accelerated accumulation we get an acceleration of 36.9 % in the softmax.

Our paper introduces several key contributions:

- We highlight the softmax layer as a significant computational bottleneck in modern NN.
- We propose an analytical approach to quantize the input to the exponent to below 4 bits, thereby enabling the utilization of a lookup table (LUT) based approach. This method notably diminishes the cycle consumption for computing e^x to a single cycle. In contrast to FP32/BF16/FP16 formats, where creating a reasonably sized table is impractical.
- We propose a technique to leverage the low-bit quantization of the softmax inputs and consolidate 4 consecutive summations into a lookup table (LUT), enabling up to 4x acceleration of the denominator accumulation process.
- Our method achieves state-of-the-art accuracy for low-bit quantization of the softmax operation in LLMs. With 2-bit quantization, it reaches baseline accuracy in several tasks with no degradation, and when averaging across all tasks, it shows an average degradation of only 1.9%. This exceptional efficiency allows for the creation of an exceedingly compact LUT with just 4 entries, rendering our approach highly suitable for deployment on edge devices with extremely limited computational resources.

2 Motivation

This section aims to illustrate the considerable computational demand imposed by the softmax operation, highlighting the advantages of improving its runtime efficiency. To establish a strong foundation for our argument, we conduct experiments to measure the runtime consumption using the "LLaMA-2-7B" LLM model on the Gaudi-2 accelerator, which is equipped with a high-speed network card for optimal performance.

In Fig.1, we depict the proportion of time allocated to each operation during the model's execution in BF16 format. This graphical representation emphasizes the softmax layer as the main computational bottleneck. With GEMM operations functioning in BF16 format, the softmax layer consumes 39% of the total runtime, while the GEMM operations contribute to 24% of the runtime. Furthermore, given

the advancement of modern accelerators supporting FP8 GEMMs acceleration, we anticipate that the softmax operation will consume an even larger portion of the total runtime.

To conclude, accelerating the softmax operation, particularly through techniques like quantization, has the potential to significantly increase the runtime efficiency of LLMs.

3 Exponent-Aware Quantization (EXAQ)

In this section, we introduce a novel quantization method, "exponent-aware quantization" (EXAQ), specifically designed for softmax inputs. The softmax function, defined as $\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$, operates by exponentiating its input logits and normalizing these values to form a probability distribution. Moreover, usually for numeric stability the maximum of x is subtracted before the exponent function. Our method focuses on minimizing the quantization error of the exponentiated outputs, ensuring a more precise representation of the softmax function's output. We manipulate the quantization in the input domain but target the mean squared error of the exponentiated outputs, minimizing $\text{MSE}(e^x, e^{Q(x)})$.

Inspired by the ACIQ paper [3], we limit the range of the tensor by clipping its values. While this introduces some distortion to the original tensor, it significantly reduces the rounding error in the part of the distribution containing most of the information. Since $x < 0$, we set a threshold $C < 0$, so that if $x < C$, then $x = C$. Clipping is particularly useful because it preserves the less negative values, which after exponentiation become significantly larger compared to very negative values that become negligible after the exponential function is applied. Values in the range are quantized on a smaller scale, improving resolution for the more common and important values. The method approximates the optimal clipping value analytically from the distribution of the tensor by minimizing the MSE between e^x and $e^{Q(x)}$. This analytical threshold is simple to use during run-time and can easily be integrated with other quantization techniques.

3.1 Problem Formulation

We begin by modifying the inputs for the function e^x through the subtraction of the maximum value, $\max(x)$, from these inputs. Thus, it is assumed that $x \leq 0$. The MSE due to quantization and clipping can be expressed as a sum of two integrals: one for the quantization error for $x \in [C, 0]$ and another for the clipping error for $x < C$. The quantization error integral is given by:

$$\text{MSE}_{\text{quant}} = \int_C^0 (e^{\text{Q}(x)} - e^x)^2 \cdot f(x)\, dx, \tag{1}$$

where $f(x)$ represents the probability density function of x, assumed to be gaussian distributed with mean μ and standard deviation σ. The clipping error integral is:

$$\text{MSE}_{\text{clip}} = \int_{-\infty}^C (e^C - e^x)^2 \cdot f(x)\, dx \tag{2}$$

Thus, total MSE is given by

$$\text{MSE} = \text{MSE}_{\text{clip}} + \text{MSE}_{\text{quant}} = \int_{-\infty}^C (e^C - e^x)^2 \cdot f(x)\, dx + \int_C^0 (e^{\text{Q}(x)} - e^x)^2 \cdot f(x)\, dx. \tag{3}$$

In Fig. 2 we present an illustration of the distortion of the proposed scheme. Before we get into the calculation of the mean squared error due to quantization, it is important to define the quantization process. We approximate the quantized value $\text{Q}(x)$ as $x + \epsilon$, where ϵ represents the quantization noise. This noise is assumed to be drawn from a uniform distribution [16, 3] within the range $[-\Delta/2, \Delta/2]$, where Δ is the quantization step size. For an M-bit integer quantization, the quantization step size Δ is defined as $\Delta = \frac{0-C}{2^M-1}$, accommodating the range of input values that need to be quantized. Given this quantization process, the MSE due to quantization can be analyzed as follows (full equations appears in appendix B):

$$\text{MSE}_{\text{quant}} = \int_C^0 (e^{\text{Q}(x)} - e^x)^2 \cdot f(x)\, dx, = \frac{\Delta^2}{12} \int_C^0 e^{2x} \cdot f(x)\, dx \tag{4}$$

Substituting Equation 4 into 3 we conclude that

$$\text{MSE} = \text{MSE}_{\text{quant}} + \text{MSE}_{\text{clip}} = -\frac{C^2}{12 \cdot (2^M - 1)^2} \int_C^0 e^{2x} \cdot f(x)\, dx + \int_{-\infty}^{C} (e^C - e^x)^2 \cdot f(x)\, dx \tag{5}$$

Solving equation 5 numerically for bit-widths $M = 2, 3$ and finding the optimal clipping value that minimizes MSE yields optimal clipping values as functions of the standard deviation (σ). These results are visualized in Figure 3 for a normal density function with standard deviation σ.

