LoftQ: LoRA-Fine-Tuning-Aware Quantization for Large Language Models

Anonymous Author(s)
Affiliation
Address
email

Abstract

Quantization is an indispensable technique for serving Large Language Models (LLMs) and has recently found its way into LoRA fine-tuning (Dettmers et al., 2023). In this work we focus on the scenario where quantization and LoRA fine-tuning are applied together on a pre-trained model. In such cases it is common to observe a consistent gap in the performance on downstream tasks between full fine-tuning and quantization plus LoRA fine-tuning approach. In response, we propose LoftQ (**Lo**RA-**F**ine-**T**uning-aware **Q**uantization), a novel quantization framework that simultaneously quantizes an LLM and finds a proper low-rank initialization for LoRA fine-tuning. Such an initialization alleviates the discrepancy between the quantized and full-precision model and significantly improves generalization in downstream tasks. We evaluate our method on natural language understanding, question answering, summarization, and natural language generation tasks. Experiments show that our method is highly effective and outperforms existing quantization methods, especially in the challenging 2-bit and 2/4-bit mixed precision regimes. We will release our code.

1 Introduction

The advent of Pre-trained Language Models (PLMs) has marked a transformative shift in the field of Natural Language Processing (NLP), offering versatile solutions across various applications (He et al., 2021b; Lewis et al., 2019; Touvron et al., 2023). However, the extensive computational and memory demands of these models pose significant challenges, especially in real-world deployments where resources are often constrained and need to be shared among many users.

To mitigate the extensive storage requirements of pre-trained models, quantization serves as a pivotal compression technique (Zafrir et al., 2019; Shen et al., 2020; Bai et al., 2022; Dettmers et al., 2022), converting high-precision numerical values into a discrete set of values. Typically, model parameters, originally stored in a 16-bit float format, are transformed into a 4-bit integer format through quantization, resulting in a substantial 75% reduction in storage overhead.

When quantizing pre-trained models, however, practitioners often concentrate primarily on the quantization technique, inadvertently neglecting the importance of subsequent task adaption, e.g., LoRA fine-tuning (Dettmers et al., 2023; Diao et al., 2023). For example, QLoRA inherits the fixup initialization (Zhang et al., 2019) used in LoRA, which (Dettmers et al., 2023) attaches zero initialized low-rank adapters to the quantized pre-trained model. The inevitable discrepancy introduced by quantization during the approximation of the original high-precision numbers, a scenario particularly pronounced in low-bit situations such as the 2-bit regime, can adversely impact the initialization of LoRA fine-tuning. This deviation often results in an inferior fine-tuning performance.

In this paper, we introduce a novel quantization framework, called **Lo**RA-**F**ine-**T**uning-aware **Q**uantization (LoftQ). It is designed specifically for pre-trained models that require quantization and LoRA fine-tuning. This framework actively integrates low-rank approximation, working in tandem with quantization to jointly approximate the original high-precision pre-trained weights. This synergy significantly enhances alignment with the original pre-trained weights. Consequently, our method provides an advantageous initialization point for subsequent LoRA fine-tuning, leading to improvements in downstream tasks.

We evaluate our quantization framework by conducting extensive experiments on downstream tasks. Experiments show that LoftQ consistently outperforms QLoRA across all precision levels. LoftQ is also competitive with different quantization methods, e.g., NormalFloat (Dettmers et al., 2023) and the uniform quantization.

2 Method

We propose **Lo**RA-**F**ine-**T**uning-aware **Q**uantization (LoftQ), a quantization framework for LLMs. It alternatively applies quantization and low-rank approximation to approximate original pre-trained weights. This quantization framework provides a promising initialization for LoRA fine-tuning, which alleviates the quantization discrepancy in QLoRA and improves generalization in downstream tasks significantly.

2.1 LoRA-Aware Quantization

We use an N-bit quantized weight $Q \in \mathbb{R}_N^{d_1 \times d_2}$ and low-rank approximations $A \in \mathbb{R}^{d_1 \times r}, B \in \mathbb{R}^{d_2 \times r}$ to approximate the original high-precision pre-trained weight $W \in \mathbb{R}^{d_1 \times d_2}$ as the initialization of LoRA fine-tuning. Specifically, before fine-tuning, we initialize the network by minimizing the following objective:

$$\min_{Q,A,B} \left\| W - Q - AB^\top \right\|_{\mathrm{F}}, \tag{1}$$

where $\|\cdot\|_F$ denotes the Frobenious norm. This objective in (1) takes LoRA fine-tuning into consideration by jointly optimizing the initial values of the quantized backbone Q and low-rank adapters A, B. Contrarily, practitioners typically convert the pre-trained weight W into a quantized weight Q outright and initialize the low-rank adapters by $A \sim \mathcal{N}(0, \sigma^2),\ B = 0$, neglecting the subsequent LoRA fine-tuning process. This oversight leads to notable performance degradation in downstream tasks arising from the quantization discrepancy.

