Beware of the Batch Size: Hyperparameter Bias in Evaluating LoRA

Anonymous Authors[1]

Abstract

Low-rank adaptation (LoRA) is a standard for fine-tuning large language models, yet its many variants report conflicting empirical gains, often on the same benchmarks. We show that these contradictions arise from a single overlooked factor: the batch size. When properly tuned, vanilla LoRA often matches the performance of more complex variants. We further propose a proxy-based, cost-efficient strategy for batch size tuning, revealing the impact of rank, dataset size, and model capacity on the optimal batch size. Our findings elevate batch size from a minor implementation detail to a first-order design parameter, reconciling prior inconsistencies and enabling more reliable evaluations of LoRA variants.

1. Introduction

Low-rank adaptation, or simply LoRA (Hu et al., 2022), has emerged as the de facto standard method for the parameter-efficient fine-tuning of large language models (LLMs). Building on this success, numerous LoRA variants have been proposed, each claiming a performance gain over the vanilla LoRA (Zhang et al., 2023; Liu et al., 2024; Meng et al., 2024; Wang et al., 2025; Kalajdzievski, 2023).

A disturbing fact about LoRA variants is that they often contradict each other. For example, two recent works, PiSSA and MiLoRA, propose conflicting initialization strategies for LoRA, each focusing on the preservation of the principal singular vectors and minor singular vectors of the pretrained LLM weights, respectively (Meng et al., 2024; Wang et al., 2025). More intriguingly, experiments suggest that both methods provide performance gains on seemingly identical benchmarks. Why does such a contradiction take place?

In this work, we reveal that this apparent contradiction stems from the inconsistency in a critical yet overlooked hyperparameter: the ***batch size***. In particular, we observe that, given a properly tuned batch size and learning rates, the vanilla LoRA can match or even outperform its more complex variants (Figure 1). This finding highlights the pivotal role of a careful batch size selection, not only for an effective LLM fine-tuning, but also for a reliable and reproducible evaluation of LoRA variants.

Despite such an importance, the selection of LoRA batch sizes primarily relies on crude heuristics, e.g., "smaller is better" (Schulman & Thinking Machines Lab, 2025). While the influence of batch size has been extensively studied for full-parameter training (Keskar et al., 2017; Shallue et al., 2019; Zhang et al., 2025; Pareja et al., 2025), these insights do not directly extend to LoRA, operating under unique constraints and demonstrating distinct optimization dynamics (Shuttleworth et al., 2025; Biderman et al., 2025). Furthermore, given that LoRA is primarily used in resource-constrained settings, an exhaustive hyperparameter sweep of the batch size may be infeasible as well.

To address this gap, we initiate a systematic study toward understanding how various scale parameters of LoRA workloads—rank, dataset scale, and model size—affect the optimal batch size. In particular, we find that the optimal batch size remains relatively consistent under the changes in terms of the rank and the model size, but not for the dataset size. This observation proposes a low-cost proxy that one may tune the batch size on small-scale LLMs with low rank, then transfer the batch size to the larger models.

In summary, our contributions are threefold:

1. We demonstrate that vanilla LoRA remains a highly competitive baseline with a properly tuned batch size, suggesting that reported gains in its variants are partially artifacts of suboptimal hyperparameter choices.

2. We find that using a larger batch does not necessarily lead to strictly worse accuracy. Instead, we identify an optimal batch size that maximizes test performance.

3. We establish a practical guideline for small-scale proxies, showing that the optimal batch configurations can be identified using lower ranks and smaller model capacities as long as dataset scale is preserved.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

Taken together, our findings shed light on a critical yet overlooked role of batch size in LoRA. We hope this work encourages more robust and reproducible evaluation practices and provides actionable guidance for practitioners deploying LLMs under real world constraints.

2. Experimental Setup

Our experimental setup primarily follows that of Meng et al. (2024), with varying selections of the batch sizes. All reported figures, except for those in Figures 3, 4, and 5, are an average of three random seeds. Other details are as follows.

Model. We mainly use the LLaMA-2-7B model (Touvron et al., 2023). This choice is to ensure consistency with the key baselines (Meng et al., 2024; Wang et al., 2025). We also provide additional experiments with other recent model families, Qwen3-0.6B (Yang et al., 2025) and Gemma3-1B (Kamath et al., 2025), in Appendix G.

Datasets. We follow the standard evaluation pipeline for the mathematical reasoning task: We fine-tune the model on MetaMathQA (Yu et al., 2023), and then evaluate on GSM8K benchmark (Cobbe et al., 2021). By default, we use first 100K samples in the training split of the MetaMathQA. To demonstrate the generalizability of our approach, the result for the Humaneval benchmark (Chen et al., 2021), fine-tuned on the CodeFeedback dataset (Zheng et al., 2025), is provided in Appendix G.

Learning rate sweep. For each batch size selected, we conduct a hyperparameter sweep to find the optimal learning rate. This is mainly due to the prior observations in the full-parameter training literature which indicates that the batch size closely interacts with the learning rate (Smith et al., 2018). We select learning rates in the range of 1×10^{-5} to 3×10^{-3}; see Appendix E for details.

