Post-Training Quantization of State Space Models: A Practical Study

Anonymous Author(s)
Affiliation
Address
email

Abstract

State Space Models (SSMs) are developed to address the quadratic dependency of the attention-based model and their computational inefficiency in long-sequence token generation. However, it is unclear how such models behave in the presence of low-precision quantization, which is a crucial step in deploying large models in low-resource environments. In this paper, we study the memory-accuracy tradeoff of such models by applying post-training quantization in moderate (8-bit) and extreme low-precision (4-bit) cases with simple *Round-to-Nearest* and the more popular *GPTQ* quantization scheme on both generation and zero-shot tasks. Interestingly, we observe that SSMs outperform OPT and Pythia on the same number of parameters up to 8-bit. In addition, we found that in the 4-bit case, small OPT and Pythia ($<$ 1 Billion parameters) are more accurate than SSMs. However, 4-bit SSMs with at least 1 Billion parameters still outperform OPT and Pythia on the same model sizes.

1 Introduction

Large Language Models (LLMs) have achieved remarkable performance on a wide range of natural language tasks, revolutionizing the field of natural language processing [Wang et al., 2024]. However, scaling these models to larger sizes poses significant computational challenges. Even during the inference, these models 1) have a quadratic dependency on the input which limits their usage for long contexts, and 2) have a significant number of parameters which leads to large memory requirements.

State Space Models (SSMs) [Alonso et al., 2024] have emerged as an alternative architecture to address the first computational challenge of attention-based models by removing their quadratic dependency on the input [Fu et al., 2022]. These models scale linearly (instead of $O(n^2)$ in attention-based LLMs) which makes them more appealing for processing long contexts. They do this by applying a recursive formula on the input tokens at different layers and generating the next token [Alonso et al., 2024, Gu and Dao, 2023]. However, these models still have a huge number of parameters (ranging from a few million to billions), which require significant memory to store the model. Scaling these models will become even worse in the future.

Post-training Quantization is a well-known approach to tackle the memory issue of the neural networks [Gholami et al., 2022] by allocating fewer bits for every element. These techniques are widely used for weight-only [Frantar et al., 2022, Tseng et al., 2024, Lin et al., 2023, Dettmers et al., 2023] and weight-activation quantization [Dettmers et al., 2022, Xiao et al., 2023, Ashkboos et al., 2023, 2024] as they do not need to involve expensive training steps. However, different models have different sensitivity to quantization and it is hard to find a universal precision for all models.

In this work, we provide an empirical study of applying various quantization schemes to SSMs, in particular Mamba family [Gu and Dao, 2023]. To this end, we focus on analyzing the model size

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

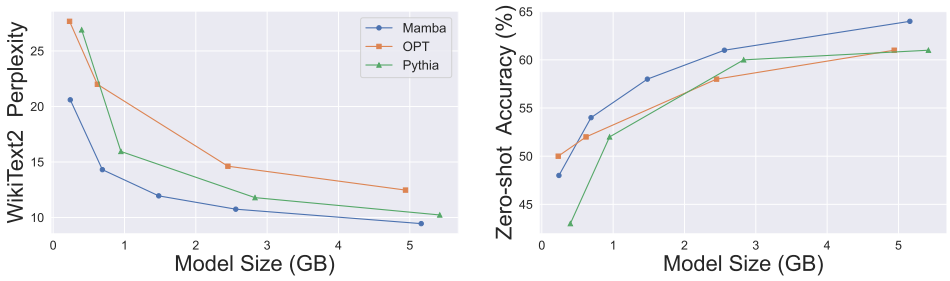

Figure 1: Accuracy of the baseline models on various tasks. **Left:** WikiText-2 generation perplexity. **Right:** Average accuracy over four popular tasks: HellaSwag, LAMBADA, PIQA, and WinoGrande.

vs. accuracy against state-of-the-art attention-based language models on various tasks (including generation and zero-shot tasks). More specifically, we apply **GPTQ** [Frantar et al., 2022] and **Round-to-Nearest** [Yao et al., 2022] quantization schemes on the Mamba models and compare them against popular LLM models (OPT and Pythia), while controlling for their number of parameters. Our ultimate goal is to find the most accurate model for a given memory size and compare the SSMs and attention-based models in a limited memory environment.

