GradPower: Powering Gradients for Faster Language Model Pre-Training

Jinbo Wang [*1] **Mingze Wang** [*†1] **Jiaqi Zhang** [†2] **Wei Wang** [2]
Peng Pei [2] **Xunliang Cai** [2] **Weinan E** [1 3 4] **Lei Wu** [†1 3 4]

Abstract

We propose **GradPower**, a lightweight gradient-transformation technique for accelerating language model pre-training. Given a gradient vector $\boldsymbol{g} = (g_i)_i$, GradPower first applies the elementwise `sign-power` transformation $\varphi_p(\boldsymbol{g}) = (\mathrm{sign}(g_i)|g_i|^p)_i$ for a fixed $p > 0$, and then feeds the transformed gradient into a base optimizer. Notably, GradPower requires only a **single-line code change** and no modifications to the base optimizer's internal logic, including the hyperparameters. When applied to AdamW (termed **AdamWPower**), GradPower consistently achieves lower terminal loss across diverse architectures (LLaMA, Qwen2MoE), parameter scales (66M to 2B), datasets (C4, OpenWebText), and learning-rate schedules (cosine, warmup-stable-decay). The most pronounced gains are observed when training modern mixture-of-experts models with warmup-stable-decay schedules. GradPower also integrates seamlessly with other state-of-the-art optimizers, such as Muon, yielding further improvements. Finally, we provide theoretical analyses that reveal the underlying mechanism of GradPower and highlight the influence of gradient noise.

1. Introduction

Large language models (LLMs) have revolutionized modern artificial intelligence (Brown et al., 2020; OpenAI et al., 2023; DeepSeek-AI et al., 2024). However, pre-training LLMs is computationally intensive due to the massive scale of model size and training data. Improving the pre-training efficiency has thus become a primary objective in the continued scaling of LLMs. Among the factors affecting efficiency, the choice of optimizer is critical. In practice, the Adam optimizer (Kingma & Ba, 2014; Loshchilov & Hutter, 2017) has emerged as the de facto choice in most LLM pre-training pipelines, owing to its adaptive learning rate features (Zhang et al., 2024; Kunstner et al., 2024).

To further accelerate AdamW, several approaches have been proposed to refine or simplify its moment estimation (Xie et al., 2024; Pagliardini et al., 2025; Chen et al., 2024b; Liu et al., 2025b; Zhang et al., 2025). Other strategies modify the update rule directly, such as direction correction (Wang et al., 2024; Liang et al., 2024), incorporating curvature information (Liu et al., 2024; Wang et al., 2025), or applying matrix-based preconditioning (Jordan et al., 2024; Liu et al., 2025a). While these methods often deliver tangible gains, they typically require substantial modifications to the existing training pipeline and careful extra hyperparameter tuning, which hinders their practical adoption.

In contrast to these intrusive modifications, we instead propose a lightweight, plug-in approach by revisiting the core component of optimization: the *gradient itself*. We apply a simple elementwise transformation to the gradient vector – enhancing its informativeness while leaving the base optimizer entirely unchanged. This design preserves compatibility with existing training pipelines and avoids additional tuning burden. Specifically, **our contributions** are as follows:

- **Our approach.** We propose GradPower, a simple but effective approach for boosting the convergence of general gradient-based optimizers. Specifically, given a raw gradient $\boldsymbol{g} = (g_i)_i \in \mathbb{R}^d$ and a fixed $p > 0$, we define the **powered gradient** as

$$\begin{aligned}\varphi_p(\boldsymbol{g}) &:= |\boldsymbol{g}|^p \,\mathrm{sign}(\boldsymbol{g}) \\ &= \left(|g_1|^p \,\mathrm{sign}(g_1), \ldots, |g_d|^p \,\mathrm{sign}(g_d)\right)^\top .\end{aligned} \tag{1}$$

  GradPower applies this powered gradient to the base optimizer, preserving its original structure.

- **Empirical performance.** We first evaluate the effectiveness of GradPower by integrating it into AdamW,

[*]Equal contribution [1]School of Mathematical Sciences, Peking University [2]Meituan, Beijing [3]Center for Machine Learning Research, Peking University [4]AI for Science Institute, Beijing, China. Correspondence to: Mingze Wang <mingzewang@stu.pku.edu.cn>, Jiaqi Zhang <zhangjiaqi39@meituan.com>, Lei Wu <leiwu@math.pku.edu.cn>.

Proceedings of the 43rd International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

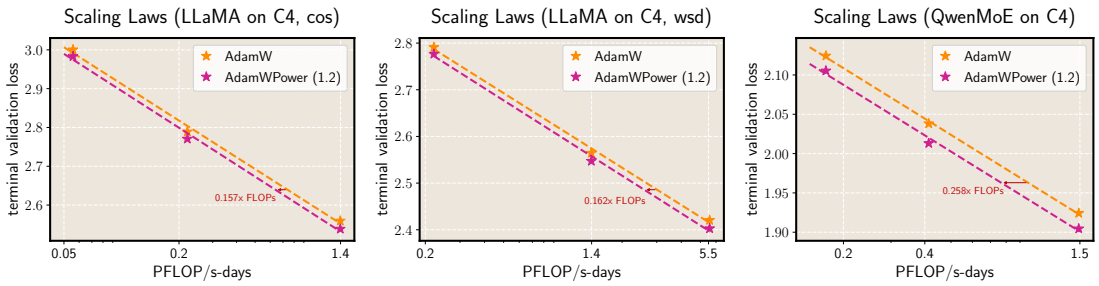

Figure 1. Scaling-law comparison of AdamWPower and AdamW on the C4 dataset for dense LLaMA models and mixture-of-experts Qwen2MoE models.

termed **AdamWPower**. We test its performance across a broad LLM pre-training landscape: **dense** models (LLaMA (Touvron et al., 2023)) and **mixture-of-experts** models (Qwen2MoE (Yang et al., 2024)), ranging from **66M** to **2B** parameters, using the C4 and OpenWebText corpora, and under both cosine-decay (`cos`) and warmup-stable-decay (`wsd`) learning-rate schedules. Across all settings, AdamWPower consistently achieves lower terminal loss and exhibits more favorable scaling laws compared to vanilla AdamW (see Figure 1), demonstrating its potential for improved scalability to larger models. Notably, the performance gains are most significant for modern MoE architectures and `wsd` schedules.

Furthermore, we show that GradPower can be also seamlessly integrated with other state-of-the-art optimizers, such as Muon (Jordan et al., 2024; Liu et al., 2025a) and Blockwise LR (Wang et al., 2025), yielding additional performance improvements.

- **Theoretical analysis.** 1) Recent analyses suggest that steady progress along flat "river-like" directions is crucial for reducing loss in LLM pre-training (Wang et al., 2024; Wen et al., 2025). We show that AdamWPower amplifies these directions, thereby accelerating optimization. 2) Moreover, for general *smooth non-convex objectives*, we prove that augmenting adaptive optimizers (e.g., AdaGrad) with GradPower strictly accelerates their convergence in both low- and high-noise regimes, supporting the intuitions developed in Section 3.

Notations. For $\{a_s\}_{s=1}^{\infty}$, its β-exponential moving average at time t is denoted as $\mathtt{EMA}_\beta(\{a_s\}_1^t) := (\sum_{s=1}^t \beta^{t-s} a_s)/(\sum_{s=1}^t \beta^{t-s})$. For $g \in \mathbb{R}$ and $p \in \mathbb{R}_+$, we denote $g^p := |g|^p \operatorname{sign}(g)$; for a vector $\boldsymbol{g}$, the notation $\boldsymbol{g}^p$ denotes element-wise application of this transformation. For simplicity, we use *a.s.* to denote "almost surely", and use *w.r.t.* to denote "with respect to". We use standard big-O notations $\mathcal{O}(\cdot), \Omega(\cdot), \Theta(\cdot)$ to hide problem-independent constants, and use $o(\cdot)$ to denote the infinitesimal. Let $\|\cdot\|_q$ denote the ℓ_q norm for vectors for a $q \geqslant 1$. We denote $[n] = \{1, \cdots, n\}$ for an integer $n \in \mathbb{N}_+$.

2. Related Works

Optimizer design in LLM pre-training. In LLM pre-training, Adam (Kingma & Ba, 2014) has become the de facto optimizer. Recent efforts to improve its efficiency focus on two aspects: accelerating convergence and reducing memory usage. Techniques for *accelerating convergence* include introducing curvature information (Liu et al., 2024; Wang et al., 2024; 2025), mixing momentum (Xie et al., 2024; Pagliardini et al., 2025), variance reduction (Yuan et al., 2024), cautious update (Liang et al., 2024), and applying matrix-based preconditioners (Jordan et al., 2024; Vyas et al., 2024). *Memory-efficient* techniques include reducing the moments usage in Adam (Zhang et al., 2025), sign-based updates (Chen et al., 2024b; Liu et al., 2025b), low precision optimizer states (Dettmers et al., 2022; Li et al., 2023), low-rank approximation (Hu et al., 2022; Zhao et al., 2024; Chen et al., 2024a), and structured learning rates (Zhu et al., 2025). Among these, Muon (Jordan et al., 2024) stands out for improving both convergence and memory usage and showing strong scalability (Liu et al., 2025a). *In contrast*, our method, GradPower, improves training efficiency without altering the base optimizer's internal updates. Notably, GradPower is orthogonal and complementary to the methods above: it can serve as a lightweight plug-in that further enhances existing optimizers.

The fast-slow dynamics in neural network training. Recent works (Wu et al., 2018; Jastrzebski et al., 2020; Cohen et al., 2020; 2022) show that neural network training typically occurs at the so-called Edge of Stability (EoS) stage. This regime is characterized by the optimizer exhibiting oscillatory behavior along sharp directions without divergence, while steadily progressing along *flat directions*, leading to loss reduction. Several studies (Wen et al., 2025; Song et al., 2025; Cohen et al., 2025; Wang et al., 2024; 2025; Zhu et al., 2026) have emphasized the importance of the slow dynamics along flat directions (referred to as stable direction in Wang et al. (2024), river directions by Wen et al. (2025) and bulk directions by Song et al. (2025)), in *reducing total loss*. Moreover, Wen et al. (2025) further showed that this picture is crucial for understanding the behavior of LLM

pre-training. In addition, Fig.3 in (Wen et al., 2025) and Fig.8 (Song et al., 2025) suggest that, the optimizer's trajectory within flat directions tends to *remain aligned* for a period of time.

The terminology of flat directions. In classical optimization theory, flat directions refer to Hessian eigenvectors associated with small eigenvalues. However, *our flat directions correspond to small stochastic gradient directions*. Our usage of flat directions is approximate and follows two lines of prior works: *(i) The anisotropy of gradient noise closely reflects the Hessian's curvature structure.* A line of works (Zhu et al., 2019; Wu et al., 2020; Mori et al., 2022; Wu et al., 2022; Wang & Wu, 2023) have established that *directions with small Hessian curvature exhibit low gradient-noise variance*, while directions with large Hessian curvature exhibit high gradient-noise variance. *(ii) LLM pre-training typically operates in a noise-dominated regime, where the stochastic gradient is largely governed by the gradient noise*. The pretraining dataset is massive, and each mini-batch constitutes only a tiny fraction of it. As a result, the stochastic gradient is dominated by sampling noise. Therefore, relative differences across coordinates in stochastic gradient g primarily arise from differences in the noise ξ, rather than differences in the true gradient $\nabla\mathcal{L}$ (Note: $\xi = g - \nabla\mathcal{L}$). Moreover, prior empirical studies in deep-learning settings show that as training progresses, the gradient noise constitutes the dominant component of the stochastic gradient, and the dynamics occurs in the noise-dominated regime (McCandlish et al., 2018; Sun et al., 2023). For LLMs, direct empirical verification is challenging because computing the full-batch gradient is infeasible, but the above intuition suggests that LLM training should also lie in this noise-dominated regime. *Therefore*, our flat directions approximately correspond to small stochastic gradient directions.

3. The GradPower Approach

Let $\boldsymbol{g}_t \in \mathbb{R}^d$ denote the stochastic gradient at step t. A gradient-based optimizer can be expressed as $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \mathcal{Q}(\boldsymbol{g}_1, \cdots, \boldsymbol{g}_t)$, where η_t is learning rate and $\mathcal{Q}$ denotes update rule.

A unified view of preconditioning. In practice, raw gradients may not be sufficiently informative or stable, and thus, it is common to transform the gradients before applying the update rule. This leads to the general form of *preconditioned optimizers*:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \mathcal{Q}\left(\varphi(\boldsymbol{g}_1), \cdots, \varphi(\boldsymbol{g}_t)\right) \tag{2}$$

where φ denotes a transformation (or preconditioning) applied to each gradient.

To *avoid computational overhead*, we restrict attention to *elementwise* transformations. For a gradient vector $\boldsymbol{g} = (g_i)_i \in \mathbb{R}^d$, we consider $\varphi(\boldsymbol{g}) := (\varphi(g_1), \ldots, \varphi(g_d))^\top \in \mathbb{R}^d$, where $\varphi : \mathbb{R} \to \mathbb{R}$ is a scalar nonlinearity applied coordinate-wise. The function φ is designed to enhance the informativeness of the raw gradient. The simplest choice is a linear transformation $\varphi(z) = cz$ with $c \in \mathbb{R}$. However, as shown in Appendix C, such linear preconditioners may exhibit limited effectiveness when used in modern optimizers for LLM pre-training. In this work, we explore **nonlinear preconditioning** as an alternative.

The GradPower family. Empirically, we find that a simple power transformation already yields non-trivial improvements in LLM pre-training. Specifically, we define the `sign-power` transformation $\varphi_p : \mathbb{R} \to \mathbb{R}$ with exponent $p > 0$ as

$$\varphi_p(z) = |z|^p \mathrm{sign}(z).$$

The powered gradient is shown in Equation (1). Incorporating this transformation into AdamW yields a new optimizer, **AdamWPower**, described in Algorithm 1. Remarkably, AdamWPower introduces only one additional line of code compared to standard AdamW.