Fixed sample protocol. To decouple the impact of batch size from the total number of training tokens used, our primary experiments are conducted under a fixed sample protocol restricted to a single epoch. This setup aligns with data efficiency objectives and reflects standard practice in the supervised fine-tuning literature (Pareja et al., 2025), where limited data availability necessitates precise control over training duration to mitigate overfitting.

Note that this configuration implies that larger batch sizes undergo fewer gradient updates, which is one potential reason why the heuristic of "smaller is better." holds. To provide a more comprehensive view, we include results where the number of optimization steps is fixed across the setups (Appendix B). In such case, the total number of processed samples varies. Unless otherwise stated, we adopt the fixed sample protocol to align with common fine-tuning practices.

Warm-up ratio. We set the warm-up ratio to zero for all experiments. This decision stems from the fact that varying the batch size alters the total number of training steps for a fixed sample protocol, making it difficult to adjust the warm-up period across different configurations. Furthermore, Pareja et al. (2025) suggest that warm-up steps and learning rate schedulers have a negligible impact on performance in supervised fine-tuning scenarios. Our empirical analysis in Appendix C confirms that omitting the warm-up phase does not degrade performance; rather, it leads to greater robustness across a wide range of batch sizes. In contrast, we observed that the learning rate scheduler itself significantly influences accuracy. Therefore, we exclude only the warm-up phase from our setup.

Other hyperparameters. We keep all other hyperparameters identical across the setups, in order to study the effect of batch size in isolation. See more details in Appendix D.

3. Re-evaluating LoRA Variants: The Impact of Batch Size

In this section, we evaluate the performance of LoRA alongside two prominent variants, PiSSA (Meng et al., 2024) and MiLoRA (Wang et al., 2025). PiSSA utilizes the principal singular vectors of pretrained weight matrices to align adapters with dominant directions. Conversely, MiLoRA adopts a conflicting initialization strategy using minor singular vectors, aiming to preserve the knowledge of the base model. Both studies report state-of-the-art results on the same benchmarks, while their original evaluations were conducted under disparate configurations, leading to contradictory conclusions. To ensure a rigorous comparison, we re-evaluate these methods within a unified experimental framework, as illustrated in Figure 1.

3.1. Dissecting reported gains: Methodological superiority or hyperparameter bias?

In Figure 1, a performance crossover exists between recent LoRA variants: PiSSA achieves superior accuracy in the large batch regime, whereas it performs worse with smaller batches. This discrepancy aligns with the experimental settings reported in their respective studies, where MiLoRA was evaluated using smaller batch sizes than PiSSA. These observations suggest that the reported gains and opposing insights in both papers may stem not only from algorithmic differences but also from the confounding bias induced by batch size configurations. Thus, no single method appears to be universally superior across the entire batch size spectrum.

Furthermore, we emphasize that vanilla LoRA still remains a competitive baseline when evaluated under fair conditions.

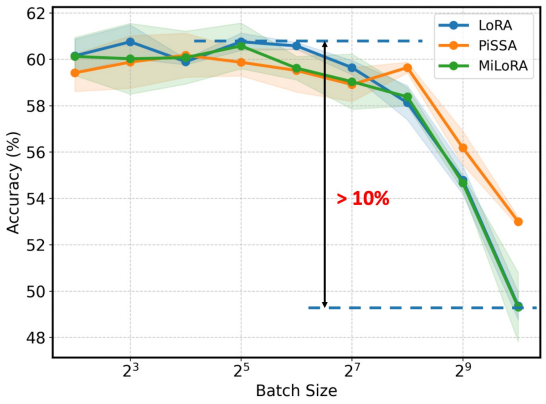


Figure 1. **Impact of batch size across LoRA variants.** We observe that batch size selection alone can lead to a performance gap of over 10% in accuracy. Notably, when evaluated at its optimal batch size, vanilla LoRA beats both PiSSA and MiLoRA in math reasoning task.

LoRA exhibits a similar performance level to its variants across varying batch sizes and achieves the best overall result when its hyperparameters are optimally tuned. These findings underscore that the influence of batch size has been largely overlooked in the landscape of LoRA-based training. In an era of rapidly emerging LoRA research, establishing a fundamental understanding of batch size dynamics is essential to isolating the intrinsic effectiveness of any given method from confounding hyperparameter effects.

We demonstrate that fine-tuning performance does not scale monotonically as batch size decreases. Our findings stand in contrast to prevalent heuristics, such as those proposed by Schulman & Thinking Machines Lab (2025), which suggest that smaller batch sizes are universally superior for LoRA-based fine-tuning. Instead, we characterize the optimal batch size, which can be leveraged to maximize computational efficiency without compromising final accuracy. This trend is robust across multiple LoRA variants, while the critical threshold varies. Given that LoRA is typically deployed in resource constrained environments, maximizing hardware utility is pivotal; however, this must be balanced against systemic variables such as LoRA rank, dataset scale, and base model capacity. To facilitate more accessible batch size tuning, in Section 4, we provide a rigorous analysis of the interactions between batch size and these determinants to establish a more systematic guideline for optimal hyperparameter configuration.

