QuAILoRA: Quantization-Aware Initialization for LoRA

Neal Lawton
lawtneal@amazon.com

Aishwarya Padmakumar
padmakua@amazon.com

Judith Gaspers
gaspers@amazon.com

Jack FitzGerald
jgmf@amazon.com

Anoop Kumar
anooamzn@amazon.com

Greg Ver Steeg
gssteeg@amazon.com

Aram Galstyan
argalsty@amazon.com

Abstract

QLoRA reduces the memory-cost of fine-tuning a large language model (LLM) with LoRA by quantizing the base LLM. However, quantization introduces quantization errors that negatively impact model performance after fine-tuning. In this paper we introduce QuAILoRA, a quantization-aware initialization for LoRA that mitigates this negative impact by decreasing quantization errors at initialization. Our method spends a small amount of computational overhead to compute this quantization-aware initialization, without increasing the memory-cost of fine-tuning. We evaluate our method on several causal language modeling and downstream evaluation tasks using several different model sizes and families. We observe that almost all LLMs fined-tuned with QuAILoRA achieve better validation perplexity. When evaluated on downstream tasks, we find that QuAILoRA yields improvements proportional to the negative effect of quantization error. On average, applying QuAILoRA to 4-bit QLoRA models yields 75% of the validation perplexity decrease and 86% of the downstream task accuracy increase as doubling the quantization precision to 8-bit, without increasing GPU memory utilization during fine-tuning.

1 Introduction

Fine-tuning state-of-the-art large language models (LLMs) requires a large amount of computational resources due to their increasingly large size. QLoRA [Dettmers et al., 2023], a quantized version of LoRA [Hu et al., 2021], reduces the memory-cost of fine-tuning sufficiently to fine-tune LLMs on the order of 70B parameters on a single GPU, making fine-tuning much more convenient and accessible. Although quantization greatly reduces memory costs, it also introduces quantization errors that negatively impact the task performance of the model after fine-tuning. In this paper we propose Quantization-Aware Initialization for LoRA (QuAILoRA), a method for initializing the LoRA matrices of a QLoRA model to reduce quantization errors. When fine-tuning a model with QLoRA, each parameter matrix of the fine-tuned model takes the form $Q + AB^\top$, where Q is the quantization of the base parameter matrix W and $AB^\top$ is the low-rank LoRA update. Typically the matrix A is initialized random normal and the matrix B is initialized zero so that the input-output mapping of the QLoRA model is the same as the quantized base model at initialization. Instead, we propose spending a small amount of computational overhead to find an initialization for the LoRA matrices so that the input-output mapping of the QLoRA model is more similar to the full-precision base model at initialization.

4th NeurIPS Efficient Natural Language and Speech Processing Workshop (ENLSP-IV 2024).

We conduct an extensive set of experiments and establish that QuAILoRA (1) is robust to the choice of calibration set; (2) yields better validation perplexity than the baseline initialization across many model families and sizes on several causal LM tasks; (3) yields consistently positive results on downstream task evaluations for smaller, lower-precision quantized LLaMA models. Additionally, we establish that our method is increasingly effective with larger LoRA ranks and does not appear to affect the rate of convergence of fine-tuning compared to the baseline initialization. QuAILoRA provides the largest benefit when the negative effect of quantization error is significant.

2 Related Work

Our method improves the performance of QLoRA, a parameter-efficient fine-tuning (PEFT) technique, but many other PEFT methods exist in the literature, including adapter-based strategies [Houlsby et al., 2019], BitFit [Zaken et al., 2021], diff-pruning [Guo et al., 2020], NAS for PEFT [Lawton et al., 2023], and AdaLoRA [Zhang et al., 2023]. These methods and others can be combined to form PEFT design spaces [He et al., 2021].

We propose a method for computing a quantization-aware initialization of the trainable parameters of a QLoRA model [Hu et al., 2021, Dettmers et al., 2023]. Our method exploits the special low-rank structure of the updates to efficiently compute such an initialization. However, it is possible our method could be extended to other reparamterization-based PEFT strategies that exist in the literature, such as Kroncker-product fine-tuning updates [He et al., 2022].

We seek a quantization-aware initialization of the trainable LoRA parameters of a QLoRA model that reduces quantization errors between the QLoRA model and the full-precision model. In order to do so, we build off techniques from the literature on post-training quantization, such as GPT-Q [Frantar et al., 2022], OPS [Frantar and Alistarh, 2022], Bit-stitching [Wang et al., 2020], and QuIP [Chee et al., 2023]. Like GPT-Q, we optimize a calibrated quantization objective to decrease quantization errors on a target calibration dataset. Our method is similar in strategy to other recent methods in the literature such as LQ-LoRA [Guo et al., 2023], LoftQ [Li et al., 2023], and ApiQ [Liao and Monz, 2024].

3 Method

3.1 Background and notation

LoRA [Hu et al., 2021] is a parameter-efficient fine-tuning method that fine-tunes a model by learning a low-rank update $AB^\top$ for each parameter matrix W, where W is an $m \times n$ matrix, A is an $m \times r$ matrix, and B is an $n \times r$ matrix, where r is the user-specified LoRA rank. After fine-tuning, the parameter matrix in the fine-tuned LoRA model is $W + AB^\top$. In contrast, QLoRA [Dettmers et al., 2023] fine-tunes a quantized version of the model with LoRA so that the parameter matrix in the fine-tuned QLoRA model is $Q + AB^\top$, where Q is the quantization of W.

