

# IS YOUR BATCH SIZE THE PROBLEM? REVISITING THE ADAM-SGD GAP IN LANGUAGE MODELING

**Anonymous authors**

Paper under double-blind review

## ABSTRACT

Adam is known to perform significantly better than Stochastic Gradient Descent (SGD) in language models, a phenomenon for which a number of explanations have been proposed. In this work, we revisit this “optimizer gap” through a series of comprehensively tuned baseline training runs for language modeling with Transformers. We exhaustively study how momentum, gradient clipping, and batch size affect the gap between SGD and Adam. Our empirical findings show that SGD with momentum can actually perform similarly to Adam in small-batch settings, if tuned correctly. We revisit existing explanations for Adam’s advantage, including heavy-tailed class imbalance, directional sharpness, and Hessian heterogeneity, which struggle to directly explain this phenomenon. Towards bridging this gap in our understanding, by analyzing our Transformer training runs and simple quadratic settings inspired by the literature, we provide new insights, driven by stochastic differential equation models, into the role of batch size on the training dynamics.

## 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 the success of Adam, and thousands of GPU hours every day are spent at large companies using Adam to train their next-generation large language models.

Even in new sophisticated optimization pipelines looking to dethrone Adam, such as Muon (Jordan et al., 2024), most current implementations (Liu et al., 2025; Shah et al., 2025) rely on plain Adam with weight decay (AdamW, Loshchilov & Hutter (2019)) for critical subsets of parameters, such as normalization layers, text embeddings and prediction heads. This new world is still a bit surprising. Until around 2018, Adam was used only occasionally, while stochastic gradient descent (SGD) with momentum was known to lead to 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, language modeling with Transformers required Adam. In fact, Transformer LMs have been reportedly untrainable with SGD (Xiong et al., 2020), especially due to the critical parameters listed above.

Over the years, researchers have offered a number of compelling explanations regarding the remarkable performance of Adam compared to SGD in language modeling, attributing it either to the peculiar noisy nature of text data (Zhang et al., 2020b;a) or the heterogeneous structure (Noci et al., 2022; Zhang et al., 2024a) of the Transformer architecture (Vaswani et al., 2017) — comprising semantically and structurally dissimilar layers. While most hypotheses regarding the Adam-SGD gap can help guide our understanding (Ahn et al., 2024), a particularly crucial insight was recently brought to light by Kunstner et al. (2023): the Adam-SGD gap is also observable in full-batch training, and is hence clear that the stochastic and potentially heavy-tailed nature of stochastic gradients may not be the challenge Adam is able to tackle. Inspired by the latter discussion, we take an **orthogonal approach**:

*Instead of asking why Adam often outperforms SGD, we wonder:  
In which Transformer-based language model training setting, if any, does SGD work?*

In other words, while most recent works try to maximize the gap between SGD and Adam in order to explain it more easily, we here try to minimize it – without sacrificing scale or performance. We

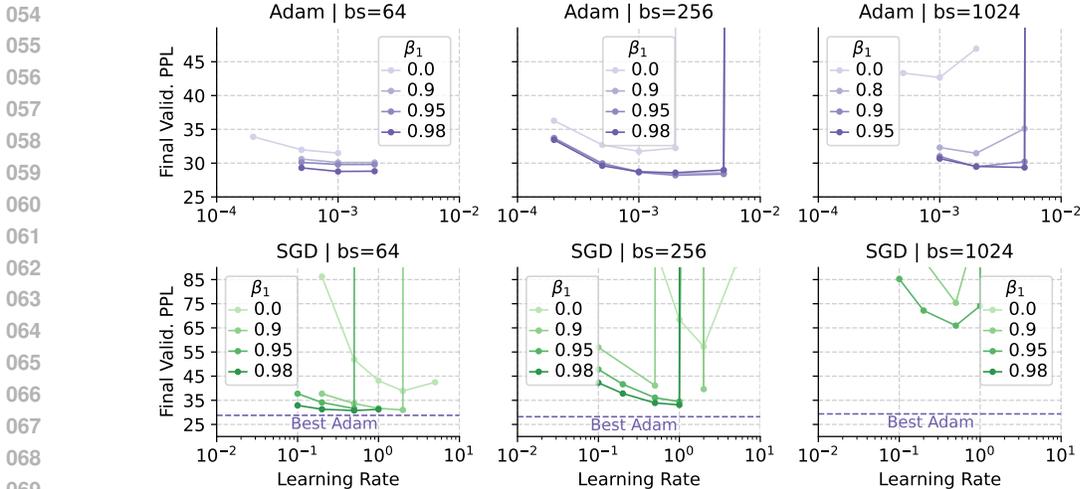


Figure 1: Learning rate and momentum sweep for SGD and Adam across batch sizes under a fixed 1.3B-token compute budget. Experiments use a 160M-parameter model and report perplexities on 100M held-out tokens. **Adam performs consistently across batch sizes, while SGD performs poorly at large batch sizes but improves at smaller ones.** The dashed line marks the best Adam configuration per batch size. Best hyperparameters and perplexities are listed in Table 1.

believe such a view is novel in the literature and can provide valuable insights into the Adam-SGD gap. In particular, it can help identify settings that falsify existing hypotheses about the gap and enumerate necessary criteria that explanations must fulfill. Our contributions are as follows:

- Despite our surprise, we show that LMs can be trained with SGD and achieve performance close to Adam at the same token budget, as long as the batch size is small. We found that this holds even at 1B parameters. While this setting is clearly inconvenient for standard multi-device pretraining, it provides a new lens for understanding the Adam-SGD gap. We note that this finding is consistent with both previously observed trends for small-scale models in the large/full-batch case (Kunstner et al., 2023) and with works observing that adaptive preconditioning affects the critical batch size (Zhang et al., 2019). Yet, taken in isolation, our small-batch results urge us to revisit the theoretical underpinnings for the Adam-SGD gap.
- To inspect this phenomenon, 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., 2024a) — in our setup. While our experiments confirm that these explanations can shed light and are useful to describe settings where Adam outperforms SGD, we find that no prior work can directly explain why SGD can outperform Adam at low batch sizes, while achieving satisfactory performance. Notably, in stark contrast with works attributing the gap to heavy-tail noise, we observe that increased stochasticity actually reduces the Adam-SGD gap.
- We enhance our intuition by further studying what makes SGD suboptimal and potentially unstable at high batch sizes. To do this, we present ablations on gradient clipping and learning rate grafting (Agarwal et al., 2020), and inspect their effect on performance.
- Inspired by our observed empirical correspondence between Transformers dynamics and the simplified heterogeneous quadratic setup of (Zhang et al., 2024a) at different noise levels, we leverage this setup to further study why adaptive optimization may have a different batch size sensitivity compared to SGD. Our analysis is rooted in recent works on SDE models (Malladi et al., 2023; Compagnoni et al., 2025b), and our findings and theoretical connections provide evidence of acceleration for Adam in the large batch settings.

Together, our findings paint a new picture of the optimizer gap, and suggest that batch size — and thus the scale and structure of gradient noise — should be explicitly considered in future analyses. Moreover, our results shift the discussion to considerations on the critical batch size of each optimizer, and can provide practical hints in low-resource and small-scale settings, where small batches are common and optimizer memory usage is critical. This last point was recently developed in a concurrent paper by Marek et al. (2025a), where the authors study hyperparameter robustness for small-batch size training in Adam and SGD. In this work, our focus lies primarily in revisiting explanations for the optimizer gap in light of our empirical results.

## 1.1 RELATED WORK

**Class imbalance.** [Kunstner et al. \(2024\)](#) explains the advantage of Adam over SGD on language tasks through the heavy-tailed class imbalance in token distributions. They show that with SGD, loss for rare tokens decrease much more slowly, making training inefficient, while Adam makes steady progress even on low-frequency tokens. Their empirical findings hold across architectures and settings, including non-Transformer architectures and non-textual imbalanced data. This suggests that the performance gap is primarily driven by class imbalance.

**Transformer architecture.** Another line of work focuses on the specific characteristics of Transformer architectures. [Zhang et al. \(2024a\)](#) provide a Hessian-based perspective, showing Transformers have block-heterogeneous Hessian spectrum. In such settings, Adam strongly outperforms SGD, unlike in architectures with more homogeneous Hessian. This holds across modalities, including ViTs, differing from [Kunstner et al. \(2024\)](#).

In contrast, [Tomihari & Sato \(2025\)](#) focus on gradient heterogeneity, explaining Hessian heterogeneity through gradient–Hessian correlations. They show that large disparities in gradient norms across parameters cause challenges for SGD, which Adam’s adaptivity addresses. Finally, [Zhang et al. \(2024b\)](#) show that full Adam-style adaptivity is not necessary and can be applied blockwise, as also noted by [Zhao et al. \(2024\)](#), who emphasize the importance of adaptivity for normalization layers.

**Heavy-tailed gradient noise.** Earlier work by [Zhang et al. \(2020b\)](#) asks whether the nature of stochastic gradient noise explains why the Adam–SGD gap exists in Transformers but not in other architectures. They show that Transformers produce gradient noise with heavy-tailed distributions, unlike the nearly Gaussian noise in CNNs, and argue that this degrades SGD performance while Adam remains robust. However, [Kunstner et al. \(2023\)](#) show that noise alone is not the primary cause of Adam’s superiority, since the gap persists even in full-batch training and Adam’s advantage grows as stochastic noise vanishes. Although their analysis focuses on smaller-scale setups and very large batch sizes, our findings align with extrapolations of their trends to the small-batch regime.

**Optimization trajectories.** Several studies investigate how Adam differs from SGD by analyzing optimization trajectories. [Jiang et al. \(2022\)](#) examine local geometry and define a statistic measuring the uniformity of the Hessian diagonal. On LMs, they find Adam consistently moves through regions with smaller values than SGD with momentum. Rather than examining the entire Hessian, [Pan & Li \(2023\)](#) look at the sharpness along the update direction at each step, showing that Adam makes updates in directions with much smaller sharpness than SGD.

