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

Anonymous authors
Paper under double-blind review

Abstract

Adam is often observed to significantly outperform Stochastic Gradient Descent (SGD) on Transformer-based language models, a phenomenon that has puzzled optimization researchers for years and for which several explanations have been proposed. In this work, we revisit this "optimizer gap" by training over 1000 Transformer language models and studying how batch size, training budget, model scale, momentum, and gradient clipping affect rankings. The resulting picture is not a monolithic Adam-SGD separation: at a fixed token budget, SGD is inefficient compared to Adam when common batch size values are used; however, while Adam's performance is only marginally affected when reducing the batch size, well-tuned SGD quickly catches up in this regime, drastically shrinking the gap. Interestingly, SignSGD with momentum behaves similarly to Adam as the batch size varies. We revisit existing explanations for the optimizer gap, including heavy-tailed class imbalance, directional sharpness, and heterogeneity. These lenses describe important aspects of SGD failure, but do not explain why increasing stochasticity can shrink the Adam-SGD gap without performance deterioration. Motivated by this, we study SignSGD not as a complete model of Adam but as the simplest algorithm that exhibits the relevant batch-size dependence. Combining intuition from stochastic differential equations with nonconvex bounds, we isolate a simple mechanism directly affecting convergence rates: signed methods have a batch-dependent mean update direction, whereas SGD's drift is unchanged by batch size.

1 Introduction

The Adam optimizer (Kingma & Ba, 2014) is used pervasively in deep learning, especially when training large language models (LMs) (Grattafiori et al., 2024; Liu et al., 2024; Biderman et al., 2023) and vision Transformers (Dosovitskiy et al., 2020; Kumar et al., 2022) at scale. Industrial practice relies on Adam's success, and thousands of GPU-hours every day are spent on Adam-based training. Its role remains central even in optimization pipelines designed to move beyond Adam. For example, recent Muon (Jordan et al., 2024) implementations often still use Adam with weight decay (AdamW, Loshchilov & Hutter (2019)) on critical subsets of parameters, such as normalization layers, text embeddings, prediction heads or decoupled gain parameters (Liu et al., 2025; Shah et al., 2025; Lion et al., 2026; Hübler et al., 2026; Hägele et al., 2026). This new world is still a bit surprising: until around the year 2018, before Transformers became standard, the Adam optimizer was used only occasionally, while stochastic gradient descent (SGD) with momentum was known to yield neural networks with better accuracy on unseen data (Wilson et al., 2017), relegating Adam to speed runs and quick comparisons (Goyal et al., 2017). Yet, from the start, Transformer LMs have often been reported to be difficult to train with SGD (Xiong et al., 2020).

Over the years, optimization researchers have offered a number of compelling explanations regarding the remarkable performance of Adam compared to SGD in modern language modeling, often attributing it either to the peculiar nature of text data (Zhang et al., 2020b;a; Kunstner et al., 2024; Kunstner & Bach, 2026) or the structure of the Transformer architecture (Vaswani et al., 2017), comprising semantically and structurally dissimilar layers (Noci et al., 2022; Zhang et al., 2024; Zhao et al., 2024), with a potential depth-dependent anisotropic geometry (Ahn et al., 2024). In this complex research landscape, an insight by Kunstner et al. (2023) is particularly intriguing: the Adam-SGD gap is also observable in full-batch training on small-scale

datasets. As a result, later papers attribute the gap to the distribution of text data (causing ill-conditioned Hessians) (Kunstner et al., 2024; Kunstner & Bach, 2026) rather than its implied gradient noise.

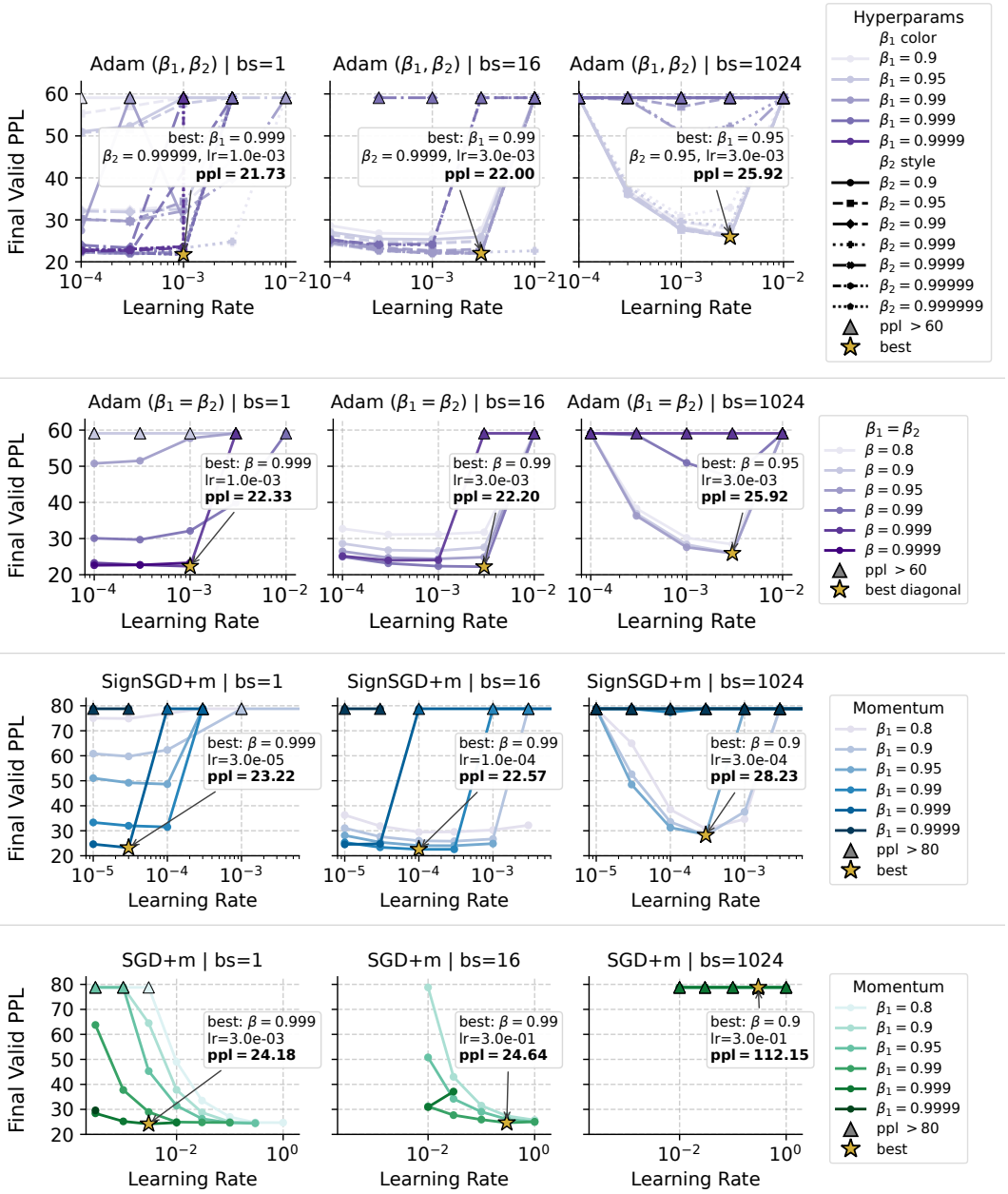


Figure 1: Aggregated result from 677 runs. Language modeling with a standard Transformer (Section A for details) with 160M parameters. We train at 1× Chinchilla-optimal (Hoffmann et al., 2022) budget (larger budgets appear later, e.g. in Figures 4 and 6), i.e., on 3.2B SlimPajama (Soboleva et al., 2023) tokens, with a sequence length of 2048. We report validation perplexity on 100M held-out SlimPajama tokens. All methods use cosine annealing for the learning rate and learning rate warmup (10%); weight decay is omitted to avoid inflating the search space and introducing a confounder. After extensive tuning, we notice that (1) **SGD with momentum (SGD+m) works well in low batch size settings, but is extremely suboptimal compared to Adam if large batch sizes are used**; (2) In Adam, $\beta_1 = \beta_2$ (Orvieto & Gower, 2025) leads to near-optimal performance across batch sizes, yet large β_2 values help achieve the best performance; (3) SignSGD+m, conceptually linked to Adam with $\beta_1 = \beta_2$, recovers satisfactory near-Adam performance in the high batch size setting. A more detailed plot for Adam performance at different values of β_1 is provided in Figure 17. Training curves for the best tuning configuration in each setup are reported in Figure 2.

Building on the full-batch insight of Kunstner et al. (2023), we use batch size as a complementary theoretical probe for the optimizer gap. Their result shows that the gap cannot be explained by minibatch noise alone; at the same time, a full account of the gap should also explain how and why it changes as stochasticity is varied. We therefore ask: *how does the Adam-SGD gap evolve across batch-size regimes, and what mechanism accounts for this dependence?* In the context of critical batch size[1] analyses, Zhang et al. (2019) already pointed out that, in standard supervised learning, deterministic Hessian-based preconditioning increases the algorithm's critical batch size, suggesting different batch-size sensitivities for methods that can be interpreted as preconditioning, such as Adam. More recently, Marek et al. (2025) showed that, with proper batch-size-aware tuning, very small batches can train LMs stably and match or outperform larger batches. They further argue that small batches reduce the need for sophisticated optimizers: vanilla SGD becomes competitive, while gradient accumulation can be wasteful unless needed for multi-device throughput.

Taken together, these works make batch size a natural axis for understanding the Adam–SGD gap, but leave open how batch size reshapes optimizer rankings in realistic LM pretraining and what mechanism explains this dependence: Zhang et al. (2019) connect batch size and deterministic Hessian-based preconditioning, but in supervised learning tasks far from modern LM practice; Kunstner et al. (2023) show that the gap can persist even without minibatch noise, but focus on small-scale datasets where full-batch training is feasible; Marek et al. (2025) provide compelling evidence that small batches can make SGD competitive, but their investigation focuses on efficiency, hyperparameter transfer and numerical stability, rather than theoretical insights and harmonization with previous research on the Adam-SGD gap.

Through a series of ablations, we show that batch size changes optimizer rankings in language-model pretraining, and isolate a mechanism absent from SGD: for signed methods, reducing gradient noise changes the expected update direction. Our paper is organized as follows:

- In Subsection 2.2–Subsection 2.4, we study the Adam-SGD gap under standard Chinchilla-optimal training: large batches hurt SGD, while SignSGD remains close to Adam, and low-batch SGD stays competitive even up to 1B parameters. In Subsection 2.3, we show that SGD does not fail to optimize at large batch sizes; rather, it improves more slowly while scaling similarly across batches. Finally, in Subsection 2.5–Subsection 2.6, we move beyond Chinchilla-optimal training and show that SignSGD behaves consistently across batch sizes, whereas SGD exhibits batch-dependent traces. Without momentum, this contrast becomes sharper and motivates the theoretical intuition in Section 5.

- In Section 3 we carefully revisit prior explanations, such as heavy-tailed class imbalance (Kunstner et al., 2024), directional sharpness (Pan & Li, 2023), and Hessian heterogeneity (Zhang et al., 2024), in our setup. While our experiments confirm that these explanations shed light on settings where Adam outperforms SGD, we find that prior explanations do not directly explain why carefully tuned SGD can match Adam at low batch sizes *without degrading performance*. In Section 4 we deepen our intuition by examining what makes SGD suboptimal and potentially unstable at high batch sizes. To do this, we present ablations on warmup length, gradient clipping, and direction-magnitude grafting (Agarwal et al., 2020), and examine their effects on performance.

- We study SignSGD without momentum as a minimal model to isolate a batch-size-sensitive mechanism that is also visible in Adam-like behavior. Our analysis is grounded in our experimental results on SignSGD without momentum, which we found to scale similarly to Adam (Subsection 2.6). Our intuition comes from recent stochastic differential equation (SDE) models of adaptive optimization (Malladi et al., 2023; Compagnoni et al., 2025) and isolates a simple mechanism: increasing the batch size changes the *mean update direction* of signed methods. We first make this intuition precise via the SDE approximation in Theorem 1, and then via the discrete decomposition and bounds in Subsection 5.2.

Of course, our goal is not to give prescriptions for large-scale LM training, where throughput, parallelism, and wall-clock constraints make very small batches impractical. The purpose of our experiments is to probe theoretical gaps, rather than to make recommendations for practical choices.

[1]The critical batch size is the workload/optimizer-dependent batch size where "perfect scaling" ends: doubling the batch size no longer halves the steps needed to reach a fixed target loss (Shallue et al., 2019).

1.1 Related work

Transformer architecture. Early works already show that training deep Transformers with SGD is challenging (Xiong et al., 2020). Noci et al. (2022) claim this is due to learnable parameters inside softmax attention, which can have significantly smaller gradients (and hence need per-layer adaptive learning rates) under rank-collapse, a Transformer-specific phenomenon (Dong et al., 2021). Zhang et al. (2024) provide a more comprehensive Hessian-based perspective, showing that Transformers have a block-heterogeneous Hessian spectrum, i.e., 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. Tomihari & Sato (2025) 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 Adam's adaptivity can address. Through empirical studies, Zhao et al. (2024) find that adaptive optimizers exhibit stable performance across a wide range of hyperparameter settings, whereas SGD is highly sensitive and often requires careful tuning. They also show that Adam-style adaptivity is not necessary in all layers: applying Adam only to normalization layers and the final layer can close most of the gap compared to Adam in their setting. Finally, recent work shows that the Adam-SGD gap persists even in simplified Transformer architectures. Ahn et al. (2024) demonstrate that the characteristic optimization challenges mentioned above also appear in shallow linear Transformers, models without nonlinear activations, on a linear regression task.

Impact of gradient noise and text data. Early work by Zhang et al. (2020b) asks whether the nature of stochastic gradient noise can explain why an Adam-SGD gap can be observed in Transformer-based language models, but not in other architectures such as convolutional networks (CNNs) for image classification. They show that Transformer models trained on text produce gradient noise with heavy-tailed distributions, in contrast to the nearly Gaussian gradient noise of CNNs. Heavy-tailed gradient noise degrades SGD performance, whereas Adam is more robust because it effectively clips magnitudes (Zhang et al., 2020a). However, evidence from Kunstner et al. (2023) suggests that heavy-tailed noise is unlikely to be the sole reason for Adam's superiority: on small datasets, a gap can be observed even in the full-batch setting. In their setup, Adam benefits more from the large- and full-batch regimes, while SGD improves comparatively little beyond moderate batch sizes. In addition, sign descent with momentum (Bernstein et al., 2018) qualitatively follows Adam's large-batch behavior, as would be expected from early works establishing a connection between these methods (Balles & Hennig, 2018). Our work is closely motivated by Kunstner et al. (2023), yet rather than discussing the noiseless limit, we examine a wide range of batch sizes and token budgets in academic-scale yet realistic causal language model pretraining setups where one is far from full-batch training.

Class imbalance. Connecting optimization landscape and data-distribution perspectives, Kunstner et al. (2024) explain the advantage of Adam over SGD on language tasks through the heavy-tailed class imbalance in token distributions — causing a non-isotropic Hessian in specific layers. They show that the loss for rare tokens decreases slowly when training with SGD. In contrast, Adam makes steady progress even on these low-frequency tokens. Their findings are not limited to language modeling and Transformers; they extend to other architectures and settings, suggesting that the performance gap is primarily driven by class imbalance in datasets, which effectively modifies the loss landscape and favors adaptivity. A theoretical result demonstrating this property was later provided by Kunstner & Bach (2026) in an axis-aligned quadratic setup, where Hessian eigenvalues are shaped by the Zipf distribution of tokens in text.

Batch size sensitivity. As discussed, Kunstner et al. (2023) showed that both Adam and SignSGD with momentum can perform substantially better than SGD in full-batch training of Transformer models, opening up a discussion around batch-size sensitivity for different optimizers in language modeling, which we develop in this paper. This discussion relates to recent literature on optimal and critical batch sizes in large-scale settings (Li et al., 2025; Shulgin et al., 2026; Islamov et al., 2026). Zhang et al. (2019) were perhaps the first to study in detail the effect of Hessian-based gradient preconditioning on the critical batch size for gradient methods. Their theoretical analysis, inspired by empirical results on standard supervised learning tasks, can help explain why some forms of preconditioning can extend the range over which increasing batch size reduces the number of steps to a target loss. This work provides an important abstraction, but its analysis is limited

to deterministic Hessian-based preconditioning; moreover, Adam cannot be cleanly reduced to Hessian-based preconditioning, as shown, e.g., in Kunstner et al. (2019).