2.2 Alternating Optimization

We solve the minimization problem in (1) by alternating between quantization and singular value decomposition (SVD). Initially, we set A_0, and B_0 equal to 0. The total alternating steps are T.

Quantization. At the t-th step, we quantize the difference between the original pre-trained weight W and the low-rank approximation $A_{t-1}B_{t-1}^\top$ from the last step to obtain the quantized weight Q_t by

$$Q_t = q_N(W - A_{t-1}B_{t-1}^\top), \tag{2}$$

where $q_N(\cdot)\colon \mathbb{R}^{m\times n} \mapsto \mathbb{R}_N^{m \times n}$ is a quantization function that maps a high-precision matrix into a quantized matrix, and $\mathbb{R}_N : \{\mathcal{T}[i] \in \mathbb{R} | 0 \le i < 2^N\}$.

We remark that our algorithm is compatible with different quantization functions $q_N(\cdot)$. We apply NF4 and the uniform quantization in Section 3 as examples. We also remark that Q_t is not an exact solution of the minimization in (1), given the fixed $A_{t-1}B_{t-1}^\top$, but it is an efficient approximation.

SVD. After obtaining the t-th quantized weight Q_t, SVD is applied to the residual of the quantization denoted by $R_t = W - Q_t$ by

$$R_t = \sum_{i=1}^{d} \sigma_{t,i} u_{t,i} v_{t,i}^\top, \tag{3}$$

where $d = \min\{d_1, d_2\}$, $\sigma_{t,1} \ge \sigma_{t,2} \ge ... \ge \sigma_{t,d}$ are the singular values of R_t, $u_{t,i}$'s and $v_{t,i}$'s are the associated left and right singular vectors of R_t. We then obtain a rank-r approximation of R_t by $A_t B_t^\top$, where

$$\begin{aligned} A_t &= [\sqrt{\sigma_{t,1}} u_{t,1}, ..., \sqrt{\sigma_{t,r}} u_{t,r}], \\ B_t &= [\sqrt{\sigma_{t,1}} v_{t,1}, ..., \sqrt{\sigma_{t,r}} v_{t,r}]. \end{aligned} \tag{4}$$

It is worth noting that $T = 1$ is a special case where Q_1 is the exact quantized weight obtained by QLoRA, and low-rank approximations A_1, B_1 are obtained by the SVD of the quantization residual $W - Q_1$. $T = 1$ is sufficient to mitigate the quantization discrepancy, and alternating optimization helps to find a closer initialization to the pre-trained weight W, which further improves the performance.

We remark that the computational cost of LoftQ is negligible because it is applied to individual weight matrices and therefore can be executed in parallel. Also, it only requires being applied once to a pre-trained model, and the initialization obtained can be reused for various downstream tasks

We then use Q_T, A_T, B_T obtained by the alternating optimization above as LoRA fine-tuning initialization. We freeze the quantized backbone Q_T and optimize the low-rank adapters, starting from A_T, B_T, with an efficient optimization algorithm, e.g., AdamW (Loshchilov & Hutter, 2017).

3 Experiments

We evaluate our method on NLU and NLG tasks. We apply LoftQ for quantizing DeBERTaV3-base (He et al., 2021b), BART-large (Lewis et al., 2019), and LLAMA-2 series (Touvron et al., 2023).

Implementation Details. Following the prior works of LoRA variants (Zhang et al., 2023; He et al., 2021a), we freeze all the backbone weight matrices and add low-rank adapters to weight matrices in MHA and FFN of all layers. We quantize the weight matrices that are attached by low-rank adapters.

Quantization Methods. We apply two quantization methods to demonstrate LoftQ is compatible with different quantization functions. We apply uniform quantization and NF4 quantization [1] in our experiments. We perform 2-bit and 4-bit quantization on all models, achieving compression ratios of 25-30% and 15-20% at the 4-bit and 2-bit levels, respectively. The compression ratios and trainable parameter ratios for all models are detailed in the Appendix A.

