Optimistic Dual Averaging Unifies Modern Optimizers

Thomas Pethick
Harvard University
tmpethick@gmail.com

Wanyun Xie
EPFL (LIONS)
wanyun.xie@epfl.ch

Roman Machacek
University of Bern
roman.machacek@unibe.ch

Volkan Cevher
EPFL (LIONS)
volkan.cevher@epfl.ch

Abstract

We introduce SODA, a generalization of Optimistic Dual Averaging, which provides a common perspective on state-of-the-art optimizers like Muon, Lion, AdEMAMix and NAdam, showing that they can all be viewed as optimistic instances of this framework. Based on this framing, we propose a practical SODA wrapper for any base optimizer that eliminates weight decay tuning through a theoretically-grounded $1/k$ decay schedule. Empirical results across various scales and training horizons show that SODA consistently improves performance without any additional hyperparameter tuning.

1 Introduction

Deep learning optimization has developed along two largely complementary axes. The first axis is *geometry and adaptation*: designing updates whose implicit norm, preconditioner, or constraint set reflects the structure of modern models, potentially improving scaling in high dimensions. Adam [Kingma and Ba, 2014] is a central example, coupling gradient averaging with an ℓ_∞-type geometry through elementwise normalization. In parallel, *spectral* geometry and stochastic dualization were developed in Stochastic Spectral Descent (SSD) [Carlson et al., 2015a, 2016], and are now resurfacing in recent methods such as Muon and Scion [Jordan et al., 2024b, Pethick et al., 2025a].

The second axis is *composition of ingredients*: how gradient feedback and iterates are combined through momentum, averaging, and schedules to obtain a stable and performant training recipe. Algorithms such as Lion [Chen et al., 2023] and practical recipes such as the schedule-free wrapper [Defazio et al., 2024] highlight that the assembly of these ingredients can be as important as the ingredients themselves. A striking pattern over the last decade is that many seemingly disparate new optimizers can be explained as different points in this 2D design space.

A useful lens for both axes is the classical dualization framework, where the next iterate is obtained by minimizing a linear surrogate regularized by a geometry-inducing term:

$$x^{k+1} \in \arg\min_{x\in\mathcal{X}} \gamma_k \langle d^k, x\rangle + h_k(x). \tag{1}$$

Here, $\gamma_k > 0$ is a stepsize schedule, d^k is a (possibly averaged) gradient feedback, and h_k is a (possibly time-varying) regularizer/mirror map that determines the geometry.

For instance, choosing $h_k(x) = \frac{1}{2}\|x - x^k\|_2^2$ recovers gradient descent, while non-Euclidean choices of h_k yield normalized and geometry-aware updates, that can depend more favorably on the dimensionality of the problem. In parallel, *how* we form d^k (momentum, averaging, extrapolation) and *how* we average iterates (primal averaging / schedule-free) have a strong effect on the properties of the resulting method, including noise robustness, acceleration, and anytime guarantees.

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

Table 1: Instances of SODA for $\bar{\lambda}_k = 0$ (*c.f.* Section 3).

Method	**Reference**	**Grad. avg** ($\alpha_k < 1$)	**Optimism** ($\bar{\alpha}_k > 0$)	**Geometry** (h_k)
Adam	Kingma and Ba [2014]	✓	✗	Smoothed ℓ_∞
NAdam[1]	Dozat [2016]	✓	✓	Smoothed ℓ_∞
Stoch. ℓ_∞	Carlson et al. [2015a, 2016]	✗	✗	ℓ_∞
Signum	Bernstein et al. [2018]	✓	✗	ℓ_∞
Lion	Chen et al. [2023]	✓	✓	ℓ_∞
Shampoo	Gupta et al. [2018]	✗	✗	Smoothed Spectral
SSD	Carlson et al. [2015a, 2016]	✗	✗	Spectral
Scion (Spectral)	Pethick et al. [2025a]	✓	✗	Spectral
Muon	Jordan et al. [2024b]	✓	✓	Spectral
Multi-Norm	Scetbon et al. [2025]	✗	✗	Doubly Stochastic

[1] Rediscovered as Simplified-AdEMAMix [Morwani et al., 2025].

One major challenge is how to set hyperparameters of these optimizers, which becomes particularly important as model size becomes prohibitively expensive to tune and we instead seek predictable scaling rules.

One particularly challenging parameter is weight decay. In the context of multi-epoch training, weight decay has a precise characterization, since it constrains the norm of the iterates, thus acting as a regularizer [Xie and Li, 2024, Pethick et al., 2025a]. However, even in single-epoch training, where overfitting is not a concern, weight decay can surprisingly still be beneficial.

Very recently a scaling rule was developed choosing weight decay as $1/d$ with model dimension d [Xiao, 2024, Qiu et al., 2025]. However, since many large-scale experiments follow compute-optimal training regimes where d and the training horizon n are coupled, it is difficult to disentangle whether the effective dependence is on d or on time. Specifically, Wang and Aitchison [2024], Bergsma et al. [2025] instead suggest weight decay should scale as $1/n$. In addition, weight decay typically needs to be tuned on a per-optimizer basis, as observed for Lion, which requires a significantly larger weight decay than AdamW [Chen et al., 2023]. It raises the following question:

Is it possible to ground these two axes of recent progress in classical methods and provide guidance on hyperparameters?

In this paper, we contend that the two axes of algorithmic development are not separate threads. To this end, we introduce SODA, a generalization of Optimistic Dual Averaging [Rakhlin and Sridharan, 2013] that explicitly couples *dual processing* (gradient averaging + optimism) and *primal processing* (iterate averaging + primal extrapolation). The resulting framework provides a single template that recovers several widely used optimizers as special cases and makes it straightforward to inject non-Euclidean geometry, such as ℓ_∞ or spectral geometry, into modern training recipes.

In particular, when one of the averaging parameters is simplified (our "modernized" regime), SODA yields a practical wrapper around any base optimizer that eliminates weight decay tuning via a theoretically grounded $1/k$ decay.

Contributions. Our contributions are as follows:

- (i) *Unification via optimism and dualization:* We show that SODA provides a unified perspective on several state-of-the-art optimizers. Most notably, it captures Muon, Lion-$\mathcal{K}$, and NAdam as *optimistic* instances within this broad framework (*cf.* Table 1).
- (ii) *Theoretical guarantees:* The SODA framing allows us to theoretically derive hyperparameter choices that works remarkably well in practice. Our formulation provides a new perspective on weight decay, not only as a regularizer, but as a form of primal averaging. This results in a new update rule that anchors each update at the initialization and induces a $1/k$ weight decay schedule.
- (iii) *A simple wrapper:* We propose a practical wrapper around any base optimizer (e.g., Adam, Lion, Muon, Scion) that introduces no new hyperparameters and removes the need to tune a weight decay parameter.
- (iv) *Empirical results without additional tuning:* We demonstrate consistent performance improvements when wrapping Adam, Muon, and Scion across various model sizes and training horizons, even outperforming baselines with tuned weight decay. Our proposed instantiation

of our framework, SODA†, removes the need to tune weight decay while leading to better performance at scale.

Limitations This work focuses on the single-epoch setting. For multi-epoch training it might be necessary to have a hyperparameter controlling the regularization strength, due to overfitting.

2 Preliminaries

We are interested in the following minimization problem

$$\min_{x\in\mathcal{X}} f(x) := \mathbb{E}_\xi[f(x,\xi)], \tag{2}$$

which covers a host of machine learning problems, where ξ represents a data sample, x is the parameter of the model being optimized, and $f(x,\xi)$ is a loss function.

Dual averaging and Fenchel conjugate The Dual Averaging framework [Nesterov, 2009] and its variants typically rely on the Fenchel conjugate to map gradient information back to the primal space. The Fenchel conjugate of a function $h : \mathcal{X} \to \mathbb{R} \cup \{\infty\}$ is defined as:

$$h^*(d) = \sup_{x\in\mathcal{X}} \left\{\langle d, x\rangle - h(x)\right\},$$

where $d \in \mathcal{X}^*$. The subdifferential ∂h^* is equivalent to the set of maximizers of this conjugate operation:

$$\partial h^*(d) = \operatorname*{argmax}_{x\in\mathcal{X}} \left\{\langle d, x\rangle - h(x)\right\}.$$

This identity is a direct consequence of the Fenchel-Young inequality [Bauschke and Lucet, 2012]. The general optimization template (1) can then be written compactly as:

$$x^{k+1} \in \partial h_k^*(-\gamma_k d^k), \tag{3}$$

where d^k is an average (or momentum) of past stochastic gradients, γ_k is a stepsize schedule, and h_k is a sequence of geometry defining regularizers (or mirror maps).

Standard regularizers Different choices of the regularizer $h(x)$ recover well-known optimization steps. We highlight three instances that are particularly relevant for our work, especially when the constraint set is the norm ball $\mathcal{D} = \{x \in \mathcal{X} \mid \|x\| \le 1\}$ for some arbitrary norm $\|\cdot\|$:

(i) *Unconstrained:* Let $h(x) = \frac{1}{2}\|x\|^2$. Then, the ∂h^* is the sharp operator [Nesterov, 2012]:

$$\partial h^*(d) = [d]^\sharp := \arg\max_x \left\{\langle d, x\rangle - \tfrac{1}{2}\|x\|^2\right\}.$$

(ii) *Linear minimization oracle (*lmo*):* Let $h(x) = \iota_\mathcal{D}(x)$ be the indicator function of a set $\mathcal{D}$ (i.e., 0 if $x \in \mathcal{D}$ and ∞ otherwise). The ∂h^* is then the lmo:

$$\partial h^*(-d) = \mathrm{lmo}_\mathcal{D}(d) := \arg\min_{x\in\mathcal{D}} \langle d, x\rangle.$$

This is the core operation in Frank-Wolfe (conditional gradient) methods [Frank and Wolfe, 1956, Clarkson, 2010, Jaggi, 2013], which are projection-free but typically require f to be smooth.

(iii) *Clipping:* By combining the above, we can handle a non-smooth f via "clipping." Let $h(x) = \frac{1}{2}\|x\|^2 + \iota_\mathcal{D}(x)$. Then, we have

$$\partial h^*(-d) = \mathrm{clip}_\mathcal{D}(d) := \arg\min_{x\in\mathcal{D}}\{\langle d, x\rangle + \tfrac{1}{2}\|x\|^2\}.$$

This operation clips the gradient step into the feasibility set $\mathcal{D}$, essentially performing a projection of the negative gradient. For the Euclidean norm, this is the classical gradient norm clipping, which has a long history in training neural networks [Pascanu et al., 2013, Zhang et al., 2019a]. Clipping with respect to non-Euclidean norms was more recently used in Pethick et al. [2025b], Crawshaw et al. [2025] for training neural networks.

These three choices of ∂h^* can all be related through the lmo, since $[d]^\# = -\|d\|_* \,\mathrm{lmo}_\mathcal{D}(d)$ and $\mathrm{clip}_\mathcal{D}(d) = \min\{1, \|d\|_*\}\,\mathrm{lmo}_\mathcal{D}(d)$ (see Theorem A.1).

Norm choices Important special cases arise by taking $h = \iota_{\mathcal{D}}$ with $\mathcal{D}$ the unit ball of a norm. For the ℓ_∞ ball, $\mathrm{lmo}_{\mathcal{D}}(d) = -\operatorname{sign}(d)$ as used in SignSGD and Lion [Bernstein et al., 2018, Chen et al., 2023]. For the spectral norm ball, $\mathrm{lmo}_{\mathcal{D}}(G) = -\operatorname{msign}(G)$ with $\operatorname{msign}(G) := UV^\top$, where $U\Sigma V^\top$ is the singular value decomposition of G as used in Carlson et al. [2016], Jordan et al. [2024b], Pethick et al. [2025a]. In practice, msign is not computed exactly, but approximated with a few Newton-Schulz iterations [Jordan et al., 2024b] or PolarExpress [Amsel et al., 2025] (see Section B.3 for the implications for our theory).

3 Method

We propose the following algorithm which generalizes Optimistic Dual Averaging (ODA) by introducing a primal extrapolation sequence (y^k) from Tseng [2008], Lan [2012], Defazio et al. [2024]:

$$\begin{aligned} m^{k+1} &= (1-\alpha_k)m^k + \alpha_k \nabla f(y^k, \xi_k) \\ \bar{m}^{k+1} &= (1-\bar{\alpha}_k)m^{k+1} + \bar{\alpha}_k \nabla f(y^k, \xi_k) \quad \text{(optimism)} \\ z^{k+1} &\in \partial h_k^*(-\gamma_k \bar{m}^{k+1}) = \arg\min_{x \in \mathcal{X}} \gamma_k \langle \bar{m}^{k+1}, x \rangle + h_k(x) \\ x^{k+1} &= (1-\lambda_k)x^k + \lambda_k z^{k+1} \\ y^{k+1} &= (1-\bar{\lambda}_k)x^{k+1} + \bar{\lambda}_k z^{k+1} \quad \text{(primal extrapolation)} \end{aligned} \tag{SODA}$$

with $\alpha_k, \bar{\alpha}_k, \lambda_k, \bar{\lambda}_k \in [0,1]$ and $\gamma_k > 0$. We initialize $m^0 = 0$ and $x^0 = y^0 = z^0 \in \partial h_0^*(0)$. Observe the elegant symmetry between how the dual (gradients) and primal (iterates) are being processed. Our analysis in Section 4 builds on the Schedule-Free framework of Defazio et al. [2024] and ODA Rakhlin and Sridharan [2013], so we refer to the method as SODA to pay homage.

Note that typically in machine learning libraries such as PyTorch the momentum parameters are instead defined as $\beta_k = 1 - \alpha_k$, $\bar{\beta}_k = 1 - \bar{\alpha}_k$, $\tau_k = 1 - \lambda_k$, and $\bar{\tau}_k = 1 - \bar{\lambda}_k$.