Optimization trajectories. Several researchers have investigated how Adam differs from SGD by characterizing the training trajectory. Jiang et al. (2022) analyze local geometry along training paths and define a statistic that measures the uniformity of the Hessian diagonal. On LMs, they find that Adam's trajectory consistently moves through regions where this measure is significantly smaller than the values found along the trajectory of SGD with momentum. Similarly, Pan & Li (2023) introduce directional sharpness as a metric 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.

2 Experimental Results

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

2.1 Outline

We conduct most of our experiments on SlimPajama (Soboleva et al., 2023) and FineWeb datasets (Penedo et al., 2024) using a nanoGPT (Karpathy, 2022) implementation augmented with standard modern LM components: plainLM (Ajroldi, 2024). MLP blocks use a GLU (Shazeer, 2020) with an expansion ratio of 8/3. The model uses RMSNorm normalization (Zhang & Sennrich, 2019) layers and RoPE positional encodings (Su et al., 2024). Full details are provided in Appendix A. We experiment with models up to 1B parameters, following the Pythia configuration (Biderman et al., 2023) for depth/width ratios.
Explanations for the Adam-SGD gap should not be contingent on the use of weight decay; for this reason, we do not apply weight decay in any of our experiments, thereby avoiding the expansion of our already large hyperparameter grid. For both SGD and Adam, we perform global gradient-norm clipping on the raw gradients (i.e., before applying momentum) unless otherwise stated. Unless explicitly stated otherwise, SGD denotes SGD with momentum (SGD+m); in Subsection 2.6, SGD and SignSGD denote their no-momentum variants. Other details regarding sequence length, batch size, training budget, and hyperparameter grids are reported directly in the respective sections.

The section is organized as follows:

- In Subsection 2.2 we present our largest ablation, sweeping learning rate and momentum parameters for SGD+m, Adam, SignSGD+m in a Chinchilla-optimal setting (20 tokens per parameter) under cosine decay and standard warmup. We find that high batch sizes hurt SGD+m, yet SignSGD+m can deliver performance close to Adam in this setting. At low batch sizes, SGD is competitive.
- In Subsection 2.3 we inspect whether SGD "*just cannot optimize at large batch sizes*". We find this to be false: SGD is just slower in this setting, yet performance improvements scale similarly across batch sizes.
- In Subsection 2.4 we verify that SGD at low batch sizes is still competitive at larger scales. We present results here for a 1B-parameter model.
- In Subsection 2.5, we build on an insight from Subsection 2.2: SignSGD+m is much less sensitive to large batch sizes. We then compare SignSGD+m with SGD+m on a setup where training curves are comparable at each token budget, i.e., when training with a constant learning rate for a larger number of tokens ($4\times$ Chinchilla-optimal). In this setup, we see that well-tuned SignSGD+m evolves similarly in both large- and small-batch regimes, as a function of the number of training tokens. In SGD+m, traces are instead distinct.
- In Subsection 2.6, we ask whether momentum is necessary to study this behavior. We then repeat the analysis in Subsection 2.5 under no momentum. We find that, as a function of the optimizer step, SignSGD accelerates drastically as the batch size increases, whereas SGD shows only a moderate improvement at high batch sizes. This empirical minimal experiment is our starting point for the theory in Section 5.

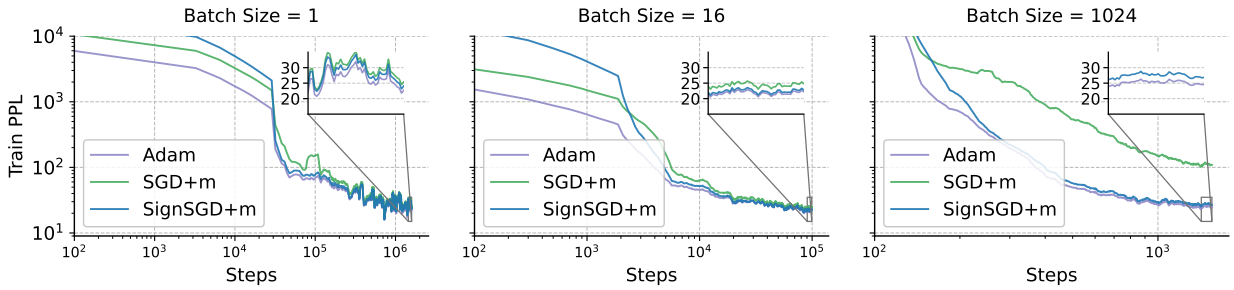


Figure 2: Reported are the best-performing configurations resulting from the sweep in Figure 1. At high batch sizes, SGD+m is substantially worse than both SignSGD+m and Adam. The gap drastically shrinks at low batch sizes. We compare loss curves across different batch sizes for each optimizer in Figure 6 at a larger token budget, using a constant learning rate.

2.2 Fixed token budget with heavy tuning: poor batch size sensitivity for SGD

We first study how the gap between Adam and SGD varies with batch size under a fixed compute budget, with momentum and learning rate carefully tuned. The results are shown in Figure 1 and Figure 2.

Setup. We train a standard 160M-parameter Transformer at a 1× Chinchilla-optimal budget (Hoffmann et al., 2022), using 3.2B SlimPajama tokens and sequence length 2048. All methods use cosine learning-rate annealing with 10% warmup. We also test whether SignSGD with momentum is a good model for Adam as the batch size varies (Kunstner et al., 2023), and if choosing $\beta_1 = \beta_2$ in Adam leads to robust performance.

Results. After extensive tuning, we find that: (1) SGD with momentum (SGD+m) performs well at small batch sizes but is highly suboptimal relative to Adam at large batch sizes; (2) for Adam, $\beta_1 = \beta_2$ (Orvieto & Gower, 2025) is near-optimal across batch sizes, though large β_2 helps achieve the best performance; and (3) SignSGD+m, theoretically related to Adam with $\beta_1 = \beta_2$, has near-Adam performance at large batch size. Figure 17 provides a more detailed sweep over Adam β_1. Momentum in all methods becomes crucial as the batch size decreases, as noted by Kunstner et al. (2023); Zhao et al. (2024); Marek et al. (2025).

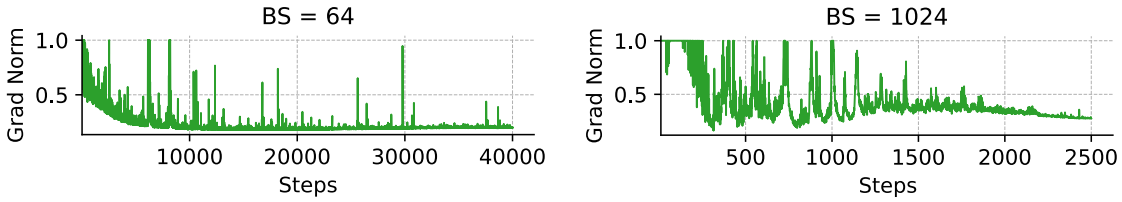


Figure 3: SGD+m gradient norm after clipping (threshold 1.0): clipping is more frequent for large batches.

Clipping and warmup. The clear outlier in Figure 1 is SGD+m at batch size 1024. Gradients are always clipped to unit norm before being fed to optimizers; hence, we wonder how much this operation affects performance in this setting. We observe (Figure 3) that gradients are clipped more frequently when training with SGD at large batch sizes. In fact, we found that at small batch sizes, SGD performs equally well even without clipping; however, at large batch sizes, training diverges without clipping. In Figure 18, we present 112 additional runs to ablate the impact of clipping and warmup on large-batch SGD+m, finding that careful tuning can indeed help, yet performance does not approach that of Adam and SignSGD+m.

2.3 When does SGD perform well?

Our previous experiments in Subsection 2.2 show that SGD can perform similarly to Adam in small-batch settings, with a fixed model size and training budget. This observation raises an important question: *can SGD reach low perplexity also at higher batch sizes, if given more training steps? Or is large-batch training fundamentally broken?* To investigate this beyond the restrictive yet heavily ablated setting of Figure 1,

we inspect here models of several sizes, trained on different data (FineWeb and SlimPajama), for different numbers of tokens. We compare performance across batch sizes under two training regimes: (1) a fixed token budget and (2) a fixed number of steps. This comparison allows us to disentangle the effects of slow convergence from actual poor optimization performance.

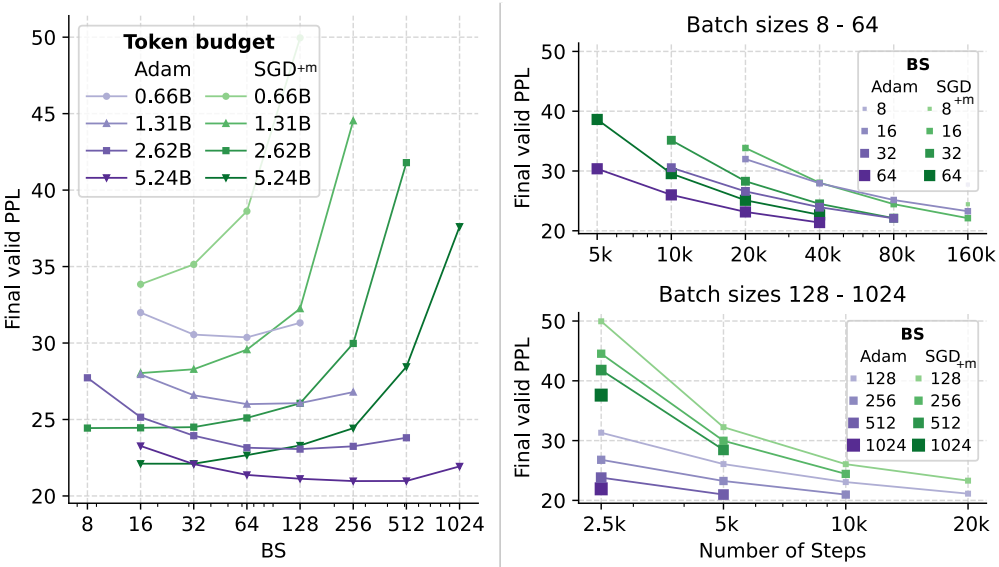


Figure 4: SGD+m and Adam performance (160M parameters, WSD scheduler, SlimPajama with a sequence length of 2048) across batch sizes. Momentum parameters are fixed; the learning rate is lightly tuned; see the note on experimental design. **(Left)** Fixed token budget (darker colors = more tokens); the gap increases with batch size across all token budgets. **(Right)** Fixed number of steps (darker colors = larger batch sizes); the gap decreases with the number of steps. SGD improves with longer training and can match Adam, given a sufficiently small batch size. Further details, training curves comparisons, and runs for different depths and training datasets can be found in Subsection A.3.

Note on experiment design. Given the large set of configurations explored, we limit our investigation to SGD with momentum $\beta = 0.98$ (Zhao et al., 2024) and Adam with $(\beta_1, \beta_2) = (0.95, 0.95)$: this configuration works uniformly well in Figure 1, and was also suggested by Orvieto & Gower (2025); Zhang et al. (2025). Although here we tune learning rates, as observed in Figure 1, higher momentum parameters can be beneficial, especially with Adam, at lower batch sizes. *Here, however, our objective is not to achieve optimal tuning for each configuration, but to show qualitative trends as the budget increases.* While in Figure 1 we tune momentum heavily, here we sweep the batch size extensively. We later provide, in a restricted setup, more careful ablations with momentum tuning at an even higher token budget (4× Chinchilla, Subsection 2.5).

Optimal batch size for Adam and SGD (fixed momentum). Recent literature (Shulgin et al., 2026) has pointed out that LMO-based methods (Pethick et al., 2025), such as SignSGD — with an arbitrary yet *fixed* momentum parameter (see previous paragraph) — achieve optimal convergence rates, as the token budget increases, once the batch size is chosen to grow with the token budget. In other words, the optimal batch size for adaptive methods grows with token budget, while for SGD a batch size of 1 is optimal (Garrigos & Gower, 2023). While we go back to similar theoretical considerations in Section 5, we note already that Figure 4 exactly shows this trend, related to modern findings in the literature on batch size scaling (von Rütte et al., 2025; Li et al., 2025; Islamov et al., 2026).

Setup. The experimental setting for Figure 4 is the same as in Subsection 2.2, except for learning rate scheduling: we consider here, to ablate on this component and allow finite-time comparison between loss curves (see Figure 19), a WSD scheduler (Hägele et al., 2024) with a fixed warmup of 2000 steps and linear learning rate rampdown for 20% of the token budget. We train models across a range of batch sizes, from 8 to 1024, and run for different numbers of steps, ranging from 2.5k to 160k. The resulting total token budget

is between approximately 650M and 5.2B tokens: models using larger batches are not trained for the largest number of steps, while models using the smallest batches are trained only for a large number of steps. For SGD (with momentum), we report the best performance among three distinct learning rates: $1, 0.5$, and 0.25. For Adam, we choose to report performance among the best performing learning rates in the grid $[1 \cdot 10^{-3}, 2 \cdot 10^{-3}, 4 \cdot 10^{-3}]$. The learning rate grids are inspired by inspection of Figure 1.

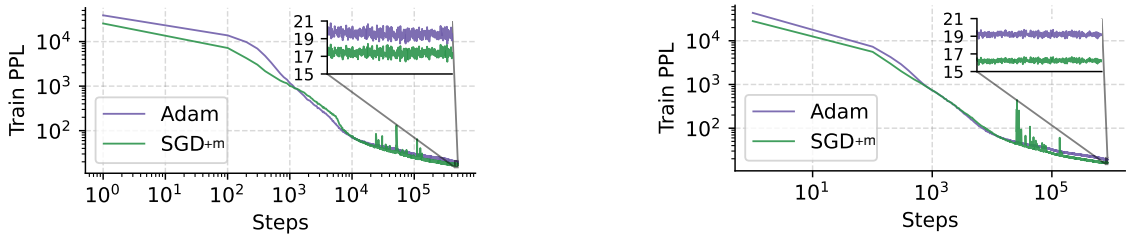
Results. The left panel of Figure 4 clearly shows that the insight in Figure 1 holds across different token budgets: SGD+m becomes extremely inefficient for large batch sizes, but its performance drastically improves at low batch sizes, outperforming Adam when its (β_1, β_2) are close to literature defaults. In the right panel of Figure 4, we show performance after training with varying numbers of steps: at large batch sizes, *SGD improves significantly with more steps*. In addition to Figure 4, we report training curves for SGD and Adam with batch sizes 16 and 128 in Figure 19, to be compared with Figure 2.

We repeat this experiment on FineWeb in Figure 20 (left), with similar findings.

2.4 Do we observe similar trends in larger models?

Findings in Subsection 2.3 also hold for deeper models in the same setup, as shown in Figure 20 (right), where we replicate Figure 4 for models with 24 layers.

Finally, to test whether the good performance of small batch SGD persists at scale, we experiment with larger models in Figure 5: 410M- and 1B-parameter models. Setup details are reported in Appendix A and Subsection A.3.



(a) 410M model on SlimPajama (seq. length 2048, batch size 8, 500k steps) – 1.5 days of training.

(b) 1B model on FineWeb (seq. length 1024, batch size 16, 850k steps) – 5 days of training.

Figure 5: SGD can outperform Adam with standard βs and a tuned learning rate, even at the 410M and 1B scales, in small-batch regimes. Learning rate tuning plots are available in Figure 21.

2.5 Simplifying the picture: SignSGD vs SGD

In Subsection 2.2 and Subsection 2.3 we established that SGD can train deep models, competing with Adam at low batch sizes and improving even at higher batch sizes as the token count grows. We now dive deeper into this phenomenon, leveraging a crucial finding in Figure 1: unlike SGD+m, SignSGD+m performs well across batch sizes, similarly to Adam. The experiments in this section aim to sharpen this finding. We further isolate a minimal setup in Subsection 2.6.