**Evidence from simplified settings.** 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.

## 2 ADAM VS. SGD: EFFECTS OF HYPERPARAMETERS AND TRAINING REGIMES

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

### 2.1 EXPERIMENTAL SETUP

We conduct most experiments on the SlimPajama ([Soboleva et al., 2023](#)) dataset using a 160M-parameter nanoGPT ([Karpathy, 2022](#)) model, enhanced with recent improvements. Full model details are provided in [Appendix A](#). We also experiment with larger models, up to 1B parameters in a Pythia configuration ([Biderman et al., 2023](#)), and on the Fineweb dataset ([Penedo et al., 2024](#)).

All experiments are conducted without weight decay. Global gradient norm clipping is applied to raw gradients for both SGD (with momentum) and Adam. Other experiment-specific details are described in the following subsections, with additional information available in [Appendix A](#).

### 2.2 EFFECT OF BATCH SIZE ON THE ADAM-SGD GAP

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

162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215

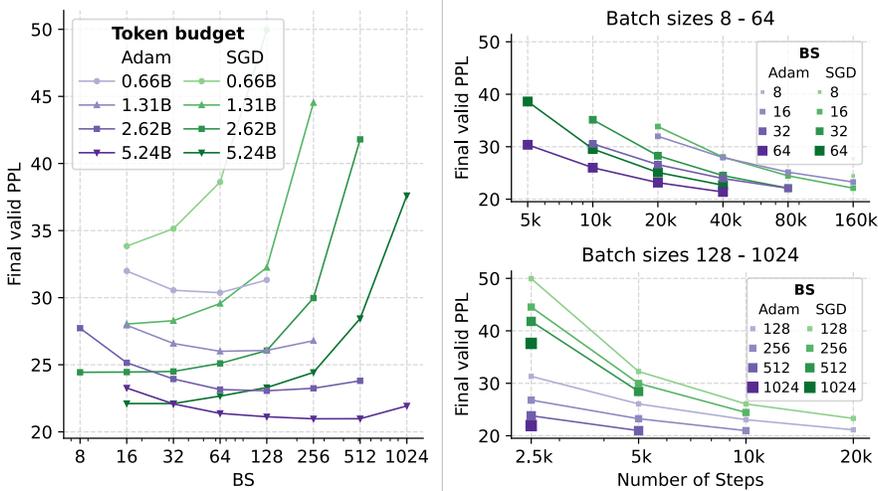


Figure 2: SGD (green) and Adam (purple) performance across batch sizes. Left: fixed token budget (darker colors – more tokens); the gap increases with batch size across all token budgets. Right: fixed number of steps (darker colors – larger batch sizes); the gap decreases with the number of steps. SGD improves with longer training and can match Adam, given a sufficiently small batch size.

**Setup.** All experiments use a sequence length of 512, a fixed token budget of 1.3B tokens, and a cosine learning rate scheduler (Loshchilov & Hutter, 2016) with 10% warmup. We compare three batch sizes: 64, 256, and 1024. The learning rate and momentum values are tuned for both optimizers at a batch size of 256. A sweep is performed over 5 learning rates and momentum values of 0.9, 0.95, and 0.98, including runs without momentum. High momentum values are motivated by findings from Zhao et al. (2024), where SGD performs best with momentum 0.98. For Adam, we fix  $\beta_2 = 0.95$ . Based on the optimal learning rate found at batch size 256, we scale down the learning rate grid for batch size 64 and scale it up for batch size 1024, sweeping over 3 values in each case. Results are reported in Figure 1. Some settings become unstable at very large learning rates, where one run may succeed, even if the median run diverges. In those settings, we report runs at the largest stable learning rate as optimal.

**Results.** Adam shows similar performance across batch sizes under a fixed token budget, as shown in Figure 1. Surprisingly, SGD achieves performance close to Adam with small batches, but the gap grows as batch size increases. For both SGD and Adam, momentum becomes crucial once batch size increases, as noted by Kunstner et al. (2023) and Zhao et al. (2024).

**Additional observations.** We find that using a relatively small sequence length of 512 is not a crucial factor in these dynamics. As we show in the next section, qualitatively similar behavior occurs at a sequence length of 2048, as long as the number of tokens per iteration is held constant. This suggests that performance differences can be attributed to the effective batch size (in tokens) rather than sequence length alone. Additionally, we observe that gradient clipping acts differently across batch sizes and is more important at larger batches (see Appendix B.1). Finally, we observe that warm-up length is not a confounder – sweeping 5–20% warm-up schedules in our cosine with warmup scheduler does not affect these dynamics.

### 2.3 ARE LARGE BATCH SIZES THE PROBLEM, OR IS IT THE NUMBER OF STEPS?

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

To investigate this, we compare performance across batch sizes under two training regimes: (1) fixed token budget and (2) fixed number of steps. This comparison allows us to separate the impact of slow SGD convergence from the inherent difficulty of optimizing in large-batch regimes.

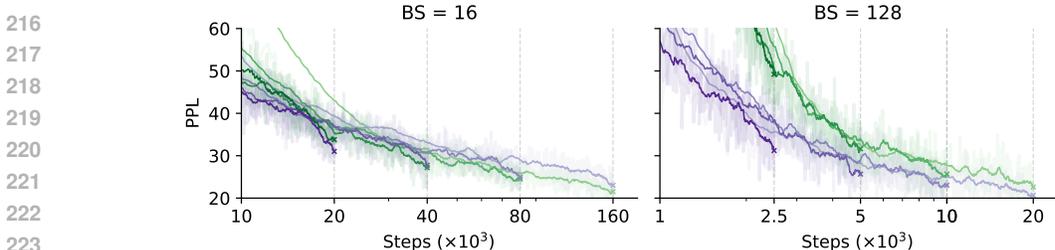


Figure 3: Perplexity during training for SGD (green) and Adam (purple) across training lengths in small- and large-batch settings. High-opacity lines show EMA; low-opacity lines show raw values. The gap shrinks with training, and for small batches, SGD outperforms Adam at the longest run.

**Setup.** The experimental setting is as in the previous section, except we increased the sequence length to 2048 to study its effect. We train models across batch sizes 8–1024 and steps 2.5k–160k. Total token budgets range from approximately 650M to 5.2B: models using larger batches are not trained for the largest number of steps, while models using smaller batches are trained for more steps. We switch from the cosine scheduler used previously to a WSD scheduler (Hägele et al., 2024) to better compare runs before learning rate decay begins, using a fixed 2000-step warmup. For SGD, we use momentum  $\beta = 0.98$ ; for Adam, we set  $\beta_1 = \beta_2 = 0.95$ . See Appendix A.2 for full details.

**Results.** The left panel of Figure 2 clearly shows that, at a fixed token budget, Adam improves with larger batch sizes, up to some **critical batch size**. In contrast, SGD shows a drastically opposite trend — performance consistently degrades as batch size increases. Under a fixed token budget, matching performance between Adam and SGD is conditional on using very small batch sizes, leading to significantly longer training and poor memory usage. This result highlights a key limitation of SGD: it is highly inefficient in realistic model training, where large batches are required for efficiency.

In the right panel of Figure 2, we show performance after training with various numbers of steps. The gap between Adam and SGD grows with batch size, but SGD improves significantly with more steps and can eventually match or even outperform Adam with long enough training. To illustrate this, we report perplexity during training for SGD and Adam with batch sizes 16 and 128 in Figure 3, and show the same plots for other batch sizes in Appendix A.2.

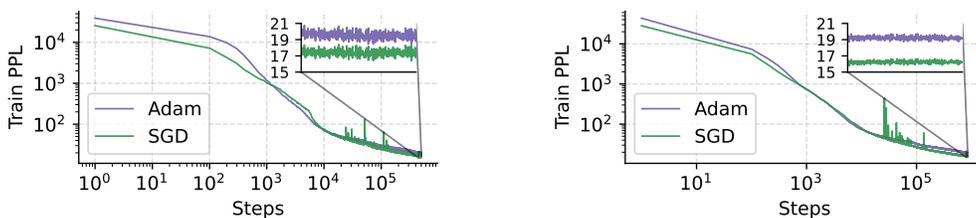
This observation shows that SGD is not necessarily unable to optimize in large-batch settings, just very slow to converge. Or, to put it differently: while Adam can accelerate (in terms of progress per step) with increased batch size, SGD cannot — its **critical batch size** is close to 1.

**Scaling experiments.** To test whether our findings persist at scale and across datasets, we experiment with larger models and the FineWeb dataset (Penedo et al., 2024) in addition to SlimPajama. We repeat the same experiments — varying token budgets and number of training steps — first by changing the dataset for the 160M model to FineWeb, and then by scaling the model to 250M on FineWeb. Setup details results are reported in Appendix A and Appendix B.2, showing that our core claims hold when scaling up the model and switching datasets.

To further test whether SGD can outperform Adam at larger scales, we scale the model to 410M and 1B. For both, we only tune the learning rate; SGD momentum is set to  $\beta = 0.98$  and Adam  $\beta_1 = \beta_2 = 0.95$ . We use the largest batch size that fits on a single NVIDIA A100 80GB GPU without gradient accumulation. This results in batch size 8 for the 410M model (SlimPajama, sequence length 2048) and batch size 16 for the 1B model (FineWeb, sequence length 1024). Full training details are in Appendix A. Trajectories for both models are shown in Figure 4, demonstrating that **SGD can outperform Adam even at 410M and 1B parameters**.

## 2.4 TUNING ADAM IN SMALL-BATCH SETTINGS

In previous sections, we showed that SGD can perform on par with Adam or even outperform it. One important detail is that even in Section 2.2, where hyperparameters are carefully tuned, we did not explore the effect of tuning Adam’s  $\beta_2$ . In all experiments so far, we used default values recommended in the literature. We believe working in this setup is valuable, since  $\beta_2$  is often kept constant while scaling (Biderman et al., 2023; Wortsman et al., 2023; Zhang et al., 2025), except in critically large scenarios (Molybog et al., 2023). However, Zhang et al. (2022) shows that higher  $\beta_2$  values substantially improve small-batch training, and Marek et al. (2025b) highlights the importance



(a) 410M model on SlimPajama – 1.5 days of training. (b) 1B model on FineWeb – 5 days of training.