In the following sections, we first present empirical evidence demonstrating the effectiveness of AdamWPower, followed by a theoretical analysis that sheds light on its underlying mechanisms.

Algorithm 1 AdamWPower

Given hyperparameters $\beta_1, \beta_2, \epsilon, \lambda$; Learning rates $\{\eta_t\}_{t=1}^T$; power $p \in \mathbb{R}_+$;
Initialize $\boldsymbol{\theta}_0 \in \mathbb{R}^d$, first momentum $\boldsymbol{m}_0 \leftarrow \mathbf{0}$, second momentum $\boldsymbol{v}_0 \leftarrow \mathbf{0}$;
for $t = 1, \ldots, T$ **do**
  Compute mini-batch gradient $\boldsymbol{g}_t$;
  GradPower: compute powered gradient $\boldsymbol{g}_t \leftarrow |\boldsymbol{g}_t|^p \,\mathrm{sign}(\boldsymbol{g}_t)$ using Eq. (1);
  $\boldsymbol{m}_t \leftarrow \beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1)\boldsymbol{g}_t$;
  $\hat{\boldsymbol{m}}_t \leftarrow \boldsymbol{m}_t / (1-\beta_1^t)$;
  $\boldsymbol{v}_t \leftarrow \beta_2 \boldsymbol{v}_{t-1} + (1-\beta_2)\boldsymbol{g}_t^2$;
  $\hat{\boldsymbol{v}}_t \leftarrow \boldsymbol{v}_t / (1-\beta_2^t)$;
  $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \eta_t \left(\hat{\boldsymbol{m}}_t / \left(\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon\right) + \lambda \boldsymbol{\theta}_{t-1} \right)$;
end for
Output: Optimized parameters $\boldsymbol{\theta}_T$.

4. Empirical Evaluation

4.1. Experimental Setup

We evaluate AdamWPower for the task of LLM pre-training across a range of model architectures, parameter scales,

datasets, and learning rate (LR) schedulers. The main experimental configurations are summarized below, while additional implementation details are provided in Appendix B.

- **Models.** We consider two widely used LLM architectures: **LLaMA** (dense) models (Touvron et al., 2023) and **Qwen2MoE** (MoE) models (Yang et al., 2024). We experiment with model sizes ranging **from 66M to 2B** parameters.
- **Datasets.** We evaluate our methods on the **Colossal Clean Crawled Corpus (C4)** dataset (Raffel et al., 2020)[1] and **OpenWebText** dataset (Gokaslan & Cohen, 2019)[2]. For pre-training on C4, we follow the setup of Wortsman et al. (2024); Zhao et al. (2025), using a batch size of 512. The total number of training tokens is set to be approximately 20 times the number of model parameters, in accordance with the Chinchilla scaling law (Hoffmann et al., 2022).
- **LR schedulers.** We evaluate two popular LR scheduling strategies: (i) `cos`: a linear warm-up to peak `lr_max`, followed by cosine decay to a terminal LR `lr_min`. (ii) `wsd` (warmup-stable-decay scheduler) (Zhai et al., 2022; Hu et al., 2024; Hägele et al., 2024): a linear warm-up LR to peak `lr_max`, followed by a stable phase where LR remains at `lr_max` (up to 80% of the total training steps), and then a linear decay to `lr_min`. It should be noticed that `wsd` introduces a non-traditional loss curve: slowly decrease during the stable phase and suddenly drop during the final decay phase.

We further evaluate our method on vision tasks, and report detailed implementation settings in Appendix B.

AdamW baselines. We use the standard Adam optimizer (with decoupled weight decay) as the baseline in most experiments (except Section 4.4). The baseline is configured with hyperparameters $\beta_1 = 0.9$, $\beta_2 = 0.95$, weight decay $\lambda = 0.1$, and gradient clipping threshold of 1.0, following protocols used in LLaMA pre-training (Touvron et al., 2023). For each experiment, we first tune `lr_max` over {`1e-4, 2e-4, 3e-4, 6e-4, 1e-3, 1.5e-3`} to be optimal for AdamW, and the baselines are trained using these optimal `lr_max`'s. Details can be found in Appendix B.

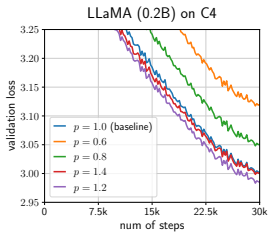

Figure 2. Pre-training LLaMA (0.2B) on C4 using AdamWPower with different power p's. The optimal power is 1.2.

The tuning of power p and its transferability. We only tune the power p in a single small-scale experiment: pre-training LLaMA (0.2B) on C4. As shown in Figure 2, the tuned power is 1.2. Interestingly, this aligns with the optimal power observed in the high-noise regime of our illustrative toy example (Figure 7). We adopt $p = 1.2$ as the default in most experiments (except Section 4.6). Importantly, the power proves to be **highly robust**: AdamWPower with $p = 1.2$ **consistently outperforms** AdamW and exhibits better scaling laws, across model types, model sizes, datasets, and LR schedulers.

Lightweight wall-clock time overhead. We empirically measured wall-clock time using LLaMA (0.25B) on OpenWebText with batch size 480 and observed that AdamWPower introduces only a 0.4% overhead per step compared to AdamW, with 0.7565s versus 0.7534s, almost negligible in practice, confirming its lightweight nature.

4.2. Results on Dense Models

Main findings. Figure 3 compares the performance of AdamWPower (with $p = 1.2$) to that of vanilla AdamW across a range of settings, including LLaMA models of size 66M, 0.2B, 0.4B, 1B and 2B; both `cos` and `wsd` LR schedulers; and the C4 and OpenWebText datasets. Across all experiments, AdamWPower **consistently achieves a lower terminal loss** than well-tuned AdamW baseline. To further assess its scalability, we visualize the **scaling laws** of AdamWPower versus AdamW in Figure 1 (left and middle). We observe that the performance gain of AdamWPower over AdamW remains consistent across a wide range of model scales, **highlighting the potential scalability of AdamWPower**.

Evaluation on downstream tasks. Additionally, we also evaluate zero-shot performances of our method on common benchmarks including ARC (Yadav et al., 2019), PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), OBQA (Mihaylov et al., 2018), WinoGrande (Sakaguchi et al., 2021), using the lm-evaluation-harness codebase (Gao et al., 2024). The results are reported in Table 1. The model pre-trained with AdamWPower outperforms that trained with AdamW on five out of six tasks, as well as on the overall average score, demonstrating improved downstream performance under the same number of pre-training steps.

4.3. Results on MoE Models

Mixture-of-experts (MoE) architectures have become a key design choice in modern LLMs, as exemplified by Qwen-

[1] A large-scale public language dataset, widely used for LLM pre-training such as T5 (Raffel et al., 2020), and prior pre-training studies (Zhao et al., 2024; 2025).

[2] An open-source recreation of the WebText corpus, commonly used in pre-training models such as RoBERTa (Liu et al., 2019), GPT-2, and NanoGPT (Karpathy, 2022).

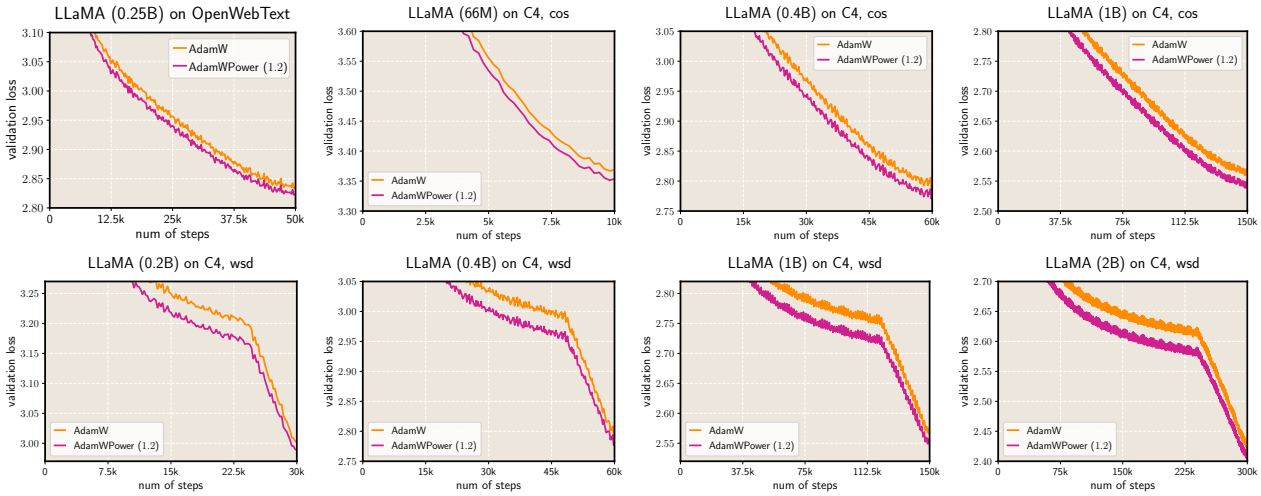

Figure 3. AdamWPower ($p = 1.2$) consistently outperforms AdamW in LLaMA pre-training tasks across a range of model sizes, datasets and LR schedulers.

Table 1. The evaluation results of LLaMA (2B) models pre-trained using the C4 dataset. The best scores in each column are bolded.

METHOD	ARC-E	ARC-C	PIQA	HELLASWAG	OBQA	WINOGRANDE	AVG.
AdamW	60.02	**26.45**	73.56	44.65	24.80	56.83	47.72
AdamWPower (1.2)	**60.35**	26.28	**73.61**	**44.93**	**25.00**	**59.43**	**48.26**

2.5 (Qwen et al., 2024) and DeepSeek-V3 (DeepSeek-AI et al., 2024). Compared to dense models, MoE models often exhibit greater training instability. To assess whether the benefits of AdamWPower extend to MoE models, we conduct experiments on Qwen2MoE (Yang et al., 2024).

Main findings. Figure 4 compares the performance of AdamWPower ($p = 1.2$) and standard AdamW for pre-training Qwen2MoE models of sizes 0.5B, 1B, and 2B on the C4 dataset, using the `wsd` scheduler. Across all settings, AdamWPower **consistently achieves a lower terminal loss** than the well-tuned AdamW baseline. To further examine scaling behavior, Figure 1 (right) visualizes the **scaling laws** of AdamWPower versus AdamW during Qwen2MoE pre-training. The performance gap between the two optimizers remains stable across model scales, with the corresponding scaling curves remaining nearly parallel – *suggesting that the gains offered by AdamWPower may persist at larger model scales*.

Special potential for MoE models. Additionally, we observe three surprising phenomena, suggesting that AdamWPower may offer unique advantages for MoE model training:

- Although the power $p = 1.2$ was originally tuned for LLaMA, it generalizes well to Qwen2MoE models without further tuning. (it is likely that an even better p exists for MoE-specific training.) Remarkably, the absolute improvement achieved by AdamWPower on Qwen2MoE-2B (0.028) is **more significant** than that on LLaMA-2B (0.022). Notably, Qwen2MoE-2B reaches a much lower loss (1.93) compared to LLaMA-2B (2.43), making further improvements more challenging – yet AdamWPower still yields remarkable gains.
- AdamWPower also exhibits improved **training stability**, reducing the occurrence of loss spikes seen with AdamW. This effect is particularly visible in the 1B and 2B curves in Figure 4 (middle, right). Based on recent understanding in Section 2, the fast vibrations along the sharp (valley) directions mainly decide the training (in)stability. We *hypothesize* that the gradient power transformation in AdamWPower may help suppress the vibrations along these directions. We leave a detailed investigation of this phenomenon to future work.
- The `wsd` scheduler has become increasingly popular in recent LLM pre-training (DeepSeek-AI et al., 2024; Hägele et al., 2024), always taking a long stable phase. We observe that the advantage of AdamWPower **gradually increases** *throughout the LR stable phase*. This suggests that AdamWPower may be particularly suited for modern training pipelines that adopt `wsd` schedules.

4.4. Compatibility with Other Optimizers

As discussed in Section 2, several optimizers have recently been proposed to enhance LLM pre-training. While AdamWPower has demonstrated superiority over AdamW in both dense and MoE models, we now ask: *can GradPower also improve the performance of other state-of-the-art optimizers?*

To investigate this, we focus on two representative optimiz-

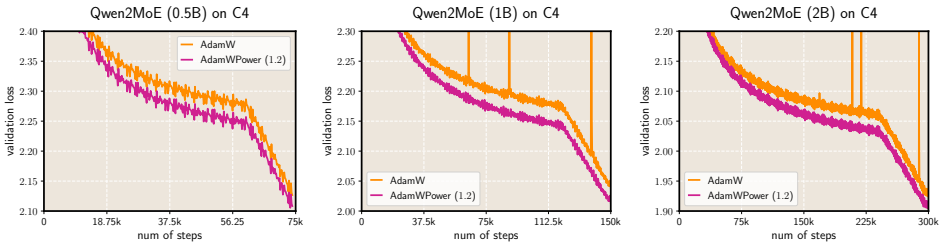

Figure 4. AdamWPower ($p = 1.2$) consistently outperforms AdamW in Qwen2MoE pre-training tasks on C4, across varying model sizes. The learning rate schedule is `wsd`.