Setup. We train the same model as Figure 1 and Figure 4, but this time for a substantially larger budget of 13B tokens (4× Chinchilla-optimal) from FineWeb, using a sequence length of 1024 and a batch size of either 16 or 512. After a linear warm-up of 0.25B tokens, we use a constant learning rate: *this allows us to compare training curves at finite time*, aligning with theoretical investigations such as those by Shulgin et al. (2026); Islamov et al. (2026), as well as our discussion in Section 5. As detailed in Subsection A.4, we tune both momentum and learning rate thoroughly, with a similar procedure as for Subsection 2.2. The total compute for producing Figure 6 amounts to approximately one hundred NVIDIA A100-SXM4-80GB GPU days.

Results. As also detailed in the caption of Figure 6, we observe that optimally tuned SignSGD+m delivers good performance at both low and high batch sizes as training progresses. Instead, for SGD+m, large batch sizes always lag by approximately 4B tokens. Consistent with the findings of Subsection 2.3, we confirm that

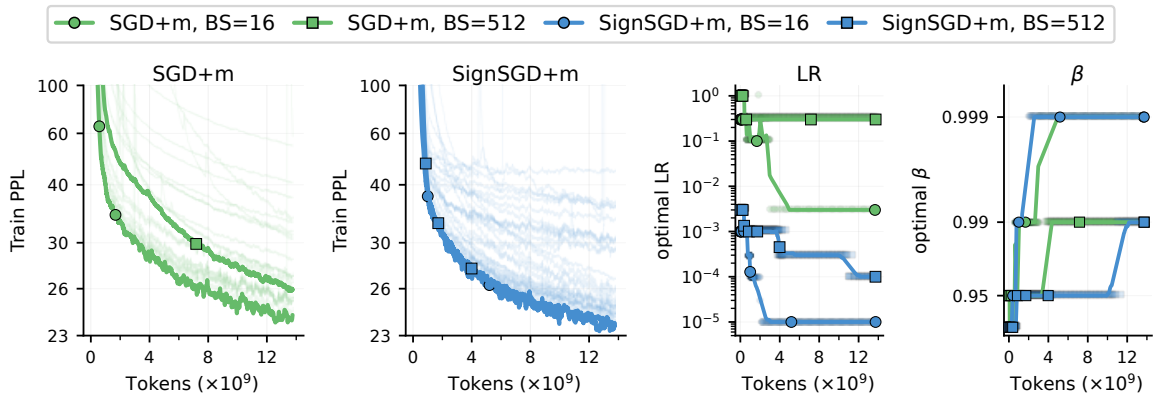


Figure 6: Performance comparison for SGD+m and SignSGD+m at different batch sizes. We train the same Transformer as in Figure 1 for 13B tokens from FineWeb at a sequence length of 1024, using a constant learning rate with warmup. We tune the learning rate and momentum heavily and report **(right)** how optimal choices depend on the token count. Trends align with the findings in Mlodozeniec et al. (2025); Shulgin et al. (2026). **(Left)** We show in light colors all training curves for all configurations, *boldfacing the lower envelope (Pareto frontier)* for each batch size choice. As expected from Figure 1 and Subsection 2.3, while SignSGD performance is (as token budget increases) almost independent of batch size, in SGD the gap between small and large batch sizes is consistent and does not shrink as training progresses. Full training curves for each configuration are provided in Figure 22 in the Appendix.

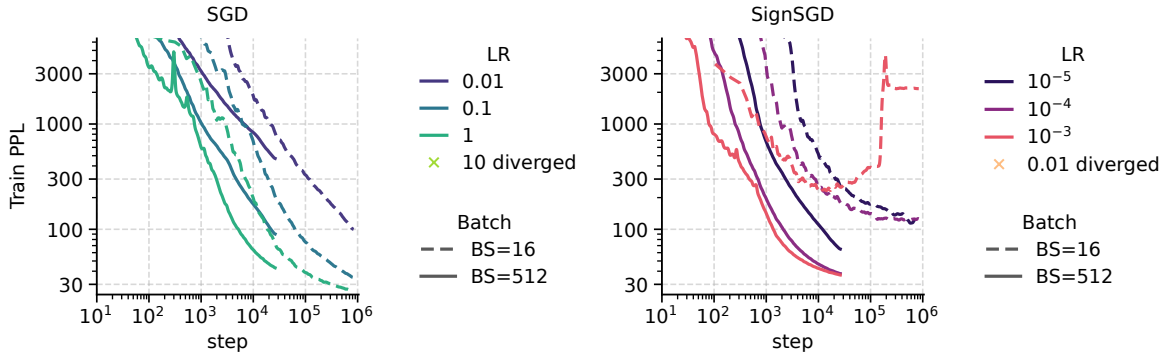


Figure 7: We plot the performance (as a function of training step) for SignSGD and SGD at different batch sizes, *both without momentum*. SignSGD can drastically accelerate convergence at high batch sizes, showing a preference for lower noise in early training. This can also be observed in a toy quadratic setup in Figure 16 (inspired by language modeling), and is in agreement with our theoretical results in Section 5.

even at higher token budgets, SGD+m with large batch sizes consistently reduces the loss, with a 4B-token gap that accounts for a decreasing percentage of the training tokens as training progresses.

2.6 Even simpler: is momentum necessary to study the gap?

All our experiments, especially Figure 1 and Figure 6, show that high momentum is especially important for small batch sizes and large training horizons. However, now we ask: *is studying SGD and SignSGD with momentum necessary to understand why they react to batch size differently?* To inspect this, we present one last set of experimental results. Our setup here is exactly the same as in Figure 6, except we set the momentum to zero in all runs.

In Figure 7 we report the training PPL as a function of the optimizer step for tuned SGD and SignSGD, both without momentum, trained for 13B tokens. We chose here to plot performance against step to show a

clear trend: while SGD performance only mildly improves with batch size as a function of step[2], SignSGD effectively accelerates at higher batch sizes: the curves separate drastically in early training, as also shown in our quadratic model in Figure 11.

This finding is our starting point for our theoretical analysis in Section 5. Before proceeding, however, we revisit prior literature and assess whether its claims remain robust in light of our results.

3 Ablations revisiting prior literature

Several works have offered compelling explanations for Adam's advantage over SGD through the lens of data or architectural properties (see Subsection 1.1). These perspectives have substantially improved our understanding of the optimizer gap, especially in regimes where the gap is pronounced. We revisit them from a complementary angle and ask: *Can they also account for the strong performance of carefully tuned SGD in small-batch settings, as observed in Section 2?*

Our evidence in this section suggests that *existing explanations do not, on their own, fully account for the empirical picture* observed in Section 2. At the same time, the heterogeneous toy quadratic example of Zhang et al. (2024) provides particularly useful intuition, which we build on in Section 5.

Our experiments in this section use a setup similar to that in Figure 1, with a sequence length of 512. All methods use momentum even when not specified. We use a shorter sequence length than in the previous sections. However, we find that sequence length on its own is not crucial, and that what matters is the number of tokens per iteration. In other words, the performance differences can be attributed to the effective batch size (in tokens) rather than sequence length alone.

3.1 Heavy-tailed class imbalance

Prior work by Kunstner et al. (2024) 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, which contains the 10% least frequent tokens, to the last group, which contains the 10% most frequent ones.

We apply this analysis to the setting from Figure 1, comparing batch sizes 64 and 1024, where SGD performs drastically differently, using runs with the optimal combination of momentum parameter and learning rate for each case. We find class imbalance in both cases: the relative frequency group statistics are similar, as shown in Figure 8. However, this does not appear to cause problems for small-batch SGD, suggesting that *class imbalance alone does not imply an Adam-SGD gap across regimes.*

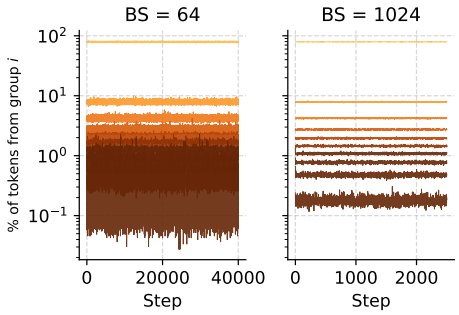


Figure 8: Batch token distribution for batch sizes 64 and 1024. Lighter colors show less frequent tokens. As expected, statistics at lower batch sizes are noisier but of similar magnitude.

We further compute perplexity separately for each frequency group and report it over training. From Figure 9 (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 pronounced for SGD in the large-batch setting than in the small-batch setting, whereas the opposite holds for Adam.

Comparing Adam and SGD across frequency groups in Figure 9, we observe that in the large-batch setting, SGD underperforms Adam across all groups, as shown in Figure 9 (b). However, the gap is larger for less frequent tokens, which aligns with findings from Kunstner et al. (2024), suggesting that rare tokens could be more challenging for SGD in imbalanced settings. In contrast, this effect is not observed in the small-batch setting, as is also clear from the results in Section 2. We would expect this SGD problem to hold, independently of batch size, but in settings where SGD works well, the issue disappears.

[2]Note, though, that this would still be drastically slower as a function of token budget, as shown in Subsection 2.2.

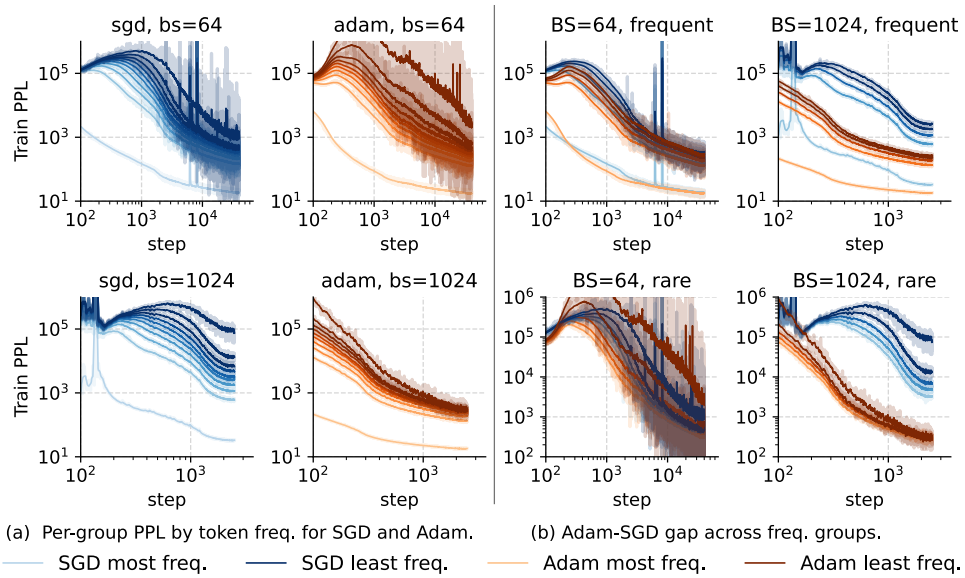


Figure 9: **(Left)** Perplexity during training for Adam and SGD in small- and large-batch settings, computed per frequency group. SGD shows a larger gap between groups in the large-batch setting, whereas the opposite holds for Adam. **(Right)** Comparison of the Adam–SGD gap across frequency groups, with more frequent groups shown on the top row and less frequent ones on the bottom. SGD underperforms Adam across all groups in large batches, with a larger gap on rare tokens. This effect is absent in the small-batch setting.

3.2 Directional sharpness

Pan & Li (2023) introduce directional sharpness to explain the optimizer gap using a Taylor expansion for the loss decrease. 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 correlation with the gradient and low directional sharpness. Let f be the loss to optimize and x_k denote the model parameters at iteration k, then

$$f(x_{k+1}) = f(x_k) + \underbrace{\nabla f(x_k)^\top (x_{k+1} - x_k)}_{\text{gradient correlation}} + \frac{1}{2} \underbrace{(x_{k+1} - x_k)^\top \nabla^2 f(x_k)(x_{k+1} - x_k)}_{\text{directional sharpness}} + O(\eta^3) \tag{1}$$

In Figure 10, we visualize gradient correlation, directional sharpness, and their sum — a second-order approximation of loss change, which indicates 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 diagnose SGD's success or failure, they do not by themselves explain why the diagnosis changes with batch size, nor why SGD performs well in small-batch regimes.

3.3 Hessian heterogeneity

From the line of work focusing on the architectural properties of Transformers, Zhang et al. (2024) 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 structure at initialization, it is possible to predict whether SGD will perform well. We revisit the simplified quadratic setting from their work and extend it by including batch size variation. We compare optimization on problems with homogeneous (CNN-like) and heterogeneous (Transformer-like) Hessians, where both share the same eigenvalue spectrum. We train with SGD+m and Adam using a cosine learning rate schedule and no clipping.

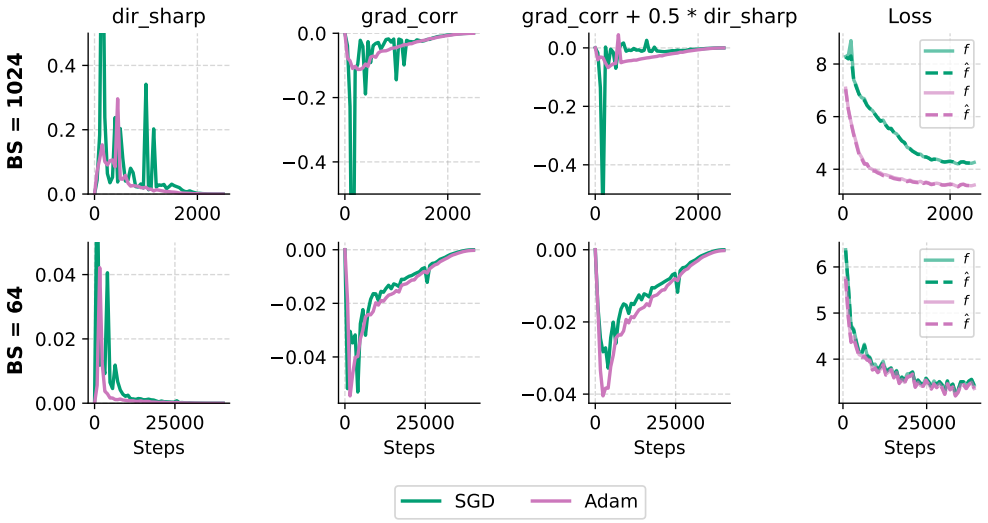


Figure 10: Directional sharpness, gradient correlation, their sum (the second-order approximation to the loss change) and the loss during training. Irrespective of the optimizer, good training trajectories have strong negative gradient correlation and low directional sharpness.

We observe the following in Figure 11. Details of the setting are provided in Appendix B.

- Across batch sizes, the largest Adam-SGD gap is observed in the heterogeneous setting — this is consistent with Zhang et al. (2024). As noted by Kunstner et al. (2023), a similar pattern can be observed for signed momentum (Bernstein et al., 2018). We build on this in Section 5.
- Both Adam and SignSGD+m benefit from increasing batch size, *in both homogeneous and heterogeneous Hessian problems*. SGD+m does not seem to profit from an increased batch size.

This pattern is consistent with our discussion in Subsection 2.6: early progress in SGD performance seems driven by number of iterations, while batch size accelerates Adam and SignSGD. We explore this further in Section 5, connecting back to our previous findings in Section 2.