Typically, A is initialized random normal and B is initialized zero. We refer to this as the "baseline initialization".

3.2 Calibrated quantization

To find a quantization-aware initialization for the LoRA matrices A and B, we minimize a *calibrated quantization* objective that aims to keep the activations of the QLoRA model close to those of the full-precision base model on a given *calibration dataset* at initialization. For each parameter matrix W, we propose initializing the LoRA matrices A and B by minimizing a calibrated quantization objective, defined

$$\min_{A,B} \frac{1}{2} \|(W - (Q + AB^\top))X\|_F^2, \tag{1}$$

where Q is the quantization of W, A and B are the real (stored in high-precision) LoRA matrices, $\|\cdot\|_F$ is the Frobenius norm, and X is an $n \times s$ real (stored in high-precision) matrix consisting of the input activations of the full-precision base model to the parameter matrix W on a calibration dataset of s examples. We minimize this objective with respect to A and B independently for each parameter matrix W in the model before proceeding to fine-tune as usual with QLoRA.

The samples used for the calibration dataset may come from the training data of the task that we plan to fine-tune on or from another source.

3.3 Uncalibrated quantization

A related objective that we will make repeated reference to is the *uncalibrated quantization* objective, which does not use a calibration dataset or matrix of input activations X and which instead aims to keep the weights, rather than the activations, of the QLoRA model close to those of the full-precision base model at initialization. The uncalibrated quantization objective is defined

$$\min_{A,B} \frac{1}{2} \|(W - (Q + AB^\top))\|_F^2. \tag{2}$$

Note that if the input activations for the parameter matrix W are uncorrelated, so that $XX^\top = c \cdot I$ for some scalar c, then optimizing the calibrated and uncalibrated quantization objectives are equivalent. However, we find that initializing A and B to minimize this uncalibrated quantization objective is ineffective for improving the performance of QLoRA over the baseline initialization.

3.4 Optimization

To minimize the calibrated quantization objective with respect to A and B, we propose a simple alternating optimization algorithm. To begin the optimization, we initialize A and B by minimizing the uncalibrated quantization objective. This optimization problem is solved by computing the SVD of the parameter quantization error $W - Q$:

$$W - Q = U\Sigma V^\top. \tag{3}$$

Let Σ_r be the diagonal matrix consisting of the r largest singular values of $W - Q$, and let U_r and V_r be matrices consisting of the corresponding top left and right singular vectors, respectively. Then we initialize

$$A = U_r\sqrt{\Sigma_r} \qquad B = V_r\sqrt{\Sigma_r}. \tag{4}$$

After initializing A and B, our algorithm proceeds to optimize the calibrated quantization objective alternately over A and B. Define the activation correlation matrix $H := XX^\top$. Then the updates for A and B each involve solving an $r \times r$ linear system:

$$A := (W - Q)HB^\top(BHB^\top)^{-1} \tag{5}$$

$$B := (A^\top A)^{-1}A^\top(W - Q) \tag{6}$$

Since r is small, typically on the order of 64, these linear systems are computationally inexpensive to solve. Note that to compute these updates, we only need to compute and store the $n \times n$ activation correlation matrix H rather than the $n \times s$ matrix of input activations X, which is typically much larger.

In all our experiments with our method we use $s = 2000$ calibration samples (similar to Frantar et al. [2022]) and execute 20 steps of alternating optimization, so that A and B are each updated 20 times.

4 Experiments

We compare our method against a QLoRA baseline, quantized to either 4-bit or 8-bit precision, that uses the baseline initialization: A random normal and B zero. Whenever we fine-tune a QLoRA model, we use learning rate 2×10^{-4} and fine-tune for one epoch using total batch size of 16.

Fine-tuned → Calibrated ↓	alpaca	chip2	s-i	rlhf
alpaca	6.94	6.04	3.58	8.36
chip2	6.95	6.04	3.56	8.35
s-i	6.94	6.04	3.58	8.36
rlhf	6.95	6.05	3.58	8.35
QLoRA 4-bit	7.02	6.13	3.65	8.43
QLoRA 8-bit	6.87	6.01	3.57	8.35

Table 1: Affect of calibration dataset choice on validation perplexity after fine-tuning for 4-bit models, averaged across six LLMs. We observe that the choice of calibration dataset does not significantly affect validation perplexity after fine-tuning.

We perform quantization using the BitsAndBytes library [Dettmers et al., 2022a,b], using double quantization (quantization of the affine quantization scaling constants) and the NormalFloat4 (NF4) data type for 4-bit quantization. This quantization configuration makes quantization errors small, even before applying our method. In all experiments, we use LoRA $\alpha = 16$, gradient accumulation, warm-up ratio 0.03, and optimize using the AdamW optimizer during fine-tuning.