3.1 Special Cases

There are two important extremes depending on the choice of the primal extrapolation parameter $\bar{\lambda}_k$.

Optimistic Dual Averaging For $\bar{\lambda}_k = 1$:

$$\begin{aligned} m^{k+1} &= (1-\alpha_k)m^k + \alpha_k \nabla f(z^k, \xi_k) \\ \bar{m}^{k+1} &= (1-\bar{\alpha}_k)m^{k+1} + \bar{\alpha}_k \nabla f(z^k, \xi_k) \\ z^{k+1} &\in \partial h^*(-\gamma_k \bar{m}^{k+1}) \\ x^{k+1} &= (1-\lambda_k)x^k + \lambda_k z^{k+1} \end{aligned} \tag{ODA}$$

This is an optimistic version of the celebrated Dual Averaging scheme [Nesterov, 2005], also known as optimistic follow-the-regularized-leader (FTRL) in online learning [Rakhlin and Sridharan, 2013]. Notice that the output of the algorithm (x^k) can be different from where the gradient is evaluated (z^k).

Modernized Optimistic Dual Averaging For $\bar{\lambda}_k = 0$:

$$\begin{aligned} m^{k+1} &= (1-\alpha_k)m^k + \alpha_k \nabla f(x^k, \xi_k) \\ \bar{m}^{k+1} &= (1-\bar{\alpha}_k)m^{k+1} + \bar{\alpha}_k \nabla f(x^k, \xi_k) \\ x^{k+1} &= (1-\lambda_k)x^k + \lambda_k \partial h_k^*(-\gamma_k \bar{m}^{k+1}) \end{aligned} \tag{MODA}$$

This can be seen as an optimistic version of Double Averaging of Nesterov and Shikhman [2015]. We refer to this as "modernized" following Jelassi and Defazio [2020], which applied Double Averaging to deep learning. The output of the algorithm (x^k) is the same as where the gradient is evaluated (x^k).

This recovers Stochastic Frank-Wolfe and Scion [Mokhtari et al., 2020, Pethick et al., 2025a] with $\bar{\alpha}_k = 0$ and $h_k(x) = \iota_{\mathcal{D}}(x)$, since then $\partial h_k^*(-\gamma_k d) = \rho \,\mathrm{lmo}(\gamma_k d) = \rho \,\mathrm{lmo}(d)$ for some constrained radius $\rho > 0$ where the first equality follows from scale invariance of the lmo.

More importantly, for $\bar{\alpha}_k \neq 0$, MODA captures NAdam Dozat [2016], Lion [Chen et al., 2023] and Muon [Jordan et al., 2024b] with weight decay through the choice of geometry h_k:

Algorithm 1 SODA Wrapper

Input: Horizon n, initialization $z^0 = x^0 \in \mathcal{X}$

1: **for** $k = 0, \dots, n-1$ **do**
2: Sample $\xi_k \sim \mathcal{P}$ and compute the gradient $g^k = \nabla f(x^k, \xi_k)$
3: $z^{k+1} = z^0 + (k+2)\,\mathrm{BaseUpdate}(g^k)$
4: $x^{k+1} = (1 - \frac{1}{k+2})x^k + \frac{1}{k+2} z^{k+1}$

Return x^n

BaseUpdate refers to the update delta of the base optimizer *without weight decay*, i.e., $u^{k+1} = u^k + \mathrm{BaseUpdate}(g^k)$.

- Lion-$\mathcal{K}$: choose $h_k = \mathcal{K}^*$, so that $\partial h_k^* = \partial \mathcal{K}$. For Lion specifically, $\partial h_k^*(-u) = -\operatorname{sign}(u)$.
- Muon: choose a spectral mirror map h_k such that $\partial h_k^*(-u) = -\operatorname{msign}(u)$. The so-called Nesterov momentum corresponds to choosing the optimistic parameter as $\bar{\alpha}_k = \alpha_k$.
- NAdam: choose $h_k(x) = \frac{1}{2}\langle x, \mathrm{Diag}(\sqrt{v^k}+\varepsilon)x\rangle$, where $v^k = \tau v^{k-1} + (1-\tau)\nabla f(x^k, \xi_k) \odot \nabla f(x^k, \xi_k)$. Then $\nabla h_k^*(-\gamma_k \bar{m}^{k+1}) = -\gamma_k \bar{m}^{k+1} \oslash (\sqrt{v^k} + \varepsilon)$.

In this light, all of the above methods can be interpreted as optimistic versions of Dual Averaging. We note that NAdam was also rediscovered as a simplification of AdEMAMix [Pagliardini et al., 2024] named Simplified-AdEMAMix from [Morwani et al., 2025]. See Table 1 for an overview.

Notice that λ_k plays the role of the stepsize in this case. This is in contrast with the following where we will instead let $\lambda_k = 1/(k+2)$ and let η_k be the stepsize of the base optimizer (typically using a linear or cosine schedule).

SODA Wrapper In Algorithm 1, we provide a particularly practical instantiation of SODA, that wraps an existing base optimizer, based on hyperparameter choices from Theorem 4.6. Following the theory, this approach uses $\bar{\lambda}_k = 0$ (aka MODA), uniform iterate averaging ($\lambda_k = 1/k+2$), a stepsize $\gamma_k = (k+2)\eta_k$ and defines the regularizer relative to a reference iterate z^0 as $h_k(x) = \psi_k(x - z^0)$ with $\psi_k(x) := \frac{1}{2}\|x\|_{H_k}^2$ where H_k is a positive-definite matrix. Since $\partial \psi_k^*$ is positively homogeneous, we have $\partial h_k^*(-\gamma_k \bar{m}^{k+1}) = z^0 + \partial \psi_k^*(-\gamma_k \bar{m}^{k+1}) = z^0 + \gamma_k \partial \psi_k^*(-\bar{m}^{k+1})$.

Any optimizer based on (optimistic) gradient momentum and dualization, such as Adam, NAdam, Scion, Muon, Signum, or Lion, can serve as the base optimizer. Notice that the BaseUpdate involves the stepsize η_k, while the wrapper introduces no new hyperparameters.

The resulting Algorithm 1 has a clear interpretation when simplifying the expression:

$$x^{k+1} = \underbrace{\tfrac{1}{k+2} z^0}_{\text{centering}} + \underbrace{(1 - \tfrac{1}{k+2})x^k}_{\text{scheduled weight decay}} + \underbrace{\mathrm{BaseUpdate}(g^k)}_{\text{includes a stepsize schedule}} \tag{4}$$

Let us compare (4) with the original (*independent*) weight decay Hanson and Pratt [1988] formulation

$$x^{k+1} = (1 - \mu_k)x^k - \gamma_k \nabla f(x^k, \xi_k) \tag{5}$$

where $\mu_k \geq 0$. First the iterates in (4) are anchored in the initialization z^0. Second, the weight decay is decaying as $\mu_k = 1/k+2$ and is *not* multiplied by the stepsize schedule as otherwise standard in e.g., decoupled weight decay [Loshchilov and Hutter, 2019], whereas the base optimizer is multiplied with the stepsize schedule (absorbed into BaseUpdate).

The weight decay plays a different role in SODA than typically. It does not act as a regularizer, but is rather related to the averaging of iterates, which improves optimization. Remarkably, as we shall see in Section 5, the SODA wrapper (Algorithm 1) consistently improves upon the wrapped base optimizer, even outperforming the baseline with tuned weight decay while the SODA wrapper requires no such tuning.

Remark 3.1. Algorithm 1 appears to require storing the initialization z^0 throughout training. Since z^0 is typically generated from a fixed seed, it can instead be reconstructed on demand at each iteration, trading a small amount of computation for a method with no additional memory.

3.2 Related Work

We organize related work through the lens of SODA, which separates (i) dual averaging and optimism, (ii) primal averaging and primal extrapolation, and (iii) dualization through geometry (mirror maps). This decomposition clarifies how several modern optimizers arise as special cases.

Gradient averaging. Several works for deep learning have highlighted the benefit of maintaining multiple dual sequences. AdEMAMix [Pagliardini et al., 2024] showed that keeping an additional dual averaging sequence on top of Adam-style momentum can significantly improve performance. Simplified-AdEMAMix [Morwani et al., 2025] later demonstrated that explicitly adding back the current gradient is sufficient, yielding an update equivalent to NAdam [Dozat, 2016]. Similarly, Muon [Jordan et al., 2024b] uses the PyTorch implementation of Nesterov momentum, which can be captured by SODA with $\bar{\alpha}_k = \alpha_k$. The sign-based optimizer Lion [Chen et al., 2024] also moves beyond simple averaging. From the perspective of SODA, these gradient averaging schemes correspond precisely to forming the optimistic dual variable $\bar{m}^{k+1}$.

Iterate averaging. A complementary line of work in the deep learning literature focuses on averaging and extrapolation in the *primal* space. Lookahead [Zhang et al., 2019b] performs iterate averaging by maintaining a slow sequence (x^k) and generating extrapolated points (z^{k+1}) via an inner optimization loop, corresponding to the special case $\bar{\lambda} = 0$. DiLoCo [Douillard et al., 2023] replaces simple averaging in the outer optimizer with PyTorch-style Nesterov momentum, demonstrating improvements in distributed training and, subsequently, even in single-node settings [Kallusky et al., 2025].

Generalized Primal Averaging (GPA) [Defazio et al., 2025] further simplifies DiLoCo by showing that an explicit inner loop is unnecessary: it suffices to average the iterates (x^{k+1}) and extrapolate (y^{k+1}). Specifically, GPA can be written as

$$\begin{aligned} z^{k+1} &= (1-\eta_k\mu)z^k + \eta_k \,\mathrm{BaseUpdate}(\nabla f(y^k, \xi_k)), \\ x^{k+1} &= (1-\lambda)x^k + \lambda z^{k+1}, \\ y^{k+1} &= (1-\bar{\lambda})x^{k+1} + \bar{\lambda} z^{k+1}. \end{aligned} \tag{GPA}$$

where $\mu \geq 0$ and $\lambda, \bar{\lambda} \in [0, 1]$. GPA uses a stepsize schedule for the base optimizer's η_k and fixes the iterate-averaging parameters $(\lambda, \bar{\lambda})$. From the SODA viewpoint, GPA corresponds to constant primal averaging parameters $(\lambda, \bar{\lambda})$. Typical choices are $1 - \bar{\lambda} = 0.9$ and $1 - \lambda = (1 - \bar{\lambda})^{1/H} \approx 0.9967$ with $H = 32$, implying $\lambda < \bar{\lambda}$. This contrasts with the smooth SODA regime, which requires the sufficient condition $\bar{\lambda}_k \leq \lambda_k/10$ (Theorems B.7 and 4.6). Moreover, whereas GPA relies on a base optimizer *with* a tuned weight decay parameter μ, the SODA wrapper (Algorithm 1) wraps a base optimizer *without* weight decay and eliminates the need for any additional hyperparameters.

Weight decay Our construction shows that weight decay can be viewed as a form of primal averaging. Xiao [2024] observed a $1/d$ scaling with model size d and used it for hyperparameter transfer, while Qiu et al. [2025] adopted this rule for spectral methods such as Muon. In contrast, we separate model size and horizon effects: if the horizon is fixed and only model size varies, our framework does not predict a necessary $1/d$ scaling. Wang and Aitchison [2024] interpret AdamW as an exponential moving average and argue that its timescale should be held fixed relative to the training horizon, and Bergsma et al. [2025] derive scaling laws for this timescale in LLM pre-training; at a fixed relative timescale, both correspond to a weight decay that scales as $1/n$ with the horizon n. These empirical findings are in agreement with our theory, which grounds a $1/k$ schedule in optimization theory. Concurrently, Ferbach et al. [2026] considered the time-decaying rule $\lambda_k = \lambda\eta_k/k$. In contrast, the SODA wrapper uses the parameter-free schedule $\lambda_k = 1/(k+2)$, thereby also providing a theoretical motivation for such time-decaying schedules. Our work focuses on removing hyperparameters, while Ferbach et al. [2026] introduces new hyperparameters.

Alternatives to tuning weight decay. Weight decay plays at least three roles: (i) regularization through norm control [Xie and Li, 2024, Pethick et al., 2025a], (ii) control of the ratio between the weight and update norms, and (iii) improved rates through primal averaging, which is the focus of this work. Hyperball [Wen et al., 2026], HyperP [Ren et al., 2026], Spectral Sphere [Xie et al., 2026] and nGPT [Loshchilov et al., 2024] target (ii) directly through weight normalization instead of weight decay. The SODA wrapper on the other hand targets (iii) leading to a scheduled weight decay. The two are thus complementary, and combining them is an interesting direction for future work.

Acceleration and universality. Primal extrapolation, corresponding to $\bar{\lambda}_k > 0$ in SODA, originates in accelerated gradient and proximal-gradient methods [Tseng, 2008, Lan, 2012]. Querying gradients at the averaged point, corresponding to $\bar{\lambda}_k = 0$, was later used concurrently by Cutkosky [2019], Kavis et al. [2019] to obtain universal methods, i.e., a single algorithm simultaneously attaining both the optimal smooth stochastic convex rate $O(L/n^2 + \sigma/\sqrt{n})$ and the nonsmooth rate $O(1/\sqrt{n})$. Joulani et al. [2020] combined this averaging mechanism with adaptive Optimistic Dual Averaging/FTRL [Rakhlin and Sridharan, 2013, Mohri and Yang, 2016]. Defazio et al. [2024] later extended this analysis to allow for the larger range $\bar{\lambda}_k \leq \lambda_k/10$ and used it to develop a optimization wrapper for deep learning with unknown training horizon.