Figure 4: SGD can outperform Adam even at 410M and 1B scales in small-batch regimes.

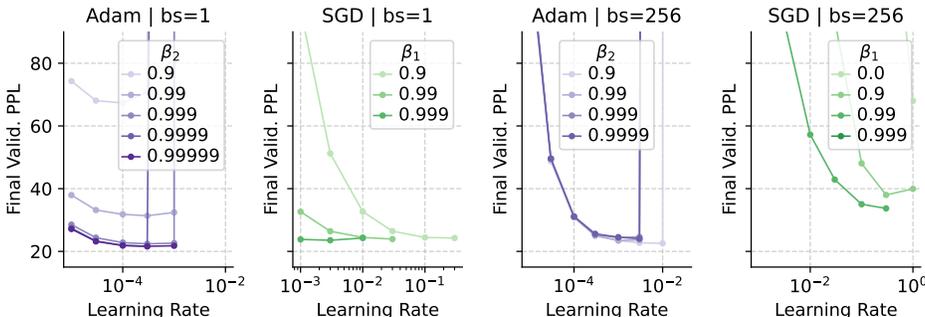


Figure 5: Effect of Adam’s  $\beta_2$  on the gap for batch sizes 1 and 256. High  $\beta_2$  improves Adam’s small-batch performance, while large-batch performance is insensitive.

of scaling  $\beta_2$  in this regime. In this section, we study this effect. The experimental setup is similar to Section 2.3, and we compare two extreme cases, batch size 1 and 256, under the same token budget. We fix Adam’s  $\beta_1 = 0.9$ , vary  $\beta_2$ , and tune the learning rate for each. To analyze the gap, we also tune momentum  $\beta_1$  for SGD. Details are in Appendix A.3. From Figure 5, for batch size 1, we observe that Adam requires higher  $\beta_2$  values, larger than 0.99, to achieve good final perplexity. In contrast, for batch size 256, Adam’s final performance is not sensitive to  $\beta_2$ . Compared to SGD, Adam consistently achieves lower final perplexity, even at batch size 1. Although SGD does not outperform Adam, the gap shrinks substantially as batch size decreases, showing that SGD remains competitive in the small-batch regime, even with carefully tuned Adam  $\beta_2$ . Results for batch size 8, along with full perplexity curves during training, are provided in Appendix B.3.

### 3 REVISITING PRIOR EXPLANATIONS

Several recent works have proposed explanations for Adam’s advantage over SGD through the lens of data or architectural properties (see Section 1.1). All these explanations improve our understanding of the performance gap, yet most are limited to scenarios where the gap between Adam and SGD is pronounced. Our goal is to revisit these explanations through the lens of our findings in Section 2 and ask: *Can they also account for strong SGD performance in small-batch settings?*

In this section, we focus on two approaches: the heavy-tailed class imbalance hypothesis (Kunstner et al., 2024) and the explanation based on heterogeneous Hessian structure (Zhang et al., 2024a). We also analyze the explanation proposed by Pan & Li (2023), with detailed discussion in Appendix C, which shows that directional sharpness correlates with the gap but does not explain it or relate to batch size. While no discussion in the literature fully accounts for our empirical evidence, the heterogeneous toy quadratic example of Zhang et al. (2024a) offers particularly valuable insights, which we develop in Section 4.1.

#### 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

324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377

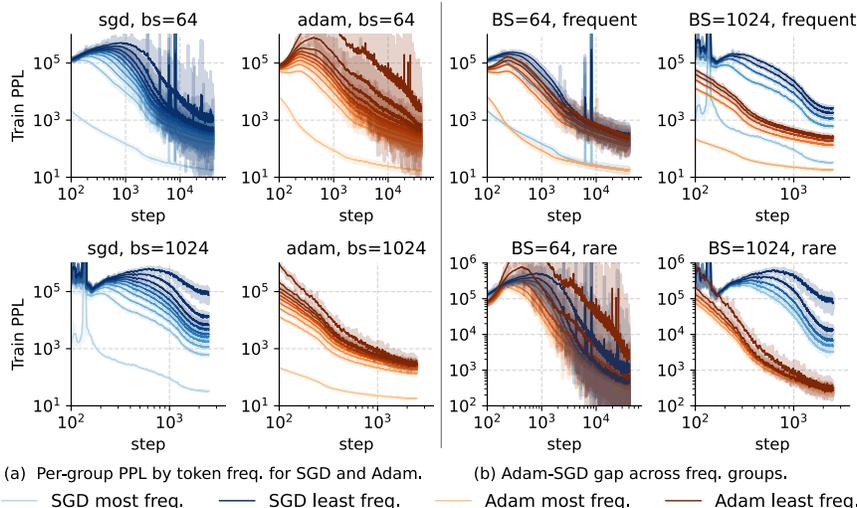


Figure 7: **(left)** Perplexity during training for Adam and SGD in small- and large-batch settings, computed per frequency group. SGD shows a larger gap in the large-batch setting, while the opposite holds for Adam. **(right)** Comparison of the Adam–SGD gap across frequency groups, with frequent tokens on the top row and rare ones on the bottom. SGD underperforms Adam in large batches, especially on rare tokens. This effect is not present in the small-batch setting.

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 Section 2.2, comparing batch sizes 64 and 1024, where SGD performs drastically differently, using runs with the optimal combination of  $\beta_1$  and learning rate for each case. We find that class imbalance exists in both cases: the persistence of low- and high-frequency tokens is similar, as shown in Figure 6. 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 all training regimes*.

We further compute perplexity separately for each frequency group and report it over training. From Figure 7 (a), we observe that both optimizers make faster progress on more frequent tokens in all settings, as expected. The relative difference in perplexity between frequency groups is more significant for SGD in the large-batch setting than for the small, while the opposite holds for Adam.

Comparing Adam and SGD across frequency groups in Figure 6, we observe that in the large-batch setting, SGD underperforms Adam across all groups, as shown in Figure 7 (b). However, the gap is notably more significant for less frequent tokens, which aligns well 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 regime. Here, not only is the overall Adam-SGD gap small (as seen in Section 2), but the gap across token frequencies is also small. We would expect this problem with SGD to hold independent of batch size, but in settings where SGD works well, the issue disappears.

### 3.2 HESSIAN HETEROGENEITY

From the line of work focusing on the architectural properties of Transformers, Zhang et al. (2024a) 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, **offering an explanation**

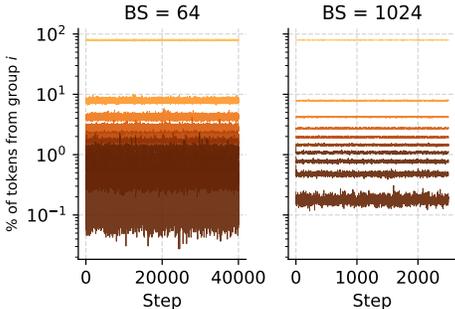


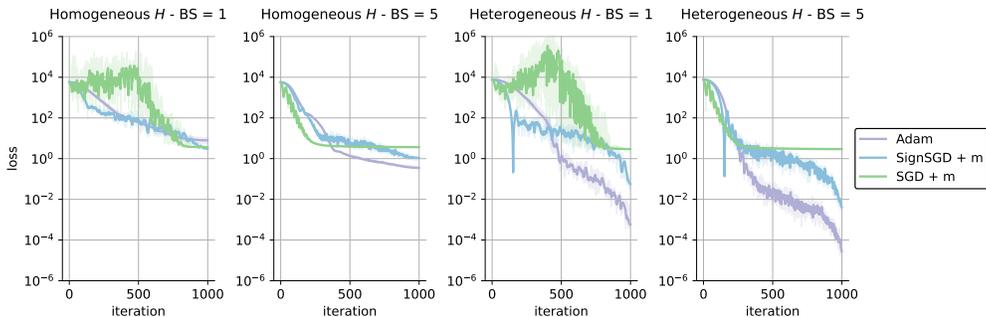
Figure 6: Batch token distribution for batch sizes 64 and 1024. Lighter colors show less frequent tokens. Statistics at lower batch sizes are noisier but of similar magnitude.

378 **that is invariant to batch size.** To further explore the effect of batch size on heterogeneous  
 379 problems, we revisit the simplified quadratic setting from their work and extend it by including  
 380 batch size variation. We compare optimization on problems with homogeneous (CNN-like) and  
 381 heterogeneous (Transformer-like) Hessians, where both share the same eigenvalue spectrum. We  
 382 train with SGD and Adam using a cosine learning rate schedule and no clipping.

383 From Figure 8, we observe the following:

- 384 • Across batch sizes, the largest Adam–SGD gap occurs in the heterogeneous setting. As noted  
 385 by Kunstner et al. (2023), a similar pattern appears for signed momentum (Bernstein et al., 2018),  
 386 which we develop further in Section 4.1.
- 387 • Adam benefits from larger batch sizes in both homogeneous and heterogeneous Hessian problems,  
 388 whereas SGD shows little improvement. Details of the setting are provided in Appendix F.

389 In summary, higher batch sizes boost performance for both SignSGD and Adam, **regardless of**  
 390 **heterogeneity.** While heterogeneity amplifies the Adam–SGD gap, this shows that the phenomenon  
 391 we study is not limited to heterogeneous settings, highlighting that the landscape structure plays a  
 392 less crucial role, which we further develop in Section 4.1.



394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405 Figure 8: Adam–SGD performance gap across batch sizes in quadratic models from Zhang et al.  
 406 (2024a). Adam benefits from larger batches in both, more so in Heterogeneous. Learning rates are  
 407 tuned so to give similar performance at Homogeneous batch size 1. Shown is mean and 2-sigma  
 408 standard deviation for 10 runs.