We evaluate our method on publicly available LLMs across four different LLM families: LLaMA [Touvron et al., 2023], OPT [Zhang et al., 2022], BLOOM [Scao et al., 2022], and Pythia [Biderman et al., 2023]. We use publicly available causal language modeling datasets for calibration, training, and evaluation: Alpaca [Taori et al., 2023], Unified-Chip2 [LAION, 2023], Self-Instruct [Wang et al., 2022], HH-RLHF [Bai et al., 2022], and SlimOrca [Lian et al., 2023]. Unless stated explicitly otherwise, we use LoRA rank $r = 64$ and fine-tune for 1000 steps, except for Pythia-70m, which we fine-tune for 10k steps.

4.1 Choice of calibration set

First we evaluate the effect of the choice of the calibration dataset on performance after fine-tuning. In each experiment, we calibrate and/or fine-tune on Alpaca, Unified-Chip2, Self-Instruct, or HH-RLHF. For each choice of calibration dataset and fine-tuning dataset, we report the validation perplexity after fine-tuning, averaged over six LLMs: Pythia-12b, Pythia-410m, Pythia-70m, BLOOM-3b, BLOOM-560m, and LLaMa-13b.

The results are in Table 1. For comparison, for each fine-tuning dataset we include the average validation perplexity after fine-tuning for the 4-bit and 8-bit baseline QLoRA models. We observe that the choice of calibration dataset does not significantly affect task performance after fine-tuning: the difference in performance between our method and the 4-bit and 8-bit QLoRA baselines for any calibration dataset choice is much larger than the difference between the performance of our method with different calibration dataset choices. We conclude that our method is robust to the choice of calibration dataset.

4.2 Perplexity after fine-tuning

Here we compare the validation perplexity of 4-bit and 8-bit QLoRA models initialized with our method versus the baseline initialization after fine-tuning. Each QuAILoRA model in this section is calibrated on Alpaca.

The average validation perplexity after fine-tuning on Alpaca, Unified-Chip2, Self-Instruct, or HH-RLHF is in Table 2a, and a breakdown of this average by task is in Appendix Table 4. Results for the 8-bit OPT models are omitted due to errors encountered while fine-tuning these models using the BitsAndBytes and `peft` libraries. We observe that in the vast majority of cases, fine-tuning a 4-bit or 8-bit QLoRA model from our initialization achieves lower validation perplexity than fine-tuning from the baseline initialization. In a small number of cases, the model initialized with our method achieves worse validation perplexity compared to the baseline initialization, which we present as failure cases.

In most cases, the 4-bit QuAILoRA model outperforms the 4-bit QLoRA baseline and underperforms the 8-bit QLoRA baseline. We can use the results in Table 2a to compare the decrease in validation perplexity yielded by applying our method to a 4-bit QLoRA model versus doubling the quantization

Table 2: Validation perplexity results

Model	Bits	Method	Avg.
LLaMA-7b	4	QLoRA	3.51
LLaMA-7b	4	Ours	**3.49**
LLaMA-7b	8	QLoRA	3.49
LLaMA-7b	8	Ours	**3.48**
LLaMA-13b	4	QLoRA	3.33
LLaMA-13b	4	Ours	**3.32**
LLaMA-13b	8	QLoRA	3.32
LLaMA-13b	8	Ours	**3.31**
LLaMA-30b	4	QLoRA	**3.30**
LLaMA-30b	4	Ours	**3.30**
LLaMA-30b	8	QLoRA	3.31
LLaMA-30b	8	Ours	**2.29**
OPT-13b	4	QLoRA	3.77
OPT-13b	4	Ours	**3.71**
OPT-30b	4	QLoRA	3.66
OPT-30b	4	Ours	**3.60**
BLOOM-560m	4	QLoRA	6.84
BLOOM-560m	4	Ours	**6.73**
BLOOM-560m	8	QLoRA	**6.73**
BLOOM-560m	8	Ours	6.76
BLOOM-3b	4	QLoRA	4.82
BLOOM-3b	4	Ours	**4.75**
BLOOM-3b	8	QLoRA	4.78
BLOOM-3b	8	Ours	**4.76**
Pythia-70m	4	QLoRA	10.98
Pythia-70m	4	Ours	**10.80**
Pythia-70m	8	QLoRA	10.72
Pythia-70m	8	Ours	**10.69**
Pythia-410m	4	QLoRA	6.73
Pythia-410m	4	Ours	**6.67**
Pythia-410m	8	QLoRA	6.57
Pythia-410m	8	Ours	**6.54**
Pythia-12b	4	QLoRA	5.14
Pythia-12b	4	Ours	**5.11**
Pythia-12b	8	QLoRA	5.09
Pythia-12b	8	Ours	**5.08**

(a) Validation perplexity after fine-tuning various LLMs on four causal language modeling tasks, with and without QuAILoRA. Our method provides a moderate improvement over the baseline in the vast majority of cases.