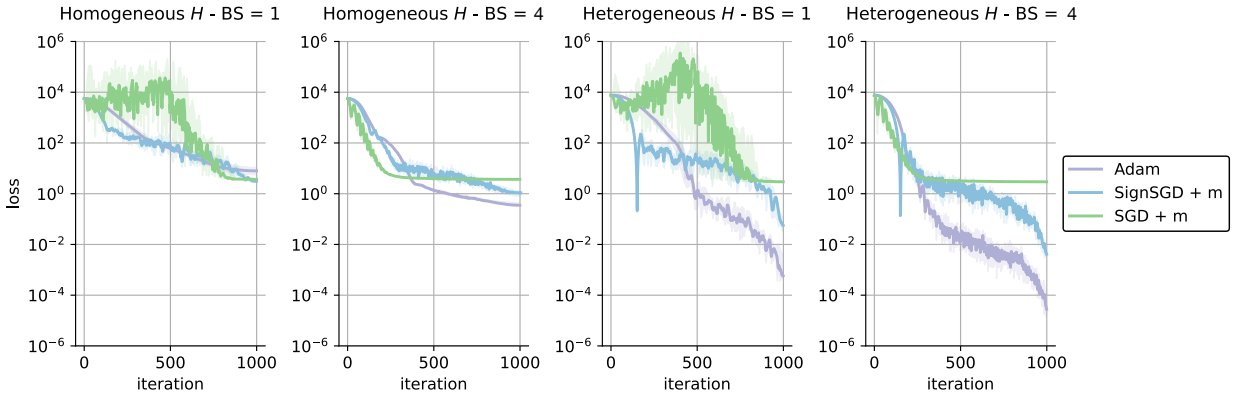


Figure 11: The gap between Adam and SGD+m relative to batch size also appears when studying only noisy quadratic models in Zhang et al. (2024). The heterogeneous setting is inspired by the Transformer structure, while the homogeneous setting is inspired by the CNN structure. See Appendix B for details. We note that Adam's *advantage with larger batch sizes is also evident in the homogeneous setting*, but much more pronounced in the heterogeneous setting. Learning rates are tuned to give similar performance at homogeneous batch size 1. We show mean and two-standard-deviation bands for 10 runs.

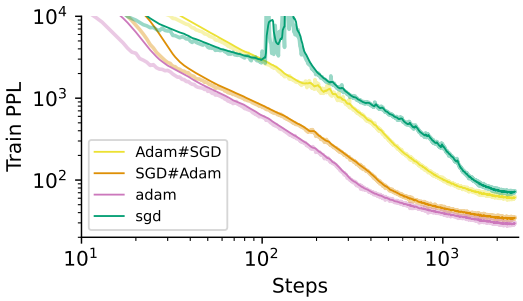


Figure 13: Grafting in large-batch training: using Adam's direction leads to results closer to Adam, while SGD direction leads to results closer to SGD.

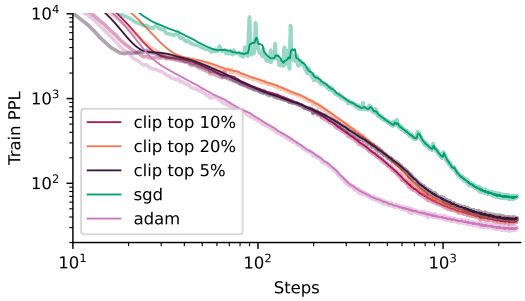


Figure 14: Adaptive clipping with different percentages of clipped coordinates in large-batch training. It improves SGD but still does not fully match Adam.

4 Further ablations: can we fix SGD?

In Section 3, we saw that while prior work sheds light on the Adam-SGD gap in the large-batch regime, it remains unclear how batch size itself factors into these explanations. However, ablations inspired by discussions in the literature can give us further insights. In Subsection 4.1 and Subsection 4.2, we ask: *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 — is the problem its direction or its magnitude? We focus on the setting from Section 3, using a batch size of 1024 and the optimal combination of momentum parameter and learning rate.

4.1 Insights from Grafting

To isolate the role of update direction and magnitude, we use the grafting technique proposed by Agarwal et al. (2020), which applies the update direction of one optimizer with the magnitude of another. We train the model in the large-batch setting, using two combinations: 1) SGD magnitude with Adam direction (SGD#Adam), and 2) Adam magnitude with SGD direction (Adam#SGD). We use a well-performing momentum 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 13. 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.

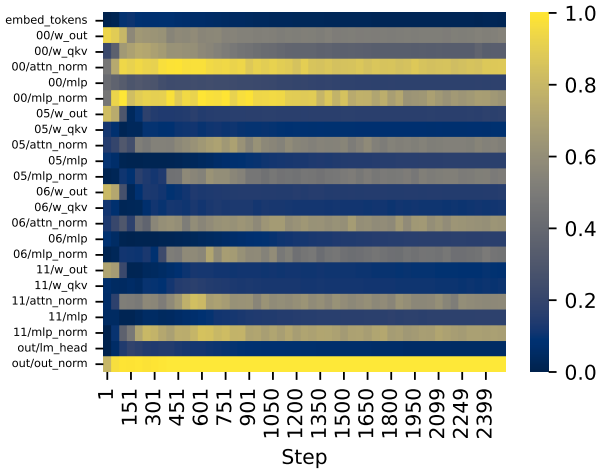


Figure 12: Fraction of clipped momentum coordinates per layer during training, using $p = 10\%$ adaptive clipping. Only a subset of blocks is shown for clarity, as similar patterns are observed across all blocks. Clipping is present across all parameters, but most pronounced in normalization layers.

4.2 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 (Subsection 2.2). 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, motivated by Pan & Li (2023). As shown in Algorithm 1 (Appendix), we clip the top $p\%$ of the largest momentum coordinates at each step. We test several values of p (5%, 10%, and 20%). For each value, we keep the optimal β 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 14. This suggests that a subset of larger update coordinates consistently contributes to poor update directions and slows down SGD in large-batch training. In contrast, small-batch training does not appear to produce the same problematic coordinates.

Further, we ask whether certain groups of parameters are more likely to produce problematic coordinates. To explore this, we inspect which layers the clipped momentum coordinates come from, using the best-performing setting with $p = 10\%$. In Figure 12, we show the fraction of parameters within each layer that are clipped, relative to the total number of parameters in that layer. We find that normalization layers are clipped the most, which aligns with findings from Zhao et al. (2024) and Tomihari & Sato (2025). However, this does not imply that only normalization layers are problematic. As we observe significant clipping across other layers as well, this suggests that large coordinates persist across all parameters, though *they are most pronounced in normalization layers and the output head*.

5 Theoretical Analysis

The experiments in Section 2 suggest that the Adam-SGD gap is strongly mediated by batch size. In fixed-token-budget training, SGD+m is competitive with Adam at small batch sizes but deteriorates at large batch sizes, while SignSGD+m remains much closer to Adam (Figure 1, Figure 2). Longer step-budget sweeps show that large-batch SGD is not fundamentally unable to optimize; rather, it is more step-limited than Adam and sign-like methods (Figure 4). This pattern persists at larger scales (Figure 5) and becomes especially clear in the SGD/SignSGD comparison: increasing the batch size strongly accelerates SignSGD, but only mildly helps SGD when measured by optimizer steps (Figure 7).

The ablations in Section 3, Subsection 4.1, and Subsection 4.2 further indicate that prior explanations such as class imbalance, directional sharpness, and Hessian heterogeneity are informative but do not by themselves explain this optimizer-specific batch-size sensitivity. We therefore study SignSGD not as a complete model of Adam, but as the simplest algorithm in which the relevant batch-size mechanism appears. The key mechanism is that the batch size changes the *mean update direction* of signed methods: unlike SGD, whose drift is independent of B up to variance reduction, SignSGD has a batch-dependent drift through the signal-to-noise ratio. We make this intuition precise first through the SDE approximation in Theorem 1, and then through the discrete decomposition and bounds in Proposition 2, Proposition 3, and Corollary 1.

Notation. We consider minimizing a potentially non-convex stochastic objective in the form

$$\min_{x \in \mathbb{R}^d} f(x) := \mathbb{E}_{\xi \sim \mathcal{D}} \left[\ell(x; \xi) \right], \qquad \ell(\cdot; \xi) : \mathbb{R}^d \to \mathbb{R}, \tag{2}$$

where ξ is a random variable sampled from a probability distribution $\mathcal{D}$. We assume access to a stochastic gradient estimator $g(x; \xi)$ satisfying

$$\mathbb{E}_{\xi \sim \mathcal{D}} \left[g(x; \xi) \right] = \nabla f(x), \qquad \forall x \in \mathbb{R}^d. \tag{3}$$

Consider iterative methods such as SGD and SignSGD with batch size $B \in \mathbb{N}$. At iteration k, conditionally on the current iterate x_k, we sample $\xi_{k,1}, \ldots, \xi_{k,B} \overset{\text{i.i.d.}}{\sim} \mathcal{D}$ and define the gradient estimator

$$g_k := \frac{1}{B} \sum_{b=1}^{B} g(x_k; \xi_{k,b}). \tag{4}$$

Then g_k is conditionally unbiased: $\mathbb{E}\left[g_k | x_k \right] = \nabla f(x_k)$. Let $\| \cdot \|_2$ denote the standard Euclidean norm.

5.1 Intuition from SDE models

Let X_t denote the continuous-time parameter process and let Σ denote the batch-size-one gradient-noise covariance[3]. The standard SDE approximation of SGD reads (Mil'shtein, 1986; Liu et al., 2021)

$$dX_t = -\nabla f(X_t)dt + \sqrt{\frac{\eta}{B}}\,\Sigma^{1/2}dW_t. \tag{5}$$

Thus, for SGD, the batch size enters the diffusion term but not the drift. Signed updates have a different structure: the mean signed update is a nonlinear function of the gradient-to-noise ratio, and therefore of B.

Theorem 1 (Compagnoni et al. (2025))**.** *Assume that the mini-batch gradient noise has independent Gaussian coordinates with covariance Σ/B, where $\Sigma = \mathrm{Diag}(\sigma_1^2, \ldots, \sigma_d^2)$ is positive definite. Then the following SDE provides an order-1 weak approximation (Mil'shtein, 1986) of SignSGD:*

$$dX_t = -\operatorname{erf}\left(\sqrt{\frac{B}{2}}\,\Sigma^{-\frac{1}{2}}\nabla f(X_t)\right)dt + \sqrt{\eta}\sqrt{I_d - \mathrm{Diag}\left(\operatorname{erf}\left(\frac{\sqrt{B}\,\Sigma^{-\frac{1}{2}}\nabla f(X_t)}{\sqrt{2}}\right)\right)^2}\,dW_t, \tag{6}$$

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

Proof intuition. For one coordinate, write $m_i(x) \sim \mathcal{N}(\mu_i, \sigma_i^2/B)$, where $\mu_i = \partial_i f(x)$. Then, with Φ the standard Gaussian CDF,

$$\mathbb{E}[\mathrm{sign}(m_i(x))] = \mathbb{P}[m_i(x) \geq 0] - \mathbb{P}[m_i(x) < 0] = 2\Phi\left(\frac{\sqrt{B}\,\mu_i}{\sigma_i}\right) - 1 = \operatorname{erf}\left(\sqrt{\frac{B}{2}}\frac{\mu_i}{\sigma_i}\right).$$

SignSGD moves in the negative of this mean direction, giving the drift term in Theorem 1; the conditional variance of the sign gives the diffusion coefficient. □

Takeaway. Near the origin, $\operatorname{erf}(z) = \frac{2}{\sqrt{\pi}}z + O(z^3)$, while for large $|z|$ it saturates at ± 1. Hence, in the small signal-to-noise regime, the signed drift is approximately $-\sqrt{\frac{2B}{\pi}}\,\Sigma^{-\frac{1}{2}}\nabla f(X_t)dt$, so increasing B increases the effective drift magnitude like $\sqrt{B}$, until the signs become essentially noise-free and the gain saturates. In contrast, the SGD drift remains $-\nabla f(X_t)dt$ for all B, with batch size only reducing the diffusion. This analysis provides a mechanism consistent with 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 (see Figure 7 and Figure 11).

This is the SDE-level motivation for the discrete mean–variance decomposition presented in Subsection 5.2, where the same mechanism appears through the centered CDF map $\mathsf{G}(\sigma_B^{-1}\nabla f)$.

5.2 A nonconvex analysis

The Gaussian assumption is not essential for the drift intuition behind Theorem 1. As discussed by Compagnoni et al. (2025) and clear from the proof sketch — studying bounds on the cumulative distribution — a similar expression can hold even for distributions with heavier tails, such as the Student-t (see Corollary C.10 in Compagnoni et al. (2025)). In this last subsection, we build on the mean-variance decomposition of SignSGD inspired by the continuous-time analysis, and provide convergence rates for this algorithm under some special noise assumptions.

Assumption 1 (Objective)**.** *The function $f : \mathbb{R}^d \to \mathbb{R}$ is differentiable, L-smooth, and bounded from below. Namely, $\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2$ for all $x, y \in \mathbb{R}^d$ and $f^* := \inf_x f(x) > -\infty$.*

All proofs for this section are deferred to Appendix C.

[3]For this intuition, we treat this covariance as well-defined and constant.

5.2.1 Short recap on SGD

The standard convergence guarantee below assumes bounded variance (Zhang et al., 2020a).

Assumption 2 (Bounded variance)**.** *There exists $\sigma^2 \geq 0$ such that, for every $x \in \mathbb{R}^d$ and every $u \in \mathbb{R}^d$,*

$$\mathbb{E}_{\xi\sim\mathcal{D}}\left[\langle g(x;\xi) - \nabla f(x), u\rangle^2\right] \leq \sigma^2\|u\|_2^2. \tag{7}$$

Under this assumption the mini-batch estimator can be written as $g_k = \nabla f(x_k) + \sigma_B\zeta_k$, $\sigma_B^2 := \frac{\sigma^2}{B}$, where $\mathbb{E}\left[\zeta_k|x_k\right] = 0$ and $\mathbb{E}\left[\|\zeta_k\|_2^2|x_k\right] \leq d$.

We provide below the standard convergence rate for SGD in our standard setup (Ghadimi & Lan, 2013).

Proposition 1 (SGD convergence)**.** *Let $(x_k)_k$ be the iterates of SGD with batch size B. Suppose Assumption 1 and Assumption 2 hold. Let $\Delta_0 = f(x_0) - f^*$, if $\eta \leq 1/L$, then for any $K \geq 1$,*

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\big[\|\nabla f(x_k)\|_2^2\big] \leq \frac{2\Delta_0}{\eta K} + \frac{dL\sigma^2\eta}{B}. \tag{8}$$

5.2.2 A simple rate for SignSGD without momentum, under regular noise

For the SignSGD analysis below, we use the following direct noise model for the stochastic gradient input. This assumption replaces the conditional unbiasedness and bounded-variance requirements used in subsubsection 5.2.1. In particular, we do not require the noise variables to have a finite first or second moment, since the analysis only uses bounded functions of the gradient input, namely $\mathrm{sign}(g_k)$. The assumption below resembles the assumptions in Theorem 1 and allows us to model SignSGD in a "smooth" way.

Assumption 3 (Diagonal well-behaved noise)**.** *Fix $\sigma > 0$ and define $\sigma_B := \sigma/\sqrt{B}$. For each iteration $k < K$, define the normalized gradient noise*

$$\zeta_k := \sigma_B^{-1}\big(g_k - \nabla f(x_k)\big), \qquad \zeta_k = (\zeta_{k,1}, \ldots, \zeta_{k,d}), \tag{9}$$

so that $g_k = \nabla f(x_k) + \sigma_B\zeta_k$. Conditionally on the iterate x_k, we assume the coordinates $\zeta_{k,1}, \ldots, \zeta_{k,d}$ are independent and identically distributed, absolutely continuous with respect to the Lebesgue measure on $\mathbb{R}$, and with density p_ζ symmetric around zero. We further assume that for every $D > 0$ there exist $0 < m \leq M < \infty$ such that $m \leq \mathsf{p}_\zeta(x) \leq M$ for all $x \in [-D, D]$[4]. This SignSGD noise model replaces the conditional unbiasedness and bounded-variance assumptions used above; in particular, we do not require the variables $\zeta_{k,i}$ to have a finite first or second moment. In this context, we define the noise cumulative distribution function

$$\mathsf{F}(x) := \Pr[\zeta_{k,i} \leq x]. \tag{10}$$

Note. Some aspects of the assumption above are slightly stronger than needed for the proofs. However, our objective here is clean intuition rather than maximal generality.

Assumption 3 is a direct model of the mini-batch gradient input. It allows ζ to lack moments, but it still assumes the mini-batch noise scale is $\sigma/\sqrt{B}$. Deriving this scaling from raw samples requires additional assumptions. The next proposition, proved in Appendix C, characterizes the update distribution.