Finally, we use a linear approximation to estimate the optimal clipping values in the range [0.9, 3.4], where most standard deviations occur in practice (as seen in Figure 4). This approach allows us to avoid maintaining a detailed table that maps the standard deviation to optimal clipping values (C^*). Instead, we focus on keeping only two variables (slope and intercept) to estimate the linear approximation for the optimal clipping value. This way, once we have the standard deviation, we can immediately calculate the estimated optimal clipping value using the following table:

Table 1: Linear approximation for optimal clipping value (C^*)

Number of bits (M)	C^*
2	$-1.66 \cdot \sigma - 1.85$
3	$-1.75 \cdot \sigma - 2.06$

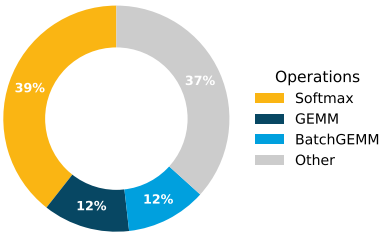


Figure 1: Distribution of runtime consumption by the layer type. The chart illustrates the proportional runtime spent on various layer types during model execution, highlighting the significant computational burden imposed by the softmax layer, which accounts for 39% of the total runtime. The data was measured on Llama2-7B on Gaudi2 device.

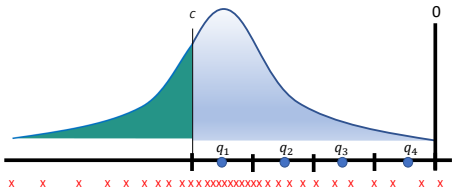


Figure 2: Illustration of the distortion at the output of e^x due to the quantization and clipping of the inputs. The clipping value C is the threshold we aim to optimize. A very negative C reduces clipping error but increases quantization error. The total mean squared error is the sum of these two contributions.

4 Algorithm implementation

The computation of the softmax function is typically broken down into three essential steps: (1) Exponent calculation: this step involves computing e^x for each input x (2) Accumulation: this step involves summing all the exponential values to form the denominator of the softmax function, and (3) Normalization: this step divides each exponential value by the computed sum to produce the final softmax output. Our algorithm primarily focuses on steps (1) and (2), leveraging the ultra-low precision of softmax inputs facilitated by the EXAQ method, to accelerate these two operations. Fig.5 compares the original softmax algorithm and our optimized 2-bit version, highlighting the computational efficiencies achieved.

Exponent calculation We replace the traditional direct exponent calculation (line 4 in Algo.2) with the following two steps: (1) We quantize each element in the normalized tensor into a 2-bit integer (line 4 in Algo.2). (2) We utilize pre-computed values from a lookup table LUT_{exp} to derive the exponents of the quantized values (lines 5-6 in Algo.2). This LUT_{exp} maps between all possible quantized values and their resulting exponents and is notably compact as it needs to store only 4

values. This approach not only reduces memory usage but also speeds up the algorithm since the exponential values can be retrieved in a single cycle.[3]

Accelerated denominator accumulation Originally, the accumulation process within the softmax layer's denominator requires summing up N exponential outputs (Algo.1 lines 7-12). In contrast, our algorithm requires only $N/4$. Since the input tensor's values are quantized to 2 bits, each byte can now represent 4 values. We utilize an additional lookup table, LUT_{sum}, that maps between all possible combinations of 4 quantized values to the value of the sum of their exponents. We calculate the denominator using the following steps. First, we divide the quantized tensor x_q into $N/4$ sequences (s denotes a sequence) of 4 values $[s_0, s_1, \dots s_{N/4-1}] = [x_q[0:4], x_q[4:8], \dots x_q[N-4:N]]$. Next, we apply LUT_{sum} on each s_i to obtain the sum of the exponents of the corresponding sequence. Finally, we sum the resulting $N/4$ values. In Figure 6 we show an illustration of the proposed accelerated denominator accumulation.

Our algorithm simplifies the accumulation step from 4 separate accumulations (4 cycles) to a single LUT access (1 cycle), enhancing the speed of the denominator calculation by a factor of 4. A pseudo-code of the proposed algorithm appears in Algo.2 (lines 10:13). The entire denominator accumulation process is completed within $N/4$ iterations, compared to the original algorithm shown in Algo.1 (lines 9:12), which requires N iterations for the same purpose. This algorithm also decouples the exponential computation from the denominator accumulation, allowing these steps to be executed concurrently, as opposed to the original softmax algorithm. Moreover, this approach can be extended to a 4-bit quantization, providing a 2x acceleration, as each byte can accommodate two 4-bit values.

5 Experiments

This section details the experimental framework used to evaluate the performance of our quantization method. We evaluate the accuracy across various language tasks, comparing our softmax quantization method (EXAQ) to the quantization implemented in A^3[12]. Our method achieves state-of-the-art accuracy scores in almost all experiments.