2 Background and Methods

In this section, we provide the main concepts related to Quantization. Then we explain the key characteristics of the Mamba models. Finally, we present our method for our study.

2.1 Quantization

Quantization is the process of converting a set of values into a lower-precision representation. In the simplest case, the representation contains a set of integers, known as *Integer Quantization*. For a given input vector $\mathcal{X}$, we use column-wise asymmetric quantization with a quantization function

$$\mathcal{Q}(\mathcal{X}) = round_{int}(\frac{X}{s}) - z, \tag{1}$$

where s is a scaling value to transform the input into the representable range, $round_{int}$ is a round-to-nearest integer operation, and z is a zero point to shift the rounded values to improve the quantization accuracy. For a b-bit integer, the representable range is $[-2^b, 2^b - 1]$.

While *Round-to-Nearest* (RtN) quantization applies $\mathcal{Q}(\mathcal{X})$ on each weight column independently, GPTQ starts from the first column and quantizes the weight columns one by one. At each step, the algorithm quantizes the column and compensates for the introduced error by updating the not-yet-quantized columns using the inverse of the Hessian matrix, extracted on a calibration set.

2.2 Mamba Models

State space models (SSMs) are a class of models that use a recursive formulation to generate the next token during the inference. For a fixed time step t and an input matrix $\mathbf{U}(t)$, the output vector $\mathbf{y}(t)$ is generated using

$$\dot{\mathbf{X}}(t) = \mathbf{A}(t)\mathbf{X}(t) + \mathbf{B}(t)\mathbf{U}(t), \quad \mathbf{y}(t) = \mathbf{C}(t)\mathbf{X}(t) + \mathbf{D}(t)\mathbf{U}(t)$$

where $\mathbf{A}(t)$ captures the internal dynamics, the input matrix $\mathbf{B}(t)$ determines the influence of external inputs on the states, and $\mathbf{C}(t)$ and the feed-forward matrix $\mathbf{D}(t)$ define how the internal states and inputs are transformed into observable outputs. The Mamba architecture follows the same structure and integrates the design of H3 [Fu et al., 2022] with the MLP block of Transformers into a unified block, eliminating the need for separate attention or MLP components.

Mamba is a state-of-the-art SSM designed to enhance the performance of SSMs to match the effectiveness of attention mechanisms in natural language tasks and to overcome the limitations of traditional Transformer architectures in handling long sequences. It introduces input-dependent parameterization which enables the model to selectively propagate or forget information based on the current token. This addresses the weaknesses of previous SSMs in managing discrete and information-dense data such as text (see Figure 1).

3 Experiments

Setup. We use HuggingFace [Wolf et al., 2020] and implement the quantization schemes in PyTorch [Paszke et al., 2019]. We study a range of parameter sizes up to 3B for OPT [Zhang et al., 2022] family (125M, 350M, 1.3B, and 2.7B), Pythia [Biderman et al., 2023] family (160M, 410M, 1.4B, and 2.8B), and Mamba [Gu and Dao, 2023] family (130M, 370M, 790M, 1.4B, and 2.8B). We also evaluate H3 models but exclude their results as their accuracy is always worse than Mamba models (see Appendix A).

We evaluate our quantized model on the WikiText-2 [Merity et al., 2016] generation task with 2048 sequence length. We also evaluate on four popular zero-shot tasks: PIQA [Bisk et al., 2020], WinoGrande [Sakaguchi et al., 2021], HellaSwag [Zellers et al., 2019], and LAMBADA (OpenAI). We use the LM Evaluation Harness [Gao et al., 2021] with default parameters for our experiments. For GPTQ quantization, we use default parameters with 64 samples from the WikiText-2 [Merity et al., 2016] training dataset as the calibration set. Following GPTQ, we quantize all linear layers in different models and keep the lm-head (and convolutions) in higher precision. All experiments are run on a single NVIDIA A100 GPU. A.

