Beyond a *Single Explanation* of the Adam–SGD Gap

Chenxiang Zhang[1,3,4,*] **Rustem Islamov**[2,*] **Enea Monzio Compagnoni**[2]
Jun Pang[1] **Aurelien Lucchi**[2] **Antonio Orvieto**[3,4,5]
[1]University of Luxembourg [2]University of Basel
[3]MPI for Intelligent Systems [4]ELLIS Tübingen [5]Tübingen AI Center

Abstract

Existing explanations for the performance gap between Adam and SGD span data, architecture, and optimization properties. Yet these factors are often studied in isolation, leaving their relative importance unclear. In this work, we revisit these explanations through a rigorous empirical study across vision, language, genomics, and graph tasks, spanning Transformer and classical architectures under a modern training recipe. Our results suggest that no single existing factor can consistently explain the Adam–SGD gap. For instance, the Adam advantage can (1) persist under a uniform vocabulary distribution yet nearly disappear under a heavy-tailed one; (2) reverse in favor of SGD in softmax-attention models; and (3) become larger under soft architectural modifications, e.g., when ReLU is replaced by a GeLU nonlinearity. These results suggest that the gap arises from nontrivial data and architecture interactions, rather than from a single common factor. Yet, we observe a recurring pattern across most settings: a *crossover batch size* at which the relative advantage shifts from SGD to Adam as the batch size scales. The empirical trends are captured by our theoretical model, which also predicts the crossover. Our perspective connects previously proposed explanations and provides practical optimization insights across domains.

https://github.com/orientino/gap

1 Introduction

For over a decade, understanding which optimizer properties drive performance in deep learning has been a central question, particularly in Transformer-based language modeling (LLMs). At the scale of modern LLMs, optimization efficiency is crucial in terms of resource utilization. At present, most industrial and open-source models rely on adaptive methods such as Adam [Kingma and Ba, 2015, Loshchilov and Hutter, 2019]. The success of this optimizer, however, is not limited to language or foundation models. Early benchmarks [Schmidt et al., 2021] report that Adam outperforms SGD (Adam>SGD) in simple autoencoders trained on MNIST and F-MNIST datasets. AlgoPerf [Dahl et al., 2023] also shows an Adam advantage on tasks such as speech recognition with LSTMs and click-through-rate prediction with MLP-based recommender models. Recently, Zucchet and Orvieto [2024] show that large scale is also not a necessary ingredient, as simple recurrent networks on synthetic Gaussian data are faster to optimize with Adam than SGD.

A growing body of work has made several advancements in explaining the Adam–SGD gap by isolating individual factors. While these studies provide valuable insights, their conclusions are often drawn within specific domains or training setups, making it difficult to assess their relative importance or understand how they interact across settings. To navigate this landscape, we group existing explanations into:

*Co-first authors (see §6). Correspondence: chenxiang.zhang@uni.lu, antonio@tue.ellis.eu

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Data	Vocab. Distribution	Vocab. Size v
FineWeb	Zipf	50257
I21K	Zipf	19167
HG38	Uniform	5
ZINC	-	-

Data	Transf. Arch.	Non-Transf. Arch.
FineWeb	GPT	GCNN
I21K	ViT	ResNet
ZINC	GRIT	GAT
HG38	GPT	GDN

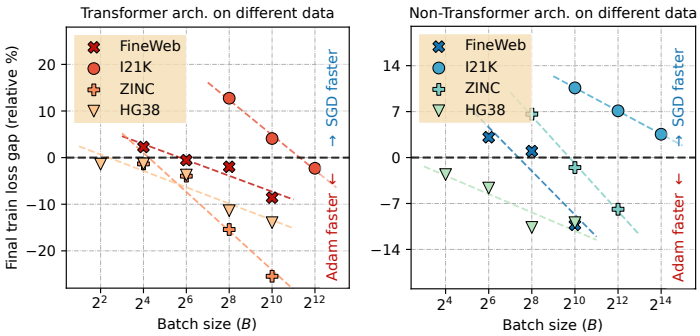


Figure 1: **Optimizer advantage shifts towards Adam as the batch size scales across all setups, with the trend shaped by interactions between data and architecture.** The y-axis reports the relative gap $(\mathcal{L}_{\text{Adam}} - \mathcal{L}_{\text{SGD}})/\mathcal{L}_{\text{Adam}}$ at the end of training, where positive values indicate SGD outperforms Adam. **(left)** Our experimental setup spans four data modalities with different vocabulary distributions and sizes, trained on both Transformer and non-Transformer architectures. **(middle)** Transformer and **(right)** non-Transformer models trained on language, vision, genomics, and graph learning. Across all settings, increasing batch size B consistently shifts the advantage towards Adam; however, the crossover point and transition slope vary by configuration. Rather than attributing the Adam–SGD gap to a single factor, these results point to a *crossover batch size* whose value is jointly shaped by data and architecture properties. In all the experiments, the learning rate and momentum are jointly and extensively tuned, see §3.

- *Data* explanations attribute the gap to data properties, in particular the Zipfian vocabulary distribution of text data, which induces a highly anisotropic optimization landscape [Kunstner et al., 2024, Kunstner and Bach, 2025]. In contrast, SGD has historically been successful on vision with CNNs [Zhou et al., 2020, Dahl et al., 2023], where labels are more uniformly distributed.
- *Architecture* explanations argue that the gap arises from the component heterogeneity of Transformers, including embedding [Kumar et al., 2024], softmax-attention [Noci et al., 2022, Ahn et al., 2024], normalization and head layers [Zhao et al., 2025]. This perspective emphasizes that Adam's advantage is a consequence of the architectural complexity of modern networks.
- *Optimization* explanations attribute the gap to batch size and gradient noise. In the small-batch regime, SGD can match or even outperform Adam in Transformer-based LLMs [Srećković et al., 2025, Marek et al., 2025], suggesting that large-batch training in modern pipelines contributes to Adam's advantage. A separate line of work attributes Adam's advantage to its robustness under heavy-tailed gradient noise [Cutkosky and Mehta, 2021, Fatkhullin et al., 2025, Hübler et al., 2025].

These explanations are mostly motivated by observations from LLM training, each emphasizing a different source of the gap. In this paper, our primary goal is to connect these existing hypotheses.

1.1 Contributions

We answer the following question: *which of the existing factors genuinely drive the Adam–SGD gap, and do these explanations transfer outside the standard Transformer language modeling setting?*

We broaden the Adam–SGD gap study under a unified, modern, and controlled setup. Our experiments cover classic domains such as language and vision, as well as underrepresented ones like genomics and graphs, with both Transformer and classical architectures. Importantly, we adopt the recent architectural advances and a modern optimization recipe while carefully controlling for confounders by jointly sweeping over the learning rate, momentum, and batch size (§3).

We find no single-factor hypothesis that fully explains the gap. Vocabulary imbalance is not required: the gap favoring Adam persists on the genomics dataset HG38 [GRC, 2013], whose vocabulary is small and near-uniform (§4.2). Softmax-attention is likewise not necessary (§4.1), and component heterogeneity is not sufficient: a Vision Transformer trained on ImageNet21K can exhibit the reverse trend, with SGD outperforming Adam. Rather than a sharp Transformer versus non-Transformer effect, the architectural contribution shifts gradually with individual design choices; e.g., replacing ReLU with GeLU already moves the advantage toward Adam (§4.3).

A *crossover batch size* connects the existing single-factor explanations. Building on recent observations that the Adam–SGD gap in language modeling has a batch size dependence [Srećković et al., 2025, Marek et al., 2025], we show that, under a fixed number of training samples, the advantage shifts from SGD to Adam as the batch size scales (Figure 1). Both data properties and architectural design interact with this phase change – including nuances such as the nonlinearity function. Notably, this transition occurs at substantially larger batch sizes for image data regardless of the architecture, consistent with the strong empirical performance of SGD on vision tasks.

A theoretical gap model captures this batch-size dependence. We propose a simple geometry-based model of how the gap changes with batch size. Since all experiments use global gradient-norm clipping, we compare normalized SGD with momentum to a signed-momentum proxy for Adam, SignSGD [Balles and Hennig, 2018, Orvieto and Gower, 2025]. Both methods can be analyzed using the non-Euclidean stochastic-update framework of Kovalev [2025]. Our model can predict different scenarios consistent with the numerical results: either a crossover from SGD to Adam occurs as the batch size increases, or Adam or SGD can dominate for all batch sizes (§5).

The gap behaves qualitatively differently across training regimes. In the classical deep learning setting of overparametrized multi-epoch training, where the optimization problem is solvable, the Adam–SGD gap shrinks as training can approach zero loss. In the modern sub-one-epoch under-parametrized regime of LLMs, the gap persists throughout training (§4.4). This perspective helps to understand and reconcile the conflicting views between classic and modern optimization.

2 Preliminaries / Related work

We introduce and categorize the extensive Adam–SGD gap literature under four groups.

Optimization explanations: gradient noise imbalance and large batch size. Classical literature shows a nontrivial interaction between gradient noise and preconditioning [Kohler and Lucchi, 2017]. Zhang et al. [2019] studied how preconditioning alters the critical batch size, hinting at properties that are related to our study. Yet, Adam cannot be put in direct relation to classical preconditioning [Kunstner et al., 2019, Balles and Hennig, 2018]. Recent works take a different approach, claiming that Adam's advantage comes from its coordinate-wise scaling, which better handles gradient stochasticity than SGD. For example, Zhang et al. [2020] observed that Adam outperforms SGD when the stochastic gradient norm follows a heavy-tailed distribution for Transformers. Adam, being closely related to SignSGD [Bernstein et al., 2018], can naturally clip noise and stabilize updates. Complementarily, Fatkhullin et al. [2025] demonstrate that SGD is provably slower under heavy-tailed noise than adaptive methods, and that normalization can achieve optimal theoretical convergence [Cutkosky and Mehta, 2021, Hübler et al., 2025]. However, Kunstner et al. [2023] challenge this explanation, showing that under deterministic full-batch, Adam retains the advantage. Recently, Srećković et al. [2025], Marek et al. [2025] provide additional evidence: SGD can match, or even outperform, Adam at small batch sizes for Transformers. In support, Compagnoni et al. [2025] provide theoretical evidence that the convergence rate of SignSGD improves with increasing batch size, whereas SGD does not. Our findings extend these works: across a wide range of setups, we show that a small batch size reduces Adam's advantage over SGD, and can even flip the advantage depending on the data and architecture properties.

Data explanations: heavy-tailed class imbalance and vocabulary size. Moving one layer above the noise hypothesis, Kunstner et al. [2024] present heavy-tailed class imbalance as the source of the gap. They showed that SGD reduces the loss of frequent classes faster than that of rare classes. Therefore, the advantage of Adam and SignSGD-like methods may be attributed to a lower sensitivity to class imbalance. Kunstner and Bach [2025] formalize this idea using a linear bigram model with Zipf-distributed data, proving that the gain of adaptive methods over full-batch GD scales with vocabulary size v. While compelling, our results suggest that *vocabulary-based hypotheses are not necessary for observing the Adam–SGD gap.* A nontrivial gap exists for genomics data with near-uniform token distribution and a small vocabulary size ($v = 5$). This setup represents the opposite scenario proposed by the class imbalance hypothesis, yet the gap still exists. Schaipp [2025] recently reported the existence of the Adam–SGD gap in diffusion model training, where no clear notion of class imbalance exists. Similarly, we also show that the gap persists for regression tasks, which have no class structure.

Architecture explanations: Hessian heterogeneity of Transformers. A parallel line of work attributes the advantage of adaptive optimizers to architectural properties. Zhang et al. [2024a] argue that Transformers induce stronger Hessian heterogeneity across layers than CNNs. For the attention layers, Noci et al. [2022] connect the gap to the different training dynamics between Q, K, and V layers. In language modeling, Zhao et al. [2025] emphasize the necessity of tuning the head and normalization layers with Adam. On the other hand, in transfer-learning vision tasks, Kumar et al. [2024] demonstrate that freezing the embedding layer is sufficient to eliminate Adam's advantage. Even the role of the softmax is disputed: Ahn et al. [2024] show that attention without softmax can still reproduce the Adam–SGD gap. The presented literature consistently indicates that architecture matters, but each points to different mechanisms. We aim to clarify which ones are necessary. First, we replace the attention layer of GPT with gated convolution [Dauphin et al., 2017] and observe that the gap persists, suggesting that softmax-attention may not be necessary. Then, we show that architecture design alone can continuously shift the advantage from SGD to Adam, with the choice of GeLU and LayerNorm favoring Adam (Table 2), highlighting that the effect of architecture on the gap interacts with the data properties.

Insights and limitations of optimization benchmarks. Optimization benchmarks [Schmidt et al., 2021, Dahl et al., 2023, Kasimbeg et al., 2025] provide valuable empirical signals, but their conclusions depend on metrics, tuning budgets, schedules, batch-size policies, and hardware. The resulting evidence is mixed: after tuning, Adam-family methods are often strong in machine translation, generative modeling, speech recognition, and recommender systems; SGD-family methods can instead win in classical convolutional image classification, graph prediction, and MRI reconstruction. Notably, batch size is treated differently across benchmarks: Schmidt et al. [2021] fix it, while AlgoPerf [Dahl et al., 2023] treats it as task-specific and tunable as part of the submitted training algorithm, where it interacts with wall-clock performance on the target hardware. As a result, neither isolates batch size as a potential confounder. Here, we instead treat batch size as a central hyperparameter and study how it affects the sign and magnitude of the gap.

3 Methodology

We define the Adam–SGD gap and describe the common experimental details used across the paper.

Problem setup. The main objective is to study the optimization gap between Adam and SGD at the end of training, while controlling the confounders listed in §2. Throughout this work, the focus is on minimizing the training loss rather than improving generalization (although results are reported in §B.6). Unless specified otherwise, we apply the moving average (EMA) to smooth the noisy training loss history and report the last value, which is particularly useful for small batch size training. The gap $\Delta^* = \mathcal{L}^*_{\text{Adam}} - \mathcal{L}^*_{\text{SGD}}$ measures the final loss difference between the best runs across all the η and β sweeps, where

$\Delta^* > 0$ indicates SGD achieves lower final loss than Adam, and $\Delta^* < 0$ indicates otherwise.

Architecture setup. We systematically analyze Transformer-based architectures applied to language, vision, graphs, and genomics data domains. To control for architecture source (§2), we also study classic attention-free architectures. Appendix Table 3 presents an overview of all the architectures and datasets. Unless specified otherwise, for GPT and ViT, we adopt the recent architectural advances such as RoPE [Su et al., 2024], RMSNorm [Zhang and Sennrich, 2019], QKNorm [Touvron et al., 2023], EmbedNorm [Loshchilov et al., 2025], and the ReLU^2 nonlinearity [So et al., 2021, Zhang et al., 2024b]. For GCNN, we modify the GPT architecture by replacing the attention layers with the gated convolution layer [Dauphin et al., 2017]. The remaining architectures use their original implementations. A complete description of the experiments is in §A.