Baselines. We compare LoftQ with baseline methods of Full fine-tuning, Full precision LoRA (LoRA), and QLoRA. Specific introductions of baseline methods are given in Appendix B

3.1 Encoder-only Model: DeBERTaV3

Models, Datasets, Implementations, and Results We quantize the DeBERTaV3-base (He et al., 2021b) with LoftQ, then finetune the model on the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2019), SQuADv1.1 (Rajpurkar et al., 2016), and ANLI (Nie et al., 2019). We show the implementation details in Appendix E.3. Table 1 summarize the results for 2-bit quantization on the GLUE, SQuADv1.1, and ANLI datasets, by NF2 quantization. Our method consistently outperforms QLoRA on all settings with respect to different ranks, and datasets. The 4-bit quantization experiment results are presented in Appendix E.1 as both LoftQ and QLoRA achieve performance close to full fine-tuning. We also show our method excels over baseline using uniform quantization shown in Table 8, indicating our method is applicable to different methods.

Table 1: Results with 2-bit LoftQ of DeBERTaV3-base models on GLUE development set, SQuADv1.1 development set, ANLI test set using **NF2 quantization**. *N.A.* indicates the model does not converge. The best results on each dataset are shown in **bold**.

Rank	Method	MNLI m / mm	QNLI Acc	RTE Acc	SST Acc	MRPC Acc	CoLA Matt	QQP Acc	STSB P/S Corr	SQuAD EM/F1	ANLI Acc
	Full FT	90.4/90.5	94.6	85.1	95.1	89.9/93.6	69.9	92.0/89.4	91.7/91.1	87.3/93.1	59.8
16	LoRA	90.5/90.6	94.8	85.2	95.0	89.9/93.6	69.8	92.0/89.4	91.6/91.0	87.0/93.1	60.2
16	QLoRA	75.4/75.6	82.4	55.9	86.5	73.8/82.8	N.A.	86.8/82.3	83.0/82.8	61.5 / 71.2	N.A.
	LoftQ	**84.7/85.1**	**86.6**	**61.4**	**90.2**	**83.8/88.6**	**37.4**	**90.3/86.9**	**87.1/86.9**	**81.5/88.6**	**47.1**
32	QLoRA	78.5/78.7	80.4	56.7	86.9	73.8/82.7	N.A.	87.1/82.7	83.6/83.3	64.6/73.8	N.A.
	LoftQ	**86.0/86.1**	**89.9**	**61.7**	**92.0**	**83.6/87.2**	**47.5**	**91.0/87.9**	**87.5/87.0**	**82.9/89.8**	**49.0**

3.2 Encoder-Decoder Model: BART

Models, Datasets, Implementations, and Results We quantize BART-large model (Lewis et al., 2020) with LoftQ, then finetune and evaluate the model on two commonly used summarization datasets: XSum (Narayan et al., 2018) and CNN/DailyMail(Hermann et al., 2015). The implementation details are given in Appendix F. The 2-bit quantization results are shown in Table 2. Our observation is consistent with the NLU experiments, that LoftQ demonstrates the convergence to reasonable results, while QLoRA does not converge. This indicates our method is robuster by nar-

[1]We abbreviate NF4 for NormalFloat quantization used in QLoRA. NF2 is its 2-bit variant

rowing the initialization gap. We remark that our method is also successful within 4-bit quantization precision. Results shown in Table 13

Table 2: Results with 2-bit LoftQ of BART-large on XSum and CNN/DailyMail using **NF2 quantization**. *N.A.* indicates the model does not converge. We report ROUGE-1/2/L.

Rank	Method	XSum	CNN/DailyMail
-	Lead-3	16.30/1.60/11.95	40.42/17.62/36.67
	Full FT	45.14/22.27/37.25	44.16/21.28/40.90
8	FP LoRA	43.40/20.20/35.20	44.72/21.58/41.84
	QLoRA	N.A.	N.A.
	LoftQ	39.63/16.65/31.62	42.24/19.44/29.04
16	FP LoRA	43.95/20.72/35.68	45.03/21.84/42.15
	QLoRA	N.A.	N.A.
	LoftQ	40.81/17.85/32.80	42.52/19.81/39.51