5.1 Accuracy experiments

Experimental settings Our accuracy experiments focus on the inference setting and are conducted on 8 RTX A6000 GPUs, utilizing a batch size of 4 for all evaluations. We use the LLaMA-1 models [22], specifically the 7B, 13B, 30B and 70B variants, and assess these models on a variety of question-answering and reasoning tasks, such as BoolQ [7] and WinoGrande [19]. The experiments are implemented using modifications to the lm-evaluation-harness [10], an open-source framework that utilizes pre-trained models from the *HuggingFace Project* [4].

Quantization settings. The softmax input quantization function parameters need to be tuned based on tensor statistics collected from a calibration set. In our experiments, we run a calibration set of size 100 by running 25 iterations each with a batch size of 4.

Inference accuracy evaluation Table 2 provides an insightful visual comparison of inference accuracy using different scales of LLaMA models (7B, 13B, 30B and 70B parameters) across 7 NLP tasks: BoolQ [7], HellaSwag [27], PIQA [4], WinoGrande [19], ARC Challenge [8], ARC Easy [8] and OpenBookQA [17]. All models have their softmax inputs quantized to 2-bit and 3-bit precision using our method EXAQ and the our implementation of A^3 method. EXAQ calculates the optimal clipping parameter using the standard deviation (σ) of the input tensor, as detailed in Table.1, while A^3 sets the clipping parameters by the entire range. Our method achieves state-of-the-art accuracy scores in almost all experiments (noted with bold marks in Table.2). With 3-bit softmax inputs, EXAQ reaches the baseline within 0.65% on average, with 43% of the results either meeting or exceeding the baseline accuracy (noted with green color in Table.2). With 2-bit softmax inputs, EXAQ approaches the baseline within 1.9% on average and reaches the baseline without degradation in several tasks. Additional experiments appear in section D.

5.2 Runtime experiments

We conducted runtime experiments to evaluate the overall performance of our algorithm, isolating the softmax operation to measure its runtime. Results are shown in Table.3. Our optimized algorithm

[3] While direct exponent calculation typically takes 5-12 cycles, depending on the hardware design.

[4] `https://huggingface.co/docs/transformers/main/en/model_doc/llama`

Table 2: Inference accuracy evaluation for different LLaMA-1 models across various tasks.

Q method	Prec.	BoolQ	HellaSwag	PIQA	WinoGrande	ARC Challenge	ARC Easy	OpenBookQA	avg score
NONE	BF16	75.1	76.2	79.2	69.6	45.1	73.2	44.4	66.1
A^3	INT2	46.6	54.5	69.0	55.6	30.5	55.9	36.8	49.8
EXAQ		**73.0**	**72.9**	**79.2**	**69.6**	**43.9**	**72.4**	**41.4**	**64.6**
A^3	INT3	71.3	73.7	78.5	67.2	42.8	71.0	43.4	63.9
EXAQ		**75.1**	**74.8**	**79.3**	**69.7**	**44.2**	**72.9**	**43.8**	**65.7**

(a) LLaMA-1-7B

Q method	Prec.	BoolQ	HellaSwag	PIQA	WinoGrande	ARC Challenge	ARC Easy	OpenBookQA	avg score
NONE	BF16	84.9	84.2	82.3	77.2	55.5	79.9	47.4	73.0
A^3	INT2	39.2	61.7	68.6	55.0	32.9	64.2	39.8	51.6
EXAQ		**72.3**	**79.4**	**81.6**	**76.7**	**53.9**	**78.9**	**47.2**	**70.0**
A^3	INT3	**82.5**	82.5	81.5	76.2	53.8	79.2	46.6	**71.8**
EXAQ		77.7	**82.7**	**82.5**	**77.0**	**54.7**	**79.9**	**47.6**	71.7

(b) LLaMA-1-65B

demonstrates a significant improvement in runtime performance for the softmax operation, achieving an enhancement of 36.9%.

6 Related work

In the quest to optimize neural networks for practical deployment, particularly LLMs, studies like [14, 26] have introduced innovative approaches to reduce computational demands. [14], for example, implements an integer-only quantization scheme for Transformers that conducts all inference operations with integer arithmetic, using INT32 for softmax inputs. Similarly, [26] applies selective quantization to Transformers, focusing specifically on GEMM layers while keeping softmax in FP32.

Other works focus on softmax acceleration as it has become a bottleneck in recent years for LLMs. [21] proposes to use basic-split calculation method, which allows to split the exponentiation calculation of the softmax into several specific basics which are implemented by LUTs and multipliers. [15, 11, 20, 28] compute the exponential operations of integer and fractional parts separately using a combination of LUTs and piecewise linear (PWL) function fitting. [15, 11] also accelerate the division operation by replacing the divider with shifter units. The most closely related works to ours are [12] and [25], both of which aim to accelerate the softmax operation and, like our method, do not require a fine-tuning phase. [12] addresses only the exponent calculation acceleration, disregarding the denominator. A detailed comparison is in C.1. [25] introduces two methods for softmax acceleration: one using two 1D-LUTs combined with a multiplier, and another using a combination of 1D-LUT and 2D-LUT, without a multiplier. A detailed comparison is in C.2

7 Discussion

Summary This study analyzes the execution time of various operations during LLMs inference and demonstrates that the softmax operation emerges as one of the primary bottlenecks, likely to become even more critical with advances in GEMM acceleration.