Optimization tuning. The recipe is common across all the setups, unless modifications are specifically reported. We use `bfloat16` mixed precision, 10% linear warmup with a cosine scheduler to 0, and global gradient norm clipping to 1 for Adam and SGD. Weight decay is not applied in the main experiments as it introduces another confounder, while blowing up the hyperparameter search space (yet, for completeness, a subset of the results is repeated with weight decay in §B.1, with consistent findings). When referring to SGD, we always mean SGD with momentum. For SGD, we jointly tune both the learning rate η and the momentum β for each batch size and training configuration. For Adam, jointly tuning (η, β_1, β_2) would lead to a much larger search space, while fixing (β_1, β_2) to

standard defaults such as $(0.9, 0.95)$ [Biderman et al., 2023] can be suboptimal [Orvieto and Gower, 2025, Zhang et al., 2025]. Recent work further shows that the optimal momentum parameters depend on the training budget and batch size [Shulgin et al., 2026, Marek et al., 2025]: Marek et al. [2025] find that the best β_2 increases as batch size decreases, and their Figure 4 suggests a related trend for β_1. This batch-size dependence is also predicted by theoretical analyses [Malladi et al., 2022, Compagnoni et al., 2025], suggesting that both β_1 and β_2 should be adapted when the batch size changes. We therefore tune Adam's momentum, but use the tied parameterization $\beta_1 = \beta_2$ (*which we denote as* β) following Orvieto and Gower [2025]. This keeps a comparable tuning budget to SGD in terms of learning rate and one momentum parameter, while connecting Adam to SignSGD with momentum and to the non-Euclidean analysis in §5. We note, however, that for batch sizes approaching 1, independently tuning (β_1, β_2) can further boost Adam performance, bringing it closer to SGD; we report and explain this ablation in §B.3. This setup goes beyond the tied-momentum parameterization used in our theoretical analysis, and is less directly aligned with one-momentum methods such as SignSGD with momentum and Muon [Jordan et al., 2024], for which the single momentum parameter naturally corresponds to the constraint $\beta_1 = \beta_2$ [Orvieto and Gower, 2025].

4 Experiments

In this section, we present experimental results revisiting existing gap explanations. This includes the role of softmax-attention, vocabulary distribution, architecture heterogeneity, and batch size.

4.1 Softmax-attention hypothesis.

We investigate whether the gap primarily arises from the coupled Q, K, V interactions and normalization induced by softmax-attention [Noci et al., 2022, Ahn et al., 2024]. We replace all the GPT's softmax-attention layers with gated-convolution token mixers (GCNN, Dauphin et al. [2017]), a competitive language modeling approach before the advent of Transformers. Our GCNN architecture, interleaved with pure feedforward layers, mixes tokens only using stacked causal 1D convolutions, as follows: $h(X) = (X * W + b) \otimes \sigma(X * V + c)$.

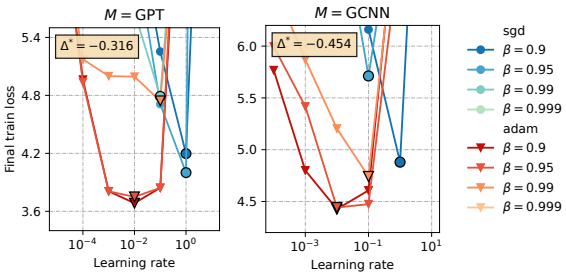


Figure 2: GPT and GCNN of 50M parameters trained on 1B FineWeb tokens at $B = 1024$. GCNN only replaces the softmax-attention of GPT with a gated convolution along the sequence length. The Adam–SGD gap persists in both architectures, suggesting that softmax-attention is not necessary for the gap. Δ^* denotes the gap between the best Adam and SGD runs. The learning rate and momentum are jointly tuned, see §A.8.

Figure 2 shows that GCNN trained on FineWeb still has a wide gap $\Delta^* = -0.454$ despite the removal of the softmax-attention. SGD achieves $\approx 8\%$ higher final loss than Adam for GPT, and $\approx 10\%$ higher for GCNN. The persistence of the gap in GCNN suggests that Adam's advantage is neither specific to Transformers nor to the normalization induced by softmax, but extends to simpler convolution-based sequence models (and to state-space models with linear attention, see Appendix Figure 10). These results complement "Observation 2" by Zhang et al. [2024a], who show that Adam can outperform SGD on MLP-Mixer architectures for vision tasks, where the Hessian exhibits block heterogeneity. Our findings reinforce this claim in a standardized LM setup through optimizer tuning informed by contemporary research[2]. The persistence of the gap without softmax-attention shifts the question from whether softmax-attention is necessary to which architectural components modulate the gap. Indeed, block-level heterogeneity in both architectures can be caused by several other components, which we study later in §4.3.

[2] Zhang et al. [2024a] report default PyTorch settings: $(\beta_1, \beta_2) = (0.9, 0.999)$ for Adam and $\beta = 0.9$ for SGD. This choice may be suboptimal: Zhao et al. [2025], Srećković et al. [2025] suggest a much higher momentum for SGD.

4.2 Heavy-tailed vocabulary imbalance and size hypothesis

In §4.1, we found that the Adam–SGD gap persists across distinct token-mixing strategies. Here, we investigate whether the gap arises from the nature of language data itself. We show that neither a heavy-tailed vocabulary distribution nor a large vocabulary is necessary for the gap to appear.

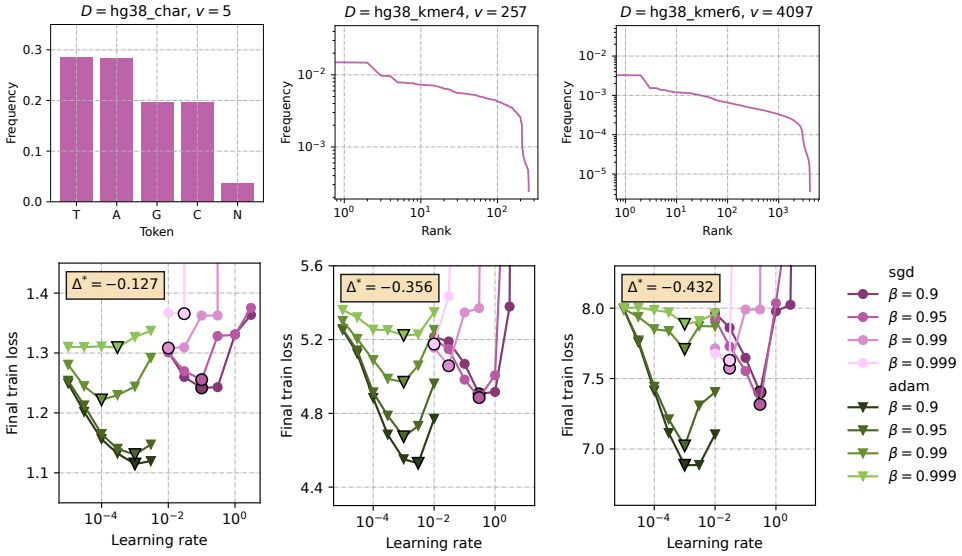


Figure 3: GPT trained on human genomics HG38 data processed with different tokenizers at $B = 256$. *The gap is present regardless of the vocabulary distribution and size.* Under the `char` tokenizer with 2.7B tokens and a near-uniformly distributed small vocabulary ($v = 5$), the gap is nontrivial. For the `kmer` tokenizer, the number of tokens decreases, the vocabulary size grows, and its distribution becomes Zipfian with the gap persisting. The learning rate and momentum are jointly tuned, see §A.6.

Vocabulary imbalance and size. We revisit the role of the heavy-tailed vocabulary [Kunstner et al., 2024] and the vocabulary size [Kunstner and Bach, 2025]. We train a GPT model on human genomics data HG38 [GRC, 2013, Kelley, 2020], which enables fine control of the vocabulary distribution and size v by changing the tokenizer between `char` and `kmer` tokenizer (i.e., single or non-overlapping groups of k nucleotides).

Figure 3 shows that, regardless of the vocabulary distribution and size, the gap persists. Even with a `char`-level tokenizer, where the distribution is almost uniform (excluding the unsequenced nucleotide 'N' as masks) with a small vocabulary $v = 5$, the gap is in favor of Adam $\Delta^* = -0.127$, which translates into $\approx 11\%$ SGD slowdown. For a larger vocabulary $v = 257$ or $v = 4097$, the gap remains as expected. These results show that vocabulary imbalance and large vocabulary size alone do not fully explain the Adam–SGD gap.

Vocabulary in regression tasks. We show that, even in regression tasks where *no class structure is defined*, the gap can still exist. We train GRIT [Ma et al., 2023], a graph-Transformer model, on the real-world molecular dataset ZINC250K [Irwin et al., 2012], where the task is to regress the constrained solubility with heavy atoms as nodes and atomic bonds as edges. The metric used here is MAE (Mean Absolute Error). We follow the graph learning literature and train for 100 epochs while sweeping both the learning rate and momentum, see §A.7. Figure 4 shows that the absolute gap between the best SGD and Adam runs is $\Delta^* = -0.026$, which means that the best run of SGD achieves $\approx 20\%$ worse final loss than Adam. This result complements Schaipp [2025], who reports a similar gap in diffusion tasks, another setting without class structure.

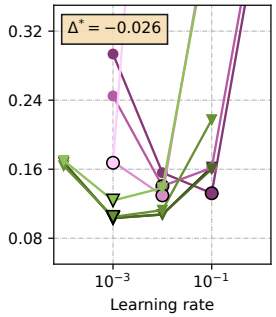
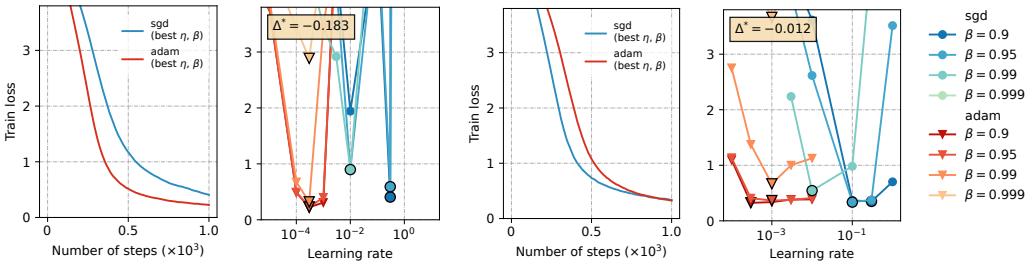


Figure 4: Graph Transformer trained on molecular regression ZINC250K data at $B = 1024$. Adam outperforms SGD in this setting without class structure and imbalance. The learning rate and momentum are tuned, see §A.7.



(a) SimpleViT (Kunstner et al. [2024])

(b) ViT + gradient norm clipping (ours)

Figure 5: ViT trained on imbalanced I1K at $B = 1024$. **(a)** Reproduction of Kunstner et al. [2024]. **(b)** Modern ViT with gradient norm clipping reduces the gap by $\approx 93\%$. Moreover, the training curve of SGD is faster than Adam throughout most of the training, highlighting the joint effect of architecture and optimization recipe. Learning rate and momentum β_1 are tuned, see §A.5.

Vocabulary in vision tasks. Following the same vision setup as Kunstner et al. [2024], we use the heavy-tailed ImageNet1K (HT-I1K) dataset, a subset of I1K with a heavy-tailed class distribution. We train a 12-layer ViT for 300 epochs with constant learning rate, $B = 1024$, fixed $\beta_2 = 0.999$, and gradient norm clipping to 1 (more details in §A.5). Figure 5b shows that the gap is $\Delta = -0.012$, indicating that SGD is only $\approx 3\%$ slower than Adam. Interestingly, the training curve shows that SGD achieves lower loss than Adam throughout most of training, falling behind only towards the end. Note that our result here is in apparent contrast to Kunstner et al. [2024], who show that class imbalance causes Adam to have a clear advantage over SGD in this setup. This difference may reflect the use of gradient norm clipping plus different architectural considerations: we adopt recent design choices (e.g., ReLU^2 instead of GeLU, see §A.2) and a convolutional `stem` instead of a linear layer, which helps to stabilize the training dynamics [Xiao et al., 2021]. These changes together bring the original gap from $\Delta^* = -0.183$ to $\Delta^* = -0.012$, a reduction of $\approx 93\%$ in training loss gap.

The cumulative results presented in this section indicate that class imbalance alone does not consistently produce the gap; rather, its effect is coupled with architecture and training choices. This motivates us to examine architectural components in the next section.

4.3 Dissecting architectural components

Beyond softmax-attention results in §4.1, recent works attribute the Adam–SGD gap to Transformer component heterogeneity. Zhang et al. [2024a] link it to block-level Hessian heterogeneity, while Zhao et al. [2025] localize Adam's advantage to the head and normalization layers.

Embedding, normalization, and head layers. To revisit whether additional architectural heterogeneity contributes to the gap, we broaden the set of experiments of Zhao et al. [2025]. Similarly, we

Table 1: GPT 50M parameters trained on 1B FineWeb tokens with a hybrid optimizer: SGD used for all layers except those in "Adam layers". The Δ measures the difference between the best Adam-only run and each hybrid configuration ($\Delta < 0$ means Adam has lower loss). Each setup jointly tunes the learning rate and momentum at $B = 1024$, see §A.9.

Adam layers	Train loss	Δ	Δ%
-	3.995	−0.312	−8.47%
N (Norm)	3.962	−0.279	−7.57%
E (Embed)	3.891	−0.208	−5.65%
H (Head)	3.780	−0.097	−2.63%
N + H [Zhao et al., 2025]	3.737	−0.054	−1.47%
N + H + E (ours)	3.719	−0.036	−0.98%
All	3.683	0	0%

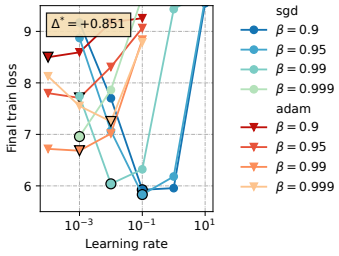


Figure 6: ViT 18M parameters on I21K for one epoch at $B = 256$. Despite component heterogeneity, SGD reaches $\approx 12\%$ lower final loss than Adam, suggesting that component heterogeneity alone does not *always* imply an Adam advantage.

train specific layers such as the head, embedding, normalization, or their combinations with Adam, while optimizing all remaining parameters with SGD. The reported loss is averaged over the last 100 iterations. For each configuration, we *jointly* tune the learning rate η and momentum β, see §A.9. Table 1 reveals a hierarchy among components. Applying Adam only to the normalization layers yields little speedup relative to using SGD for all layers. In contrast, Adam for the embedding layer reduces the gap by 33%, extending the vision results from Kumar et al. [2024]. Applying Adam to the output head yields an even larger improvement, reducing the gap by 69%, as in [Zhao et al., 2025]. Combining Adam on several of these layers can narrow the gap further, but does not eliminate it. These findings indicate that modern architecture design contributes to the gap in language modeling. Yet, component heterogeneity alone does not necessarily imply Adam>SGD. Figure 6 shows that a ViT architecture trained on ImageNet21K (I21K) exhibits the reverse trend, SGD>Adam, with a gap of $\Delta^* = +0.851$, or $\approx 12\%$ in favor of SGD (see §A.4). While adaptivity in the head and embedding layer narrows the gap, other blocks can also play a role. To understand these effects, we next examine how progressive architectural changes shift the optimizer preference.

From SGD to Adam via architecture interpolation. We next test if architectural design can shift optimizer preference. Training on I21K, we progressively introduce ConvNext-style [Liu et al., 2022] design choices, which are inspired by ViTs, to ResNet50 [He et al., 2016], demonstrating that a few simple architecture changes can shift the gap from SGD to Adam advantage. The following three architecture choices are sufficient: (1) BatchNorm $\rightarrow$ LayerNorm; (2) ReLU $\rightarrow$ GeLU; (3) Dense $\rightarrow$ Depthwise convolution. Further experimental details are reported in §A.10.

Table 2: Architecture interpolation from ResNet50 to ConvNext trained on I21K at $B = 1024$. Three architecture choices (L: LayerNorm, G: GeLU, and D: Depthwise) are identified to shift the advantage from SGD to Adam, and are progressively stacked on top of one another. The Δ measures the difference between the best run trained with SGD and Adam ($\Delta < 0$ means Adam has lower loss). Each setup tunes the learning rate and momentum, see §A.10.