Dualization and geometry. The choice of mirror map h determines the geometry of the update and plays a central role in modern deep learning. Recent work has emphasized that many deep learning optimizers are best understood through their induced norm [Bernstein and Newhouse, 2024]. Elementwise sign methods are the simplest example: the ℓ_∞ geometry gives rise to SignSGD [Bernstein et al., 2018] and Lion [Chen et al., 2023], and has been used to partially explain the effectiveness of the popular Adam optimizer [Kunstner et al., 2023]. For matrix parameters, spectral descent methods [Carlson et al., 2015a, 2016, 2015b] and their modern variants, such as Muon [Jordan et al., 2024b] and Scion [Pethick et al., 2025a], arise from spectral geometries. Beyond single-norm geometries, multi-norm constructions enforcing both row- and column-normalization (doubly stochastic structure) have also been explored [Scetbon et al., 2025, Xie et al., 2025]. Whenever the mirror map admits a tractable Fenchel conjugate, such geometries naturally fit within the SODA framework. The conjugate need not be available in closed form: approximate evaluations of ∂h_k^* with absolutely summable errors can be handled by the standard argument for inexact methods (see, e.g., Schmidt et al. [2011]).

4 Analysis

We now derive convergence guarantees for SODA, in order to set the hyperparameters. The proof is based on an online regret argument, so we first let g^k denote an arbitrary gradient-feedback sequence. In the stochastic optimization setting of SODA, we take $g^k := \nabla f(y^k, \xi_k)$. We use the following assumptions, which are standard except for Assumption 4.4.

Assumption 4.1 (Convex)**.** *For every sample ξ, the function $f(\cdot, \xi)$ is convex.*

Assumption 4.2 (L-smooth)**.** *The function f is L-smooth with respect to $\|\cdot\|$.*

Assumption 4.3 (Unbiased)**.** *Let $\mathcal{F}_k$ be the natural filtration. The gradients satisfy*

$$\mathbb{E}[g^k \mid \mathcal{F}_{k-1}] \in \partial f(y^k).$$

Assumption 4.4 (Gradient variation)**.** *The gradients satisfy, for $k \geq 0$ and $\rho > 0$,*

$$\mathbb{E}\left[\left\|g^k - g^{k-1}\right\|_*^2\right] \leq \rho\, \mathbb{E}\left[\left\|\nabla f(y^k) - \nabla f(y^{k-1})\right\|_*^2\right] + \sigma^2.$$

Remark 4.5. Assumption 4.4 is satisfied under unbiasedness (Assumption 4.3) and bounded variance, $\mathbb{E}[\|g^k - \nabla f(y^k)\|_*^2] \leq v^2$, with $\rho = 3$ and $\sigma^2 = 6v^2$. The noise condition in Defazio et al. [2024, Cor. 2] is recovered in the Euclidean case with $\rho = 1$.

Our analysis is a fairly direct extension of Defazio et al. [2024] in that we combine their online-to-batch conversion for primal averaging and extrapolation with Optimistic Dual Averaging [Rakhlin and Sridharan, 2013] as the underlying no-regret algorithm, instead of an adaptive version of Optimistic Mirror Descent. The main purpose of the analysis is to predetermine the hyperparameters of the practical method in Algorithm 1. The technical differences are that we (i) allow for non-Euclidean norms, so that the rates depend on L and $R_\star$ measured in the chosen geometry, (ii) relax the noise condition by allowing $\rho > 1$ in Assumption 4.4, and (iii) cover constrained problems, whereas the argument of Defazio et al. [2024] relies on $\nabla f(x^\star) = 0$, which introduces an additional startup term that must be balanced against the deterministic and stochastic terms (see the proof of Theorem 4.6).

Corollary 4.6 (Convergence under L-smoothness)**.** *Let $x^\star \in \arg\min_{x \in \mathcal{X}} f(x)$ and let $R_\star := h(x^\star) - \inf h$. Consider SODA with a fixed regularizer $h_k \equiv h$. For every $k = 0, \ldots, n-1$, choose*

$$\alpha_k = \tfrac{1}{k+1}, \qquad \bar{\alpha}_k = \lambda_k = \tfrac{1}{k+2}, \qquad \bar{\lambda}_k \leq \tfrac{\lambda_k}{10}, \qquad \gamma_k = \eta(k+2), \qquad \eta = \min\left\{\tfrac{\mu}{6\rho L}, \tfrac{\sqrt{\mu R_\star}}{\sigma\sqrt{n}}\right\}.$$

Suppose Assumptions 4.1 to 4.4 hold and that h is μ-strongly convex with respect to $\|\cdot\|$. Then, for every $n \geq 1$,

$$\mathbb{E}[f(x^{n-1}) - f(x^\star)] = O\left(\frac{(\rho+1)LR_\star}{\mu n} + \frac{\sigma\sqrt{R_\star}}{\sqrt{\mu n}}\right).$$

Remark 4.7 (Agreement with empirical scaling laws). Theorem 4.6 anticipates, and is in remarkable agreement with, the empirical scaling laws recently reported in MAI [Microsoft AI, 2026]. They finds that $\mathrm{LR}^\star \times \mathrm{WD}^\star \propto 1/\mathrm{steps}$ and $\mathrm{LR}^\star \approx 1/\sqrt{\mathrm{steps}}$. Their effective weight decay $\mathrm{LR} \times \mathrm{WD}$ corresponds exactly to our $\lambda_k = 1/(k+2)$ ($\bar{\lambda} = 0$), and their learning rate to our $\lambda_k\gamma_k = \eta \propto 1/\sqrt{n}$.

Consequences for practice The SODA wrapper in Algorithm 1 directly uses the theoretically suggested choices of λ_k and γ_k. Taking the primal extrapolation constant $\bar{\lambda}_k$ small is used to exploit smoothness to cancel the gradient-variation term from Assumption 4.4. Since $\bar{\lambda}_k$ has no effect on the rate once it is small enough, we can conveniently set $\bar{\lambda}_k = 0$ in practice, matching the "modernized" parameterization in Section 3 and recovering Muon, Lion, and NAdam. The choice $\bar{\lambda}_k = 0$ (i.e., $y^k = x^k$) is moreover the one for which the analysis is tightest, since it avoids an application of Young's inequality and thereby improves the constants (Theorem B.6). Under the bounded-gradient analysis of Theorem B.4, the larger range $\bar{\lambda}_k \in [0, 1]$ is also admissible.

It is optimism that lets us establish an accelerate rate in Theorem B.7. Specifically, the smooth analysis requires the optimistic averaging parameter $\bar{\alpha}_k$ to be slightly smaller than α_k This is consistent with common choices: Lion often uses $\alpha_k = 0.1$ with $\bar{\alpha}_k \in \{0.05, 0.01\}$, while Muon takes $\alpha_k = \bar{\alpha}_k$. Finally, the rate depends on the initial regularizer gap $R_\star$. For $h(x) = \|x - z^0\|^2$, we have $R_\star = \|x^\star - z^0\|^2$, so choosing $z^0 = 0$ can be a poor anchor when the solution is far from the origin. This motivates the centered choice in Section 3.

Acceleration Theorem B.7 shows that SODA also admits an accelerated parameterization following Defazio et al. [2024, Cor. 1]. This is obtained by choosing $\alpha_k = 2/k+2$, $\bar{\alpha}_k = \lambda_k = 2/k+3$, and $\bar{\lambda}_k \leq \lambda_k/10$. Equivalently, this accelerated regime corresponds to using increasing weights $a_k = k+1$ for $k \geq 0$ and taking $\alpha_k = a_k/\sum_{i=0}^k a_i$ and $\lambda_k = a_{k+1}/\sum_{i=0}^{k+1} a_i$. We further comment in Section B.4.

Limitations Our analysis is convex. Although convex theory often remains empirically informative for deep learning, as observed by Defazio et al. [2024], Schaipp et al. [2025] and in our experiments, extending the guarantees to less restrictive assumptions is an important direction. Strictly speaking, our smooth analysis only uses star-convexity and cocoercivity along the trajectory.

A second limitation is the strong convexity requirement on the regularizer. For a regularizer h that is not strongly convex, one can instead use $h_\tau(x) := h(x) + \tau\psi(x)$, where ψ is 1-strongly convex with respect to the chosen norm and $\tau > 0$. Substituting h_τ for h and τ for μ preserves the same $O(n^{-1/2})$ dependence on n, but at the cost of a $1/\sqrt{\tau}$ factor in the constant. In particular, Scion and Muon are instances of the SODA *algorithm* through the indicator of a (time-varying) spectral norm ball, but the exact LMO-based updates are *not* covered by Theorem 4.6. What is covered are the smooth updates induced by strongly convex spectral regularizers, such as spectral clipping. The finite-step Newton-Schulz iteration used in practice is formally neither of the two, but, as we show in Section B.3, it is closely modeled by the strongly convex case.

5 Experiments

Throughout the experiments, Muon refers to the official implementation, which uses Adam for the first and last layer and Nesterov momentum for hidden layers (corresponding to optimism in SODA with $\bar{\alpha}_k = \alpha_k$). The Scion optimizer, on the other hand, disables the Nesterov momentum ($\bar{\alpha}_k = 0$) and uses Signum instead of Adam for the first and last layer. Experiments are conducted on nanoGPT on FineWeb100 (see Table 8 for full details). In the experiments, the token budget is expressed in units of Chinchilla [Hoffmann et al., 2022], where $1\times$ Chinchilla corresponds to $20 \times \#(\text{parameters})$.

Transfer across horizon Zero-shot hyperparameter transfer results, most notably μP, show that learning rates and related optimization hyperparameters can transfer reliably across width [Yang et al., 2021]. However, these transfer results are typically demonstrated at a fixed training horizon. This leaves open the more practically important question of how weight decay should transfer across horizon, since for longer runs the optimal weight decay typically changes.

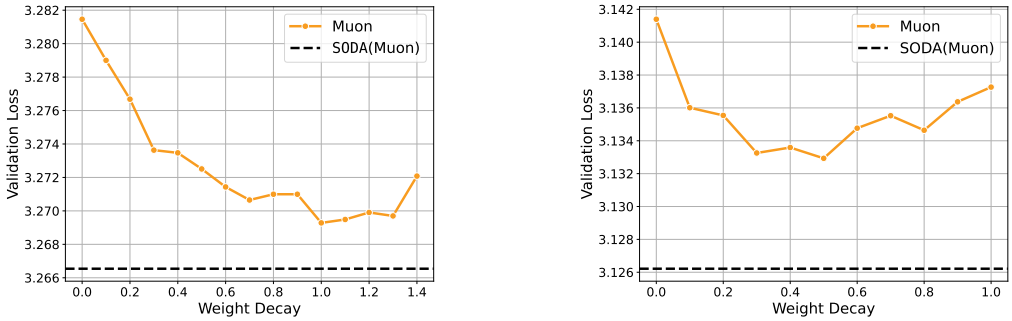

Figure 1: Muon with swept weight decay is outperformed by SODA(Muon), without any additional tuning, on 124M models trained for both $1\times$ Chinchilla steps (left) and $4\times$ Chinchilla steps (right).

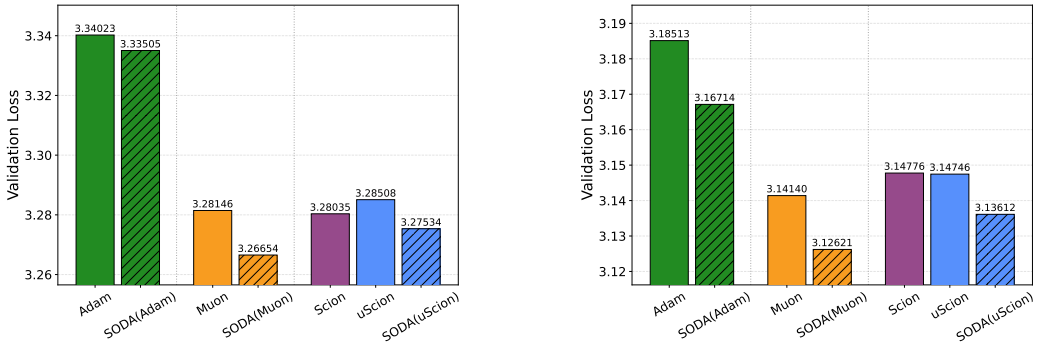

Figure 2: The SODA wrapper yields consistent improvement across various base optimizers without any additional tuning as illustrated on 124M model trained for both $1\times$ Chinchilla steps (left) and $4\times$ Chinchilla steps (right).

To isolate the effect of the horizon, we keep the model width fixed and vary only the training horizon. We find that the optimal weight decay *decreases* with the horizon for Muon (*c.f.* Figure 1), thus demonstrating that the $1/\texttt{model_size}$ choice made in Xiao [2024], Qiu et al. [2025], which is constant in the horizon, would be suboptimal outside the Chinchilla scaling rule where the model size and horizon is scaled proportionally [Hoffmann et al., 2022].

Furthermore, we find that the theoretically motivated $1/(k+2)$ weight decay scaling used by the SODA wrapper (Algorithm 1) consistently leads to further improvement without requiring any additional tuning. In Figure 1, SODA outperforms the base optimizer (Muon) even when the latter is given its best-tuned weight decay, showing that the gain is not simply due to a favorable retuning of that hyperparameter. At $1\times$ Chinchilla this holds even when the learning rate and weight decay of Muon are tuned jointly (Table 4), while at $4\times$ Chinchilla the learning rate and momentum are transferred from $1\times$ Chinchilla and only the weight decay is swept (see Section C.2 for the full tuning protocol). Taken together, these results suggest that SODA provides a principled mechanism for transferring weight decay across horizon without the need for tuning the weight decay even of the smaller proxy model. The SODA wrapper relies on a non-zero center iterate z^0, which we further ablate the importance of in Figure 7.