Based on this conclusion, we introduce EXAQ - an analytical approach aimed at reducing the dynamic range of the exponent input, thereby enabling sub-4-bit quantization and accelerating the exponent calculation. Additionally, leveraging ultra-low quantization, we propose a method to accelerate the accumulation step by up to 4 times. The proposed full solution is able to get 36.9% acceleration in the softmax operation. We demonstrate that our proposed method achieves minimal to no degradation for the first time, in 2-bit and 3-bit quantization across various LLM sizes and a range of evaluated tasks.

Limitations We focused on minimizing the quantization error of the exponential output. A more precise approach, however, would involve minimizing the quantization error of the softmax outputs or the entire attention block. This alternative approach was not explored in the current research and is identified as an important avenue for future work. Additionally, our methodology was tested only during the inference stage of the model's lifecycle. Exploring its effects during the training phase remains an area for future investigation.

References

[1] Habana gaudi. URL https://docs.habana.ai/en/latest/PyTorch/Inference_on_PyTorch/Inference_Using_FP8.html.

[2] Nvidia h100. URL https://www.nvidia.com/en-us/data-center/h100/.

[3] Ron Banner, Yury Nahshan, and Daniel Soudry. Post training 4-bit quantization of convolutional networks for rapid-deployment. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 7948–7956, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/c0a62e133894cdce435bcb4a5df1db2d-Abstract.html.

[4] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7432–7439. AAAI Press, 2020. doi: 10.1609/AAAI.V34I05.6239. URL https://doi.org/10.1609/aaai.v34i05.6239.

[5] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *ArXiv*, abs/2005.14165, 2020. URL https://api.semanticscholar.org/CorpusID:218971783.

[6] Brian Chmiel, Ron Banner, Elad Hoffer, Hilla Ben-Yaacov, and Daniel Soudry. Accurate neural training with 4-bit matrix multiplications at standard formats. In *ICLR*, 2023.

[7] 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 Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 2924–2936. Association for Computational Linguistics, 2019. doi: 10.18653/V1/N19-1300. URL https://doi.org/10.18653/v1/n19-1300.

[8] 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*, abs/1803.05457, 2018. URL https://api.semanticscholar.org/CorpusID:3922816.

[9] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *ArXiv*, abs/2210.17323, 2022. URL https://api.semanticscholar.org/CorpusID:253237200.

[10] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 12 2023. URL https://zenodo.org/records/10256836.

[11] Xue Geng, Jie Lin, Bin Zhao, Anmin Kong, Mohamed M. Sabry Aly, and Vijay Ramaseshan Chandrasekhar. Hardware-aware softmax approximation for deep neural networks. In *Asian Conference on Computer Vision*, 2018. URL https://api.semanticscholar.org/CorpusID:59242579.

[12] Tae Jun Ham, Sungjun Jung, Seonghak Kim, Young H. Oh, Yeonhong Park, Yoonho Song, Jung-Hun Park, Sanghee Lee, Kyoung Park, Jae W. Lee, and Deog-Kyoon Jeong. A^3: Accelerating attention mechanisms in neural networks with approximation. In *IEEE International Symposium on High Performance Computer Architecture, HPCA 2020, San Diego, CA, USA, February 22-26, 2020*, pages 328–341. IEEE, 2020. doi: 10.1109/HPCA47549.2020.00035. URL https://doi.org/10.1109/HPCA47549.2020.00035.

[13] Gamze Islamoglu, Moritz Scherer, Gianna Paulin, Tim Fischer, Victor J. B. Jung, Angelo Garofalo, and Luca Benini. ITA: an energy-efficient attention and softmax accelerator for quantized transformers. In *IEEE/ACM International Symposium on Low Power Electronics and Design, ISLPED 2023, Vienna, Austria, August 7-8, 2023*, pages 1–6. IEEE, 2023. doi: 10.1109/ISLPED58423.2023.10244348. URL https://doi.org/10.1109/ISLPED58423.2023.10244348.

[14] Sehoon Kim, Amir Gholami, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. I-bert: Integer-only bert quantization. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5506–5518. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/kim21d.html.

[15] Zhenmin Li, Henian Li, Xiange Jiang, Bangyi Chen, Yue Zhang, and Gaoming Du. Efficient fpga implementation of softmax function for dnn applications. *2018 12th IEEE International Conference on Anti-counterfeiting, Security, and Identification (ASID)*, pages 212–216, 2018. URL https://api.semanticscholar.org/CorpusID:121348740.

[16] D. Marco and D. L. Neuhoff. The validity of the additive noise model for uniform scalar quantizers,. In *IEEE Transactions on Information Theory, vol. 51, no. 5, pp. 1739-1755*, 2005.

[17] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Conference on Empirical Methods in Natural Language Processing*, 2018. URL https://api.semanticscholar.org/CorpusID:52183757.

[18] Houwen Peng, Kan Wu, Yixuan Wei, Guoshuai Zhao, Yuxiang Yang, Ze Liu, Yifan Xiong, Ziyue Yang, Bolin Ni, Jingcheng Hu, Ruihang Li, Miaosen Zhang, Chen Li, Jia Ning, Ruizhe Wang, Zheng Zhang, Shuguang Liu, Joe Chau, Han Hu, and Peng Cheng. Fp8-lm: Training fp8 large language models. *ArXiv*, abs/2310.18313, 2023. URL https://api.semanticscholar.org/CorpusID:264555252.