	Train loss			
Arch.	**Adam**	**SGD**	**Δ**	**Δ%**
ResNet50	6.133	5.337	+0.797	+13.00%
+ L	7.725	7.162	+0.563	+7.29%
+ L + G	6.275	6.306	−0.031	−0.49%
+ L + G + D	6.071	6.791	−0.720	−11.86%
ConvNeXt	5.955	6.880	−0.925	−15.53%

Table 2 shows the gap for one-epoch training on I21K. Starting from the ResNet50, SGD is $+13.00\%$ faster than Adam. Stacking each architecture change on top of another, the gap Δ shifts from SGD to Adam advantage as follows: (1) + LayerNorm= $+7.29\%$; (2) + GeLU= -0.49%; (3) + Depthwise convolution= -11.86%. Finally, the original ConvNext has a $\Delta = -15.53\%$. This interpolation highlights how the architectural design is coupled with the optimizer choice (the effects of each independent modification is reported in §B.7). While architecture can shift the advantage between optimizers, it does not determine a universal winner. This motivates our next step, beyond data and architecture.

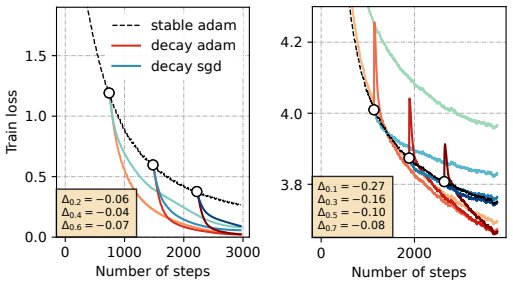
4.4 Batch size and the number of steps

Previous sections found that the gap is not fully explained by single-factor explanations. We now connect prior results from yet another angle: *batch size*, and its relation with data and architecture.

Batch size. In Transformer language modeling (LM), recent works have shown that the gap nearly vanishes at small batch sizes [Marek et al., 2025] and can even reverse to an SGD advantage [Srećković et al., 2025]. This seems yet another explanation. Does it connect with previous data and architecture hypotheses? Does the same finding transfer to other domains? We answer these questions by broadening the batch size studies, including additional data domains (vision, graphs, and genomics) and classic non-Transformer architectures. Figure 1 summarizes the results across a wide range of experimental setups at different batch sizes B. The SGD to Adam shift presented in prior works generalizes across all our setups, though the rate at which the advantage shifts varies by configuration. For example, ViT trained on I21K has a steeper slope compared to GPT on FineWeb or HG38, with Graph Transformer (GRIT) sitting in between. Similar trends hold for non-Transformer architectures such as GCNN, ResNet, and GAT trained on the same datasets. Interestingly, models trained on I21K have the largest *crossover batch size* across all the settings, regardless of the architecture family. This is also consistent with the strong empirical performance of SGD in vision tasks compared to language modeling [Kasimbeg et al., 2025]. *These results suggest that both data and architecture determine how rapidly, and at what batch size, the advantage shifts from SGD to Adam.*

A natural question arises from the previous finding, which sweeps the batch size under a fixed number of training samples. This implies that a small B takes more steps than a large one. Is the number of steps a confounder? Srećković et al. [2025] show that under a simple Gaussian-noisy quadratic model, the early training phase of SGD is B-independent and driven mainly by the number of steps, whereas SignSGD gains a $\sqrt{B}$ speedup factor up to the critical batch size [Compagnoni et al., 2025]. This suggests that adaptive methods' advantage may be isolated to the early training phase.

We revisit this hypothesis, showing that the gap is large in the modern underparametrized regime (one-epoch training), as opposed to the classical overparametrized setting (multi-epoch), where SGD can reach Adam since optimization can reach the minimizer [Belkin et al., 2019].



(a) Overparametrized (b) Underparametrized

Figure 7: GPT 50M parameters trained on (a) 10M TinyStories tokens for 80 epochs and (b) 1B FineWeb tokens for 1 epoch at $B = 1024$. At different stages of training, we isolate a portion of the early training phase by branching Adam/SGD from a shared constant learning rate trajectory (`stable`). **(a)** For overparametrization, both branches converge to low loss as the problem is solvable with more epochs, while **(b)** for underparametrization, SGD lags behind. For every branch, the learning rate and momentum are jointly tuned, see §A.11.

Early training phase. In classical optimization, overparameterization refers to the interpolation regime: the model is expressive enough to fit the training set perfectly [Vaswani et al., 2020, Loizou et al., 2021]. We adopt the term in an operational sense to distinguish between LMs trained for many epochs on a small dataset (overparametrized) from those trained for one epoch on a large dataset (underparametrized). Under this view, with step count and batch size fixed, "earliness" in training is inversely proportional to the number of steps performed. We design an experiment to isolate the early training phase and measure the gap in both regimes: (a) an *overparametrized* setup, training on 10M TinyStories tokens for multiple epochs until train loss ≈ 0; and (b) an *underparametrized* setup, training on 1B FineWeb tokens for one epoch. Following the WSD scheduler [Hu et al., 2024], we first train a shared trajectory `stable_adam` with a constant learning rate, saving multiple checkpoints along training to isolate different stages of the early phase. From each checkpoint, we branch and continue training with either Adam or SGD with a linear decay schedule to 0. For a fair comparison, the momentum state is reset at each branch point, with η and β extensively tuned, see §A.11.

Figure 7 shows the Adam–SGD gap in the overparametrized setting, where `decay_sgd` is slower but reaches loss ≈ 0, similar to Adam. For underparametrization, `decay_sgd` can only approximately match the `stable_adam` curve, while `decay_adam` consistently reaches a lower loss. These two scenarios present a striking contrast that bridges the classic overparametrized literature and the modern literature dominated by Transformer-based LMs. This experiment suggests that the early training hypothesis is better suited to describe an overparametrized setup. In Appendix §B.2, we ablate and isolate the number of steps, showing that under fixed number of steps, larger B still favors Adam over SGD.

5 Theoretical model of the gap

We consider the stochastic optimization problem $\min_{x\in\mathbb{R}^d}\left[f(x) = \mathbb{E}_{\xi\sim\mathcal{D}}[f(x,\xi)]\right]$, and compare NSGD with momentum and SignSGD with momentum in solving this problem. These methods serve as theoretical proxies for studying clipped SGD and Adam, respectively. Here, we present the main theoretical claims, while deferring a detailed discussion to §C.

Let U_2 and U_∞ denote the convergence bounds for NSGD and SignSGD from Kovalev [2025], respectively. We define the surrogate gap as $\Delta_U = U_\infty - U_2$, which expands to

$$\Delta_U = \frac{(\sigma^2\delta_0)^{1/4}\left((L_\infty d)^{1/4} - L_2^{1/4}\right)}{T^{1/4}} + \frac{\sigma^2}{\sqrt{\delta_0 TB}}\left(\frac{d}{\sqrt{L_\infty}} - \frac{1}{\sqrt{L_2}}\right) + \frac{\delta_0^{3/4}B}{\sqrt{\sigma}T^{3/4}}\left(\frac{L_\infty^{3/4}}{d^{1/4}} - L_2^{3/4}\right).$$

Thus, $\Delta_U > 0$ means that $U_2 < U_\infty$, so NSGD surrogate is smaller, while $\Delta_U < 0$ that SignSGD surrogate is smaller. Let $r = L_\infty / L_2$ be the geometry smoothness ratio, then we always have the relation $1 \leq r \leq d$, since $L_2 \leq L_\infty \leq dL_2$. Depending on r, the surrogate gap Δ_U falls into two possible regimes, summarized in Figure 8 (see §C for the derivation).

Regime	Condition	Better surrogate across B
I	$1 < r < d^{1/3}$	SignSGD everywhere, or NSGD to SignSGD *crossover*
II	$d^{1/3} < r < d$	NSGD everywhere

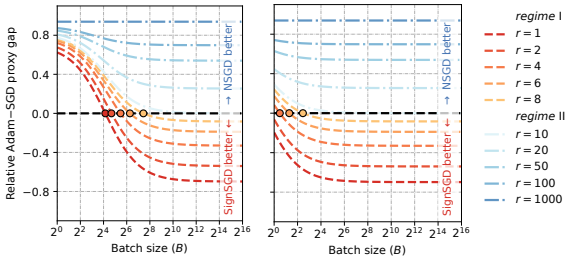


Figure 8: Theoretical surrogate model for the Adam–SGD gap. **(top)** Two regimes as a function of $r = L_\infty / L_2$, the geometry ratio. **(bottom)** Relative surrogate gap $(U_\infty(B) - U_2(B)) \,/\, (U_\infty(B) + U_2(B))$ where U_∞ and U_2 are surrogates for SignSGD and NSGD. *Regime I* show a crossover from NSGD to SignSGD in the left panel and favor SignSGD for all B in the right panel; *Regime II* consistently favor NSGD for all B. Circles mark the crossover points. The problem constants are fixed at $d = 10^3$, $T = 10^6$, $\delta_0 = 10$, $\sigma = 0.1$, $L_2 = 1$ (left), and $L_2 = 1000$ (right).

In *Regime I*, the derivative of the gap Δ_U with respect to B is always negative. Therefore, two outcomes are possible. If $\Delta_U(1) > 0$, NSGD achieves the lower bound at small batch sizes, but increasing B eventually produces a crossover beyond which SignSGD is favored. If $\Delta_U(1) < 0$, SignSGD achieves the lower bound already at $B = 1$ and remains favored for all Bs. *In Regime II*, the gap $\Delta_U > 0$ for all Bs, indicating that NSGD always achieves the lower convergence bound. The ratio r determines which regime applies, while the remaining problem-dependent quantities T, σ, δ_0 determine whether Regime I exhibits a crossover or favors SignSGD at different Bs. This predicts that the relative performance of Adam and SGD depends jointly on the optimization geometry, dataset, and gradient noise characteristics.

Connecting to our experiments in Figure 1, most setups qualitatively resemble Regime I, exhibiting a crossover from SGD to Adam as B increases. The remaining cases are also present: for GPT trained on HG38, Adam outperforms SGD from $B = 1$, corresponding to Regime I without a crossover (see §B.5); for ResNet50 on I21K, SGD remains superior even at $B = 65536$, corresponding to Regime II (see §B.2). Thus, our experiments recover all regimes predicted by the theoretical model.

6 Discussion

We presented a broad empirical study on the Adam–SGD gap, pushing beyond the typical Transformer-based LM setup while carefully controlling for confounders. We revisited the existing single-factor gap explanations, while offering a unified perspective centered on a *crossover batch size* shaped by both data and architecture properties. We complement these empirical findings with a theoretical model that captures how optimization geometry, gradient noise, and batch size jointly affect the crossover. Lastly, we consistently observe that Adam is more stable under our joint hyperparameter sweeps (Figures 9 to 14), leading to a consistent practical advantage.

Looking forward, our results show that every dataset-architecture pair has a different *crossover* slope and point (Figure 1). This perspective suggests several follow-up questions: What data properties beyond vocabulary imbalance drive the Adam–SGD gap? How do specific architectural choices modulate the crossover point? And can this crossover inform the design of novel optimizers?

Limitations. Although the Adam–SGD gap is a central question in modern large-scale optimization, recent practice has also reported promising results for newer optimizers, including Muon [Jordan et al., 2024], Scion [Pethick et al., 2025], and SOAP [Vyas et al., 2025]. Evaluating these methods in our setting is an important future direction; the literature is still evolving, with large-scale reports [Essential AI et al., 2025, Kimi Team, 2025, DeepSeek-AI, 2026] using distinct variants of Muon while the algorithm itself continues to be refined theoretically [Amsel et al., 2026]. Notably, Marek et al. [2025] already show that Muon is effective in the large batch regime, suggesting that trends and conclusions in this work may generalize. Lastly, the numerical experiments depend on the coverage of the hyperparameter grid, and the choice of the datasets and architectures.

Acknowledgments

Chenxiang Zhang acknowledges the financial support of the University of Luxembourg. Rustem Islamov, Enea Monzio Compagnoni, and Aurelien Lucchi acknowledge the financial support of the Swiss National Science Foundation, SNSF grant No 207392. Antonio Orvieto acknowledges the financial support of the Hector Foundation and the AI2050 Early Career Fellowship from Schmidt Sciences. We acknowledge computing resources provided by MPI-IS, MeluXina[3], and sciCORE[4].

Author contributions

Chenxiang Zhang led the project, contributing to the experiments, infrastructure, figures, framing, direction, and writing. Rustem Islamov contributed to the experiments, framing, direction, writing, and theoretical result. Enea Monzio Compagnoni and Jun Pang contributed to the framing and writing. Aurelien Lucchi contributed to the framing, direction, and writing. Antonio Orvieto is the lead advisor, contributing substantially to the framing, direction, writing, and theoretical result.

References

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. (cited on pages 2, 4, and 5)

Noah Amsel, David Persson, Christopher Musco, and Robert M. Gower. The polar express: Optimal matrix sign methods and their application to the Muon algorithm. In *International Conference on Learning Representations*, 2026. (cited on page 10)

Lukas Balles and Philipp Hennig. Dissecting adam: The sign, magnitude and variance of stochastic gradients. In *International Conference on Machine Learning*, pages 404–413, 2018. (cited on page 3)

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019. (cited on page 9)

Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signSGD: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*, volume 80, pages 560–569, 2018. (cited on pages 3 and 39)

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, Usvsn Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, volume 202, pages 2397–2430, 2023. (cited on page 5)

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 *International Conference on Learning Representations*, 2025. (cited on pages 3, 5, and 9)

Ashok Cutkosky and Harsh Mehta. High-probability bounds for non-convex stochastic optimization with heavy tails. *Advances in Neural Information Processing Systems*, 34:4883–4895, 2021. (cited on pages 2 and 3)

George E. Dahl, Frank Schneider, Zachary Nado, Naman Agarwal, Chandramouli Shama Sastry, Philipp Hennig, Sourabh Medapati, Runa Eschenhagen, Priya Kasimbeg, Daniel Suo, Juhan Bae, Justin Gilmer, Abel L. Peirson, Bilal Khan, Rohan Anil, Mike Rabbat, Shankar Krishnan, Daniel Snider, Ehsan Amid, Kongtao Chen, Chris J. Maddison, Rakshith Vasudev, Michal Badura, Ankush Garg, and Peter Mattson. Benchmarking neural network training algorithms. *arXiv preprint arXiv:2306.07179*, 2023. (cited on pages 1, 2, and 4)

[3] https://www.luxprovide.lu/meluxina/
[4] http://scicore.unibas.ch/

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *International Conference on Machine Learning*, volume 70, pages 933–941, 2017. (cited on pages 4, 5, and 17)

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence. Technical report, DeepSeek-AI, 2026. URL https://huggingface.co/deepseek-ai/DeepSeek-V4-Pro. Preview release. (cited on page 10)

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition*, pages 248–255, 2009. (cited on page 17)

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. (cited on pages 17 and 18)

Ronen Eldan and Yuanzhi Li. Tinystories: How small can language models be and still speak coherent english? *arXiv preprint arXiv:2305.07759*, 2023. (cited on page 17)

Essential AI, Ishaan Shah, Anthony M. Polloreno, Karl Stratos, Philip Monk, Adarsh Chaluvaraju, Andrew Hojel, Andrew Ma, Anil Thomas, Ashish Tanwer, Darsh J. Shah, Khoi Nguyen, Kurt Smith, Michael Callahan, Michael Pust, Mohit Parmar, Peter Rushton, Platon Mazarakis, Ritvik Kapila, Saurabh Srivastava, Somanshu Singla, Tim Romanski, Yash Vanjani, and Ashish Vaswani. Practical efficiency of Muon for pretraining. *arXiv preprint arXiv:2505.02222*, 2025. (cited on page 10)

Ilyas Fatkhullin, Florian Hübler, and Guanghui Lan. Can sgd handle heavy-tailed noise? *arXiv preprint arXiv:2508.04860*, 2025. (cited on pages 2 and 3)

GRC. Genome Reference Consortium Human Build 38 (GRCh38). NCBI Assembly, 2013. URL https://www.ncbi.nlm.nih.gov/assembly/88331. (cited on pages 2, 6, and 17)

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition*, pages 770–778, 2016. (cited on pages 8 and 17)

Byeongho Heo, Song Park, Dongyoon Han, and Sangdoo Yun. Rotary position embedding for vision transformer. In *European Conference on Computer Vision*, pages 289–305, 2024. (cited on page 18)