Proposition 2. *Let x_k denote iterate after k steps. Under Assumption 3, for all $k < K$,*

$$\mathbb{E}\left[sign\left(\nabla f(x_k) + \sigma_B\zeta_k\right)|x_k\right] = 2\mathsf{F}\left(\sigma_B^{-1}\nabla f(x_k)\right) - \mathbf{1}, \tag{11}$$

$$Var\left[sign\left(\nabla f(x_k) + \sigma_B\zeta_k\right)|x_k\right] = 4\mathsf{F}\left(\sigma_B^{-1}\nabla f(x_k)\right)\left(\mathbf{1} - \mathsf{F}\left(\sigma_B^{-1}\nabla f(x_k)\right)\right) \leq \mathbf{1}, \tag{12}$$

where $\sigma_B := \sigma/\sqrt{B}$ and F, sign, and Var are applied coordinatewise. Equivalently, there exists a random vector ε_k with $\mathbb{E}[\varepsilon_k|x_k] = 0$ and $\mathbb{E}[\|\varepsilon_k\|_2^2|x_k] \leq d$ such that one can write

$$sign(\nabla f(x_k) + \sigma_B\zeta_k) = \mathsf{G}(\sigma_B^{-1}\nabla f(x_k)) + \varepsilon_k, \qquad \mathsf{G}(x) := 2\mathsf{F}(x) - 1. \tag{13}$$

[4]This rules out pathological cases, see the counterexample for SignSGD convergence in (Safaryan & Richtárik, 2021), where p_ζ is the sum of two Dirac delta functions. In particular, this rules out the vanishing noise condition $\sigma = 0$.

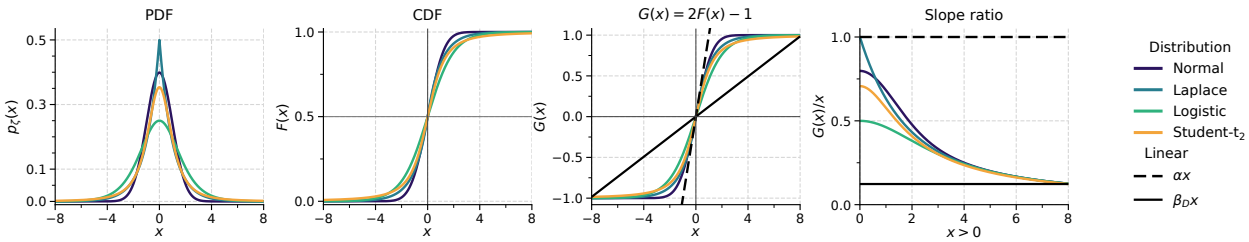


Figure 15: Illustration of key quantities in our analysis. Starting from the leftmost panel: we plot, in the context of Assumption 3, the coordinatewise noise distribution p_ζ. We assume it is symmetric around zero, but it can deviate substantially from a Gaussian, including distributions without second moment, pathological for SGD (Zhang et al., 2020a). The second panel shows the cumulative distribution function F of p_ζ. We then plot G, which is crucial in our SignSGD model (Equation 13). In the same panel, we plot the sandwich bound $\beta_D|x| \leq |\mathsf{G}| \leq \alpha|x|$ for $D = 8$, derived in Lemma 1. Finally, in the rightmost panel we plot $|\mathsf{G}(x)|/|x|$, illustrating the definition in Equation 15.

To carry out the proof, we need to bound G linearly. To do that, regularity of p_ζ in Assumption 3 is crucial.

Lemma 1. *Let Assumption 3 hold and recall that $\mathsf{G}(x) := 2\mathsf{F}(x) - 1$. Then, for all $x \in \mathbb{R}$,*

$$x\mathsf{G}(x) \geq 0. \tag{14}$$

Moreover, for every $D > 0$, define the sharp lower-slope constant

$$\beta_D := \inf_{0<|x|\leq D} \frac{|\mathsf{G}(x)|}{|x|} > 0. \tag{15}$$

Then β_D is strictly positive, and for all $x \in [-D, D]$, $\beta_D|x| \leq |\mathsf{G}(x)|$. Finally, there exists a constant $\alpha > 0$, independent of D, such that for all $x \in \mathbb{R}$, $|\mathsf{G}(x)| \leq \alpha|x|$.

For an illustration of this property, refer to Figure 15.

Proposition 3. *Let $(x_k)_k$ be the iterates of SignSGD with update $x_{k+1} = x_k - \eta sign(g_k)$ and batch size B. Fix an integer K and suppose Assumption 1 and Assumption 3 hold. Pick $D > 0$ so that, almost surely, for all $k < K$, $\|\nabla f(x_k)\|_\infty \leq \frac{D\sigma}{\sqrt{B}}$. Recall that $\mathsf{G}(x) := 2\mathsf{F}(x) - 1$ and let $\beta_D := \inf_{0<|u|\leq D} \frac{|\mathsf{G}(u)|}{|u|} > 0$ be the lower-slope constant from Lemma 1. Let $\Delta_0 = f(x_0) - f^*$, for any stepsize $\eta > 0$,*

$$\frac{1}{K}\sum_{k=0}^{K-1} \mathbb{E}[\|\nabla f(x_k)\|_2^2] \leq \frac{\sigma\Delta_0}{\beta_D\eta\sqrt{B}\,K} + \frac{dL\sigma\eta}{2\beta_D\sqrt{B}}. \tag{16}$$

The rate contains the factor $O(B^{-1/2})$ on the first term – in stark contrast to SGD (see Proposition 1). However, it also depends on β_D, and D is tied to the batch size-dependent quantity $\sqrt{B}\|\nabla f(x_k)\|_\infty/\sigma$. This implicit dependency requires further investigation. Indeed, if one instead fixes an absolute gradient bound $\|\nabla f(x_k)\|_\infty \leq R$, then $D = D_B = R\sqrt{B}/\sigma$ and one can clearly see that β_D also depends on B. It is then a valid concern that the $B^{-1/2}$ gain over SGD is canceled by the deterioration of β_D. This saturation indeed occurs: acceleration from batch size eventually plateaus, as shown in Figure 11.

Corollary 1 (Batch-size dependence under an absolute gradient bound)**.** *Assume the setting of Proposition 3. Assume that, almost surely, for all $k < K$, $\|\nabla f(x_k)\|_\infty \leq R$ for some $R > 0$. Assume $t \mapsto \mathsf{p}_\zeta(t)$ is non-increasing on $(0, \infty)$, then $t \mapsto \mathsf{G}(t)/t$ is non-increasing[5] on $(0, \infty)$. The bound of Proposition 3 becomes*

$$\frac{1}{K}\sum_{k=0}^{K-1} \mathbb{E}\big[\|\nabla f(x_k)\|_2^2\big] \leq \frac{R\,\Delta_0}{\mathsf{G}(R\sqrt{B}/\sigma)\,\eta K} + \frac{dL\eta R}{2\,\mathsf{G}(R\sqrt{B}/\sigma)}. \tag{17}$$

In particular, the iteration-dependent term accelerates with batch size, until G saturates (Figure 15).

[5] $\frac{|\mathsf{G}(t)|}{|t|} = \frac{2}{t}\int_0^t \mathsf{p}_\zeta(s)\,ds$. This is twice the average of the non-increasing density p_ζ on $[0, t]$, hence it is non-increasing in t.

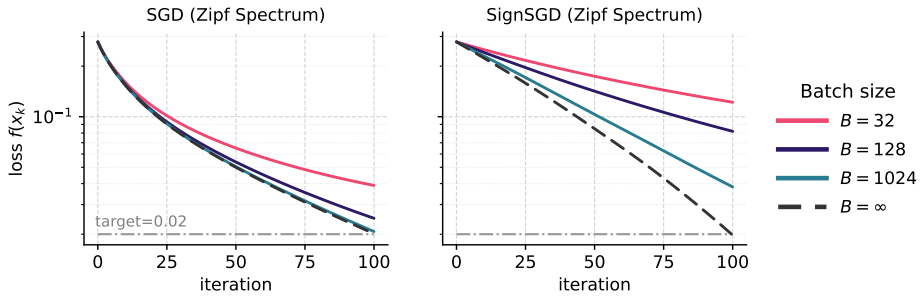


Figure 16: Batch-size sweep for SGD and SignSGD on a stochastic diagonal quadratic $f(x) = \frac{1}{2}x^\top Hx$ in dimension $d = 100$, with Zipf spectrum $\lambda_i \propto i^{-1}$, resembling language data (Kunstner & Bach, 2026). We initialize $x_0 = 0.1z$, $z \sim \mathcal{N}(0, I_d)$, and use stochastic gradients $g_k = Hx_k + B^{-1/2}\zeta_k$, $\zeta_k \sim \mathcal{N}(0, I_d)$. We plot the mean trajectory for three realizations of the noise sequence. Increasing B only mildly changes SGD, but strongly accelerates SignSGD toward its full-batch trajectory, matching the Transformer behavior in Figure 7 and the drift-saturation intuition of Corollary 1.

Proof. If $\|\nabla f(x_k)\|_\infty \le R$ for some $R > 0$, then $D = \frac{R\sqrt{B}}{\sigma}$. Since $t \mapsto \mathsf{G}(t)/t$ is non-increasing on $(0, \infty)$, $\beta_D = \inf_{0<t\le D} \mathsf{G}(t)/t = \frac{\sigma\, \mathsf{G}(R\sqrt{B}/\sigma)}{R\sqrt{B}}$. □

Summary of the mechanism. Figure 16 summarizes the mechanism studied in this section. For SGD, increasing the batch size mainly reduces variance; the leading drift is unchanged, as in Equation 5 and Proposition 1. For SignSGD, the expected update is instead $\mathsf{G}(\sigma_B^{-1}\nabla f(x_k))$ by Proposition 2, so increasing B increases the signal-to-noise ratio and can amplify progress per step until the map G saturates. This explains why large batches can be inefficient for SGD at a fixed token budget, while Adam-like and sign-like methods can better convert reduced gradient noise into faster per-step progress. Thus, Figure 16 links the empirical trends in Figure 1, Figure 4, Figure 6, and Figure 7 to the theoretical analysis developed above.

Relation to prior theory and batch-size effects. The difficulty in analyzing signed and normalized stochastic methods is that this nonlinearity turns an unbiased gradient estimate into a biased update direction, leading to counterexamples for naive sign methods and motivating stabilization mechanisms such as momentum or error feedback (Karimireddy et al., 2019; Cutkosky & Mehta, 2020; Safaryan & Richtárik, 2021). The proof using momentum as a variance-reduction method (Cutkosky & Mehta, 2020) naturally extends to LMO methods (Kovalev, 2025; Pethick et al., 2025) and optimizers such as Muon (Jordan et al., 2024). SignSGD without momentum, as studied in this section, is nevertheless known to converge under special assumptions, such as success-probability conditions (Safaryan & Richtárik, 2021) and unimodal symmetric noise (Bernstein et al., 2018). Several works have also noted that SignSGD exhibits different batch-size sensitivity than plain SGD when computing scaling laws or when studying learning rate adaptation (Li et al., 2024; Shulgin et al., 2026). Our contribution is complementary: under regular symmetric coordinatewise additive noise, resembling the unimodality assumption in Bernstein et al. (2018), we compute the deterministic SignSGD drift exactly as $G(\sigma_B^{-1}\nabla f(x_k))$, with $G = 2F - 1$, and use it in a finite-time nonconvex no-momentum proof. This procedure, inspired by the continuous-time analysis in Compagnoni et al. (2025), *is not aimed at deriving stronger rates; its purpose is to make explicit that batch size changes the drift itself*, yielding $\sqrt{B}$ amplification at low signal-to-noise ratios and saturation once signs become nearly deterministic. Meterez et al. (2026) support our central claim that normalized or signed methods convert reduced gradient noise into greater per-step progress as batch size grows, although their analysis only covers normalized SGD in noisy linear regression.

6 Conclusion and Limitations

This work revisits the widely observed gap between Adam and SGD in Transformer language-model pretraining. Across over 1000 training runs spanning batch sizes, token budgets, datasets, model scales, momentum choices, clipping schemes, and optimizer variants, we find that the Adam–SGD gap is not monolithic: it is

strongly shaped by batch size. At commonly used batch sizes, SGD with momentum is substantially less efficient than Adam at the same token budget. However, when the batch size is reduced and hyperparameters are carefully tuned, SGD can match Adam in our academic-scale pretraining regimes, including at 410M and 1B parameters. Further experiments show that SignSGD, which can be viewed as a simple model of the Adam update, behaves similarly to Adam as batch size increases. Our theory suggests that this sharp difference between SGD and SignSGD arises because batch size changes the expected update direction in SignSGD, whereas for SGD it primarily changes only the amount of noise in the update. This provides a simple explanation for why Adam-like methods can benefit more from large batches than SGD.

We believe this perspective can help build a clearer theory of optimizer behavior in language modeling. More broadly, understanding how batch size changes the update itself may guide the design of new optimizers. Our analysis is only a first step: it relies on regularity and diagonality assumptions on the gradient noise, and is mainly meant to provide intuition rather than a complete theory. Future work can establish if the same mechanism appears in optimizers such as Muon (Jordan et al., 2024), or attempt a more direct analysis of Adam under relaxed assumptions.

References

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

Kwangjun Ahn, Xiang Cheng, Minhak Song, Chulhee Yun, Ali Jadbabaie, and Suvrit Sra. Linear attention is (maybe) all you need (to understand transformer optimization). In *International Conference on Learning Representations*, 2024.

Niccolò Ajroldi. plainlm: Language model pretraining in pytorch. https://github.com/Niccolo-Ajroldi/plainLM, 2024.

Lukas Balles and Philipp Hennig. Dissecting adam: The sign, magnitude and variance of stochastic gradients. In *International Conference on Machine Learning*, pp. 404–413. PMLR, 2018.

Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Anima Anandkumar. signSGD: Compressed Optimisation for Non-Convex Problems, August 2018. URL http://arxiv.org/abs/1802.04434. arXiv:1802.04434 [cs, math].

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.

Sidney Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, et al. Gpt-neox-20b: An open-source autoregressive language model. In *Proceedings of BigScience Episode# 5–Workshop on Challenges & Perspectives in Creating Large Language Models*, pp. 95–136, 2022.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

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.

Ashok Cutkosky and Harsh Mehta. Momentum improves normalized sgd. In *International conference on machine learning*, pp. 2260–2268. PMLR, 2020.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35, 2022.

Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. Attention is not all you need: Pure attention loses rank doubly exponentially with depth. In *International conference on machine learning*, pp. 2793–2803. PMLR, 2021.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Guillaume Garrigos and Robert M Gower. Handbook of convergence theorems for (stochastic) gradient methods. *arXiv preprint arXiv:2301.11235*, 2023.

Saeed Ghadimi and Guanghui Lan. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM journal on optimization*, 23(4):2341–2368, 2013.

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.

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.

Alexander Hägele, Alejandro Hernández-Cano, Atli Kosson, and Martin Jaggi. Improving neural network training by decoupling the magnitude and direction of weight vectors. *arXiv preprint arXiv:2606.25971*, 2026.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training Compute-Optimal Large Language Models, March 2022. URL http://arxiv.org/abs/2203.15556. arXiv:2203.15556 [cs].

Florian Hübler, Kai Lion, Antonio Orvieto, and Niao He. Muown implicitly performs angular step-size decay. *arXiv preprint arXiv:2606.23637*, 2026.

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].

Rustem Islamov, Roman Machacek, Aurelien Lucchi, Antonio Silveti-Falls, Eduard Gorbunov, and Volkan Cevher. On the role of batch size in stochastic conditional gradient methods. *arXiv preprint arXiv:2603.21191*, 2026.

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].

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/.

Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Stich, and Martin Jaggi. Error feedback fixes signsgd and other gradient compression schemes. In *International conference on machine learning*, pp. 3252–3261. PMLR, 2019.

Andrej Karpathy. Nanogpt, 2022.

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

Dmitry Kovalev. Understanding gradient orthogonalization for deep learning via non-euclidean trust-region optimization. *arXiv preprint arXiv:2503.12645*, 2025.

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

Frederik Kunstner and Francis Bach. Scaling laws for gradient descent and sign descent for linear bigram models under zipf's law. *Advances in Neural Information Processing Systems*, 38:144435–144467, 2026.

Frederik Kunstner, Philipp Hennig, and Lukas Balles. Limitations of the empirical fisher approximation for natural gradient descent. *Advances in neural information processing systems*, 32, 2019.