SODA Wrapper The benefit of the SODA Wrapper (Algorithm 1) is not exclusive to Muon. We additionally apply the wrapper across Adam and Scion and observe consistent improvements across training horizons in Figure 2, notably without introducing any additional tuning. The improvement is robust to the hyperparameters of the base optimizer: when sweeping the learning rate and momentum of uScion, the SODA wrapper improves upon the base optimizer for every tested value without shifting the optimal value (Table 5), and the improvement persists under a cosine learning rate

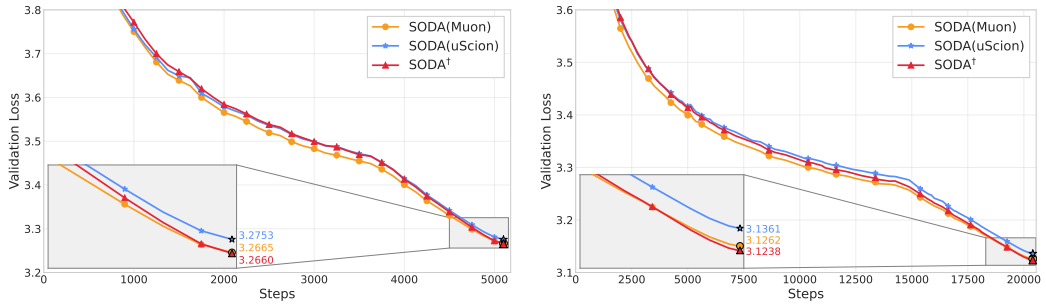

Figure 3: SODA with optimism (referred to as SODA†) is competitive with the best wrapped optimizer. In comparison with SODA(Muon), the configuration simplifies the method by replacing Adam with Lion and reusing the same hyperparameters for the momentum across all layers.

schedule. Beyond pre-training, the SODA wrapper also improves full fine-tuning of Llama-2-7B with Muon on MetaMathQA, as measured on GSM8K (Table 6).

Optimism and SODA† Considering the benefit of optimism (Muon) in the overtrained regime of Figure 2 we systematically investigate the impact of optimism in SODA, reported in Table 7 of Section C. We use the ℓ_∞-norm for the input and output layer and spectral norm for hidden layers following Pethick et al. [2025a]. SODA with optimism, which borrows the same optimized configuration as the Muon baseline ($\alpha_k = \bar{\alpha}_k = 0.05$), shows the best performance under both $1\times$ and $4\times$ Chinchilla. This SODA configuration notably removes the use of Adam in Muon and uses the same momentum hyperparameter across all layers in contrast, greatly simplifying tuning. We mark SODA with this specific optimistic setting as SODA†. Figure 3 shows validation loss compared with the best two settings found in Figure 2.

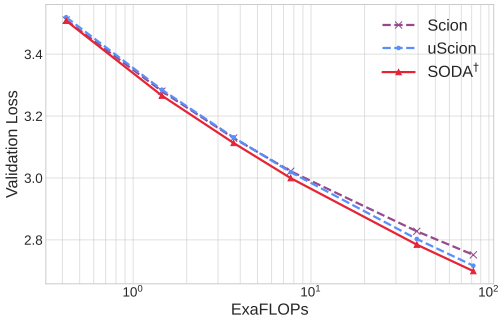

Figure 4: SODA is effective under $1\times$ Chinchilla scaling and the benefit increases with scale.

Transfer across horizon & width In Figure 4, we test SODA† on NanoGPT with between 64M and 1B parameters following Chinchilla-style scaling, where width and horizon grow proportionally. We choose (u)Scion as the base optimizer since its stepsize can be transferred along the width and horizon [Pethick et al., 2025a]. SODA† consistently outperforms Scion/uScion with different sizes of models and the gap becomes more apparent as the model grows larger. A similar conclusion holds for SODA(uScion) (*c.f.* Figure 6).

6 Conclusion

This work provides a new perspective on weight decay: beyond acting as a regularizer, it can be understood as a form of primal averaging. We show that carefully scheduling this averaging parameter (λ_k in SODA) yields acceleration in a precise theoretical sense and can lead to practical speedups without hyperparameter tuning.

Several directions open up. One is applying SODA to finetuning, where standard weight decay is often suboptimal. Since SODA regularizes with respect to the pretrained model z^0 rather than the origin, it may be better suited for this setting, as suggested by our preliminary fine-tuning results (Section C.4). Another promising direction is to develop practical accelerated instantiations of SODA.

7 Acknowledgments

This work was funded by the Swiss National Science Foundation (SNSF) under grant number 2000-1-240094. This work was supported by the Swiss AI Initiative (2025 Fellowship Program). This work was supported with project ID #37 as part of the Swiss AI Initiative, through a grant from the ETH Domain and computational resources provided by the Swiss National Supercomputing Centre (CSCS) under the Alps infrastructure. This research was funded in part by the Swiss National Science Foundation (SNSF) under grant number P500-2_242086.

References

Noah Amsel, David Persson, Christopher Musco, and Robert M Gower. The polar express: Optimal matrix sign methods and their application to the muon algorithm. *arXiv preprint arXiv:2505.16932*, 2025.

H Bauschke and Yves Lucet. What is a Fenchel conjugate. *Notices of the AMS*, 59(1):44–46, 2012.

Shane Bergsma, Nolan Dey, Gurpreet Gosal, Gavia Gray, Daria Soboleva, and Joel Hestness. Power lines: Scaling laws for weight decay and batch size in LLM pre-training. *arXiv preprint arXiv:2505.13738*, 2025.

Jeremy Bernstein and Laker Newhouse. Old optimizer, new norm: An anthology. *arXiv:2409.20325*, 2024.

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

David Carlson, Volkan Cevher, and Lawrence Carin. Stochastic spectral descent for restricted boltzmann machines. In *Artificial Intelligence and Statistics*, 2015a.

David Carlson, Ya-Ping Hsieh, Edo Collins, Lawrence Carin, and Volkan Cevher. Stochastic spectral descent for discrete graphical models. *IEEE Journal of Selected Topics in Signal Processing*, 2016.

David E Carlson, Edo Collins, Ya-Ping Hsieh, Lawrence Carin, and Volkan Cevher. Preconditioned spectral descent for deep learning. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, pages 2971–2979, 2015b.

Lizhang Chen, Bo Liu, Kaizhao Liang, and Qiang Liu. Lion secretly solves a constrained optimization: As lyapunov predicts. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=e4xS9ZarDr.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V Le. Symbolic discovery of optimization algorithms. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=ne6zeqLFCZ.

Kenneth L. Clarkson. Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm. *ACM Trans. Algorithms*, 2010.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Michael Crawshaw, Chirag Modi, Mingrui Liu, and Robert M Gower. An exploration of non-euclidean gradient descent: Muon and its many variants. *arXiv preprint arXiv:2510.09827*, 2025.

Ashok Cutkosky. Anytime online-to-batch, optimism and acceleration. In *International conference on machine learning*, pages 1446–1454. PMLR, 2019.

Aaron Defazio, Xingyu Yang, Harsh Mehta, Konstantin Mishchenko, Ahmed Khaled, and Ashok Cutkosky. The road less scheduled. In *Advances in Neural Information Processing Systems*, volume 37, pages 9974–10007, 2024. doi: 10.52202/079017-0320. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/136b9a13861308c8948cd308ccd02658-Paper-Conference.pdf.

Aaron Defazio, Konstantin Mishchenko, Parameswaran Raman, Hao-Jun Michael Shi, and Lin Xiao. Smoothing DiLoCo with primal averaging for faster training of LLMs. *arXiv preprint arXiv:2512.17131*, 2025.

Arthur Douillard, Qixuan Feng, Andrei A Rusu, Rachita Chhaparia, Yani Donchev, Adhiguna Kuncoro, Marc'Aurelio Ranzato, Arthur Szlam, and Jiajun Shen. Diloco: Distributed low-communication training of language models. *arXiv preprint arXiv:2311.08105*, 2023.

Timothy Dozat. Incorporating Nesterov momentum into Adam. 2016.

Dmitrii Feoktistov, Timofey Belinsky, Andrey Veprikov, Amir Zainullin, and Aleksandr Beznosikov. Softsign: Smooth sign in your optimizer for better parameter heterogeneity handling. *arXiv preprint arXiv:2605.31371*, 2026.

Damien Ferbach, Courtney Paquette, Gauthier Gidel, Katie Everett, and Elliot Paquette. Logarithmic-time schedules for scaling language models with momentum. *arXiv preprint arXiv:2602.05298*, 2026.

Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 1956.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, 2018.

Stephen Hanson and Lorien Pratt. Comparing biases for minimal network construction with back-propagation. *Advances in neural information processing systems*, 1, 1988.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. In *Advances in Neural Information Processing Systems*, 2022.

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

Martin Jaggi. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *International Conference on Machine Learning*, 2013.

Samy Jelassi and Aaron Defazio. Dual averaging is surprisingly effective for deep learning optimization. *arXiv preprint arXiv:2010.10502*, 2020.

Keller Jordan, Jeremy Bernstein, Brendan Rappazzo, @fernbear.bsky.social, Boza Vlado, You Jiacheng, Franz Cesista, Braden Koszarsky, and @Grad62304977. modded-nanogpt: Speedrunning the nanogpt baseline, 2024a. URL https://github.com/KellerJordan/modded-nanogpt.

Keller Jordan, Yuchen Jin, Vlado Boza, You Jiacheng, Franz Cecista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024b.

Pooria Joulani, Anant Raj, Andras Gyorgy, and Csaba Szepesvári. A simpler approach to accelerated optimization: iterative averaging meets optimism. In *International conference on machine learning*, pages 4984–4993. PMLR, 2020.

Dominik Kallusky, Vinay Rao, Vishal Nandavanam, and Hao-Jun Michael Shi. SNOO: Step-k Nesterov outer optimizer-the surprising effectiveness of Nesterov momentum applied to pseudo-gradients. *arXiv preprint arXiv:2510.15830*, 2025.

Ali Kavis, Kfir Y Levy, Francis Bach, and Volkan Cevher. Unixgrad: A universal, adaptive algorithm with optimal guarantees for constrained optimization. *Advances In Neural Information Processing Systems 32 (Nips 2019)*, 32(CONF), 2019.

DP Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2014.

Frederik Kunstner, Jacques Chen, Jonathan Wilder Lavington, and Mark Schmidt. Noise is not the main factor behind the gap between sgd and adam on transformers, but sign descent might be. *arXiv preprint arXiv:2304.13960*, 2023.

Guanghui Lan. An optimal method for stochastic composite optimization. *Mathematical Programming*, 133(1):365–397, 2012.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Ilya Loshchilov, Cheng-Ping Hsieh, Simeng Sun, and Boris Ginsburg. nGPT: Normalized transformer with representation learning on the hypersphere. *arXiv preprint arXiv:2410.01131*, 2024.

Microsoft AI. Frontier pre-training at Microsoft AI. https://microsoft.ai/news/frontier-pre-training-at-microsoft-ai/, 2026. Blog post, October 15, 2026.

Mehryar Mohri and Scott Yang. Accelerating online convex optimization via adaptive prediction. In *Artificial Intelligence and Statistics*, pages 848–856. PMLR, 2016.

Aryan Mokhtari, Hamed Hassani, and Amin Karbasi. Stochastic conditional gradient methods: From convex minimization to submodular maximization. *Journal of Machine Learning Research*, 2020.

Depen Morwani, Nikhil Vyas, Hanlin Zhang, and Sham Kakade. Connections between schedule-free optimizers, AdEMAMix, and accelerated sgd variants. *arXiv preprint arXiv:2502.02431*, 2025.

Yu Nesterov. Smooth minimization of non-smooth functions. *Mathematical programming*, 103: 127–152, 2005.

Yu Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

Yu Nesterov and Vladimir Shikhman. Quasi-monotone subgradient methods for nonsmooth convex minimization. *Journal of Optimization Theory and Applications*, 165(3):917–940, 2015.

Yurii Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical programming*, 120(1):221–259, 2009.

Konstantinos Oikonomidis, Jan Quan, Kimon Antonakopoulos, Antonio Silveti-Falls, Volkan Cevher, and Panagiotis Patrinos. Constrained stochastic spectral preconditioning converges for nonconvex objectives. *arXiv preprint arXiv:2605.11850*, 2026.

Francesco Orabona. A modern introduction to online learning. *CoRR*, abs/1912.13213, 2019. URL http://arxiv.org/abs/1912.13213.

Matteo Pagliardini, Pierre Ablin, and David Grangier. The AdEMAMix optimizer: Better, faster, older. *arXiv preprint arXiv:2409.03137*, 2024.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. *ICML (3)*, 28:1310–1318, 2013.

Thomas Pethick, Wanyun Xie, Kimon Antonakopoulos, Zhenyu Zhu, Antonio Silveti-Falls, and Volkan Cevher. Training deep learning models with norm-constrained LMOs. In *International Conference on Machine Learning*, 2025a.

Thomas Pethick, Wanyun Xie, Mete Erdogan, Kimon Antonakopoulos, Tony Silveti-Falls, and Volkan Cevher. Generalized gradient norm clipping & non-euclidean (L_0, L_1)-smoothness. *arXiv preprint arXiv:2506.01913*, 2025b.

Shikai Qiu, Zixi Chen, Hoang Phan, Qi Lei, and Andrew Gordon Wilson. Hyperparameter transfer enables consistent gains of matrix-preconditioned optimizers across scales. *arXiv preprint arXiv:2512.05620*, 2025.

Xingyu Qu, Peigeng Huang, and Samuel Horvath. Can Muon fine-tune Adam-pretrained models? *arXiv preprint arXiv:2605.10468*, 2026.

Alexander Rakhlin and Karthik Sridharan. Online learning with predictable sequences. In *Conference on Learning Theory*, pages 993–1019. PMLR, 2013.

Liliang Ren, Yang Liu, Yelong Shen, and Weizhu Chen. Rethinking language model scaling under transferable hypersphere optimization. *arXiv preprint arXiv:2603.28743*, 2026.

