Is your batch size the problem? Revisiting the Adam-SGD gap in language modeling

author names withheld

Under Review for the Workshop on High-dimensional Learning Dynamics, 2024

Abstract

Adam is known to perform significantly better than Stochastic Gradient Descent in language models, a phenomenon for which a number of explanations have been proposed. In this work, we revisit this ”optimizer gap” through a series of comprehensively tuned baseline training runs for language modeling with transformers. We exhaustively study how momentum, gradient clipping, and batch size affect the gap between SGD and Adam. Our empirical findings show that SGD with momentum can actually perform similarly to Adam in small-batch settings, if tuned correctly. We revisit existing explanations for Adam’s advantage, including heavy-tailed class imbalance, directional sharpness, and Hessian heterogeneity, which struggle to explain these findings. Finally, by analyzing our transformer training runs and a simple quadratic setting, we provide new insights into what makes SGD perform poorly - showing that batch size has to be a necessary component of any explanation of the optimizer gap.

1. Introduction

The Adam optimizer [11] is used pervasively in deep learning, especially when training large language models (LMs) [3, 6, 15] and vision transformers [12]. Industrial practice relies on the success of Adam, and thousands of GPU hours every day are spent at large companies using Adam to train their next-generation large language models.

Even new work looking to dethrone Adam, such as Muon [10, 16, 24, 25] and SOAP [28] (A Shampoo variant [7]) still rely on plain Adam with weight decay (AdamW, Loshchilov and Hutter [19]) for critical subsets of parameters, such as normalization layers, text embeddings and prediction heads. This new world is still a bit surprising. Up until around the year 2018, the Adam optimizer was in occasional use, but stochastic gradient descent (SGD) with momentum was known to lead to neural networks with better accuracy on unseen data [29], relegating Adam to speed runs and quick comparisons [5]. Yet, language modeling with transformers has, from the start, required Adam, with models even being reported as untrainable with SGD–especially due to critical parameters.

Over the years, researchers have offered a number of compelling explanations regarding the remarkable performance of Adam compared to SGD in language modeling, attributing it either to the peculiar noisy nature of text data [31, 32] or the heterogeneous structure [22, 33] of the transformer architecture [27] – comprising semantically and structurally dissimilar layers. While most hypotheses regarding the Adam-SGD gap seem correct to some extent [2], a crucial insight was recently brought to light by Kunstner et al. [13]: such a gap is also observable in full-batch training, making it clear that the stochastic and potentially heavy-tailed nature of stochastic gradients may not be the challenge Adam is able to tackle.

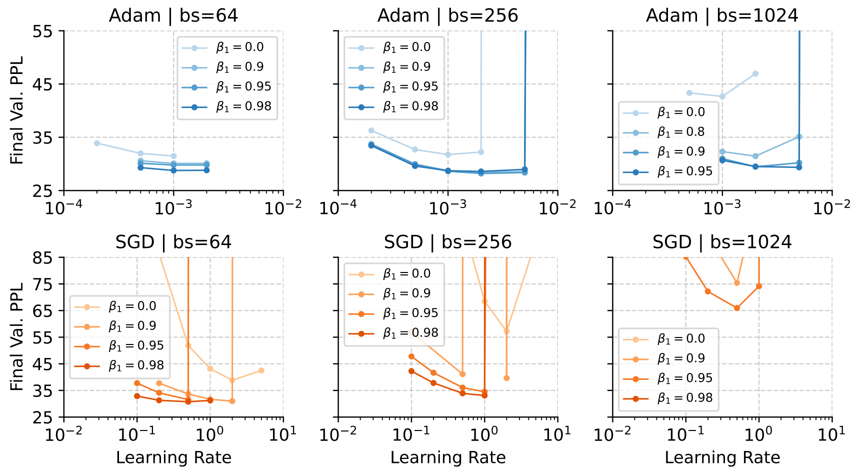

Figure 1: Learning rate and momentum sweep for SGD and Adam across batch sizes under a fixed token budget.

Inspired by the latter discussion, we take in this paper an orthogonal approach: instead of asking why Adam can outperform SGD, we ask instead

In which language model training setting, if any, does SGD work?

In other words, while most recent works try to maximize the gap between SGD an Adam in order to explain it more easily, we here try to minimize it. We believe such view is novel in the literature, and can provide many valuable insights on the Adam-SGD gap, especially to discover settings that falsify existing hypotheses, and to enumerate necessary criteria that explanations have to fulfill. Our contributions are as follows:

- Despite our own surprise, we show that LMs can be trained with SGD as effectively as Adam at the same token budget, as long as the batch size is chosen small enough, and hyperparameters, such as clipping and momentum are chosen correctly.
- We find that prior explanations, such as heavy-tailed class imbalance, directional sharpness and Hessian heterogeneity fail to distinguish this setting, showing that batch size, and hence the reduction or amplification of gradient noise, and its distribution are a necessary component of any explanations of the Adam-SGD gap.
- We enhance our intuition by further studying gradient clipping and learning rate grafting [1] affect performance.
- We cross-check our findings in the toy quadratic setup of Zhang et al. [33], and further study how the dynamics of signed gradient descent is influenced by batch size, inspired by recent works on SDE models [4, 20].