FP16 Case. First, we study the memory-accuracy trade-off in the baseline models. Figure 1 shows that Mamba outperforms OPT and Pythia of the same size in almost all cases. Pythia performs better than OPT in larger than 200M models. As Mamba outperforms all other models with the same size as expected, it motivates the use of such models in practice as they have better accuracy given a fixed memory size.

8-bit Case. Next, we compare the accuracy of our models in the presence of quantization. Figures 2 and 3 show the results of applying GPTQ and RtN with 8-bit (which is known to be moderated quantization) on the generation and zero-shot abilities of our models. Mamba could be quantized without a noticeable accuracy loss in 8-bit using GPTQ and there isn't a considerable difference in RtN and GPTQ quantization. In addition, Mamba still performs the best in 8-bit quantization in all model sizes on both WikiText and Zero-shot evaluations.

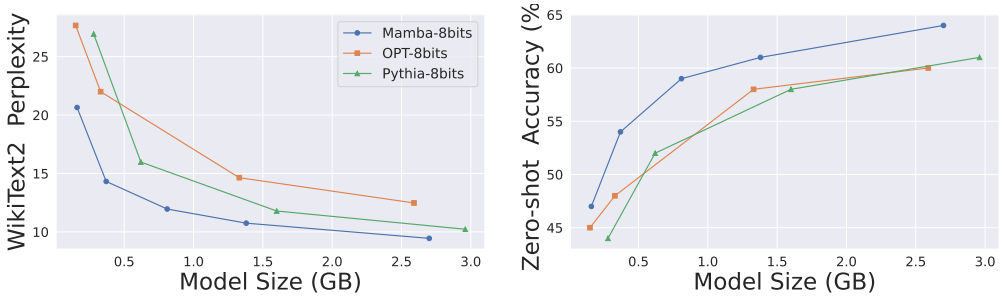

Figure 2: Accuracy of the 8-bit models on various tasks using GPTQ quantization. **Left:** WikiText-2 generation perplexity. **Right:** Average zero-shot accuracy. Mamba outperforms OPT and Pythia in all model sizes across all tasks.

4-bit Case. Next, we study extreme weight quantization (4-bit) with both RtN and GPTQ quantization. Figure 4 and 5 show our results of quantizing our models using GPTQ and RtN quantization. Unlike 8-bit, it is hard to preserve the accuracy of the model with 4-bits and RtN cannot preserve a reasonable accuracy in most of the models. However, GPTQ provides reasonable accuracy in the cost of needs for a calibration set.

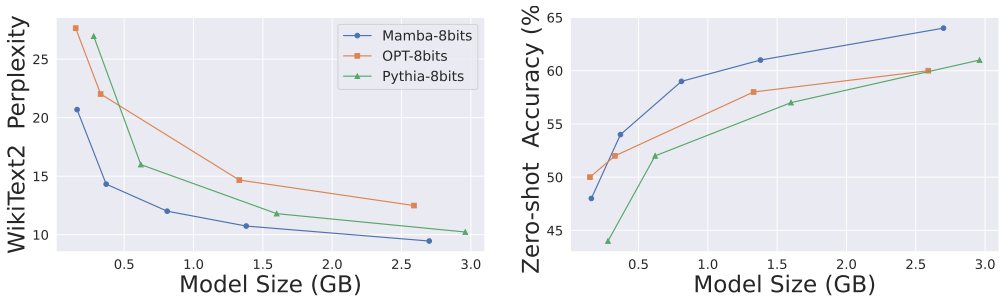

Figure 3: Accuracy of the 8-bit models on various tasks using RtN quantization. **Left:** WikiText-2 generation perplexity. **Right:** Average zero-shot accuracy. Mamba outperforms OPT and Pythia in all model sizes across all tasks.