Artem Riabinin, Egor Shulgin, Kaja Gruntkowska, and Peter Richtárik. Gluon: Making Muon & Scion great again! (Bridging theory and practice of LMO-based optimizers for LLMs). *arXiv preprint arXiv:2505.13416*, 2025.

Meyer Scetbon, Chao Ma, Wenbo Gong, and Edward Meeds. Gradient multi-normalization for stateless and scalable LLM training. *arXiv preprint arXiv:2502.06742*, 2025.

Fabian Schaipp, Alexander Hägele, Adrien Taylor, Umut Simsekli, and Francis Bach. The surprising agreement between convex optimization theory and learning-rate scheduling for large model training. *arXiv preprint arXiv:2501.18965*, 2025.

M. Schmidt, N. Le Roux, and F. Bach. Convergence rates of inexact proximal-gradient methods for convex optimization. In *Adv. Neur. Inf. Proc. Sys. (NIPS)*, 2011.

Yuki Takezawa, Anastasia Koloskova, Xiaowen Jiang, and Sebastian U. Stich. FedMuon: Federated learning with bias-corrected LMO-based optimization. *arXiv preprint arXiv:2509.26337*, 2025.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Paul Tseng. On accelerated proximal gradient methods for convex-concave optimization. *submitted to SIAM Journal on Optimization*, 2(3), 2008.

Xi Wang and Laurence Aitchison. How to set AdamW's weight decay as you scale model and dataset size. *arXiv preprint arXiv:2405.13698*, 2024.

Kaiyue Wen, Xingyu Dang, Kaifeng Lyu, Tengyu Ma, and Percy Liang. Fantastic pretraining optimizers and where to find them II: Hyperball optimization. *arXiv preprint arXiv:2606.16899*, 2026.

Lechao Xiao. Rethinking conventional wisdom in machine learning: From generalization to scaling. *arXiv preprint arXiv:2409.15156*, 2024.

Shuo Xie and Zhiyuan Li. Implicit bias of AdamW: ℓ_∞ norm constrained optimization. *arXiv preprint arXiv:2404.04454*, 2024.

Tian Xie, Haoming Luo, Haoyu Tang, Yiwen Hu, Jason Klein Liu, Qingnan Ren, Yang Wang, Wayne Xin Zhao, Rui Yan, Bing Su, Chong Luo, and Baining Guo. Controlled LLM training on spectral sphere. *arXiv preprint arXiv:2601.08393*, 2026.

Zhenda Xie, Yixuan Wei, Huanqi Cao, Chenggang Zhao, Chengqi Deng, Jiashi Li, Damai Dai, Huazuo Gao, Jiang Chang, Liang Zhao, et al. mHC: Manifold-constrained hyper-connections. *arXiv preprint arXiv:2512.24880*, 2025.

Ge Yang, Edward Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tuning large neural networks via zero-shot hyperparameter transfer. In *Advances in Neural Information Processing Systems*, 2021.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. MetaMath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023.

Jingzhao Zhang, Tianxing He, Suvrit Sra, and Ali Jadbabaie. Why gradient clipping accelerates training: A theoretical justification for adaptivity. *arXiv:1905.11881*, 2019a.

Michael R. Zhang, James Lucas, Geoffrey Hinton, and Jimmy Ba. Lookahead Optimizer: K steps forward, 1 step back. *arXiv*, December 2019b.

Appendix

Table of Contents

A Preliminaries **17**

B Proofs for Section 4 (Analysis) **17**

C Experiments **23**

A Preliminaries

Lemma A.1. *Let $\|\cdot\|$ be a norm with dual norm $\|\cdot\|_*$, and let $\mathcal{D} = \{x : \|x\| \le 1\}$. Then*

$$s^\sharp = -\|s\|_* \operatorname{lmo}(s).$$

Proof. Let

$$u \in \arg\max_{\|v\|\le 1} \langle s, v\rangle,$$

so that $\langle s, u\rangle = \|s\|_*$. Then

$$\operatorname{lmo}(s) \in \arg\min_{\|x\|\le 1} \langle s, x\rangle = -u.$$

Also, writing $x = tv$ with $t \ge 0$ and $\|v\| = 1$,

$$s^\sharp \in \arg\max_{t\ge 0,\ \|v\|=1} \left\{ t\langle s, v\rangle - \tfrac{1}{2}t^2 \right\}.$$

For fixed t, the maximization over v is independent of t, hence uses the same u. Therefore

$$s^\sharp \in \arg\max_{t\ge 0} \left\{ t\|s\|_* - \tfrac{1}{2}t^2 \right\} = \|s\|_* u.$$

Combining the two identities gives

$$u = -\operatorname{lmo}(s),$$

and thus

$$s^\sharp = -\|s\|_* \operatorname{lmo}(s). \qquad \square$$

B Proofs for Section 4 (Analysis)

We first state the regret guarantee for ODA, and then employ it to analyze SODA. The ODA analysis is based on a non-adaptive version of Rakhlin and Sridharan [2013], which is classical (see e.g., [Orabona, 2019, Sec. 7.12]). The smooth online-to-batch Lemma below builds on the argument of Defazio et al. [2024] and extends it to non-Euclidean norms and slightly relaxes the noise condition.

Lemma B.1 (ODA regret)**.** *Consider the online-learning indexing of ODA, with the gradient term $\nabla f(z^k, \xi_k)$ replaced by g^k. Choose positive weights $a_0, \dots, a_{n-1}$ and write*

$$A_k := \textstyle\sum_{i=0}^{k} a_i, \qquad A_{-1} := 0.$$

We use the initial-hint convention $g^{-1} = 0$ and set $z^0 \in \partial h^*(0)$. For $k = 0, \dots, n-2$, choose

$$\alpha_k = \tfrac{a_k}{A_k}, \quad \bar\alpha_k = \tfrac{a_{k+1}}{A_{k+1}}, \quad \gamma_k = \eta\tfrac{A_k}{1-\bar\alpha_k}$$

for some $\eta > 0$. Let h be a proper closed μ-strongly convex regularizer on $\mathcal{X}$. Then, for every $x \in \mathcal{X}$,

$$\textstyle\sum_{k=0}^{n-1} a_k \langle g^k, z^k - x\rangle \le \frac{h(x) - \inf_{u\in\mathcal{X}} h(u)}{\eta} + \frac{\eta}{2\mu}\sum_{k=0}^{n-1} a_k^2 \left\|g^k - g^{k-1}\right\|_*^2. \tag{6}$$

Proof. By the choice of γ_k and $\bar\alpha_k$ we have, for $k = 0, \dots, n-2$,

$$\gamma_k(1-\bar\alpha_k) = \eta A_k, \quad \text{and} \quad \gamma_k\bar\alpha_k = \eta a_{k+1}$$

Combined with the choice of α_k, the ODA update can then be written as

$$z^k \in \partial h^*\left(-\eta\textstyle\sum_{i=0}^{k-1} a_i g^i - \eta a_k g^{k-1}\right), \qquad k = 0, \dots, n-1,$$

where the sum is empty when $k = 0$ and the case $k = 0$ is exactly $z^0 \in \partial h^*(0)$ because $g^{-1} = 0$. Set

$$\theta_k := -\eta\textstyle\sum_{i=0}^{k} a_i g^i, \qquad \theta_{-1} := 0, \qquad \hat\theta_k := \theta_{k-1} - \eta a_k g^{k-1}.$$

Thus $z^k \in \partial h^*(\hat\theta_k)$ and

$$\theta_k = \hat\theta_k - \eta a_k (g^k - g^{k-1}).$$

Since h is μ-strongly convex, h^* is $1/\mu$-smooth, hence

$$h^*(\theta_k) \le h^*(\hat{\theta}_k) - \eta a_k \langle g^k - g^{k-1}, z^k \rangle + \tfrac{\eta^2 a_k^2}{2\mu} \left\| g^k - g^{k-1} \right\|_*^2 .$$

By Fenchel–Young,

$$\langle \hat{\theta}_k, z^k \rangle = h(z^k) + h^*(\hat{\theta}_k), \qquad \langle \theta_k, x \rangle \le h(x) + h^*(\theta_k).$$

Therefore

$$\begin{aligned}
\eta a_k \langle g^k, z^k - x \rangle &= \eta a_k \langle g^k - g^{k-1}, z^k \rangle + \eta a_k \langle g^{k-1}, z^k \rangle - \eta a_k \langle g^k, x \rangle \\
&= \eta a_k \langle g^k - g^{k-1}, z^k \rangle + \langle \theta_{k-1} - \hat{\theta}_k, z^k \rangle + \langle \theta_k - \theta_{k-1}, x \rangle \\
&= \eta a_k \langle g^k - g^{k-1}, z^k \rangle + \langle \theta_{k-1}, z^k \rangle - \langle \hat{\theta}_k, z^k \rangle + \langle \theta_k - \theta_{k-1}, x \rangle \\
&= \eta a_k \langle g^k - g^{k-1}, z^k \rangle + \langle \theta_{k-1}, z^k \rangle - h(z^k) - h^*(\hat{\theta}_k) + \langle \theta_k - \theta_{k-1}, x \rangle \\
&\le h^*(\theta_{k-1}) - h^*(\theta_k) + \tfrac{\eta^2 a_k^2}{2\mu} \left\| g^k - g^{k-1} \right\|_*^2 + \langle \theta_k - \theta_{k-1}, x \rangle .
\end{aligned}$$

Summing over $k = 0, \dots, n-1$ yields

$$\eta \textstyle\sum_{k=0}^{n-1} a_k \langle g^k, z^k - x \rangle \le h^*(0) - h^*(\theta_{n-1}) + \langle \theta_{n-1}, x \rangle + \tfrac{\eta^2}{2\mu} \sum_{k=0}^{n-1} a_k^2 \left\| g^k - g^{k-1} \right\|_*^2 .$$

Finally, $h^*(0) = -\inf_{u \in \mathcal{X}} h(u)$ and $\langle \theta_{n-1}, x \rangle - h^*(\theta_{n-1}) \le h(x)$, so dividing by η proves the claim. □

B.1 Bounded gradients

Under bounded gradient we rely on the following online-to-batch conversion which holds for any primal extrapolation parameter $\bar{\lambda}_k \in [0, 1]$.

Lemma B.2 (SODA online-to-batch)**.** *Consider SODA with positive weights $a_0, \dots, a_{n-1}$ and*

$$\lambda_{k-1} = \tfrac{a_k}{A_k}, \qquad A_k := \textstyle\sum_{i=0}^{k} a_i, \qquad k = 1, \dots, n-1.$$

Suppose Assumptions 4.1 and 4.3 hold. Then, for every $x \in \mathcal{X}$,

$$A_{n-1} \, \mathbb{E}[f(x^{n-1}) - f(x)] \le \mathbb{E}\left[\textstyle\sum_{k=0}^{n-1} a_k \langle g^k, z^k - x \rangle \right]. \tag{7}$$

Proof. The result is obtained as an application of Defazio et al. [2024, Thm. 2] with

$$w_t = a_k, \quad z_{t+1} = z^k, \quad \beta_{t+1} = \bar{\lambda}_k$$

for which

$$x_{t+1} = x^k, \qquad y_{t+1} = (1 - \bar{\lambda}_{k-1}) x^k + \bar{\lambda}_{k-1} z^k = y^k.$$

□

Assumption B.3 (Bounded gradients)**.** *The gradients satisfies $\left\| g^k \right\|_* \le G$ for all $k \ge 0$.*

Corollary B.4 (Convergence under bounded gradients)**.** *Let $x^\star \in \arg\min_{x \in \mathcal{X}} f(x)$ and let $R_\star := h(x^\star) - \inf h$. Consider SODA with a fixed regularizer $h_k \equiv h$. For every $k = 0, \dots, n-1$, choose*

$$\alpha_k = \tfrac{1}{k+1}, \qquad \lambda_k = \tfrac{1}{k+2}, \qquad \bar{\alpha}_k = \tfrac{1}{k+2}, \qquad \bar{\lambda}_k \in [0, 1], \qquad \gamma_k = \eta(k+2), \qquad \eta = \tfrac{\sqrt{\mu R_\star}}{\sqrt{2}\, G \sqrt{n}}.$$

Let h be μ-strongly convex with respect to $\|\cdot\|$. Suppose Assumptions 4.1, 4.3 and B.3 hold. Then, for every $n \ge 1$,

$$\mathbb{E}[f(x^{n-1}) - f(x^\star)] \le \frac{2\sqrt{2}\, G \sqrt{R_\star}}{\sqrt{\mu n}}.$$

Proof. Choose the weights $a_k \equiv 1$, so that $A_k = k + 1$ for $k = 0, \dots, n-1$. Then

$$\alpha_k = \tfrac{1}{k+1} = \tfrac{a_k}{A_k}, \qquad \bar{\alpha}_k = \tfrac{1}{k+2} = \tfrac{a_{k+1}}{A_k + a_{k+1}}, \qquad \gamma_k (1 - \bar{\alpha}_k) = \eta A_k,$$

$$\gamma_k \bar{\alpha}_k = \eta (k+2) \tfrac{1}{k+2} = \eta = \eta a_{k+1},$$

for $k = 0, \dots, n-2$, while $\lambda_{k-1} = 1/(k+1) = a_k/A_k$ for $k = 1, \dots, n-1$. Therefore Theorem B.1 and Theorem B.2 yield

$$\mathbb{E}[f(x^{n-1}) - f(x^\star)] \le \tfrac{R_\star}{\eta n} + \tfrac{\eta}{2\mu n} \textstyle\sum_{k=0}^{n-1} \mathbb{E}\left[\left\|g^k - g^{k-1}\right\|_*^2\right].$$

Since $g^{-1} = 0$ by convention and $\left\|g^k\right\|_* \le G$, the sum is at most $4nG^2$. Hence

$$\mathbb{E}[f(x^{n-1}) - f(x^\star)] \le \tfrac{R_\star}{\eta n} + \tfrac{2\eta G^2}{\mu}.$$

Substituting the stated choice of η gives the claim. □

B.2 Gradient Lipschitz

The following refinement of Theorem B.2 is used to exploit smoothness of f. It follows the argument in Defazio et al. [2024, Thm. 5] directly, but allows for an non-Euclidean norm.

