LION SECRETLY SOLVES CONSTRAINED OPTIMIZATION, AS LYAPUNOV PREDICTS

Lizhang Chen* **Bo Liu*** **Kaizhao Liang*** **Qiang Liu**
The University of Texas at Austin
{lzchen,bliu,kaizhaol,lqiang}@utexas.edu

ABSTRACT

Lion (Evolved Sign Momentum), a new optimizer discovered through program search, has shown promising results in training large AI models. It performs comparably or favorably to AdamW but with greater memory efficiency. As we can expect from the results of a random search program, Lion incorporates elements from several existing algorithms, including signed momentum, decoupled weight decay, Polak, and Nesterov momentum, but does not fit into any existing category of theoretically grounded optimizers. Thus, even though Lion appears to perform well as a general-purpose optimizer for a wide range of tasks, its theoretical basis remains uncertain. This lack of theoretical clarity limits opportunities to further enhance and expand Lion's efficacy.

This work aims to demystify Lion. Based on both continuous-time and discrete-time analysis, we demonstrate that Lion is a theoretically novel and principled approach for minimizing a general loss function $f(x)$ while enforcing a bound constraint $\|x\|_\infty \leq 1/\lambda$. Lion achieves this through the incorporation of decoupled weight decay, where λ represents the weight decay coefficient. Our analysis is made possible by the development of a new Lyapunov function for the Lion updates. It applies to a broader family of Lion-$\mathcal{K}$ algorithms, where the $\text{sign}(\cdot)$ operator in Lion is replaced by the subgradient of a convex function $\mathcal{K}$, leading to the solution of a general composite optimization problem of $\min_x f(x) + \mathcal{K}^*(x)$. Our findings provide valuable insights into the dynamics of Lion and pave the way for further improvements and extensions of Lion-related algorithms.

1 INTRODUCTION

Optimization serves as the cornerstone in training contemporary AI models. Given the immense computational demands associated with training large AI models, the design of an effective optimizer emerges as a paramount endeavor.

Traditionally, efficient optimizers are devised by machine learning experts based on theoretical insights [4, 15, 20, 11]. Adam [14] and its variant AdamW [20] remain the most widely employed methods in deep learning. Recently, however, a new optimization named Lion (Evolved Sign Momentum) [7] was discovered by an evolutionary search algorithm [32] applied to a symbolically represented program space [3]. Lion has been shown to achieve at least comparable performance to AdamW on a wide range of tasks while reducing memory cost and training time [7].

However, as the outcome of a stochastic search algorithm, Lion does not have an *a priori theoretical guarantee by design*. It is still uncertain whether Lion can be regarded as a reliable and legitimate general-purpose optimization algorithm, despite the reported positive results on a large, yet finite, set of tasks [7]. The lack of theoretical understanding also significantly restricts the potential for improving and extending Lion to obtain better new optimizers.

In this work, we demonstrate that Lion, along with a broader family of Lion-$\mathcal{K}$ algorithms, can be established as a theoretically novel and intriguing approach for solving optimization problems with convex regularization or constraints. This is surprising because Lion was discovered in a search

*Equal Contribution

space that includes arbitrary symbolic operations and was not designed with any theoretical guarantees. This discovery opens up promising opportunities for developing improved optimizers by leveraging the existing success of Lion.

Lion: Evolved Sign Momentum The update rule of Lion for minimizing a loss $f(x)$ on $\mathbb{R}^d$ is

$$\text{Lion:} \qquad \begin{aligned} m_{t+1} &= \beta_2 m_t - (1-\beta_2)\nabla f(x_t), \\ x_{t+1} &= x_t + \epsilon(\text{sign}(\beta_1 m_t - (1-\beta_1)\nabla f(x_t)) - \lambda x_t), \end{aligned} \tag{1}$$

where $m_t \in \mathbb{R}^d$ is the momentum, $\epsilon > 0$ is the learning rate, $\beta_1, \beta_2 \in [0,1]$ are two momentum related coefficients, and $\lambda \geq 0$ is a weight decay coefficient. A default value of $\beta_1 = 0.9$ and $\beta_2 = 0.99$ was suggested in Chen et al. [7], with which the Lion update rule can be written directly as

$$x_{t+1} \leftarrow (1-\epsilon\lambda)x_t - \epsilon \,\text{sign}\left((10+1)g_t + 0.99 g_{t-1} + 0.99^2 g_{t-2} + \cdots 0.99^k g_{t-k} + \cdots\right),$$

where $g_t = \nabla f(x_t)$. Here the update of x_t combines a weight decay term with coefficient $(1-\epsilon\lambda)$, and the sign of a weighted average of the trajectory gradients. Notably, the weight of the current gradient g_t is increased by $(\beta_2 - \beta_1)/((1-\beta_2)\beta_1) \approx 10$ times compared with typical exponential moving average of gradients as used in the classical Polyak momentum [30].

One can think of Lion as made by "splicing" the elements of many existing algorithms in Lion, which is exactly what an efficient search program can do when given a proper search space [29, 7, 3]. The update of the momentum m_t is common to the Polyak momentum-based algorithms and yields the exponential moving average part of the update. What sets it apart is the unique update of x_t, which uses the combination of three key elements:

i) **[Sign Reshaper]** The use of the $\text{sign}(\cdot)$ function for update, similar to signed gradient descent and signed momentum [5, 8], can be viewed as an extreme way of normalizing the magnitude of the coordinate-wise updates. It is closed related to normalized gradient [19, 25] and adaptive gradient methods such as Adam [14] and RMSprop [36]. Note that Adam can be viewed as signed momentum with an adaptive variance based step size [2], which might be the key factor explaining the gap between Adam and SGD [18].

ii) **[Gradient Enhancement]** When using $\beta_2 > \beta_1$, the importance of the current gradient g_t is increased compared to the exponential moving average in standard Polyak momentum update. It can be shown that Polyak momentum with this gradient enhancement results in Nesterov momentum, and leads to the well-known acceleration phenomenon [e.g., 35].

iii) **[Decoupled Weight Decay]** The weight decay term λx_t outside of the gradient and $\text{sign}(\cdot)$. Such idea of the *decoupled* weight decay is what make AdamW [21] significantly outperform the vanilla Adam in training large AI models.

As demonstrated by the empirical findings of Chen et al. [7] and subsequent research, the combination of these elements has been shown to make Lion perform well on a wide range of problems, including image classification, language models, and diffusion models [7].

However, it remains unclear whether the combination of these elements yield a theoretically valid and convergent general-purpose optimizer. Furthermore, the use of decoupled weight decay adds to the uncertainty regarding what optimization problem Lion aims to solve: due to its interaction with other parts of the algorithm, decoupled weight decay is always not equivalent to simply introducing ℓ_2 regularization [20].