Taken together these findings paint a new picture of the optimizer gap, provide practical hints for practitioners in small-scale settings where small batch sizes are the norm, and optimizer memory usage is critical, and constrains future theoretical investigations.

2. Adam vs. SGD: Effects of hyperparameters and training regimes

To systematically investigate the performance gap between Adam and SGD, we conduct a series of experiments in language modeling using a Transformer architecture. Our goal is to understand how this gap evolves under various training regimes and hyperparameter configurations.

2.1. Experimental setup

We conduct all experiments on the SlimPajama webtext dataset using a 160M-parameter LLaMA-style transformer, trained with Adam and SGD with momentum. Full model and training details are provided in Appendix B.

2.2. Effect of batch size on the Adam-SGD gap

We first study how the gap between Adam and SGD changes with batch size under a fixed compute budget, when momentum and learning rate are tuned.

Setup. All experiments use 512-token sequences, a 1.3 B-token budget, and a cosine learning-rate schedule with 10 % warm-up. We test batch sizes of 64, 256, and 1024, tuning the learning rate and momentum β_1 for each, as detailed in the in the Appendix C. Results are reported as the final validation perplexity evaluated on 100M held-out tokens and are shown in Figure 1. When very large learning rates cause instability, we report the best run at the largest stable rate.

Results. Adam shows similar performance across different batch sizes under a fixed token budget. Surprisingly, SGD can match Adam when training with small batch sizes, but the gap increases as batch size grows. For both SGD and Adam, momentum becomes crucial once the batch size is increased, as noted also by Kunstner et al. [13] and Zhao et al. [34].

We also find that using a relatively small sequence length of 512 is not a crucial factor in these dynamics. As we show in the next section, qualitatively the same behavior can be observed when using a sequence length of 2048 – as long as the number of tokens per iteration is held constant. This suggests that performance differences can be attributed to the effective batch size (in tokens) at a given sequence length, rather than sequence length alone. Further analysis is provided in Appendix C.

2.3. Are large batch sizes the problem, or is it the number of steps?

Our previous experiments show that SGD can match Adam in small-batch settings when both optimizers are carefully tuned. Crucially, note that in Figure 1 all methods see a total of 1.3B tokens. This implies that, e.g. at batch size 1024, methods perform 1/16 of the steps compared to batch size 64. This observation raises an important question: does SGD truly break at large batch sizes, or is it simply slower to converge, compared to Adam, at higher batch sizes? In other words, *can SGD reach Adam-level performance even at higher batch sizes, if given more training steps?*

To investigate this, we compare performance across batch sizes under two training regimes: a fixed token budget and a fixed number of steps. This comparison allows us to disentangle the effects of slow convergence from actual poor optimization performance.

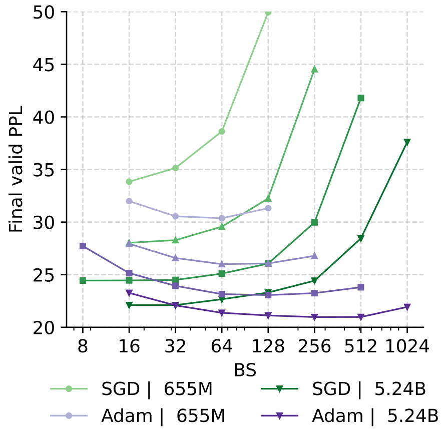

Figure 2: Under a fixed token budget (darker color – more tokens), the gap increases with batch size.

Setup. The experimental setting is the same as in the previous section, except that we increase the sequence length to 2048 to ablate this factor. We train 108 models spanning batch sizes 8–1024 and run lengths 5k–160k steps, each with a 2k-step warm-up. We limit the total token budget between approximately 655M and 5.24B tokens, so larger batches are trained for fewer steps and smaller batches for more. We switch from the cosine scheduler used previously to a WSD scheduler [8], in order to better compare runs before learning rate decay begins. SGD and Adam hyper-parameters follow the tuned configurations in Figure 1. Full details are provided in the Appendix D.

Results: Figure 2 clearly shows that Adam improves with larger batch sizes at a fixed token budget, while SGD shows a drastically opposite trend – performance consistently degrades as batch size increases. Under a fixed token budget, matching performance between Adam and SGD is conditional on using very small batch sizes, leading to significantly longer training and poor memory usage. This result highlights a key limitation of SGD: it is highly inefficient in realistic large-scale language model training, where large batches are required for practical efficiency. In the Figure 3, we show performance after training with various numbers of steps. At the same step count, the gap between Adam and SGD grows with batch size, but SGD improves significantly with more steps and can eventually match or even outperform Adam with long enough training. This illustrates that SGD is not necessarily bad, just very slow to converge in large-batch settings. More results are reported in Appendix D

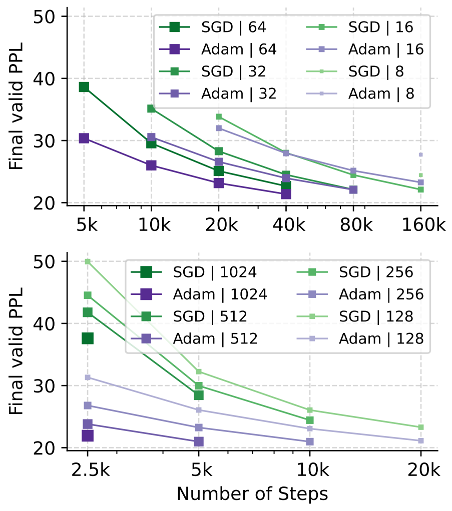

Figure 3: The gap decreases with the number of steps. SGD improves with longer training and can match Adam.