4. Determinants of Batch Size Effect

This section identifies the fundamental factors required to construct a reliable low-cost proxy for batch size tuning. We investigate how the impact of batch size varies through three key determinants: (i) LoRA rank, (ii) dataset scale, and (iii) base model capacity. These variables are selected as they represent the primary factors of the total computational overhead in LoRA-based training. For each analysis, we maintain our setup established in Section 2, while altering each determinant independently. We reveal a critical insight for establishing an effective proxy: while batch size effects are largely robust to changes in LoRA rank and base model size, they are highly sensitive to dataset scale. Consequently, we demonstrate that one can reliably tune batch sizes for LoRA using lower rank with smaller models, while the full dataset scale should be preserved.

4.1. Impact of LoRA rank

We first investigate the interaction between batch size and the rank r of LoRA adapters. Figure 2a illustrates the test accuracies across a range of batch sizes for ranks spanning from 32 to 256. While minor variances exist, the fundamental performance trend remains consistent across all evaluated ranks. Notably, in Figure 6a, increasing the rank beyond a certain threshold yields marginal performance gains. These results suggest that in some scenarios, employing a moderate LoRA rank in conjunction with a larger batch size can expect a more Pareto optimal configuration for balancing performance and throughput.

4.2. Impact of dataset scale

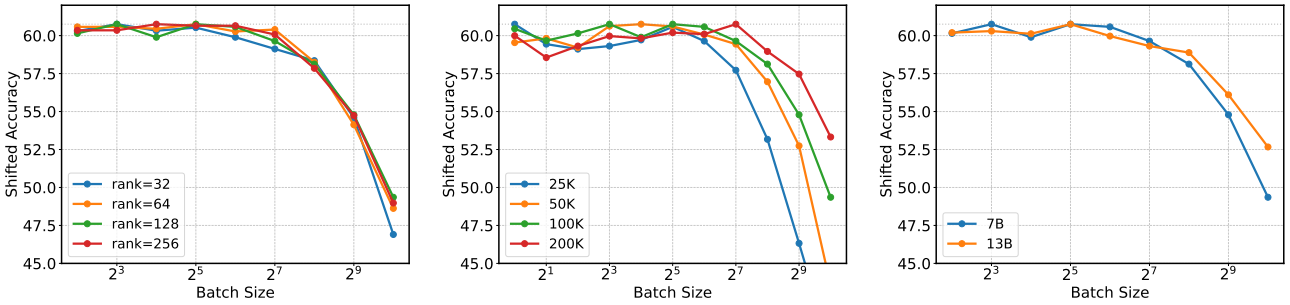
We compare training configurations using subsets of the MetaMathQA dataset ranging from 25K to 200K samples. As illustrated in Figure 2b, LoRA training on larger datasets effectively accommodates larger batch sizes without the performance degradation observed in smaller data regimes. This phenomenon can be attributed to the reduction in relative gradient noise; as the dataset scale increases, the optimization process becomes more robust, allowing for increased batch sizes while maintaining stable convergence. This highlights that the batch size is not a static hyperparameter but a dynamic one that must be carefully calibrated relative to the total available data scale.

4.3. Impact of base model capacity

To examine how the capacity of the backbone model influences batch size effect, we compared two different models, LLaMA-2-7B and 13B. Our empirical results in Figure 2c demonstrate that the impact of batch size remains remarkably consistent regardless of the model scale. This suggests that the relationship between batch size and the model capacity is scale invariant.

4.4. Guidelines for Small-Scale Proxies

Building on our analysis, we establish a practical strategy for constructing a low-cost proxy to identify the optimal batch



(a) LoRA rank *(b)* Dataset scale *(c)* Base model capacity

Figure 2. **Effect of batch size across key determinants.** We examine the interaction between batch size and three factors: **(a) LoRA rank**: the impact of batch size remains consistent across varying ranks r; **(b) dataset scale**: larger data regimes effectively leverage larger batch training; and **(c) base model capacity**: batch size effects are largely invariant to model scale. For a unified comparison, accuracies are normalized by shifting the maximum value of each setup to match the default configuration ($r = 128$, 100K samples, 7B model). Original accuracy values are provided in Figure 6 for reference.

size for LoRA fine-tuning. The batch size effect is scale invariant regarding LoRA rank and model capacity, while it is highly sensitive to the total data scale. In conclusion, the most reliable proxy is to employ a smaller model with a lower rank while training on the full target dataset. This approach ensures high fidelity hyperparameter transfer with a fraction of the original computational overhead.