## 4 UNDERSTANDING HOW PERFORMANCE RELATES TO BATCH SIZE

410  
411  
412 In Section 3, we saw that while prior work sheds light on the Adam-SGD gap in the large-batch  
 413 regime, it remains unclear how batch size itself factors into these explanations. Towards gaining more  
 414 insights, we proceed as follows:

- 415 • We approach this from the SGD angle and ask: *What goes wrong for SGD in large-batch settings*  
 416 *that does not appear at small batch sizes?* To investigate, we separate which component of the  
 417 SGD update is more problematic — its direction or magnitude — focusing on the setting from  
 418 Section 2.2 with batch size 1024 and the optimal combination of  $\beta_1$  and learning rate. Using  
 419 grafting and adaptive clipping, we find that the main issue is the *update direction* rather than  
 420 magnitude. Full discussion and experimental results are provided in the Appendix D.
- 421 • In Section 4.1 we take a different approach, one based on noise statistics and adaptivity in a setup  
 422 which is non-specific to the Transformer architecture. This analysis is inspired by the results  
 423 in Figure 8, showing how adaptive methods may profit from large batch sizes regardless of the  
 424 Hessian structure. Using theoretical tools, we prove here that while SGD performance in early  
 425 training is dominated by number of iterations, the dynamics of signed momentum methods (cf.  
 426 Figure 8) showcase a strong dependency on batch size from the very first iterations.

### 4.1 THEORETICAL INSIGHTS

428  
429  
430  
431 Towards explaining the phenomena observed in this paper — and specifically the quadratic example  
 in Figure 8 — we provide an analysis based on results around the interaction between noise and

adaptivity in (Compagnoni et al., 2025b). The discussion below is rooted on the observed similarity between behaviors of Adam and SignSGD in Figure 8, as well as recent literature on their relation (Kunstner et al., 2023; Jordan et al., 2024).

Let  $X$  denote the model parameters and  $\gamma$  denote a batch of size  $B$ . We denote the stochastic gradient as  $\nabla f_\gamma(x) := \frac{1}{B} \sum_{i \in \gamma} \nabla(f_i(x))$  and by  $\Sigma(x)$  the noise covariance at batch size 1. The stochastic differential equation (SDE) approximation of SGD reads (Mil’shtein, 1986; Liu et al., 2021)

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

We now state a recent result showing that the drift of signed updates – driving performance in early training – has an extra dependency on the batch size. A proof sketch is provided in Appendix E.

**Theorem 1** ((Compagnoni et al., 2025b)). *Under the assumption of i.i.d. Gaussian noise with diagonal covariance (and, with minor modifications, for other noise structures and non-diagonal covariance), the following SDE provides a 1-weak approximation (Mil’shtein, 1986) of SignSGD:*

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

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

While Compagnoni et al. (2025b) provide a similar result for the Heavy-tail noise setting, the Gaussian case already highlights a crucial distinction between signed gradient methods and classical SGD.

**Takeaway.** Recall that  $\operatorname{erf}$  is linear on a large interval around zero. The local update of parameters is then driven by  $-\operatorname{erf}\left(\sqrt{\frac{B}{2}}\Sigma^{-\frac{1}{2}}\nabla f(X_t)\right)dt$  in the signSGD case, while in the SGD setting, this term is simply  $-\nabla f(X_t)dt$ . When everything else is kept constant, increasing the batch size  $B$ , increases the drift in the direction of the negative gradient by  $\sqrt{B}$ , up until the saturation point of  $\operatorname{erf}$ , i.e. a critical batch size.

This analysis provides evidence for our results: *using large batch sizes accelerates convergence* (larger drift) in signSGD (and likely also in closely-related algorithms, like Adam), while the performance of SGD in early training is batch-size agnostic and hence driven by the number of iterations (see Figure 9).

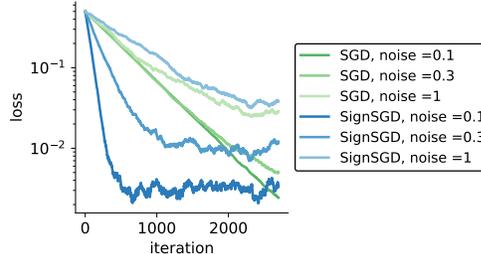


Figure 9: Illustration of theory presented in the section. Optimizing  $f(x) = \frac{1}{2}\|x\|^2$ ,  $x \in \mathbb{R}^{100}$ . All methods use same learning rate of  $1e-3$  and no momentum. Shown is performance under different injected Gaussian noise standard deviation. SGD in early training is dominated by the drift component, which is independent of noise – progress is driven by number of iterations. For SignSGD, noise (hence batch size in the more general case) directly affects drift and early progress.

## 5 DISCUSSION

Is it impossible to train language models with SGD? In this work, we show that there exist settings, namely small batch sizes with carefully tuned momentum, where SGD is competitive, even for 1B-scale language models. These findings are interesting in their own right for small-scale training runs, for example, on commodity GPUs, where memory is limited. Yet, these findings also crucially inform the space of possible theories for the optimizer gap between Adam and SGD. We revisit a number of promising theories from the literature based on our findings and find that they have limited explanatory power. We argue that, instead, the effect of batch size is a symptom of the importance of gradient noise for this question, and discuss a stronger explanation based on SDEs.

We believe that future theoretical work might be able to better explain the Adam-SGD gap as a function of the batch size. A promising direction is recent work on  $(L_0, L_1)$ -smoothness (Zhang et al., 2020a), specifically in the context of signed gradient descent (Compagnoni et al., 2025a), as well as  $\ell_\infty$  geometry (Xie & Li, 2024; Xie et al., 2024) of transformer models.

## 486 ACKNOWLEDGMENT OF AI-ASSISTED TOOLS

487  
488 AI-assisted editing tools were used to check grammar.  
489

## 490 REFERENCES

- 491
- 
- 492 Naman Agarwal, Rohan Anil, Elad Hazan, Tomer Koren, and Cyril Zhang. Disentangling adaptive
- 
- 493 gradient methods from learning rates.
- arXiv preprint arXiv:2002.11803*
- , 2020.
- 
- 494
- 
- 495 Kwangjun Ahn, Xiang Cheng, Minhak Song, Chulhee Yun, Ali Jadbabaie, and Suvrit Sra. Linear
- 
- 496 attention is (maybe) all you need (to understand transformer optimization), March 2024. URL
- 
- 497
- <http://arxiv.org/abs/2310.01082>
- . arXiv:2310.01082 [cs, math].
- 
- 498
- 
- 499 Jeremy Bernstein, Yu-Xiang Wang, Kamyar Aizzadenesheli, and Anima Anandkumar. signSGD:
- 
- 500 Compressed Optimisation for Non-Convex Problems, August 2018. URL
- <http://arxiv.org/abs/1802.04434>
- . arXiv:1802.04434 [cs, math].
- 
- 501
- 
- 502 Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O’Brien, Eric
- 
- 503 Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al.
- 
- 504 Pythia: A suite for analyzing large language models across training and scaling. In
- ICML, 2023*
- .
- 
- 505
- 
- 506 Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He,
- 
- 507 Connor Leahy, Kyle McDonell, Jason Phang, et al. Gpt-neox-20b: An open-source autoregressive
- 
- 508 language model.
- arXiv preprint arXiv:2204.06745*
- , 2022.
- 
- 509
- 
- 510 Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam
- 
- 511 Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm:
- 
- 512 Scaling language modeling with pathways.
- Journal of Machine Learning Research*
- , 24(240):1–113,
- 
- 513 2023.
- 
- 514
- 
- 515 Enea Monzio Compagnoni, Rustem Islamov, Antonio Orvieto, and Eduard Gorbunov. On the
- 
- 516 interaction of noise, compression role, and adaptivity under
- $(l_0, l_1)$
- smoothness: An sde-based
- 
- 517 approach.
- arXiv preprint arXiv:2506.00181*
- , 2025a.
- 
- 518
- 
- 519 Enea Monzio Compagnoni, Tianlin Liu, Rustem Islamov, Frank Norbert Proske, Antonio Orvieto,
- 
- 520 and Aurelien Lucchi. Adaptive methods through the lens of SDEs: Theoretical insights on the role
- 
- 521 of noise. In
- The Thirteenth International Conference on Learning Representations*
- , 2025b.
- 
- 522
- 
- 523 Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-
- 
- 524 efficient exact attention with io-awareness.
- Advances in neural information processing systems*
- , 35,
- 
- 525 2022.
- 
- 526
- 
- 527 Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas
- 
- 528 Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An
- 
- 529 image is worth 16x16 words: Transformers for image recognition at scale.
- arXiv preprint*
- 
- 530
- arXiv:2010.11929*
- , 2020.
- 
- 531
- 
- 532 Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola,
- 
- 533 Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet
- 
- 534 in 1 hour.
- arXiv preprint arXiv:1706.02677*
- , 2017.
- 
- 535
- 
- 536 Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad
- 
- 537 Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of
- 
- 538 models.
- arXiv preprint arXiv:2407.21783*
- , 2024.
- 
- 539
- 
- 540 Alexander Hägele, Elie Bakouch, Atli Kosson, Loubna Ben Allal, Leandro Von Werra, and Martin
- 
- 541 Jaggi. Scaling Laws and Compute-Optimal Training Beyond Fixed Training Durations, October
- 
- 542 2024. URL
- <http://arxiv.org/abs/2405.18392>
- . arXiv:2405.18392 [cs].
- 
- 543
- 
- 544 Kaiqi Jiang, Dhruv Malik, and Yuanzhi Li. How Does Adaptive Optimization Impact Local Neu-
- 
- 545 ral Network Geometry?, November 2022. URL
- <http://arxiv.org/abs/2211.02254>
- .
- 
- 546 arXiv:2211.02254 [cs].