Shengding Hu, Yuge Tu, Xu Han, Ganqu Cui, Chaoqun He, Weilin Zhao, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, et al. Minicpm: Unveiling the potential of small language models with scalable training strategies. In *First Conference on Language Modeling*, 2024. (cited on page 9)

Florian Hübler, Ilyas Fatkhullin, and Niao He. From gradient clipping to normalization for heavy tailed sgd. In *International Conference on Artificial Intelligence and Statistics*, pages 2413–2421, 2025. (cited on pages 2 and 3)

John J. Irwin, Teague Sterling, Michael M. Mysinger, Erin S. Bolstad, and Ryan G. Coleman. ZINC: A free tool to discover chemistry for biology. *Journal of Chemical Information and Modeling*, 52 (7):1757–1768, 2012. (cited on pages 6 and 17)

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. In *International Conference on Machine Learning*, 2026. (cited on page 39)

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/. (cited on pages 5 and 10)

Andrej Karpathy. nanoGPT, 2022. URL https://github.com/karpathy/nanoGPT. GitHub repository. (cited on page 17)

Priya Kasimbeg, Frank Schneider, Runa Eschenhagen, Juhan Bae, Chandramouli Shama Sastry, Mark Saroufim, Boyuan Feng, Less Wright, Edward Z. Yang, Zachary Nado, Sourabh Medapati, Philipp Hennig, Michael Rabbat, and George E. Dahl. Accelerating neural network training: An analysis of the AlgoPerf competition. In *International Conference on Learning Representations*, 2025. (cited on pages 4 and 8)

David R. Kelley. Cross-species regulatory sequence activity prediction. *PLOS Computational Biology*, 16(7):e1008050, 2020. (cited on page 6)

Kimi Team. Kimi K2: Open agentic intelligence, 2025. Technical report. (cited on page 10)

Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. (cited on page 1)

Jonas Moritz Kohler and Aurelien Lucchi. Sub-sampled cubic regularization for non-convex optimization. In *International Conference on Machine Learning*, pages 1895–1904, 2017. (cited on page 3)

Dmitry Kovalev. Understanding gradient orthogonalization for deep learning via non-euclidean trust-region optimization. *arXiv preprint arXiv:2503.12645*, 2025. (cited on pages 3, 9, and 39)

Ananya Kumar, Ruoqi Shen, Sébastien Bubeck, and Suriya Gunasekar. How to fine-tune vision models with SGD. In *International Conference on Learning Representations*, 2024. (cited on pages 2, 4, and 8)

Frederik Kunstner and Francis Bach. Scaling laws for gradient descent and sign descent for linear bigram models under zipf's law. In *Advances in Neural Information Processing Systems*, volume 38, 2025. (cited on pages 2, 3, 6, and 41)

Frederik Kunstner, Philipp Hennig, and Lukas Balles. Limitations of the empirical fisher approximation for natural gradient descent. In *Advances in Neural Information Processing Systems*, volume 32, 2019. (cited on page 3)

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 *International Conference on Learning Representations*, 2023. (cited on page 3)

Frederik Kunstner, Alan Milligan, Robin Yadav, Mark Schmidt, and Alberto Bietti. Heavy-tailed class imbalance and why Adam outperforms gradient descent on language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024. (cited on pages 2, 3, 6, 7, 17, and 22)

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Computer Vision and Pattern Recognition*, pages 11976–11986, 2022. (cited on pages 8, 17, and 27)

Nicolas Loizou, Sharan Vaswani, Issam Hadj Laradji, and Simon Lacoste-Julien. Stochastic polyak step-size for sgd: An adaptive learning rate for fast convergence. In *International Conference on Artificial Intelligence and Statistics*, pages 1306–1314, 2021. (cited on page 9)

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. (cited on page 1)

Ilya Loshchilov, Cheng-Ping Hsieh, Simeng Sun, and Boris Ginsburg. nGPT: Normalized transformer with representation learning on the hypersphere. In *International Conference on Learning Representations*, 2025. (cited on pages 4 and 18)

Liheng Ma, Chen Lin, Derek Lim, Adriana Romero-Soriano, Puneet K. Dokania, Mark Coates, Philip Torr, and Ser-Nam Lim. Graph inductive biases in transformers without message passing. In *International Conference on Machine Learning*, volume 202, pages 23321–23337, 2023. (cited on pages 6, 17, and 18)

Sadhika Malladi, Kaifeng Lyu, Abhishek Panigrahi, and Sanjeev Arora. On the sdes and scaling rules for adaptive gradient algorithms. *Advances in Neural Information Processing Systems*, 35: 7697–7711, 2022. (cited on page 5)

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. In *Advances in Neural Information Processing Systems*, volume 38, 2025. (cited on pages 2, 3, 5, 8, and 10)

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. In *Advances in Neural Information Processing Systems*, volume 35, pages 27198–27211, 2022. (cited on pages 2, 4, and 5)

Antonio Orvieto and Robert M. Gower. In search of Adam's secret sauce. In *Advances in Neural Information Processing Systems*, volume 38, 2025. (cited on pages 3, 5, 33, and 39)

Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems*, volume 37, pages 30811–30849, 2024. (cited on page 17)

Thomas Pethick, Wanyun Xie, Kimon Antonakopoulos, Zhenyu Zhu, Antonio Silveti-Falls, and Volkan Cevher. Training deep learning models with norm-constrained LMOs. In *International Conference on Machine Learning*, volume 267, pages 49069–49104, 2025. (cited on page 10)

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf. (cited on page 17)

Fabian Schaipp. Optimization benchmark for diffusion models on dynamical systems. In *EurIPS 2025 Workshop on Principles of Generative Modeling (PriGM)*, 2025. (cited on pages 3 and 6)

Robin M. Schmidt, Frank Schneider, and Philipp Hennig. Descending through a crowded valley - benchmarking deep learning optimizers. In *International Conference on Machine Learning*, volume 139, pages 9367–9376, 2021. (cited on pages 1 and 4)

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. (cited on pages 5, 39, and 40)

David So, Wojciech Mańke, Hanxiao Liu, Zihang Dai, Noam Shazeer, and Quoc V. Le. Searching for efficient transformers for language modeling. In *Advances in Neural Information Processing Systems*, volume 34, pages 6010–6022, 2021. (cited on pages 4 and 18)

Teodora Srećković, Jonas Geiping, and Antonio Orvieto. Is your batch size the problem? revisiting the Adam-SGD gap in language modeling. In *3rd Workshop on High-dimensional Learning Dynamics (HiLD) at ICML*, 2025. (cited on pages 2, 3, 5, 8, and 9)

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. (cited on pages 4 and 17)

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and efficient foundation language models, 2023. (cited on pages 4, 17, and 18)

Sharan Vaswani, Issam Laradji, Frederik Kunstner, Si Yi Meng, Mark Schmidt, and Simon Lacoste-Julien. Adaptive gradient methods converge faster with over-parameterization (but you should do a line-search). *arXiv preprint arXiv:2006.06835*, 2020. (cited on page 9)

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. (cited on pages 17 and 19)

Nikhil Vyas, Depen Morwani, Rosie Zhao, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham M. Kakade. SOAP: Improving and stabilizing shampoo using Adam for language modeling. In *International Conference on Learning Representations*, 2025. (cited on page 10)

Mitchell Wortsman, Peter J. Liu, Lechao Xiao, Katie E. Everett, Alexander A. Alemi, Ben Adlam, John D. Co-Reyes, Izzeddin Gur, Abhishek Kumar, Roman Novak, Jeffrey Pennington, Jascha Sohl-Dickstein, Kelvin Xu, Jaehoon Lee, Justin Gilmer, and Simon Kornblith. Small-scale proxies for large-scale transformer training instabilities. In *International Conference on Learning Representations*, 2024. (cited on pages 29 and 30)

Tete Xiao, Mannat Singh, Eric Mintun, Trevor Darrell, Piotr Dollár, and Ross Girshick. Early convolutions help transformers see better. In *Advances in Neural Information Processing Systems*, volume 34, pages 30392–30400, 2021. (cited on pages 7 and 18)

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*, volume 119, pages 10524–10533, 2020. (cited on pages 17 and 18)

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *International Conference on Learning Representations*, volume 2025, pages 29687–29707, 2025. (cited on pages 17 and 19)

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*, volume 32, pages 12360–12371, 2019. (cited on pages 4, 17, and 18)

Guodong Zhang, Lala Li, Zachary Nado, James Martens, Sushant Sachdeva, George Dahl, Chris Shallue, and Roger B Grosse. Which algorithmic choices matter at which batch sizes? insights from a noisy quadratic model. *Advances in Neural Information Processing Systems*, 32, 2019. (cited on page 3)

Hanlin Zhang, Depen Morwani, Nikhil Vyas, Jingfeng Wu, Difan Zou, Udaya Ghai, Dean Foster, and Sham M. Kakade. How does critical batch size scale in pre-training? In *International Conference on Learning Representations*, 2025. (cited on page 5)

Jingzhao Zhang, Sai Praneeth Karimireddy, Andreas Veit, Seungyeon Kim, Sashank Reddi, Sanjiv Kumar, and Suvrit Sra. Why are adaptive methods good for attention models? In *Advances in Neural Information Processing Systems*, volume 33, pages 15383–15393, 2020. (cited on page 3)

Yushun Zhang, Congliang Chen, Tian Ding, Ziniu Li, Ruoyu Sun, and Zhi-Quan Luo. Why transformers need Adam: A hessian perspective. In *Advances in Neural Information Processing Systems*, volume 37, pages 131786–131823, 2024a. (cited on pages 4, 5, and 7)

Zhengyan Zhang, Yixin Song, Guanghui Yu, Xu Han, Yankai Lin, Chaojun Xiao, Chenyang Song, Zhiyuan Liu, Zeyu Mi, and Maosong Sun. ReLU^2 wins: Discovering efficient activation functions for sparse LLMs, 2024b. (cited on page 4)

Rosie Zhao, Depen Morwani, David Brandfonbrener, Nikhil Vyas, and Sham M. Kakade. Deconstructing what makes a good optimizer for autoregressive language models. In *International Conference on Learning Representations*, 2025. (cited on pages 2, 4, 5, 7, and 8)

Pan Zhou, Jiashi Feng, Chao Ma, Caiming Xiong, Steven Chu Hong Hoi, et al. Towards theoretically understanding why sgd generalizes better than adam in deep learning. *Advances in Neural Information Processing Systems*, 33:21285–21296, 2020. (cited on page 2)

Nicolas Zucchet and Antonio Orvieto. Recurrent neural networks: Vanishing and exploding gradients are not the end of the story. In *Advances in Neural Information Processing Systems*, volume 37, pages 139402–139443, 2024. (cited on page 1)

Appendix contents

A Experimental details **17**

A.1 Datasets . . . 17
A.2 Architectures . . . 17
A.3 Experiments on FineWeb . . . 20
A.4 Experiments on ImageNet21K . . . 21
A.5 Experiments on ImageNet1K with heavy-tail . . . 22
A.6 Experiments on HG38 . . . 23
A.7 Experiments on ZINC250K . . . 24
A.8 Experiments on softmax-attention . . . 25
A.9 Experiments on hybrid optimizers . . . 26
A.10 Experiments on architecture interpolation . . . 27
A.11 Experiments on early training phase . . . 28

B Additional results **29**

B.1 Effect of weight decay . . . 29
B.2 Effect of number of steps . . . 31
B.3 Effect of independent βs . . . 33
B.4 Large scale GPT and ViT . . . 34
B.5 Pushing the batch size to 1 . . . 35
B.6 Validation performance gap . . . 36
B.7 Architecture interpolation with independent components . . . 37
B.8 Isolating early training phase with `stable_sgd` shared trajectory . . . 38

C Detailed theoretical study **39**

C.1 Notation and setting . . . 39
C.2 Convergence bounds . . . 39
C.3 Comparison of the bounds for NSGD and SignSGD . . . 40
C.4 Power-law spectrum . . . 41

A Experimental details

We present details for all the experiments in the main paper. Each run is performed on a single A100–40G NVIDIA GPU machine.

Table 3: Detailed overview of the main experimental setups proposed to study the Adam–SGD gap.

Architecture	Dataset	Modality	Transformer
GPT [Radford et al., 2019]	TinyStories [Eldan and Li, 2023]	Language	✓
GPT [Radford et al., 2019]	Fineweb [Penedo et al., 2024]	Language	✓
GPT [Radford et al., 2019]	HG38 [GRC, 2013]	Genomics	✓
ViT [Dosovitskiy et al., 2021]	HT-I1K [Kunstner et al., 2024]	Vision	✓
ViT [Dosovitskiy et al., 2021]	I21K [Deng et al., 2009]	Vision	✓
GRIT [Ma et al., 2023]	ZINC250K [Irwin et al., 2012]	Graphs	✓
GCNN [Dauphin et al., 2017]	TinyStories [Eldan and Li, 2023]	Language	✗
GCNN [Dauphin et al., 2017]	Fineweb [Penedo et al., 2024]	Language	✗
GDN [Yang et al., 2025]	HG38 [GRC, 2013]	Genomics	✗
ConvNext [Liu et al., 2022]	I21K [Deng et al., 2009]	Vision	✗
ResNet [He et al., 2016]	I21K [Deng et al., 2009]	Vision	✗
GAT [Veličković et al., 2018]	ZINC250K [Irwin et al., 2012]	Graphs	✗

A.1 Datasets

TinyStories. Synthetic dataset of short stories that contain only simple words, generated by GPT-3.5 and GPT-4. We use the first 10M tokens for overparametrized settings, preprocessed with `gpt2` tokenizer.

FineWeb. Large dataset derived from 96 Common Crawl snapshots. We use the first 1B tokens from the `fineweb-edu-sample-10BT`, preprocessed with `gpt2` tokenizer.

I21K (ImageNet21K). Large-scale image classification dataset containing approximately 14 million images across 21K synset classes from the ImageNet hierarchy. We use the `winter21` version: `https://huggingface.co/datasets/timm/imagenet-w21-wds` .

I1K heavy-tailed (ImageNet1K). Subset of $n = 10217$ samples from the ImageNet1K with a heavy-tailed class imbalance, constructed by sorting classes by frequency and sampling $\lceil \frac{1300}{k} \rceil$ images from the k-th class [Kunstner et al., 2024].

HG38. Human genome reference assembly GRCh38 (hg38), consisting of 2.7B nucleotides. Commonly used for training genomic sequence models. We use the preprocessed version: `https://storage.googleapis.com/basenji_barnyard2/hg38.ml.fa.gz` .

ZINC250K. Subset of 250,000 drug-like molecules drawn from the ZINC database of commercially available compounds. Molecules are represented as SMILES strings and used as a benchmark for molecular generation and optimization tasks.

A.2 Architectures

GPT: Transformer for language tasks. For pre-training Transformers under the causal language modeling objective, we build on the nanoGPT implementation [Karpathy, 2022], extending it with several recent model advances:

- Causal scaled dot-product attention
- RoPE [Su et al., 2024]
- PreNorm [Xiong et al., 2020]
- RMSNorm [Zhang and Sennrich, 2019]
- QKNorm [Touvron et al., 2023]

- EmbedNorm [Loshchilov et al., 2025]
- ReLU^2 MLP with ratio 4 [So et al., 2021]
- No biases
- Linear head

The configuration:

- Layers: 6
- Heads: 6
- Embedding size: 384
- Sequence length: 256
- Vocabulary size: [data dependent]

ViT: Transformer for vision tasks. We follow and extend the original model from [Dosovitskiy et al., 2021] with recent model advances:

- Scaled dot-product attention
- Conv2D stem [Xiao et al., 2021]
- Mean pooling
- RoPE2D axial / sincos2D [Heo et al., 2024]
- PreNorm [Xiong et al., 2020]
- RMSNorm [Zhang and Sennrich, 2019]
- QKNorm [Touvron et al., 2023]
- EmbedNorm [Loshchilov et al., 2025]
- ReLU^2 MLP with ratio 4 [So et al., 2021]
- No biases
- Linear head

The configuration:

- Layers: 6
- Heads: 6
- Embedding size: 384
- Patch size: 16
- Vocabulary size: [data dependent]

GPT: Transformer for genomic tasks. The model is the same as the GPT for language tasks, but with a smaller vocabulary size depending on the tokenizer {`char`, `kmer`}. The model configuration is the same as the GPT for language, except for a sequence length of 1024 and a smaller vocabulary.