4.5. Theoretical Justification of Batch Size Trend

Our finding that the optimal batch size is invariant to the base model capacity is consistent with recent theoretical findings on the infinite-width regime (Yang & Hu, 2021). Beyond a certain threshold of model capacity, the training dynamics become largely independent of the base model size, explaining the consistent batch size trends observed across different model scales. Furthermore, the dominance of the dataset scale over the LoRA rank can be analyzed through the lens of the gradient noise scale (GNS) (McCandlish et al., 2018). We characterize the relationship between dataset size N and the rank r with the following lemma:

Lemma 4.1. *Consider a dataset* $\{(\mathbf{x}_i, y_i)\}_{i=1}^N \overset{\text{i.i.d.}}{\sim} \mathcal{N}(\mathbf{0}, \mathbf{I}_r) \times \text{Unif}(\{-1, +1\})$*, where* $\tilde{\mathbf{x}}_i$ *can be written as* $\tilde{\mathbf{x}}_i := \mathbf{U}\mathbf{x}_i$*, where* $\mathbf{U} \in \{\mathbf{A} \in \mathbb{R}^{d\times r} | \mathbf{A}^\top \mathbf{A} = \mathbf{I}_r\}$*. For a quadratic objective* $\hat{L}(\mathbf{w}) := \frac{1}{2N}\sum_{i=1}^N (\mathbf{w}^\top \tilde{\mathbf{x}}_i - y_i)^2$*. For any* $r > 2$*, the expected GNS proxy* $\mathbb{E}[B_{\text{simple}}]$ *is*

$$\mathbb{E}[B_{\text{simple}}] = \frac{rN}{r-2} \tag{1}$$

The derivation implies that for sufficiently large r, the expected gradient noise scale correlates with the dataset while becoming insensitive to LoRA rank. This provides a theoretical basis for our empirical observation: the dataset scale (N) is the primary determinant of the optimal batch size, while the influence of the LoRA rank (r) becomes marginal as it increases. The concept of gradient noise scale and proof of lemma 4.1 are provided in Appendix F.

5. Conclusion

In this study, we provide a comprehensive re-evaluation of LoRA and its prominent variants, specifically focusing on the critical yet overlooked impact of batch size on fine-tuning performance. Our analysis demonstrates that reported advantages of some LoRA variants are confounded by hyperparameter bias. Notably, we highlight that vanilla LoRA remains a remarkably fine baseline when evaluated under optimized conditions.

Contrary to conventional heuristics suggesting that smaller batches yield lower regret, we identify the optimal point, where large batch does not harm model performance. Crucially, our experiments reveal that the optimal batch size of LoRA can be found efficiently by utilizing a smaller model along with small rank, while preserving the scale of dataset. These insights provide a more rigorous foundation for batch size effects in LoRA and offer practical guidance for efficient resource utilization in limited environments.

Limitations

While our analysis provides empirical and practical insights, several avenues remain for further investigation. First, although we provide a theoretical justification via the gradient noise scale, a comprehensive and unified theoretical framework that fully captures the non-convex optimization dynamics of LoRA is still lacking. This prevents the derivation of universal rules for optimal batch size selection across all scenarios. Second, while our batch size evaluations encompass diverse model families and tasks, computational constraints limited our ability to conduct the in-depth determinant analysis across every possible task and architectural configuration. Future research is required to validate these observed patterns across even broader scales and diverse training objectives.

References

Biderman, D., Portes, J., Ortiz, J. J. G., Paul, M., Greengard, P., Jennings, C., King, D., Havens, S., Chiley, V., Frankle, J., Blakeney, C., and Cunningham, J. P. Lora learns less and forgets less. In *International Conference on Learning Representations*, 2025.

Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. Evaluating large language models trained on code, 2021.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems, 2021.

Golmant, N., Vemuri, N., Yao, Z., Feinberg, V., Gholami, A., Rothauge, K., Mahoney, M. W., and Gonzalez, J. On the computational inefficiency of large batch sizes for stochastic gradient descent, 2018.

Hoffer, E., Hubara, I., and Soudry, D. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In *Advances in Neural Information Processing Systems*, 2017.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Kalajdzievski, D. A rank stabilization scaling factor for fine-tuning with lora. *arXiv e-prints*, 2023.