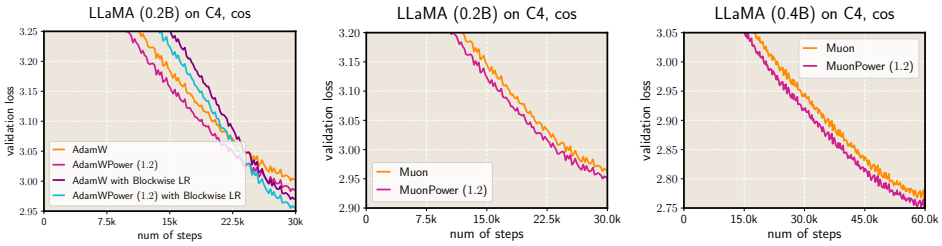

Figure 5. (left) AdamWPower with Blockwise LR outperforms both AdamWPower and AdamW with Blockwise LR in LLaMA pre-training. (middle, right) MuonPower (with $p = 1.2$) outperforms Muon in LLaMA pre-training.

ers: AdamW with **Blockwise LR** (Wang et al., 2025) and **Muon optimizer** (Jordan et al., 2024; Liu et al., 2025a). Blockwise LR assigns separate learning rates to different Transformer blocks and has shown substantial improvements over standard AdamW. Muon, on the other hand, breaks away from the AdamW framework entirely and has recently been shown to achieve better scaling laws than AdamW (Liu et al., 2025a). We refer to the application of GradPower to Muon as **MuonPower**.

The results shown in Figure 5, highlight two key findings. **(i) AdamWPower with Blockwise LR** achieves a lower terminal loss than both AdamWPower and AdamW with Blockwise LR alone. Notably, the observed improvement (0.045) is *nearly the sum* of the gains from AdamWPower alone (0.015) and Blockwise LR alone (0.03), suggesting that their benefits are largely orthogonal. **(ii) MuonPower** ($p = 1.2$), the GradPower-augmented Muon, also outperforms the well-tuned Muon baseline. These results demonstrate the versatility of GradPower as a general enhancement that can be seamlessly integrated into other optimizers.

4.5. Interaction between GradPower and Clipping

Gradient clipping is a standard technique for stabilizing training in LLM pre-training (Shoeybi et al., 2019; Touvron et al., 2023). In our default implementation, gradient clipping is applied before the GradPower transformation.

Here, we examine whether the *ordering between gradient clipping and GradPower* impacts performance. We conduct a controlled experiment based on the setting of Figure 2 on LLaMA (0.2B). Specifically, we switch the order of gradient clipping and the GradPower transformation. We refer to this variant as AdamWPower-II, in contrast to the standard AdamWPower implementation. As shown in Figure 6 (left), the training curves are nearly indistinguishable throughout training, indicating that the ordering of clipping and GradPower has negligible effect on convergence behavior. Crucially, both procedures ensure that the final gradient updates remain bounded, preserving training stability.

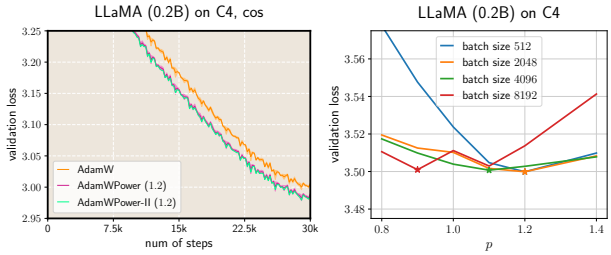

Figure 6. (left) Comparison of gradient clipping before vs. after applying GradPower in LLaMA pre-training. The performance is virtually indistinguishable. The shaded regions in the plots denote the standard deviation. (right) The influence of batch size for the optimal power p in LLaMA pre-training tasks.

4.6. Influence of Batch Size

Finally, we investigate how batch size influences the performance of GradPower. Batch size plays a critical role in deep learning, with larger batch sizes producing lower gradient noise and more accurate gradient (Keskar et al., 2017; McCandlish et al., 2018; Wang et al., 2026).

Unlike the previous experimental settings, here we conduct the experiments on C4 dataset, varying the batch size from the standard 512 up to 8192. For each batch size, we evaluate AdamWPower with multiple values of p, and record

their validation loss of when the optimal validation loss reaches approximately 3.5. The experimental details are provided in Appendix B.

Main findings. The results, shown in Figure 6 (right), demonstrate a clear trend: the optimal power p decreases as batch size increases, i.e., as the gradient noise level decreases. This finding reveals a strong correlation between batch size and the optimal power p in AdamWPower. For standard (small) batch sizes, the optimal power p tends to be greater than 1; in contrast, for large batch sizes, the optimal power p might fall below 1.

Vision tasks. We also conduct the experiments using ResNet-34 model (He et al., 2016) on CIFAR-10 dataset (Krizhevsky & Hinton, 2009), varying the batch size from 32 to 128. The results in Table 2 further validates above point. Moreover, it demonstrates the generalizability of our method beyond language model pre-training.

In the next section, we provide a theoretical explanation for this phenomenon.

Table 2. The influence of batch size for the optimal power p in vision tasks.

batch size	128	64	32
$p = 0.8$	**94.35**	94.22	94.04
$p = 0.85$	94.27	**94.40**	94.07
$p = 0.9$	94.22	94.22	94.15
$p = 1.0$	93.98	94.10	**94.30**
$p = 1.1$	93.38	93.97	94.25
$p = 1.2$	93.15	93.77	93.85

5. Theoretical Insights

5.1. An Illustrative Case Study

This subsection investigates a phenomenological example, both theoretically and empirically, to illustrate how varying the power p in AdamWPower affects the update magnitude. Motivated by the empirical findings in Section 4.6, which show that batch size (gradient noise) affects the optimal value of p, we study our example under varying signal-to-noise regimes.

Slow dynamics along flat directions. As discussed in Section 2, recent studies have revealed that the landscape in LLM training can be decomposed into flat and sharp directions, also referred to as river and valley components (Wen et al., 2025). The loss along river component typically determines the loss at the bottom of the landscape. Along these *flat directions*, the optimizer tends to make *slow but steady progress*, and appears to *remain aligned* for a period of time.

The anisotropy of stochastic gradients. Several works in Section 2 show that stochastic gradients closely reflect the curvature structures: *flat (low-curvature) directions* exhibit *small stochastic gradient magnitudes*, whereas sharp (high-curvature) directions exhibit larger stochastic gradients.

Motivated by these insights, we consider a one-dimensional example to study whether varying p in AdamWPower can *accelerate these slow dynamics along the flat directions*, thereby leading to more efficient loss descent.

Example 5.1. *For simplicity, consider a 1-dimensional flat direction. Let the stochastic gradients at time $t \in \mathbb{N}$ follow $g_t \overset{\text{i.i.d.}}{\sim} \mathrm{Unif}(\mu - \sigma, \mu + \sigma)$, where $0 < \mu, \sigma \ll 1$[3]. Here, μ reflects the full-batch gradient, and σ captures the stochastic noise level.*

Our goal is to investigate the values of p that maximize the update magnitude $u_t = m_t/(\sqrt{v_t} + \epsilon)$ in AdamWPower (Alg. 1). For simplicity, we set weight decay to 0. We now present both empirical and theoretical analysis.

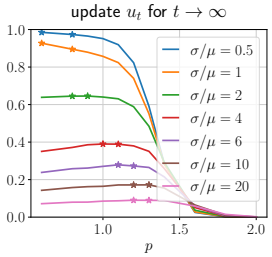

Figure 7. Numerical results for Example 5.1. We plot the value of u_t at $t = 10^6$ for AdamWPower across different p's under varying noise-to-signal ratios. For each curve, the optimal and suboptimal p values are marked with stars. The μ is set to $\mu = 10^{-6}$. Other hyperparameters follow standard values: $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$, and $\lambda = 0$. The learning rate η does not affect the result.

Empirical findings. We begin by numerically simulating the update u_t. The results are presented in Figure 7. Notably, the optimal value of p varies across noise-to-signal regimes, exhibiting two distinct behaviors:

- *Low-noise regime* $\sigma/\mu \leqslant 1$ (blue and orange curves), it is clear that the update magnitude decreases monotonically with increasing p, and the optimal power is small, satisfying $p^\star < 1$.
- *High-noise regime* $\sigma/\mu > 1$, the update magnitude increases and then decreases with increasing p. Moreover, for noise-dominant regime, the optimal power satisfies $p^\star > 1$ (red, purple, brown, and pink curves).

These findings closely align with our empirical results in real-world LLM pre-training tasks in Section 4. As the batch size increases (corresponding to lower gradient noise), the optimal power $p^\star$ decreases accordingly, transitioning from $p^\star > 1$ to $p^\star < 1$, as observed in Section 4.6. Remarkably, the optimal power $p^\star = 1.2$ in the high-noise regime matches the value used across most LLM pre-training experiments in Section 4.

Theoretical analysis. To better understand these interesting behaviors, we theoretically analyze this problem. To

[3]Empirical studies suggest that gradient scales in LLM training are often very small (Huang et al., 2025).

facilitate analytical derivation, we consider the limiting case where $\beta_2 \to 1$, which closely approximates typical settings in practice (e.g., 0.95 or 0.999). We define the limiting update of AdamWPower as:

$$u := \lim_{t\to\infty} \lim_{\beta_2\to 1} u_t, \tag{3}$$

where $u_t = m_t/(\sqrt{v_t}+\epsilon)$, with $m_t = \texttt{EMA}_{\beta_1}(\{g_s^p\}_1^t)$ and $v_t = \texttt{EMA}_{\beta_2}(\{(g_s^p)^2\}_1^t)$. In this limit, we obtain the *closed-form expression*: $u = \mathbb{E}[g^p]/(\sqrt{\mathbb{E}[(g^p)^2]}+\epsilon)$, *a.s.*, $g \sim \mathrm{Unif}(\mu-\sigma, \mu+\sigma)$. This formulation allows explicit computation and facilitates verification of the empirical trends. We present two propositions corresponding to the low-noise and high-noise regimes.

Proposition 5.2 (low-noise regime, $\sigma \ll \mu$)**.** *It holds that* $u = \frac{1+o(1)}{1+\frac{\epsilon}{\mu^p}}$, a.s.. *Letting* $\tilde{u} = \frac{1}{1+\frac{\epsilon}{\mu^p}}$, *we observe that* $\tilde{u}$ *is monotonically decreasing w.r.t.* p.

This proposition quantitatively explains the monotonicity observed in the low-noise regime. Furthermore, it shows that the maximum update is approximately $\frac{1}{1+\epsilon} \approx 1$, achieved in the limit as $p \to 0$. This aligns with Figure 7.

Proposition 5.3 (high-noise regime, $\mu \ll \sigma$)**.** *It holds that* $u = \frac{\mu}{\sigma}\frac{1+o(1)}{\frac{1}{\sqrt{2p+1}}+\frac{\epsilon}{\sigma^p}}$, a.s.. *Letting* $\tilde{u} = \frac{\mu}{\sigma}\frac{1}{\frac{1}{\sqrt{2p+1}}+\frac{\epsilon}{\sigma^p}}$, *we observe the following: If* $\epsilon\log(1/\sigma) < 1$, *then there exists an optimal power* $p^\star$ *such that* $\tilde{u}$ *increases for* $0 < p < p^\star$ *and decreases for* $p > p^\star$. *Moreover, we have a tight estimate:* $p^\star = \Theta\left(\frac{\log(\epsilon\log(1/\sigma))}{\log\sigma}\right)$.

Notably, in practice, ϵ is typically chosen sufficiently small (e.g., $\epsilon \ll \sigma$), ensuring $\frac{\log(\epsilon\log(1/\sigma))}{\log\sigma} > 1$. This again aligns with our empirical observation that $p^\star > 1$ in the high-noise regime.

The intuition behind Proposition 5.3 is as follows. When p is relatively small, the denominator is dominated by $\sqrt{\mathbb{E}[(g^p)^2]}$. Since $g \ll 1$, increasing p reduces both the numerator $\mathbb{E}[g^p]$ and denominator $\sqrt{\mathbb{E}[(g^p)^2]}$. In the high-noise regime, the reduction in the denominator outweighs that in the numerator, resulting in a larger update. In contrast, when p is relatively large, the denominator is dominated by ϵ, and AdamWPower degenerates to SGDPower, where the update is approximately $\mathbb{E}[g^p]/\epsilon$. In this regime, increasing p reduces the update magnitude.

Although the above example is synthetic, it reveals several non-trivial phenomena highly aligned with LLM pre-training tasks, particularly the existence and behavior of the best $p^\star$ across noise-to-signal regimes. These insights deepen our understanding of how GradPower influences the performance of AdamWPower and suggest practical guidance for selecting p.

5.2. Convergence Guarantees

Let $\mathcal{L}: \mathbb{R}^d \to \mathbb{R}$ be a non-convex loss function. For any $\boldsymbol{\theta} \in \mathbb{R}^d$, let $\boldsymbol{g}(\boldsymbol{\theta})$ denote the stochastic gradient satisfying $\mathbb{E}[\boldsymbol{g}(\boldsymbol{\theta})] = \nabla\mathcal{L}(\boldsymbol{\theta})$.

In this subsection, we consider the classical setting of smooth non-convex optimization and investigate the theoretical benefits of applying GradPower within adaptive optimizers. Since the analysis of Adam is technically complex, to gain clear theoretical insights, we instead analyze its predecessor, Adagrad, a foundational adaptive optimization algorithm (Duchi et al., 2011). The update rule of **AdagradPower** (Adagrad using GradPower) is given by:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta\frac{\boldsymbol{g}_t^p}{\sqrt{\boldsymbol{v}_t+\epsilon}}, \quad \boldsymbol{v}_t = \sum_{s=1}^{t}(\boldsymbol{g}_t^p)^2, \tag{4}$$

where the power $p > 0$, and we denote $\boldsymbol{g}_t$ as the stochastic gradient $\boldsymbol{g}(\boldsymbol{\theta}_t)$ for simplicity.

To establish the convergence results, we adopt the following standard assumptions, consistent with Section 2.3 in Défossez et al. (2022).