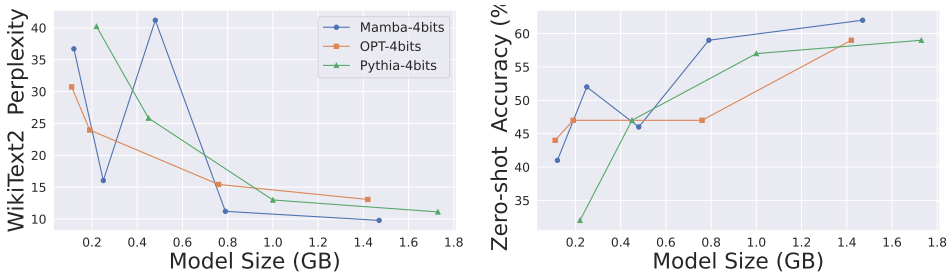

Figure 4: Accuracy of the 4-bit models on various tasks using GPTQ quantization. **Left:** WikiText-2 generation perplexity. **Right:** Average zero-shot accuracy. Mamba outperforms OPT and Pythia in all model sizes across all tasks.

Interestingly, we found that smaller Mamba models are extremely sensitive to quantization and they are no longer better than the OPT or Pythia with the same number of parameters. However, Mamba still performs the best in larger models (1.4B and 2.8B) on the same model sizes.

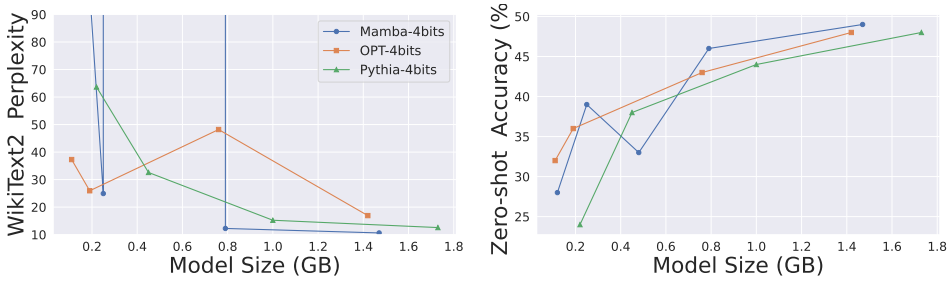

Figure 5: Accuracy of the 4-bit models on various tasks using RtN quantization. **Left:** WikiText-2 generation perplexity. **Right:** Average zero-shot accuracy. Mamba outperforms OPT and Pythia in all model sizes across all tasks.

4 Conclusion

In conclusion, this study demonstrates the potential of State Space Models, particularly Mamba models, as a scalable and memory-efficient alternative to traditional LLMs like OPT and Pythia under low-precision quantization. The results highlight that Mamba outperforms traditional LLMs of equivalent sizes, especially in the 8-bit quantization case, without significant accuracy loss. However, under extreme quantization (4-bit), Mamba models remain competitive only when scaled to larger sizes (above 1 billion parameters), whereas smaller Mamba models exhibit sensitivity to low-precision quantization. In all cases, the GPTQ quantization scheme proved more resilient, maintaining reasonable accuracy compared to the simpler Round-to-Nearest approach.

References

Carmen Amo Alonso, Jerome Sieber, and Melanie N Zeilinger. State space models as foundation models: A control theoretic overview. *arXiv preprint arXiv:2403.16899*, 2024.

Saleh Ashkboos, Ilia Markov, Elias Frantar, Tingxuan Zhong, Xincheng Wang, Jie Ren, Torsten Hoefler, and Dan Alistarh. Towards end-to-end 4-bit inference on generative large language models. *arXiv preprint arXiv:2310.09259*, 2023.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated llms. *arXiv preprint arXiv:2404.00456*, 2024.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR, 2023.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

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.

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. Spqr: A sparse-quantized representation for near-lossless llm weight compression. *arXiv preprint arXiv:2306.03078*, 2023.

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.

Daniel Y Fu, Tri Dao, Khaled K Saab, Armin W Thomas, Atri Rudra, and Christopher Ré. Hungry hungry hippos: Towards language modeling with state space models. *arXiv preprint arXiv:2212.14052*, 2022.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, et al. A framework for few-shot language model evaluation. *Version v0. 0.1. Sept*, 2021.

Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. In *Low-Power Computer Vision*, pages 291–326. Chapman and Hall/CRC, 2022.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. Awq: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978*, 2023.