"Lion King Meets Mr. Lyapunov" We propose and analyze a general family of Lion-$\mathcal{K}$ algorithms, in which we replace the $\text{sign}(\cdot)$ function in Lion with a subgradient $\nabla\mathcal{K}$ of a general convex function $\mathcal{K}\colon \mathbb{R}^d \to \mathbb{R}$:

$$\text{Lion-}\mathcal{K}\text{:} \qquad \begin{aligned} m_{t+1} &= \beta_2 m_t - (1-\beta_2)\nabla f(x_t), \\ x_{t+1} &= x_t + \epsilon(\nabla\mathcal{K}(\beta_1 m_t - (1-\beta_1)\nabla f(x_t)) - \lambda x_t). \end{aligned} \tag{2}$$

Lion is recovered when $\mathcal{K}(x) = \|x\|_1$ and $\nabla\mathcal{K}(x) = \text{sign}(x)$. Taking the continuous time limit of (2), we obtain the following ordinary differential equation (ODE):

$$\text{Lion-}\mathcal{K}\text{ (ODE):} \qquad \begin{aligned} \dot{m}_t &= -\alpha\nabla f(x_t) - \gamma m_t \\ \dot{x}_t &= \nabla\mathcal{K}(m_t - \varepsilon(\alpha\nabla f(x_t) + \gamma m_t)) - \lambda x_t, \end{aligned} \tag{3}$$

Polyak Momentum [30]	$\mathcal{K}(x) = \|x\|_2^2/2, \gamma\lambda = 0, \varepsilon = 0$
Nesterov Momentum [27]	$\mathcal{K}(x) = \|x\|_2^2/2, \gamma\lambda = 0$
Signed Momentum [5]	$\mathcal{K}(x) = \|x\|_1^2, \varepsilon = 0, \lambda = 0$
Hamiltonian Descent [22]	$\varepsilon = 0, \lambda = 0$
Hamiltonian Descent for Composite Objectives [22]	$\varepsilon = 0, \lambda > 0$
Dual Space Preconditioning [23], Mirror Descent [26]	$\varepsilon\gamma = 1, \lambda = 0$
Signed Gradient Descent [5]	$\mathcal{K}(x) = \|x\|_1, \varepsilon\gamma = 1, \lambda = 0$
Accelerated Mirror Descent [16]	$\gamma = 0, \varepsilon = 0, \lambda > 0$
Frank–Wolfe [10]	$\varepsilon\gamma = 1, \lambda > 0$

Table 1: Lion-$\mathcal{K}$ includes a large family algorithms as special cases. See Section 3.1

Eq. (2) is the Euler discretization of Eq. (3) with step size ϵ in the case of $\alpha = \gamma$, with $\beta_1 = 1 - \varepsilon\gamma$, and $\beta_2 = 1 - \epsilon\gamma$. Lion-$\mathcal{K}$ includes a broad set of algorithms as special cases, as shown in Table 1.

To avoid the complexities associated with regularity conditions, we can assume that $\mathcal{K}$ is continuously differentiable when discussing the ODE. But parallel results hold for the time discrete algorithm (2) for general non-differentiable convex functions $\mathcal{K}$.

The crest of this work is to show that, when $\varepsilon\gamma \leq 1$, Lion-$\mathcal{K}$ ODE solves the following optimization:

$$\min_{x\in\mathbb{R}^d} F(x) \coloneqq \alpha f(x) + \frac{\gamma}{\lambda}\mathcal{K}^*(\lambda x), \tag{4}$$

where $\mathcal{K}^*(x) \coloneqq \sup_z(x^\top z - \mathcal{K}(z))$ is the conjugate function of $\mathcal{K}$. Because we may have $\mathcal{K}^*(x) = +\infty$ for some x, solving (4) requires to enforce a constraint of $\lambda x \in \mathrm{dom}\mathcal{K}^*$, where $\mathrm{dom}\mathcal{K}^* \coloneqq \{x\colon \mathcal{K}^*(x) < +\infty\}$ is the effective domain of $\mathcal{K}^*$. In the case of Lion, we have $\mathcal{K}(x) = \|x\|_1$ and hence $\mathcal{K}^*(x) = \delta(\|x\|_\infty \leq 1)$, where δ the ∞-indicator function with $\delta(\texttt{True}) = 0$, $\delta(\texttt{False}) = +\infty$. Hence, Lion solves the following bound-constrained optimization problem:

$$\min_{x\in\mathbb{R}^d} f(x) \quad s.t. \quad \|x\|_\infty \leq 1/\lambda, \tag{5}$$

where the bound $1/\lambda$ is solely decided by the weight decay coefficient λ.

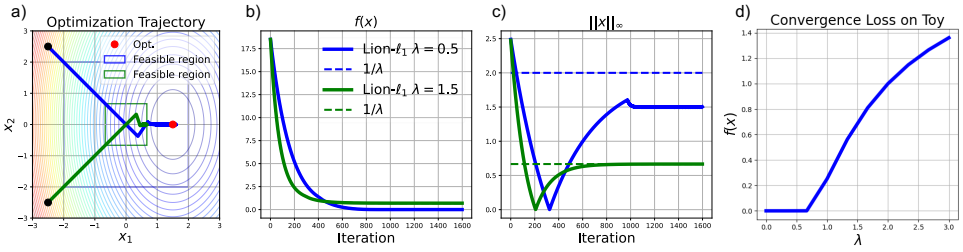

Figure 1: (a)-(c) Trajectories of Lion on 2D function $f(x) = (x_1 - 1.5)^2 + x_2^2$, with $\lambda = 1.5$ and $\lambda = 0.5$ ((a)-(c)). The boxes in a) represent the constraint set : blue box is for $\|x\|_\infty \leq 1/\lambda$ with $\lambda = 0.5$, green box is for $\lambda = 1.5$. (d) λ vs. the converged loss We can see that the converged loss starts to increase only when λ excel a threshold ($\lambda \geq 0.6$) to excluded the unconstrained minimum from the constrained set.

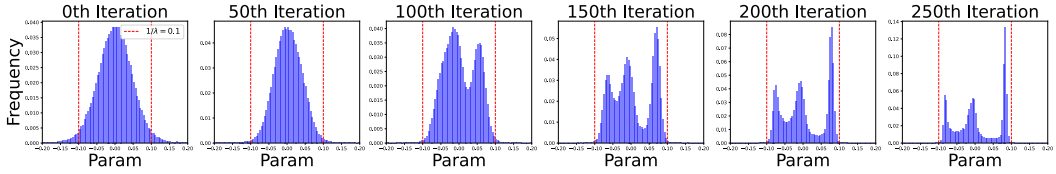

Figure 2: Histograms of the network parameters of ResNet-18 on CIFAR-10 trained by Lion with $\lambda = 10$. The constraint of $\|x\|_\infty \leq 1/\lambda$ (indicated by the red vertical lines) is satisfied within only $\sim$200 steps.

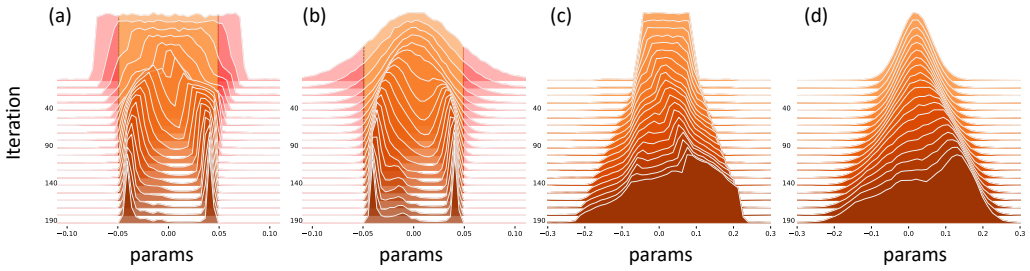

Figure 3: Evolution of histogram of parameter weights trained by Lion on ResNet-18 on CIFAR-10 [13, 17], with different λ and initialization methods. Frequency of network parameters in ResNet on the CIFAR-10 dataset across iterations. (a): Kaiming uniform initialization [12] and $\lambda = 20$. (b): Kaiming normal initialization [12] and $\lambda = 20$. (c): Kaiming uniform initialization [12] and $\lambda = 0$. (d): Kaiming normal initialization [12] and $\lambda = 0$. The weights are quickly confined into the bound $[-0.05, 0.05]$ with $\lambda = 20$, while keep growing with zero weight decay ($\lambda = 0$).