Model	Gap Closed
LLaMA-7b	100%
LLaMA-13b	61%
LLaMA-30b	N/A
OPT-13b	N/A
OPT-30b	N/A
BLOOM-560m	96%
BLOOM-3b	100%
Pythia-70m	69%
Pythia-410m	37%
Pythia-12b	64%
Avg.	75%

(b) The gap closed in validation perplexity after fine-tuning between QLoRA 4-bit and QLoRA 8-bit quantization by QuAILoRA. The result for LLaMA 30b is omitted because the 8-bit model underperforms the 4-bit model. The results for OPT 13b and 30b are omitted because we were not able to generate results for 8-bit quantization.

precision of the 4-bit QLoRA model to 8-bit. For each model, we can compute the gap in average validation perplexity closed by our method as the difference in average validation perplexity between the 4-bit QLoRA and QuAILoRA models, divided by the difference in average validation perplexity between the 4-bit and 8-bit QLoRA models, capping the computed ratio at 1. The results of this analysis are in Table 2b. We exclude LLaMA-30b from this analysis, since we observed the 8-bit baseline model underperform the 4-bit baseline on average for this model. We also exclude the OPT-13b and OPT-30b LLMs since we could not generate results for the 8-bit versions of these models. Averaging across the other models, the average gap in perplexity between 4-bit and 8-bit quantization closed by applying our method to the 4-bit QLoRA models is 75%. We conclude that applying our method to 4-bit quantized QLoRA models yields approximately 75% of the decrease in validation perplexity achieved by doubling the quantization precision, without increasing GPU memory utilization during fine-tuning.

Table 3: Downstream Task Results

Model	Bits	Method	Avg.
LLaMA-7b	4	QLoRA	62.1
LLaMA-7b	4	Ours	**62.8**
LLaMA-7b	8	QLoRA	63.0
LLaMA-7b	8	Ours	**63.1**
LLaMA-13b	4	QLoRA	65.4
LLaMA-13b	4	Ours	**65.8**
LLaMA-13b	8	QLoRA	**65.8**
LLaMA-13b	8	Ours	65.7

(a) Downstream task accuracy averaged across 7 downstream tasks for LLaMA models fine-tuned on Alpaca.

Model	Gap Closed
LLaMA-7b	74%
LLaMA-13b	100%

(b) The gap in average accuracy between 4-bit and 8-bit QLoRA models closed by QuAILoRA for models fine-tuned on Alpaca.

Model	Bits	Method	Avg.
LLaMA-7b	4	QLoRA	63.2
LLaMA-7b	4	Ours	**63.9**
LLaMA-7b	8	QLoRA	63.8
LLaMA-7b	8	Ours	**63.9**
LLaMA-13b	4	QLoRA	66.7
LLaMA-13b	4	Ours	**67.0**
LLaMA-13b	8	QLoRA	**67.2**
LLaMA-13b	8	Ours	67.1

(c) Downstream task accuracy averaged across 7 downstream tasks for LLaMA models fine-tuned on SlimOrca.

Model	Gap Closed
LLaMA-7b	89%
LLaMA-13b	84%

(d) The gap in average accuracy between 4-bit and 8-bit QLoRA models closed by QuAILoRA for models fine-tuned on SlimOrca.

We observe that in the LLaMA family of models, quantization error does not appear to significantly negatively affect validation perplexity after fine-tuning: the difference in validation perplexity after fine-tuning between the 4-bit and 8-bit baseline QLoRA models is small. Since our method improves performance by reducing quantization error, we observe that the gain in performance provided by our method over the baselines for the LLaMA models is proportionately small. For the other families, the difference in performance between the 4-bit and 8-bit baselines is significant enough for our method to provide a larger advantage over the baselines.

From these results, we conclude that QuAILoRA reduces validation perplexity after fine-tuning on average, proportional to the negative affect of quantization error.

4.3 Performance on downstream tasks

Here we compare how LLaMA models fine-tuned with QuAILoRA versus QLoRA on Alpaca or SlimOrca perform on seven downstream tasks: Arc-Challenge (Arc-C) [Clark et al., 2018], Arc-Easy (Arc-E), BoolQ [Clark et al., 2019], HellaSwag (HS) [Zellers et al., 2019], OpenBookQA (OBQA) [Mihaylov et al., 2018], PIQA [Bisk et al., 2020], and WinoGrande (WinoG) [Keisuke et al., 2019]. We use the EleutherAI LM Evaluation Harness [Gao et al., 2021] for evaluation. For Alpaca experiments, we calibrate on Alpaca and fine-tune for one epoch. For SlimOrca experiments, we calibrate on SlimOrca and fine-tune on a random size-10000 subset of SlimOrca.

The average accuracy achieved across the evaluation tasks is in Tables 3a and 3c, and a breakdown of this average by task is in Appendix Table 5. The gap in accuracy between 4-bit and 8-bit quantization closed by applying our method to each 4-bit model is computed in Tables 3b and 3d. We compute the gap closed for the downstream task experiments in the same way as for the perplexity experiments in the previous subsection. Averaged over all the downstream task experiments, the average gap closed by our method is approximately 86%. We conclude that our method improves the downstream task performance of QLoRA on average, and that applying our method to 4-bit quantized QLoRA models yields approximately 86% of the increase in downstream task performance achieved by doubling the quantization precision, without increasing GPU memory utilization during fine-tuning.