Kamath, G. T. A., Ferret, J., Pathak, S., Vieillard, N., Merhej, R., Perrin, S., Matejovicova, T., Ram'e, A., Rivière, M., Rouillard, L., Mesnard, T., Cideron, G., Grill, J.-B., Ramos, S., Yvinec, E., Casbon, M., Pot, E., Penchev, I., Liu, G., Visin, F., Kenealy, K., Beyer, L., Zhai, X., Tsitsulin, A., Busa-Fekete, R. I., Feng, A., Sachdeva, N., Coleman, B., Gao, Y., Mustafa, B., Barr, I., Parisotto, E., Tian, D., Eyal, M., Cherry, C., Peter, J.-T., Sinopalnikov, D., Bhupatiraju, S., Agarwal, R., Kazemi, M., Malkin, D., Kumar, R., Vilar, D., Brusilovsky, I., Luo, J., Steiner, A., Friesen, A., Sharma, A., Sharma, A., Gilady, A. M., Goedeckemeyer, A., Saade, A., Kolesnikov, A., Bendebury, A., Abdagic, A., Vadi, A., Gyorgy, A., Pinto, A. S., Das, A., Bapna, A., Miech, A., Yang, A., Paterson, A., Shenoy, A., Chakrabarti, A., Piot, B., Wu, B., Shahriari, B., Petrini, B., Chen, C., Lan, C. L., Choquette-Choo, C. A., Carey, C., Brick, C., Deutsch, D., Eisenbud, D., Cattle, D., Cheng, D., Paparas, D., Sreepathihalli, D. S., Reid, D., Tran, D., Zelle, D., Noland, E., Huizenga, E., Kharitonov, E., Liu, F., Amirkhanyan, G., Cameron, G., Hashemi, H., Klimczak-Pluci'nska, H., Singh, H., Mehta, H., Lehri, H. T., Hazimeh, H., Ballantyne, I., Szpektor, I., Nardini, I., Pouget-Abadie, J., Chan, J., Stanton, J., Wieting, J. M., Lai, J., Orbay, J., Fernandez, J., Newlan, J., Ji, J., Singh, J., Black, K., Yu, K., Hui, K., Vodrahalli, K., Greff, K., Qiu, L., Valentine, M., Coelho, M., Ritter, M., Hoffman, M., Watson, M., Chaturvedi, M., Moynihan, M., Ma, M., Babar, N., Noy, N., Byrd, N., Roy, N., Momchev, N., Chauhan, N., Bunyan, O., Botarda, P., Caron, P., Rubenstein, P. K., Culliton, P., Schmid, P., Sessa, P. G., mei Xu, P., Stańczyk, P., Tafti, P. D., Shivanna, R., Wu, R., Pan, R., Rokni, R. A., Willoughby, R., Vallu, R., Mullins, R., Jerome, S., Smoot, S., Girgin, S., Iqbal, S., Reddy, S., Sheth, S., Põder, S., Bhatnagar, S., Panyam, S. R., Eiger, S., Zhang, S., Liu, T., Yacovone, T., Liechty, T., Kalra, U., Evci, U., Misra, V., Roseberry, V., Feinberg, V., Kolesnikov, V., Han, W., Kwon, W., Chen, X., Chow, Y., Zhu, Y., Wei, Z., Egyed, Z., Cotruta, V., Giang, M., Kirk, P., Rao, A., Lo, J., Moreira, E., Martins, L. G., Sanseviero, O., Gonzalez, L., Gleicher, Z., Warkentin, T., Mirrokni, V. S., Senter, E., Collins, E., Barral, J., Ghahramani, Z., Hadsell, R., Matias, Y., Sculley, D., Petrov, S., Fiedel, N., Shazeer, N., Vinyals, O., Dean, J., Hassabis, D., Kavukcuoglu, K., Farabet, C., Buchatskaya, E., Alayrac, J.-B., Anil, R., Lepikhin, D., Borgeaud, S., Bachem, O., Joulin, A., Andreev, A., Hardin, C., Dadashi, R., and Hussenot, L. Gemma 3 technical report, 2025.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*, 2017.

Li, S., Zhao, P., Zhang, H., Sun, X., Wu, H., Jiao, D., Wang, W., Liu, C., Fang, Z., Xue, J., Tao, Y., Cui, B., and Wang, D. Surge phenomenon in optimal learning rate and batch size scaling. In *Advances in Neural Information Processing Systems*, 2024.

Liu, S.-Y., Wang, C.-Y., Yin, H., Molchanov, P., Wang, Y.-C. F., Cheng, K.-T., and Chen, M.-H. Dora: Weight-decomposed low-rank adaptation. In *International Conference on Machine Learning*, 2024.

Marek, M., Lotfi, S., Somasundaram, A., Wilson, A. G., and Goldblum, M. Small batch size training for language

models: When vanilla sgd works, and why gradient accumulation is wasteful. In *Advances in Neural Information Processing Systems*, 2025.

McCandlish, S., Kaplan, J., Amodei, D., and Team, O. D. An empirical model of large-batch training, 2018.

Meng, F., Wang, Z., and Zhang, M. Pissa: Principal singular values and singular vectors adaptation of large language models. In *Advances in Neural Information Processing Systems*, 2024.

Pareja, A., Nayak, N. S., Wang, H., Killamsetty, K., Sudalairaj, S., Zhao, W., Han, S., Bhandwaldar, A., Xu, G., Xu, K., Han, L., Inglis, L., and Srivastava, A. Unveiling the secret recipe: A guide for supervised fine-tuning small llms. 2025.

Schulman, J. and Thinking Machines Lab. Lora without regret. *Thinking Machines Lab: Connectionism*, 2025.

Shallue, C. J., Lee, J., Antognini, J., Sohl-Dickstein, J., Frostig, R., and Dahl, G. E. Measuring the effects of data parallelism on neural network training. In *Journal of Machine Learning Research*, 2019.

Shuttleworth, R., Andreas, J., Torralba, A., and Sharma, P. Lora vs full fine-tuning: An illusion of equivalence, 2025.

Smith, S. L., Kindermans, P.-J., Ying, C., and Le, Q. V. Don't decay the learning rate, increase the batch size. In *International Conference on Learning Representations*, 2018.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models, 2023. arXiv preprint arXiv:2307.09288.

Wang, H., Li, Y., Wang, S., Chen, G., and Chen, Y. MiLoRA: Harnessing minor singular components for parameter-efficient LLM finetuning. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 2025.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., et al. Qwen3 technical report, 2025.