For differentiable f, we write the objective Bregman divergence as

$$D_f(u, v) := f(u) - f(v) - \langle \nabla f(v), u - v \rangle.$$

Lemma B.5 (SODA online-to-batch smooth refinement [Defazio et al., 2024, Thm. 5])**.** *Suppose the assumptions of Theorem B.2 hold, and additionally that f is L-smooth with respect to $\|\cdot\|$ (Assumption 4.2). If*

$$\bar{\lambda}_{k-1} \le \tfrac{a_k}{10 A_k} \qquad \text{for all } k = 1, \dots, n-1,$$

then, for every $x \in \mathcal{X}$, SODA satisfies

$$\begin{aligned} A_{n-1}\, \mathbb{E}[f(x^{n-1}) - f(x)] \le \mathbb{E}\left[\textstyle\sum_{k=0}^{n-1} a_k \langle g^k, z^k - x\rangle\right] \\ - \tfrac{1}{6L} \textstyle\sum_{k=0}^{n-1} A_{k-1} \mathbb{E}\left[\left\|\nabla f(y^k) - \nabla f(y^{k-1})\right\|_*^2\right], \end{aligned} \tag{8}$$

where $A_{-1} = 0$.

Proof. The choice $\lambda_{k-1} = a_k/A_k$ implies

$$x^k = \tfrac{1}{A_k} \textstyle\sum_{i=0}^{k} a_i z^i, \qquad k = 0, \dots, n-1.$$

For $k = 0, \dots, n-1$, following Defazio et al. [2024, Thm. 5], a direct expansion gives

$$\begin{aligned} A_k\big(f(x^k) - f(x)\big) - A_{k-1}\big(f(x^{k-1}) - f(x)\big) = a_k \langle \nabla f(y^k), z^k - x\rangle \\ - \tfrac{a_k}{\bar{\lambda}_{k-1}} D_f(y^k, x^k) - \tfrac{a_k(1-\bar{\lambda}_{k-1})}{\bar{\lambda}_{k-1}} D_f(x^k, y^k) \\ - A_{k-1} D_f(x^{k-1}, x^k) - a_k D_f(x, y^k), \end{aligned}$$

with the convention that the terms involving A_{-1} vanish, and that the two terms involving $\bar{\lambda}_{k-1}^{-1}$ vanish when $k = 0$ or $\bar{\lambda}_{k-1} = 0$, since then $y^k = x^k$. Summing over $k = 0, \dots, n-1$, taking expectations, and using $\mathbb{E}[g^k \mid \mathcal{F}_{k-1}] = \nabla f(y^k)$ gives

$$\begin{aligned} A_{n-1}\mathbb{E}[f(x^{n-1}) - f(x)] \le \mathbb{E}\left[\textstyle\sum_{k=0}^{n-1} a_k \langle g^k, z^k - x\rangle\right] \\ - \textstyle\sum_{k=0}^{n-1}\left(\tfrac{a_k}{\bar{\lambda}_{k-1}} \mathbb{E} D_f(y^k, x^k) + \tfrac{a_k(1-\bar{\lambda}_{k-1})}{\bar{\lambda}_{k-1}} \mathbb{E} D_f(x^k, y^k)\right) \\ - \textstyle\sum_{k=0}^{n-1}\left(A_{k-1} \mathbb{E} D_f(x^{k-1}, x^k) + a_k \mathbb{E} D_f(x, y^k)\right). \end{aligned} \tag{9}$$

For convex L-smooth f, co-coercivity gives $D_f(u, v) \ge \frac{1}{2L} \|\nabla f(u) - \nabla f(v)\|_*^2$, so

$$\tfrac{a_k}{\bar{\lambda}_{k-1}} D_f(y^k, x^k) + \tfrac{a_k(1-\bar{\lambda}_{k-1})}{\bar{\lambda}_{k-1}} D_f(x^k, y^k) \ge \tfrac{a_k(2-\bar{\lambda}_{k-1})}{2L\bar{\lambda}_{k-1}} \|\nabla f(y^k) - \nabla f(x^k)\|_*^2 \tag{10}$$

$$A_{k-1} D_f(x^{k-1}, x^k) + a_k D_f(x, y^k) \ge \tfrac{A_{k-1}}{2L} \|\nabla f(x^{k-1}) - \nabla f(x^k)\|_*^2 \tag{11}$$

Developing (11) with Young's inequality, $-\|a+b+c\|_*^2 \le -(1-2/\tau)\|a\|_*^2+(2\tau-1)(\|b\|_*^2+\|c\|_*^2)$ for $\tau=3$, we have

$$\begin{aligned}\tfrac{A_{k-1}}{2L}\|\nabla f(x^{k-1})-\nabla f(x^k)\|_*^2 \ge \tfrac{A_{k-1}}{6L}\|\nabla f(y^k)-\nabla f(y^{k-1})\|_*^2 \\ -\tfrac{5A_{k-1}}{2L}(\|\nabla f(y^k)-\nabla f(x^k)\|_*^2+\|\nabla f(y^{k-1})-\nabla f(x^{k-1})\|_*^2)\end{aligned} \tag{12}$$

Summing (10), (11) and using (12) and that $\bar\lambda_{k-1}\le a_k/(10A_k)$ we have,

$$\begin{aligned}&-\tfrac{a_k}{\bar\lambda_{k-1}}D_f(y^k,x^k)-\tfrac{a_k(1-\bar\lambda_{k-1})}{\bar\lambda_{k-1}}D_f(x^k,y^k)-A_{k-1}D_f(x^{k-1},x^k)-a_kD_f(x,y^k)\\ &\quad\le -\tfrac{A_{k-1}}{6L}\left\|\nabla f(y^k)-\nabla f(y^{k-1})\right\|_*^2-\tfrac{5A_k}{2L}\left\|\nabla f(y^k)-\nabla f(x^k)\right\|_*^2\\ &\qquad+\tfrac{5A_{k-1}}{2L}\left\|\nabla f(y^{k-1})-\nabla f(x^{k-1})\right\|_*^2.\end{aligned}$$

Summing over $k=0,\dots,n-1$ makes the final two terms telescope. Dropping the nonpositive telescoping remainder and combining with (9) gives (8). □

Remark B.6 (Tighter constant for $y^k=x^k$). We state Theorem B.5 for general $\bar\lambda_k$, similarly to Defazio et al. [2024]. The only inequality that is lossy is Young's inequality in (12), which is needed to relate the gradient variation at the averaged iterates x^k to the gradient variation at the query points y^k. For $\bar\lambda_k=0$, we have $y^k=x^k$, so the terms in (10) vanish and (11) directly gives the bound in terms of $\left\|\nabla f(y^k)-\nabla f(y^{k-1})\right\|_*^2$ without invoking (12). Consequently, the factor $\frac{1}{6L}$ in (8) improves to $\frac{1}{2L}$, and the stepsize condition $\eta\le\mu/(6\rho L)$ in Theorem 4.6 relaxes to $\eta\le\mu/(2\rho L)$. This is part of the motivation for using $\bar\lambda_k=0$ in the practical wrapper (Algorithm 1).

Corollary 4.6 (Convergence under L-smoothness). *Let $x^\star\in\arg\min_{x\in\mathcal{X}}f(x)$ and let $R_\star:=h(x^\star)-\inf h$. Consider SODA with a fixed regularizer $h_k\equiv h$. For every $k=0,\dots,n-1$, choose*

$$\alpha_k=\tfrac{1}{k+1},\qquad \bar\alpha_k=\lambda_k=\tfrac{1}{k+2},\qquad \bar\lambda_k\le\tfrac{\lambda_k}{10},\qquad \gamma_k=\eta(k+2),\qquad \eta=\min\left\{\tfrac{\mu}{6\rho L},\tfrac{\sqrt{\mu R_\star}}{\sigma\sqrt{n}}\right\}.$$

Suppose Assumptions 4.1 to 4.4 hold and that h is μ-strongly convex with respect to $\|\cdot\|$. Then, for every $n\ge1$,

$$\mathbb{E}[f(x^{n-1})-f(x^\star)]=O\left(\tfrac{(\rho+1)LR_\star}{\mu n}+\tfrac{\sigma\sqrt{R_\star}}{\sqrt{\mu n}}\right).$$

Proof. Choose the weights $a_k\equiv1$, so that $A_k=k+1$ for $k=0,\dots,n-1$. Then

$$\alpha_k=\tfrac{1}{k+1}=\tfrac{a_k}{A_k},\qquad \bar\alpha_k=\tfrac{1}{k+2}=\tfrac{a_{k+1}}{A_{k+1}},\qquad \gamma_k(1-\bar\alpha_k)=\eta A_k,$$

$$\gamma_k\bar\alpha_k=\eta(k+2)\tfrac{1}{k+2}=\eta=\eta a_{k+1},$$

for $k=0,\dots,n-2$, while $\lambda_{k-1}=1/(k+1)=a_k/A_k$ for $k=1,\dots,n-1$. Under L-smoothness, Theorem B.5 gives

$$\begin{aligned}n\,\mathbb{E}[f(x^{n-1})-f(x^\star)]\le\tfrac{R_\star}{\eta}+\tfrac{\eta}{2\mu}\textstyle\sum_{k=0}^{n-1}\mathbb{E}\left[\left\|g^k-g^{k-1}\right\|_*^2\right]\\ -\tfrac{1}{6L}\textstyle\sum_{k=0}^{n-1}k\,\mathbb{E}\left[\left\|\nabla f(y^k)-\nabla f(y^{k-1})\right\|_*^2\right].\end{aligned}$$

Using Assumption 4.4 with the boundary conventions $g^{-1}=0$ and $\nabla f(y^{-1})=\nabla f(x^\star)$, the startup term satisfies

$$\mathbb{E}\left[\left\|g^0-g^{-1}\right\|_*^2\right]\le\rho\,\mathbb{E}\left[\left\|\nabla f(y^0)-\nabla f(x^\star)\right\|_*^2\right]+\sigma^2.$$

For $k=1,\dots,n-1$, the same assumption gives

$$\mathbb{E}\left[\left\|g^k-g^{k-1}\right\|_*^2\right]\le\rho\,\mathbb{E}\left[\left\|\nabla f(y^k)-\nabla f(y^{k-1})\right\|_*^2\right]+\sigma^2.$$

Substituting this estimate into the previous bound gives

$$\begin{aligned}n\,\mathbb{E}[f(x^{n-1})-f(x^\star)]\le\tfrac{R_\star}{\eta}+\tfrac{\eta n\sigma^2}{2\mu}+\tfrac{\eta\rho}{2\mu}\mathbb{E}\left[\left\|\nabla f(y^0)-\nabla f(x^\star)\right\|_*^2\right]\\ +\textstyle\sum_{k=1}^{n-1}\left(\tfrac{\eta\rho}{2\mu}-\tfrac{k}{6L}\right)\mathbb{E}\left[\left\|\nabla f(y^k)-\nabla f(y^{k-1})\right\|_*^2\right].\end{aligned}$$

If $\eta \leq \mu/(6\rho L)$, then the coefficients in the final sum are nonpositive and the final sum can be dropped. Thus we obtain

$$\mathbb{E}[f(x^{n-1}) - f(x^\star)] \leq \tfrac{R_\star}{\eta n} + \tfrac{\eta\sigma^2}{2\mu} + \tfrac{\eta\rho}{2\mu n}\mathbb{E}\left[\left\|\nabla f(y^0) - \nabla f(x^\star)\right\|_*^2\right].$$

The displayed initialization bound gives $\mathbb{E}[\left\|\nabla f(y^0) - \nabla f(x^\star)\right\|_*^2] \leq 2L^2 R_\star/\mu$. Since $\eta \leq \mu/(6\rho L)$, the startup term is $O(LR_\star/(\mu n))$. The remaining two terms are optimized by the stated choice of η, yielding the claim. □

B.3 Scope of the theory for spectral methods

In this section we make precise to what extent the convergence guarantees cover the spectral methods used in our experiments. All spectral updates considered below act on the singular values. Specifically, for $G = U\Sigma V^\top$, they take the form $Um(\Sigma)V^\top$ for a scalar response map m applied elementwise to the singular values. We distinguish between three cases.

(i) Exact LMO-based updates Scion and Muon are recovered by SODA with h_k the indicator of a spectral norm ball, for which $\partial h_k^*(-u) = -\operatorname{msign}(u)$, i.e., $m(x) = 1$ for all $x > 0$. In the implementation of SODA (Equation (13)) the radius of the ball additionally grows with k. Since the indicator is not strongly convex, the exact updates are algorithmic instances of SODA, but are *not* covered by Theorems B.7 and 4.6. Exact LMO-based updates can instead be analyzed through stochastic Frank–Wolfe/conditional gradient arguments [Mokhtari et al., 2020, Pethick et al., 2025a]. Whether such an analysis can retain the benefit of optimism and the accelerated guarantee is an open question.

(ii) Finite-step Newton-Schulz Practical implementations of Muon and Scion do not compute msign exactly. Instead, they normalize the input by its Frobenius norm, so that all singular values lie in $(0, 1]$, and apply t iterations of an odd polynomial

$$\phi_{j+1}(x) = a\,\phi_j(x) + b\,\phi_j(x)^3 + c\,\phi_j(x)^5, \qquad \phi_0(x) = x.$$

Takezawa et al. [2025, Thm. 3] make the geometry of this finite-step implementation precise by showing that Muon with t Newton-Schulz iterations converges in terms of the gradient measured in a finite q-norm, where $q \to 1$ only as $t \to \infty$. In this sense, Muon with finite Newton-Schulz iterations behaves as a finite-p-norm method instead of the exact spectral LMO.