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

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

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.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. Quip#: Even better llm quantization with hadamard incoherence and lattice codebooks. *arXiv preprint arXiv:2402.04396*, 2024.

Jiayin Wang, Fengran Mo, Weizhi Ma, Peijie Sun, Min Zhang, and Jian-Yun Nie. A user-centric benchmark for evaluating large language models, 2024.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2020.

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*, pages 38087–38099. PMLR, 2023.

Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in Neural Information Processing Systems*, 35:27168–27183, 2022.

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.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. Opt: Open pre-trained transformer language models, 2022.

A Detailed Results

Table 1 shows all results we use in this paper both for RtN and GPTQ quantization of Mamba, Mamba2, H3, OPT, and Pythia, models.

Model	Params.	Size (GB)			Baseline (PPL)	RtN (PPL)		GPTQ (PPL)	
		16-bit	8-bit	4-bit	16-bit	8-bit	4-bit	8-bit	4-bit
Mamba	130M	0.24GB	0.16GB	0.12GB	20.6	20.69	184.2	20.65	36.7
	370M	0.69GB	0.37GB	0.25GB	14.31	14.32	24.93	14.32	16.03
	790M	1.48GB	0.81GB	0.48GB	11.95	12.01	3.52e+25	11.96	41.19
	1.4B	2.56GB	1.38GB	0.79GB	10.75	10.74	12.25	10.75	11.2
	2.8B	5.16GB	2.7GB	1.47GB	9.45	9.46	10.62	9.45	9.78
Mamba2	130M	0.24GB	0.16GB	0.11GB	20.04	20.05	32.23	N.A	N.A
	370M	0.69GB	0.39GB	0.24GB	14.17	14.18	19.84	14.17	18.73
	780M	1.45GB	0.80GB	0.47GB	11.81	11.82	17.87	11.82	17.82
	1.3B	2.50GB	1.35GB	0.77GB	10.42	10.42	11.69	10.42	12.23
	2.7B	5.03GB	2.64GB	1.44GB	9.06	9.06	9.81	9.06	10.16
H3	125M	0.24GB	0.16GB	0.12GB	24.02	24.03	28.68	24.01	30
	355M	0.67GB	0.39GB	0.25GB	15.99	16.01	18.65	16.01	19.39
	1.3B	2.47GB	1.34GB	0.78GB	12.07	12.08	13.68	12.07	13.91
	2.7B	4.97GB	2.63GB	1.46GB	10.21	10.22	11.45	10.22	11.46
OPT	125M	0.23GB	0.15GB	0.11GB	27.67	27.65	37.28	27.67	30.72
	350M	0.62GB	0.33GB	0.19GB	22	22.02	25.94	22	23.96
	1.3B	2.45GB	1.33GB	0.76GB	14.62	14.67	48.19	14.64	15.42
	2.7B	4.94GB	2.59GB	1.42GB	12.47	12.49	16.92	12.48	13.06
Pythia	160M	0.4GB	0.28GB	0.22GB	26.9	26.96	63.65	26.94	40.22
	410M	0.95GB	0.62GB	0.45GB	15.95	15.99	32.59	15.98	25.83
	1.4B	2.83GB	1.60GB	1GB	11.79	11.8	15.23	11.79	12.98
	2.8B	5.42GB	2.96GB	1.73GB	20.23	10.23	12.57	10.23	11.11

Table 1: Detailed results of applying Round-to-Nearest and GPTQ quantizations on different models.

Table 2: Results of zero-shot tasks on SSM models with different precisions and quantization. For zero-shot tasks, we use HellaSwag (HS), LAMBADA (LA), PIQA (PQ), and WinoGrande (WG). We show the average (Avg.) of the tasks using Avg.