3. Revisiting and extending explanations for the Adam–SGD gap

Several recent works have proposed explanations for Adam's advantage over SGD through the lens of data or architectural properties (see Appendix A), focusing on scenarios where the Adam–SGD gap is particularly pronounced. In contrast, we ask: do these explanations also account for SGD's strong performance in small-batch settings? In this section, we revisit the heavy-tailed class imbalance hypothesis proposed by Kunstner et al. [14], while we present analyses of directional sharpness and Hessian heterogeneity in Appendix E, and show that they struggle to explain good SGD performance. We give theoretical insight based on an SDE-based model in Appendix G.

Complementing this perspective, to understand what limits SGD in large-batch settings, we analyze the roles of update direction and magnitude, using grafting and adaptive clipping. We find that direction is the key issue as we show in Appendix F.

3.1. Heavy-tailed class imbalance

Prior work by Kunstner et al. [13] attributes Adam's advantage over SGD to heavy-tailed class imbalance in token distributions, showing that SGD has difficulty optimizing rare (least common) tokens. We follow their methodology and group all tokens from the training set into 10 frequency groups, from the first group contains the 10% least frequent tokens to the last group contains the 10% most frequent ones.

We apply this analysis to the setting from subsection 2.2, comparing batch sizes 64 and 1024, where SGD performs drastically differently, using runs with the optimal combination of β_1 and learning rate for each case. We find that class imbalance exists in both cases: the persistence of low- and high-frequency tokens is similar, as shown in Figure 5. However, this does not appear to cause

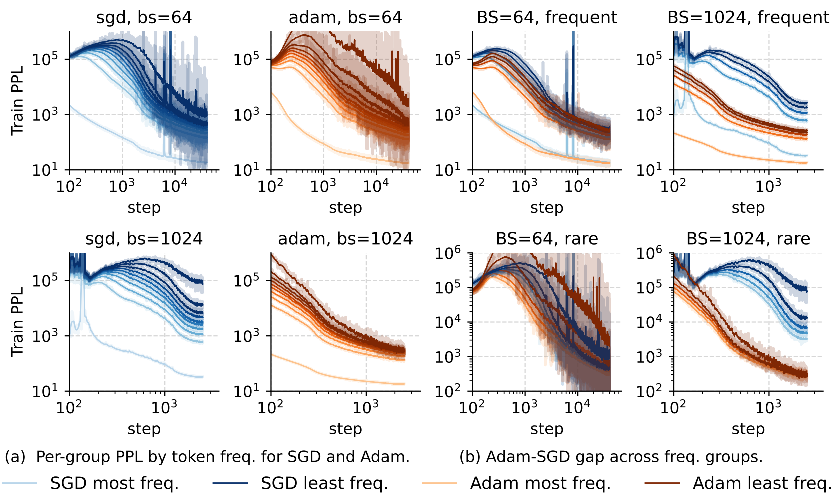

Figure 4: **(a)** Perplexity during training by frequency group, in small- and large-batch settings. **(b)** Adam–SGD gap per group: in large batches, SGD lags behind across all groups–especially on rare tokens. This effect is absent in small batches.

problems for small-batch SGD, suggesting that class imbalance alone does not explain the origin of the Adam–SGD gap.

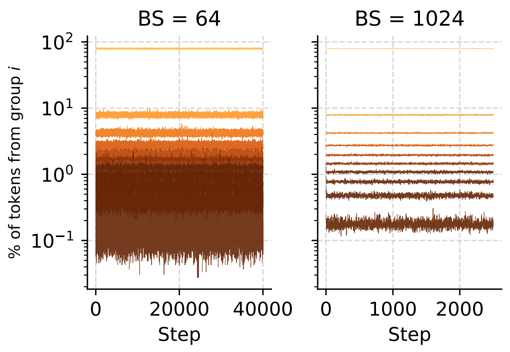

Figure 5: Token distribution for batch sizes 64 and 1024. Lighter colors show less frequent tokens. Token statistics at lower batch sizes are nosier but of similar magnitude.

We further compute perplexity separately for each frequency group and report it over training. From Figure 4 (a), we observe that both optimizers make faster progress on more frequent tokens in all settings, as expected. The relative difference in perplexity between frequency groups is more significant for SGD in the large-batch setting than for the small, while the opposite holds for Adam.

Comparing Adam and SGD across frequency groups in Figure 5 we observe that in the large-batch setting, SGD underperforms Adam across all groups, as shown in Figure 4 (b). However, the gap is notably larger for less frequent tokens, which aligns with findings from Kunstner et al. [13] suggesting that rare tokens could be more challenging for SGD in imbalanced settings. In contrast, this effect is not observed for the small-batch regime in our setting, as also clear from the results in section 2. We would expect this problem of SGD to hold, independent of batch size, but in the setting where SGD works well, the issue disappears.