Our proof shows that the Lion-$\mathcal{K}$ dynamics consists of two phases:

1) **[Phase 1]** When $\lambda x \notin \mathrm{dom}\mathcal{K}^*$, it exponentially decays the distance from λx_t to the set $\mathrm{dom}\mathcal{K}^*$:

$$\mathrm{dist}(\lambda x_t, \mathrm{dom}\mathcal{K}^*) \leq \exp(-\lambda(t-s))\,\mathrm{dist}(\lambda x_s, \mathrm{dom}\mathcal{K}^*), \quad \forall s \leq t.$$

Hence, λx_t converges to $\mathrm{dom}\mathcal{K}^*$ rapidly and stays within $\mathrm{dom}\mathcal{K}^*$ once it arrived.

2) **[Phase 2]** After λx_t enters $\mathrm{dom}\mathcal{K}^*$, the dynamics minimizes the finite valued objective $F(x)$. This is proved by showing that the Lion-$\mathcal{K}$ dynamics minimizes the following Lyapunov function:

$$H(x,m) = \alpha f(x) + \frac{\gamma}{\lambda}\mathcal{K}^*(\lambda x) + \frac{1-\varepsilon\gamma}{1+\varepsilon\lambda}(\mathcal{K}^*(\lambda x) + \mathcal{K}(m) - \lambda m^\top x). \tag{6}$$

We show that, whenever $H(x_t, m_t)$ is finite, it is decreased monotonically (i.e., $\frac{\mathrm{d}}{\mathrm{d}t}H(x_t,m_t) \leq 0$) along trajectories of (3) until a local minimum of point of $H(x,m)$ is reached.

Furthermore, we have $F(x) = \min_m H(x,m)$, and hence minimizing $H(x,m)$ is equivalent to minimizing $F(x)$; this is because the minimum of the last term in (6) equals zero, $\min_m \mathcal{K}^*(\lambda x) + \mathcal{K}(m) - \lambda m^\top x = 0$, for any fixed x, by Fenchel-Young inequality.

The discovery of this Lyapunov function is a new and non-trivial mathematical result. But intuitively, one can see easily the connection of (3) and (4) by comparing their fixed points. Assume $\mathcal{K}$ and $\mathcal{K}^*$ are differentiable, then a fix point of (3) must implies a stationary point of (4):

$$\underbrace{\alpha\nabla f(x_t) + \gamma m_t = 0, \quad \nabla\mathcal{K}(m_t) = \lambda x_t}_{\text{fixed point of (3)}} \quad \Longrightarrow \quad \underbrace{\alpha\nabla f(x_t) + \gamma\nabla\mathcal{K}^*(\lambda x_t) = 0,}_{\text{stationary point of (4)}}$$

where we used $\nabla\mathcal{K}(\nabla\mathcal{K}^*(x)) = x$, and $\nabla_x\left(\frac{1}{\lambda}\mathcal{K}^*(\lambda x)\right) = \nabla\mathcal{K}^*(\lambda x)$.

Why Should Lion Decay Weight? From the analysis above, the role of weight decay λ in Lion is two-fold:

1) It alternates the solution if λ is large and the constraint $\|x\|_\infty \leq 1/\lambda$ is strong enough to exclude the unconstrained minimum x^*_{unc} of $f(x)$. This may improve the generalization and stability of the solution while sacrificing the training loss.

2) If λ is sufficiently small to include the unconstrained minimum x^*_{unc} in the constrained set, it does not alter the final solution. In this case, the main role of weight decay is to speed up the convergence because Phase 1 brings the solution into the constrained set with a linear rate. Hence, the ideal choice of λ is $\lambda = 1/\left\|x^*_{\text{unc}}\right\|_\infty$.

In Figure 4 we plot Lion's performance with different λ. The right plot confirms that larger λ results in faster convergence but might sacrifice the performance. The left plot shows that there exists an optimal λ (=0.56), beyond which the training loss starts to increase.

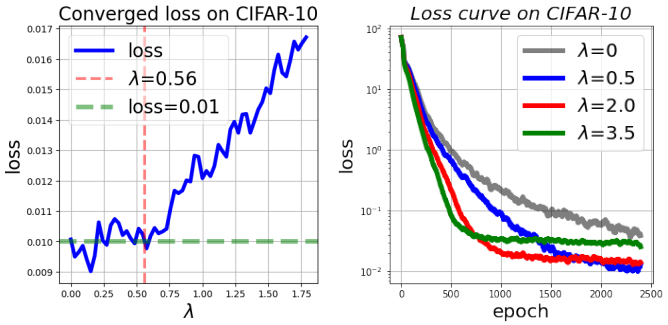

Figure 4: Analysis of weight decay on CIFAR-10 using Lion. a) The converged Loss vs. weight decay in Lion. We can see that the loss starts to increase only when λ excel a threshold, which is expected from the constrained optimization view. b) The loss curves vs. epochs with different weight decays. Larger weight decay λ yields faster convergence (due to stronger Phase 1), but may yield larger final loss when it is too large.

Line ID	$\mathcal{K}(x)$	$\nabla\mathcal{K}(x)$	$\min_x f(x)+\mathcal{K}^*(x)$
①	$\|x\|_1$	$\operatorname{sign}(x)$	$\min f(x)\ \ s.t.\ \ \|x\|_\infty \le 1$
②	$\|x\|_p$	$\frac{\operatorname{sign}(x)\|x\|^{p-1}}{\|x\|_p^{p-1}}$	$\min f(x)\ \ s.t.\ \ \|x\|_q \le 1$
③	$\sum_i \max(\|x_i\|-e,0)$	$\operatorname{sign}(x)\mathbb{I}(\|x\|>e)$	$\min f(x)+e\|x\|_1\ \ s.t.\ \ \|x\|_\infty \le 1$
④	$\sum_{i\le i^{cut}} \|x_{(i)}\|$	$\operatorname{sign}(x)\mathbb{I}(\|x\|>\|x_{(i^{cut})}\|)$	$\min f(x)\ \ s.t.\ \ \|x\|_1 \le i^{cut},\ \ \|x\|_\infty \le 1$
⑤	$\sum_i \operatorname{huber}_e(x_i)$	$\operatorname{clip}(x,-e,e)/e$	$\min f(x)+\frac{e}{2}\|x\|_2^2\ \ s.t.\ \ \|x\|_\infty < 1$

Table 2: Examples of $\mathcal{K}$ and $\nabla\mathcal{K}$, and the optimization problems they solved (we set $\gamma=\lambda=1$ for simplicity). We assume $x=[x_1,\dots,x_d]\in\mathbb{R}^d$ and $|x_{(1)}|\ge|x_{(2)}|\ge\cdots$ is a monotonic sorting of the elements of x, and i^{cut} is an integer in $\{1,\dots,d\}$. The Huber loss is $\operatorname{huber}_e(x_i)=\mathbb{I}(|x_i|\ge e)(|x_i|-\frac{e}{2})+\mathbb{I}(|x_i|<e)\frac{1}{2e}x_i^2$, $e>0$. See Appendix A for more examples.

Going Beyond Lion Different $\mathcal{K}$ yield optimization with different convex constraints and/or regularizations. For example, using the ℓ_p norm $\mathcal{K}(x)=\|x\|_p$ yields a constraint on the dual norm $\|x\|_q\le 1/\lambda$ where $1/p+1/q=1$ (Table 2, Line ②); zeroing out the coordinates with small magnitude corresponds to introducing an ℓ_1 regularization (Line ③) or ℓ_1 constraint (④), which is useful for sparse learning; replacing $\nabla\mathcal{K}(x)=\operatorname{sign}(x)$ with a continuous function would introduce an extra regularization term on the loss (e.g., ⑤). This work will focus on building the basic theoretical framework, and leave the vast opportunities of practical applications as future directions.

Outline The rest of the paper is organized as follows. Section 2 introduces preliminaries on convex functions. Section 3 analyzes the continuous-time Lion-$\mathcal{K}$ dynamics and discusses connections with existing algorithms. Section 4 presents the discrete-time analysis. Section 5 presents experiments that study and verify the behavior of using different $\mathcal{K}$s.