Model	Params.	Prec.	HS RtN/GPTQ	LA RtN/GPTQ	PQ RtN/GPTQ	WG RtN/GPTQ	Avg. RtN/GPTQ
Mamba	130M	FP16	0.31	0.44	0.64	0.52	0.48
		8	0.31/0.31	0.44/0.43	0.64/0.63	0.52/0.52	0.48/0.47
		4	0.17/0.28	0.00/0.21	0.54/0.60	0.52/0.53	0.28/0.41
	370M	FP16	0.37	0.55	0.69	0.55	0.54
		8	0.37/0.37	0.55/0.56	0.69/0.70	0.55/0.56	0.54/0.54
		4	0.26/0.37	0.17/0.50	0.62/0.68	0.53/0.54	0.39/0.52
	790M	FP16	0.42	0.62	0.72	0.56	0.58
		8	0.42/0.42	0.62/0.62	0.73/0.73	0.57/0.57	0.59/0.59
		4	0.25/0.36	0.01/0.34	0.55/0.61	0.51/0.52	0.33/0.46
	1.4B	FP16	0.45	0.65	0.74	0.61	0.61
		8	0.45/0.45	0.65/0.65	0.74/0.74	0.61/0.61	0.61/0.61
		4	0.33/0.44	0.27/0.60	0.66/0.72	0.58/0.59	0.46/0.59
	2.8B	FP16	0.49	0.69	0.75	0.63	0.64
		8	0.49/0.49	0.69/0.69	0.75/0.75	0.63/0.63	0.64/0.64
		4	0.37/0.48	0.33/0.66	0.69/0.75	0.62/0.63	0.49/0.62

Table 3: Results of zero-shot tasks on LLMs with different precisions and quantization. For zero-shot tasks, we use HellaSwag (HS), LAMBADA (LA), PIQA (PQ), and WinoGrande (WG). We show the average (Avg.) of the tasks using Avg.

Model	Params.	Prec.	HS RtN/GPTQ	LA RtN/GPTQ	PQ RtN/GPTQ	WG RtN/GPTQ	Avg. RtN/GPTQ
OPT	125M	FP16	0.29	0.38	0.63	0.63	0.50
		8	0.29/0.29	0.38/0.37	0.63/0.63	0.63/0.50	0.50/0.45
		4	0.16/0.28	0.04/0.36	0.57/0.61	0.50/0.51	0.32/0.44
	350M	FP16	0.32	0.56	0.65	0.53	0.52
		8	0.32/0.32	0.56/0.45	0.65/0.64	0.53/0.52	0.52/0.48
		4	0.19/0.33	0.12/0.41	0.59/0.64	0.52/0.52	0.36/0.47
	1.3B	FP16	0.41	0.58	0.72	0.60	0.58
		8	0.41/0.41	0.58/0.58	0.72/0.72	0.60/0.60	0.58/0.58
		4	0.28/0.34	0.25/0.35	0.66/0.68	0.54/0.54	0.43/0.47
	2.7B	FP16	0.46	0.64	0.74	0.61	0.61
		8	0.46/0.46	0.64/0.64	0.73/0.73	0.60/0.60	0.60/0.60
		4	0.33/0.43	0.31/0.58	0.68/0.73	0.60/0.60	0.48/0.59
Pythia	160M	FP16	0.28	0.33	0.62	0.50	0.43
		8	0.28/0.28	0.33/0.33	0.61/0.61	0.49/0.49	0.44/0.44
		4	0.15/0.28	0.01/0.17	0.48/0.50	0.32/0.32	0.24/0.32
	410M	FP16	0.34	0.52	0.67	0.53	0.52
		8	0.33/0.34	0.51/0.52	0.67/0.67	0.53/0.54	0.52/0.52
		4	0.21/0.32	0.19/0.36	0.61/0.66	0.52/0.52	0.38/0.47
	1.4B	FP16	0.40	0.62	0.71	0.58	0.60
		8	0.40/0.40	0.61/0.62	0.69/0.71	0.57/0.58	0.57/0.58
		4	0.27/0.40	0.29/0.60	0.65/0.70	0.54/0.56	0.44/0.57
	2.8B	FP16	0.45	0.65	0.74	0.59	0.61
		8	0.45/0.45	0.65/0.65	0.74X/0.74	0.59/0.59	0.61/0.61
		4	0.32/0.43	0.32/0.62	0.68/0.73	0.58/0.58	0.48/0.59