4. Discussion

Is it impossible to train language models with SGD? We show that in small-batch settings, with tuned momentum and clipping, SGD can be competitive, even for GPT-2-scale models. This challenges common explanations for the Adam–SGD gap. Revisiting prior theories, we find they fall short, and argue instead that gradient noise, amplified by batch size, plays a central role, motivating a stronger explanation based on SDEs.

References

[1] Naman Agarwal, Rohan Anil, Elad Hazan, Tomer Koren, and Cyril Zhang. Disentangling adaptive gradient methods from learning rates. *arXiv preprint arXiv:2002.11803*, 2020.

[2] Kwangjun Ahn, Xiang Cheng, Minhak Song, Chulhee Yun, Ali Jadbabaie, and Suvrit Sra. Linear attention is (maybe) all you need (to understand transformer optimization), March 2024. URL `http://arxiv.org/abs/2310.01082`. arXiv:2310.01082 [cs, math].

[3] Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *ICML*, 2023.

[4] Enea Monzio Compagnoni, Tianlin Liu, Rustem Islamov, Frank Norbert Proske, Antonio Orvieto, and Aurelien Lucchi. Adaptive methods through the lens of SDEs: Theoretical insights on the role of noise. In *The Thirteenth International Conference on Learning Representations*, 2025.

[5] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[6] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[7] Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization, 2018.

[8] Alexander Hägele, Elie Bakouch, Atli Kosson, Loubna Ben Allal, Leandro Von Werra, and Martin Jaggi. Scaling Laws and Compute-Optimal Training Beyond Fixed Training Durations, October 2024. URL `http://arxiv.org/abs/2405.18392`. arXiv:2405.18392 [cs].

[9] Kaiqi Jiang, Dhruv Malik, and Yuanzhi Li. How Does Adaptive Optimization Impact Local Neural Network Geometry?, November 2022. URL `http://arxiv.org/abs/2211.02254`. arXiv:2211.02254 [cs].

[10] Keller Jordan, Yuchen Jin, Vlado Boza, You Jiacheng, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL `https://kellerjordan.github.io/posts/muon/`.

[11] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[12] Ananya Kumar, Ruoqi Shen, Sébastien Bubeck, and Suriya Gunasekar. How to fine-tune vision models with sgd. *arXiv preprint arXiv:2211.09359*, 2022.

[13] Frederik Kunstner, Jacques Chen, J Wilder Lavington, and Mark Schmidt. NOISE IS NOT THE MAIN FACTOR BEHIND THE GAP BETWEEN SGD AND ADAM ON TRANSFORMERS, BUT SIGN DESCENT MIGHT BE. 2023.

[14] Frederik Kunstner, Robin Yadav, Alan Milligan, Mark Schmidt, and Alberto Bietti. Heavy-Tailed Class Imbalance and Why Adam Outperforms Gradient Descent on Language Models, July 2024. URL http://arxiv.org/abs/2402.19449. arXiv:2402.19449 [cs, math, stat].

[15] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[16] Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, et al. Muon is scalable for LLM training. *arXiv preprint arXiv:2502.16982*, 2025.

[17] Tianyi Liu, Zhehui Chen, Enlu Zhou, and Tuo Zhao. A diffusion approximation theory of momentum stochastic gradient descent in nonconvex optimization. *Stochastic Systems*, 2021.

[18] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[19] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

[20] Sadhika Malladi, Kaifeng Lyu, Abhishek Panigrahi, and Sanjeev Arora. On the SDEs and Scaling Rules for Adaptive Gradient Algorithms, February 2023. URL http://arxiv.org/abs/2205.10287. arXiv:2205.10287 [cs].

[21] GN Mil'shtein. Weak approximation of solutions of systems of stochastic differential equations. *Theory of Probability & Its Applications*, 30(4):750–766, 1986.

[22] Lorenzo Noci, Sotiris Anagnostidis, Luca Biggio, Antonio Orvieto, Sidak Pal Singh, and Aurelien Lucchi. Signal propagation in transformers: Theoretical perspectives and the role of rank collapse. *Advances in Neural Information Processing Systems*, 2022.

[23] Yan Pan and Yuanzhi Li. Toward Understanding Why Adam Converges Faster Than SGD for Transformers, May 2023. URL http://arxiv.org/abs/2306.00204. arXiv:2306.00204 [cs].

[24] Thomas Pethick, Wanyun Xie, Kimon Antonakopoulos, Zhenyu Zhu, Antonio Silveti-Falls, and Volkan Cevher. Training deep learning models with norm-constrained lmos. *arXiv preprint arXiv:2502.07529*, 2025.

[25] Ishaan Shah, Anthony M Polloreno, Karl Stratos, Philip Monk, Adarsh Chaluvaraju, Andrew Hojel, Andrew Ma, Anil Thomas, Ashish Tanwer, Darsh J Shah, et al. Practical efficiency of muon for pretraining. *arXiv preprint arXiv:2505.02222*, 2025.

[26] Akiyoshi Tomihari and Issei Sato. Understanding Why Adam Outperforms SGD: Gradient Heterogeneity in Transformers, January 2025. URL http://arxiv.org/abs/2502.00213. arXiv:2502.00213 [cs].