Assumption 5.4 (Défossez et al. (2022))**.** The following conditions hold:

- $\mathcal{L}$ *is bounded below by* $\mathcal{L}^\star$, i.e., for all $\boldsymbol{\theta} \in \mathbb{R}^d$, $\mathcal{L}(\boldsymbol{\theta}) \geqslant \mathcal{L}^\star$.
- *The loss function is* H*-smooth*, i.e., there exists a constant $H > 0$ such that for all $\boldsymbol{\theta}, \boldsymbol{\theta}' \in \mathbb{R}^d$, $\|\nabla\mathcal{L}(\boldsymbol{\theta}) - \nabla\mathcal{L}(\boldsymbol{\theta}')\|_2 \leqslant H\|\boldsymbol{\theta}-\boldsymbol{\theta}'\|_2$.
- *The* ℓ_∞ *norm of the stochastic gradients is uniformly almost surely bounded*, i.e., there exists a constant $R > 0$ such that for all $\boldsymbol{\theta} \in \mathbb{R}^d$, $\|\boldsymbol{g}(\boldsymbol{\theta})\|_\infty + \epsilon \leqslant R$, *a.s.*.

Under this assumption, the convergence guarantee of Adagrad is well established:

Theorem 5.5 (Adagrad; Theorem 1 in Défossez et al. (2022))**.** *Suppose Assumption 5.4 holds. Let* $\{\boldsymbol{\theta}_t\}_{t=0}^T$ *are trained by* **Adagrad** (4) *with* $p = 1$. *Then for any* $T \in \mathbb{N}$, *we have:*

$$\min_{1\leqslant t\leqslant T}\mathbb{E}\left[\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|_2^2\right] \leqslant \frac{2R(\mathcal{L}(\boldsymbol{\theta}_0)-\mathcal{L}^\star)}{\eta\sqrt{T}} + \frac{Rd(4R+\eta H)\log(1+R^2T/\epsilon)}{\sqrt{T}}. \tag{5}$$

We now study the convergence of AdagradPower in both low-noise and high-noise regimes.

Low-noise regime. We introduce an additional assumption about the noise scale.

Assumption 5.6 (Low-noise regime)**.** There exist constants $p \in (0,1)$ and $c > 0$ such that $\mathbb{E}[g_i^p(\boldsymbol{\theta})]\nabla_i\mathcal{L}(\boldsymbol{\theta}) \geqslant c|\nabla_i\mathcal{L}(\boldsymbol{\theta})|^{p+1}$ holds for all $\boldsymbol{\theta} \in \mathbb{R}^d$ and $i \in [d]$.

This assumption is satisfied in many low-noise scenarios:

Example 5.7. *(I) Deterministic regime (the limit case of low noise): if $g_i(\boldsymbol{\theta}) = \nabla_i\mathcal{L}(\boldsymbol{\theta})$, then Assumption 5.6 holds for all $p \in (0,1)$ with $c = 1$. (II) Uniform distribution: if $g_i \sim \mathrm{Unif}(\nabla_i\mathcal{L} - \sigma, \nabla_i\mathcal{L} + \sigma)$ with $\sigma \ll |\nabla_i\mathcal{L}|$, then Assumption 5.6 holds for all $p \in (0,1)$ as $\mathbb{E}[g_i^p]\nabla_i\mathcal{L} = |\nabla_i\mathcal{L}|^{p+1}(1 + o(\sigma/|\nabla_i\mathcal{L}|)) \geqslant 0.99|\nabla_i\mathcal{L}|^{p+1}$.*

Theorem 5.8 (AdagradPower, low-noise regime)**.** *Suppose Assumption 5.4 and 5.6 hold, as well as $R < 1$[4]. Let $\{\boldsymbol{\theta}_t\}_{t=0}^T$ are trained by* **AdagradPower** *(4), with the power $p \in (0,1)$ as given in Assumption 5.6. Then for any $T \in \mathbb{N}$, we have:*

$$\min_{1\leqslant t\leqslant T} \mathbb{E}\left[\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|_{p+1}^{p+1}\right] \leqslant \mathcal{O}\left(\frac{\log T}{\sqrt{T}}\right). \tag{6}$$

by Jensen, Theorem 5.5 yields the rate for Adagrad: $\min_t \mathbb{E}||\nabla L(\theta_t)||_{p+1}^{p+1} \leqslant O(\log^{(p+1)/2} T/T^{(p+1)/4}), 0 < p < 1$. In comparison, Theorem 5.8 shows that AdagradPower has the rate $O(\log T/T^{1/2})$. AdagradPower achieves a convergence rate $2/(p+1)$ times faster than Adagrad. For Example 5.7, this yields nearly a $2\times$ acceleration for $p \to 0$. This result is consistent with observations in Section 5.1 and 4.6 that the optimal power p for adaptive optimizers is less than 1 in the low-noise regime. The proof is presented in Appendix D.

High-noise regime. We introduce an additional assumption regarding the noise scale:

Assumption 5.9 (High-noise regime)**.** (C1) There exist constants $p > 1, \sigma > 0$ such that $\mathbb{E}[g_i^p(\boldsymbol{\theta})]\nabla_i\mathcal{L}(\boldsymbol{\theta}) \geqslant \sigma|\nabla_i\mathcal{L}(\boldsymbol{\theta})|^2$ holds for all $\boldsymbol{\theta} \in \mathbb{R}^d$ and $i \in [d]$. (C2). It holds that $\sigma > R^{p-1}$.

The first condition asserts that the gradient noise is non-degenerate. The second condition further asserts that the gradient noise is in a high level. Notably, these conditions are naturally satisfied in many high-noise settings:

Example 5.10. *Consider g_i satisfy binary distribution $\mathbb{P}(g_i = \nabla_i\mathcal{L} - \sigma_i) = \mathbb{P}(g_i = \nabla_i\mathcal{L} + \sigma_i) = \frac{1}{2}$. Then for any odd number $p > 1$, $\mathbb{E}[g_i^p]\nabla_i\mathcal{L} \geqslant p\sigma_i^{p-1}|\nabla_i\mathcal{L}|^2$. Thus, (C1) in Assumption 5.9 holds with $\sigma = p\sigma_i^{p-1}$. As for (C2), in high-noise regime with $|\nabla_i\mathcal{L}| \ll \sigma_i$, we have $\frac{R^{p-1}}{\sigma} \leqslant \frac{(|\nabla_i\mathcal{L}|+\sigma_i)^{p-1}}{p\sigma_i^{p-1}} \leqslant \frac{1.01}{p} < 1$.*

Theorem 5.11 (AdagradPower, high-noise regime)**.** *Suppose Assumption 5.4 and 5.9 hold, as well as $R < 1$. Let $\{\boldsymbol{\theta}_t\}_{t=0}^T$ be trained by* **AdagradPower** *(4), with the power $p > 1$ as given in Assumption 5.9. Then for any $T \in \mathbb{N}$, we have:*

$$\min_{1\leqslant t\leqslant T} \mathbb{E}\left[\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|_2^2\right] \leqslant \frac{R^{p-1}}{\sigma} \cdot \Big(\text{R.H.S. of (5)}\Big), \tag{7}$$

where $R^{p-1}/\sigma < 1$.

Comparing Theorems 5.5 and 5.11, we observe that AdagradPower accelerates convergence of Adagrad by a constant factor $\frac{R^{p-1}}{\sigma}$ in high-noise regime. For Example 5.10, the acceleration is significant, due to $\frac{R^{p-1}}{\sigma} \leqslant \frac{1.01}{p}$ for any positive odd p. This result provides theoretical support for the empirical superiority of adaptive optimizers using GradPower in LLM pretraining in Section 4 Notably, the theoretical insights are highly aligned with those in Proposition 5.3. In the high-noise regime, using $p > 1$ reduces both numerator $\boldsymbol{g}_t$ and denominator $\sqrt{\boldsymbol{v}_t + \epsilon}$. However, reduction in the denominator outweighs that in the numerator, resulting in a faster convergence speed. The formal proof refines this argument and is presented in Appendix D.

6. Conclusion

We propose GradPower, a simple yet effective method for improving the efficiency of gradient-based optimizers. Experimentally, AdamWPower (AdamW using GradPower) consistently achieves lower terminal loss and improved scaling laws than AdamW across various LLM pre-training tasks.

For future work, it would be interesting to investigate why AdamWPower exhibits particular potential for MoE models and `wsd` LR scheduler. Experimentally, exploring the applicability of GradPower beyond LLMs, as well as its integration with other optimizers, could further extend its impact. In addition, developing a dynamic schedule for the GradPower exponent p, adapted to the evolving SNR throughout training, presents both a challenging and a potentially valuable direction.

Acknowledgments

Lei Wu is supported by the National Natural Science Foundation of China (NSFC12522120, NSFC92470122, and NSFC12288101). Mingze Wang is supported by Young Scientists (PhD) Fund of the National Natural Science Foundation of China (No. 124B2028).

Impact Statement

This paper presents work whose goal is to advance the field of deep learning, with a specific focus on improving large language model (LLM) pre-training. While our work has the potential to impact society in various ways, we do not

[4]Empirical studies suggest that gradient scales in LLM training are often very small (Huang et al., 2025).

identify any immediate negative societal impacts requiring special mitigation beyond standard responsible use.

References

Baiesi, M. Power gradient descent. *arXiv preprint arXiv:1906.04787*, 2019. 14

Bernstein, J., Wang, Y.-X., Azizzadenesheli, K., and Anandkumar, A. signsgd: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*, pp. 560–569. PMLR, 2018. 17

Bisk, Y., Zellers, R., Gao, J., Choi, Y., et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020. 4

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020. 1

Chen, X., Feng, K., Li, C., Lai, X., Yue, X., Yuan, Y., and Wang, G. Fira: Can we achieve full-rank training of llms under low-rank constraint? *arXiv preprint arXiv:2410.01623*, 2024a. 2, 16

Chen, X., Liang, C., Huang, D., Real, E., Wang, K., Liu, Y., Pham, H., Dong, X., Luong, T., Hsieh, C.-J., Lu, Y., and Le, Q. V. Symbolic discovery of optimization algorithms. *Advances in Neural Information Processing Systems*, 36, 2024b. 1, 2, 17

Cohen, J., Kaur, S., Li, Y., Kolter, J. Z., and Talwalkar, A. Gradient descent on neural networks typically occurs at the edge of stability. In *International Conference on Learning Representations*, 2020. 2

Cohen, J. M., Ghorbani, B., Krishnan, S., Agarwal, N., Medapati, S., Badura, M., Suo, D., Cardoze, D., Nado, Z., Dahl, G. E., et al. Adaptive gradient methods at the edge of stability. *arXiv preprint arXiv:2207.14484*, 2022. 2

Cohen, J. M., Damian, A., Talwalkar, A., Kolter, Z., and Lee, J. D. Understanding optimization in deep learning with central flows. In *International Conference on Learning Representations*, 2025. 2

DeepSeek-AI, Liu, A., Feng, B., Xue, B., Wang, B., Wu, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024. 1, 5

Défossez, A., Bottou, L., Bach, F., and Usunier, N. A simple convergence proof of adam and adagrad. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. URL https://openreview.net/forum?id=ZPQhzTSWA7. 8, 19, 21

Dettmers, T., Lewis, M., Shleifer, S., and Zettlemoyer, L. 8-bit optimizers via block-wise quantization. *International Conference on Learning Representations*, 2022. 2

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011. 8

Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., Le Noac'h, A., et al. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602. 4

Gokaslan, A. and Cohen, V. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019. 4, 15

Hägele, A., Bakouch, E., Kosson, A., Von Werra, L., Jaggi, M., et al. Scaling laws and compute-optimal training beyond fixed training durations. *Advances in Neural Information Processing Systems*, 37:76232–76264, 2024. 4, 5, 15

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016. 7, 15

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de Las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., et al. Training compute-optimal large language models. *Advances in neural information processing systems*, 35:30016–30030, 2022. 4, 16

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models. *International Conference on Learning Representations*, 1(2):3, 2022. 2

Hu, S., Tu, Y., Han, X., He, C., Cui, G., Long, X., Zheng, Z., Fang, Y., Huang, Y., Zhao, W., et al. Minicpm: Unveiling the potential of small language models with scalable training strategies. In *Conference on Language Modeling*, 2024. 4, 15

Huang, T., Zhu, Z., Jin, G., Liu, L., Wang, Z., and Liu, S. Spam: Spike-aware adam with momentum reset for stable llm training. *International Conference on Learning Representations*, 2025. 7, 9

Jastrzebski, S., Szymczak, M., Fort, S., Arpit, D., Tabor, J., Cho, K., and Geras, K. The break-even point on optimization trajectories of deep neural networks. In *International Conference on Learning Representations*, 2020. 2

Jordan, K., Jin, Y., Boza, V., You, J., Cesista, F., Newhouse, L., and Bernstein, J. Muon optimizer. https://kellerjordan.github.io/posts/muon, 2024. 1, 2, 6

Karpathy, A. NanoGPT. https://github.com/karpathy/nanoGPT, 2022. 4, 15, 16

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*, 2017. 6

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1, 2

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images, 2009. URL https://www.cs.toronto.edu/~kriz/cifar.html. 7, 15

Kunstner, F., Milligan, A., Yadav, R., Schmidt, M., and Bietti, A. Heavy-tailed class imbalance and why Adam outperforms gradient descent on language models. *Advances in Neural Information Processing Systems*, 37: 30106–30148, 2024. 1

Li, B., Chen, J., and Zhu, J. Memory efficient optimizers with 4-bit states. *Advances in Neural Information Processing Systems*, 36:15136–15171, 2023. 2

Liang, K., Chen, L., Liu, B., and Liu, Q. Cautious optimizers: Improving training with one line of code. *arXiv preprint arXiv:2411.16085*, 2024. 1, 2

Liu, H., Li, Z., Hall, D., Liang, P., and Ma, T. Sophia: A scalable stochastic second-order optimizer for language model pre-training. *International Conference on Learning Representations*, 2024. 1, 2, 15, 16