GRIT: Transformer for graph tasks. We use the implementation from the original publicly released code: `https://github.com/LiamMa/GRIT/tree/main`.

This configuration has 0.27M parameters:

- Layers: 6
- Heads: 6
- Embedding size: 64
- k-steps: 21 [Ma et al., 2023]

ResNet50: Conv model for vision tasks. We use timm's implementation: `resnet50.a1_in1k`

ConvNext: Conv model for vision tasks. We use timm's implementation: `convnext_tiny`

GCNN: Conv model for language tasks. This revamped version of the gated convolution architecture is identical to the GPT for language tasks except for two changes:

- Softmax-attention is replaced by the GLU layer: $h(X) = (X * W + b) \otimes \sigma(X * V + c)$
- Sinusoidal positional embedding

The configuration is identical to the GPT model.

GAT: Message-passing model for graph tasks. Graph attention network is the model used for graph-level regression [Veličković et al., 2018]. The model architecture is:

- Embedding layer
- GATConv $\rightarrow$ Batch norm $\rightarrow$ ReLU blocks
- Global mean pooling
- Linear head

This configuration has 0.27M parameters:

- Layers: 4
- Heads: 8
- Embedding size: 256

GDN: linear Transformer for language tasks. We implement the GatedDeltaNet [Yang et al., 2025] identical to the GPT architecture used for language tasks except for two changes:

- Softmax-attention is replaced by the GatedDeltaNet block[5]
- Positional embedding is removed

For genomic task, same as GPT, we use a sequence length of 1024 and smaller vocabulary with the `char` tokenizer.

[5] https://github.com/fla-org/flash-linear-attention

A.3 Experiments on FineWeb

For the experiments in Figures 1 and 2, we use the model GPT 50M parameters (see §A.2) trained on 1B tokens (Chinchilla-optimal ratio of 20 tokens/parameter). We use `bfloat16` precision, 10% linear warmup with a cosine scheduler to 0, and global gradient norm clipping to 1 for both Adam and SGD.

We tune the learning rate η, momentum β, and batch size B as

- SGD (96 runs)

$$(\eta, \beta, B) \in \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}\} \times \{0.9, 0.95, 0.99, 0.999\} \times \{16, 64, 256, 1024\}.$$

- Adam (96 runs) ($\beta_1 = \beta_2 = \beta$)

$$(\eta, \beta, B) \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}\} \times \{0.9, 0.95, 0.99, 0.999\} \times \{16, 64, 256, 1024\}.$$

For $B = 16$, we perform a more finegrained sweep by reducing the edge of the grid and adding more points in between, adding 8 more configurations.

This setup trains a total of 200 model configurations.

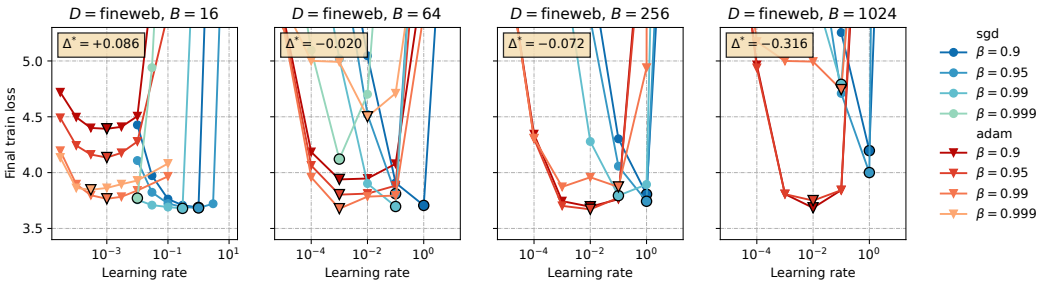


Figure 9: GPT 50M parameters trained on FineWeb 1B tokens.

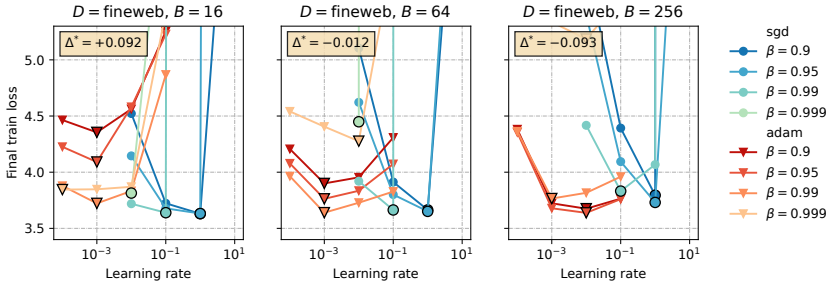


Figure 10: GDN 50M parameters trained on FineWeb 1B tokens.

A.4 Experiments on ImageNet21K

For the experiments in Figure 1, we use ViT and ResNet50 models (see §A.2) trained for 1 epoch on ImageNet21K. We use `bfloat16` precision, 10% linear warmup with cosine scheduler to 0, and global norm clipping to 1 for both Adam and SGD.

ViT. We tune the learning rate η, momentum β, and batch size B as

- SGD (48 runs)

$$(\eta, \beta, B) \in \{10^{-2}, 10^{-1}, 10^{0}, 10^{1}\} \times \{0.9, 0.95, 0.99, 0.999\} \times \{256, 1024, 4096\}.$$

- Adam (48 runs) ($\beta_1 = \beta_2 = \beta$)

$$(\eta, \beta, B) \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\} \times \{0.9, 0.95, 0.99, 0.999\} \times \{256, 1024, 4096\}.$$

ResNet50. We tune the learning rate η, momentum β, and batch size B as

- SGD (48 runs)

$$(\eta, \beta, B) \in \{10^{-2}, 10^{-1}, 10^{0}, 10^{1}\} \times \{0.9, 0.95, 0.99, 0.999\} \times \{1024, 4096, 16384\}.$$

- Adam (48 runs) ($\beta_1 = \beta_2 = \beta$)

$$(\eta, \beta, B) \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\} \times \{0.9, 0.95, 0.99, 0.999\} \times \{1024, 4096, 16384\}.$$

This setup trains a total of 192 model configurations.

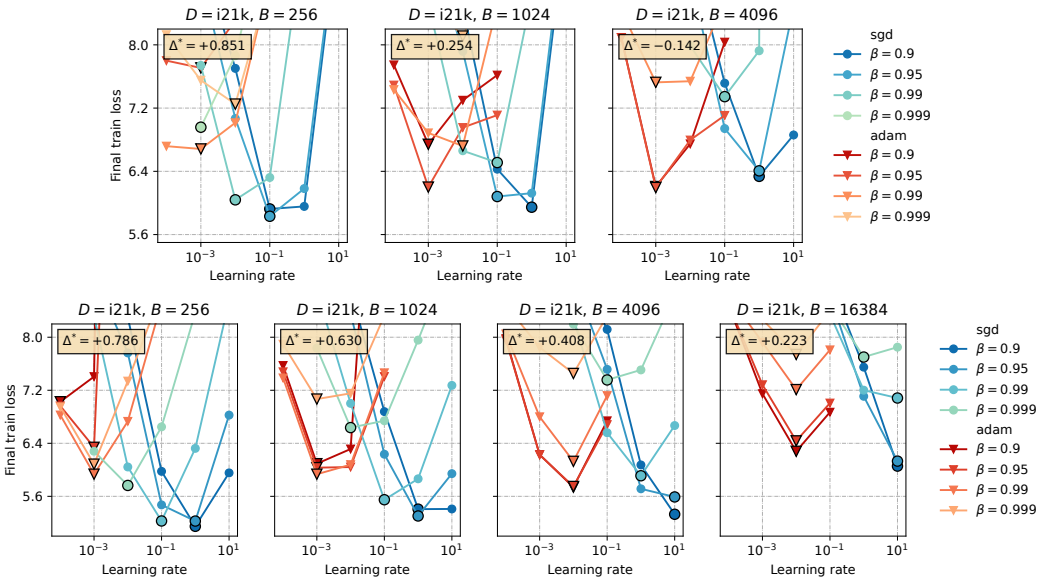


Figure 11: ViT (top) and ResNet50 (bottom) trained on I21K (13M training samples).

A.5 Experiments on ImageNet1K with heavy-tail

For the experiments in Figure 5, we reproduce the Kunstner et al. [2024]'s results using their SimpleViT architecture from `https://github.com/lucidrains/vit-pytorch` (`version=1.6.5`) compared to our modernized ViT architecture (see §A.2). Both models have 12 layers and are trained for 300 epochs with a constant learning rate, flip and random augmentations, $B = 1024$, sinusoidal position embedding, and fixed $\beta_2 = 0.999$. We use `bfloat16` precision. We apply global gradient norm clipping to 1 to our ViT model in Figure 5b.

We tune the learning rate η, momentum β, and batch size B as

- SGD (28 runs)

$$\begin{aligned}(\eta, \beta, B) \in &\{10^{-3}, 3 \cdot 10^{-3}, 10^{-2}, 3 \cdot 10^{-2}, 10^{-1}, 3 \cdot 10^{-1}, 10^{0}\} \\ &\times \{0.9, 0.95, 0.99, 0.999\} \\ &\times \{1024\}.\end{aligned}$$

- Adam (28 runs) ($\beta_2 = 0.999$):

$$\begin{aligned}(\eta, \beta_1, B) \in &\{10^{-5}, 3 \cdot 10^{-5}, 10^{-4}, 3 \cdot 10^{-4}, 10^{-3}, 3 \cdot 10^{-3}, 10^{-2}\} \\ &\times \{0.9, 0.95, 0.99, 0.999\} \\ &\times \{1024\}.\end{aligned}$$

This setup trains a total of 112 model configurations, as there are two configurations.

A.6 Experiments on HG38

For the experiments in Figures 1 and 3, we use the GPT model (see §A.2) trained on the HG38 dataset preprocessed using three different tokenizers: `char` (10M parameters model on 2.7B tokens), `4mer` (10M parameters model on 0.68B tokens), and `6mer` (13M parameters model 0.45B tokens). We use `bfloat16` precision. We apply global gradient norm clipping to 1 for both Adam and SGD.

Tokenizer `char`. We tune the learning rate η, momentum β, and batch size B as

- SGD (96 runs)

$$\begin{aligned}(\eta, \beta, B) \in &\{10^{-2}, 3\cdot 10^{-2}, 10^{-1}, 3\cdot 10^{-1}, 10^{0}, 3\cdot 10^{0}\} \\ &\times \{0.9, 0.95, 0.99, 0.999\} \\ &\times \{16, 64, 256, 1024\}.\end{aligned}$$

- Adam (96 runs) ($\beta_1 = \beta_2 = \beta$)

$$\begin{aligned}(\eta, \beta, B) \in &\{10^{-5}, 3\cdot 10^{-5}, 10^{-4}, 3\cdot 10^{-4}, 10^{-3}, 3\cdot 10^{-3}\} \\ &\times \{0.9, 0.95, 0.99, 0.999\} \\ &\times \{16, 64, 256, 1024\}.\end{aligned}$$

Tokenizer `4mer` and `6mer`. We tune the learning rate η, momentum β, with batch size $B = 256$ as

- SGD (24 runs)

$$\begin{aligned}(\eta, \beta) \in &\{10^{-2}, 3\cdot 10^{-2}, 10^{-1}, 3\cdot 10^{-1}, 10^{0}, 3\cdot 10^{0}\} \\ &\times \{0.9, 0.95, 0.99, 0.999\}.\end{aligned}$$

- Adam (24 runs) ($\beta_1 = \beta_2 = \beta$)

$$\begin{aligned}(\eta, \beta) \in &\{10^{-5}, 3\cdot 10^{-5}, 10^{-4}, 3\cdot 10^{-4}, 10^{-3}, 3\cdot 10^{-3}\} \\ &\times \{0.9, 0.95, 0.99, 0.999\}.\end{aligned}$$

Tokenizer `char` with GDN. Same setup as GPT on genomics but using the model GDN. We tune the hyperparams grid as GPT on genomics above.

This setup trains a total of 432 model configurations.

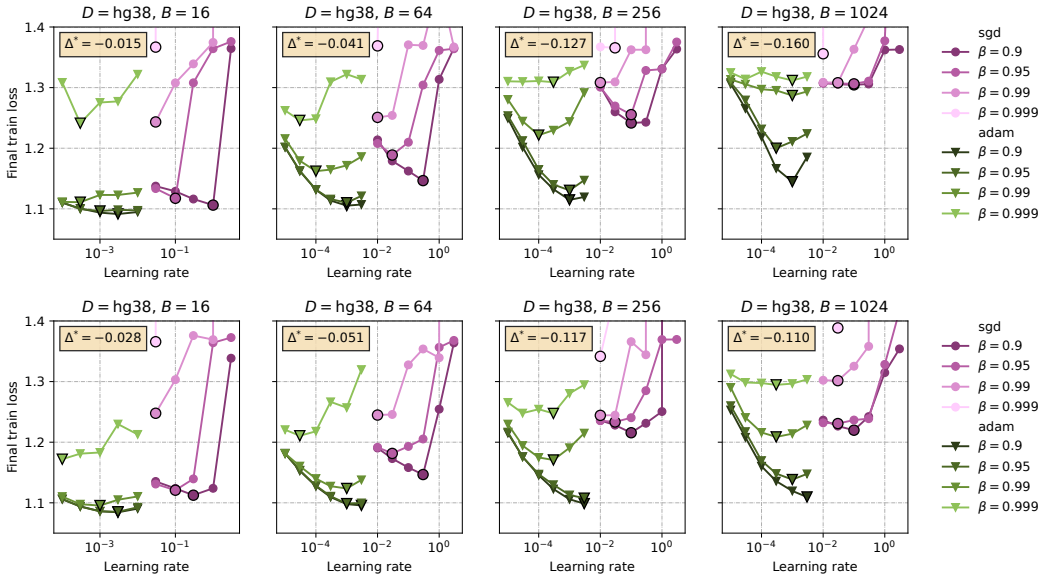


Figure 12: GPT (top) and GDN (bottom) 10M parameters trained on HG38 2.7B tokens.

A.7 Experiments on ZINC250K

For the experiments in Figures 1 and 4, we use GRIT 0.27M parameters and GAT 0.27M model parameters (see §A.2) trained for 100 epochs on ZINC250K. We use `bfloat16` precision, 10% linear warmup with cosine scheduler to 0, and global norm clipping to 1 for both Adam and SGD.

GRIT. We tune the learning rate η, momentum β, and batch size B as

- SGD (64 runs)

$$(\eta, \beta, B) \in \{10^{-3}, 10^{-2}, 10^{-1}, 10^{0}\} \times \{0.9, 0.95, 0.99, 0.999\} \times \{16, 64, 256, 1024\}.$$

- Adam (64 runs) ($\beta_1 = \beta_2 = \beta$)

$$(\eta, \beta, B) \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\} \times \{0.9, 0.95, 0.99, 0.999\} \times \{16, 64, 256, 1024\}.$$

GAT. We tune the learning rate η, momentum β, and batch size B as

- SGD (48 runs)

$$(\eta, \beta, B) \in \{10^{-2}, 10^{-1}, 10^{0}, 10^{1}\} \times \{0.9, 0.95, 0.99, 0.999\} \times \{256, 1024, 4096\}.$$

- Adam (48 runs) ($\beta_1 = \beta_2 = \beta$)

$$(\eta, \beta, B) \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\} \times \{0.9, 0.95, 0.99, 0.999\} \times \{256, 1024, 4096\}.$$

This setup trains a total of 224 model configurations.