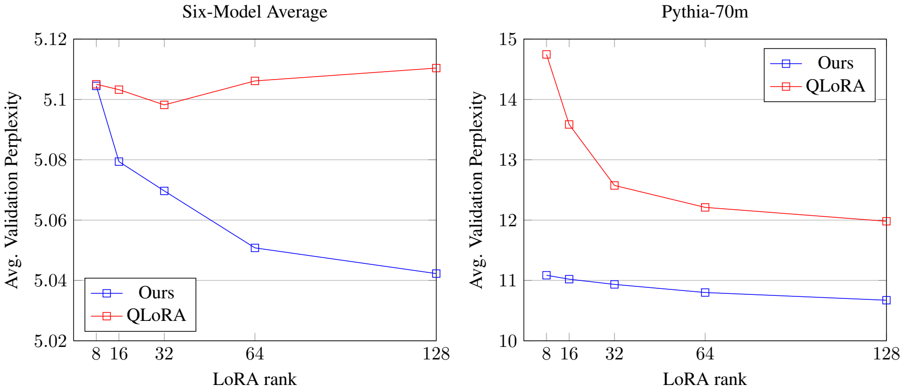

Figure 1: Effect of LoRA rank on validation perplexity after fine-tuning 4-bit models, averaged across six 4-bit LLMs and 4 causal language modeling tasks. Increasing the LoRA rank results in a continual decrease in average validation perplexity when initializing with our method, albeit with diminishing returns. In contrast, increasing the LoRA rank does not significantly affect performance when using the baseline initialization. We plot results for Pythia-70m separately (not included in the six-model average) as this was the only baseline to show a strong decrease in validation perplexity with increasing LoRA rank.

4.4 Effect of LoRA rank

Here we examine the effect of the choice of the LoRA rank hyperparameter r on performance after fine-tuning. We expect that using larger r will allow our initialization of A and B to reduce a greater part of the quantization error and result in better performance after fine-tuning. A plot illustrating the effect of changing the LoRA rank when using our initialization versus the baseline initialization, averaged across six 4-bit LLMs (excluding Pythia-70m, LLaMa-30b and OPT-30b) and 4 causal language modeling tasks, is in Figure 1. We observe that the performance of QLoRA generally increases as we increase the LoRA rank r, albeit with diminishing returns. In contrast, the choice of r does not appear to significantly affect the performance of QLoRA when using the baseline initialization.

4.5 Convergence of fine-tuning

Here we examine how our method affects the speed of convergence of fine-tuning. In Figure 2, we plot the convergence curve for each of the fine-tuning experiments used to generate the validation perplexity results in Table 2a. We plot fine-tuning steps on the horizontal axis and the average validation perplexity across the four causal language modeling tasks, measured every 100 fine-tuning steps, on the vertical axis. We observe that the 4-bit QLoRA models, when initialized with our method, achieve lower average validation perplexity at all stages of fine-tuning compared to the 4-bit QLoRA baselines. The difference in performance between the 8-bit QLoRA models fine-tuned with and without our method is on average much smaller compared to the 4-bit models, likely because the quantization error is already quite small for the 8-bit models and there are diminishing returns for reducing quantization error. From these plots, it appears that fine-tuning does not converge more quickly or slowly when initialized with our method compared to the baseline initialization. Rather, the benefit of decreased validation perplexity after fine-tuning observed for our method is due to decreased validation perplexity at initialization from decreased quantization error.

5 Conclusion

In this paper we introduced QuAILoRA, a method for increasing the performance of QLoRA without additional memory cost during fine-tuning by initializing the LoRA matrices to decrease quantization error. To find such an initialization, we optimized a calibrated quantization objective using alternating optimization, solving a small rank-r linear system in each step. In our experiments, we demonstrated that our LoRA initialization can moderately improve the performance of QLoRA when the impact of