Liu, J., Su, J., Yao, X., Jiang, Z., Lai, G., Du, Y., Qin, Y., Xu, W., Lu, E., Yan, J., et al. Muon is scalable for llm training. *arXiv preprint arXiv:2502.16982*, 2025a. 1, 2, 6, 16

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019. 4, 15

Liu, Y., Liu, Z., and Gore, J. Focus: First order concentrated updating scheme. *arXiv preprint arXiv:2501.12243*, 2025b. 1, 2

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 1

McCandlish, S., Kaplan, J., Amodei, D., and Team, O. D. An empirical model of large-batch training. *arXiv preprint arXiv:1812.06162*, 2018. 3, 6

Mihaylov, T., Clark, P., Khot, T., and Sabharwal, A. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018. 4

Mori, T., Ziyin, L., Liu, K., and Ueda, M. Power-law escape rate of sgd. In *International Conference on Machine Learning*, pp. 15959–15975. PMLR, 2022. 3

OpenAI, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 1

Orvieto, A. and Gower, R. In search of adam's secret sauce. *Advances in Neural Information Processing Systems*, 38: 63404–63442, 2026. 17

Pagliardini, M., Ablin, P., and Grangier, D. The ademamix optimizer: Better, faster, older. *International Conference on Learning Representations*, 2025. 1, 2

Qin, C., Cai, Z., and Guo, Y. A stochastic recursive gradient algorithm integrating momentum and the powerball function with adaptive step sizes. *International Journal of Machine Learning and Cybernetics*, pp. 1–21, 2025. 14

Qwen, Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. 5

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020. 4, 15

Sakaguchi, K., Bras, R. L., Bhagavatula, C., and Choi, Y. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021. 4

Schwarz, J., Jayakumar, S., Pascanu, R., Latham, P. E., and Teh, Y. Powerpropagation: A sparsity inducing weight reparameterisation. *Advances in neural information processing systems*, 34:28889–28903, 2021. 14

Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J., and Catanzaro, B. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019. 6

Song, M., Ahn, K., and Yun, C. Does sgd really happen in tiny subspaces? *International Conference on Learning Representations*, 2025. 2, 3

Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. 15

Sun, Z., Sun, Y., Yang, L., Lu, S., Mei, J., Zhao, W., and Hu, Y. Unleashing the power of gradient signal-to-noise ratio for zero-shot nas. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 5763–5773, 2023. 3

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 2, 4, 6, 15, 16

Vyas, N., Morwani, D., Zhao, R., Shapira, I., Brandfonbrener, D., Janson, L., and Kakade, S. Soap: Improving and stabilizing shampoo using adam. *arXiv preprint arXiv:2409.11321*, 2024. 2

Wang, J., Wang, M., Zhou, Z., Yan, J., E, W., and Wu, L. The sharpness disparity principle in transformers for accelerating language model pre-training. In *International Conference on Machine Learning*, pp. 64859–64879. PMLR, 2025. 1, 2, 6, 16

Wang, J., Li, B., Zhou, Z., Wang, M., Sun, Y., Zhang, J., Cai, X., and Wu, L. Fast catch-up, late switching: Optimal batch size scheduling via functional scaling laws. *arXiv preprint arXiv:2602.14208*, 2026. 6

Wang, M. and Wu, L. A theoretical analysis of noise geometry in stochastic gradient descent. *arXiv preprint arXiv:2310.00692*, 2023. 3

Wang, M., Wang, J., He, H., Wang, Z., Huang, G., Xiong, F., Li, Z., E, W., and Wu, L. Improving generalization and convergence by enhancing implicit regularization. *Advances in Neural Information Processing Systems*, 2024. 1, 2

Wen, K., Li, Z., Wang, J., Hall, D., Liang, P., and Ma, T. Understanding warmup-stable-decay learning rates: A river valley loss landscape perspective. *International Conference on Learning Representations*, 2025. 2, 3, 7

Wortsman, M., Liu, P. J., Xiao, L., Everett, K., Alemi, A., Adlam, B., Co-Reyes, J. D., Gur, I., Kumar, A., Novak, R., et al. Small-scale proxies for large-scale transformer training instabilities. *International Conference on Learning Representations*, 2024. 4

Wu, J., Hu, W., Xiong, H., Huan, J., Braverman, V., and Zhu, Z. On the noisy gradient descent that generalizes as SGD. In *International Conference on Machine Learning*, pp. 10367–10376. PMLR, 2020. 3

Wu, L., Ma, C., and E, W. How SGD selects the global minima in over-parameterized learning: A dynamical stability perspective. *Advances in Neural Information Processing Systems*, 31:8279–8288, 2018. 2

Wu, L., Wang, M., and Su, W. J. The alignment property of SGD noise and how it helps select flat minima: A stability analysis. *Advances in Neural Information Processing Systems*, 35:4680–4693, 2022. 3

Xie, X., Zhou, P., Li, H., Lin, Z., and Yan, S. Adan: Adaptive nesterov momentum algorithm for faster optimizing deep models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. 1, 2

Yadav, V., Bethard, S., and Surdeanu, M. Quick and (not so) dirty: Unsupervised selection of justification sentences for multi-hop question answering. *arXiv preprint arXiv:1911.07176*, 2019. 4

Yang, A., Yang, B., Hui, B., Zheng, B., Yu, B., Zhou, C., Li, C., Li, C., Liu, D., Huang, F., et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024. 2, 4, 5, 15

Yang, Z. The powerball method with biased stochastic gradient estimation for large-scale learning systems. *IEEE Transactions on Computational Social Systems*, 11(6): 7435–7447, 2024. 14

Yuan, H., Liu, Y., Wu, S., Zhou, X., and Gu, Q. Mars: Unleashing the power of variance reduction for training large models. *arXiv preprint arXiv:2411.10438*, 2024. 2

Yuan, Y., Li, M., Liu, J., and Tomlin, C. On the powerball method: Variants of descent methods for accelerated optimization. *IEEE Control Systems Letters*, 3(3):601–606, 2019. 14

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019. 4

Zhai, X., Kolesnikov, A., Houlsby, N., and Beyer, L. Scaling vision transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12104–12113, 2022. 4, 15

Zhang, Y., Chen, C., Ding, T., Li, Z., Sun, R., and Luo, Z. Why transformers need Adam: A hessian perspective. *Advances in Neural Information Processing Systems*, 37: 131786–131823, 2024. 1

Zhang, Y., Chen, C., Li, Z., Ding, T., Wu, C., Ye, Y., Luo, Z.-Q., and Sun, R. Adam-mini: Use fewer learning rates to gain more. *International Conference on Learning Representations*, 2025. 1, 2

Zhao, J., Zhang, Z., Chen, B., Wang, Z., Anandkumar, A., and Tian, Y. Galore: Memory-efficient llm training by gradient low-rank projection. *International Conference on Machine Learning*, 2024. 2, 4, 15, 16

Zhao, R., Morwani, D., Brandfonbrener, D., Vyas, N., and Kakade, S. Deconstructing what makes a good optimizer for language models. *International Conference on Learning Representations*, 2025. 4, 15

Zhou, B., Liu, J., Sun, W., Chen, R., Tomlin, C. J., and Yuan, Y. pbsgd: Powered stochastic gradient descent methods for accelerated non-convex optimization. In *IJCAI*, pp. 3258–3266, 2020. 14

Zhu, H., Zhang, Z., Cong, W., Liu, X., Park, S., Chandra, V., Long, B., Pan, D. Z., Wang, Z., and Lee, J. Apollo: Sgd-like memory, adamw-level performance. *Conference on Machine Learning and Systems*, 2025. 2, 16

Zhu, S., Hu, R., Wang, M., Sun, M., Wang, X., Yuan, K., and Wen, Z. Accelerating llm pre-training through flat-direction dynamics enhancement. *arXiv preprint arXiv:2602.22681*, 2026. 2

Zhu, Z., Wu, J., Yu, B., Wu, L., and Ma, J. The anisotropic noise in stochastic gradient descent: Its behavior of escaping from sharp minima and regularization effects. In *International Conference on Machine Learning*, pp. 7654–7663. PMLR, 2019. 3

Appendix

A Relationship with Other Similar Works. **14**

B Experimental Details **15**

B.1 Experimental details for Section 4.2 and 4.3 16

B.2 Experimental details for Section 4.4 16

B.3 Experimental details for Section 4.6 16

B.4 Additional experiments with multiple random seeds 16

B.5 Robustness to (β_1, β_2) tuning 17

C Proofs in Section 3 and 5.1 **17**

C.1 Support for the motivation in Section 3 17

C.2 Proof of Propositions 5.2 and 5.3 18

D Proofs in Section 5.2 **19**

D.1 Key Lemmas 20

D.2 Proof of Theorem 5.8 22

D.3 Proof of Theorem 5.11 23

A. Relationship with Other Similar Works.

Powerball method. After completing this work, we found that the Powerball method (Yuan et al., 2019) shares the similar methodology as our approach. However, prior studies on Powerball method have been restricted to traditional optimizers—such as GD (Yuan et al., 2019), SGD (Zhou et al., 2020; Yang, 2024), and SARAH (Qin et al., 2025)——and evaluated primarily on relatively small-scale benchmarks including CIFAR-10, CIFAR-100 and MNIST. Although Baiesi (2019) combined Powerball with Adam, the experiments were limited to small and illustrative problems. In contrast, our work focuses on modern adaptive optimizers such as Adam and Muon in the context of language model pre-training, a modern and practically important setting. Moreover, previous Powerball studies examined only the narrow regime with $p < 1$, our work studies both $p < 1$ and $p > 1$ regimes, and further develop a comprehensive theoretical study of the relationship between optimal p and batch size.

Powerpropagation method. Powerpropagation (Schwarz et al., 2021) also employs a sign-power transformation, but applies it to the model parameters rather than to the gradients. Specifically, Powerpropagation reparameterizes weights as $\theta = \phi|\phi|^{\alpha-1}$, which induces an effective gradient scaling through the chain rule, $g_\phi = \alpha g_\theta |\phi|^{\alpha-1}$. In contrast, GradPower transformation is applied directly to gradient, $\varphi_p(g) = \mathrm{sign}(g)|g|^p$, before the base optimizer step and without modifying the model parameterization. Thus, although the two methods share a similar sign-power functional form, they operate in fundamentally different spaces and induce different mechanisms: Powerpropagation scales updates by parameter magnitude, whereas GradPower scales updates by gradient magnitude.

B. Experimental Details

Models. We utilize two popular classes of LLM models for our pre-training experiments:

- **LLaMA.** LLaMA (Touvron et al., 2023) is a popular Dense decoder-only Transformer architecture, incorporating Rotary Positional Encoding (RoPE) (Su et al., 2024), Swish-Gated Linear Unit (SwiGLU), and Root mean square layer normalization (RMSNorm). We pre-train LLaMA models of sizes ranging from 66M to 2B parameters. Additional model configurations are detailed in Table 3.
- **Qwen2MoE.** Comparing with Llama, Qwen2MoE is decoder-only mixture-of-experts architecture. Following Qwen2MoE and (Yang et al., 2024), we activate 4 experts per token for all models. For detailed model configurations, refer to Table 4.

Datasets. Models are pre-trained on the following datasets:

- **Colossal Clean Crawled Corpus (C4)** (Raffel et al., 2020). It is a large-scale public language dataset, widely used for LLM pre-training such as T5 (Raffel et al., 2020), and prior pre-training studies (Zhao et al., 2024; 2025). We use the T5 tokenizer, with the vocabulary size 32100.
- **OpenWebText** (Gokaslan & Cohen, 2019). It is an opensource recreation of the WebText corpus, is extensively utilized for LLM pre-training such as RoBERTa (Liu et al., 2019) and nanoGPT (Karpathy, 2022). Following Karpathy (2022); Liu et al. (2024), we use the GPT-2 tokenizer, with the vocabulary size 50304.

LR schedulers. We evaluate two popular LR scheduling strategies:

- `cos` (cosine scheduler) (Karpathy, 2022; Touvron et al., 2023): a linear warm-up to peak `lr_max`, followed by cosine decay to a terminal LR `lr_min`.
- `wsd` (warmup-stable-decay scheduler) (Zhai et al., 2022; Hu et al., 2024; Hägele et al., 2024): a linear warm-up LR to peak `lr_max`, followed by a stable phase where LR remains at `lr_max` (up to 80% of the total training steps), and then a linear decay to `lr_min`.

All experiments are conducted on 8 A100 80G GPUs.

Table 3. Dense model configurations and optimally-tuned peak learning rates for AdamW.

Acronym	Size	d_{model}	d_{FF}	n_head	depth	`lr_max`
LLaMA (66M)	66M	512	2048	8	8	1e-3 (on C4)
LLaMA (0.2B)	200M	1024	4096	16	8	1e-3 (on C4)
LLaMA (0.25B)	237M	1024	4096	16	8	8e-4 (on OpenWebText)
LLaMA (0.4B)	400M	1280	5120	16	12	6e-4 (on C4)
LLaMA (1B)	1004M	1600	6400	25	22	3e-4 (on C4)
LLaMA (2B)	1994M	2048	8096	32	28	2e-4 (on C4)

Table 4. MoE model configurations and optimally-tuned peak learning rates for AdamW on C4.

Acronym	Size	Activated Size	d_{model}	d_{FF}	n_head	depth	n_experts	`lr_max`
Qwen2MoE (0.5B)	502M	247M	768	3072	12	12	16	6e-4
Qwen2MoE (1B)	1040M	297M	768	3072	12	15	32	3e-4
Qwen2MoE (2B)	1945M	536M	1024	4096	16	16	32	2e-4

For the vision experiment, we used the standard 34 layer ResNet model (He et al., 2016) on the CIFAR-10 dataset (Krizhevsky & Hinton, 2009). We use AdamW optimizer and the commonly used `cos` learning rate scheduler.