- 540 Keller Jordan, Yuchen Jin, Vlado Boza, You Jiacheng, Franz Cesista, Laker Newhouse, and Jeremy  
541 Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL <https://kellerjordan.github.io/posts/muon/>.
- 542  
543
- 544 Andrej Karpathy. Nanogpt, 2022.
- 545 Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint*  
546 *arXiv:1412.6980*, 2014.
- 547 Ananya Kumar, Ruoqi Shen, Sébastien Bubeck, and Suriya Gunasekar. How to fine-tune vision  
548 models with sgd. *arXiv preprint arXiv:2211.09359*, 2022.
- 549
- 550 Frederik Kunstner, Jacques Chen, Jonathan Wilder Lavington, and Mark Schmidt. Noise is not the  
551 main factor behind the gap between sgd and adam on transformers, but sign descent might be. In  
552 *ICLR*, 2023.
- 553
- 554 Frederik Kunstner, Robin Yadav, Alan Milligan, Mark Schmidt, and Alberto Bietti. Heavy-Tailed  
555 Class Imbalance and Why Adam Outperforms Gradient Descent on Language Models, July 2024.  
556 URL <http://arxiv.org/abs/2402.19449>. arXiv:2402.19449 [cs, math, stat].
- 557 Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao,  
558 Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint*  
559 *arXiv:2412.19437*, 2024.
- 560
- 561 Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin,  
562 Weixin Xu, Enzhe Lu, Junjie Yan, et al. Muon is scalable for LLM training. *arXiv preprint*  
563 *arXiv:2502.16982*, 2025.
- 564 Tianyi Liu, Zhehui Chen, Enlu Zhou, and Tuo Zhao. A diffusion approximation theory of momentum  
565 stochastic gradient descent in nonconvex optimization. *Stochastic Systems*, 2021.
- 566
- 567 Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv*  
568 *preprint arXiv:1608.03983*, 2016.
- 569 Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.
- 570
- 571 Sadhika Malladi, Kaifeng Lyu, Abhishek Panigrahi, and Sanjeev Arora. On the SDEs and Scaling  
572 Rules for Adaptive Gradient Algorithms, February 2023. URL <http://arxiv.org/abs/2205.10287>. arXiv:2205.10287 [cs].
- 573
- 574 Martin Marek, Sanae Lotfi, Aditya Somasundaram, Andrew Gordon Wilson, and Micah Goldblum.  
575 Small batch size training for language models: When vanilla sgd works, and why gradient  
576 accumulation is wasteful. *arXiv preprint arXiv:2507.07101*, 2025a.
- 577
- 578 Martin Marek, Sanae Lotfi, Aditya Somasundaram, Andrew Gordon Wilson, and Micah Goldblum.  
579 Small batch size training for language models: When vanilla sgd works, and why gradient  
580 accumulation is wasteful, 2025b. URL <https://arxiv.org/abs/2507.07101>.
- 581 GN Mil’shtein. Weak approximation of solutions of systems of stochastic differential equations.  
582 *Theory of Probability & Its Applications*, 30(4):750–766, 1986.
- 583
- 584 Igor Molybog, Peter Albert, Moya Chen, Zachary DeVito, David Esiobu, Naman Goyal, Punit Singh  
585 Koura, Sharan Narang, Andrew Poulton, Ruan Silva, et al. A theory on adam instability in  
586 large-scale machine learning. *arXiv preprint arXiv:2304.09871*, 2023.
- 587
- 588 Toan Q Nguyen and Julian Salazar. Transformers without tears: Improving the normalization of  
589 self-attention. *arXiv preprint arXiv:1910.05895*, 2019.
- 590
- 591 Lorenzo Noci, Sotiris Anagnostidis, Luca Biggio, Antonio Orvieto, Sidak Pal Singh, and Aurelien  
592 Lucchi. Signal propagation in transformers: Theoretical perspectives and the role of rank collapse.  
593 *Advances in Neural Information Processing Systems*, 2022.
- Antonio Orvieto and Robert Gower. In search of adam’s secret sauce. *arXiv preprint*  
*arXiv:2505.21829*, 2025.

- 594 Yan Pan and Yuanzhi Li. Toward Understanding Why Adam Converges Faster Than SGD for  
595 Transformers, May 2023. URL <http://arxiv.org/abs/2306.00204>. arXiv:2306.00204  
596 [cs].  
597
- 598 Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin  
599 Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for  
600 the finest text data at scale. In *The Thirty-eight Conference on Neural Information Processing  
601 Systems Datasets and Benchmarks Track*, 2024. URL [https://openreview.net/forum?  
602 id=n6SCkn2QaG](https://openreview.net/forum?id=n6SCkn2QaG).
- 603 Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language  
604 models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.  
605
- 606 Ishaan Shah, Anthony M Polloreno, Karl Stratos, Philip Monk, Adarsh Chaluvvaraju, Andrew Hojel,  
607 Andrew Ma, Anil Thomas, Ashish Tanwer, Darsh J Shah, et al. Practical efficiency of muon for  
608 pretraining. *arXiv preprint arXiv:2505.02222*, 2025.
- 609 Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.  
610
- 611 Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hes-  
612 tness, and Nolan Dey. SlimPajama: A 627B token cleaned and dedu-  
613 plicated version of RedPajama. [https://www.cerebras.net/blog/  
614 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),  
615 2023. URL <https://huggingface.co/datasets/cerebras/SlimPajama-627B>.
- 616 Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced  
617 transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.  
618
- 619 Akiyoshi Tomihari and Issei Sato. Understanding Why Adam Outperforms SGD: Gradient Het-  
620 erogeneity in Transformers, January 2025. URL <http://arxiv.org/abs/2502.00213>.  
621 arXiv:2502.00213 [cs].
- 622 Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée  
623 Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and  
624 efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- 625 Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz  
626 Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing  
627 systems*, 30, 2017.  
628
- 629 Ben Wang and Aran Komatsuzaki. Gpt-j-6b: A 6 billion parameter autoregressive language model.  
630 2021. URL <https://github.com/kingoflolz/mesh-transformer-jax>, pp. 8, 2022.
- 631 Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nati Srebro, and Benjamin Recht. The marginal  
632 value of adaptive gradient methods in machine learning. *Advances in neural information processing  
633 systems*, 30, 2017.  
634
- 635 Mitchell Wortsman, Peter J Liu, Lechao Xiao, Katie Everett, Alex Alemi, Ben Adlam, John D  
636 Co-Reyes, Izzeddin Gur, Abhishek Kumar, Roman Novak, et al. Small-scale proxies for large-scale  
637 transformer training instabilities. *arXiv preprint arXiv:2309.14322*, 2023.
- 638 Shuo Xie and Zhiyuan Li. Implicit bias of adamw:  $\ell_\infty$ -norm constrained optimization. In *Internation-  
639 al Conference on Machine Learning*, pp. 54488–54510. PMLR, 2024.  
640
- 641 Shuo Xie, Mohamad Amin Mohamadi, and Zhiyuan Li. Adam exploits  $\ell_\infty$ -geometry of loss landscape  
642 via coordinate-wise adaptivity. *arXiv preprint arXiv:2410.08198*, 2024.
- 643 Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang,  
644 Yanyan Lan, Liwei Wang, and Tiejun Liu. On layer normalization in the transformer architecture.  
645 In *International conference on machine learning*, pp. 10524–10533. PMLR, 2020.  
646
- 647 Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural  
Information Processing Systems*, 32, 2019.

648 Guodong Zhang, Lala Li, Zachary Nado, James Martens, Sushant Sachdeva, George E. Dahl,  
649 Christopher J. Shallue, and Roger Grosse. Which Algorithmic Choices Matter at Which Batch  
650 Sizes? Insights From a Noisy Quadratic Model, October 2019. URL <http://arxiv.org/abs/1907.04164>. arXiv:1907.04164 [cs].  
651  
652 Hanlin Zhang, Depen Morwani, Nikhil Vyas, Jingfeng Wu, Difan Zou, Udaya Ghai, Dean Foster,  
653 and Sham Kakade. How Does Critical Batch Size Scale in Pre-training?, February 2025. URL  
654 <http://arxiv.org/abs/2410.21676>. arXiv:2410.21676 [cs].  
655  
656 Jingzhao Zhang, Tianxing He, Suvrit Sra, and Ali Jadbabaie. Why gradient clipping accelerates  
657 training: A theoretical justification for adaptivity, February 2020a. arXiv:1905.11881 [cs, math].  
658  
659 Jingzhao Zhang, Sai Praneeth Karimireddy, Andreas Veit, Seungyeon Kim, Sashank J. Reddi, Sanjiv  
660 Kumar, and Suvrit Sra. Why are Adaptive Methods Good for Attention Models?, October 2020b.  
661 URL <http://arxiv.org/abs/1912.03194>. arXiv:1912.03194 [cs, math].  
662  
663 Yushun Zhang, Congliang Chen, Naichen Shi, Ruoyu Sun, and Zhi-Quan Luo. Adam can converge  
664 without any modification on update rules. *Advances in neural information processing systems*, 35:  
665 28386–28399, 2022.  
666  
667 Yushun Zhang, Congliang Chen, Tian Ding, Ziniu Li, Ruoyu Sun, and Zhi-Quan Luo. Why Trans-  
668 formers Need Adam: A Hessian Perspective, June 2024a. URL <http://arxiv.org/abs/2402.16788>. arXiv:2402.16788 [cs].  
669  
670 Yushun Zhang, Congliang Chen, Ziniu Li, Tian Ding, Chenwei Wu, Yinyu Ye, Zhi-Quan Luo,  
671 and Ruoyu Sun. Adam-mini: Use Fewer Learning Rates To Gain More, July 2024b. URL  
<http://arxiv.org/abs/2406.16793>. arXiv:2406.16793 [cs].  
672  
673 Rosie Zhao, Depen Morwani, David Brandfonbrener, Nikhil Vyas, and Sham Kakade. Deconstructing  
674 What Makes a Good Optimizer for Language Models, July 2024. URL <http://arxiv.org/abs/2407.07972>. arXiv:2407.07972 [cs].  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701

# Appendix

## A FURTHER EXPERIMENTS AND EXPERIMENTAL DETAILS

For pre-training Transformers on Causal Language Modeling, we use a setup that builds upon the nanoGPT (Karpathy, 2022) implementation, augmenting it with Rotational Positional Embedding (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, made popular 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 to  $1e - 5$  of 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) to 25% of dimensions ((Su et al., 2024))
- *Initialization:* the MLP and Attention output weights are initialized with variance  $0.02/\sqrt{2\#\text{layers}}$  (scaling also similar to (Radford et al., 2019)). All other weights (comprising embeddings) are initialized with a standard deviation of 0.02 (Nguyen & Salazar (2019); Wang & Komatsuzaki (2022), Sec. 2.2). Biases are always initialized at zero.
- *Precision:* Mixed precision FP16 enabled.
- *Dropout:* Disabled for both hidden and attention layers (see also Chowdhery et al. (2023)).