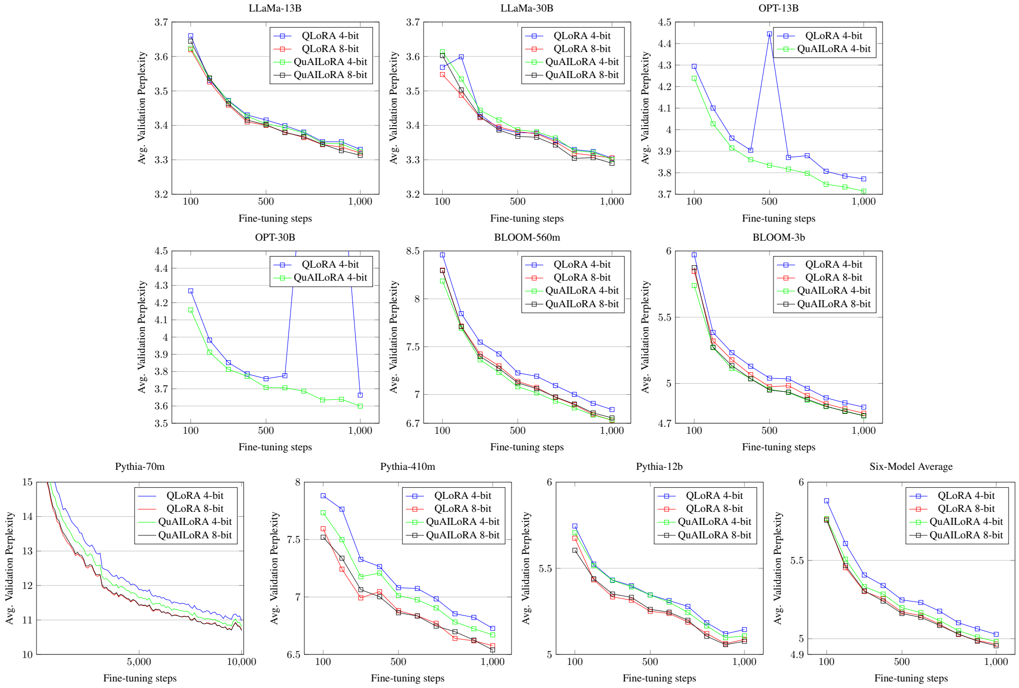

Figure 2: Fine-tuning convergence for each of our 9 base models. We also include a six-model average convergence curve that excludes OPT-13b and OPT-30b (due to the perplexity spikes in the middle of fine-tuning) as well as Pythia-70m (which we fine-tune for 10 times as many steps as the other models).

quantization errors is significant. Furthermore, we found that our results are robust to the choice of the calibration dataset.

6 Limitations

In our experiments, we showed that our method provides a small positive advantage over the baseline QLoRA when quantization error significantly impacts performance. However, quantization error appears to have less of a negative impact for the larger models, including the 13b parameter models we experimented with, and our method may provide only a small or no statistically significant advantage in such cases. We also only present experiments on models up to size 13b on downstream tasks, whereas experiments with models of size 30b and 70b are also common in the literature.

We only explore the affects of our method when quantizing to 4-bit or 8-bit. However, more recent work has also explored quantizing to 3, 2, or 1 bits. It is possible that our method provides a more significant positive advantage in these scenarios where the quantization error is expected to be larger, but we do not present experimental results for these scenarios.

References

Y. Bai, A. Jones, K. Ndousse, A. Askell, A. Chen, N. DasSarma, D. Drain, S. Fort, D. Ganguli, T. Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

S. Biderman, H. Schoelkopf, Q. G. Anthony, H. Bradley, K. O'Brien, E. Hallahan, M. A. Khan, S. Purohit, U. S. Prashanth, E. 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.

Y. Bisk, R. Zellers, R. L. Bras, J. Gao, and Y. Choi. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

J. Chee, Y. Cai, V. Kuleshov, and C. De Sa. Quip: 2-bit quantization of large language models with guarantees. *arXiv preprint arXiv:2307.13304*, 2023.

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

P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*, 2018.

T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer. Llm.int8(): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*, 2022a.

T. Dettmers, M. Lewis, S. Shleifer, and L. Zettlemoyer. 8-bit optimizers via block-wise quantization. *9th International Conference on Learning Representations, ICLR*, 2022b.

T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. *arXiv preprint arXiv:2305.14314*, 2023.

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

E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

L. Gao, J. Tow, S. Biderman, S. Black, A. DiPofi, C. Foster, L. Golding, J. Hsu, K. McDonell, N. Muennighoff, J. Phang, L. Reynolds, E. Tang, A. Thite, B. Wang, K. Wang, and A. Zou. A framework for few-shot language model evaluation, Sept. 2021. URL `https://doi.org/10.5281/zenodo.5371628`.

D. Guo, A. M. Rush, and Y. Kim. Parameter-efficient transfer learning with diff pruning. *arXiv preprint arXiv:2012.07463*, 2020.

H. Guo, P. Greengard, E. P. Xing, and Y. Kim. Lq-lora: Low-rank plus quantized matrix decomposition for efficient language model finetuning. *arXiv preprint arXiv:2311.12023*, 2023.

J. He, C. Zhou, X. Ma, T. Berg-Kirkpatrick, and G. Neubig. Towards a unified view of parameter-efficient transfer learning. *arXiv preprint arXiv:2110.04366*, 2021.

X. He, C. Li, P. Zhang, J. Yang, and X. E. Wang. Parameter-efficient model adaptation for vision transformers. *arXiv preprint arXiv:2203.16329*, 2022.

N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR, 2019.

E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

S. Keisuke, L. B. Ronan, B. Chandra, and C. Yejin. Winogrande: An adversarial winograd schema challenge at scale. 2019.

LAION. Open-instruction-generalist dataset. https://github.com/LAION-AI/, 2023.

N. Lawton, A. Kumar, G. Thattai, A. Galstyan, and G. V. Steeg. Neural architecture search for parameter-efficient fine-tuning of large pre-trained language models. *arXiv preprint arXiv:2305.16597*, 2023.

Y. Li, Y. Yu, C. Liang, P. He, N. Karampatziakis, W. Chen, and T. Zhao. Loftq: Lora-fine-tuning-aware quantization for large language models. *arXiv preprint arXiv:2310.08659*, 2023.

W. Lian, G. Wang, B. Goodson, E. Pentland, A. Cook, C. Vong, and "Teknium". Slimorca: An open dataset of gpt-4 augmented flan reasoning traces, with verification, 2023. URL `https://https://huggingface.co/Open-Orca/SlimOrca`.

B. Liao and C. Monz. Apiq: Finetuning of 2-bit quantized large language model. *arXiv preprint arXiv:2402.05147*, 2024.

T. Mihaylov, P. Clark, T. Khot, and A. Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*, 2018.

T. L. Scao, A. Fan, C. Akiki, E. Pavlick, S. Ilić, D. Hesslow, R. Castagné, A. S. Luccioni, F. Yvon, M. Gallé, et al. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022.