2 Preliminaries on Convex Functions

Assume $\mathcal{K}\colon\mathbb{R}^d\to\mathbb{R}$ is convex. A vector $u\in\mathbb{R}^d$ is said to be a subgradient of $\mathcal{K}$ at x, denoted as $u\in\partial\mathcal{K}(x)$, if

$$\mathcal{K}(y)-\mathcal{K}(x)\ge u^\top(y-x),\quad \forall y\in\mathbb{R}^d.$$

With an abuse of notation, we use $\nabla\mathcal{K}(x)$ to denote a subgradients of $\mathcal{K}$, that is, $\nabla\mathcal{K}(x)\in\partial\mathcal{K}(x)$. When $\mathcal{K}$ is differentiable at x, there is an unique subgradient $\nabla\mathcal{K}(x)$ which coincides with the regular derivative.

The conjugate function $\mathcal{K}^*$ of $\mathcal{K}$ is defined as

$$\mathcal{K}^*(x)=\sup_{z\in\mathbb{R}^d}(x^\top z-\mathcal{K}(z)).$$

Hence, by definition, we have the following Fenchel-Young inequality:

$$\mathcal{K}(x) + \mathcal{K}^*(y) \geq x^\top y, \quad \forall x, y. \tag{7}$$

The conjugate function $\mathcal{K}^*$ can take values in the extended real set $\overline{\mathbb{R}} = \mathbb{R} \cup \{\pm\infty\}$, and $\mathcal{K}^*$ is always closed and convex, even when $\mathcal{K}$ is not. Recall that a function f is said to be closed if for each $b \in \mathbb{R}$, its sublevel sets $\{x\colon f(x) \leq b\}$ is a closed set.

If $\mathcal{K}$ is closed and convex, we have $\mathcal{K}^{**} = \mathcal{K}$, and

$$y \in \partial\mathcal{K}(x) \qquad \Longleftrightarrow \qquad x \in \partial\mathcal{K}^*(y) \qquad \Longleftrightarrow \qquad \mathcal{K}(x) + \mathcal{K}^*(y) = x^\top y. \tag{8}$$

When $\mathcal{K}$ and $\mathcal{K}^*$ are differentiable, (8) suggests that $\nabla\mathcal{K}$ and $\nabla\mathcal{K}^*$ is a pair of inverse maps: $\nabla\mathcal{K}(\nabla\mathcal{K}^*(x)) = x$. Combining (7) and (8), we get $\min_m \mathcal{K}(m) + \mathcal{K}^*(x) - x^\top m = 0$, which yields $F(x) = \min_m H(x, m)$. We refer to Rockafellar [33] for a systematic introduction to convex functions.

A key property of any subgradient $\nabla\mathcal{K}$ and $\nabla\mathcal{K}^*$ is that they are monotonic maps, which plays a crucial rule in our results.

Lemma 2.1. *Assume $\mathcal{K}, \mathcal{K}^*$ is a closed convex conjugate pair and $\nabla\mathcal{K}$, $\nabla\mathcal{K}^*$ are their subgradients, we have*

$$(\nabla\mathcal{K}(x) - \nabla\mathcal{K}(y))^\top (x - y) \geq 0, \qquad (\nabla\mathcal{K}(x) - y)^\top (x - \nabla\mathcal{K}^*(y)) \geq 0. \tag{9}$$

See Appendix B.1 for the proof. These two inequalities are crucial because they allow us to identify vectors that have a non-negative inner product with a given direction to achieve monotonic descent in optimization.

Example 2.2. *In the case of Lion, we take $\mathcal{K}(x) = \|x\|_1$ with $\nabla\mathcal{K}(x) = \operatorname{sign}(x)$, and*

$$\mathcal{K}^*(y) = \begin{cases} 0 & \text{if } \|y\|_\infty \leq 1 \\ +\infty & \text{if } \|y\|_\infty > 1 \end{cases}, \qquad [\nabla\mathcal{K}^*(y)]_i = \begin{cases} 0 & \text{if } |y_i| \leq 1 \\ +\infty & y_i > 1 \\ -\infty & y_i < -1. \end{cases}$$

One can verify that the inequalities in (9) hold (even though the values on the left side can be $+\infty$). The Lyapunov function in (6) becomes

$$H(x, m) = \begin{cases} f(x) + \frac{1-\varepsilon\gamma}{1+\varepsilon\lambda}(\|m\|_1 - \lambda x^\top m) & \text{if } \|x\|_\infty \leq 1 \\ +\infty & \text{if } \|x\|_\infty > 1. \end{cases}$$

3 Main Result: Continuous-Time

We study the continuous-time Lion-$\mathcal{K}$ dynamics (3), and discuss its connection to existing algorithms listed in Table 1. We defer the detailed proofs to Appendix B.7, but outline a novel *implicit Hamiltonian + descent decomposition* that underpins the construction of the Lyapunov function $H(x, m)$.

Theorem 3.1. *Let (x_t, m_t) be a continuously differentiable trajectory of the Lion-$\mathcal{K}$ ODE (3), where $\mathcal{K}$ is differentiable convex with conjugate $\mathcal{K}^*$. Assume $\alpha, \gamma, \lambda, \varepsilon > 0$ and $\epsilon\gamma \leq 1$.*

*1) **[Phase 1]** Define $\operatorname{dist}(\lambda x_t, \operatorname{dom}\mathcal{K}^*) = \inf_{z \in \operatorname{dom}\mathcal{K}^*} \|z - \lambda x_t\|$ w.r.t. any norm $\|\cdot\|$. We have*

$$\operatorname{dist}(\lambda x_t, \operatorname{dom}\mathcal{K}^*) \leq \exp(\lambda(s - t)) \operatorname{dist}(\lambda x_s, \operatorname{dom}\mathcal{K}^*), \quad \forall 0 \leq s \leq t.$$

Hence, λx_t converges linearly to set $\operatorname{dom}\mathcal{K}^*$ and stays within $\operatorname{dom}\mathcal{K}^*$ once it enters it.

*2) **[Phase 2]** When $H(x, m)$ in (6) is finite and continuously differentiable, it is decreased monotonically along the trajectory:*

$$-\frac{\mathrm{d}}{\mathrm{d}t} H(x_t, m_t) = \Delta(x_t, m_t) \coloneqq \frac{\lambda + \gamma}{1 + \varepsilon\lambda} \Delta_1(x_t, \tilde{m}_t) + \frac{1 - \varepsilon\gamma}{1 + \varepsilon\lambda} \Delta_2(m_t, \tilde{m}_t) \geq 0,$$

where we define $\tilde{m}_t = m_t - \varepsilon(\alpha\nabla f(x_t) + \gamma m_t)$, and

$$\begin{aligned} \Delta_1(x_t, \tilde{m}_t) &= (\tilde{m}_t - \nabla\mathcal{K}^*(\lambda x_t))^\top (\nabla\mathcal{K}(\tilde{m}_t) - \lambda x_t) \geq 0, \\ \Delta_2(m_t, \tilde{m}_t) &= \frac{1}{\varepsilon} (\tilde{m}_t - m_t)^\top (\nabla\mathcal{K}(\tilde{m}_t) - \nabla\mathcal{K}(m_t)) \geq 0. \end{aligned} \tag{10}$$

*3) **[Stationarity]** Assume $\nabla\mathcal{K}^*$ is strictly monotonic. All the accumulation points of (x_t, m_t) as $t \to +\infty$ are stationary points of the objective function $F(x) = \alpha f(x) + \frac{\gamma}{\lambda}\mathcal{K}^*(\lambda x)$, and satisfy $\lambda x \in \mathrm{dom}\mathcal{K}^*$.*

$\Delta(x_t, m_t)$ can be viewed as an indication of the stationarity of the system. If $H(x_0, m_0)$ is finite and $H_b := \inf_{x,m} H(x, m) > -\infty$, we have $\frac{1}{T}\int_0^T \Delta(x_t, m_t)\mathrm{d}t \le \frac{H(x_0,m_0)-H_b}{T} \to 0$ when $T \to +\infty$.