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[28] Nikhil Vyas, Depen Morwani, Rosie Zhao, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham M. Kakade. SOAP: Improving and stabilizing shampoo using adam for language modeling. In *ICLR*, 2025.

[29] Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nati Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. *Advances in neural information processing systems*, 30, 2017.

[30] Hanlin Zhang, Depen Morwani, Nikhil Vyas, Jingfeng Wu, Difan Zou, Udaya Ghai, Dean Foster, and Sham Kakade. How Does Critical Batch Size Scale in Pre-training?, February 2025. URL http://arxiv.org/abs/2410.21676. arXiv:2410.21676 [cs].

[31] Jingzhao Zhang, Tianxing He, Suvrit Sra, and Ali Jadbabaie. Why gradient clipping accelerates training: A theoretical justification for adaptivity, February 2020. URL http://arxiv.org/abs/1905.11881. arXiv:1905.11881 [cs, math].

[32] Jingzhao Zhang, Sai Praneeth Karimireddy, Andreas Veit, Seungyeon Kim, Sashank J. Reddi, Sanjiv Kumar, and Suvrit Sra. Why are Adaptive Methods Good for Attention Models?, October 2020. URL http://arxiv.org/abs/1912.03194. arXiv:1912.03194 [cs, math].

[33] Yushun Zhang, Congliang Chen, Tian Ding, Ziniu Li, Ruoyu Sun, and Zhi-Quan Luo. Why Transformers Need Adam: A Hessian Perspective, June 2024. URL http://arxiv.org/abs/2402.16788. arXiv:2402.16788 [cs].

[34] Rosie Zhao, Depen Morwani, David Brandfonbrener, Nikhil Vyas, and Sham Kakade. Deconstructing What Makes a Good Optimizer for Language Models, July 2024. URL http://arxiv.org/abs/2407.07972. arXiv:2407.07972 [cs].

Appendix A. Related work

Class imbalance. Kunstner et al. [13] explain the advantage of Adam over SGD on language tasks through the heavy-tailed class imbalance in token distributions. They show that the loss for rare tokens decreases significantly more slowly when training with SGD than with Adam, making SGD training inefficient in such settings. In contrast, Adam makes steady progress even on these low-frequency tokens. Their empirical findings show that this holds across different architectures and settings, suggesting that the performance gap is primarily driven by class imbalance. It is not limited to textual data and transformers, since it consistently appears in class-imbalanced scenarios, but vanishes when the data are balanced.

Transformer architecture. Another line of work focuses on the specific characteristics of Transformer architectures. Zhang et al. [33] provide a Hessian-based perspective, showing that Transformers have a block-heterogeneous Hessian spectrum, meaning the Hessian spectrum varies significantly across parameter blocks. In such settings, Adam outperforms SGD by a large margin, while both optimizers perform similarly on architectures with a more homogeneous Hessian. They empirically show that this finding holds across different data modalities and architectures, and that Adam outperforms SGD even on ViT models, differing from the findings of Kunstner et al. [14].

In contrast, Tomihari and Sato [26] focus on gradient heterogeneity, explaining Hessian heterogeneity as a consequence of the correlation between gradients and the Hessian. They observe that in Transformers, a large disparity in gradient norms across parameters leads to optimization challenges for SGD, which can be addressed by Adam's adaptivity.

Zhao et al. [34] find through empirical studies that adaptive optimizers have stable performance over a wide range of hyperparameter settings, while SGD is highly sensitive and often requires careful tuning. They also confirm that full Adam-style adaptivity is not always necessary, showing that their proposed method, which applies adaptivity only to normalization layers and the final layer, can close most of the gap compared to Adam in their setting.

Heavy-tailed gradient noise. Earlier work by Zhang et al. [32] asks whether the nature of stochastic gradient noise explains why the Adam–SGD gap exists in Transformer models but not in other architectures. They show that Transformer models produce gradient noise with heavy-tailed distributions, in contrast to nearly Gaussian noise in CNNs. They argue that heavy-tailed gradient noise degrades the performance of SGD, while Adam demonstrates greater robustness.

However, evidence from Kunstner et al. [14] shows that noise alone is not the primary cause of Adam's superiority, since the gap exists even in the full-batch setting. They find that the performance gap persists, and even that Adam's advantage grows, as stochastic noise vanishes.

Optimization trajectories. Several researchers have investigated how Adam differs from SGD in terms of the characteristics of the loss landscape and the path taken during training. Jiang et al. [9] analyze local geometry along training paths and define a statistic that measures the uniformity of the diagonal of the Hessian. They find that, on language models, Adam's trajectory consistently moves through regions where this measure is significantly smaller than the values found along the SGD with momentum trajectory.

Similarly, Pan and Li [23] introduce the directional sharpness to explain Adam's faster convergence in Transformers. Rather than examining the entire Hessian, they look at the sharpness along the update direction at each step, showing that Adam makes updates in directions with much smaller sharpness than SGD. Although these measures reveal something about the trajectories of SGD and Adam, they cannot fully explain why the performance gap exists.

Evidence from simplified settings. Recent work shows that the Adam–SGD gap persists even in simplified Transformer architectures. Ahn et al. [2] demonstrate that the characteristic optimization challenges mentioned above also appear in shallow linear Transformers, models without nonlinear activations, on a linear regression task.