R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`, 2023.

H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

P. Wang, Q. Chen, X. He, and J. Cheng. Towards accurate post-training network quantization via bit-split and stitching. In *International Conference on Machine Learning*, pages 9847–9856. PMLR, 2020.

Y. Wang, Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi, and H. Hajishirzi. Self-instruct: Aligning language model with self generated instructions. *arXiv preprint arXiv:2212.10560*, 2022.

E. B. Zaken, S. Ravfogel, and Y. Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*, 2021.

R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Q. Zhang, M. Chen, A. Bukharin, P. He, Y. Cheng, W. Chen, and T. Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*, 2023.

S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

A Appendix

Table 4: Validation perplexity of QLoRA and QuAILoRA models.

Model	Bits	Method	Alpaca	Chip2	Self-Instruct	HH-RLHF	Avg.
LLaMA-7b	4	QLoRA	3.69	**3.44**	**2.08**	4.84	3.51
LLaMA-7b	4	Ours	**3.65**	**3.44**	2.09	**4.79**	**3.49**
LLaMA-7b	8	QLoRA	**3.65**	3.43	2.08	4.82	3.49
LLaMA-7b	8	Ours	**3.65**	**3.41**	**2.06**	**4.81**	**3.48**
LLaMA-13b	4	QLoRA	3.45	**3.24**	**2.11**	4.52	3.33
LLaMA-13b	4	Ours	**3.44**	**3.24**	**2.11**	**4.50**	**3.32**
LLaMA-13b	8	QLoRA	3.44	**3.23**	**2.09**	4.52	3.32
LLaMA-13b	8	Ours	**3.43**	**3.23**	**2.09**	**4.51**	**3.31**
LLaMA-30b	4	QLoRA	**3.42**	**3.22**	**2.10**	4.48	**3.30**
LLaMA-30b	4	Ours	**3.42**	**3.22**	**2.10**	**4.46**	**3.30**
LLaMA-30b	8	QLoRA	3.44	3.21	2.10	4.48	3.31
LLaMA-30b	8	Ours	**3.42**	**3.20**	**2.08**	**4.47**	**2.29**
OPT-13b	4	QLoRA	3.89	3.68	2.25	5.27	3.77
OPT-13b	4	Ours	**3.82**	**3.59**	**2.21**	**5.24**	**3.71**
OPT-30b	4	QLoRA	3.78	3.57	2.14	5.15	3.66
OPT-30b	4	Ours	**3.68**	**3.48**	**2.13**	**5.11**	**3.60**
BLOOM-560m	4	QLoRA	6.85	6.40	3.67	10.46	6.84
BLOOM-560m	4	Ours	**6.73**	**6.27**	**3.60**	**10.34**	**6.73**
BLOOM-560m	8	QLoRA	**6.70**	**6.31**	**3.60**	**10.31**	**6.73**
BLOOM-560m	8	Ours	6.71	6.36	3.63	10.33	6.76
BLOOM-3b	4	QLoRA	4.71	4.44	2.63	7.51	4.82
BLOOM-3b	4	Ours	**4.64**	**4.35**	**2.59**	**7.45**	**4.75**
BLOOM-3b	8	QLoRA	4.66	4.39	2.60	7.45	4.78
BLOOM-3b	8	Ours	**4.63**	**4.36**	**2.59**	**7.44**	**4.76**
Pythia-70m	4	QLoRA	13.55	11.08	6.26	13.03	10.98
Pythia-70m	4	Ours	**13.39**	**10.86**	**6.08**	**12.87**	**10.80**
Pythia-70m	8	QLoRA	13.18	**10.73**	**5.97**	13.00	10.72
Pythia-70m	8	Ours	**13.13**	10.74	6.00	**12.90**	**10.69**
Pythia-410m	4	QLoRA	7.61	6.57	4.18	8.55	6.73
Pythia-410m	4	Ours	**7.57**	**6.52**	**4.09**	**8.50**	**6.67**
Pythia-410m	8	QLoRA	7.42	6.40	4.12	**8.36**	6.57
Pythia-410m	8	Ours	**7.35**	**6.38**	**4.06**	**8.36**	**6.54**
Pythia-12b	4	QLoRA	5.93	5.06	3.08	6.50	5.14
Pythia-12b	4	Ours	**5.90**	**5.00**	**3.03**	**6.50**	**5.11**
Pythia-12b	8	QLoRA	5.83	5.00	3.04	6.48	5.09
Pythia-12b	8	Ours	**5.81**	**4.99**	**3.03**	**6.47**	**5.08**