Proof Sketch. See Appendix B.7 for the full proof. The original discovery of the Lyapunov function was made possible by starting from the inequalities in (10) as guaranteed by Lemma 2.1, and working backwards with some guesswork. The following is a simplified proof that highlights the essential mathematical structure that makes $H(x, m)$ Lyapunov. Define

$$\dot{x} = V_x(x, m) := \nabla\mathcal{K}(\tilde{m}) - \lambda x, \qquad \dot{m} = V_m(x, m) := -\alpha\nabla f(x) - \gamma m = \frac{\tilde{m} - m}{\varepsilon}$$

and related

$$\hat{V}_x(x, m) = \tilde{m} - \nabla\mathcal{K}^*(\lambda x), \qquad \hat{V}_m(x, m) = \nabla\mathcal{K}(\tilde{m}) - \nabla\mathcal{K}(m).$$

The $\hat{V}_x$ and $\hat{V}_m$ have two critical properties:

1) By Lemma 2.1, $\hat{V}_x$ and $\hat{V}_m$ have non-negative inner products with V_x, V_m, respectively:

$$\hat{V}_x(x, m)^\top V_x(x, m) \ge 0, \qquad \hat{V}_m(x, m)^\top V_m(x, m) \ge 0, \qquad \forall x, m.$$

2) By Lemma B.5 in Appendix B.7, the gradients of H can be decomposed as follows:

$$\begin{aligned}\nabla_x H(x, m) &= -\eta'\hat{V}_x - \eta V_m \\ \nabla_m H(x, m) &= -\eta\hat{V}_m + \eta V_x,\end{aligned} \qquad \textbf{(Implicit Hamiltonian + Descent)} \tag{11}$$

where $\eta = \frac{1-\varepsilon\gamma}{1+\varepsilon\lambda}$ and $\eta' = \frac{\gamma+\lambda}{1+\varepsilon\lambda}$. We call (11) an *"implicit" Hamiltonian + descent* decomposition, in connection with the Hamiltonian + descent decomposition we introduce in sequel.

Then we have,

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}H(x_t, m_t) &= \nabla_x H^\top V_x + \nabla_m H^\top V_m = (-\eta'\hat{V}_x - \eta V_m)^\top V_x + (-\eta\hat{V}_m + \eta V_x)^\top V_m \\ &= -(\eta'\hat{V}_x^\top V_x + \eta\hat{V}_m^\top V_m) \le 0.\end{aligned}$$

The key here is that the cross term $\eta V_x^\top V_m$ is canceled, leaving only the negative terms. The convergence property uses Lasselle's invariance principle; see Appendix B.7 for details. □

Hamiltonian + Descent Decomposition The decomposition structure (11) is a key characterization of Lion-$\mathcal{K}$ ODE. An interesting remark is that $H(x, m)$ is also Lyapunov if we have the following *Hamiltonian + descent* structure [22, 28] in which the roles of $[\nabla_x H, \nabla_m H]$ and $[V_x, V_m]$ in (11) are switched:

$$\begin{aligned}V_x &= -\hat{H}_x - \eta\nabla_m H \\ V_m &= -\hat{H}_m + \eta\nabla_x H,\end{aligned} \qquad \textbf{(Hamiltonian + Descent)} \tag{12}$$

where $\hat{H}_x, \hat{H}_m$ are two vector fields satisfying $\hat{H}_x^\top(\nabla_x H) \ge 0$ and $\hat{H}_m^\top(\nabla_m H) \ge 0$, then

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}H(x_t, m_t) &= \nabla_x H^\top V_x + \nabla_m H^\top V_m = \nabla_x H^\top(-\hat{H}_x - \eta\nabla_m H) + \nabla_m H^\top(-\hat{H}_m + \eta H_x) \\ &= -(\hat{H}_x^\top(\nabla_x H) + \hat{H}_m^\top(\nabla_m H)) \le 0.\end{aligned}$$

The structure in (12) can be intuitively viewed as a generalized damped Hamiltonian system with $H(x, m)$ as the total energy, where $[-\hat{H}_x, -\hat{H}_m]$ serves a damping force that monotonically decreases the total energy, and $[-\nabla_m H, \nabla_x H]$ is the Hamiltonian vector field which preserves the energy but introduces an inertia-like effect into system. One can easily verify (12) on the classical Polayk's momentum. The more general idea is explored in the Hamiltonian descent method of [22, 28], which considers systems of structure (12) for the separatiable Hamiltonian of form $H(x, m) = f(x) + \mathcal{K}(m)$ with $\hat{H}_x = 0$. In contrast, (11) do not seem to have a clear physical interpretation, yet provides a handy tool for understanding the general Lion-$\mathcal{K}$ dynamics. Some special cases of Lion-$\mathcal{K}$, such as when $\lambda = 0$ or $\varepsilon = 0$, can also be alternatively viewed from the Hamiltonian + descent structure as shown in Section 3.1.

3.1 Connection with Existing Algorithms

What makes Lion-$\mathcal{K}$ unique is the combination of the gradient enhancement ($\varepsilon > 0$), the decoupled weight decay ($\lambda > 0$), and the momentum damping ($\gamma > 0$), the use of reshaper function $\nabla\mathcal{K}(\cdot)$. We discuss the effects of these elements in connection to existing algorithms as shown in Table 1.

Lion-$\mathcal{K}$ Without Weight Decay When $\lambda = 0$ and $\nabla\mathcal{K}^*(0) = 0$, we have $\lim_{\lambda\to 0}\frac{1}{\lambda}\mathcal{K}^*(\lambda x) = \nabla\mathcal{K}(0)^\top x = 0$, and the Lyapunov function can be defined as

$$H(x, m) = \alpha f(x) + (1 - \varepsilon\gamma)\mathcal{K}(m),$$

for which we have

$$-\frac{\mathrm{d}}{\mathrm{d}t}H(x_t, m_t) = \gamma\nabla\mathcal{K}(\tilde{m}_t)\tilde{m}_t + \frac{(1-\varepsilon\gamma)}{\varepsilon}(\tilde{m}_t - m_t)^\top(\nabla\mathcal{K}(\tilde{m}_t) - \nabla\mathcal{K}(m_t)) \geq 0.$$

In this case, the algorithm solves $\min_x f(x)$, without the regularization term $\mathcal{K}^*(\lambda x)$.