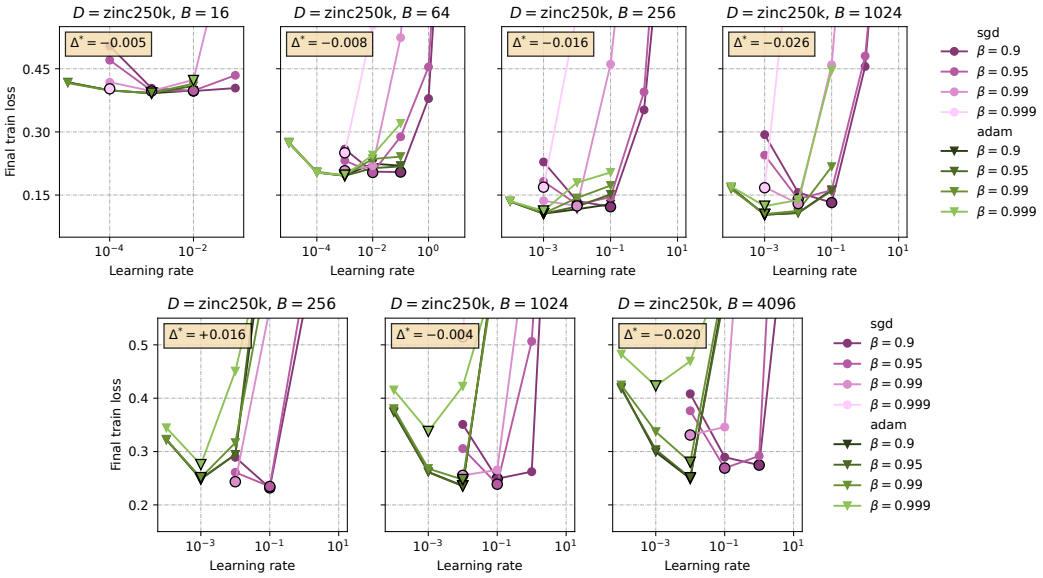


Figure 13: GRIT (top) and GAT (bottom) trained on ZINC250K.

A.8 Experiments on softmax-attention

For the experiments in Figures 1 and 2, we use the model GCNN 53M parameters (see §A.2) trained on 1B tokens (Chinchilla-optimal 20 tokens/parameter). We use `bfloat16` precision, 10% linear warmup with cosine scheduler to 0, and global gradient norm clipping to 1 for both Adam and SGD.

We tune the learning rate η, momentum β, and batch size B as

- SGD (48 runs)

$$(\eta, \beta, B) \in \{10^{-2}, 10^{-1}, 10^{0}, 10^{1}\} \times \{0.9, 0.95, 0.99, 0.999\} \times \{64, 256, 1024\}.$$

- Adam (48 runs) ($\beta_2 = 0.999$):

$$(\eta, \beta_1, B) \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\} \times \{0.9, 0.95, 0.99, 0.999\} \times \{64, 256, 1024\}.$$

For $B = 1024$, an additional sweep is run for both SGD ($\eta = 10^{-3}$) and Adam ($\eta = 10^{0}$) to cover the edges induced by batch size scaling, resulting in 8 more runs.

This setup trains a total of 104 model configurations.

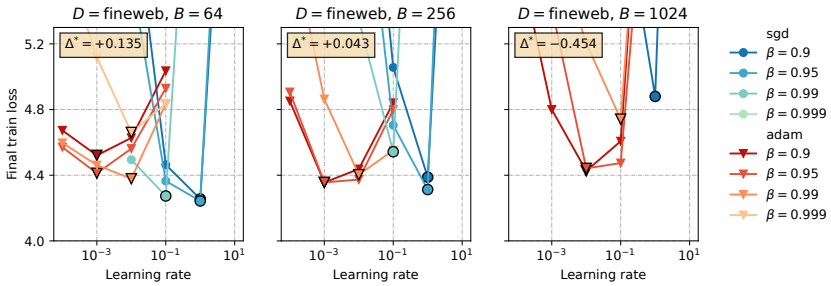


Figure 14: GCNN 53M parameters trained on FineWeb 1B.

A.9 Experiments on hybrid optimizers

For the experiments in Table 1, we use the model GPT 50M parameters (see §A.2) trained on 1B tokens (Chinchilla-optimal 20 tokens/parameter). In this set of experiments, particular layers are trained with Adam, while the rest of the network is trained with SGD. For both the Adam and SGD layers, we use `bfloat16` precision, 10% linear warmup with cosine scheduler to 0, global gradient norm clipping to 1, and a batch size of 1024.

For each hybrid configuration of the optimizer, we tune the learning rates η_{SGD} and η_{Adam}, momentum β_{SGD} and β_{Adam}, and batch size B as

- Hybrid optimizer (225 runs) (for Adam layers $\beta_1 = \beta_2 = \beta_{\text{Adam}}$)

$$\begin{aligned}(\eta_{\text{SGD}}, \eta_{\text{Adam}}, \beta_{\text{SGD}}, \beta_{\text{Adam}}, B) \in &\{10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}\} \\ &\times \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}\} \\ &\times \{0.9, 0.95, 0.99\} \\ &\times \{0.9, 0.95, 0.99\} \\ &\times \{1024\}.\end{aligned}$$

This setup trains a total of 1125 model configurations, as there are 5 hybrid configurations in Table 1.

A.10 Experiments on architecture interpolation

For the experiments in Table 2, we use different variants of ResNet50 (see §A.2) trained on I21K for one epoch. We use `bfloat16` precision, 10% linear warmup with cosine scheduler to 0, and global gradient norm clipping to 1 for both Adam and SGD.

We keep the same ResNet50 structure through the interpolation, that is the same channel dimensions (256, 512, 1024, 2048) and depths (3, 4, 6, 3). Only specific layers are modified. The three modifications are:

- ReLU $\to$ GeLU. Substitute all ReLU with GeLU.
- BatchNorm $\to$ LayerNorm. Substitute all BatchNorm with LayerNorm.
- Dense mixing $\to$ Depthwise mixing. Each output channel only looks at its corresponding input channel. There is no cross-channel mixing as there is in Dense mixing [Liu et al., 2022].

We tune the learning rate η, momentum β, and batch size B as

- SGD (12 runs)

$$\begin{aligned}(\eta, \beta, B) \in &\{10^{-2}, 10^{-1}, 10^{0}\} \\ &\times \{0.9, 0.95, 0.99, 0.999\} \\ &\times \{1024\}.\end{aligned}$$

- Adam (12 runs) ($\beta_1 = \beta_2 = \beta$):

$$\begin{aligned}(\eta, \beta, B) \in &\{10^{-4}, 10^{-3}, 10^{-2}\} \\ &\times \{0.9, 0.95, 0.99, 0.999\} \\ &\times \{1024\}.\end{aligned}$$

This setup trains a total of 72 model configurations, as there are 3 configurations in Table 2.

A.11 Experiments on early training phase

For the experiments in Figure 7, we use GPT 50M parameters (see §A.2) trained on 10M TinyStories tokens for 80 epochs and 1B FineWeb tokens for one epoch. We use `bfloat16` precision, 10% linear warmup with a constant learning rate to train the shared trajectory `stable_adam`. For each branch, we reset the momentum state, tune the learning rate and momentum, apply the linear scheduler to 0, and global gradient norm clipping to 1 for both Adam and SGD.

There are 3 branching points for TinyStories at fractions $\{0.2, 0.4, 0.6\}$ of the total training steps, while for FineWeb there are 4 checkpoints at $\{0.1, 0.3, 0.5, 0.7\}$. When branching in FineWeb, the batches that were already seen in the `stable` phase are correctly skipped. The total training budget is also reduced by skipping the `stable` steps.

We tune the learning rate η, momentum β, and batch size B at each branch point as

- SGD (20 runs)

$$\begin{aligned}(\eta, \beta, B) \in &\{3 \cdot 10^{-2}, 10^{-1}, 3 \cdot 10^{-1}, 10^{0}, 3 \cdot 10^{0}\} \\ &\times \{0.9, 0.95, 0.99, 0.999\} \\ &\times \{1024\}.\end{aligned}$$

- Adam (20 runs) ($\beta_1 = \beta_2 = \beta$):

$$\begin{aligned}(\eta, \beta, B) \in &\{10^{-3}, 3 \cdot 10^{-3}, 10^{-2}, 3 \cdot 10^{-2}, 10^{-1}\} \\ &\times \{0.9, 0.95, 0.99, 0.999\} \\ &\times \{1024\}.\end{aligned}$$

This setup trains a total of 280 model configurations, as there are 7 branches.

B Additional results

B.1 Effect of weight decay

We ablate the effect of weight decay in both vision and language. The results are consistent with the trend observed in Figure 1 without weight decay. That is, we still observe an optimizer shift from SGD to Adam as the batch size scales. However, the final train loss is higher than the corresponding run without weight decay.

ViT on I21K. As in §A.4, we use a ViT model (see §A.2) trained for 1 epoch on ImageNet21K. We use `bfloat16` precision, 10% linear warmup with cosine scheduler to 0, and global norm clipping to 1 for both Adam and SGD. We use *independent* weight decay [Wortsman et al., 2024], which is decoupled and independent of learning rate η.

We tune the learning rate η, momentum β, and batch size B, also adding the weight decay λ as

- SGD (48 runs)

$$\begin{aligned}(\eta, \beta, B, \lambda) \in &\{10^{-2}, 10^{-1}, 10^{0}, 10^{1}\} \\ &\times \{0.9, 0.95, 0.99\} \\ &\times \{1024, 4096\} \\ &\times \{10^{-3}, 10^{-4}\}.\end{aligned}$$

- Adam (48 runs) ($\beta_1 = \beta_2 = \beta$)

$$\begin{aligned}(\eta, \beta, B, \lambda) \in &\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\} \\ &\times \{0.9, 0.95, 0.99\} \\ &\times \{1024, 4096\} \\ &\times \{10^{-3}, 10^{-4}\}.\end{aligned}$$

This setup trains a total of 120 model configurations. For $\lambda = 10^{-3}$, we have 24 more runs.

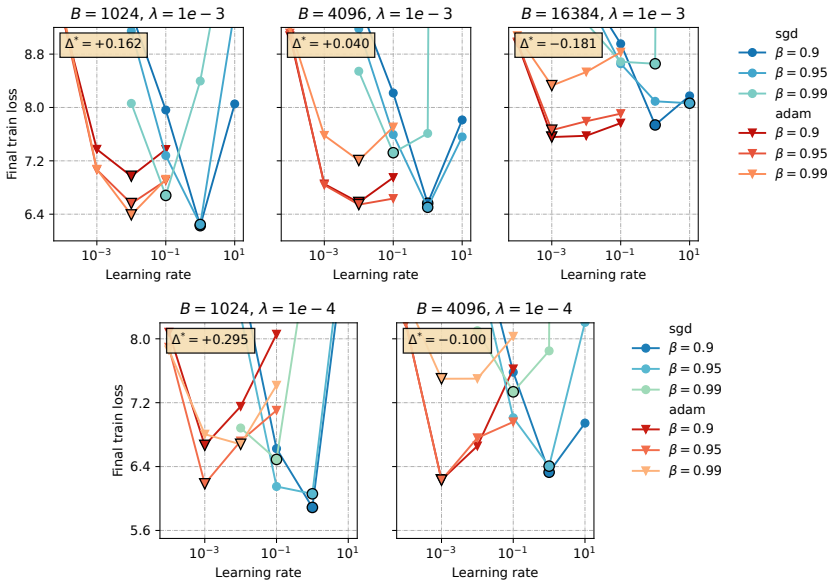


Figure 15: ViT model trained on I21K with *independent* weight decay λ [Wortsman et al., 2024]. We observe the same trend as Figure 1, that is the optimizer advantage shifts from SGD $\rightarrow$ Adam as the batch size scales.

GPT on FineWeb. Similarly to §A.3, we use a GPT model (see §A.2) trained on FineWeb 1B tokens. We use `bfloat16` precision, 10% linear warmup with cosine scheduler to 0, and global norm clipping to 1 for both Adam and SGD. We use *independent* weight decay [Wortsman et al., 2024], which is decoupled and independent of learning rate η.

We tune the learning rate η, momentum β, and batch size B, also adding the weight decay λ as

- SGD (32 runs)

$$\begin{aligned}(\eta, \beta, B, \lambda) \in &\{10^{-2}, 10^{-1}, 10^{0}, 10^{1}\} \\ &\times \{0.9, 0.95, 0.99, 0.999\} \\ &\times \{16, 1024\} \\ &\times \{10^{-4}\}.\end{aligned}$$

- Adam (32 runs) ($\beta_1 = \beta_2 = \beta$)

$$\begin{aligned}(\eta, \beta, B, \lambda) \in &\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\} \\ &\times \{0.9, 0.95, 0.99, 0.999\} \\ &\times \{16, 1024\} \\ &\times \{10^{-4}\}.\end{aligned}$$

This setup trains a total of 64 model configurations.

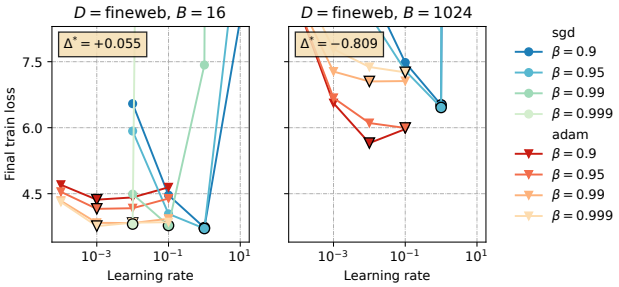


Figure 16: GPT model trained on FineWeb with *independent* weight decay $\lambda = 1e-4$ [Wortsman et al., 2024]. We observe the same trend as Figure 1, that is the optimizer advantage shifts from SGD to Adam as the batch size scales.

B.2 Effect of number of steps

Throughout this paper, we presented the batch size sweep results (Figure 1) that fix the total number of training samples (or total token budget). Here, we conduct a preliminary study on the effect of number of steps. Specifically, there are two questions: (a) under fixed batch size B, how does the Adam–SGD gap change when scaling the number of steps? (b) under fixed number of steps, how does the gap change when scaling the B? We train ViT on I21K for multiple steps (or epochs).

Figure 17 (left) shows that when fixing the batch size, training for longer epochs decreases the gap towards $\Delta = 0$ regardless of whether the initial advantage is for SGD or Adam, suggesting that more iterations reduce the gap. When fixing the number of step size, a larger B shifts the advantage towards Adam.

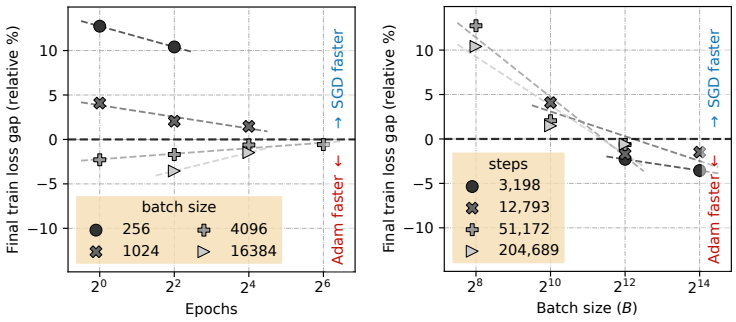


Figure 17: ViT trained on I21K for multiple epochs at different batch size B. (left) For a fixed B, training for more steps decreases the gap towards $\Delta = 0$. (right) For a fixed number of steps, a larger B shifts the advantage towards Adam.

For the experiments in Figure 17, we use the same ViT model from §A.2 but trained for a longer horizon. In particular, we use `bfloat16` precision, 10% linear warmup with cosine scheduler to 0, and global norm clipping to 1 for both Adam and SGD. We train the model for 4, 16, and 64 epochs.