3.3 Decoder-only Model: LLAMA-2

Models, Datasets, Implementations, and Results We quantize LLAMA-2-7b and LLAMA-2-13b (Touvron et al., 2023) with LoftQ. We then fine-tune and evaluate the models on two NLG datasets: GSM8K (Cobbe et al., 2021) and WikiText-2 (Merity et al., 2016). Please see Appendix G for more details about the datasets. The implementation details are given in Appendix G. Table 3 presents a summary of our experiments on LLAMA-2-7b and LLAMA-2-13b using 2-bit, 4-bit, and mixed-precision NormalFloat quantization methods on WikiText-2 and GSM8K datasets. In WikiText-2, our method consistently outperforms QLoRA across all quantization precision settings on both models. When dealing with the challenging 2-bit precision, where QLoRA fails to converge, LoftQ manages to achieve a perplexity of 7.85. In GSM8K, our method also achieves better or on par performance compared to QLoRA across different model sizes and quantization precision levels.

We also explore mixed-precision quantization where matrices in the first 4 layers are quantized using 4 bits, and the rest matrices remain 2 bits. We witness a remarkable 5.9% accuracy boost on the GSM8K dataset using LLAMA-2-7b and a 12.7% boost using LLAMA-2-13b. This result underscores the potential of LoftQ for complex mixed-precision quantization scenarios.

Table 3: Results of LoftQ using 4-bit, 2.25 bit and 2-bit for LLAMA-2 series on WikiText-2 and GSM8K. 2.25-bit indicates mixed-precision NormalFloat quantization: 4-bit precision for the first 4 layers and 2-bit precision for the rest of layers. We report the perplexity (the smaller the better) for WikiText-2 and accuracy for GSM8K. The rank of low-rank adapters is 64. *N.A.* indicates the model does not converge.

Method	Bit	LLAMA-2-7b		LLAMA-2-13b	
		WikiText-2↓	GSM8K↑	WikiText-2↓	GSM8K↑
LoRA	16	5.08	36.9	5.12	43.1
QLoRA	4	7.41	**35.1**	5.22	39.9
LoftQ	4	**5.24**	35.0	**5.16**	**45.0**
QLoRA	2.25	N.A.	N.A.	N.A.	N.A.
LoftQ	2.25	**6.13**	**26.5**	**5.45**	**38.1**
QLoRA	2	N.A	N.A.	N.A.	N.A.
LoftQ	2	**7.85**	**20.9**	**7.69**	**25.4**

4 Conclusion

We propose LoftQ, a quantization framework for LLMs, which alternatively applies quantization and low-rank approximation to the original high-precision pre-trained weights, to obtain an initialization for the subsequent LoRA fine-tuning. Experiments on natural language understanding, question answering, summarization, and natural language generation show that our framework remarkably surpasses existing methods, e.g., QLoRA, for quantizing encoder-only, encoder-decoder, and decoder-only models. We have not observed our method exhibiting worse performance over QLoRA. Moreover, our quantization framework demonstrates effectiveness and robustness particularly in low-bit quantization regimes, e.g., the 2-bit level.

References

Haoli Bai, Lu Hou, Lifeng Shang, Xin Jiang, Irwin King, and Michael R Lyu. Towards efficient post-training quantization of pre-trained language models. *Advances in Neural Information Processing Systems*, 35:1405–1418, 2022.

Roy Bar-Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. The second pascal recognising textual entailment challenge. 2006.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. The fifth pascal recognizing textual entailment challenge. In *TAC*, 2009.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pp. 1–14, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/S17-2001.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, 2007.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm. int8 (): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*, 2022.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023.

Shizhe Diao, Rui Pan, Hanze Dong, Ka Shun Shum, Jipeng Zhang, Wei Xiong, and Tong Zhang. Lmflow: An extensible toolkit for finetuning and inference of large foundation models. *arXiv preprint arXiv:2306.12420*, 2023.

William B. Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pp. 1–9, Prague, June 2007. Association for Computational Linguistics.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. *arXiv preprint arXiv:2110.04366*, 2021a.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*, 2021b.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. *Advances in neural information processing systems*, 28, 2015.

Hector Levesque, Ernest Davis, and Leora Morgenstern. The winograd schema challenge. In *Thirteenth international conference on the principles of knowledge representation and reasoning*, 2012.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7871–7880, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703.