Appendix B. Experimental setup

We conduct all of our experiments on the SlimPajama webtext dataset using a 160M-parameter LLaMA-like transformer model. The model has 12 attention layers, a width of 768, and 12 attention heads. MLP blocks use a GLU with an expansion ratio of 8/3. The model uses RMSNorm normalization layers and RoPE positional encodings.

Appendix C. Effect of batch size on the Adam–SGD Gap

Setup. All experiments use a sequence length of 512, a fixed token budget of 1.3B tokens, and a cosine learning rate scheduler [18] with 10% warmup. We compare three batch sizes: 64, 256, and 1024. The learning rate and momentum values are tuned for both optimizers at a batch size of 256. A sweep is performed over 5 learning rates and momentum values of 0.9, 0.95, and 0.98, including runs without momentum. High momentum values are motivated by findings from Zhao et al. [34], where SGD performs best with momentum 0.98. Based on the optimal learning rate found at batch size 256, we scale down the learning rate grid for batch size 64 and scale it up for batch size 1024, sweeping over 3 values in each case. Results are reported as the final validation perplexity evaluated on 100M held-out tokens and are shown in Figure 1.

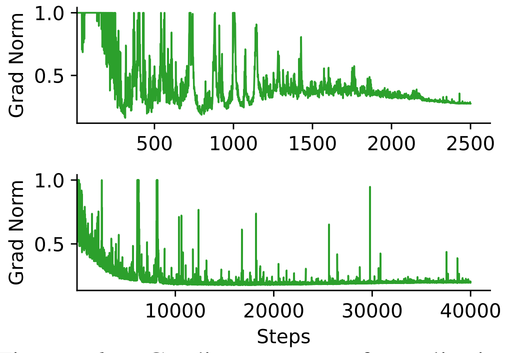

Figure 6: Gradient norm after clipping (threshold 1.0) shows that clipping is more frequent in large-batch training.

Clipping acts differently at different batch sizes. We observe that gradients are clipped more frequently when training with SGD at large batch sizes, as shown in Figure 6. Additionally, at small batch sizes, SGD performs equally well even without clipping; instead, at large batch sizes, training diverges if clipping is not employed.

Warmup length is not a confounder. We also verify that warmup length is not a confounding effect, sweeping 5–20% warmup schedules in our cosine-with-warmup scheduler.

Appendix D. Training under fixed token budget and number of steps

Setup. We train a total of 108 models across a range of batch sizes, from 16 to 256, and run for different numbers of steps, ranging from 5k to 160k. All runs use a fixed warmup of 2000 steps. We switch from the cosine scheduler used previously to a WSD scheduler [8], in order to better compare runs before learning rate decay begins. We limit the total token budget between approximately 650M and 5.2B tokens: larger batches are not trained for the largest number of steps, while the smallest batches are trained only for longer runs. For SGD, we choose a momentum $\beta = 0.98$ and run three

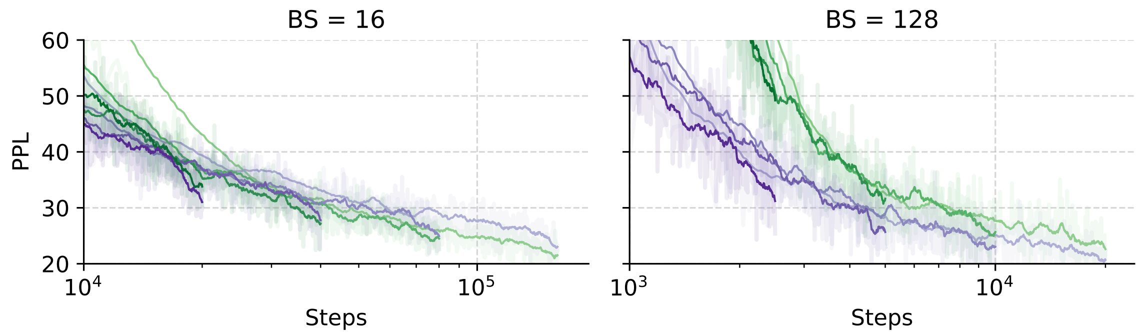

Figure 7: Perplexity during training for SGD (green) and Adam (purple) across different training lengths in small- and large-batch settings. For both, the gap decreases the longer we train. For small batch, at the max number of steps, SGD even performs better than Adam.

distinct learning rates: $1, 0.5$ and 0.25. For Adam, we choose $\beta_1 = \beta_2 = 0.95$ as suggested by modern literature [30, 34] and report performance for the best performing learning rate in the grid $[1e-3, 2e-3, 4e-3]$. Both the SGD and Adam configurations are suggested from our more careful tuning performed in Figure 1.

Results. In addition to Figure 2 and Figure 3, we report perplexity during training for SGD and Adam for batch sizes 16 and 128 in Figure 7. For both, the gap decreases the longer we train, and in the small-batch setting, SGD even performs better than Adam at the maximum number of steps.