Epochs 4. We tune the learning rate η, momentum β, and batch size B as

- SGD (64 runs)

$$\begin{aligned}(\eta, \beta, B) \in &\{10^{-2}, 10^{-1}, 10^{0}, 10^{1}\} \\ &\times \{0.9, 0.95, 0.99, 0.999\} \\ &\times \{256, 1024, 4096, 16384\}.\end{aligned}$$

- Adam (64 runs) ($\beta_1 = \beta_2 = \beta$)

$$\begin{aligned}(\eta, \beta, B) \in &\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\} \\ &\times \{0.9, 0.95, 0.99, 0.999\} \\ &\times \{256, 1024, 4096, 16384\}.\end{aligned}$$

Epochs 16. We tune the learning rate η, momentum β, and batch size B as

- SGD (27 runs)

$$\begin{aligned}(\eta, \beta, B) \in &\{10^{-2}, 10^{-1}, 10^{0}\} \\ &\times \{0.9, 0.95, 0.99\} \\ &\times \{1024, 4096, 16384\}.\end{aligned}$$

- Adam (27 runs) ($\beta_1 = \beta_2 = \beta$)

$$\begin{aligned}(\eta, \beta, B) \in &\{10^{-4}, 10^{-3}, 10^{-2}\} \\ &\times \{0.9, 0.95, 0.99\} \\ &\times \{1024, 4096, 16384\}.\end{aligned}$$

Epochs 64. We tune the learning rate η, momentum β, and batch size B as

- SGD (9 runs)

$$(\eta, \beta, B) \in \{10^{-2}, 10^{-1}, 10^{0}\} \times \{0.9, 0.95, 0.99\} \times \{4096\}.$$

- Adam (9 runs) ($\beta_1 = \beta_2 = \beta$)

$$(\eta, \beta, B) \in \{10^{-3}, 10^{-2}, 10^{-1}\} \times \{0.9, 0.95, 0.99\} \times \{4096\}.$$

For 16 and 64 epochs, training requires more resources. Therefore, we excluded the consistently worst-performing settings from epochs 1 and 4: $\beta = 0.999$ and η values.

This setup trains a total of 200 model configurations.

Epochs 4 (ResNet50). Lastly, we train ResNet50 also for a larger number of steps. We tune the learning rate η, momentum β, and batch size B as

- SGD (60 runs)

$$(\eta, \beta, B) \in \{10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 3 \cdot 10^{1}\} \times \{0.9, 0.95, 0.99, 0.999\} \times \{4096, 16384, 65536\}.$$

- Adam (60 runs) ($\beta_1 = \beta_2 = \beta$)

$$(\eta, \beta, B) \in \{10^{-4}, 10^{-3}, 10^{-2}, 3 \cdot 10^{-2}, 10^{-1}\} \times \{0.9, 0.95, 0.99, 0.999\} \times \{4096, 16384, 65536\}.$$

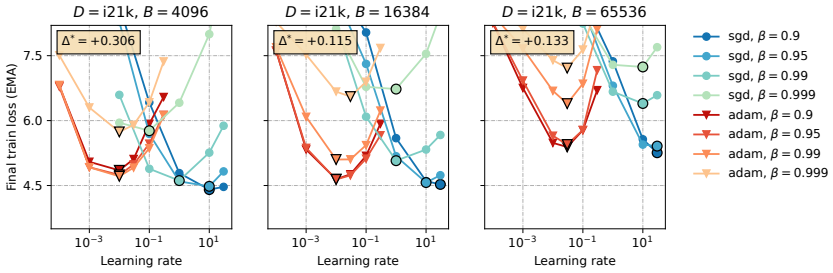


Figure 18: ResNet50 trained on I21K for 4 epochs across B. Note that even at a large $B = 65536$, SGD still has the advantage, suggesting a gap floor that depends on the architecture an data.

B.3 Effect of independent βs

To ablate our prior results that rely on $\beta = \beta_1 = \beta_2$ [Orvieto and Gower, 2025], we run independent βs sweep at small batch size. We use the same setup as §A.3.

We tune the learning rate η, β_1, β_2, and batch size B as

- Adam (240 runs) ($\beta_1 \neq \beta_2$)

$$\begin{aligned}(\eta, \beta_1, \beta_2, B) \in &\{3 \cdot 10^{-5}, 10^{-4}, 3 \cdot 10^{-4}, 10^{-3}, 3 \cdot 10^{-3}\} \\ &\times \{0.9, 0.95, 0.99, 0.999\} \\ &\times \{0.9, 0.95, 0.99, 0.999, 0.9999, 0.99999\} \\ &\times \{1, 16\}.\end{aligned}$$

Figure 19 shows that at $B = 16$, a large $\beta_2 = 0.999$ leads to similar performance as SGD. The same is observed for $B = 1$ but requiring an even larger $\beta_2 = 0.9999$. At first glance, this seems in contrast with prior results showing that SGD outperforms Adam at small B for this setup. Note however, that Adam with a $\beta_2 \to 1$ corresponds essentially to SGD. Since SGD is optimal under small $B = 1$, Adam with $\beta_2 \approx 1$ approximates SGD, suppressing its adaptivity.

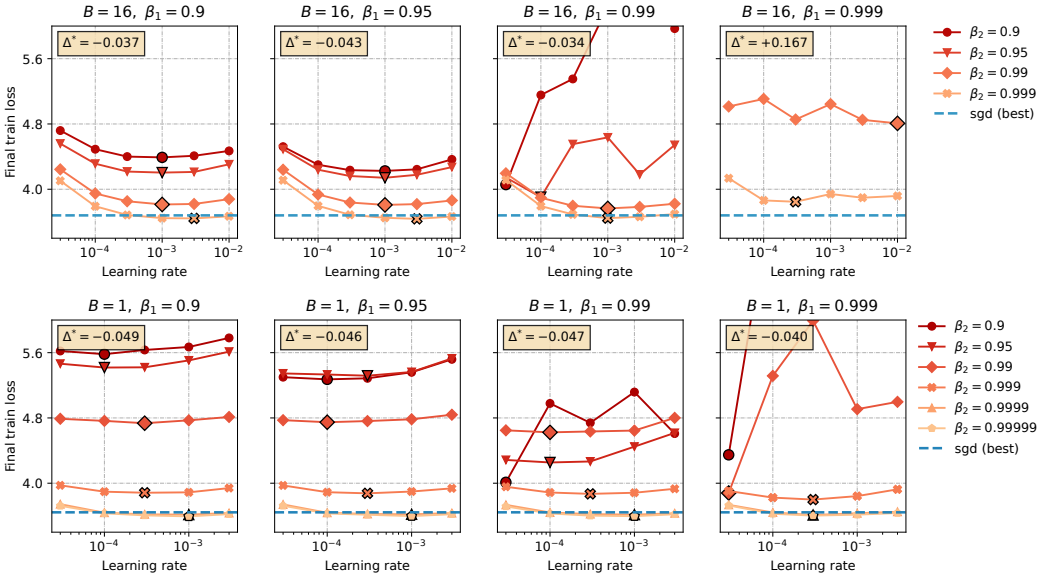


Figure 19: GPT trained on FineWeb with independent βs at $B = 1$ and $B = 16$.

B.4 Large scale GPT and ViT

We extend results in the main paper with larger scale architectures, showing that the *crossover* batch size is observed even for larger scale GPT and ViT architectures.

GPT 250M on FineWeb 5BT. Same setup as §A.3 but larger scale. The GPT here has 24 layers, 12 heads, 768 embed size, and 1024 seq length. We tune the learning rate η, β, and batch size B as

- SGD (64 runs)

$$(\eta, \beta, B) \in \{10^{-2}, 10^{-1}, 10^{0}, 10^{1}\} \times \{0.9, 0.95, 0.99, 0.999\} \times \{16, 64, 256, 1024\}.$$

- Adam (64 runs) ($\beta_1 = \beta_2 = \beta$)

$$(\eta, \beta, B) \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\} \times \{0.9, 0.95, 0.99, 0.999\} \times \{16, 64, 256, 1024\}.$$

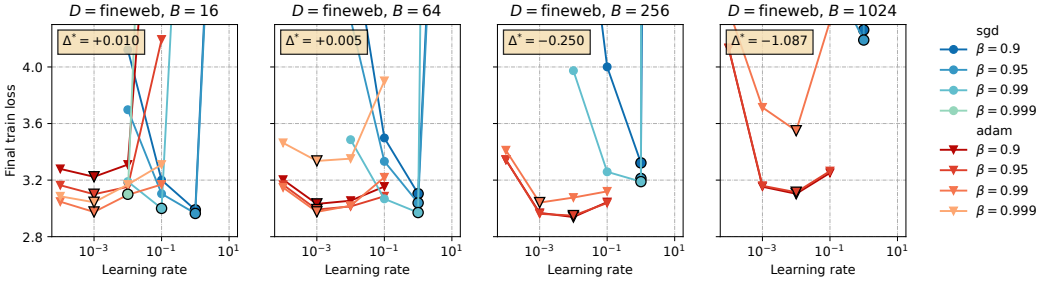


Figure 20: GPT 250M trained on FineWeb 5BT across batch sizes.

ViT 128M on I21K. Same setup as §A.4 but larger scale. The ViT architecture here has 16 layers, 16 heads, and 768 embedding size. We tune the learning rate η, β, and batch size B as

- SGD (32 runs)

$$(\eta, \beta, B) \in \{10^{-2}, 10^{-1}, 10^{0}, 10^{1}\} \times \{0.9, 0.95, 0.99, 0.999\} \times \{1024, 4096\}.$$

- Adam (32 runs) ($\beta_1 = \beta_2 = \beta$)

$$(\eta, \beta, B) \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\} \times \{0.9, 0.95, 0.99, 0.999\} \times \{1024, 4096\}.$$

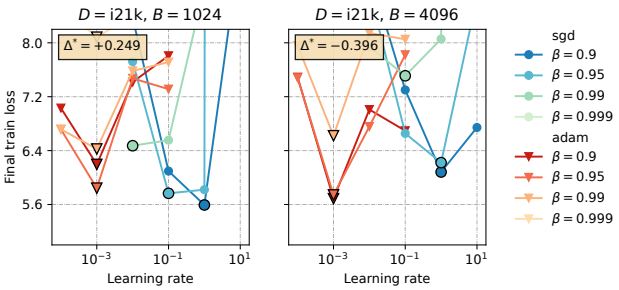


Figure 21: ViT 128M trained on I21K across batch sizes.

This setup trains a total of 192 model configurations.

B.5 Pushing the batch size to 1

Extending the §A.6, we push the GPT trained on HG38 dataset using `char` tokenizer to batch size 1 and 4. We observe that even at the smallest batch size, the gap remains in favor of Adam. This suggests that, in certain setups, Adam may always be better.

We tune the learning rate η, momentum β, and batch size B as

- SGD (32 runs)

$$\begin{aligned}(\eta, \beta, B) \in &\{10^{-2}, 3 \cdot 10^{-2}, 10^{-1}, 3 \cdot 10^{-1}, 10^{0}\} \\ &\times \{0.9, 0.95, 0.99, 0.999\} \\ &\times \{1, 4\}.\end{aligned}$$

- Adam (32 runs) ($\beta_1 = \beta_2 = \beta$)

$$\begin{aligned}(\eta, \beta, B) \in &\{10^{-4}, 3 \cdot 10^{-4}, 10^{-3}, 3 \cdot 10^{-3}\} \\ &\times \{0.9, 0.95, 0.99, 0.999\} \\ &\times \{1, 4\}.\end{aligned}$$

This setup trains a total of 64 model configurations.

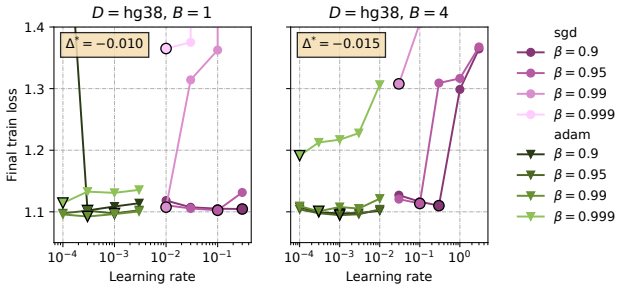


Figure 22: GPT trained on HG38 at smallest batch size still presents Adam>SGD.

B.6 Validation performance gap

Throughout this work, we have focused on training loss. Here, we report the gap in generalization performance. These are the same models from Figure 1, that is GPT from §A.3 and ViT from §A.4.

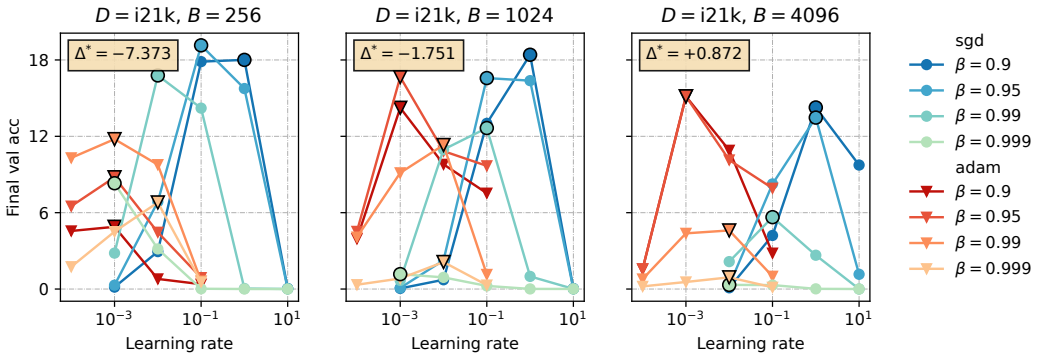


Figure 23: Top-1 validation accuracy of ViT model trained on I21K. We observe the same trend as Figure 1, that is the optimizer advantage shifts from SGD → Adam as the batch size scales.

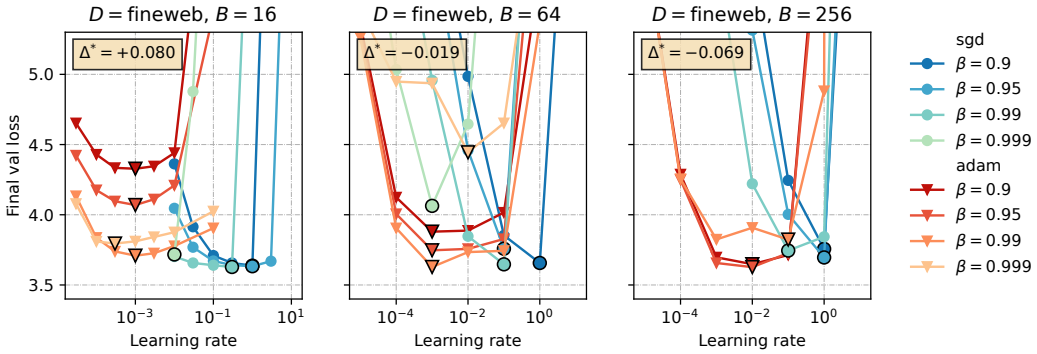


Figure 24: Validation loss of GPT model of 50M parameters trained on FineWeb 1B tokens. We observe the same trend as Figure 1, that is the optimizer advantage shifts from SGD → Adam as the batch size scales.

B.7 Architecture interpolation with independent components

Same experimental settings as §A.10, including the learning rate η and momentum β tuning grid.