Interestingly, in this case ($\lambda = 0$) and $1 - \varepsilon\gamma > 0$, there exists a second Lyapunov function:

$$\tilde{H}(x, m) = \alpha f(x) + \frac{1}{1-\varepsilon\gamma}\mathcal{K}((1-\varepsilon\gamma)m), \tag{13}$$

with which the Lion-$\mathcal{K}$ ODE ($\lambda = 0$) can be decomposed in the form of (12), as a sum of a Hamiltonian vector field and a descent direction:

$$\begin{bmatrix}\dot{x}_t\\ \dot{m}_t\end{bmatrix} = \underbrace{\begin{bmatrix}+\nabla_m\tilde{H}(x_t, m_t)\\ -\nabla_x\tilde{H}(x_t, m_t)\end{bmatrix}}_{\text{Hamiltonian}} - \underbrace{\begin{bmatrix}\nabla\mathcal{K}(\tilde{m}_t^0) - \nabla\mathcal{K}(\tilde{m}_t)\\ \gamma m_t\end{bmatrix}}_{\text{Descent}},$$

where $\tilde{m}_t^0 = (1-\varepsilon\gamma)m_t$ and hence $\tilde{m}_t^0 - \tilde{m}_t = \varepsilon\alpha\nabla f(x_t)$. If $m = 0$ is a minimum of $\mathcal{K}(m)$, one can show that the second component above is a descent direction of $\tilde{H}(x, m)$ in (13), with

$$-\frac{\mathrm{d}}{\mathrm{d}t}\tilde{H}(x_t, m_t) = \gamma\nabla\mathcal{K}(\tilde{m}_t^0)^\top m_t + \frac{1}{\varepsilon}(\tilde{m}_t^0 - \tilde{m}_t)^\top(\nabla\mathcal{K}(\tilde{m}_t^0) - \nabla\mathcal{K}(\tilde{m}_t)) \geq 0,$$

See Appendix B.6 for details.

Lion-$\mathcal{K}$ Without Momentum Damping When $\gamma = 0$, we have

$$H(x, m) = \alpha f(x) + \frac{1}{1+\varepsilon\lambda}(\mathcal{K}^*(x) + \mathcal{K}(m) - \lambda x^\top m),$$

Because $\min_m(\mathcal{K}^*(x) + \mathcal{K}(m) - \lambda x^\top m) = 0$, the algorithm also corresponds to solving $\min_x f(x)$ without regularization $\mathcal{K}^*(\lambda x)$.

It is interesting to see that the weight decay and momentum damping play a somewhat symmetric role, because turning off either one of it turns off the regularization term $\mathcal{K}^*(\lambda x)$. In particular, if $\mathcal{K}(x) = \|x\|_2^2/2$, the Lion-$\mathcal{K}$ ODE can be rewritten into a second-order ODE:

$$\ddot{x}_t + (\lambda + \gamma)\dot{x}_t + \varepsilon\alpha\nabla^2 f(x_t)\dot{x}_t + \gamma\lambda x_t + \alpha\nabla f(x_t) = 0, \tag{14}$$

in which the role of γ, λ are symmetric. Equation (21) coincides the high-resolution ODE in [35] for minimizing $F(x) = \alpha f(x) + \gamma\lambda\|x\|_2^2/2$, which is a high resolution continuous time limit of Nesterov momentum. The hessian-based damping term $\nabla^2 f(x_t)\dot{x}_t$ plays a key role for acceleration phenomenon [see e.g., 35, 1]. When we turn off the gradient enhancement ($\varepsilon = 0$), then we get ODE for Ployak momentum.

Interestingly, if we set $\lambda = \gamma = 0$, but $\varepsilon > 0$, ODE (21) still serve to minimize $f(x)$, due to the Hessian damping term.

Lion-$\mathcal{K}$ without Gradient Enhancement When $\varepsilon = 0$, we have

$$H(x, m) = \alpha f(x) + \frac{\gamma}{\lambda}\mathcal{K}^*(\lambda x) + (\mathcal{K}^*(\lambda x) + \mathcal{K}(m) - \lambda m^\top x),$$

and $\Delta_2(m, \tilde{m}) = 0$,

$$\Delta(x, m) = (\lambda + \gamma)\Delta_1(x, m) = (\lambda + \gamma)(m - \nabla\mathcal{K}^*(\lambda x))^\top(\nabla\mathcal{K}(m) - \lambda x).$$

In this case, minimizing $H(x, m)$ still yields the minimization of $F(x)$. Hence, the choice of ε does not alter the objective function.

Moreover, with $\varepsilon = 0$, one can conveniently decompose the velocity field in the form of (12), as a sum of a Hamiltonian vector field and mirror descent direction:

$$\begin{bmatrix}\dot{x}_t \\ \dot{m}_t\end{bmatrix} = \underbrace{\begin{bmatrix}+\nabla_m H(x_t, m_t) \\ -\nabla_x H(x_t, m_t)\end{bmatrix}}_{\text{Hamiltonian}} - \underbrace{\begin{bmatrix}0 \\ (\gamma + \lambda)(m_t - \nabla\mathcal{K}^*(\lambda x_t))\end{bmatrix}}_{\text{Descent}}.$$

This system can be shown to be equivalent to the Hamiltonian descent system for composite objects of [28]. Further, if $\lambda = 0$, it reduces to the conformal Hamiltonian system [e.g., 22, 24].

Mirror Descent and Frank-Wolfe If $\varepsilon\gamma = 1$, Lion-$\mathcal{K}$ reduces to

$$\dot{x}_t = \nabla\mathcal{K}(-\varepsilon\alpha\nabla f(x_t)) - \lambda x_t,$$

which can be shown to be equivalent to the Frank-Wolfe algorithm for minimizing $F(x) = \alpha f(x) + \frac{\gamma}{\lambda}\mathcal{K}^*(\lambda x)$.

When $\varepsilon\gamma = 1$, and $\lambda = 0$ with $\nabla\mathcal{K}(x) = 0$ iff $x = 0$, Lion-$\mathcal{K}$ reduces to $\dot{x}_t = \nabla\mathcal{K}(-\varepsilon\alpha\nabla f(x_t))$, which is dual space conditioning [23], or a variant of mirror descent for $\min_x f(x)$. See Appendix B.4 for more discussion.

Accelerated Mirror Descent The accelerated mirror descent of Krichene et al. [16] is

$$\dot{x}_t = \lambda_t(\nabla\mathcal{K}(m_t) - x_t), \qquad \dot{m}_t = -\alpha_t\nabla f(x_t),$$

which is shown to exhibit an acceleration behavior for minimizing a convex f (without the $\mathcal{K}^*$ regularization) when $\alpha_t = t/r$ and $\lambda_t = r/t$ and $r \geq 2$. This can be viewed as Lion-$\mathcal{K}$ ODE with $\gamma = 0, \varepsilon = 0$ and but a special time-dependent coefficient.

4 Discrete Time Analysis

We now present a result on the discrete-time Lion-$\mathcal{K}$ parallel to the continous-time results in Theorem 3.1, but work for non-differentiable convex functions $\mathcal{K}$. We analyze a slight reform of (2):

$$\begin{aligned} m_{t+1} &= \beta_2 m_t - (1 - \beta_2)\nabla f(x_t) \\ \tilde{m}_{t+1} &= \beta_1 m_t - (1 - \beta_1)\nabla f(x_t) \\ x_{t+1} &= x_t + \epsilon(\nabla\mathcal{K}(\tilde{m}_{t+1}) - \lambda x_{t+1}), \end{aligned} \tag{15}$$

in which we use an implicit scheme for the x_t-update, replacing λx_t with λx_{t+1}. It is equivalent to the explicit scheme in (2) with ϵ replaced by $\epsilon' = \frac{\epsilon}{1+\epsilon\lambda}$.