Yixiao Li, Yifan Yu, Qingru Zhang, Chen Liang, Pengcheng He, Weizhu Chen, and Tuo Zhao. Losparse: Structured compression of large language models based on low-rank and sparse approximation. *arXiv preprint arXiv:2306.11222*, 2023.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *ArXiv*, abs/1808.08745, 2018.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. Adversarial nli: A new benchmark for natural language understanding. *ArXiv*, abs/1910.14599, 2019. URL `https://api.semanticscholar.org/CorpusID:207756753`.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary 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. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264.

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Q-bert: Hessian based ultra low precision quantization of bert. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8815–8821, 2020.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

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*, 2023.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*, 2019.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641, 2019. doi: 10.1162/tacl_a_00290.

Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1112–1122, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1101.

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)*, pp. 36–39. IEEE, 2019.

Hongyi Zhang, Yann N Dauphin, and Tengyu Ma. Fixup initialization: Residual learning without normalization. *arXiv preprint arXiv:1901.09321*, 2019.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*, 2023.

A Model Compression Ratio

Table 4: Compression ratios of backbones.

Model	Compression ratio (%)	Trainable ratio (%)	Rank	Bits	Quantization method
DeBERTaV3-base	15.6	3.1	16	2	uniform
DeBERTaV3-base	18.8	6.3	32	2	uniform
DeBERTaV3-base	17.2	3.1	16	2	nf2
DeBERTaV3-base	20.4	6.3	32	2	nf2
BART-large	15.3	1.2	8	2	nf2
BART-large	16.7	2.5	16	2	nf2
BART-large	27.8	1.2	8	4	nf4
BART-large	29.0	2.5	16	4	nf4
BART-large	26.2	1.2	8	4	uniform
BART-large	27.5	2.5	16	4	uniform
LLAMA-2-7b	16.6	2.4	64	2	nf2
LLAMA-2-7b	29.0	2.4	64	4	nf4
LLAMA-2-13b	16.0	1.9	64	2	nf2
LLAMA-2-13b	28.5	1.9	64	4	nf4

B Baseline Methods

- *Full fine-tuning* is the most common approach for adapting a pre-trained model to down-stream tasks. The model is initialized with pre-trained weights and all parameters are updated through an SGD-type optimization method.
- *Full precision LoRA (LoRA)* is a lightweight method for task adaptation, where it stores the backbone using 16-bit numbers and optimizes the low-rank adaptors only. The adaptors are applied to the same matrices as in LoftQ.
- *QLoRA* is similar to *LoRA* except the backbone is quantized into low-bit regime. The low-rank adapters are zero initialized using and are applied to the same matrices as in LoftQ.

C GLUE Dataset Statistics

We present the dataset statistics of GLUE Wang et al. (2019) in the following table.

Corpus	Task	#Train	#Dev	#Test	#Label	Metrics
Single-Sentence Classification (GLUE)						
CoLA	Acceptability	8.5k	1k	1k	2	Matthews corr
SST	Sentiment	67k	872	1.8k	2	Accuracy
Pairwise Text Classification (GLUE)						
MNLI	NLI	393k	20k	20k	3	Accuracy
RTE	NLI	2.5k	276	3k	2	Accuracy
QQP	Paraphrase	364k	40k	391k	2	Accuracy/F1
MRPC	Paraphrase	3.7k	408	1.7k	2	Accuracy/F1
QNLI	QA/NLI	108k	5.7k	5.7k	2	Accuracy
Text Similarity (GLUE)						
STS-B	Similarity	7k	1.5k	1.4k	1	Pearson/Spearman corr

Table 5: Summary of the GLUE benchmark.

GLUE includes two single-sentence classification tasks: SST-2 (Socher et al., 2013) and CoLA (Warstadt et al., 2019), and three similarity and paraphrase tasks: MRPC (Dolan & Brockett, 2005), STS-B (Cer et al., 2017), and QQP. GLUE also includes four natural language inference tasks in GLUE: MNLI (Williams et al., 2018), QNLI (Rajpurkar et al., 2016), RTE (Dagan et al., 2007; Bar-Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009), and WNLI (Levesque et al., 2012).

D Decompose Time

We report the execution time of LoftQ applying to a single weight matrix. The time is tested on Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz.

Table 6: Execution time of LoftQ applying to different weight matrices.

Model	Size	Step T	Quantization method	Time
DeBERTaV3-base	768*768	5	Uniform	1s
BART-large	1024*1024	5	NF4	1s
LLAMA-2-7b	4096*4096	5	NF4	21s
LLAMA-2-13b	5120*5120	5	NF4	43s

E Natural Language Understanding

E.1 GLUE with 4-bit

We show the 4-bits results in the Table 7. Both methods can achieve performance close to full-finetuning.