[19] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106, 2021. doi: 10.1145/3474381. URL https://doi.org/10.1145/3474381.

[20] Jacob R. Stevens, Rangharajan Venkatesan, Steve Dai, Brucek Khailany, and Anand Raghunathan. Softermax: Hardware/software co-design of an efficient softmax for transformers. *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 469–474, 2021. URL https://api.semanticscholar.org/CorpusID:232257991.

[21] Qiwei Sun, Zhixiong Di, Zhe Yuan Lv, Fengli Song, Qianyin Xiang, Quanyuan Feng, Yibo Fan, Xulin Yu, and Wenqiang Wang. A high speed softmax vlsi architecture based on basic-split. *2018 14th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*, pages 1–3, 2018. URL https://api.semanticscholar.org/CorpusID:54450044.

[22] 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*, 2023.

[23] Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian

Khabsa, Isabel M. Kloumann, A. V. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288, 2023. URL `https://api.semanticscholar.org/CorpusID:259950998`.

[24] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf`.

[25] Ihor Vasyltsov and Wooseok Chang. Efficient softmax approximation for deep neural networks with attention mechanism. *arXiv preprint arXiv:2111.10770*, 2021.

[26] Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. Q8bert: Quantized 8bit bert. In *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing - NeurIPS Edition (EMC2-NIPS)*. IEEE, December 2019. doi: 10.1109/emc2-nips53020.2019.00016. URL `http://dx.doi.org/10.1109/EMC2-NIPS53020.2019.00016`.

[27] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4791–4800. Association for Computational Linguistics, 2019. doi: 10.18653/V1/P19-1472. URL `https://doi.org/10.18653/v1/p19-1472`.

[28] Danyang Zhu, Siyuan Lu, Meiqi Wang, Jun Lin, and Zhongfeng Wang. Efficient precision-adjustable architecture for softmax function in deep learning. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 67(12):3382–3386, 2020.

.

Appendix

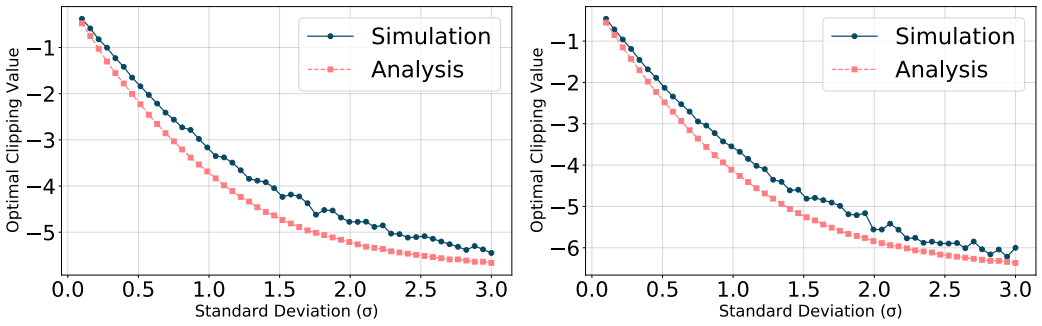
A Further discussion

A.1 Broader impacts

The acceleration of large language model (LLM) runtime significantly impacts modern life, particularly as tools like ChatGPT and Gemini become more integrated into daily use. Speeding up these models is essential for ongoing development and growth in this area, as it tackles a critical bottleneck: the processing speed and efficiency of the algorithms. Moreover, enhancing these models' speed and reducing their memory footprint not only improves their performance but also makes them more accessible to a wider audience. This allows more users to customize and advance these models for their specific needs and developments.

B Full MSE equation

$$\begin{aligned}
\text{MSE}_{\text{quant}} &= \int_C^0 (e^{\text{Q}(x)} - e^x)^2 \cdot f(x)\, dx, && (6)\\
&= \frac{1}{\Delta}\int_{-\Delta/2}^{\Delta/2}\int_C^0 (e^{x+\epsilon} - e^x)^2 \cdot f(x)\, d\epsilon\, dx, && (7)\\
&\approx \frac{1}{\Delta}\int_{-\Delta/2}^{\Delta/2}\int_C^0 (e^x + \epsilon e^x - e^x)^2 \cdot f(x)\, d\epsilon\, dx, \quad (e^{x+\epsilon} \approx e^x + \epsilon e^x) && (8)\\
&= \frac{1}{\Delta}\int_{-\Delta/2}^{\Delta/2}\int_C^0 (\epsilon e^x)^2 \cdot f(x)\, d\epsilon\, dx, && (9)\\
&= \frac{1}{\Delta}\int_{-\Delta/2}^{\Delta/2} \epsilon^2\, d\epsilon \int_C^0 (e^x)^2 \cdot f(x)\, dx && (10)\\
&= \frac{1}{\Delta}\left[\frac{\epsilon^3}{3}\right]_{-\Delta/2}^{\Delta/2}\int_C^0 e^{2x} \cdot f(x)\, dx && (11)\\
&= \frac{\Delta^2}{12}\int_C^0 e^{2x} \cdot f(x)\, dx && (12)
\end{aligned}$$



(a) 2-bit (b) 3-bit

Figure 3: Optimal clipping value vs. standard deviation of softmax input for different bit widths. The analysis (Equation 5) and simulation (grid-search) results agree, demonstrating the accuracy of the analytical model. The simulation was conducted from Llama2 7b dumps.