We tune the learning rate η, momentum β, and batch size B as

- SGD (12 runs)

$$\begin{aligned}(\eta, \beta, B) \in &\{10^{-2}, 10^{-1}, 10^{0}\} \\ &\times \{0.9, 0.95, 0.99, 0.999\} \\ &\times \{1024\}.\end{aligned}$$

- Adam (12 runs) ($\beta_1 = \beta_2 = \beta$):

$$\begin{aligned}(\eta, \beta, B) \in &\{10^{-4}, 10^{-3}, 10^{-2}\} \\ &\times \{0.9, 0.95, 0.99, 0.999\} \\ &\times \{1024\}.\end{aligned}$$

Table 4: Single architecture-choice ablation (L: LayerNorm, G: GeLU, and D: Depthwise) at $B = 1024$. Each individual modification slightly moves the advantage towards Adam. However, as we have seen in Table 2, the cumulative effect is even stronger. The Δ measures the difference between the best run trained with SGD and Adam ($\Delta < 0$ means Adam has lower loss).

	Train loss			
Arch.	**Adam**	**SGD**	**Δ**	**Δ%**
ResNet50	6.133	5.337	+0.797	+13.00%
+ L	7.725	7.162	+0.563	+7.29%
+ G	5.817	5.233	+0.584	+11.16%
+ D	6.002	5.348	+0.654	+12.23%

This setup trains a total of 72 model configurations, as there are 3 configurations in Table 4.

B.8 Isolating early training phase with `stable_sgd` shared trajectory

Same setup as in §A.11 but training the shared trajectory `stable_sgd` instead of `stable_adam`. This shared SGD trajectory uses $\eta = 3 \cdot 10^{-1}$ and $\beta = 0.95$.

We tune the learning rate η, momentum β, and batch size B at every branching as

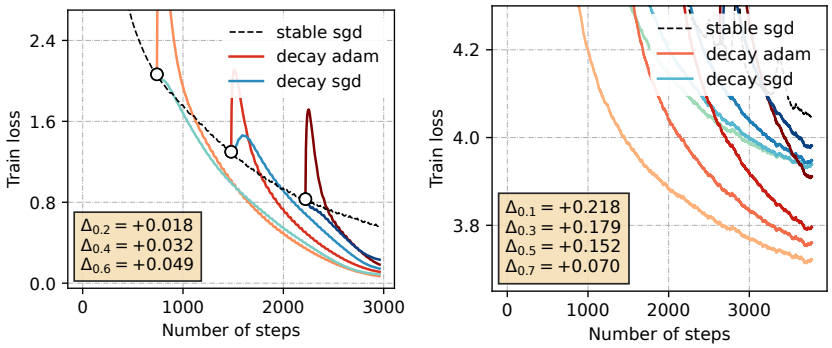
- SGD (20 runs)

$$\begin{aligned}(\eta, \beta, B) \in &\{3 \cdot 10^{-2}, 10^{-1}, 3 \cdot 10^{-1}, 10^{0}, 3 \cdot 10^{0}\} \\ &\times \{0.9, 0.95, 0.99, 0.999\} \\ &\times \{1024\}.\end{aligned}$$

- Adam (20 runs) ($\beta_1 = \beta_2 = \beta$):

$$\begin{aligned}(\eta, \beta, B) \in &\{10^{-3}, 3 \cdot 10^{-3}, 10^{-2}, 3 \cdot 10^{-2}, 10^{-1}\} \\ &\times \{0.9, 0.95, 0.99, 0.999\} \\ &\times \{1024\}.\end{aligned}$$

This setup trains a total of 280 model configurations, as there are 7 branches.



(a) Overparametrized w/ TinyStories

(b) Underparametrized w/ FineWeb

Figure 25: GPT model of 50M parameters trained on (a) 10M TinyStories tokens for 80 epochs and (b) 1B FineWeb tokens for 1 epoch at batch size $B = 1024$. We isolate the effect of early training dynamics by branching Adam and SGD from a shared constant learning rate trajectory (`stable_sgd`). Branching happens at different intervals to probe different stages of training. (a) For overparametrization, both branches converge to low loss as the problem is solvable with more epochs, while (b) for underparametrization, SGD lags behind. For every branch, the momentum state is reset, and the learning rate η and momentum β are tuned, see §A.11.

C Detailed theoretical study

We model the relationship between the Adam–SGD gap with batch size and the existence of a crossover batch size. We use the non-convex convergence bounds from Kovalev [2025], based on the gradient norm. These bounds were found to have a strong predictive ability for the scaling of the algorithm's parameters with the token budget T [Shulgin et al., 2026, Islamov et al., 2026].

C.1 Notation and setting

The goal is to solve the stochastic optimization problem

$$\min_{x \in \mathcal{X}} \left[f(x) = \mathbb{E}_{\xi \sim \mathcal{D}}[f(x, \xi)] \right], \tag{1}$$

where the space $\mathcal{X}$ is equipped with an inner product $\langle \cdot, \cdot \rangle$ and a norm $\| \cdot \|$, which need not be the norm induced by this inner product. To solve Problem 1, we use the Stochastic Trust-Region Gradient Method, summarized in Algorithm 1. Depending on the choice of $\| \cdot \|$, this framework recovers different optimization algorithms. For example, when $\| \cdot \| = \| \cdot \|_2$, the update in Algorithm 1 reduces to

$$x_{k+1} = x_k - \eta \frac{m_{k+1}}{\|m_{k+1}\|_2},$$

which corresponds to a step of normalized SGD (NSGD) with momentum. When $\| \cdot \| = \| \cdot \|_\infty$, the update recovers SignSGD with momentum,

$$x_{k+1} = x_k - \eta \text{sign}(m_{k+1}).$$

We approximate the convergence behavior of SGD with momentum and clipping by that of normalized SGD with momentum. Similarly, we use SignSGD with momentum as a proxy for Adam with clipping, motivated by their empirical similarity [Bernstein et al., 2018, Orvieto and Gower, 2025].

C.2 Convergence bounds

Assuming that

- $\|\nabla f(x) - \nabla f(y)\|_* \leq L\|x - y\|$ for all $x, y \in \mathbb{R}^d$,
- there exists $\rho > 0$ such that $\|x\|_* \leq \rho \|x\|_2$, and
- there exists σ^2 such that $\mathbb{E}_{\xi \sim \mathcal{D}}[\|\nabla f(x, \xi) - \nabla f(x)\|_2^2] \leq \frac{\sigma^2}{B}$, where B is the batch size,

Kovalev [2025] provide the following non-convex bounds

$$\begin{aligned} \min_{1 \leq k \leq K} \mathbb{E}[\|\nabla f(x_k)\|_*] &\lesssim \frac{\delta_0}{\eta K} + \frac{\rho\sigma}{\alpha\sqrt{B}K} + \frac{\sqrt{\alpha}\rho\sigma}{\sqrt{B}} + L\eta + \frac{L\eta}{\alpha} \\ &= \frac{\delta_0 B}{\eta T} + \frac{\rho\sigma\sqrt{B}}{\alpha T} + \frac{\sqrt{\alpha}\rho\sigma}{\sqrt{B}} + L\eta + \frac{L\eta}{\alpha}, \end{aligned} \tag{2}$$

where K is the number of optimization steps, T is the total budget, $\delta_0 = f(x_0) - f^*$ is the initial suboptimality gap, and numerical constants are ignored. We use the connection $T = KB$ to obtain

Algorithm 1 Stochastic Trust-region Gradient Method with Momentum

Input: norm $\| \cdot \|$, step size $\eta > 0$, momentum $\alpha \in (0, 1)$, number of iterations $K \in \{1, 2, \dots\}$
for $k = 1, \dots, K - 1$ **do**
  Sample $\xi_k \in \mathcal{D}$
  Compute $m_{k+1} = (1 - \alpha)m_k + \alpha \nabla f(x_k, \xi_k)$
  Compute $d_{k+1} = \text{argmin}_d \langle d, m_{k+1} \rangle$ such that $\|d\| \leq 1$
  Update $x_{k+1} = x_k + \eta d_{k+1}$
end for
return

the second line in Equation (2). Following derivations of Shulgin et al. [2026] in the large $T \gg 1$ and small α regime, we obtain the optimal η and α as

$$\eta \propto \frac{B\delta_0^{3/4}}{L^{1/4}(\rho\sigma)^{1/2}T^{3/4}}, \quad \alpha \propto \frac{\sqrt{L\delta_0}B}{\rho\sigma\sqrt{T}}. \tag{3}$$

Using this choice of hyperparameters, the bound minimized with respect to both η and α is

$$\min_{1\le k\le K} \mathbb{E}[\|\nabla f(x_k)\|_*] \lesssim \underbrace{\frac{(L\delta_0)^{1/4}\sqrt{\rho\sigma}}{T^{1/4}} + \frac{(\rho\sigma)^2}{\sqrt{L\delta_0 TB}} + \frac{(L\delta_0)^{3/4}B}{\sqrt{\rho\sigma}T^{3/4}}}_{:=U_{\|\cdot\|}}. \tag{4}$$

The native guarantees for NSGD and SignSGD are given in $\|\cdot\|_2$ and $\|\cdot\|_1$ norms, respectively. Since $\|x\|_2 \le \|x\|_1$ for all $x \in \mathbb{R}^d$, then we have

$$\min_{1\le k\le K} \mathbb{E}[\|\nabla f(x_k)\|_2] \le U_2 \quad \text{for NSGD,}$$
$$\min_{1\le k\le K} \mathbb{E}[\|\nabla f(x_k)\|_2] \le \min_{1\le k\le K} \mathbb{E}[\|\nabla f(x_k)\|_1] \le U_\infty \quad \text{for SignSGD.}$$

C.3 Comparison of the bounds for NSGD and SignSGD

We now compare the surrogate bounds Equation (4) for NSGD and SignSGD with momentum using specific constant choices. For the ℓ_2 norm, we have $L = L_2$ and $\rho = 1$, while for the ℓ_∞, it is $L = L_\infty$ and $\rho = \sqrt{d}$. Since we solve the same problem, both methods use the same noise σ. Therefore, the difference in the surrogate bounds for NSGD and SignSGD with momentum is[6]

$$\begin{aligned}\Delta_U &= U_\infty - U_2 \\ &= \frac{(\sigma^2\delta_0)^{1/4}\left((L_\infty d)^{1/4} - L_2^{1/4}\right)}{T^{1/4}} + \frac{\sigma^2}{\sqrt{\delta_0 TB}}\left(\frac{d}{\sqrt{L_\infty}} - \frac{1}{\sqrt{L_2}}\right) + \frac{\delta_0^{3/4}B}{\sqrt{\sigma}T^{3/4}}\left(\frac{L_\infty^{3/4}}{d^{1/4}} - L_2^{3/4}\right).\end{aligned} \tag{5}$$

The derivative of the surrogate gap with respect to the batch size B is

$$\frac{\partial \Delta_U}{\partial B} = -\frac{\sigma^2}{2\sqrt{\delta_0 TB^3}}\left(\frac{d}{\sqrt{L_\infty}} - \frac{1}{\sqrt{L_2}}\right) + \frac{\delta_0^{3/4}}{\sqrt{\sigma}T^{3/4}}\left(\frac{L_\infty^{3/4}}{d^{1/4}} - L_2^{3/4}\right). \tag{6}$$

We observe that the surrogate gap between the two optimizers and how it changes with batch size depends on the interaction of the geometry L and dimensionality of the problem d. Different relations between L_2, L_∞, and d can lead to different outcome, as well as different dynamics with the batch size. Since that $L_2 \le L_\infty \le dL_2$, let us consider several cases below.

$1 < L_\infty/L_2 < d^{1/3}$. In this regime, we have

$$\Delta_U \propto \frac{1}{T^{1/4}} + \frac{1}{\sqrt{TB}} - \frac{B}{T^{3/4}}, \quad \frac{\partial \Delta_U}{\partial B} \propto -\frac{1}{\sqrt{TB^3}} - \frac{1}{T^{3/4}}. \tag{7}$$

We observe that the derivative of the surrogate gap Δ_U is always negative. Therefore, if $\Delta_U < 0$ at $B = 1$, SignSGD is a better choice starting from $B = 1$ and onwards. If the $\Delta_U > 0$ at $B = 1$, there is a *crossover* from NSGD to SignSGD being better. Note that Δ_U depends on the relation between L_2, L_∞, d, as well as σ and δ_0, which serves as additional support that model, dataset, and gradient noise all jointly affect the gap.

Most of the numerical results from Figure 1 lie in this regime, since we observe a crossover from SGD to Adam being better. Moreover, for the Genomics experiment (§B.5), Adam is better than SGD even at $B = 1$, which is also possible according to the convergence analysis.

[6] We denote by U_2 and U_∞ the bounds obtained from Equation (4) with the norm chosen as ℓ_2 and ℓ_∞.

$\mathbf{d^{1/3} < L_\infty/L_2 < d.}$ In this regime, we have

$$\Delta_U \propto \frac{1}{T^{1/4}} + \frac{1}{\sqrt{TB}} + \frac{B}{T^{3/4}}, \quad \frac{\partial \Delta_U}{\partial B} \propto -\frac{1}{\sqrt{TB^3}} + \frac{1}{T^{3/4}}. \tag{8}$$

The relation between L_2, L_∞, and d implies that the Δ_U is always positive, i.e., NSGD is better than SignSGD, and there exists a batch size $B \propto T^{1/6}$ when the Δ_U is the smallest. In our numerical result, this case might be linked to ResNet50/ImageNet21K setup, where we observe that SGD remains better than Adam even at a large batch size.

C.4 Power-law spectrum

We now use a simple stochastic quadratic model to make the geometry ratio $r = L_\infty/L_2$ more explicit. This lets us relate the regimes above to the Hessian spectrum. The model we consider is

$$f(x, \xi) = \frac{1}{2} x^\top \operatorname{diag}(\lambda_1, \ldots, \lambda_d) x + \langle \xi, x \rangle,$$

where $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_d|$, $\mathbb{E}[\xi] = 0$, and $\mathbb{E}[\|\xi\|^2] \leq \sigma^2/B$. For this function, $L_2 = |\lambda_1|$, while $L_\infty = \sum_{i=1}^d |\lambda_i|$. To understand how the smoothness constants scale with dimension, we assume a power-law spectrum $|\lambda_j| \propto \frac{|\lambda_1|}{j^\beta}$ [Kunstner and Bach, 2025]. For $d \gg 1$, this gives

$$L_\infty \propto |\lambda_1| \times \begin{cases} d^{1-\beta}, & \text{if } \beta < 1, \\ \log d, & \text{if } \beta = 1, \\ 1, & \text{if } \beta > 1. \end{cases}$$

Equivalently, we have $r \propto d^{1-\beta}$ for $\beta < 1$, $r \propto \log d$ for $\beta = 1$, and $r \propto 1$ for $\beta > 1$. Hence, if $\beta < 2/3$, then $r \gtrsim d^{1/3}$, which corresponds to a regime in which NSGD is favored. In this sense, a relatively homogeneous Hessian spectrum can make NSGD preferable. By contrast, when $\beta > 2/3$, we have $r \lesssim d^{1/3}$, so the preferred optimizer may depend on the batch size. In this case, SignSGD may be favored for all batch sizes, or a batch-size-dependent crossover may occur. Because our analysis is stochastic and takes into account not only the Hessian heterogeneity but also the gradient noise, this behavior differs from the deterministic-regime conclusions of Kunstner and Bach [2025].

This example illustrates that Hessian heterogeneity plays an important role in the performance of an optimizer. At the same time, the best optimizer is not determined by the spectrum alone: it also depends on the batch size and the stochastic gradient noise, as discussed in the previous sections.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Each section dives deep into the claims.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See the last Section.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: This paper empirically revisits and extends prior works on Adam–SGD gap under realistic state-of-the-art settings.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: See Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: See Section 3 and the Appendix.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: See Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We do not report seed-level error bars; Instead, each experiment shows a consistent trend across tens to hundreds sweep of learning rate, momentum, and batch size.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See Appendix. Our results are reproducible on a single A100-40GB machine, where the runtimes vary from a several hours to several days depending on the batch size and number of training samples. We estimate a total of 20k A100 GPU hours to reproduce the results, including the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: .

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This work is on fundamental research and its possible societal impact is too vague to be considered.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: .

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: See Appendix.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: .

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: .

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: .

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: .

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.