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

**Model settings (410 M).** 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

- 756 • *Attention heads*: 16
- 757 • *Hidden size*: 1024
- 758 • *Other settings* as 160M parameters.

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

- 764 • *Layers*: 16 Transformer layers
- 765 • *Attention heads*: 8
- 766 • *Hidden size*: 2048
- 767 • *Other settings* as 160M parameters.

## 771 A.2 HYPERPARAMETER TUNING FOR SECTION 2.3

772  
 773 Combined, the experiments in this section account for full training (at different token budgets) of  
 774 more than 250 language models at different scales and batch sizes. Every reported result is relative to  
 775 the best learning rate in our grid, defined for each setup.

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

- 780 • For Figure 2 and Figure 3 (SlimPajama, 160M), we choose a sequence length of 2048. Inspired  
 781 by the careful tuning of Figure 1, we consider the learning rate grid  $[0.25, 0.5, 1.0]$  for SGD and  
 782  $[0.001, 0.002, 0.004]$  for Adam.
- 783 • For Figure 15 (Fineweb, 160M), we choose a sequence length of 1024. Our learning rate grid  
 784 here is the same as for SlimPajama (previous point). As a sequence length of  $160k$ , given our lack  
 785 of experience with extremely low batch sizes (shorter sequence length), we operate on a slightly  
 786 larger grid:  $[0.0001, 0.0003, 0.001, 0.003]$  for Adam and  $[0.03, 0.1, 0.3, 1]$  for SGD.
- 787 • For Figure 16 (SlimPajama, 250M - 24 layers), we choose a sequence length of 2048 and we also  
 788 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.

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

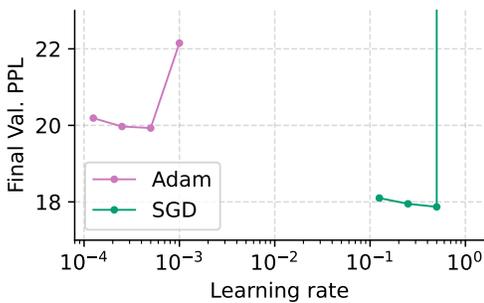
- 793 • **410M model (Figure 4a):** We train with sequence length 2048, for  $500k$  steps on SlimPajama.  
 794 Learning rate grid is  $[1.25e - 4, 2.5e - 4, 5.0e - 4, 1.0e - 3]$  for Adam and  $[0.125, 0.25, 0.5, 1]$   
 795 for SGD. The sweep results are presented in Figure 10a.
- 796 • **1B model (Figure 4b):** We train with sequence length 1024, for 850k steps on Fineweb. Learning  
 797 rate sweep, shown in Figure 10b uses  $[6.25e - 5, 1.25e - 4, 2.5e - 4, 5.0e - 4, 1.0e - 3]$  for Adam  
 798 and  $[0.0625, 0.125, 0.25, 0.5, 1]$  for SGD.

## 801 A.3 HYPERPARAMETER TUNING FOR SECTION 2.4

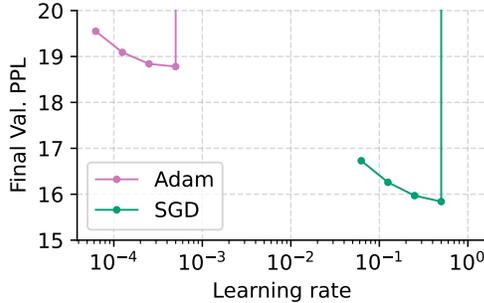
802  
 803 The experimental setup here follows Section A.2. We train a 160M model on SlimPajama, with  
 804 sequence length 2048 and the WSD scheduler. We consider batch sizes 1, 8, and 256 under a token  
 805 budget of 3.2B tokens.

806  
 807 For Adam, we fix  $\beta_1 = 0.9$ , vary  $\beta_2 \in [0.9, 0.99, 0.999, 0.9999, 0.99999]$ , and tune the learning  
 808 rate for each configuration. For SGD, we similarly tune the momentum  $\beta \in [0.9, 0.99, 0.999]$ . The  
 809 learning rate grids are: for Adam  $[1.0e - 5, 3.0e - 5, 1.0e - 4, 3.0e - 4, 1.0e - 3, 3.0e - 3, 1.0e - 2, 3.0e - 2]$  and  $[1.0e - 3, 3.0e - 3, 1.0e - 2, 3.0e - 2, 1.0e - 1, 3.0e - 1, 1.0]$  for SGD.

810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820



(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 10: Learning rate sweep for 410M and 1B models. Trajectories for the optimal learning rate are shown in Figure 4.

823  
824  
825

## B ADDITIONAL RESULTS

826  
827  
828  
829  
830  
831

In addition to Figure 1, we report the validation perplexity for the best-performing  $\beta_1$  and learning rate combination for both Adam and SGD across batch sizes in Table 1. The experimental setting is described in Section 2.2.

832  
833  
834

Table 1: Best validation perplexities and corresponding hyperparameters for Adam and SGD across batch sizes. Results correspond to the sweep shown in Figure 1.

Batch Size	Optimizer	PPL	Hyperparameters
64	Adam	<b>28.77</b>	$\beta_1 = 0.98, \text{ lr} = 1e-3$
	SGD	<b>30.76</b>	$\beta_1 = 0.98, \text{ lr} = 5e-1$
256	Adam	<b>28.20</b>	$\beta_1 = 0.95, \text{ lr} = 2e-3$
	SGD	<b>33.08</b>	$\beta_1 = 0.98, \text{ lr} = 1e+0$
1024	Adam	<b>29.36</b>	$\beta_1 = 0.95, \text{ lr} = 5e-3$
	SGD	<b>65.94</b>	$\beta_1 = 0.95, \text{ lr} = 5e-1$

835  
836  
837  
838  
839  
840  
841  
842  
843

In addition to Figure 3, we report the training perplexity for all other batch sizes in Figure 3. We repeat the experiments from Section 2.3 to verify that our findings generalize to a different dataset and a deeper model.

844  
845  
846  
847

### B.1 CLIPPING ACTS DIFFERENTLY AT DIFFERENT BATCH SIZES

848  
849  
850  
851  
852  
853

By detailed analysis of runs from Figure 1, we observe that gradients are clipped more frequently when training with SGD at large batch sizes, as shown in Figure 11. Additionally, at small batch sizes, SGD performs equally well even without clipping; instead, at large batch sizes, training diverges if clipping is not employed.

854

### B.2 SCALING EXPERIMENTS ACROSS MODEL SIZES AND DATASETS

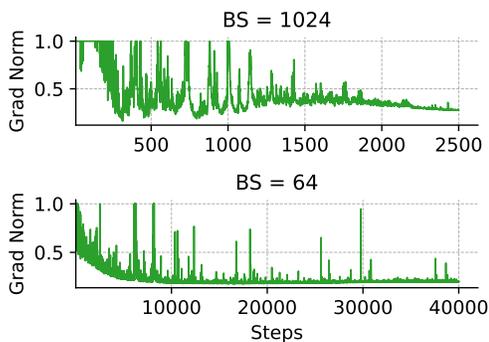
855  
856  
857  
858  
859  
860

We train the same 12-layer Transformer on the Fineweb dataset using SGD with momentum and Adam, tuning the learning rate as explained in Appendix A. Batch sizes vary from 4 to 512, and we use 3 different run lengths (i.e., different token budgets). From Figure 15, 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, consistent with the findings reported earlier.

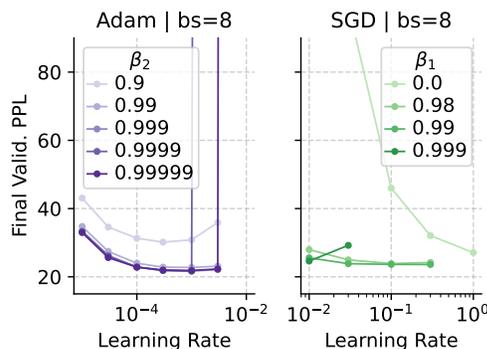
861  
862  
863

In a second experiment, we increase the model depth to 24 layers while keeping all other settings identical to Section 2.3. We vary batch sizes from 4 to 64 and training lengths, and tune the learning rate as explained in Appendix A. As shown in Figure Figure 16, the same pattern holds for a deeper model.

864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875



876 Figure 11: Gradient norm after clipping (thresh-  
877 old 1.0) shows that clipping is more frequent in  
878 large-batch training. The setup for these runs is  
879 the same as Figure 1.