C Comparing our algorithm to the latest algorithms

C.1 A competitive comparison against [12]

A key advantage of our approach is its significant improvement in denominator accumulation, reducing the number of required accumulations by a factor of 4, a notable acceleration, as the method proposed in [12] does

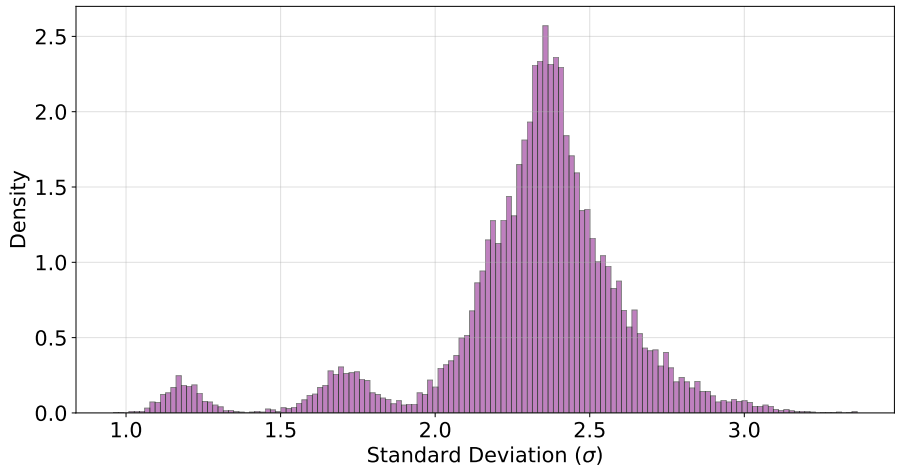


Figure 4: Standard deviation of softmax input collected across different layers and iterations.

Algorithm 1 Original softmax algorithm

Input: input tensor x
Output: softmax tensor $out(x)$
Normalize input tensor: $x = x - \max(x)$
for $i = 1$ to size(x) **do**
    $e[i] = e^{x[i]}$ ▷ multi cycle op
end for
Denominator accumulation:
$sum = 0$
$i = 1$
while $i \leq$ size(x) **do**
    $sum = sum + e[i]$
    $i+ = 1$
end while
for $i = 1$ to size(x) **do**
    $out(x[i]) = e[i]/sum$
end for

Algorithm 2 2-bit softmax algorithm

Input: input tensor x, LUT_{exp}, LUT_{sum}, scale, offset, clip
Output: softmax tensor $out(x)$
Normalize input tensor: $x = x - \max(x)$
quantize x: ▷ 3 cycles op
$x_q = \text{Q}(x, \text{scale}, \text{offset}, \text{clip})$
for $i = 1$ to size(x) **do**
    $e[i] = LUT_{\text{exp}}[x_q[i]]$ ▷ 1 cycle op
end for
Denominator accumulation:
$sum = 0$
$i = 1$
while $i \leq$ size(x) **do**
    $sum = sum + LUT_{\text{sum}}[x_q[i : i + 3]]$
    $i+ = 4$
end while
for $i = 1$ to size(x) **do**
    $out(x[i]) = e[i]/sum$
end for

Figure 5: Comparison of softmax algorithms: Algorithm 1 details the original softmax computation method, involving multiple cycle exponential operations and N accumulations in the denominator. Algorithm 2 introduces a 2-bit optimized version using lookup tables (LUTs), which involves a single cycle exponential operation and $N/4$ accumulations in the denominator.

not address this aspect. The method in [12] quantizes FP16 inputs to 16-bit fixed-point and calculates exponents using two separate 256-entry LUTs, followed by a multiplication. This process requires 3 cycles for the two LUT accesses and the subsequent multiplication. In contrast, our algorithm quantizes inputs to 2-bit integers and uses a single ultra-small LUT with only 4 entries. This streamlined approach reduces the process to just one cycle for the single LUT access, significantly enhancing both runtime and memory efficiency compared to the method in [12].

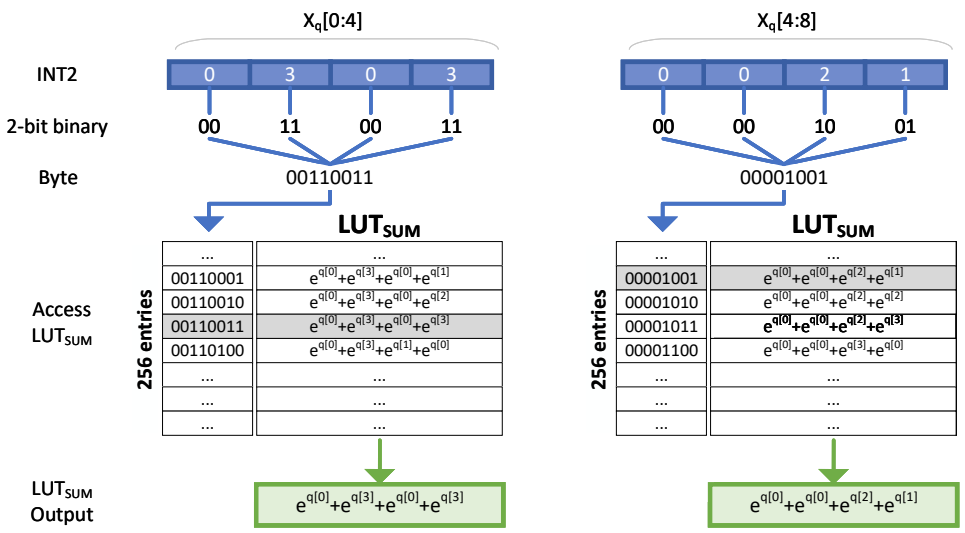


Figure 6: Illustration of the proposed accelerated denominator accumulation. The LUT_{sum} lookup table contains pre-computed values of sums of the exponents of 4 consecutive quantized tensor elements. In the left example, the integer representations of the quantized values are $X_q[0:4] = [0, 3, 0, 3]$, and their corresponding floating-point representations are $[q[0], q[3], q[0], q[3]]$. The lookup key is constructed by concatenating the 2-bit counterparts of the 4 integer representations into a single byte.