Theorem 4.1. *Assume $f\colon \mathbb{R}^d \to \mathbb{R}$ is L-smooth, and $\mathcal{K}\colon \mathbb{R}^d \to \mathbb{R}$ is closed and convex, and $\nabla\mathcal{K}$ is a subgradient of $\mathcal{K}$. Assume $\beta_1, \beta_2 \in (0, 1)$, and $\beta_2 > \beta_1$, and $\epsilon, \lambda > 0$.*

1) For any two non-negative integers $s \leq t$, we have

$$\operatorname{dist}(\lambda x_t, \operatorname{dom}\mathcal{K}^*) \leq \left(\frac{1}{1 + \epsilon\lambda}\right)^{s-t} \operatorname{dist}(\lambda x_s, \operatorname{dom}\mathcal{K}^*), \quad \forall s \leq t.$$

2) Define the following Lyapunov function:

$$H(x, m) = f(x) + \frac{1}{\lambda}\mathcal{K}^*(\lambda x) + \frac{\beta_1}{\epsilon\lambda(1 - \beta_1) + (1 - \beta_2)}(\mathcal{K}^*(\lambda x) + \mathcal{K}(m) - \lambda x^\top m),$$

and

$$\Delta_t^1 = (\nabla\mathcal{K}(\tilde{m}_{t+1}) - \lambda x_{t+1})^\top(\tilde{m}_{t+1} - \nabla\mathcal{K}^*(\lambda x_{t+1})) \geq 0,$$
$$\Delta_t^2 = (\nabla\mathcal{K}(\tilde{m}_{t+1}) - \nabla\mathcal{K}(m_{t+1}))^\top(\tilde{m}_{t+1} - m_{t+1}) \geq 0,$$

where $\nabla\mathcal{K}^*$ is a subgradient of $\mathcal{K}^*$. Then we have

$$H(x_{t+1}, m_{t+1}) - H(x_t, m_t) \leq -\epsilon\Delta_t + \frac{L\epsilon^2}{2}\left\|\nabla\mathcal{K}(\tilde{m}_{t+1}) - \lambda x_{t+1}\right\|_2^2,$$

where $\Delta_t = a\Delta_t^1 + b\Delta_t^2$, with

$$a = \frac{\beta_1}{\epsilon\lambda(1-\beta_1) + (1-\beta_2)} + 1 \geq 0, \qquad b = \frac{\beta_1(1-\beta_2)}{\epsilon\lambda(\beta_2-\beta_1)(\epsilon\lambda(1-\beta_1) + (1-\beta_2))} \geq 0.$$

Hence, a telescoping sum yields

$$\frac{1}{T}\sum_{t=0}^{T-1}\Delta_t \leq \frac{H(x_0, m_0) - H(x_T, m_T)}{\epsilon T} + \frac{L\epsilon}{2}B_T,$$

where $B_T = \frac{1}{T}\sum_{t=1}^{T}\left\|\nabla\mathcal{K}(\tilde{m}_{t+1}) - \lambda x_{t+1}\right\|_2^2$.

The result above shows that $\frac{1}{T}\sum_{t=0}^{T-1}\Delta_t$ decays with an $O(\frac{1}{\epsilon T} + \epsilon)$ rate, if B_T is a finite upper bound. This reduces to the continuous-time result of $\frac{1}{t}\int_0^t \Delta(x_s, m_s)\mathrm{d}s = O\left(\frac{1}{t}\right)$ when the step size ϵ converges to zero.

If $\mathcal{K}$ is smooth, it is possible to improve the discrete-time rate to $O\left(\frac{1}{\epsilon T}\right)$ with standard arguments based on the proof of Theorem 4.1. Hence, the impact of the non-differentiability of $\mathcal{K}$ contributes to the $O(\epsilon)$ term, which suggests that the algorithm converges upto an ϵ accuracy. This is an typical phenomenon in optimization with non-smooth objectives (like sub-gradient descent) or non-smooth update (like signed GD). Because in practice the step size is small or decaying, the $O(\epsilon)$ term may not have a substantial impact for practical performance.

5 Experiments on Different $\mathcal{K}$

This section provides a preliminary investigation on the behaviors of Lion-$\mathcal{K}$ with different $\mathcal{K}$. We experiment with the $\mathcal{K}$s listed in Table 2 on the toy example shown in Figure 1 to confirm the behavior follows exactly as what the theory predicts. Then we focus on the Lion-ℓ_p optimizer with general $p \in [1, 2]$ since it is the most straightforward extension of the original Lion (with $p = 1$).

5.1 Lion-$\mathcal{K}$s on the Toy Example

In the following, we plot the behavior of different Lion-$\mathcal{K}$s on the toy example shown in Figure 1. For each $\mathcal{K}$, we draw the optimization trajectory using the corresponding optimizer, the loss $f(x)$, and the corresponding constraint (e.g., the norm of x) v.s. iteration. The results are shown in Figure 5.

Observation From Figure 5, one can observe that for $\mathcal{K}(x) = \|x\|_2$, the constraint is a circle. For $\mathcal{K}(x) = \sum_i \max(|x_i| - e, 0)$, an additional ℓ_1 regularization is introduced in addition to the ℓ_∞ constraint, which encourages sparse solutions. When $\mathcal{K}(x) = \sum_{i\leq i^{cut}}|x_{(i)}|$, it enforces an ℓ_1 constraint (rather than regularization) in addition to the ℓ_∞ constraint. The $\mathcal{K}(x) = \sum_i \text{huber}_e(x_i)$ introduces an ℓ_2 regularization effect in addition to ℓ_∞ constraint. All optimization trajectories closely match what the theory predicts.

5.2 Lion-ℓ_p for ImageNet and Language Modeling

Lion-ℓ_p corresponds to $\mathcal{K}(x) = \|x\|_p$, $p \geq 1$ and amounts to solving $\min_x f(x)$ *s.t.* $\|x\|_q \leq 1/\lambda$ where $1/p + 1/q = 1$. In Figure 6, we plot how the parameter norms (e.g., $||\cdot||_\infty$ when $p = 1$ and $||\cdot||_2$ when $p = 2$) change over training iterations. In Figure 7, we compare the performance of using Lion-ℓ_p with different p, on ImageNet [34] and Language Modeling tasks, using ResNet-50, Vision Transformer (ViT) [9], and the GPT-2 model [31].

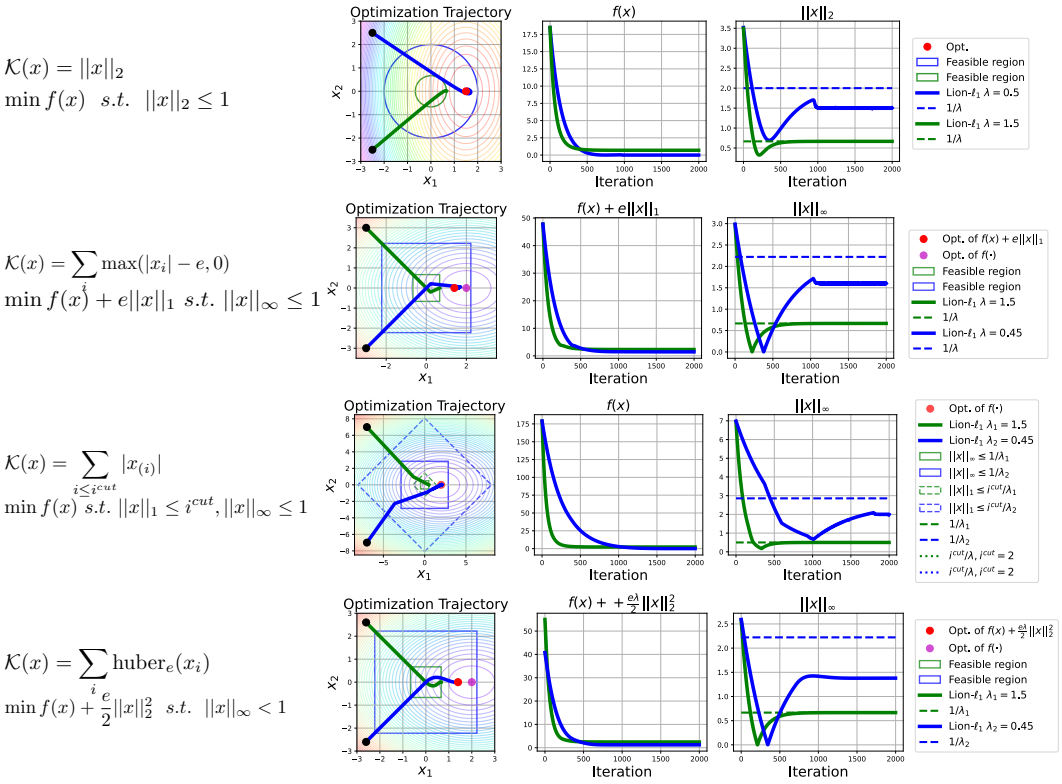

Figure 5: The behavior of Lion-$\mathcal{K}$ with different $\mathcal{K}$s from Table 2. The blue trajectory always reaches the optimum as the optimum is included in the constraint. The green trajectory converges to the boundary of the constraint.