Method	Model	Arc-C	Arc-E	BoolQ	HS	OBQA	PIQA	WinoG	avg.
QLoRA	7b 4-bit	41.8$_{\pm1.5}$	71.3$_{\pm1.0}$	73.9$_{\pm0.5}$	55.7$_{\pm0.2}$	31.5$_{\pm0.7}$	77.6$_{\pm0.3}$	82.8$_{\pm0.2}$	62.1
Ours	7b 4-bit	**42.2**$_{\pm1.5}$	**73.1**$_{\pm0.9}$	**75.7**$_{\pm0.4}$	**55.8**$_{\pm0.2}$	**31.9**$_{\pm0.7}$	**77.9**$_{\pm0.3}$	**82.9**$_{\pm0.2}$	**62.8**
QLoRA	7b 8-bit	**43.3**$_{\pm1.5}$	72.3$_{\pm0.9}$	**76.8**$_{\pm0.4}$	55.9$_{\pm0.2}$	32.1$_{\pm0.7}$	**77.8**$_{\pm0.3}$	**83.0**$_{\pm0.2}$	63.0
Ours	7b 8-bit	42.8$_{\pm1.5}$	**73.3**$_{\pm0.9}$	76.2$_{\pm0.4}$	**56.4**$_{\pm0.2}$	**32.5**$_{\pm0.7}$	**77.8**$_{\pm0.3}$	**83.0**$_{\pm0.2}$	**63.1**
QLoRA	13b 4-bit	44.6$_{\pm1.5}$	74.3$_{\pm0.9}$	82.6$_{\pm0.4}$	**59.2**$_{\pm0.2}$	33.0$_{\pm0.7}$	79.4$_{\pm0.3}$	84.9$_{\pm0.2}$	65.4
Ours	13b 4-bit	**46.6**$_{\pm1.5}$	**74.5**$_{\pm0.9}$	**83.0**$_{\pm0.4}$	58.5$_{\pm0.2}$	**33.2**$_{\pm0.7}$	**79.5**$_{\pm0.3}$	**85.2**$_{\pm0.2}$	**65.8**
QLoRA	13b 8-bit	**45.8**$_{\pm1.5}$	**74.7**$_{\pm0.9}$	82.6$_{\pm0.4}$	**58.9**$_{\pm0.2}$	**33.5**$_{\pm0.7}$	**79.5**$_{\pm0.3}$	85.4$_{\pm0.2}$	**65.8**
Ours	13b 8-bit	45.6$_{\pm1.5}$	74.5$_{\pm0.9}$	**82.8**$_{\pm0.4}$	58.8$_{\pm0.2}$	33.3$_{\pm0.7}$	**79.5**$_{\pm0.3}$	**85.6**$_{\pm0.2}$	65.7

(a) Downstream task performance for LLaMA models fine-tuned on Alpaca.

Method	Model	Arc-C	Arc-E	BoolQ	HS	OBQA	PIQA	WinoG	avg.
QLoRA	7b 4-bit	41.6$_{\pm1.5}$	71.8$_{\pm0.9}$	80.2$_{\pm0.4}$	57.2$_{\pm0.2}$	31.5$_{\pm0.7}$	77.1$_{\pm0.3}$	83.2$_{\pm0.2}$	63.2
Ours	7b 4-bit	**42.4**$_{\pm1.5}$	**73.3**$_{\pm0.9}$	**80.9**$_{\pm0.4}$	**57.6**$_{\pm0.2}$	**31.9**$_{\pm0.7}$	**77.4**$_{\pm0.3}$	**83.7**$_{\pm0.2}$	**63.9**
QLoRA	7b 8-bit	**41.8**$_{\pm1.5}$	72.8$_{\pm0.9}$	81.2$_{\pm0.4}$	57.9$_{\pm0.2}$	**32.0**$_{\pm0.7}$	**77.6**$_{\pm0.3}$	83.5$_{\pm0.2}$	63.8
Ours	7b 8-bit	41.6$_{\pm1.5}$	**73.0**$_{\pm0.9}$	**81.5**$_{\pm0.4}$	**58.1**$_{\pm0.2}$	31.9$_{\pm0.7}$	77.5$_{\pm0.3}$	**83.7**$_{\pm0.2}$	**63.9**
QLoRA	13b 4-bit	48.0$_{\pm1.5}$	77.3$_{\pm0.9}$	**84.1**$_{\pm0.4}$	59.8$_{\pm0.2}$	33.4$_{\pm0.7}$	79.4$_{\pm0.3}$	85.2$_{\pm0.2}$	66.7
Ours	13b 4-bit	**48.3**$_{\pm1.5}$	**77.8**$_{\pm0.9}$	83.9$_{\pm0.4}$	**59.9**$_{\pm0.2}$	**34.0**$_{\pm0.7}$	**79.6**$_{\pm0.3}$	**85.6**$_{\pm0.2}$	**67.0**
QLoRA	13b 8-bit	**48.8**$_{\pm1.5}$	78.1$_{\pm0.9}$	**84.6**$_{\pm0.4}$	60.1$_{\pm0.2}$	**34.0**$_{\pm0.7}$	**79.5**$_{\pm0.3}$	**85.5**$_{\pm0.2}$	**67.2**
Ours	13b 8-bit	**48.8**$_{\pm1.5}$	**78.3**$_{\pm0.9}$	83.5$_{\pm0.4}$	**60.2**$_{\pm0.2}$	**34.0**$_{\pm0.7}$	**79.5**$_{\pm0.3}$	**85.5**$_{\pm0.2}$	67.1

(b) Downstream task performance for LLaMA models fine-tuned on SlimOrca.

Table 5: Downstream task accuracy of QLoRA and QuAILoRA models. Error bars reported are one standard error.