B.1. Experimental details for Section 4.2 and 4.3

AdamW baselines. We use the standard Adam optimizer (with decoupled weight decay) as the baseline in most experiments (except Section 4.4). The baseline is configured with hyperparameters $\beta_1 = 0.9, \beta_2 = 0.95$, weight decay $\lambda = 0.1$, and gradient clipping threshold of 1.0, following protocols used in LLaMA pre-training (Touvron et al., 2023). Following Hoffmann et al. (2022), the final learning rate `lr_min` is set to $1/10$ of the peak learning rate `lr_max`. Additionally,

- **C4 pre-training.** We follow the setup of Zhao et al. (2024); Chen et al. (2024a); Zhu et al. (2025), using a sequence length of 256 and batch size of 512. Following the Chinchilla scaling law (Hoffmann et al., 2022), the total number of training tokens is set to be approximately 20 times the number of model parameters. The training includes 1,000 warm-up steps. The grid search for `lr_max` is performed over {`1e-4`, `2e-4`, `3e-4`, `6e-4`, `1e-3`, `1.5e-3`}. Optimal learning rates for each model are detailed in Tables 3 and 4.
- **OpenWebText pre-training.** The (max) sequence length is set to 1024, and the batch size is set to 480, following nanoGPT (Karpathy, 2022) and Liu et al. (2024). The total training duration is 50,000 or 100,000 steps, including 1,000 warm-up steps. The grid search for `lr_max` is performed over {`2e-4`, `4e-4`, `6e-4`, `8e-4`, `1e-3`}. Optimal learning rates for each model are detailed in Table 3.

AdamWPower experiments. We adopt $p = 1.2$ as the default in all experiments in Section 4.2 and 4.3. All other optimizer hyperparameters are kept identical to those used for the AdamW baselines. Importantly, the power $p = 1.2$ proves to be **highly robust**.

B.2. Experimental details for Section 4.4

AdamW with Blockwise LR. Following Wang et al. (2025), we adopt the same peak `lr_max` tuned for AdamW as the `lr_max` of AdamW with Blockwise LR. For the blockwise lr ratios, we adopt the recommended $r(\text{Embed}) = 10, r(\text{QK}) = 8, r(\text{FFN}) = 6, r(\text{VO}) = 4$ in Wang et al. (2025).

AdamWPower with Blockwise LR. We still adopt $p = 1.2$ in the AdamWPower with Blockwise LR. All other optimizer hyperparameters are kept identical to those used for the AdamW with Blockwise LR.

Muon baseline. We use the same techniques for Muon as Liu et al. (2025a): (1) adding weight decay (2) adjusting the per-parameter update scale. These techniques allow our Muon experiment to use the identical learning rate as the AdamW baseline without the extra effort of hyper-parameter tuning.

MuonPower. We still adopt $p = 1.2$ in the MuonPower. All other optimizer hyperparameters are kept identical to those used for the Muon baseline.

B.3. Experimental details for Section 4.6

We conduct experiments using LLaMA (0.2B) on C4 dataset with `wsd` scheduler. Unlike the previous experimental settings, here we vary the batch size from the standard 512 up to 8192.

For batch size 512, the tuned `max_lr` is `1e-3` (Table 3). For larger batch sizes (2048, 4096, 8192), we tune the `max_lr` over {`6e-4, 1e-3, 2e-3, 4e-3, 8e-3`} for AdamW. We find that `1e-3` consistently yields the best results across all batch sizes.

For each batch size, we evaluate AdamWPower with multiple values of p, and record their validation loss when the optimal validation loss reaches approximately 3.5.

We also conduct vision experiments using ResNet-34 on CIFAR-10 datset with `cos` scheduler. We tune the `max_lr` over {`6.25e-5, 1.25e-4, 2.5e-4, 5e-4, 1e-3`}. For batch size 32, 64, and 128, the tuned `max_lr` is `1.25e-4`, `2.5e-4`, `5e-4`, respectively.

B.4. Additional experiments with multiple random seeds

In this subsection, we reproduce a subset of experiments in Figure 3 with multiple random seeds to assess statistical robustness. Specifically, we rerun the experiments six times with different random seeds and report both mean and standard deviation as shown in Figure 8. The shaded regions in the plots denote the standard deviation, showing the statistical

significance of each method. These results confirm that the observed performance differences are consistent and cannot be explained by random seed variability.

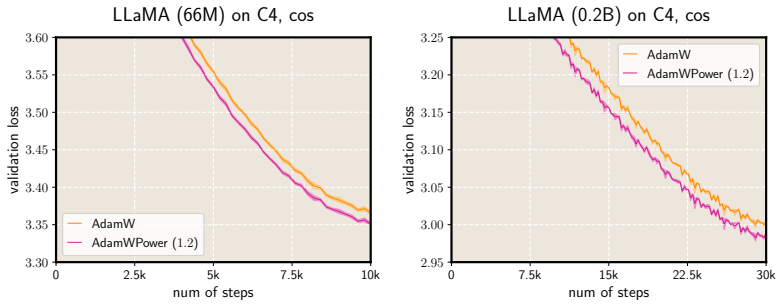

Figure 8. AdamWPower ($p = 1.2$) consistently outperforms AdamW in LLaMA pre-training tasks. The shaded regions in the plots denote the standard deviation.

B.5. Robustness to (β_1, β_2) tuning

We further examine whether the advantage of AdamWPower persists after tuning the momentum parameters (β_1, β_2) in AdamW. While the main experiments use the standard setting $(\beta_1, \beta_2) = (0.9, 0.95)$, we additionally sweep (β_1, β_2) for both AdamW and AdamWPower using LLaMA (0.25B) on OpenWebText. Specifically, we consider (β_1, β_2) configurations following the spirit of Orvieto & Gower (2026):

$$\begin{aligned}(\beta_1, \beta_2) \in \{&(0.9, 0.8),\ (0.9, 0.9),\ (0.9, 0.95),\ (0.9, 0.975),\\ &(0.95, 0.9),\ (0.95, 0.95),\ (0.95, 0.975),\ (0.95, 0.9875),\\ &(0.975, 0.95),\ (0.975, 0.975),\ (0.975, 0.9875),\ (0.975, 0.99375)\}.\end{aligned}$$

The best AdamW baseline is obtained with $(\beta_1, \beta_2) = (0.95, 0.9875)$, achieving a final validation loss of 2.832831, whereas AdamWPower with $p = 1.2$ achieves its best result with $(\beta_1, \beta_2) = (0.9, 0.8)$ and a lower final validation loss of 2.828666. These results show that AdamWPower continues to outperform the tuned AdamW baseline even after tuning β_1 and β_2, suggesting that the improvement is not merely due to a particular choice of momentum parameters.

C. Proofs in Section 3 and 5.1

C.1. Support for the motivation in Section 3

In this section, we provide a detailed justification for the claim in Section 3 that the **linear transformation** ($\varphi(z) = cz$ with $c \in \mathbb{R}$) **fails** to alter the updates of popular optimizers used in LLM pretraining. Without loss of generality, we analyze the one-dimensional case.

- **Adaptive optimizers**, including Adagrad, RMSprop, and Adam. These optimizers adjust the learning rate based on a moving average of gradients. In practice, the term ϵ (used to ensure numerical stability) is typically set to an extremely small value (e.g., `1e-8, 1e-12`), Consider the update rule of Adam in the limit $\epsilon \to 0$:

  $$\theta_{t+1} = (1 - \lambda\eta_t)\theta_t - \eta_t \frac{\mathrm{EMA}_{\beta_1}(\{g_s\}_1^t)}{\sqrt{\mathrm{EMA}_{\beta_2}(\{g_s^2\}_1^t)}}.$$

  Applying a linear transformation $\varphi(z) = cz$ with $c > 0$ does not change the ratio:

  $$\frac{\mathrm{EMA}_{\beta_1}(\{g_s\}_1^t)}{\sqrt{\mathrm{EMA}_{\beta_2}(\{g_s^2\}_1^t)}} = \frac{\mathrm{EMA}_{\beta_1}(\{\varphi(g_s)\}_1^t)}{\sqrt{\mathrm{EMA}_{\beta_2}(\{\varphi(g_s)^2\}_1^t)}}.$$

  Hence, the dynamics remains unchanged. This argument applies similarly to Adagrad and RMSprop.

- **Sign-based methods**, including Sign momentum (Bernstein et al., 2018) and Lion (Chen et al., 2024b). These methods operate on the sign of the moving average gradients. For instance, Signed Momentum (with decoupled weight decay)

follows:

$$\theta_{t+1} = (1 - \lambda\eta_t)\theta_t - \eta_t \operatorname{sign}(\mathrm{EMA}_\beta(\{g_s\}_1^t)).$$

Again, applying a linear transformation $\varphi(z) = cz$ with $c > 0$ does not change the sign of the averaged gradient, since:

$$\operatorname{sign}(\mathrm{EMA}_\beta(\{g_s\}_1^t)) = \operatorname{sign}(\mathrm{EMA}_\beta(\{\varphi(g_s)\}_1^t)).$$

Hence, the dynamics remains unchanged. This argument applies similarly to Lion.

In contrast, our proposed (nonlinear) GradPower transformation ($\varphi_p(z) = z^p := |z|^p \operatorname{sign}(z)$ with $p > 0$) *does* alter the updates of both adaptive and sign-based optimizers, when the gradients g_s are not all of the same sign.

C.2. Proof of Propositions 5.2 and 5.3

$$\mathbb{E}[\varphi_p(g)] = \frac{(\mu+\sigma)^{p+1} - |\mu-\sigma|^{p+1}}{2\sigma(p+1)},$$
$$\mathbb{E}[\varphi_p^2(g)] = \frac{(\mu+\sigma)^{2p+1} - |\mu-\sigma|^{2p+1}\operatorname{sign}(\mu-\sigma)}{2\sigma(2p+1)}.$$

The low-noise regime. ($0 \ll \sigma \ll \mu \ll 1$)

It is straightforward that

$$\begin{aligned}\mathbb{E}[\varphi_p(g)] =& \frac{\mu^{p+1}}{2\sigma(p+1)}\left(\left(1+\frac{\sigma}{\mu}\right)^{p+1} - \left(1-\frac{\sigma}{\mu}\right)^{p+1}\right)\\ =& \frac{\mu^{p+1}}{2\sigma(p+1)}\left(\frac{2(p+1)\sigma}{\mu} + o\left(\frac{\sigma}{\mu}\right)\right) = \mu^p\left(1+o(1)\right);\end{aligned}$$

$$\begin{aligned}\mathbb{E}[\varphi_p^2(g)] =& \frac{\mu^{2p+1}}{2\sigma(2p+1)}\left(\left(1+\frac{\sigma}{\mu}\right)^{2p+1} - \left(1-\frac{\sigma}{\mu}\right)^{2p+1}\right)\\ =& \frac{\mu^{2p+1}}{2\sigma(2p+1)}\left(\frac{2(2p+1)\sigma}{\mu} + o\left(\frac{\sigma}{\mu}\right)\right) = \mu^{2p}\left(1+o(1)\right).\end{aligned}$$

Therefore, we have

$$u = \frac{\mathbb{E}[\varphi_p(g)]}{\sqrt{\mathbb{E}[\varphi_p^2(g)]} + \epsilon} = \frac{\mu^p\left(1+o(1)\right)}{\mu^p\left(1+o(1)\right) + \epsilon} = \frac{1+o(1)}{1+\frac{\epsilon}{\mu^p}}.$$

The high-noise regime. ($0 \ll \mu \ll \sigma \ll 1$)

It is straightforward that

$$\begin{aligned}\mathbb{E}[\varphi_p(g)] =& \frac{\sigma^{p+1}}{2\sigma(p+1)}\left(\left(1+\frac{\mu}{\sigma}\right)^{p+1} - \left(1-\frac{\mu}{\sigma}\right)^{p+1}\right)\\ =& \frac{\sigma^{p}}{2(p+1)}\left(\frac{2(p+1)\mu}{\sigma} + o\left(\frac{\mu}{\sigma}\right)\right) = \sigma^{p-1}\mu(1+o(1));\end{aligned}$$

$$\begin{aligned}\mathbb{E}[\varphi_p^2(g)] =& \frac{\sigma^{2p+1}}{2\sigma(2p+1)}\left(\left(1+\frac{\mu}{\sigma}\right)^{2p+1} + \left(1-\frac{\mu}{\sigma}\right)^{2p+1}\right)\\ =& \frac{\sigma^{2p+1}}{2\sigma(2p+1)}\left(2 + o\left(\frac{\mu}{\sigma}\right)\right) = \frac{\sigma^{2p}}{2p+1}(1+o(1)).\end{aligned}$$

Therefore, we have

$$u = \frac{\mathbb{E}[\varphi_p(g)]}{\sqrt{\mathbb{E}[\varphi_p^2(g)]} + \epsilon} = \frac{\sigma^{p-1}\mu(1+o(1))}{\frac{\sigma^p}{\sqrt{2p+1}}(1+o(1)) + \epsilon} = \frac{\mu}{\sigma}\frac{1+o(1)}{\frac{1+o(1)}{\sqrt{2p+1}} + \frac{\epsilon}{\sigma^p}} = \frac{\mu}{\sigma}\frac{1+o(1)}{\frac{1}{\sqrt{2p+1}} + \frac{\epsilon}{\sigma^p}}.$$

To study the monotonicity of $\tilde{u} = \frac{\mu}{\sigma}\frac{1}{\frac{1}{\sqrt{2p+1}} + \frac{\epsilon}{\sigma^p}}$, we only need to study the monotonicity of

$$\psi(p) = \frac{1}{\sqrt{2p+1}} + \frac{\epsilon}{\sigma^p}.$$

It is clear that

$$\psi'(p) = \frac{\epsilon \log(1/\sigma)}{\sigma^p} - \frac{1}{(2p+1)^{3/2}}.$$

Due to $\epsilon \log(1/\sigma) < 1$, there exists a $p^\star$, such that $\psi'(p) < 0$ for all $0 < p < p^\star$; $\psi'(p) > 0$ for all $p > p^\star$. Here, $p^\star$ is the solution of the equation:

$$\frac{\sigma^p}{(2p+1)^{3/2}} = \epsilon \log(1/\sigma)$$

Noticing the relationship between ψ and $\tilde{u}$, we have: $\tilde{u}$ increases when $0 < p < p^\star$; $\tilde{u}$ decreases when $p > p^\star$.