Appendix E. Revisiting prior explanations

E.1. Directional sharpness

Pan and Li [23] introduce directional sharpness to explain the optimizer gap by studying a second-order Taylor expansion of the loss along the update direction. In this view, the first-order term (gradient correlation) measures how well the update aligns with the negative gradient, while the second-order term (directional sharpness) measures curvature along that direction. Making optimization progress requires a strong negative gradient correlation and low directional sharpness.

In Figure 8, we visualize gradient correlation, directional sharpness, and their sum – a second-order approximation of loss change, to indicate progress. As in our previous analysis, we compare two settings with drastic performance differences - batch sizes 64 and 1024 from subsection 2.2. In the large-batch setting, SGD shows low gradient correlation and high directional sharpness, resulting in weak or even positive total loss change, as reflected in the sum. In contrast, Adam has higher gradient alignment and lower directional sharpness throughout training. When SGD succeeds, its gradient correlation and directional sharpness closely match Adam's, producing a negative loss change in the sum.

While these metrics align with SGD success or failure, they do not explain why Adam outperforms SGD overall, nor why SGD performs well in small-batch regimes.

E.2. Hessian heterogeneity

From the line of work focusing on the architectural properties of transformers, Zhang et al. [33] argue that the block-wise heterogeneity of the Hessian spectrum is a key factor behind Adam's strong performance and the weakness of SGD. They propose that, based on the Hessian spectrum at

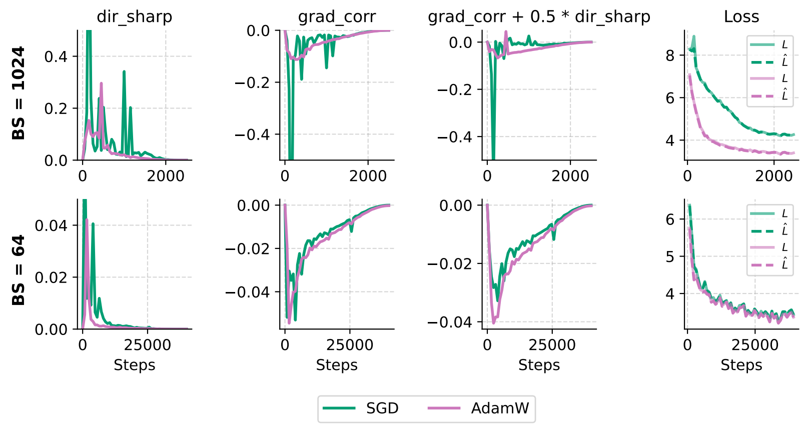

Figure 8: Gradient correlation, directional sharpness, their sum and second-order loss approximation during training, under small- and large-batch settings. Irrespective of the optimizer, good training trajectories have strong negative gradient correlation and low directional sharpness.

initialization, it is possible to predict whether SGD will perform well, offering an explanation that is invariant to batch size. To further explore the effect of batch size on such problems, we revisit the simplified quadratic setting from their work and extend it by including batch size variation. We compare optimization on problems with homogeneous and heterogeneous Hessians, where both share the same eigenvalue spectrum. We train with SGD and Adam using a cosine learning rate schedule and no clipping.

We observe that Adam benefits more from increasing batch size than SGD in both homogeneous and heterogeneous Hessian problems, as shown in Figure 9. On heterogeneous problems, Adam outperforms SGD by a large margin at both small and large batch sizes, as expected. The only setting where they perform on par is in the homogeneous case with small batches. The improvement of SGD at small batch sizes, which we observe in transformers, does not hold in the quadratic setting with a heterogeneous Hessian, suggesting that non-quadratic aspects of the loss landscape may play a critical role.

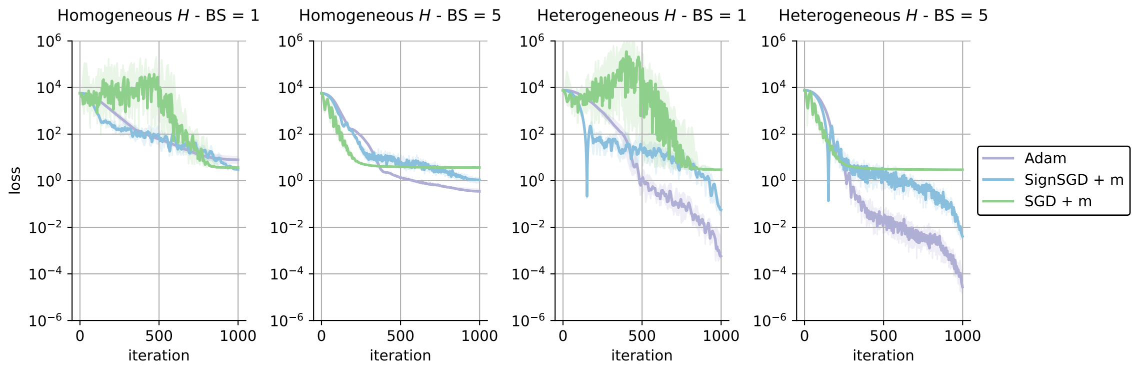

Figure 9: The gap between Adam and SGD relative to batch size also appears when studying only noisy quadratic models in [33], showing that explanations relating to the structure of transformer models or data used for language modeling may not be necessary.