Yang, G. and Hu, E. J. Tensor programs iv: Feature learning in infinite-width neural networks. In *PMLR*, 2021.

Yu, L., Jiang, W., Shi, H., Yu, J., Liu, Z., Zhang, Y., Kwok, J. T., Li, Z., Weller, A., and Liu, W. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023.

Zhang, H., Morwani, D., Vyas, N., Wu, J., Zou, D., Ghai, U., Foster, D., and Kakade, S. M. How does critical batch size scale in pre-training? In *International Conference on Learning Representations*, 2025.

Zhang, Q., Chen, M., Bukharin, A., He, P., Cheng, Y., Chen, W., and Zhao, T. Adaptive budget allocation for parameter-efficient fine-tuning. In *International Conference on Learning Representations*, 2023.

Zheng, T., Zhang, G., Shen, T., Liu, X., Lin, B. Y., Fu, J., Chen, W., and Yue, X. Opencodeinterpreter: Integrating code generation with execution and refinement, 2025.

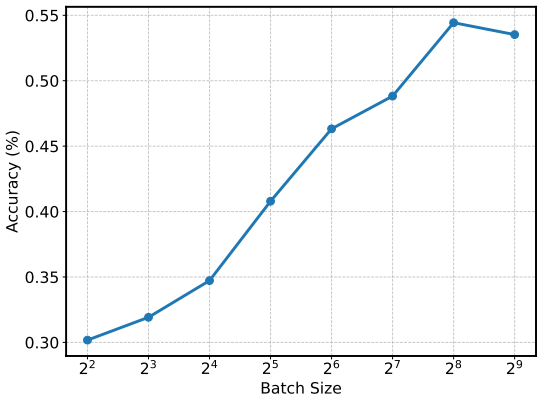


Figure 3. **Batch size effect under fixed training steps.** Under a fixed optimization steps, larger batch sizes lead to superior performance due to increased data throughput. This trend persists until a critical threshold is reached, where increasing batch size beyond no longer yields improvements in test accuracy.

A. Related Work

Batch size in deep learning and LLMs. The influence of batch size on optimization and generalization has been extensively investigated in the broader deep learning literature (Keskar et al., 2017; Hoffer et al., 2017; Shallue et al., 2019; McCandlish et al., 2018; Golmant et al., 2018). McCandlish et al. (2018) characterize a critical batch size $\mathcal{B}_{crit}$ beyond which increasing the batch size yields diminishing returns in training acceleration, along with the concept of gradient noise scale. While these principles are well established for standard architectures, their application to modern Large Language Models (LLMs) is an area of active refinement. Recent work revisits these questions in the LLM training setting. In particular, Zhang et al. (2025) systematically measure critical batch size across model and data scales and find that it is driven primarily by data size rather than parameter count. On the fine-tuning side, Pareja et al. (2025) observe that larger batches can be beneficial when paired with appropriately reduced learning rates in full fine-tuning. At the other extreme, Marek et al. (2025) show that very small batches can train stably with batch size-aware optimizer hyperparameters, and they argue that gradient accumulation can be compute-inefficient when its primary role is to emulate large batches.

Distinct dynamics of LoRA. A growing body of evidence suggests that findings from full fine-tuning (FFT) do not directly translate to parameter-efficient fine-tuning methods such as LoRA. The low rank constraint imposes a distinct optimization dynamics; Shuttleworth et al. (2025) show that FFT updates typically evolve with a significantly higher effective rank than updates of the same rank LoRA. Furthermore, LoRA tends to exhibit a "learn-less and forget-less" phenomenon (Biderman et al., 2025), characterized by smaller weight shifts and better preservation of pre-trained features compared to FFT. These differences imply that the sensitivity to batch size in LoRA may follow a trajectory distinct from standard patterns.

While Pareja et al. (2025) suggest that larger batches are generally beneficial for FFT, recent work (Schulman & Thinking Machines Lab, 2025) investigates into a "low regret regime", where LoRA can achieve similar performance to full fine-tuning, arguing that LoRA is less tolerant of large-batch training than FFT. Moreover, they suggest this gap may not matter much in practice, since smaller batches are better in both LoRA and full fine-tuning. Our work diverges from this binary view; we demonstrate that neither smaller nor larger batches are universally optimal. Instead, we identify the optimal batch size that can be determined through three fundamental determinants of LoRA rank, dataset scale, and model capacity, as detailed in Section 4.