Table 7: Results with 4-bit LoftQ of DeBERTaV3-base models on GLUE development set using NF4 quantization. We report the median over four seeds. Results with N.A. indicate the model does not converge. The best results on each dataset are shown in bold

Method	**Rank**	**MNLI** m / mm	**SST-2** Acc	**QNLI** Acc	**ANLI** Acc
Full FT	-	90.5/90.6	95.3	94.0	59.8
QLoRA	32	89.9/89.9	**95.3**	**94.2**	59.4
LoftQ	32	**89.9/90.0**	**95.3**	94.1	**59.9**

E.2 GLUE with uniform 2-bit quantization

Table 8: Results with 2-bit LoftQ of DeBERTaV3-base models on GLUE development set, SQuADv1.1 development set using **Uniform quantization** . We report the median over four seeds. *N.A.* indicates the model does not converge. The best results on each task are shown in **bold**.

Rank	**Method**	**MNLI** m / mm	**QNLI** Acc	**RTE** Acc	**SST** Acc	**MRPC** Acc	**CoLA** Acc	**QQP** Mcc	**STSB** P/S Corr	**SQuAD** Em/F1
-	Full FT	90.5/90.6	94.0	82.0	95.3	89.5/93.3	69.2	92.4/89.8	91.6/91.1	88.5/92.8
16	LoRA	90.4/90.5	94.6	85.1	95.1	89.9/93.6	69.9	92.0/89.4	91.7/91.1	87.3/93.1
16	QLoRA	76.5/76.3	83.8	56.7	86.6	75.7/84.7	N.A.	87.1/82.6	83.5/83.4	69.5/77.6
	LoftQ	**87.3/87.1**	**90.6**	**61.1**	**94.0**	**87.0/90.6**	**59.1**	**90.9/88.0**	**87.9/87.6**	**84.4/91.2**
32	QLoRA	79.9/79.5	83.7	57.8	86.9	76.5/84.5	N.A.	88.6/84.7	84.1/84.0	71.6/80.2
	LoftQ	**88.0/88.1**	**92.2**	**63.2**	**94.7**	**87.5/91.2**	**60.5**	**91.3/88.3**	**89.5/89.2**	**85.2/91.6**

E.3 Training Details

Implementation Details. The implementation of LoftQ is based on publicly available Huggingface (Paszke et al., 2019) code-base [2].

Hyper-parameter Details. We select the learning rate of $\{1 \times 10^{-5}, 5 \times 10^{-5}, 1 \times 10^{-4}, 5 \times 10^{-4}\}$, and use the selected learning rate for both uniform quantization experiments and nf2 quantization experiments. We use batch size of 32 for all GLUE tasks and ANLI. We use batch size of 16 for SQuADv1.1. We use LoftQ of 5 iterations for all GLUE tasks.

Table 9 summarizes the detailed hyperparameters for each task used in training DeBERTaV3-base using uniform quantization. Table 10 summarizes the detailed hyperparameters for each task used in training DeBERTaV3-base using nf2 quantization.

[2] `https://github.com/huggingface/transformers/tree/main/examples/pytorch`

Table 9: Hyper-parameter setup of LoftQ for GLUE benchmark for training DeBERTaV3-base using uniform quantization.

Hyper-parameter	MNLI	RTE	QNLI	MRPC	QQP	SST-2	CoLA	STS-B	SQuADv1.1	ANLI
# epochs	5	20	10	60	10	10	60	60	10	12
Learning rate	1×10^{-4}	5×10^{-4}	5×10^{-5}	1×10^{-4}	5×10^{-5}	5×10^{-5}	5×10^{-5}	5×10^{-5}	5×10^{-5}	5×10^{-5}

Table 10: Hyper-parameter setup of LoftQ for GLUE benchmark for training DeBERTaV3-base using nf2 quantization.