Table 3: Softmax layer runtime performance for INT2 experiments

Implementation	Average Runtime (ms)
Original algorithm (Algo.1)	3.274
Our algorithm (Algo.2)	2.066

C.2 A competitive comparison against [25]

This work supports softmax inputs in integer format and introduces two methods to accelerate the softmax operation via approximation. The first method employs two 1-dimensional lookup tables (1D-LUTs) to approximate e^x and $\frac{1}{x}$, combining these outputs with a multiplier to produce the final result. The second method combines a 1D-LUT and a 2D-LUT (2-dimensional lookup table). In this approach, the output from the 1D-LUT and the results from the accumulated denominator are used as the indices [i, j] for the 2D-LUT, which directly contains the final softmax result, thereby eliminating the need for multiplication or division. However, this approach has been noted to cause an additional drop in accuracy. Additionally, a de-quantization phase is conducted if the next layer requires an FP format. To conclude, assuming the softmax inputs are in floating-point format, both our method and that of [25] require an initial quantization phase. Each approach utilizes LUTs to approximate e^x, with each requiring just one cycle for LUT access. However, our work significantly accelerates the denominator accumulation phase by a factor of 4. In contrast, [25] enhances the normalization phase efficiency by combining a 1D-LUT with a multiplier or a direct use of a 2D-LUT. The distinct enhancements made by each method suggest a potential synergy if integrated. Our improvements in denominator accumulation could complement the division optimizations made by [25], offering a complete enhancement to the softmax function. Additionally, while [25]'s process concludes with a de-quantization phase that requires additional computational steps, our method eliminates the need for this phase, reducing overall cycles, thereby providing an advantage to our approach.

D Additional Experiments

Table 4: The standard deviation (σ) over multiple runs of LLaMA-1. The mean values of these runs are presented in Table 2 in the main paper.

Q method	Prec.	BoolQ	HellaSwag	PIQA	WinoGrande	ARC Challenge	ARC Easy	OpenBookQA
NONE	FP16	0.76	0.43	0.95	1.29	1.45	0.91	2.22
A^3	INT2	0.87	0.50	1.08	1.40	1.34	1.02	2.16
EXAQ	INT2	0.78	0.44	0.95	1.29	1.45	0.92	2.20
A^3	INT3	0.79	0.44	0.96	1.32	1.45	0.93	2.22
EXAQ	INT3	0.76	0.43	0.95	1.29	1.45	0.91	2.22

(a) LLaMA-1-7B

Q method	Prec.	BoolQ	HellaSwag	PIQA	WinoGrande	ARC Challenge	ARC Easy	OpenBookQA
NONE	FP16	0.73	0.41	0.93	1.25	1.46	0.89	2.23
A^3	INT2	0.87	0.50	1.13	1.40	1.29	1.03	2.18
EXAQ	INT2	0.77	0.43	0.94	1.27	1.46	0.90	2.23
A^3	INT3	0.77	0.42	0.95	1.26	1.46	0.91	2.23
EXAQ	INT3	0.75	0.41	0.93	1.27	1.46	0.89	2.22

(b) LLaMA-1-13B

Q method	Prec.	BoolQ	HellaSwag	PIQA	WinoGrande	ARC Challenge	ARC Easy	OpenBookQA
NONE	FP16	0.66	0.38	0.90	1.21	1.46	0.84	2.24
A^3	INT2	0.87	0.48	1.05	1.38	1.38	0.98	2.21
EXAQ	INT2	0.69	0.41	0.91	1.23	1.46	0.85	2.24
A^3	INT3	0.69	0.39	0.91	1.21	1.46	0.84	2.23
EXAQ	INT3	0.69	0.39	0.91	1.21	1.46	0.84	2.24

(c) LLaMA-1-30B

Q method	Prec.	BoolQ	HellaSwag	PIQA	WinoGrande	ARC Challenge	ARC Easy	OpenBookQA
NONE	FP16	0.63	0.36	0.89	1.18	1.45	0.82	2.24
A^3	INT2	0.85	0.49	1.08	1.40	1.37	0.98	2.19
EXAQ	INT2	0.78	0.40	0.90	1.19	1.46	0.84	2.23
A^3	INT3	0.66	0.38	0.91	1.20	1.46	0.83	2.23
EXAQ	INT3	0.73	0.38	0.89	1.18	1.45	0.82	2.23

(d) LLaMA-1-65B

Table 5: Inference accuracy evaluation for different LLaMA-1 and LLama-2 models across various tasks