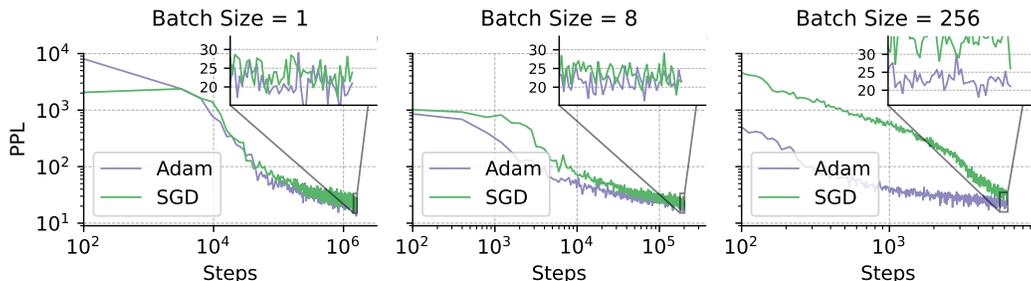


880 Figure 12: Final perplexities for batch size 8,  
881 shown in addition to Figure 5 (main text).

882 **B.3 TUNING ADAM IN SMALL-BATCH SETTINGS**

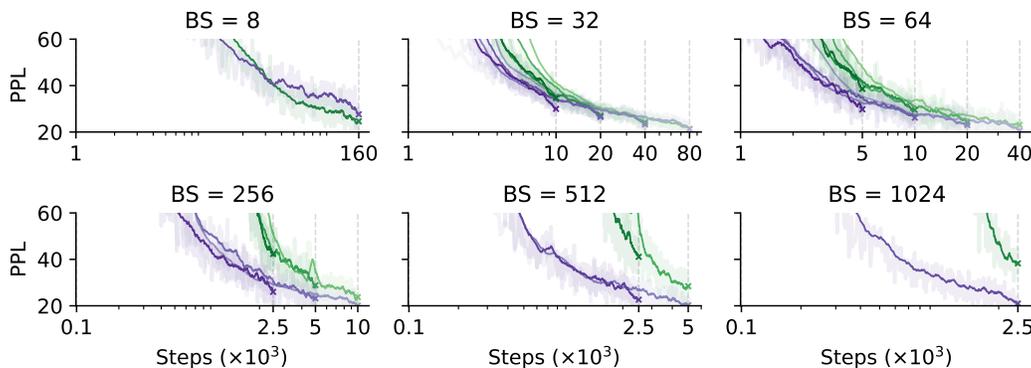
883 In addition to the results from Section 2.4, we show batch size 8 and full training trajectories.  
884 Figure 12 shows the final perplexity for batch size 8 across different  $\beta_2$  values. Figure 13 presents the  
885 full training curves for batch sizes 1, 8, and 256, illustrating convergence dynamics and the relative  
886 performance of Adam and SGD.

887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897



898 Figure 13: Training dynamics for batch sizes 1, 8, and 256, corresponding to the final perplexities  
899 shown in Figure 5 (main text) and Figure 12 (Appendix).

900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913



914 Figure 14: Perplexity during training for SGD (green) and Adam (purple) across different training  
915 lengths for all other batch sizes not shown in Figure 3. Solid lines show the rolling mean of PPL  
916 values; lighter lines show the raw values. As before, the gap decreases the longer we train, and SGD  
917 can eventually outperform Adam.

918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933

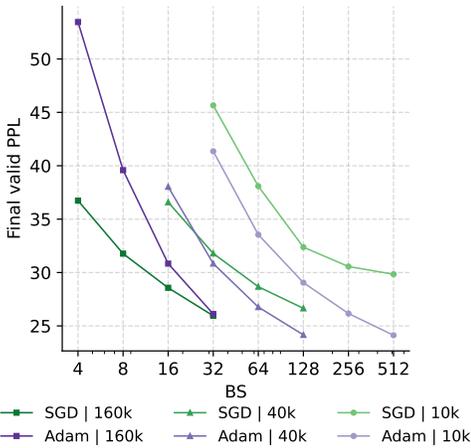


Figure 15: Fineweb dataset, sequence length 2048, 12-layer Transformer.

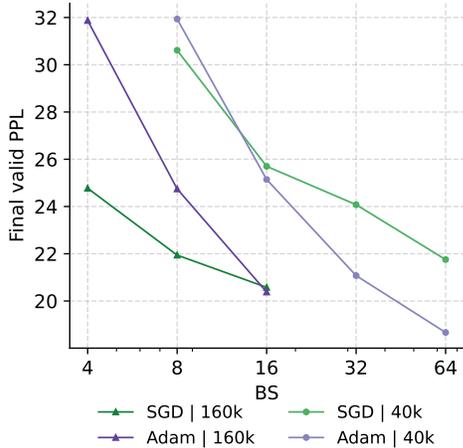


Figure 16: SlimPajama dataset, sequence length 2048, 24-layer Transformer.

B.4 ADDITIONAL RESULTS FOR SECTION 4.1

We provide additional experiments supporting Figure 9. In Figure 17, we present results for a least-squares regression problem on the California Housing dataset from scikit-learn across different batch sizes and learning rates. Consistent with Figure 9, batch size affects the progress of signSGD, while SGD performs similarly across batch sizes.

934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955

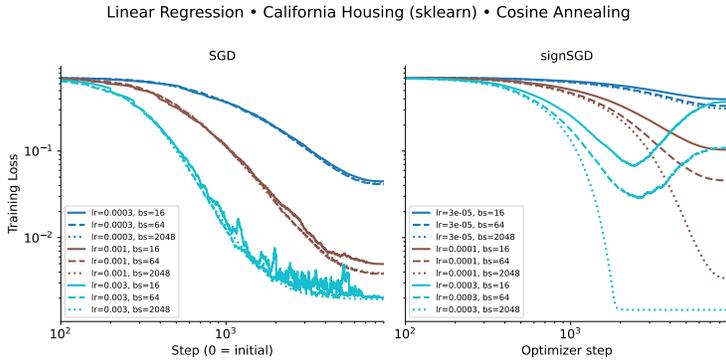


Figure 17: SGD and SignSGD on least-squares regression problem on the California Housing dataset from scikit-learn across different batch sizes and learning rates, supporting Figure 9. Different colors correspond to different learning rates.

C REVISITING PRIOR EXPLANATIONS: DIRECTIONAL SHARPNESS

Pan & Li (2023) introduce directional sharpness to explain the optimizer gap by studying a second-order Taylor expansion of the loss along the update direction. In this view, the first-order term (gradient correlation) measures how well the update aligns with the negative gradient, while the second-order term (directional sharpness) measures curvature along that direction. Making optimization progress requires a strong negative gradient correlation and low directional sharpness. Let  $f$  be a generic loss to optimize and  $x_k$  denote the model parameters at iteration  $k$ , then

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

In Figure 18, we visualize gradient correlation, directional sharpness, and their sum — a second-order approximation of loss change, to indicate progress. As in our previous analysis, we compare two

968  
969  
970  
971

settings with drastic performance differences: batch sizes 64 and 1024 from Section 2.2. In the large-batch setting, SGD shows low gradient correlation and high directional sharpness, resulting in weak or even positive total loss change, as reflected in the sum. In contrast, Adam has higher gradient alignment and lower directional sharpness throughout training. When SGD succeeds, its gradient correlation and directional sharpness closely match Adam’s, producing a negative loss change in the sum. While these metrics align with SGD’s success or failure, they do not directly explain why Adam outperforms SGD, nor why SGD performs well in small-batch regimes.

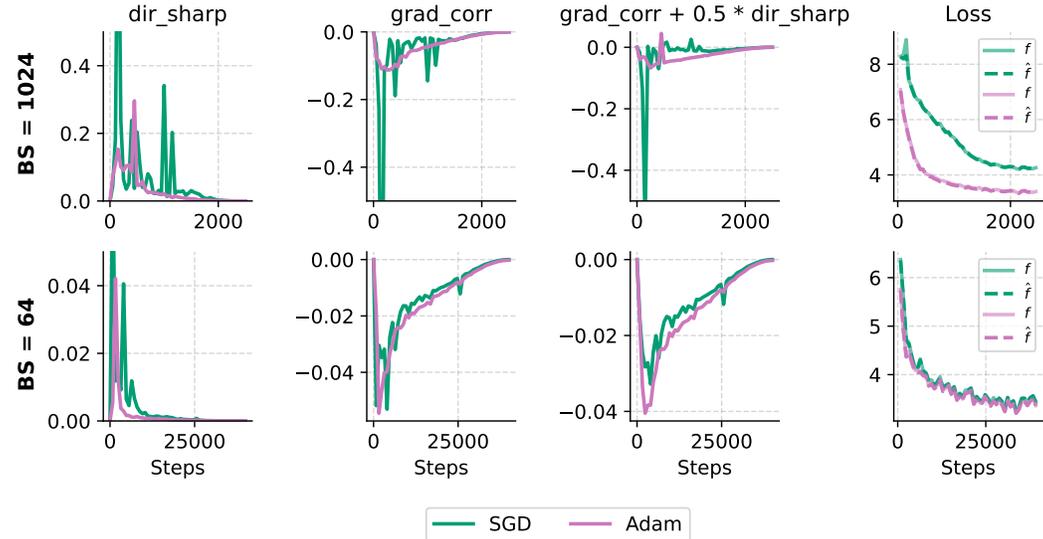


Figure 18: Gradient correlation, directional sharpness, their sum and second-order loss approximation during training, under small- and large-batch settings. Irrespective of the optimizer, good training trajectories have strong negative gradient correlation and low directional sharpness.

As discussed in Section 4, we aim to isolate which component of the SGD update is more problematic in large-batch training. To do that, we focus on the setup from Section 2.2 with batch size 1024 and the optimal  $\beta_1$  and learning rate, and perform analysis using grafting and adaptive clipping. The following subsections provide detailed results for each approach.

## D INSIGHTS FROM GRAFTING AND ADAPTIVE CLIPPING

### D.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 both combinations: 1) SGD magnitude with Adam direction (SGD#Adam), and 2) Adam magnitude with SGD direction (Adam#SGD). We use the optimal  $\beta_1$  from Section 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 19. 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.

### D.2 INSIGHTS FROM ADAPTIVE CLIPPING

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

To investigate this further, we experiment with adaptive clipping, motivated by Pan & Li (2023). As shown in Algorithm 1, we clip the top  $p\%$  of the largest momentum coordinates at each step.

1026  
1027  
1028  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038

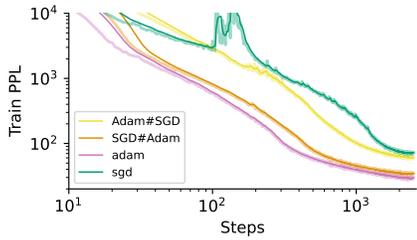


Figure 19: Grafting in large-batch training: using Adam’s direction results in performance closer to Adam, while SGD direction leads to results closer to SGD.