Hyper-parameter	MNLI	RTE	QNLI	MRPC	QQP	SST-2	CoLA	STS-B	SQuADv1.1	ANLI
# epochs	5	20	10	60	10	10	60	60	10	12
Learning rate	1×10^{-4}	5×10^{-5}	5×10^{-5}	1×10^{-4}	5×10^{-5}	5×10^{-5}	5×10^{-5}	1×10^{-4}	5×10^{-5}	5×10^{-5}

F Summarization

F.1 Training Details

We set the batch size as 32, the number of training epoch as 10. We choose Adam as the optimizer and try learning rate from$\{1\times10^{-5}, 5\times10^{-5}, 7\times10^{-5}, 2\times10^{-4}, , 3\times10^{-4}, 4\times10^{-4}\}$. We show the optimal learning rate for different settings in Table We use LoftQ of 1 iteration for all BART-large experiments. Table 11 and Table 12 summarize the learning rate for CNN/DailyMail and XSum

Table 11: Hyper-parameter setup of LoftQ BART-large on CNN/DailyMail

Hyperparameter	NF4		4-bit Uniform		NF2	
	rank8	rank16	rank8	rank16	rank8	rank16
Learning rate	2e-4	2e-4	2e-4	3e-4	2e-4	2e-4

Table 12: Hyper-parameter setup of LoftQ BART-large on XSum

Hyperparamete	NF4		4-bit Uniform		NF2	
	rank8	rank16	rank8	rank16	rank8	rank16
Learning rate	2e-4	2e-4	2e-4	2e-4	2e-4	2e-4

F.2 BART-large experiments with NF4 quantization

Table 13: Results with 4-bit LoftQ of BART-large on XSum and CNN/DailyMail. We report ROUGE-1/2/L, the higher the better. *Lead-3* means choosing the first 3 sentences as the summary. *N.A.* indicates the model does not converge. *Full FT* refers to the full fine-tuning where all parameters are tuned. We report the median over five seeds.

Quantization	Rank	Method	XSum	CNN/DailyMail
-	-	Lead-3	16.30/1.60/11.95	40.42/17.62/36.67
		Full FT	45.14/22.27/37.25	44.16/21.28/40.90
	8	LoRA	43.40/20.20/35.20	44.72/21.58/41.84
	16	LoRA	43.95/20.72/35.68	45.03/21.84/42.15
NF4	8	QLoRA	42.91/19.72/34.82	43.10/20.22/40.06
		LoftQ	**44.08/20.72/35.89**	**43.81/20.95/40.84**
	16	QLoRA	43.29/20.05/35.15	43.42/20.62/40.44
		LoftQ	**44.51/21.14/36.18**	**43.96/21.06/40.96**

G Natural Language Generation

We set the batch size as 32 for WikiText-2 and 16 for GSM8K. We train 2 epochs on WikiText-2 and 6 epochs on GSM8K. We select learning rate from$\{1\times10^{-5}, 5\times10^{-5}, 7\times10^{-5}, 1\times10^{-4}, , 3\times$

$10^{-4}, 4 \times 10^{-4}\}$. We use five iterations for all experiments. Specific settings are summarized as below

Table 14: Hyper-parameter setup of LoftQ LLAMA-2-series on GSM8K

Model	Hyperparameter	4-bit NF4	2-bit NF2	Mixed-precision
LLAMA-2-7b	learning rate	1e-4	1e-4	3e-4
LLAMA-2-13b	learning rate	1e-4	1e-4	3e-4

Table 15: Hyper-parameter setup of LoftQ LLAMA-2-series on WikiText-2

Model	Hyperparameter	4-bit NF4	2-bit NF2	Mixed-precision
LLAMA-2-7b	learning rate	1e-4	1e-4	3e-4
LLAMA-2-13b	learning rate	1e-4	1e-4	3e-4

H Comparison to Pruning

Pruning is also a widely used compression method. Here we compare LoftQ with the state-of-the-art pruning method Li et al. (2023). We show the comparison in Table 16. We can see our method significantly outperforms the pruning methods on DeBERTaV3-base model. We also remark that LoftQ can consistently reduce the memory of both training and storage. In contrast, pruning requires training the entire full-precision matrix, which implies that it can not achieve any memory savings during the training stage.

Table 16: Results of LoftQ using 2-bits uniform quantization compared with LoSparse with DeBERTaV3-base models on some of GLUE development sets. Here *Ratio* is the proportion of total remaining weights. Results with *N.A.* indicate the model does not converge.

Method	**Ratio**	**MNLI** m / mm	**SST-2** Acc	**QNLI** Acc
Full FT	100%	90.5 / 90.6	95.3	94.0
LoSparse	15%	83.3/82.9	87.6	90.4
	20%	84.5/83.8	91.7	88.6
LoftQ	15.6%	**87.3/87.1**	**94.0**	**90.6**
	18.8%	**88.0/88.1**	**94.7**	**92.4**