(iii) Strongly convex spectral regularizers Adding a strongly convex term to the indicator yields smooth response maps that *are* covered by our theory (with the $1/\sqrt{\tau}$ dependency discussed in Section 4). Two examples with parameter $\alpha > 0$ are

$$m_\alpha(x) = \min\{x/\alpha, 1\} \qquad \text{and} \qquad \psi_\alpha(x) = \frac{x}{(\alpha^4 + x^4)^{1/4}}.$$

The first is spectral clipping, induced by adding $\frac{\alpha}{2}\|\cdot\|_F^2$ to the indicator of the spectral norm ball (*cf.* Section 2). The second is induced by the strongly convex spectral reference function constructed in Oikonomidis et al. [2026, App. F.1]. A related regularized-LMO perspective is developed in Feoktistov et al. [2026], which obtains smooth sign-like updates from strongly convex regularizers.

The finite-step Newton-Schultz in (ii) is closely approximated by the strongly convex maps from (iii). Since $\phi_t'(0) = a^t$, choosing $\alpha_t = a^{-t}$ matches the slope of ϕ_t, m_{α_t} and ψ_{α_t} at the origin. Figure 5 compares the response maps for the polynomial $(a, b, c) = (2, -\frac{3}{2}, \frac{1}{2})$ with $t = 12$ used in Track 3 of modded-nanogpt [Jordan et al., 2024a], and for the quintic $(a, b, c) = (3.4445, -4.7750, 2.0315)$ with $t = 5$ used by Muon in our experiments. In both cases, both strongly convex maps capture the finite-step Newton-Schulz substantially more closely than the exact msign, which differs significantly for small singular values.

We emphasize that this does not make the practical implementation a formal instance of Theorem 4.6. Rather, it shows that the strongly convex case analyzed in this paper can closely model finite-step Newton-Schulz, while the exact nonsmooth LMO represents the limiting geometry.

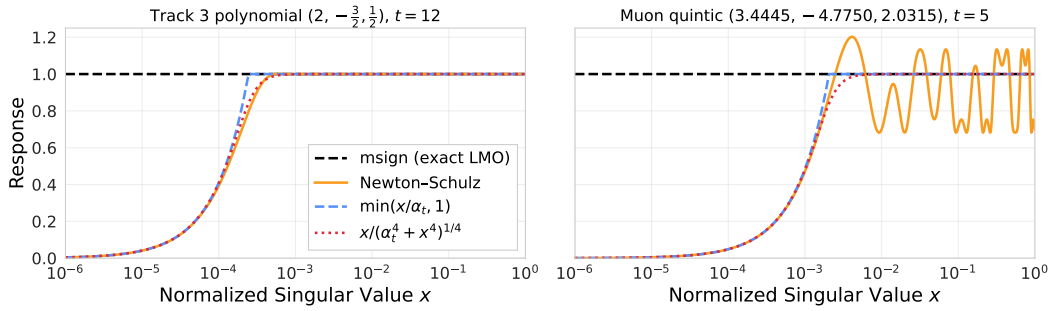

Figure 5: Response maps applied to the (Frobenius-normalized) singular values for the exact LMO (msign), finite-step Newton-Schulz, and the maps m_{α_t} and ψ_{α_t} induced by strongly convex spectral regularizers with slope-matched $\alpha_t = a^{-t}$. Left: the polynomial used in Track 3 of modded-nanogpt. Right: the Muon quintic used in our experiments. Both strongly convex maps track Newton-Schulz more closely than the exact msign map.

B.4 Accelerated rate

Corollary B.7 (Accelerated parameterization)**.** *Let $x^\star \in \arg\min_{x\in\mathcal{X}} f(x)$ and let $R_\star := h(x^\star) - \inf h$. Consider SODA with a fixed regularizer $h_k \equiv h$. For every $k = 0, \dots, n-1$, choose*

$$\alpha_k = \tfrac{2}{k+2}, \qquad \bar{\alpha}_k = \lambda_k = \tfrac{2}{k+3}, \qquad \bar{\lambda}_k \le \tfrac{\lambda_k}{10},$$
$$\gamma_k = \eta\tfrac{(k+2)(k+3)}{2}, \qquad \eta = \min\left\{\tfrac{\mu}{12\rho L}, \tfrac{\sqrt{\mu R_\star}}{\sigma n^{3/2}}\right\}.$$

Assume Assumptions 4.1 to 4.4, and that h is μ-strongly convex with respect to $\|\cdot\|$. Then, for every $n \ge 1$,

$$\mathbb{E}[f(x^{n-1}) - f(x^\star)] = O\left(\frac{(\rho+1)LR_\star}{\mu n^2} + \frac{\sigma\sqrt{R_\star}}{\sqrt{\mu n}}\right).$$

Remark B.8. Theorem B.7 suggests that optimism ($\bar{\alpha}_k$) and primal averaging (λ_k) should be scheduled in tandem to obtain acceleration. In the context of deep learning, this suggests that optimism, which is commonly referred to as Nesterov momentum, does not lead to acceleration on its own, but should be scheduled in combination with the weight decay (λ_k). We leave this as interesting future work.

Proof. Choose the weights $a_k = k+1$ for $k = 0, \dots, n-1$. Then

$$A_k = \textstyle\sum_{i=0}^k a_i = \tfrac{(k+1)(k+2)}{2}, \qquad \tfrac{a_k}{A_k} = \tfrac{2}{k+2}, \qquad \tfrac{a_{k+1}}{A_{k+1}} = \tfrac{2}{k+3}.$$

For $k = 0, \dots, n-2$, the displayed parameter choices therefore satisfy

$$\alpha_k = \tfrac{a_k}{A_k}, \quad \lambda_k = \tfrac{a_{k+1}}{A_{k+1}}, \quad \bar{\lambda}_k \le \tfrac{a_{k+1}}{10A_{k+1}}, \bar{\alpha}_k = \tfrac{a_{k+1}}{A_{k+1}}, \gamma_k = \eta\tfrac{A_k}{1-\bar{\alpha}_k},$$

Theorem B.1 and Theorem B.5 give

$$A_{n-1}\mathbb{E}[f(x^{n-1}) - f(x^\star)] \le \tfrac{R_\star}{\eta} + \tfrac{\eta}{2\mu}\textstyle\sum_{k=0}^{n-1}(k+1)^2\mathbb{E}\left[\left\|g^k - g^{k-1}\right\|_*^2\right]$$
$$- \tfrac{1}{6L}\textstyle\sum_{k=0}^{n-1} A_{k-1}\mathbb{E}\left[\left\|\nabla f(y^k) - \nabla f(y^{k-1})\right\|_*^2\right].$$

Using Assumption 4.4 with the boundary conventions $g^{-1} = 0$ and $\nabla f(y^{-1}) = \nabla f(x^\star)$, the startup term satisfies

$$\mathbb{E}\left[\left\|g^0 - g^{-1}\right\|_*^2\right] \le \rho\,\mathbb{E}\left[\left\|\nabla f(y^0) - \nabla f(x^\star)\right\|_*^2\right] + \sigma^2.$$

For $k = 1, \dots, n-1$, the same assumption controls the gradient differences, and we obtain

$$A_{n-1}\mathbb{E}[f(x^{n-1}) - f(x^\star)] \le \tfrac{R_\star}{\eta} + \tfrac{\eta\rho}{2\mu}\mathbb{E}\left[\left\|\nabla f(y^0) - \nabla f(x^\star)\right\|_*^2\right] + \tfrac{\eta\sigma^2}{2\mu}\textstyle\sum_{k=0}^{n-1}(k+1)^2$$
$$+ \textstyle\sum_{k=1}^{n-1}\left(\tfrac{\eta\rho}{2\mu}(k+1)^2 - \tfrac{A_{k-1}}{6L}\right)\mathbb{E}\left[\left\|\nabla f(y^k) - \nabla f(y^{k-1})\right\|_*^2\right],$$

Table 2: The linear minimization oracle (lmo) for various norm choices. Define $\mathrm{col}_j(A) := [A]_{\cdot,j}$ and $\mathrm{row}_i(A) := [A]_{i,\cdot}$.

Weight matrix norm	W_1 (1-hot encoded)	W_1 (image domain)	$(W_\ell)_{\ell\in[2,\dots,L-1]}$	W_L
Spectral (RMS $\to$ RMS)	$-\sqrt{d_{\text{out}}}UV^\top$	$-\sqrt{d_{\text{out}}/d_{\text{in}}}UV^\top$		
ColNorm (1 $\to$ RMS)	$\mathrm{col}_j(W_1) \mapsto -\sqrt{d_{\text{out}}}\frac{\mathrm{col}_j(W_1)}{\|\mathrm{col}_j(W_1)\|_2}$	$\mathrm{col}_j(W_\ell) \mapsto -\frac{\sqrt{d_{\text{out}}}}{d_{\text{in}}}\frac{\mathrm{col}_j(W_\ell)}{\|\mathrm{col}_j(W_\ell)\|_2}$		
RowNorm (RMS $\to \infty$)	$\mathrm{row}_i(W_1) \mapsto -\frac{\mathrm{row}_i(W_1)}{\|\mathrm{row}_i(W_1)\|_2}$	$\mathrm{row}_i(W_\ell) \mapsto -\frac{\mathrm{row}_i(W_\ell)}{\sqrt{d_{\text{in}}}\|\mathrm{row}_i(W_\ell)\|_2}$		
Sign (1 $\to \infty$)	$-\operatorname{sign}(W_1)$	$-\frac{1}{d_{\text{in}}}\operatorname{sign}(W_\ell)$		

where we used $A_{k-1} = \frac{k(k+1)}{2} \geq \frac{(k+1)^2}{4}$ for $k \geq 1$. By the stated choice of η, the coefficient of the final term is nonpositive, so it can be dropped. Since

$$A_{n-1} = \Theta(n^2), \qquad \textstyle\sum_{k=1}^{n-1}(k+1)^2 = O(n^3),$$

it follows that

$$\mathbb{E}[f(x^{n-1}) - f(x^\star)] = O\left(\frac{R_\star}{\eta n^2} + \frac{\eta\sigma^2 n}{\mu} + \frac{\eta\rho}{\mu n^2}\mathbb{E}\left[\left\|\nabla f(y^0) - \nabla f(x^\star)\right\|_*^2\right]\right).$$

Now set

$$\eta_1 := \tfrac{\mu}{12\rho L}, \qquad \eta_2 := \tfrac{\sqrt{\mu R_\star}}{\sigma n^{3/2}}, \qquad \eta = \min\{\eta_1, \eta_2\}.$$

If $\eta = \eta_1$, then

$$\tfrac{R_\star}{\eta n^2} = O\left(\tfrac{\rho L R_\star}{\mu n^2}\right), \qquad \tfrac{\eta\sigma^2 n}{\mu} = O\left(\tfrac{\sigma^2 n}{\rho L}\right) \leq \tfrac{\sigma\sqrt{R_\star}}{\sqrt{\mu n}},$$

because $\eta_1 \leq \eta_2$ implies $\sigma n^{3/2} = O(\rho L\sqrt{R_\star/\mu})$. The startup term is $\eta\rho\mathbb{E}[\|\nabla f(y^0) - \nabla f(x^\star)\|_*^2]/(\mu n^2) = O(LR_\star/(\mu n^2))$. If instead $\eta = \eta_2$, then

$$\tfrac{R_\star}{\eta n^2} = \tfrac{\eta\sigma^2 n}{\mu} = \tfrac{\sigma\sqrt{R_\star}}{\sqrt{\mu n}},$$

and the startup term is again $O(LR_\star/(\mu n^2))$ since $\eta \leq \eta_1$. Combining the two cases gives the claim. □

C Experiments

C.1 Implementation

Norm choice Consider a neural network parameterized by weight matrices $W_1, \dots, W_L$, where W_1 is the first layer and W_L is the last layer. Following Pethick et al. [2025a], we use the max-norm across layers

$$\|x\| = \max_l \tfrac{1}{\rho_l}\|W_l\|_{\alpha_l\to\beta_l}$$

where ρ_l is the layerwise radii and $\|\cdot\|_{\alpha\to\beta} := \sup_{x\neq 0}\|Ax\|_\beta/\|x\|_\alpha$ is an operator norm. Under the max-norm the lmo decomposes layerwise. Different layerwise operator norms can be found in Table 2. If not otherwise noted, we use the operator norm choice (Sign $\to$ Spectral $\to$ Sign) for both Scion and SODA, which refers to using Sign for the first layer, Spectral for intermediary layers and Sign for the last layer.

SODA Wrapper For ease of use, the provided implementation of the SODA Wrapper (Algorithm 1) can be used directly as a wrapper around any existing PyTorch optimizer. However, this ease of use comes at the cost of requiring additional memory since we need to recover the update delta $x^{k+1} - x^k$, which requires storing the previous iterate x^k. For production, we therefore recommend one of three options:

(i) modify the base optimizer to directly return the update delta $x^{k+1} - x^k$,

(ii) directly implement the simplified wrapper expression (4) inside the base optimizer,

(iii) or use the SODA algorithm, which involves the dual momentum and the mirror map.

For Adam, which uses stepsize warmup, the SODA wrapper is only applied after the warmup phase, following the protocol of Schedule-Free [Defazio et al., 2024], which starts the iteration counter k from the beginning of training.

Layer choice For all SODA Wrapper experiments, we wrap only the hidden layers, keeping the first/last layers without weight decay, following the choice of modded-nanogpt, which only sets weight decay for hidden layers. We further test this layer choice in Table 3.

For SODA(Muon), wrapping only hidden layers is consistently better than wrapping all layers under both $1\times$ and $4\times$ Chinchilla. On the other hand, when using uScion with the norm choice (Sign $\to$ Spectral $\to$ Sign) as the base optimizer, wrapping all layers is competitive with wrapping only hidden layers. It appears that the $1/(k+2)$ weight decay introduced by the SODA Wrapper improves performance in hidden layers, which benefit from weight decay, while not harming the first and last layers, which are typically more sensitive to it. This suggests that also the non-wrapped version SODA with (Sign $\to$ Spectral $\to$ Sign) can safely be applied to all layers as done in all subsequent experiments.