Now we estimate $p^\star$. Due to $1 + x \leqslant e^x$, we have $(2p+1)^{3/2} \leqslant (e^{2p})^{3/2} = (e^3)^p$. Then we obtain the two-sides estimate $1 \leqslant (2p+1)^{3/2} \leqslant (e^3)^p$, implying

$$\left(\frac{\sigma}{e^3}\right)^p \leqslant \frac{\sigma^p}{(2p+1)^{3/2}} \leqslant \sigma^p.$$

Therefore, we have the estimate:

$$\frac{\log(\epsilon \log(1/\sigma))}{\log(\sigma/e^3)} \leqslant p^\star \leqslant \frac{\log(\epsilon \log(1/\sigma))}{\log \sigma}$$

Noticing $\sigma \ll 1$, we obtain:

$$p^\star = \Theta\left(\frac{\log(\epsilon \log(1/\sigma))}{\log \sigma}\right).$$

D. Proofs in Section 5.2

Recall that the udpate rule of AdagradPower (with power p) follows:

$$\begin{aligned}
\boldsymbol{\theta}_{t+1} =& \boldsymbol{\theta}_t - \eta \boldsymbol{u}_t, \\
\boldsymbol{u}_t =& \frac{\varphi_p(\boldsymbol{g}_t)}{\sqrt{\boldsymbol{v}_t + \epsilon}}, \\
\boldsymbol{v}_t =& \sum_{s=1}^{t} \varphi_p^2(\boldsymbol{g}_t).
\end{aligned}$$

In general, our proof is inspired by the main techniques to prove Adagrad used in Défossez et al. (2022). The key difference is to establish a similar estimate of the loss descent for Adamgradpower. This generalize is not trivial, need to use the structure of the high-noise fact.

In the proof, we need an auxiliary sequence, defined as:

$$\tilde{\boldsymbol{v}}_t = \boldsymbol{v}_{t-1} + \mathbb{E}_t[\varphi_p^2(\boldsymbol{g}_t)].$$

D.1. Key Lemmas

We need two important lemmas in the proof of each Theorem. The first develops the lower bound of the descent value for the update.

Lemma D.1 (Descent estimate for the update, high-noise regime)**.** *Under Assumption 5.4, for all $t \in \mathbb{N}$, and $i \in [d]$ and any $\sigma > 0$, we have:*

$$\mathbb{E}_t\left[\nabla_i \mathcal{L}(\boldsymbol{\theta}) u_{t,i}\right] = \mathbb{E}_t\left[\frac{\nabla_i \mathcal{L}(\boldsymbol{\theta})\varphi_p(g_{t,i})}{\sqrt{v_{t,i}+\epsilon}}\right] \geqslant \frac{\mathbb{E}_t\left[\nabla_i \mathcal{L}(\boldsymbol{\theta})\varphi_p(g_{t,i})\right]}{\sqrt{\tilde{v}_{t,i}+\epsilon}} - \frac{\sigma}{2}\frac{|\nabla_i \mathcal{L}(\boldsymbol{\theta})|^2}{\sqrt{\tilde{v}_{t,i}+\epsilon}} - \frac{2R^p}{\sigma}\mathbb{E}\left[\frac{\varphi_p^2(g_{t,i})}{v_{t,i}+\epsilon}\right].$$

Proof of Lemma D.1.
Let $t \in \mathbb{N}$ and $i \in [d]$. For simplicity, we use the following notations in the proof:

$$G = \nabla_i \mathcal{L}(\boldsymbol{\theta}),\ g = g_{t,i},\ v = v_{t,i},\ \tilde{v} = \tilde{v}_{t,i}.$$

First, we decouple the descent quantity as:

$$\mathbb{E}_t\left[\frac{G\varphi_p(g)}{\sqrt{v+\epsilon}}\right] = \mathbb{E}_t\left[\frac{G\varphi_p(g)}{\sqrt{\tilde{v}+\epsilon}}\right] + \mathbb{E}_t\left[\underbrace{G\varphi_p(g)\left(\frac{1}{\sqrt{v+\epsilon}} - \frac{1}{\sqrt{\tilde{v}+\epsilon}}\right)}_{I}\right] \tag{8}$$

Then we bound the term I in the RHS of Equation (8):

$$\begin{aligned}
|I| =& |G\varphi_p(g)|\frac{|\tilde{v}-v|}{\sqrt{v+\epsilon}\sqrt{\tilde{v}+\epsilon}(\sqrt{v+\epsilon}+\sqrt{\tilde{v}+\epsilon})} \\
=& |G\varphi_p(g)|\frac{|\mathbb{E}_t[\varphi_p^2(g)] - \varphi_p^2(g)|}{\sqrt{v+\epsilon}\sqrt{\tilde{v}+\epsilon}(\sqrt{v+\epsilon}+\sqrt{\tilde{v}+\epsilon})} \\
\leqslant& |G\varphi_p(g)|\frac{\mathbb{E}_t[\varphi_p^2(g)] + \varphi_p^2(g)}{\sqrt{v+\epsilon}\sqrt{\tilde{v}+\epsilon}(\sqrt{v+\epsilon}+\sqrt{\tilde{v}+\epsilon})} \\
\leqslant& \underbrace{|G\varphi_p(g)|\frac{\mathbb{E}_t[\varphi_p^2(g)]}{\sqrt{v+\epsilon}(\tilde{v}+\epsilon)}}_{I_1} + \underbrace{|G\varphi_p(g)|\frac{\varphi_p^2(g)}{(v+\epsilon)\sqrt{\tilde{v}+\epsilon}}}_{I_2}.
\end{aligned}$$

Consequently, we will estimate I_1 and I_2 by the inequality

$$|xy| \leqslant \frac{\lambda x^2}{2} + \frac{y^2}{2\lambda}.$$

For I_1, by taking

$$|x| = \frac{|G|}{\sqrt{\tilde{v}+\epsilon}},\ |y| = \frac{|\varphi_p(g)|\mathbb{E}_t[\varphi_p^2(g)]}{\sqrt{v+\epsilon}\sqrt{\tilde{v}+\epsilon}},\ \lambda = \frac{\sigma\sqrt{\tilde{v}+\epsilon}}{2},$$

we obtain

$$\begin{aligned}
I_1 \leqslant& \frac{\sigma}{4}\frac{|G|^2}{\sqrt{\tilde{v}+\epsilon}} + \frac{1}{\sigma}\frac{(\varphi_p^2(g)(\mathbb{E}_t[\varphi_p^2(g)])^2}{(v+\epsilon)(\tilde{v}+\epsilon)^{3/2}}, \\
\mathbb{E}_t[I_1] \leqslant& \frac{\sigma}{4}\frac{|G|^2}{\sqrt{\tilde{v}+\epsilon}} + \frac{1}{\sigma}\frac{(\mathbb{E}_t[\varphi_p^2(g)])^2}{(\tilde{v}+\epsilon)^{3/2}}\mathbb{E}_t\left[\frac{\varphi_p^2(g)}{v+\epsilon}\right].
\end{aligned}$$

Given that $\sqrt{\mathbb{E}_t[\varphi_p^2(g)]} \leqslant \sqrt{\tilde{v}+\epsilon}$ and $\sqrt{\mathbb{E}_t[\varphi_p^2(g)]} \leqslant R^p$, we can simplify the above estimate as:

$$\mathbb{E}_t[I_1] \leqslant \frac{\sigma}{4}\frac{|G|^2}{\sqrt{\tilde{v}+\epsilon}} + \frac{R^p}{\sigma}\mathbb{E}_t\left[\frac{\varphi_p^2(g)}{v+\epsilon}\right].$$

For I_2, by taking

$$|x| = \frac{|G|}{\sqrt{\tilde{v}+\epsilon}},\ |y| = \frac{|\varphi_p(g)|\varphi_p^2(g)}{v+\epsilon},\ \lambda = \frac{\sigma\varphi_p^2(g)}{2\mathbb{E}_t[\varphi_p^2(g)]},$$

we obtain

$$I_2 \leqslant \frac{\sigma}{4}\frac{\varphi_p^2(g)}{\mathbb{E}_t[\varphi_p^2(g)]}\frac{|G|^2}{\sqrt{\tilde{v}+\epsilon}} + \frac{1}{\sigma}\frac{\mathbb{E}_t[\varphi_p^2(g)]}{\sqrt{\tilde{v}+\epsilon}}\frac{\varphi_p^4(g)}{(v+\epsilon)^2}$$

Given that $\varphi_p^2(g) \leqslant v+\epsilon$, we can simplify the above estimate as:

$$I_2 \leqslant \frac{\sigma}{4}\frac{\varphi_p^2(g)}{\mathbb{E}_t[\varphi_p^2(g)]}\frac{|G|^2}{\sqrt{\tilde{v}+\epsilon}} + \frac{1}{\sigma}\frac{\mathbb{E}_t[\varphi_p^2(g)]}{\sqrt{\tilde{v}+\epsilon}}\frac{\varphi_p^2(g)}{v+\epsilon}.$$

Using $\sqrt{\mathbb{E}_t[\varphi_p^2(g)]} \leqslant \sqrt{\tilde{v}+\epsilon}$, $\sqrt{\mathbb{E}_t[\varphi_p^2(g)]} \leqslant R^p$, and taking the conditional expectation, we obtain:

$$\mathbb{E}_t[I_2] \leqslant \frac{\sigma}{4}\frac{|G|^2}{\sqrt{\tilde{v}+\epsilon}} + \frac{R^p}{\sigma}\mathbb{E}\left[\frac{\varphi_p^2(g)}{v+\epsilon}\right].$$

Consequently, combing the two estimates of I_1 and I_2, we obtain:

$$\mathbb{E}_t[|I|] \leqslant \mathbb{E}_t[I_1] + \mathbb{E}_t[I_2] \leqslant \frac{\sigma}{2}\frac{|G|^2}{\sqrt{\tilde{v}+\epsilon}} + \frac{2R^p}{\sigma}\mathbb{E}\left[\frac{\varphi_p^2(g)}{v+\epsilon}\right].$$

Putting the above estimate into Equation (8), we obtain the lower bound:

$$\begin{aligned}\mathbb{E}_t\left[\frac{G\varphi_p(g)}{\sqrt{v+\epsilon}}\right] =& \mathbb{E}_t\left[\frac{G\varphi_p(g)}{\sqrt{\tilde{v}+\epsilon}}\right] + \mathbb{E}_t\left[I\right] \geqslant \mathbb{E}_t\left[\frac{G\varphi_p(g)}{\sqrt{\tilde{v}+\epsilon}}\right] - \mathbb{E}_t[|I|] \\ \geqslant& \frac{\mathbb{E}_t\left[G\varphi_p(g)\right]}{\sqrt{\tilde{v}+\epsilon}} - \frac{\sigma}{2}\frac{|G|^2}{\sqrt{\tilde{v}+\epsilon}} - \frac{2R^p}{\sigma}\mathbb{E}_t\left[\frac{\varphi_p^2(g)}{v+\epsilon}\right].\end{aligned}$$

□

The second lemma estimate the sum of the updates in adaptive methods.

Lemma D.2 (Lemma 5.2 in Défossez et al. (2022)). *Let $\{a_t\}_{t\in\mathbb{N}}$ be a non-negative sequence, $\epsilon > 0$. Then for all $T \in \mathbb{N}$, we have:*

$$\sum_{t=1}^{T}\frac{a_t}{\epsilon + \sum_{s=1}^{t} a_s} \leqslant \log\left(1 + \frac{1}{\epsilon}\sum_{t=1}^{T} a_t\right).$$

D.2. Proof of Theorem 5.8

With the help of the above Lemma D.1 and D.2, we can prove Theorem 5.8.