Frederik Kunstner, Jacques Chen, Jonathan 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. In *ICLR*, 2023.

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].

Houyi Li, Wenzhen Zheng, Qiufeng Wang, Hanshan Zhang, Zili Wang, Shijie Xuyang, Yuantao Fan, Zhenyu Ding, Haoying Wang, Ning Ding, et al. Predictable scale: Part i, step law–optimal hyperparameter scaling law in large language model pretraining. *arXiv preprint arXiv:2503.04715*, 2025.

Shuaipeng Li, Penghao Zhao, Hailin Zhang, Xingwu Sun, Hao Wu, Dian Jiao, Weiyan Wang, Chengjun Liu, Zheng Fang, Jinbao Xue, Yangyu Tao, Bin Cui, and Di Wang. Surge Phenomenon in Optimal Learning Rate and Batch Size Scaling, June 2024. URL http://arxiv.org/abs/2405.14578. arXiv:2405.14578 [cs].

Kai Lion, Florian Hübler, Bingcong Li, Antonio Orvieto, and Niao He. Muown: Row-norm control for muon optimization. *arXiv preprint arXiv:2605.10797*, 2026.

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.

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.

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

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

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].

Martin Marek, Sanae Lotfi, Aditya Somasundaram, Andrew Gordon Wilson, and Micah Goldblum. Small batch size training for language models: When vanilla sgd works, and why gradient accumulation is wasteful. *arXiv preprint arXiv:2507.07101*, 2025.

Alexandru Meterez, Depen Morwani, Jingfeng Wu, Costin-Andrei Oncescu, Cengiz Pehlevan, and Sham M Kakade. Seesaw: Accelerating training by balancing batch size and learning rate scheduling. In *The Fourteenth International Conference on Learning Representations*, 2026.

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

Bruno Mlodozeniec, Pierre Ablin, Louis Béthune, Dan Busbridge, Michal Klein, Jason Ramapuram, and Marco Cuturi. Completed hyperparameter transfer across modules, width, depth, batch and duration. *arXiv preprint arXiv:2512.22382*, 2025.

Toan Q Nguyen and Julian Salazar. Transformers without tears: Improving the normalization of self-attention. *arXiv preprint arXiv:1910.05895*, 2019.

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.

Antonio Orvieto and Robert Gower. In search of adam's secret sauce. *Advances in Neural Information Processing Systems*, 38:63404–63442, 2025.

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].

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849, 2024.

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.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Mher Safaryan and Peter Richtárik. Stochastic sign descent methods: New algorithms and better theory. In *International Conference on Machine Learning*, pp. 9224–9234. PMLR, 2021.

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.

Christopher J Shallue, Jaehoon Lee, Joseph Antognini, Jascha Sohl-Dickstein, Roy Frostig, and George E Dahl. Measuring the effects of data parallelism on neural network training. *Journal of Machine Learning Research*, 20(112):1–49, 2019.

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Egor Shulgin, Dimitri von Rütte, Tianyue H Zhang, Niccolò Ajroldi, Bernhard Schölkopf, and Antonio Orvieto. Deriving hyperparameter scaling laws via modern optimization theory. *arXiv preprint arXiv:2603.15958*, 2026.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama. https://www.cerebras.net/blog/slimpajama-a-627b-token-cleaned-and-deduplicated-version-of-redpajama, 2023. URL https://huggingface.co/datasets/cerebras/SlimPajama-627B.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

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].

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

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.

Dimitri von Rütte, Janis Fluri, Omead Pooladzandi, Bernhard Schölkopf, Thomas Hofmann, and Antonio Orvieto. Scaling behavior of discrete diffusion language models. *arXiv preprint arXiv:2512.10858*, 2025.

Ben Wang and Aran Komatsuzaki. GPT-J-6B: A 6 billion parameter autoregressive language model, 2021. URL https://github.com/kingoflolz/mesh-transformer-jax.

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.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International conference on machine learning*, pp. 10524–10533. PMLR, 2020.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019.

Guodong Zhang, Lala Li, Zachary Nado, James Martens, Sushant Sachdeva, George E. Dahl, Christopher J. Shallue, and Roger Grosse. Which Algorithmic Choices Matter at Which Batch Sizes? Insights From a Noisy Quadratic Model, October 2019. URL http://arxiv.org/abs/1907.04164. arXiv:1907.04164 [cs].

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].

Jingzhao Zhang, Tianxing He, Suvrit Sra, and Ali Jadbabaie. Why gradient clipping accelerates training: A theoretical justification for adaptivity, February 2020a. arXiv:1905.11881 [cs, math].

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 2020b. URL http://arxiv.org/abs/1912.03194. arXiv:1912.03194 [cs, math].

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].

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 Experimental Details and Additional Figures

For pre-training Transformers on causal language modeling, we use a setup that builds upon the nanoGPT (Karpathy, 2022) implementation, augmenting it with Rotary Positional Embeddings (Su et al., 2024), RMSNorm (Zhang & Sennrich, 2019), and SwiGLU (Shazeer, 2020). We do not adopt QK normalization or z-loss, as those modifications are quite recent. All our models have a vocabulary size of 50280 and make use of GPT-NeoX tokenizer (Black et al., 2022). We adopt an enhanced training recipe popularized by large language models such as LLaMA (Touvron et al., 2023). These modifications include: training in bfloat16; employing a linear learning rate warm-up for 10% of the training steps (unless specified otherwise), followed by either cosine annealing or WSD (Hägele et al., 2024). Global norm clipping is used (unless specified or ablated upon) for gradients with norms above 1 (on the raw gradient, as a first step). We have no weight tying between the embedding and the last linear layer. Validation perplexity always refers to a separate subset consisting of 100M tokens.

A.1 Experimental Setup

Computational Resources. All experiments use a single NVIDIA A100-SXM4-80GB.

Code. All our runs use the repository `https://github.com/Niccolo-Ajroldi/plainLM`.

Datasets. We test our claims on both the SlimPajama (Soboleva et al., 2023) and FineWeb (Penedo et al., 2024) datasets.

Model settings (12 Layers, 160M). We use the same configuration as (Biderman et al., 2023): `https://github.com/EleutherAI/pythia/blob/main/models/160M/pythia-160m.yml`

- *Layers:* 12 Transformer (Vaswani et al., 2017) layers
- *Attention heads:* 12
- *Hidden size:* 768
- *Attention implementation:* FlashAttention (Dao et al., 2022).
- *MLP type:* SwiGLU (Shazeer, 2020) with expansion factor 8/3.
- *Backbone:* PreLN Transformer (Xiong et al., 2020) with skip connections.
- *Normalization:* RMSNorm (Zhang & Sennrich, 2019) for both Attention and MLP.
- *Position embeddings:* Rotary embeddings (RoPE) applied to 25% of dimensions (Su et al., 2024).
- *Initialization:* the MLP and Attention output weights are initialized with standard deviation $0.02/\sqrt{2\#\text{layers}}$ (with scaling, similar to (Radford et al., 2019)). All other weights (including embeddings) are initialized with a standard deviation of 0.02 (Nguyen & Salazar (2019); Wang & Komatsuzaki (2021), Sec. 2.2). Biases are always initialized at zero.
- *Precision:* bfloat16.
- *Dropout:* Disabled for both hidden and attention layers (see also Chowdhery et al. (2023)).

Model settings (250M, 24 layers). We keep it identical to the setting above, and just increase the number of layers to 24.

Model settings (410M). We use the same setting as (Biderman et al., 2023), configuration can be found here: https://github.com/EleutherAI/pythia/blob/main/models/410M/pythia-410m-deduped.yml

- *Layers:* 24 Transformer layers
- *Attention heads:* 16
- *Hidden size:* 1024
- Other settings are as in 160M model.

Model settings (1B). We use the same setting as (Biderman et al., 2023), configuration can be found here: https://github.com/EleutherAI/pythia/blob/main/models/1B/pythia-1b-deduped.yml

- *Layers:* 16 Transformer layers
- *Attention heads:* 8
- *Hidden size:* 2048
- Other settings are as in 160M model.

A.2 Supplement for Subsection 2.2

We use a 160M-parameter Transformer, as detailed in Subsection A.1, trained on 3.2B SlimPajama tokens. All runs have sequence length $L = 2048$. All methods use cosine annealing for the learning rate and learning rate warmup (10%); weight decay is omitted to avoid inflating the search space. Below, B indicates the batch size and K the number of steps.

The main hyperparameter sweep comprises 677 runs; the warmup/clipping ablation (Figure 3) adds 112 runs, for a total of 789 runs in this section. Results are shown in Figure 1 and Figure 17. The following hyperparameter grids specify the values considered, but *not every Cartesian combination was evaluated*: clearly diverging options were avoided.

- **Adam** with learning rate η and hyperparameters (β_1, β_2). A better visualization of these runs is in Figure 17. We performed 465 Adam runs in total:

$$
\begin{aligned}
B = 1,\ K = 1587200: &\quad \eta \in [10^{-5}, 3{\cdot}10^{-5}, 10^{-4}, 3{\cdot}10^{-4}, 10^{-3}, 3{\cdot}10^{-3}, 10^{-2}],\\
&\quad \beta_1 \in [0.8, 0.9, 0.95, 0.99, 0.999, 0.9999],\\
&\quad \beta_2 \in [0.8, 0.9, 0.95, 0.99, 0.999, 0.9999, 0.99999, 0.999999],\\
B = 16,\ K = 99200: &\quad \eta \in [10^{-4}, 3{\cdot}10^{-4}, 10^{-3}, 3{\cdot}10^{-3}, 10^{-2}],\\
&\quad \beta_1 \in [0.8, 0.9, 0.95, 0.99, 0.999],\\
&\quad \beta_2 \in [0.8, 0.9, 0.95, 0.99, 0.999, 0.9999],\\
B = 1024,\ K = 1550: &\quad (\eta, \beta_1, \beta_2) \in [10^{-4}, 3{\cdot}10^{-4}, 10^{-3}, 3{\cdot}10^{-3}, 10^{-2}]\\
&\qquad \times [0.8, 0.9, 0.95, 0.99, 0.999] \times [0.8, 0.9, 0.95, 0.99, 0.999]
\end{aligned}
$$

- **SGD** with learning rate η and momentum parameter β. We performed 106 SGD runs in total.

$$
\begin{aligned}
B = 1,\ K = 1587200: &\quad \eta \in [3{\cdot}10^{-4}, 10^{-3}, 3{\cdot}10^{-3}, 10^{-2}, 3{\cdot}10^{-2}, 10^{-1}, 3{\cdot}10^{-1}, 1],\\
&\quad \beta \in [0.8, 0.9, 0.95, 0.99, 0.999, 0.9999],\\
B = 16,\ K = 99200: &\quad \eta \in [10^{-2}, 3{\cdot}10^{-2}, 10^{-1}, 3{\cdot}10^{-1}, 1],\\
&\quad \beta \in [0.9, 0.95, 0.99, 0.999],\\
B = 1024,\ K = 1550: &\quad \eta \in [10^{-2}, 3{\cdot}10^{-2}, 10^{-1}, 3{\cdot}10^{-1}, 1],\\
&\quad \beta \in [0.8, 0.9, 0.95, 0.99, 0.999].
\end{aligned}
$$

- **SignSGD** with learning rate η and momentum parameter β. We performed 106 SignSGD runs in total.

$$
\begin{aligned}
B = 1,\ K = 1587200: &\quad \eta \in [10^{-5}, 3{\cdot}10^{-5}, 10^{-4}, 3{\cdot}10^{-4}, 10^{-3}, 10^{-2}],\\
&\quad \beta \in [0.8, 0.9, 0.95, 0.99, 0.999, 0.9999],\\
B = 16,\ K = 99200: &\quad \eta \in [10^{-5}, 3{\cdot}10^{-5}, 10^{-4}, 3{\cdot}10^{-4}, 10^{-3}, 3{\cdot}10^{-3}, 10^{-2}],\\
&\quad \beta \in [0.8, 0.9, 0.95, 0.99, 0.999, 0.9999],\\
B = 1024,\ K = 1550: &\quad \eta \in [10^{-5}, 3{\cdot}10^{-5}, 10^{-4}, 3{\cdot}10^{-4}, 10^{-3}, 3{\cdot}10^{-3}, 10^{-2}],\\
&\quad \beta \in [0.8, 0.9, 0.95, 0.99, 0.999, 0.9999].
\end{aligned}
$$

Warmup-clipping experiments for SGD. We perform additional experiments on the impact of clipping and warmup for SGD in Figure 18. These experiments amount to 112 additional training runs in the same Transformer setup outlined above.

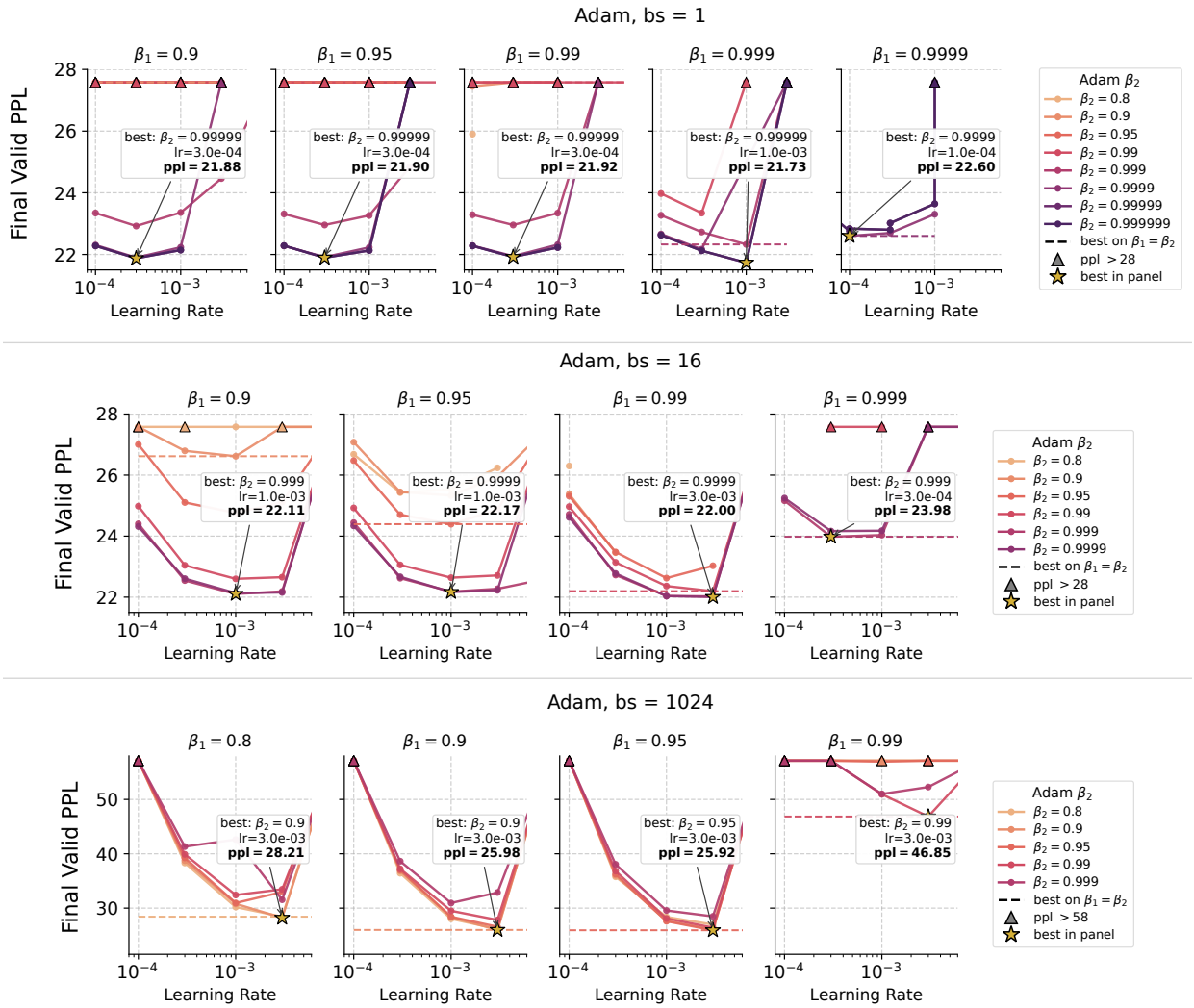


Figure 17: Complement to Figure 1: details of performance of Adam for different values of β_1.