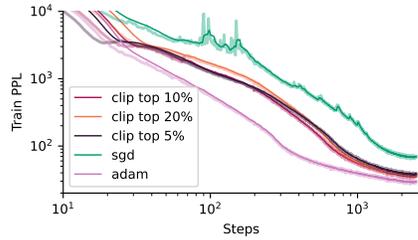


Figure 20: Adaptive clipping with different percentages of clipped coordinates in large-batch training. It improves SGD but still does not fully match Adam.

1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1050  
1051  
1052

We test several values of  $p$  (5, 10, and 20 %). For each value, we keep the optimal  $\beta_1$  from the previous setting and tune the learning rate. Clipping with  $p = 10\%$  performs best, but we observe that performance does not vary much across different values of  $p$ . This method helps reduce the gap between SGD and Adam, as shown in Figure 20. 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 present 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 21, 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.

1053  
1054  
1055  
1056  
1057  
1058  
1059  
1060  
1061  
1062  
1063  
1064  
1065

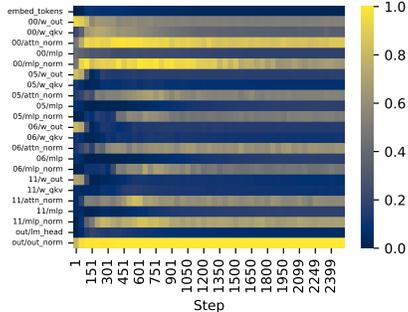


Figure 21: Fraction of clipped momentum coordinates per layer during training, using  $p = 10\%$  adaptive clipping. Only a subset of blocks is shown for clarity, as similar patterns are observed across all blocks. Clipping is present across all parameters, but most pronounced in normalization layers.

1066 **E PROOF SKETCH FOR SECTION 4.1**

1067  
1068  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079

To build intuition for the proof in (Compagnoni et al., 2025b), we begin by studying the quantity

$$\text{sign}(m(x)), \quad m(x) \sim \mathcal{N}(\nabla f(x), \sigma^2/B).$$

That is,  $m(x)$  is an estimate of the gradient, which we assume for simplicity to have Gaussian distribution centered around the full-batch gradient  $\nabla f(x)$ . In expectation, we get (coordinatewise)

$$\begin{aligned} & \mathbb{E}[\text{sign}(m(x))] \\ &= \mathbb{P}[\text{sign}(m(x)) = \text{sign}(\nabla f(x))] \cdot \text{sign}(\nabla f(x)) - \mathbb{P}[\text{sign}(m(x)) \neq \text{sign}(\nabla f(x))] \cdot \text{sign}(\nabla f(x)) \\ &= (2\mathbb{P}[\text{sign}(m(x)) = \text{sign}(\nabla f(x))] - 1) \cdot \text{sign}(\nabla f(x)) \end{aligned}$$

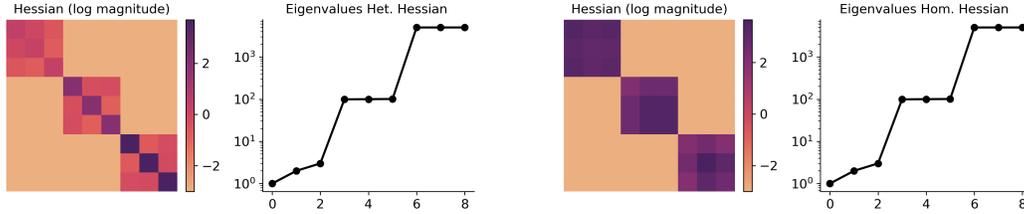
At this point, the erf function comes in. Recall that, if  $Z \sim \mathcal{N}(\mu, \varsigma^2)$ , then if  $\ell > \mu$ , we have  $\mathbb{P}[Z \leq \ell] = \frac{1}{2} + \frac{1}{2} \text{erf}\left(\frac{\ell - \mu}{\sqrt{2\varsigma^2}}\right)$ . Note that this implies, for  $\mu \geq 0$ ,  $\mathbb{P}[Z \geq 0] = \mathbb{P}[\text{sign}(Z) = \text{sign}(\mu)] = \frac{1}{2} + \frac{1}{2} \text{erf}\left(\frac{\mu}{\sqrt{2\varsigma^2}}\right)$ . Hence, if  $\nabla f(x) > 0$  for a specific coordinate,  $\mathbb{P}[\text{sign}(m(x)) =$

1080  $\text{sign}(\nabla f(x)) = \frac{1}{2} + \frac{1}{2} \text{erf}\left(\sqrt{B} \frac{\mu}{\sqrt{2\sigma^2}}\right)$ , relative to that coordinate. By symmetry of this argument  
 1081 for negative  $\nabla f(x)$ , we get exactly the drift term in [Theorem 1](#), discussed above:  
 1082

$$1083 \mathbb{E}[\text{sign}(m(x))] = \text{erf}\left(\sqrt{\frac{B}{2}}(\sigma^2)^{-\frac{1}{2}}\nabla f(x)\right).$$

1084  
 1085  
 1086 Finally, note that Gaussianity is not strictly needed for our insights on batch-size acceleration to hold.  
 1087 As discussed by [Compagnoni et al. \(2025b\)](#) and clear from the argument above on the cumulative  
 1088 distribution, a similar expression can hold even for distributions with heavier tails, such as the  
 1089  $t$ -student (see Corollary C.10 in [Compagnoni et al. \(2025b\)](#)).  
 1090

## 1091 F TOY QUADRATIC EXAMPLE



1092 Figure 22: (left) *Heterogeneous* and (right) *Homogeneous Hessian* considered in [Figure 8](#).

1093  
 1094  
 1095 Our setup is inspired from the results and discussions in [Zhang et al. \(2024a\)](#), and uses the codebase  
 1096 of [Orvieto & Gower \(2025\)](#). We consider the loss  
 1097

$$1098 L(w) = \frac{1}{2} w^\top H w$$

1099 where we construct the Homogeneous and Heterogeneous Hessians using the following procedure:  
 1100

- 1101 • We fix the eigenvalues, equal in both cases, to

$$1102 \text{eig}(H_{\text{hom}}) = \text{eig}(H_{\text{het}}) = \{1, 2, 3, 99, 100, 101, 4998, 4999, 5000\}.$$

- 1103 • We choose both Hessians to be block-diagonal, with blocks of size  $3 \times 3$ . The homogeneous  
 1104 Hessian has eigenvalues of different magnitude in each block, while the Heterogeneous keeps  
 1105 similar magnitudes in each block.  
 1106

$$1107 \begin{aligned} \text{H\_details\_het} &= [[1, 2, 3], [99, 100, 101], [4998, 4999, 5000]] \\ \text{H\_details\_hom} &= [[1, 99, 4998], [2, 100, 4999], [3, 101, 5000]] \end{aligned}$$

- 1108 • For each block, we apply a random rotation to the diagonal matrix of eigenvalues, specific to each  
 1109 block. Each rotation is sampled from the Haar measure by decomposition of a random  $3 \times 3$   
 1110 positive semidefinite matrix  $AA^\top$ , where  $A \in \mathbb{R}^{3 \times 3}$  has i.i.d. Gaussian entries.  
 1111

1112 The result is shown in [Figure 22](#). Learning rates for each method are tuned.

1113 Next, to introduce stochasticity in this setting, we simply take the square root of the Hessian to define  
 1114 a  $9 \times 9$  design matrix  $X$ :

$$1115 H = X^\top X, \quad X = H^{\frac{1}{2}},$$

1116 and subsample a number (the batchsize) of rows of  $X$  at each iteration.

1117 Additional learning rates for [Figure 22](#) are reported in [Figure 23](#).  
 1118  
 1119  
 1120  
 1121  
 1122  
 1123  
 1124  
 1125  
 1126  
 1127  
 1128  
 1129  
 1130  
 1131  
 1132  
 1133

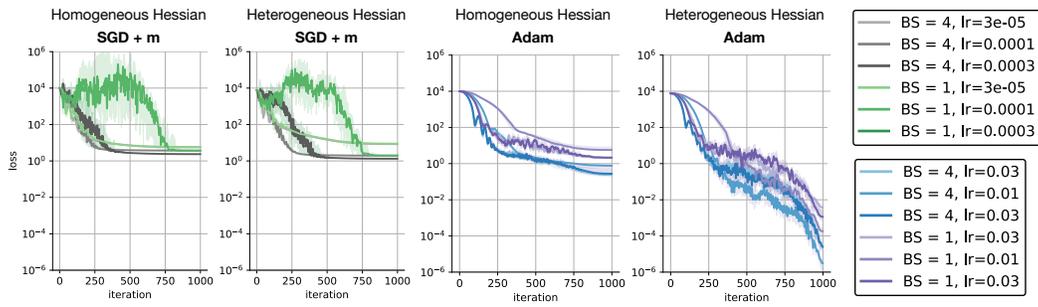


Figure 23: Complement to Figure 8.

## G ALGORITHMIC DETAILS

---

### Algorithm 1 SGD with Adaptive Momentum Clipping

---

**Require:** Initial point  $x_0$ , learning rate  $\eta$ , momentum  $\beta$ , clipping fraction  $p \in (0, 1)$

- 1: **for**  $t = 1$  to  $T$  **do**
  - 2:      $g_t \leftarrow \nabla f(x_t)$
  - 3:      $m_t \leftarrow \beta m_{t-1} + g_t$
  - 4:     Set clipping threshold  $\tau_t$  as the  $(1 - p)$ -quantile of  $|m_t|$
  - 5:      $\hat{m}_t \leftarrow \text{clip}(m_t) = \text{sign}(m_t) \cdot \min(|m_t|, \tau_t)$
  - 6:      $x_{t+1} \leftarrow x_t - \eta \hat{m}_t$
  - 7: **end for**
-