Q method	Prec.	BoolQ	HellaSwag	PIQA	WinoGrande	ARC Challenge	ARC Easy	OpenBookQA	avg score
NONE	BF16	77.7	79.1	80.1	72.8	47.9	74.8	44.6	68.2
A^3	INT2	56.1	48.5	62.3	55.6	26.6	51.7	38.4	48.5
EXAQ		**73.9**	**75.8**	**79.4**	**71.7**	**47.9**	**73.7**	**44.6**	**66.7**
A^3	INT3	73.7	77.3	79.1	**71.8**	45.1	72.6	44.6	66.3
EXAQ		**76.1**	**78.0**	**80.3**	71.7	**47.9**	**74.6**	**45.8**	**67.8**

(a) LLaMA-1-13B

Q method	Prec.	BoolQ	HellaSwag	PIQA	WinoGrande	ARC Challenge	ARC Easy	OpenBookQA	avg score
NONE	BF16	82.8	82.6	81.2	75.3	53.0	78.9	48.4	71.9
A^3	INT2	54.0	64.9	72.1	60.0	33.8	64.4	42.6	56.0
EXAQ		**80.6**	**78.1**	**81.3**	**74.3**	**51.9**	**77.9**	**47.8**	**70.3**
A^3	INT3	**81.0**	81.4	81.2	75.1	51.8	78.5	47.0	70.9
EXAQ		80.6	**80.7**	**82.2**	**75.3**	**54.0**	**78.8**	**48.0**	**71.4**

(b) LLaMA-1-30B

Q method	Prec.	BoolQ	HellaSwag	PIQA	WinoGrande	ARC Challenge	ARC Easy	OpenBookQA	avg score
NONE	BF16	77.9	76.0	78.9	69.1	46.2	74.8	44.2	66.7
A^3	INT2	58.6	33.5	61.4	51.3	25.2	40.0	29.6	42.8
EXAQ		**73.7**	**74.4**	**78.0**	**68.4**	**44.5**	**72.3**	**42.2**	**64.8**
A^3	INT3	69.9	72.5	77.6	66.9	43.4	70.2	42.8	63.3
EXAQ		**75.9**	**75.5**	**78.9**	**68.8**	**46.4**	**74.8**	**44.0**	**66.3**

(c) LLaMA-2-7B

Q method	Prec.	BoolQ	HellaSwag	PIQA	WinoGrande	ARC Challenge	ARC Easy	OpenBookQA	avg score
NONE	BF16	80.7	79.3	80.6	72.5	49.4	77.3	45.6	69.3
A^3	INT2	54.9	35.6	58.4	51.2	24.9	41.4	33.4	42.8
EXAQ		**77.5**	**77.7**	**79.4**	**70.0**	**48.5**	**76.9**	**46.6**	**68.1**
A^3	INT3	72.1	77.0	79.1	70.2	48.2	74.8	**44.8**	66.6
EXAQ		**79.7**	**78.9**	**80.0**	**71.7**	**48.5**	**77.5**	44.4	**68.7**

(d) LLaMA-2-13B

Q method	Prec.	BoolQ	HellaSwag	PIQA	WinoGrande	ARC Challenge	ARC Easy	OpenBookQA	avg score
NONE	BF16	83.6	83.8	82.8	77.9	57.5	80.9	48.4	73.6
A^3	INT2	51.1	46.5	69.3	55.5	30.3	64.9	45.4	51.9
EXAQ		**74.8**	**73.3**	**82.4**	**76.2**	**54.8**	**79.4**	**48.2**	**69.9**
A^3	INT3	**79.0**	**83.2**	82.6	75.7	57.0	**80.7**	48.6	**72.4**
EXAQ		77.9	78.9	**82.9**	**77.0**	56.9	80.0	**49.0**	71.8

(e) LLaMA-2-70B

Table 6: The standard deviation (σ) over multiple runs of LLaMA-2 models. The mean values of these runs are presented above in Table.5 in the appendix.

Q method	Prec.	BoolQ	HellaSwag	PIQA	WinoGrande	ARC Challenge	ARC Easy	OpenBookQA
NONE	FP16	0.73	0.43	0.95	1.30	1.46	0.89	2.22
A^3	INT2	0.86	0.47	1.14	1.40	1.27	1.01	2.04
EXAQ		0.77	0.44	0.97	1.31	1.45	0.92	2.21
A^3	INT3	0.80	0.45	0.97	1.32	1.45	0.94	2.21
EXAQ		0.75	0.43	0.95	1.30	1.46	0.89	2.22

(a) LLaMA-2-7B

Q method	Prec.	BoolQ	HellaSwag	PIQA	WinoGrande	ARC Challenge	ARC Easy	OpenBookQA
NONE	FP16	0.69	0.40	0.92	1.26	1.46	0.86	2.23
A^3	INT2	0.87	0.48	1.15	1.40	1.26	1.01	2.11
EXAQ		0.73	0.42	0.94	1.29	1.46	0.87	2.23
A^3	INT3	0.78	0.42	0.95	1.29	1.46	0.89	2.23
EXAQ		0.70	0.41	0.93	1.27	1.46	0.86	2.22

(b) LLaMA-2-13B

Q method	Prec.	BoolQ	HellaSwag	PIQA	WinoGrande	ARC Challenge	ARC Easy	OpenBookQA
NONE	FP16	0.65	0.37	0.88	1.17	1.44	0.81	2.24
A^3	INT2	0.87	0.50	1.08	1.40	1.34	0.98	2.23
EXAQ		0.76	0.44	0.89	1.20	1.45	0.83	2.24
A^3	INT3	0.71	0.37	0.88	1.21	1.45	0.81	2.24
EXAQ		0.73	0.41	0.88	1.18	1.45	0.82	2.24

(c) LLaMA-2-70B