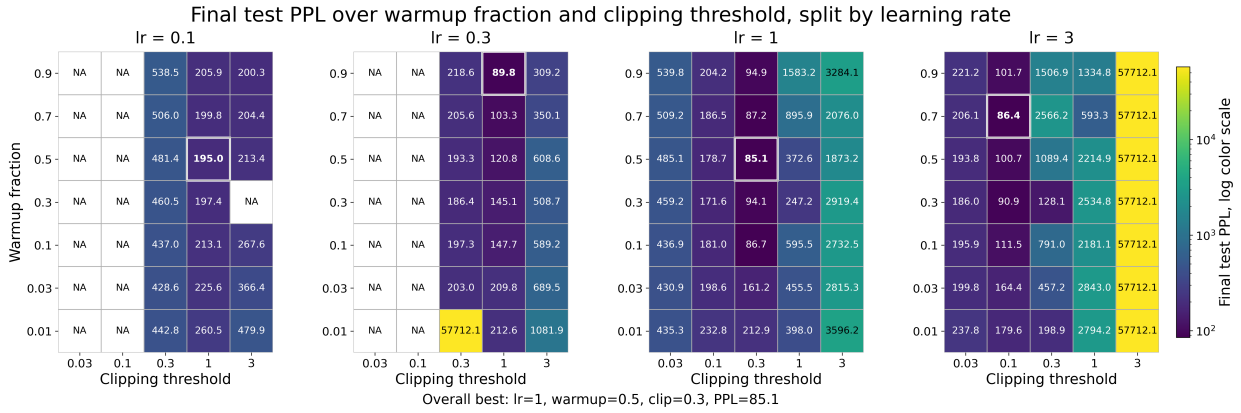


Figure 18: Complement to Figure 1: impact of learning-rate warmup and gradient clipping on SGD+m at batch size 1024 and momentum 0.9. We sweep learning rates, warmup fractions, and clipping thresholds. We find that it is possible to improve performance by tuning these parameters, yet it does not approach the performance of Adam or SignSGD+m.

A.3 Supplement for Subsection 2.3

We train a 12-layer Transformer on the SlimPajama (Figure 4 and Figure 19) and FineWeb (Figure 20, left) datasets using SGD with momentum and Adam. Batch sizes vary from 4 to 1024 across experiments, and we use different run lengths (i.e., different token budgets), with a WSD scheduler (Hägele et al., 2024). We observe that, at a fixed number of steps, the performance gap increases with batch size, and that with smaller batches and sufficiently long training, SGD outperforms Adam. We increase the model depth to 24 layers while keeping all other settings identical on SlimPajama (Figure 20, right).

Combined, the experiments in this section account for full training (at different token budgets) of more than 250 language models at different scales and batch sizes. Every reported result is relative to the best learning rate in our grid, defined for each setup.

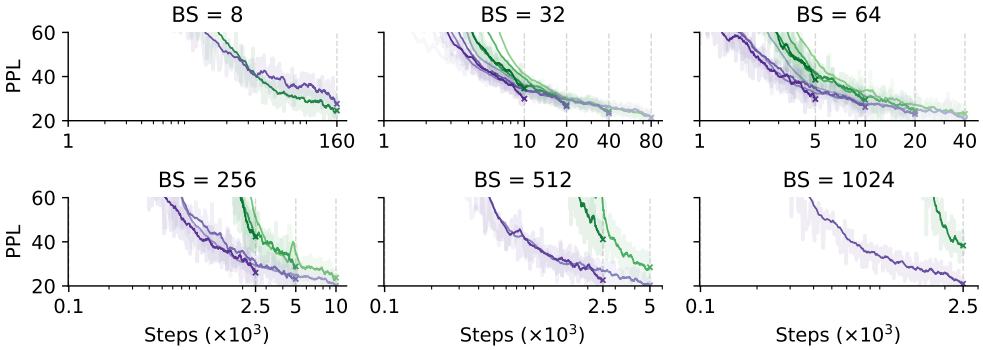


Figure 19: Perplexity during training for SGD (green) and Adam (purple) across different training lengths for a subset of batch sizes. Solid lines show the rolling mean of PPL values; lighter lines show the raw values. As before, the gap decreases the longer we train, and SGD can eventually outperform Adam.

Small-scale experiments. We consider SGD with $\beta = 0.98$ and global clipping before applying momentum. For Adam, we use the setting $\beta_1 = \beta_2 = 0.95$. Both settings are suggested by the sweep in Figure 1 and recent literature (Zhang et al., 2025; Orvieto & Gower, 2025; Zhao et al., 2024).

- For Figure 4 (SlimPajama, 160M), we choose a sequence length of 2048. Inspired by the careful tuning of Figure 1, we consider the learning rate grid $[0.25, 0.5, 1.0]$ for SGD and $[0.001, 0.002, 0.004]$ for Adam. We report training curves for this first setup in Figure 19.
- For Figure 20 (left: FineWeb, 160M parameters), we choose a sequence length of 1024. Our learning rate grid here is the same as for SlimPajama (previous point). At a run length of $160k$ steps, given our lack of experience with extremely low batch sizes, we operate on a slightly larger grid: $[0.0001, 0.0003, 0.001, 0.003]$ for Adam and $[0.03, 0.1, 0.3, 1]$ for SGD.
- For Figure 20 (right: SlimPajama, 250M parameters, 24 layers), we choose a sequence length of 2048, and we also operate on a larger grid: $[0.0001, 0.0003, 0.001, 0.003]$ for Adam and $[0.03, 0.1, 0.3, 1]$ for SGD.

Medium scale experiments. For all SGD runs, we use $\beta = 0.98$. For Adam, we use the standard choice (0.9, 0.95) (Biderman et al., 2023). All our runs use global norm clipping and no weight decay.

- **410M model** (Figure 5a): We train with sequence length 2048, for $500k$ steps on SlimPajama. Learning rate grid is $[1.25\cdot10^{-4}, 2.5\cdot10^{-4}, 5.0\cdot10^{-4}, 1.0\cdot10^{-3}]$ for Adam and $[0.125, 0.25, 0.5, 1]$ for SGD. The sweep results are presented in Figure 21a.
- **1B model** (Figure 5b): We train with sequence length 1024, for 850k steps on FineWeb. Learning rate sweep, shown in Figure 21b uses $[6.25\cdot10^{-5}, 1.25\cdot10^{-4}, 2.5\cdot10^{-4}, 5.0\cdot10^{-4}, 1.0\cdot10^{-3}]$ for Adam and $[0.0625, 0.125, 0.25, 0.5, 1]$ for SGD.

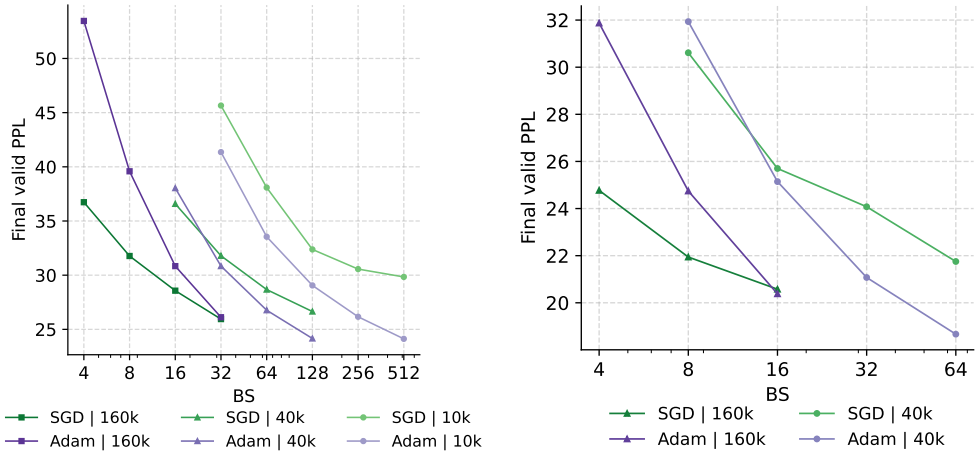
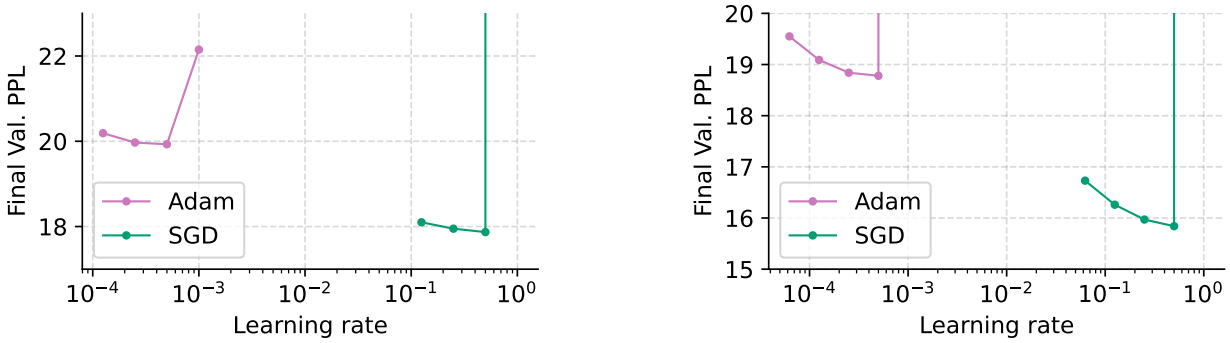


Figure 20: **(Left)** FineWeb dataset, sequence length 1024, 12-layer Transformer, **(Right)** SlimPajama dataset, sequence length 2048, 24-layer Transformer. Tuning discussed in Subsection 2.3 and Subsection A.3.



(a) 410M model on SlimPajama (seq. length 2048, batch size 8, 500k steps)

(b) 1B model on FineWeb (seq. length 1024, batch size 16, 850k steps)

Figure 21: Learning rate sweep for 410M and 1B models. Trajectories for the optimal learning rate are shown in Figure 5.

A.4 Supplement for Subsection 2.5

We use a 160M-parameter Transformer, as detailed in Subsection A.1, trained on 13B FineWeb tokens. All runs use a sequence length $L = 1024$. All methods use constant tuned learning rate with warmup (0.25B tokens); weight decay is omitted to avoid inflating the search space. Below, B indicates the batch size and K the number of steps.

Taken together, the experiments in this section cover the full training of 87 language models. Runs are summarized in the main paper in Figure 6 and presented in detail in Figure 22.

- **SGD** with momentum parameter β. 40 SGD runs total:

$$(B,K)=(512,24800):\quad\begin{cases}\beta=0.9: & \eta\in[3{\cdot}10^{-2},10^{-1},3{\cdot}10^{-1},1]\\ \beta=0.95: & \eta\in[3{\cdot}10^{-2},10^{-1},3{\cdot}10^{-1},1,3]\\ \beta=0.99: & \eta\in[3{\cdot}10^{-2},10^{-1},3{\cdot}10^{-1},1]\\ \beta=0.999: & \eta\in[3{\cdot}10^{-2},10^{-1},3{\cdot}10^{-1},1]\end{cases}\quad(17\text{ runs}),$$

$$(B,K)=(16,793600):\quad\begin{cases}\beta=0.95: & \eta\in[3{\cdot}10^{-2},10^{-1},3{\cdot}10^{-1},1]\\ \beta=0.99: & \eta\in[3{\cdot}10^{-2},10^{-1},3{\cdot}10^{-1},1]\\ \beta=0.999: & \eta\in[10^{-3},3{\cdot}10^{-3},10^{-2},3{\cdot}10^{-2},10^{-1},3{\cdot}10^{-1},1]\\ \beta=0.9999: & \eta\in[3{\cdot}10^{-4},10^{-3},3{\cdot}10^{-3},10^{-2},3{\cdot}10^{-2},10^{-1},3{\cdot}10^{-1},1]\end{cases}\quad(23\text{ runs}).$$

- **SignSGD** with momentum parameter β. 47 SignSGD runs total:

$$(B,K)=(512,24800):\quad\begin{cases}\beta=0.9: & \eta\in[10^{-4},3{\cdot}10^{-4},10^{-3},3{\cdot}10^{-3}]\\ \beta=0.95: & \eta\in[10^{-4},3{\cdot}10^{-4},10^{-3},3{\cdot}10^{-3}]\\ \beta=0.99: & \eta\in[10^{-5},3{\cdot}10^{-5},10^{-4},3{\cdot}10^{-4},10^{-3},3{\cdot}10^{-3}]\\ \beta=0.999: & \eta\in[10^{-4},3{\cdot}10^{-4},10^{-3},3{\cdot}10^{-3}]\end{cases}\quad(18\text{ runs}),$$

$$(B,K)=(16,793600):\quad\begin{cases}\beta=0.9: & \eta\in[3{\cdot}10^{-6},10^{-5},3{\cdot}10^{-5},10^{-4},3{\cdot}10^{-4},10^{-3}]\\ \beta=0.95: & \eta\in[3{\cdot}10^{-6},10^{-5},3{\cdot}10^{-5},10^{-4},3{\cdot}10^{-4},10^{-3}]\\ \beta=0.99: & \eta\in[3{\cdot}10^{-6},10^{-5},3{\cdot}10^{-5},10^{-4},3{\cdot}10^{-4},10^{-3}]\\ \beta=0.999: & \eta\in[3{\cdot}10^{-6},10^{-5},3{\cdot}10^{-5},10^{-4},3{\cdot}10^{-4},10^{-3}]\\ \beta=0.9999: & \eta\in[3{\cdot}10^{-7},10^{-6},3{\cdot}10^{-6},10^{-5},3{\cdot}10^{-5}]\end{cases}\quad(29\text{ runs}).$$

A.5 Supplement to Subsection 4.2

Algorithm 1 SGD with Adaptive Momentum Clipping

Require: Initial point x_0, learning rate η, momentum β, clipping fraction $p \in (0,1)$, $m_{-1} \leftarrow 0$.

```
for t = 0 to T − 1 do
    g_t ← g(x_t; ξ) (stochastic gradient at x_t)
    m_t ← β m_{t−1} + g_t
    Set clipping threshold τ_t as the (1 − p)-quantile of |m_t|
    m̂_t ← clip(m_t) = sign(m_t) · min(|m_t|, τ_t)
    x_{t+1} ← x_t − η m̂_t
end for
```