B. Experiments Under Fixed Steps

Figure 3 illustrates test performance across varying batch sizes under a constrained 200 optimization steps. In this configuration, where training with a batch size of 512 corresponds to approximately one epoch of the MetaMathQA100K dataset, larger batch sizes naturally entail a greater volume of training samples. Consequently, we observe a positive correlation between batch size and performance, as the increased data outweighs the potential impact of batch size. This indicates that the degradation observed in fixed sample protocol is partially attributable to insufficient updates rather than batch size itself. However, we emphasize that larger batches do not always yield superior outcomes even in a fixed step

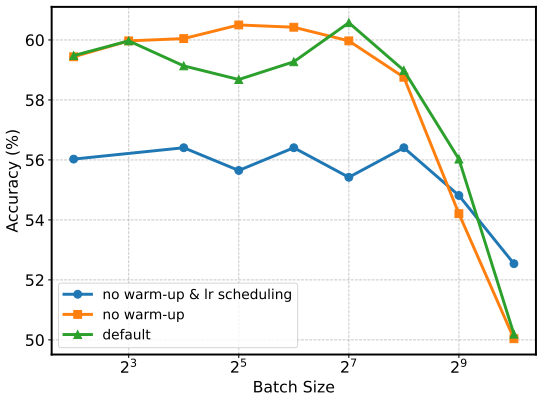


Figure 4. **Impact of warm-up phase and lr scheduling.** We show that while removing the warm-up phase maintains robust performance across all batch sizes without significant accuracy loss, the LR scheduling remains critical to model performance.

setup; rather, there is a point of diminishing returns where the effect of batch size outweighs the benefits of data volume.

C. Ablations: Warm-up & Learning Rate Scheduling.

In this section, we evaluate the sensitivity of LoRA fine-tuning to warm-up phases and learning rate (lr) scheduling. Figure 4 illustrates the test accuracies across a range of batch sizes under varying optimization configurations. Consistent with the observations of Pareja et al. (2025), our results indicate that the inclusion of a warm-up stage yields marginal performance differences; in fact, omitting the warm-up phase leads to slightly superior results across the majority of the batch size spectrum.

The presence of an lr scheduler (e.g., cosine decay) is critical for performance stability, which is contrary to Pareja et al. (2025). We find that the absence of lr scheduling results in a substantial performance degradation of approximately 5% in accuracy. Consequently, we employ an lr scheduler without a warm-up phase for all main experiments in this study.

D. Hyperparameter Configurations.

In Table 1, we provide our detailed configurations of all hyperparameters. This setup basically follows that of Meng et al. (2024), except for some settings discussed in Section 2.

Table 1. **Detailed hyperparameter configurations.**

Default Configurations	
rank r	128
α	same as rank r
Optimizer	AdamW
Dropout	0
LR Scheduler	cosine
Warmup Ratio	0
Epoch	1
Placement	query, key, value, output, gate, MLP up, MLP down

E. Optimal Learning Rate Dynamics.

As detailed in Section 2, we determine the optimal learning rate (lr) independently for each batch size configuration to ensure a fair comparison. Specifically, we conduct a grid search across an lr range of $[1 \times 10^{-5}, 3 \times 10^{-3}]$. To ensure a precise identification of the optimal point, we employ a refined logarithmic grid consisting of $\{1, 2, 5\} \times 10^n$, with additional points such as 3×10^{-4} to provide higher granularity in the high performance region where the optimal lr frequently appears. We

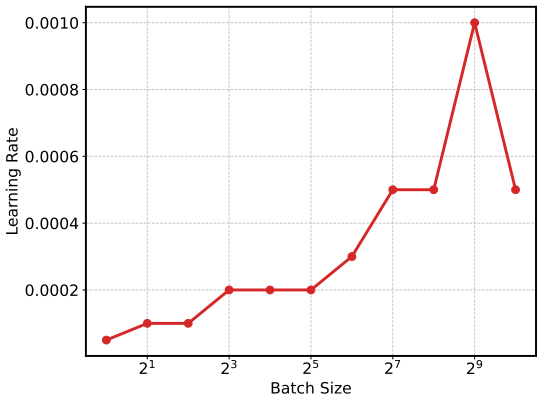
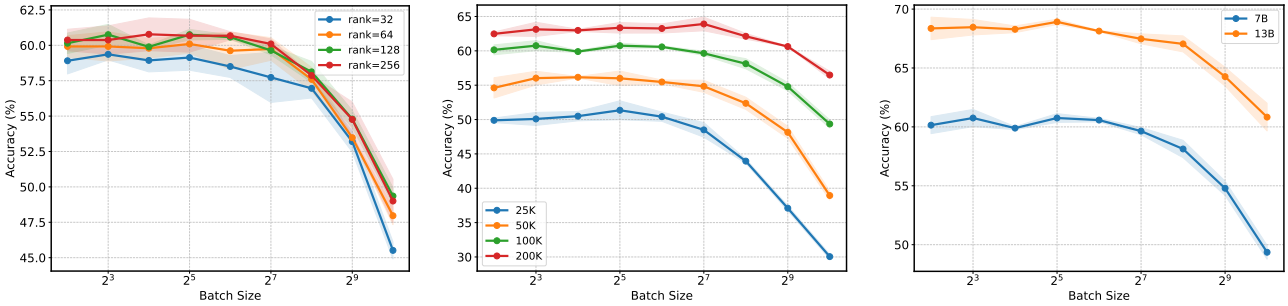


Figure 5. **Interaction between optimal learning rate and batch size.** We demonstrate that the optimal learning rate follows a non-monotonic trajectory as batch size increases, initially scaling upward before declining beyond a critical threshold.