Proof of Theorem 5.8.
Due to the H-smoothness, we have the quadratic upper bound:

$$\mathcal{L}(\boldsymbol{\theta}_{t+1}) \leqslant \mathcal{L}(\boldsymbol{\theta}_t) - \eta \left\langle \nabla \mathcal{L}(\boldsymbol{\theta}_t), \boldsymbol{u}_t \right\rangle + \frac{\eta^2 H}{2} \left\| \boldsymbol{u}_t \right\|_2^2 .$$

Taking the expectation at t, we have:

$$\begin{aligned}
\mathbb{E}_t \left[\mathcal{L}(\boldsymbol{\theta}_{t+1}) \right] \leqslant & \mathcal{L}(\boldsymbol{\theta}_t) - \eta \mathbb{E}_t \left[\langle \nabla \mathcal{L}(\boldsymbol{\theta}_t), \boldsymbol{u}_t \rangle \right] + \frac{\eta^2 H}{2} \mathbb{E}_t \left[\left\| \boldsymbol{u}_t \right\|_2^2 \right] \\
= & \mathcal{L}(\boldsymbol{\theta}_t) - \eta \sum_{i=1}^{d} \mathbb{E}_t \left[\nabla_i \mathcal{L}(\boldsymbol{\theta}_t) u_{t,i} \right] + \sum_{i=1}^{d} \frac{\eta^2 H}{2} \mathbb{E}_t \left[u_{t,i}^2 \right] .
\end{aligned}$$

Combine Lemma D.1 with $\sigma = c$ and Assumption 5.6, we get

$$\begin{aligned}
\mathbb{E}_t \left[\nabla_i \mathcal{L}(\boldsymbol{\theta}) u_{t,i} \right] = \mathbb{E}_t \left[\frac{\nabla_i \mathcal{L}(\boldsymbol{\theta}) \varphi_p(g_{t,i})}{\sqrt{v_{t,i} + \epsilon}} \right] & \geqslant c \frac{|\nabla_i \mathcal{L}(\boldsymbol{\theta})|^{p+1}}{\sqrt{\tilde{v}_{t,i} + \epsilon}} - \frac{c}{2} \frac{|\nabla_i \mathcal{L}(\boldsymbol{\theta})|^2}{\sqrt{\tilde{v}_{t,i} + \epsilon}} - \frac{2R^p}{c} \mathbb{E} \left[\frac{\varphi_p^2(g_{t,i})}{v_{t,i} + \epsilon} \right] \\
& \geqslant \frac{c}{2} \frac{|\nabla_i \mathcal{L}(\boldsymbol{\theta})|^{p+1}}{\sqrt{\tilde{v}_{t,i} + \epsilon}} - \frac{2R^p}{c} \mathbb{E} \left[\frac{\varphi_p^2(g_{t,i})}{v_{t,i} + \epsilon} \right] .
\end{aligned}$$

Where last inequality comes from $R < 1$. Using it for each dimension, we have:

$$\begin{aligned}
\mathbb{E}_t \left[\mathcal{L}(\boldsymbol{\theta}_{t+1}) \right] \leqslant & \mathcal{L}(\boldsymbol{\theta}_t) - \frac{\eta c}{2} \frac{|\nabla_i \mathcal{L}(\boldsymbol{\theta}_t)|^{p+1}}{\sqrt{\tilde{v}_{t,i} + \epsilon}} + \frac{2\eta R^p}{c} \mathbb{E} \left[\frac{\varphi_p^2(g_{t,i})}{v_{t,i} + \epsilon} \right] + \sum_{i=1}^{d} \frac{\eta^2 H}{2} \mathbb{E}_t \left[u_{t,i}^2 \right] \\
= & \mathcal{L}(\boldsymbol{\theta}_t) - \sum_{i=1}^{d} \frac{\eta c}{2} \frac{|\nabla_i \mathcal{L}(\boldsymbol{\theta}_t)|^{p+1}}{\sqrt{\tilde{v}_{t,i} + \epsilon}} + \sum_{i=1}^{d} \left(\frac{2\eta R^p}{c} + \frac{\eta^2 H}{2} \right) \mathbb{E}_t \left[\frac{\varphi_p^2(g_{t,i})}{v_{t,i} + \epsilon} \right] .
\end{aligned}$$

Noticing $\sqrt{\tilde{v}_{t_i} + \epsilon} \leqslant R^p \sqrt{t}$, we further have:

$$\mathbb{E}_t \left[\mathcal{L}(\boldsymbol{\theta}_{t+1}) \right] \leqslant \mathcal{L}(\boldsymbol{\theta}_t) - \frac{\eta c}{2} \frac{\|\nabla \mathcal{L}(\boldsymbol{\theta}_t)\|_{p+1}^{p+1}}{R^p \sqrt{t}} + \sum_{i=1}^{d} \left(\frac{2\eta R^p}{c} + \frac{\eta^2 H}{2} \right) \mathbb{E}_t \left[\frac{\varphi_p^2(g_{t,i})}{v_{t,i} + \epsilon} \right] .$$

Summing the previous inequality for all $0 \leqslant t \leqslant T-1$, taking the complete expectation, and using $\sqrt{t} \leqslant \sqrt{T}$, we have:

$$\mathbb{E} \left[\mathcal{L}(\boldsymbol{\theta}_t) \right] \leqslant \mathcal{L}(\boldsymbol{\theta}_0) - \frac{\eta c \sum_{t=1}^{T} \|\nabla \mathcal{L}(\boldsymbol{\theta}_t)\|_{p+1}^{p+1}}{2\eta R^p \sqrt{T}} + \sum_{i=1}^{d} \left(\frac{2R^p}{c} + \frac{\eta^2 H}{2} \right) \mathbb{E} \left[\sum_{t=1}^{T} \frac{\varphi_p^2(g_{t,i})}{v_{t,i} + \epsilon} \right] .$$

Then for each dimension, using Lemma D.2 for the sequence $\{(g_{t,i}^p)^2\}_{1 \leqslant t \leqslant T}$, we obtain:

$$\begin{aligned}
& \mathbb{E} \left[\mathcal{L}(\boldsymbol{\theta}_t) \right] \\
\leqslant & \mathcal{L}(\boldsymbol{\theta}_0) - \frac{\eta c \sum_{t=1}^{T} \mathbb{E} \|\nabla \mathcal{L}(\boldsymbol{\theta}_t)\|_{p+1}^{p+1}}{2R^p \sqrt{T}} + \left(\frac{2\eta R^p}{c} + \frac{\eta^2 H}{2} \right) d \, \mathbb{E} \left[\log \left(1 + \frac{1}{\epsilon} \sum_{t=1}^{T} \varphi_p^2(g_{t,i}) \right) \right] \\
\leqslant & \mathcal{L}(\boldsymbol{\theta}_0) - \frac{\eta c \sum_{t=1}^{T} \mathbb{E} \|\nabla \mathcal{L}(\boldsymbol{\theta}_t)\|_{p+1}^{p+1}}{2R^p \sqrt{T}} + \left(\frac{2\eta R^p}{c} + \frac{\eta^2 H}{2} \right) d \log \left(1 + \frac{R^{2p}}{\epsilon} T \right) .
\end{aligned}$$

This implies:

$$\begin{aligned}
&\mathbb{E}\min_{1\leqslant t\leqslant T}\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|_{p+1}^{p+1}\leqslant\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|_{p+1}^{p+1}\\
\leqslant&\frac{2R^p}{c\sqrt{T}}\left(\frac{\mathcal{L}(\boldsymbol{\theta}_0)-\mathcal{L}^\star}{\eta}+\left(\frac{2R^p}{c}+\frac{\eta H}{2}\right)d\log\left(1+\frac{R^{2p}}{\epsilon}T\right)\right)=\mathcal{O}\left(\frac{\log T}{\sqrt{T}}\right).
\end{aligned}$$

Hence

$$\mathbb{E}\min_{1\leqslant t\leqslant T}\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|_{p+1}^{p+1}\leqslant\mathcal{O}\left(\frac{\log T}{\sqrt{T}}\right).$$

□

D.3. Proof of Theorem 5.11

With the help of the above Lemma D.1 and D.2, we can prove Theorem 5.11.

Proof of Theorem 5.11.
Due to the H-smoothness, we have the quadratic upper bound:

$$\mathcal{L}(\boldsymbol{\theta}_{t+1})\leqslant\mathcal{L}(\boldsymbol{\theta}_t)-\eta\left\langle\nabla\mathcal{L}(\boldsymbol{\theta}_t),\boldsymbol{u}_t\right\rangle+\frac{\eta^2H}{2}\|\boldsymbol{u}_t\|_2^2.$$

Taking the expectation at t, we have:

$$\begin{aligned}
\mathbb{E}_t\left[\mathcal{L}(\boldsymbol{\theta}_{t+1})\right]\leqslant&\mathcal{L}(\boldsymbol{\theta}_t)-\eta\mathbb{E}_t\left[\langle\nabla\mathcal{L}(\boldsymbol{\theta}_t),\boldsymbol{u}_t\rangle\right]+\frac{\eta^2H}{2}\mathbb{E}_t\left[\|\boldsymbol{u}_t\|_2^2\right]\\
=&\mathcal{L}(\boldsymbol{\theta}_t)-\eta\sum_{i=1}^{d}\mathbb{E}_t\left[\nabla_i\mathcal{L}(\boldsymbol{\theta}_t)u_{t,i}\right]+\sum_{i=1}^{d}\frac{\eta^2H}{2}\mathbb{E}_t\left[u_{t,i}^2\right].
\end{aligned}$$

Combine Lemma D.1 with Assumption 5.9, we get

$$\mathbb{E}_t\left[\nabla_i\mathcal{L}(\boldsymbol{\theta})u_{t,i}\right]=\mathbb{E}_t\left[\frac{\nabla_i\mathcal{L}(\boldsymbol{\theta})\varphi_p(g_{t,i})}{\sqrt{v_{t,i}+\epsilon}}\right]\geqslant\frac{\sigma}{2}\frac{|\nabla_i\mathcal{L}(\boldsymbol{\theta})|^2}{\sqrt{\tilde{v}_{t,i}+\epsilon}}-\frac{2R^p}{\sigma}\mathbb{E}\left[\frac{\varphi_p^2(g_{t,i})}{v_{t,i}+\epsilon}\right].$$

Using it for each dimension, we have:

$$\begin{aligned}
\mathbb{E}_t\left[\mathcal{L}(\boldsymbol{\theta}_{t+1})\right]\leqslant&\mathcal{L}(\boldsymbol{\theta}_t)-\frac{\eta\sigma}{2}\frac{|\nabla_i\mathcal{L}(\boldsymbol{\theta}_t)|^2}{\sqrt{\tilde{v}_{t,i}+\epsilon}}+\frac{2\eta R^p}{\sigma}\mathbb{E}\left[\frac{\varphi_p^2(g_{t,i})}{v_{t,i}+\epsilon}\right]+\sum_{i=1}^{d}\frac{\eta^2H}{2}\mathbb{E}_t\left[u_{t,i}^2\right]\\
=&\mathcal{L}(\boldsymbol{\theta}_t)-\sum_{i=1}^{d}\frac{\eta\sigma}{2}\frac{|\nabla_i\mathcal{L}(\boldsymbol{\theta}_t)|^2}{\sqrt{\tilde{v}_{t,i}+\epsilon}}+\sum_{i=1}^{d}\left(\frac{2\eta R^p}{\sigma}+\frac{\eta^2H}{2}\right)\mathbb{E}_t\left[\frac{\varphi_p^2(g_{t,i})}{v_{t,i}+\epsilon}\right].
\end{aligned}$$

Noticing $\sqrt{\tilde{v}_{t_i}+\epsilon}\leqslant R^p\sqrt{t}$, we further have:

$$\mathbb{E}_t\left[\mathcal{L}(\boldsymbol{\theta}_{t+1})\right]\leqslant\mathcal{L}(\boldsymbol{\theta}_t)-\frac{\eta\sigma}{2}\frac{\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|_2^2}{R^p\sqrt{t}}+\sum_{i=1}^{d}\left(\frac{2\eta R^p}{\sigma}+\frac{\eta^2H}{2}\right)\mathbb{E}_t\left[\frac{\varphi_p^2(g_{t,i})}{v_{t,i}+\epsilon}\right].$$

Summing the previous inequality for all $0\leqslant t\leqslant T-1$, taking the complete expectation, and using $\sqrt{t}\leqslant\sqrt{T}$, we have:

$$\mathbb{E}\left[\mathcal{L}(\boldsymbol{\theta}_t)\right]\leqslant\mathcal{L}(\boldsymbol{\theta}_0)-\frac{\eta\sigma\sum_{t=1}^{T}\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|_2^2}{2\eta R^p\sqrt{T}}+\sum_{i=1}^{d}\left(\frac{2R^p}{\sigma}+\frac{\eta^2H}{2}\right)\mathbb{E}\left[\sum_{t=1}^{T}\frac{\varphi_p^2(g_{t,i})}{v_{t,i}+\epsilon}\right].$$

Then for each dimension, using Lemma D.2 for the sequence $\{(g_{t,i}^p)^2\}_{1\leqslant t\leqslant T}$, we obtain:

$$\begin{aligned}
&\mathbb{E}\left[\mathcal{L}(\boldsymbol{\theta}_t)\right] \\
\leqslant &\mathcal{L}(\boldsymbol{\theta}_0) - \frac{\eta\sigma\sum_{t=1}^T \mathbb{E}\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|_2^2}{2R^p\sqrt{T}} + \left(\frac{2\eta R^p}{\sigma} + \frac{\eta^2 H}{2}\right) d\,\mathbb{E}\left[\log\left(1 + \frac{1}{\epsilon}\sum_{t=1}^T \varphi_p^2(g_{t,i})\right)\right] \\
\leqslant &\mathcal{L}(\boldsymbol{\theta}_0) - \frac{\eta\sigma\sum_{t=1}^T \mathbb{E}\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|_2^2}{2R^p\sqrt{T}} + \left(\frac{2\eta R^p}{\sigma} + \frac{\eta^2 H}{2}\right) d\log\left(1 + \frac{R^{2p}}{\epsilon}T\right).
\end{aligned}$$

This implies:

$$\begin{aligned}
&\mathbb{E}\min_{1\leqslant t\leqslant T}\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|_2^2 \leqslant \frac{1}{T}\sum_{t=1}^T \mathbb{E}\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|_2^2 \\
\leqslant &\frac{2R^p}{\sigma\sqrt{T}}\left(\frac{\mathcal{L}(\boldsymbol{\theta}_0) - \mathcal{L}^\star}{\eta} + \left(\frac{2R^p}{\sigma} + \frac{\eta H}{2}\right) d\log\left(1 + \frac{R^{2p}}{\epsilon}T\right)\right) \\
\leqslant &\frac{R^{p-1}}{\sigma}\frac{2R}{\sqrt{T}}\left(\frac{\mathcal{L}(\boldsymbol{\theta}_0) - \mathcal{L}^\star}{\eta} + \left(2R + \frac{\eta H}{2}\right) d\log\left(1 + \frac{R^2}{\epsilon}T\right)\right) \\
= &\frac{R^{p-1}}{\sigma}\Big(\text{R.H.S. of (5)}\Big).
\end{aligned}$$

The last inequality comes from Assumption 5.9 and $R < 1$.

□