SODA The implementation of SODA makes the choice $h_k(x) = \iota_{\mathcal{B}(z^0,\gamma_k)}(x)$ where $\mathcal{B}(z,\gamma) = \{x \in \mathcal{X} \mid \|x - z\| \leq \gamma\}$ which reduces the update to

$$\begin{aligned} m^{k+1} &= (1-\alpha_k)m^k + \alpha_k \nabla f(y^k, \xi_k) \\ \bar{m}^{k+1} &= (1-\bar{\alpha}_k)m^{k+1} + \bar{\alpha}_k \nabla f(y^k, \xi_k) \\ z^{k+1} &\in z^0 + \gamma_k \operatorname{lmo}_{\mathcal{B}(0,1)}(\bar{m}^{k+1}) \\ x^{k+1} &= (1-\lambda_k)x^k + \lambda_k z^{k+1} \\ y^{k+1} &= (1-\bar{\lambda}_k)x^{k+1} + \bar{\lambda}_k z^{k+1} \end{aligned} \tag{13}$$

with $\alpha_k, \bar{\alpha}_k, \bar{\lambda}_k \in [0,1]$, $\lambda_k = 1/(k+2)$ and $\gamma_k = \eta_k(k+2)$ similar to Algorithm 1. We initialize $m^0 = \nabla f(x^0, \xi_0)$ and $x^0 = y^0 = z^0$.

For simplicity, consider $\bar{\alpha}_k = 0$ and $\bar{\lambda}_k = 0$, which disable optimism and enables primal extrapolation. In the context of Frank-Wolfe based methods such as Scion [Pethick et al., 2025a], SODA then corresponds to i) centering the update around the initialization z^0 instead of the origin, and ii) setting the Frank-Wolfe stepsize $\lambda_k = 1/(k+2)$ while the radius $\gamma_k = \eta_k(k+2)$ is linearly increasing.

We initialize the momentum as $m^0 = \nabla f(x^0, \xi_0)$ following theory, which corresponds to using `torch.clone(p.grad)`. We observe that this performs slightly better than the typical $m^0 = 0$, i.e. `torch.zeros_like(p.grad)`, when it comes to SODA.

Setup We rely on the tuned baselines for AdamW and Muon from modded-nanogpt Jordan et al. [2024a] used in Pethick et al. [2025a]. For the baselines weight decay refers to setting $\mu_k = \mu\gamma_k$ in (5) for weight decay parameter $\mu \geq 0$ and (possibly scheduled) stepsize γ_k, as is the default in e.g., PyTorch. Training a 124M model on FineWeb100 for 5k steps (i.e., $1\times$ Chinchilla) takes roughly 20-30min on 4 H200 NVIDIA GPUs.

Optimism Table 7 explores the role of optimism showing results for different choices of $\alpha_k, \bar{\alpha}_k$ in SODA. For the remaining hyperparameters we use the same default choices as for the SODA Wrapper, namely $\lambda_k = 1/(k+2)$ (uniform averaging) and $\bar{\lambda}_k = 0$ (primal extrapolation).

Table 3: NanoGPT Validation loss comparing wrapping hidden layers vs. wrapping all layers.

Steps	Muon	SODA(hidden)	SODA(all)	Scion	uScion	SODA(hidden)	SODA(all)
5k	3.28146	3.26654	3.27493	3.28035	3.28508	3.27534	3.27582
20k	3.14140	3.12621	3.12933	3.14776	3.14746	3.13612	3.13196

C.2 Tuning protocol

Baselines We use tuned baselines of AdamW, Muon and Scion/uScion for the 124M model trained for $1\times$ Chinchilla steps. Specifically, we reproduce the baselines following Figure 1 of Pethick et al. [2025a], which sweeps the learning rate over powers of two from 2^{-16} to 2^{-6} (without weight decay), and use the resulting optimal learning rate for each method, with all other hyperparameters following Pethick et al. [2025a, Table 7] (see Table 8). We verify the selected learning rate and momentum on local grids (Table 5). For Scion, the selected layer-wise learning rate is also consistent with Riabinin et al. [2025, Figure 3].

Table 4: Validation loss of Muon on the 124M NanoGPT model at $1\times$ Chinchilla for a joint grid over learning rate and weight decay. The best configuration is shown in bold. SODA(Muon), using the learning rate 2^{-11} without any further tuning, achieves a validation loss of 3.26654.

lr	wd$=0$	wd$=0.2$	wd$=0.4$	wd$=0.6$	wd$=0.8$	wd$=1.0$	wd$=1.2$
2^{-13}	3.31141	3.30656	3.30657	3.30257	3.30196	3.30069	3.29715
2^{-12}	3.28474	3.28161	3.27841	3.27915	3.27398	3.27373	3.27205
2^{-11}	3.28146	3.27668	3.27347	3.27630	3.27099	**3.26928**	3.26991
2^{-10}	3.29808	3.29042	3.28817	3.28876	3.29221	3.29571	3.29828

Table 5: Validation loss of uScion and SODA(uScion) on the 124M NanoGPT model at $1\times$ Chinchilla when sweeping the learning rate (left, with $\alpha = 0.1$) and the momentum α (right, with lr$= 2^{-12}$). The best value in each row is shown in bold.

lr	2^{-13}	2^{-12}	2^{-11}	2^{-10}
uScion	3.30444	**3.28508**	3.29110	3.31301
SODA(uScion)	3.30171	**3.27534**	3.27825	3.30861
α	0.05	0.1	0.2	0.3
uScion	3.29343	**3.28508**	3.29558	3.32025
SODA(uScion)	3.28667	**3.27534**	3.28629	3.31361

Joint tuning of learning rate and weight decay Since the learning rate sweep above does not use weight decay, we additionally tune the learning rate and weight decay of Muon jointly on a two-dimensional grid at $1\times$ Chinchilla (Table 4). The learning rate 2^{-11} gives the lowest validation loss for every weight decay value, which supports the learning rate used in Figure 1. SODA(Muon) achieves a lower validation loss than the jointly tuned Muon baseline (3.26654 vs. 3.26928).

Horizon scaling For $4\times$ Chinchilla (Figures 1 and 2), we reuse the learning rate and momentum from $1\times$ Chinchilla and only sweep the weight decay. We focus on weight decay since it has a noticeable effect on performance, while prior evidence suggests that the optimal learning rate and momentum move less across horizons than what a power-of-two grid would capture (see, e.g., Islamov et al. [2026, Figure 5]). The comparison at $4\times$ Chinchilla should therefore be understood as being against a baseline with tuned weight decay and transferred learning rate and momentum.

Width and horizon scaling For the scaling experiments (Figure 4), we rely on the μP-like transfer of Scion [Pethick et al., 2025a], which scales the effective learning rate of the output layer as $1/\texttt{dim}$.

SODA wrapper No additional tuning is performed for the SODA wrapper: every wrapped run inherits the learning rate, momentum, schedule, warmup and geometry parameters of the corresponding baseline.

C.3 Sensitivity to hyperparameters of the base optimizer

To test the interaction of the SODA wrapper with the hyperparameters of the base optimizer, we sweep the learning rate and the momentum (averaging parameter α) of uScion on the 124M model at $1\times$ Chinchilla, while keeping all other hyperparameters fixed, and apply the SODA wrapper without any retuning (Table 5). We find that (i) the SODA wrapper improves upon the base optimizer for every tested value, and (ii) the best value in each grid does not change under the SODA wrapper. Thus, within these grids, wrapping does not shift the optimal learning rate or momentum.

We also repeat the comparison with a cosine learning rate schedule. We select the best maximum learning rate for the uScion baseline and apply the SODA wrapper without retuning it. With the cosine schedule, the validation loss improves from 3.30889 to 3.29846.

Table 6: GSM8K strict exact-match accuracy (%) after full fine-tuning of Llama-2-7B on MetaMathQA (mean $\pm$ standard deviation over 3 seeds).

Method	GSM8K
Muon	56.73 ± 0.44
SODA(Muon)	$\mathbf{57.62} \pm 0.59$

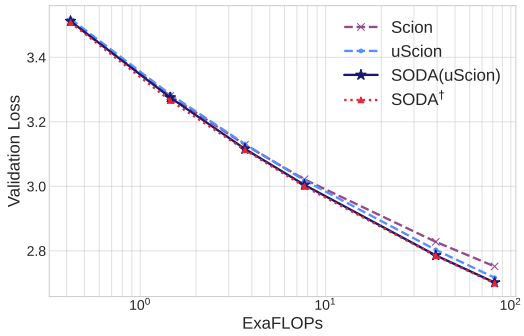

Figure 6: Both SODA† and SODA(uScion) are effective under $1\times$ Chinchilla scaling and the benefit increases with scale.

C.4 Fine-tuning

To test the SODA wrapper beyond pre-training, we consider full fine-tuning following Section 4.2 of Qu et al. [2026]: we fine-tune Llama-2-7B [Touvron et al., 2023] on MetaMathQA [Yu et al., 2023] and evaluate on GSM8K [Cobbe et al., 2021]. We sweep the learning rate of Muon while keeping all other hyperparameters as in Qu et al. [2026], and then apply the SODA wrapper on top of Muon with the same optimal learning rate. In this setting, the anchor z^0 of the SODA wrapper is the pretrained model. The results over 3 seeds are reported in Table 6. These preliminary results indicate that the improvements of the SODA wrapper extend to LLM fine-tuning, which complements our pre-training evaluation.

Table 7: NanoGPT validation loss comparing different values of the dual momentum parameters $\alpha_k, \bar{\alpha}_k$ for SODA.

	5k steps	20k steps
$\alpha_k = 0.05, \bar{\alpha}_k = 0$	3.27613	3.13250
$\alpha_k = 0.05, \bar{\alpha}_k = 0.05$	**3.26596**	**3.12376**
$\alpha_k = 0.05, \bar{\alpha}_k = 0.1$	3.27523	3.12543
$\alpha_k = 0.1, \bar{\alpha}_k = 0$	3.27582	3.13192
$\alpha_k = 0.1, \bar{\alpha}_k = 0.05$	3.27171	3.12981
$\alpha_k = 0.1, \bar{\alpha}_k = 0.1$	3.27903	3.13087

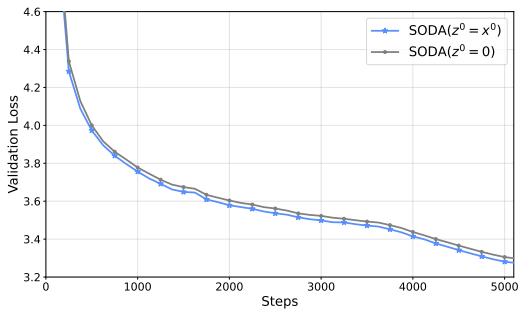

Figure 7: To illustrate the importance of the reference point z^0 being the initialization, we instantiate SODA with $z^0 = 0$, where we observe deterioration in performance. This is consistent with theory, since the rate in e.g., Theorem 4.6 depend on the distance to the anchor point, z^0, rather than the initial point where the gradient is evaluated, x^0.

Table 8: For NanoGPT we use the optimized hyperparameters for the 124M model from [Pethick et al., 2025a]. We use FineWeb100 to scale to the overtrained regime while avoiding data repetition. For ease of comparison Muon's momentum parameter β is written in terms of $\alpha = \bar{\alpha}$ (where $\beta = 1 - \alpha$). SODA† uses $\bar{\lambda} = 0$ and $\lambda_k = 1/(k+2)$ similar to the SODA Wrapper. Both SODA† and Scion use the same norm choice, Sign $\to$ Spectral $\to$ Sign (*c.f.* Table 2).

Hyperparameter	AdamW	Muon[1]	uScion	Scion	SODA†
Layers	12				
Head dim	128				
Activation function	ReLU^2		$2 \cdot \text{ReLU}^2$		
Vocabulary size	50304				
Dataset	FineWeb100				
Batch size	512				
Block size	1024				
Iterations n	5100 ($1\times$ Chinchilla)				
Warmdown m	28.5% of n				
Stepsize schedule	$\gamma_k = \begin{cases} \gamma & \text{if } k < n-m \\ \gamma \cdot \left(\frac{n-k}{m}\right) & \text{if } k \geq n-m \end{cases}$				
Warmup	5%	0			
Adam momentum β_1 / β_2	0.9 / 0.95	0.9 / 0.95	-		-
Averaging parameter α	-	0.05	0.1		0.05
Optimism parameter $\bar{\alpha}$	-	0.05	-		0.05
Radius ρ_1 / ρ_ℓ / ρ_L	-		- / 50 / 3000		
lr	2^{-9}	2^{-11}	2^{-12}	2^{-12}	2^{-12}

[1] Muon uses Adam for the first/last layer. The stepsize for the first/last layer is multiplied by 10.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The theory and practical results matches the claims.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Yes we mention limitation both explicitly after the introduction and in the theoretical section.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

Justification: All assumptions are clearly stated and proofs are provided in the appendix.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We have provided all details in the appendix, including hyperparameter choice and implementation details of our algorithm down to how the momentum buffer is initialized.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: We provide code for running our algorithm and will release a github repository upon acceptance.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: See Table 8.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: We observe consistent behaviour across horizon (Figure 1) and scale (Figure 4).

   Guidelines:

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

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: See appendix.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification:

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [N/A]

    Justification: The work is fundamental research and does not have any direct societal impact.

    Guidelines:

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

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: We do not release data or models.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We cite the experimental setup that we build on.

    Guidelines:

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

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

13. **New assets**

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

    Answer: [N/A]

    Justification:

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification:

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification:

    Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [N/A]

Justification:

Guidelines:

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