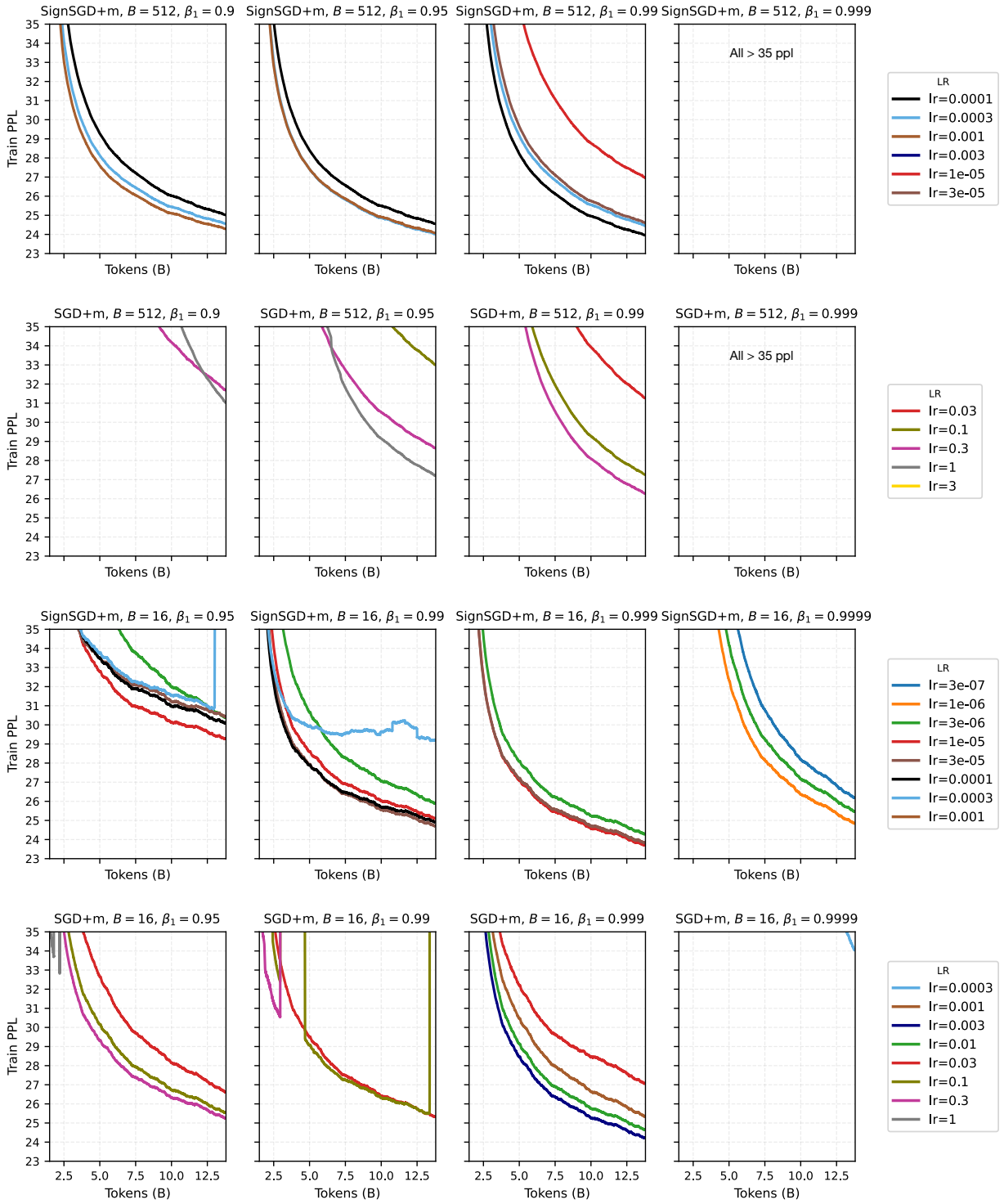


Figure 22: Training curves for all configurations in Subsection 2.5. Performance comparison for SGD+m and SignSGD+m (momentum denoted as β_1) at different batch sizes. We train the same Transformer as in Figure 1 for 13B tokens from FineWeb at a sequence length of 1024, using a constant learning rate with warmup. Figure 6 reports best-performing configurations at each token budget.

B Toy Quadratic Example

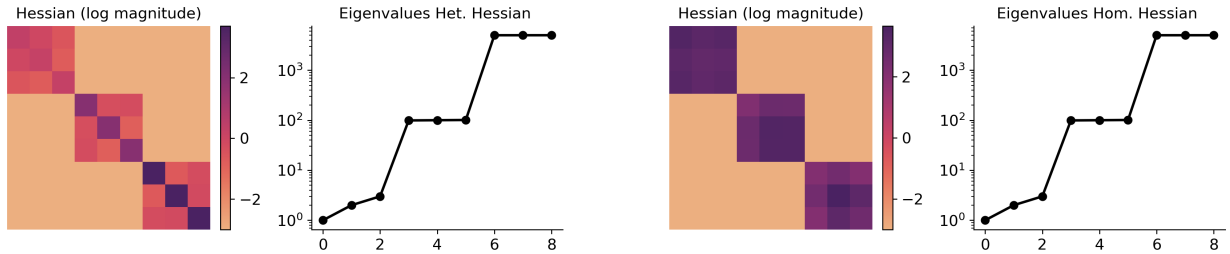


Figure 23: *(left) Heterogeneous and (right) homogeneous Hessians considered in Figure 11.*

Our setup is inspired by the results and discussions in Zhang et al. (2024), and uses the codebase of Orvieto & Gower (2025). We consider the loss

$$L(w) = \frac{1}{2} w^\top H w$$

where we construct the homogeneous and heterogeneous Hessians using the following procedure:

- We fix the eigenvalues, equal in both cases, to

$$\text{eig}(H_{\text{hom}}) = \text{eig}(H_{\text{het}}) = \{1, 2, 3, 99, 100, 101, 4998, 4999, 5000\}.$$

- We choose both Hessians to be block-diagonal, with 3×3 blocks. The homogeneous Hessian has eigenvalues of different magnitude in each block, while the heterogeneous keeps similar magnitudes in each block.

```
H_details_het = [[1,2,3],[99,100,101],[4998,4999,5000]]
H_details_hom = [[1,99,4998],[2,100,4999],[3,101,5000]]
```

- For each block, we apply a random rotation to the diagonal matrix of eigenvalues, specific to each block. Each rotation is sampled from the Haar measure by decomposition of a random 3×3 positive semidefinite matrix $AA^\top$, where $A \in \mathbb{R}^{3\times 3}$ has i.i.d. Gaussian entries.

The result is shown in Figure 23. Learning rates for each method are tuned.

Next, to introduce stochasticity in this setting, we simply take the square root of the Hessian to define a 9×9 design matrix X: $H = X^\top X, X = H^{\frac{1}{2}}$, and subsample a number (the batch size) of rows of X at each iteration. Tuning is reported in Figure 24.

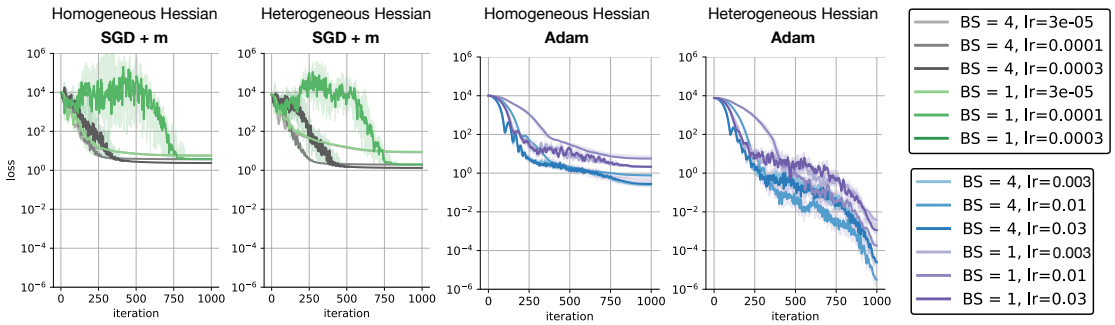


Figure 24: Complement to Figure 11.

C Proofs

C.1 Proof of Proposition 1

Using L-smoothness with $x_{k+1} - x_k = -\eta(\nabla f(x_k) + \sigma_B \zeta_k)$,

$$f(x_{k+1}) \leq f(x_k) + \langle \nabla f(x_k), x_{k+1} - x_k \rangle + \frac{L}{2}\|x_{k+1} - x_k\|_2^2 \tag{18}$$

$$= f(x_k) - \eta\langle \nabla f(x_k), \nabla f(x_k) + \sigma_B \zeta_k \rangle + \frac{L\eta^2}{2}\|\nabla f(x_k) + \sigma_B \zeta_k\|_2^2. \tag{19}$$

Let $\mathbb{E}_k[\cdot] := \mathbb{E}[\cdot|x_k]$. We have:

$$\mathbb{E}_k[f(x_{k+1}) - f(x_k)] \leq -\eta\left(1 - \frac{L\eta}{2}\right)\|\nabla f(x_k)\|_2^2 + \frac{L\sigma_B^2\eta^2}{2}\mathbb{E}_k[\|\zeta_k\|_2^2]. \tag{20}$$

Since $\mathbb{E}_k[\|\zeta_k\|_2^2] \leq d$ and $\eta \leq 1/L$, taking total expectation gives

$$\mathbb{E}[f(x_{k+1}) - f(x_k)] \leq -\frac{\eta}{2}\mathbb{E}[\|\nabla f(x_k)\|_2^2] + \frac{dL\sigma_B^2\eta^2}{2}. \tag{21}$$

Summing over $k = 0, \ldots, K-1$ and using $f(x_K) \geq f^*$, we obtain

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}[\|\nabla f(x_k)\|_2^2] \leq \frac{2(f(x_0) - f^*)}{\eta K} + dL\sigma_B^2\eta. \tag{22}$$

Since $\sigma_B^2 = \sigma^2/B$, this gives

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}[\|\nabla f(x_k)\|_2^2] \leq \frac{2(f(x_0) - f^*)}{\eta K} + \frac{dL\sigma^2\eta}{B}. \tag{23}$$

C.2 Proof of Proposition 2

Note that since $\sigma_B > 0$,

$$\mathrm{sign}(\nabla f(x_k) + \sigma_B \zeta_k) = \mathrm{sign}(\sigma_B^{-1}\nabla f(x_k) + \zeta_k) \tag{24}$$

Hence, since the problem decouples coordinatewise, it is sufficient to study the distribution of

$$\mathrm{sign}(y + \zeta), \qquad \text{where} \quad \zeta \sim \mathsf{p}_\zeta, \quad y \in \mathbb{R}. \tag{25}$$

We clearly have

$$\mathbb{E}[\mathrm{sign}(y + \zeta)] = \Pr(y + \zeta > 0) - \Pr(y + \zeta < 0) \tag{26}$$

$$= \Pr(\zeta > -y) - \Pr(\zeta < -y) \tag{27}$$

$$= 1 - 2\mathsf{F}(-y) \tag{28}$$

$$= 2\mathsf{F}(y) - 1. \tag{29}$$

where the last equation follows because p_ζ is symmetric. For a symmetric distribution we indeed have

$$\Pr[\zeta > y] = \Pr[\zeta < -y] = 1 - \Pr[\zeta > -y], \tag{30}$$

which implies

$$\mathsf{F}(y) = 1 - \mathsf{F}(-y), \tag{31}$$

so

$$1 - 2\mathsf{F}(-y) = -1 + (2 - 2\mathsf{F}(-y)) = 2\mathsf{F}(y) - 1. \tag{32}$$

Finally, note that

$$\begin{aligned}
\mathrm{Var}[\mathrm{sign}(y+\zeta)] &= \mathbb{E}[(\mathrm{sign}(y+\zeta))^2] - \mathbb{E}^2[\mathrm{sign}(y+\zeta)] && (33)\\
&= 1 - (2\mathsf{F}(y) - 1)^2 && (34)\\
&= -4\mathsf{F}(y)^2 + 4\mathsf{F}(y) && (35)\\
&= 4\mathsf{F}(y)(1 - \mathsf{F}(y)). && (36)
\end{aligned}$$

The result simply follows from plugging in $y = \sigma_B^{-1}\nabla f(x_k)$. Finally, recall that for any $p \in [0,1]$, $p(1-p) \leq 1/4$.

C.3 Proof of Lemma 1

Since p_ζ is symmetric, its CDF satisfies $\mathsf{F}(-x) = 1 - \mathsf{F}(x)$, hence $\mathsf{G}(x) = 2\mathsf{F}(x) - 1$ is an odd function. Moreover, for $x \geq 0$,

$$\mathsf{F}(x) - \tfrac{1}{2} = \int_0^x \mathsf{p}_\zeta(t)\,dt \geq 0 \tag{37}$$

which implies $\mathsf{G}(x) \geq 0$, and by oddness $x\mathsf{G}(x) \geq 0$ for all $x \in \mathbb{R}$.

Next, fix $D > 0$. By Assumption 3, there exists $m_D > 0$ such that

$$\mathsf{p}_\zeta(x) \geq m_D \qquad \text{for all } |x| \leq D. \tag{38}$$

Therefore, for $x \in (0, D]$,

$$\mathsf{G}(x) = 2\int_0^x \mathsf{p}_\zeta(t)\,dt \geq 2m_D x. \tag{39}$$

By oddness of G, this gives

$$\frac{|\mathsf{G}(x)|}{|x|} \geq 2m_D \qquad \text{for all } 0 < |x| \leq D. \tag{40}$$

Hence

$$\beta_D = \inf_{0<|x|\leq D} \frac{|\mathsf{G}(x)|}{|x|} \geq 2m_D > 0. \tag{41}$$

For the upper bound, by Assumption 3 with $D = 1$, there exists $M_1 < \infty$ such that

$$\mathsf{p}_\zeta(x) \leq M_1 \qquad \text{for all } |x| \leq 1. \tag{42}$$

If $|x| \leq 1$, then

$$|\mathsf{G}(x)| \leq 2M_1|x|. \tag{43}$$

If $|x| > 1$, then

$$|\mathsf{G}(x)| \leq 1 \leq |x|. \tag{44}$$

Thus, with

$$\alpha := \max\{2M_1, 1\}, \tag{45}$$

we have

$$|\mathsf{G}(x)| \leq \alpha|x| \qquad \text{for all } x \in \mathbb{R}. \tag{46}$$

C.4 Proof of Proposition 3

Recall that

$$\operatorname{sign}(\nabla f(x_k) + \sigma_B \zeta_k) = \mathsf{G}(\sigma_B^{-1} \nabla f(x_k)) + \varepsilon_k, \qquad \mathsf{G}(x) := 2\mathsf{F}(x) - 1. \tag{47}$$

We have:

$$f(x_{k+1}) \leq f(x_k) - \eta \left\langle \nabla f(x_k), \mathsf{G}(\sigma_B^{-1} \nabla f(x_k)) + \varepsilon_k \right\rangle + \frac{L\eta^2}{2} \left\| \mathsf{G}(\sigma_B^{-1} \nabla f(x_k)) + \varepsilon_k \right\|_2^2. \tag{48}$$

Let $\mathbb{E}_k[\cdot] := \mathbb{E}[\cdot | x_k]$. We have:

$$\mathbb{E}_k[f(x_{k+1}) - f(x_k)] \leq -\eta \langle \nabla f(x_k), \mathsf{G}(\sigma_B^{-1} \nabla f(x_k)) \rangle + \frac{L\eta^2}{2} \mathbb{E}_k[\| \mathsf{G}(\sigma_B^{-1} \nabla f(x_k)) + \varepsilon_k \|_2^2], \tag{49}$$

Now, note that the update norm is coordinatewise bounded by 1, so

$$\mathbb{E}_k[f(x_{k+1}) - f(x_k)] \leq -\eta \langle \nabla f(x_k), \mathsf{G}(\sigma_B^{-1} \nabla f(x_k)) \rangle + \frac{dL\eta^2}{2}, \tag{50}$$

Now use the assumption that

$$\| \sigma_B^{-1} \nabla f(x_k) \|_\infty \leq D. \tag{51}$$

By Lemma 1,

$$\left\langle \nabla f(x_k), \mathsf{G}(\sigma_B^{-1} \nabla f(x_k)) \right\rangle \geq \beta_D \sigma_B^{-1} \| \nabla f(x_k) \|_2^2. \tag{52}$$

Therefore,

$$\mathbb{E}_k[f(x_{k+1}) - f(x_k)] \leq -\eta \beta_D \sigma_B^{-1} \| \nabla f(x_k) \|_2^2 + \frac{dL\eta^2}{2}. \tag{53}$$

Taking total expectation, summing over $k = 0, \ldots, K-1$, and using $f(x_K) \geq f^*$, we obtain

$$\eta \beta_D \sigma_B^{-1} \sum_{k=0}^{K-1} \mathbb{E}[\| \nabla f(x_k) \|_2^2] \leq f(x_0) - f^* + \frac{dL\eta^2 K}{2}. \tag{54}$$

Dividing by $\eta \beta_D \sigma_B^{-1} K$ gives

$$\frac{1}{K} \sum_{k=0}^{K-1} \mathbb{E}[\| \nabla f(x_k) \|_2^2] \leq \frac{\sigma_B (f(x_0) - f^*)}{\beta_D \eta K} + \frac{dL\eta\sigma_B}{2\beta_D}. \tag{55}$$

Since $\sigma_B = \sigma / \sqrt{B}$, this is

$$\frac{1}{K} \sum_{k=0}^{K-1} \mathbb{E}[\| \nabla f(x_k) \|_2^2] \leq \frac{\sigma (f(x_0) - f^*)}{\beta_D \eta \sqrt{B}\, K} + \frac{dL\sigma\eta}{2\beta_D \sqrt{B}}. \tag{56}$$