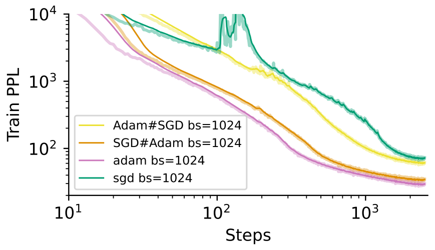

Figure 10: Grafting in large-batch training: using Adam's direction results in performance closer to Adam, while SGD direction leads to results closer to SGD.

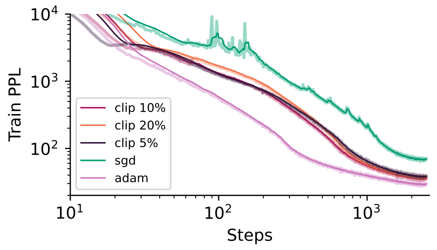

Figure 11: Adaptive clipping with different percentages of clipped coordinates in large-batch training. It improves SGD but still does not fully match Adam.

Appendix F. Understanding what limits SGD performance

We aim to gain more insight into another important question: what goes wrong for SGD in large-batch settings that does not appear at small batch sizes? To investigate this, we attempt to separate which component of the SGD update is more problematic in settings where it fails - its direction or magnitude. In the following experiments, we focus on the setting from subsection 2.2, using batch size 1024 and the optimal combination of β_1 and learning rate.

F.1. Insights from Grafting

To isolate the role of update direction and magnitude, we use the grafting technique proposed by Agarwal et al. [1], which applies the update direction of one optimizer with the magnitude of another. We train the model in the large batch setting, using both combinations: 1) SGD magnitude with Adam direction, and 2) Adam magnitude with SGD direction.We use the optimal $\beta 1$ from subsection 2.2, and sweep the learning rate for the grafted update. We report the training perplexity using the optimal learning rate for both grafting combinations in Figure 10. As shown, using SGD magnitude with Adam's direction performs comparably to Adam, while the reverse combination behaves similarly to SGD. This suggests that the update direction is the more problematic component of the SGD update in large-batch training.

F.2. Insights from Adaptive Clipping

The perspective that direction is the core problem aligns with the observation that global norm clipping does not help much in large-batch training with SGD. If the direction is the main issue, simply rescaling the gradient norm does not lead to better updates.

To investigate this further, we experiment with adaptive clipping, where we clip the top $p\%$ of the largest momentum coordinates at each step, motivated by Pan and Li [23]. We test several values of p (5, 10, and 20). For each value, we keep the optimal β_1 from the previous setting and tune the learning rate. Clipping with $p = 10$ performs best, but we observe that performance does not vary much across different values of p. This method helps reduce the gap between SGD and Adam, as shown in Figure 11. The result suggests that a subset of larger update coordinates consistently contributes to poor update directions and slows down SGD in large-batch training, whereas updates in small-batch settings not clipped as often, relative to the number of steps, so that training is not decelerated.

Appendix G. Theoretical insights.

Towards explaining the phenomena observed in this paper, we provide a preliminary yet insightful analysis based on [4], based on the observed similarity between behaviors of Adam and SignSGD in Figure 9, as well as recent literature on their relation [10, 13].

Let X denote the model parameters and γ denote a batch of size B. We denote the stochastic gradient as $\nabla f_\gamma(x) := \frac{1}{B}\sum_{i\in\gamma}\nabla(f_i(x))$ and by $\Sigma(x)$ the noise covariance at batch size 1. The stochastic differential equation (SDE) approximation of SGD reads [17, 21]

$$dX_t = -\nabla f(X_t)dt + \sqrt{\frac{\eta\Sigma}{B}}dW_t, \tag{1}$$

We now state a recent result showing that the drift of signed updates – driving performance in early training – has an extra dependency on the batch size.

Theorem 1 ([4]) *Under the assumption of i.i.d. Gaussian noise, the following SDE provides a* 1*-weak approximation [21] of the discrete update of SignSGD*

$$dX_t = -\operatorname{erf}\left(\sqrt{\frac{B}{2}}\Sigma^{-\frac{1}{2}}\nabla f(X_t)\right)dt + \sqrt{\eta}\sqrt{I_d - \operatorname{Diag}\left(\operatorname{erf}\left(\frac{\sqrt{B}\Sigma^{-\frac{1}{2}}\nabla f(X_t)}{\sqrt{2}}\right)\right)^2}dW_t, \tag{2}$$

where the error function $\operatorname{erf}(x) := \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt$ *and the square are applied component-wise.*

While Compagnoni et al. [4] provide a similar result for the Heavy-tail setting, the Gaussian case already highlights a crucial distinction between signed gradient methods and classical SGD: recall that erf (similar in shape to a $\tanh$) is linear on a large interval around zero. The local update of parameters is then driven by $-\operatorname{erf}\left(\sqrt{\frac{B}{2}}\Sigma^{-\frac{1}{2}}\nabla f(X_t)\right)$ in the signSGD case, while in the SGD setting, this term is simply $-\nabla f(X_t)dt$. This analysis provides strong evidence for our results: using large batch sizes accelerates convergence (larger drift) in signSGD, while the performance of SGD in early training is batch-size agnostic and hence driven by the number of iterations.