(a) LoRA rank *(b)* Dataset scale *(c)* Base model capacity

Figure 6. **Original accuracies of Figure 2.** We provide original accuracies of Figure 2, which are averaged over three seeds.

also include 3×10^{-3} as the learning rate approaches the stability boundary. Figure 5 illustrates the resulting optimal lr trajectory for each batch size in our default setup with a single seed.

Our analysis reveals that the optimal learning rate exhibits a non-monotonic relationship with batch size: the lr initially increases but subsequently declines as the batch size continues to increase. This behavior in LoRA scenarios aligns with recent findings by Li et al. (2024), suggesting that the standard linear scaling rule may not hold for Adam style optimizers.

F. Theoretical Analysis

F.1. Gradient noise scale proxy for estimating the optimal batch size

McCandlish et al. (2018) introduce the concept of the **gradient noise scale (GNS)** to determine the optimal batch size. Let g denote the full gradient and Σ represent the per-example gradient covariance matrix. The optimal batch size B_{crit} can be estimated as the ratio of the gradient noise to the gradient magnitude, weighted by the Hessian H:

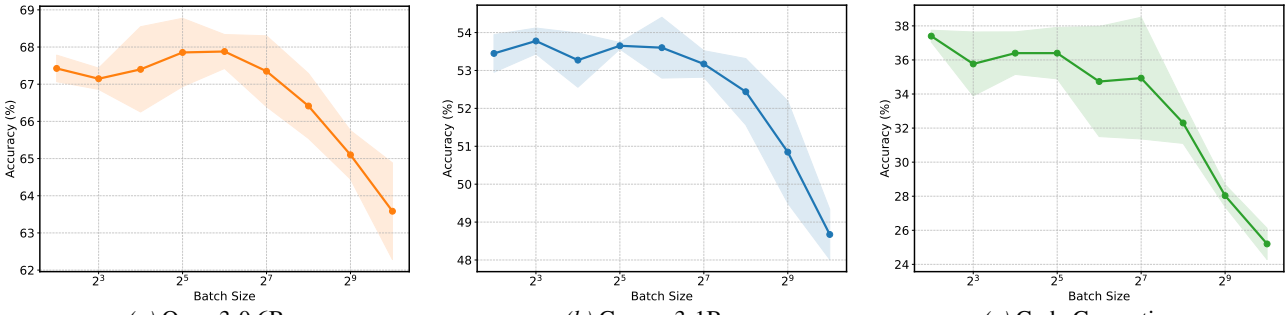
$$B_{\text{noise}} = \frac{\text{tr}(\Sigma H)}{g^\top H g} \tag{2}$$

Under the assumption of a well-conditioned optimization landscape, Equation (2) simplifies to the GNS proxy as:

$$B_{\text{simple}} \approx \frac{\text{tr}(\Sigma)}{|g|^2} \tag{3}$$

F.2. Proof of Lemma 4.1

Proof. We derive the expected gradient noise scale at initialization ($w_0 = 0$) by analyzing the trace of the gradient covariance and the norm of the full gradient.



(a) Qwen3-0.6B *(b)* Gemma3-1B *(c)* Code Generation

Figure 7. **Results on other model families and code generation task.** We demonstrate performances with batch size sweep for recent model families and the code generation task, which are averaged over three seeds.

Trace of Covariance. First, at initialization, the per-example gradient is given by $\nabla L_i(0) = -y_i\tilde{x}_i = -y_i U x_i \in \text{col}(U)$. Since all stochastic gradients reside within this r-dimensional subspace, the per-example gradient covariance Σ has a rank of at most r. Consequently, its trace is given by $\text{tr}(\Sigma) = r$.

Full Gradient Norm. The full gradient $g = \frac{1}{N}\sum \nabla L_i$ follows a Gaussian distribution $g \sim \mathcal{N}(0, \frac{1}{N}UU^\top)$. Given the orthonormality of U (i.e., $U^\top U = I_r$), the squared norm follows a scaled chi-squared distribution, $\|g\|^2 \sim \frac{1}{N}\chi^2_r$.

Expected Gradient Noise Scale. Finally, by leveraging the property of the inverse-chi-squared distribution, where $\mathbb{E}[1/\chi^2_r] = 1/(r-2)$ for $r > 2$, we calculate the expected noise scale as follows:

$$\mathbb{E}[B_{\text{simple}}] \approx \text{tr}(\Sigma) \cdot \mathbb{E}\left[\frac{1}{\|g\|^2}\right] = \frac{rN}{r-2}. \tag{4}$$

□

G. Ablations: Other Model Families & Task.

We evaluate the impact of batch size across additional model architectures, specifically Qwen3-0.6B and Gemma3-1B (see Figures 7a and 7b). Consistent with our primary findings using LLaMA-2-7B , these results highlight the existence of an optimal batch size, where increasing the batch size does not compromise accuracy. Furthermore, we provide results for the HumanEval benchmark fine-tuned on CodeFeedback100K (Figure 7c). Although the precise optimal threshold varies, these findings suggest that the observed batch size dynamics generalize across diverse model families and task domains.