Experiment Setting For the ImageNet training, we follow the standard PyTorch ImageNet training code.[1] We train the ResNet-50 and the ViT-B/16 model using batch size 1024 and cosine learning rate scheduler. For GPT-2 training, we follow the HuggingFace code[2], train it on OpenWebText[3] using cosine learning rate scheduler.

Observation From Figure 6, we observe that even on deep neural networks like ViT [9], ResNet [13], and GPT-2 [31], the behavior of the Lion-$\mathcal{K}$ optimizers strictly follow what the theory predicts. From Figure 7, we observe that Lion-ℓ_1 (the original Lion optimizer) performs better than Lion with other p on ImageNet when ViT is used, and on language modeling with the GPT-2 model. The plot indicates a trend that smaller $p \in [0, 1]$ results in better training efficiency. However, the trend is reversed when ResNet-50 [13] is used on ImageNet. Therefore, this indicates that the choice of $\mathcal{K}$ might depend on the underlying neural architecture. Based on the empirical observation, we conjecture that Lion-ℓ_1 performs well among all Lion-ℓ_p on the transformer architecture, which is consistent with the fact that Lion-ℓ_1 is found by an evolutionary search using the transformer architecture [6].

6 Discussion

As demonstrated in the analysis of the Lyapunov function in Theorem 3.1, the Lion-$\mathcal{K}$ dynamics exhibit a distinct nature when compared to typical momentum-based methods like Polyak, Nesterov momentum, and Hamiltonian descent, all of which can be conveniently understood as certain generalized dissipative Hamiltonian systems. While the Lyapunov function provides a powerful

[1] https://github.com/pytorch/examples/blob/main/imagenet/main.py.
[2] https://huggingface.co/gpt2
[3] https://huggingface.co/datasets/Skylion007/openwebtext

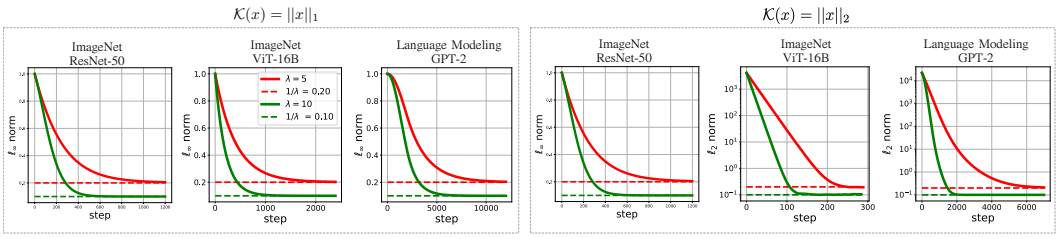

Figure 6: Constraint verification for Lion-ℓ_1 and Lion-ℓ_2 on ImageNet and Language Modeling tasks, using the ResNet-50, ViT-B/16 and the GPT-2 architectures.

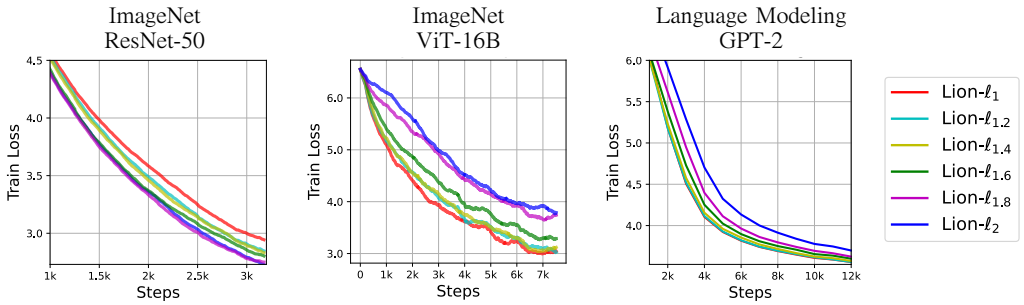

Figure 7: Performance of Lion-ℓ_p with different p, on ImageNet [34] (left 2 figures) and Language Modeling (right), using ResNet-50 [13] (left), ViT [9] (middle), and GPT-2 [31] (right).

characterization of the dynamical behavior, our intuitive understanding of the Lion-$\mathcal{K}$ dynamics remains obscured because we lack a "physical intuition" or constructive derivation like the standard optimization algorithms. This invites more studies in studies and understandings in future works.

The connection between Lion-$\mathcal{K}$ and Nesterov momentum and accelerated mirror descent suggests the possibility of acceleration phenomena in variants of Lion-$\mathcal{K}$, which opens an exciting avenue for future exploration and research. It might be possible to find novel accelerated algorithms based on the Lion-$\mathcal{K}$ family.

It is surprising and compelling that an algorithm found by a random search program has such a rich and intriguing theoretical basis. The reasons for this remain elusive, whether it is a coincidence or due to some inherent necessity. For instance, the design of the search space in Chen et al. [6] may in some way entails a high likelihood of discovering theoretically sound algorithms with random search. Understanding the underlying logic here could lead to future advancements in automatic machine-based algorithm discovery.

Regarding applications, since Lion-$\mathcal{K}$ offers a broader family than Lion, it is possible to find within the Lion-$\mathcal{K}$ family new algorithms that outperform Lion in various tasks and metrics. Additionally, by using different values of $\mathcal{K}$, Lion-$\mathcal{K}$ can be utilized to address different types of constraint optimization problems.

REFERENCES

[1] Hedy Attouch, Juan Peypouquet, and Patrick Redont. Fast convex optimization via inertial dynamics with hessian driven damping. *Journal of Differential Equations*, 261(10), January 2016. doi: 10.1016/j.jde.2016.08.020.

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

[3] Irwan Bello, Barret Zoph, Vijay Vasudevan, and Quoc V Le. Neural optimizer search with reinforcement learning. In *International Conference on Machine Learning*, pages 459–468. PMLR, 2017.

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

[5] 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.

[6] Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Yao Liu, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. Symbolic Discovery of Optimization Algorithms, 2023. arXiv:2302.06675 [cs].

[7] Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Yao Liu, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, et al. Symbolic discovery of optimization algorithms. *arXiv preprint arXiv:2302.06675*, 2023.

[8] Michael Crawshaw, Mingrui Liu, Francesco Orabona, Wei Zhang, and Zhenxun Zhuang. Robustness to unbounded smoothness of generalized signsgd. *arXiv preprint arXiv:2208.11195*, 2022.

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

[10] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, 1956. doi: 10.1002/nav.3800030109.

[11] Elad Hazan and Sham Kakade. Revisiting the Polyak step size, August 2022. URL http://arxiv.org/abs/1905.00313. arXiv:1905.00313 [math].

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

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

[15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[16] Walid Krichene, Alexandre Bayen, and Peter L Bartlett. Accelerated mirror descent in continuous and discrete time. *Advances in neural information processing systems*, 28, 2015.

[17] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[18] 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.

[19] Kfir Y Levy. The power of normalization: Faster evasion of saddle points. *arXiv preprint arXiv:1611.04831*, 2016.

[20] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[21] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

[22] Chris J Maddison, Daniel Paulin, Yee Whye Teh, Brendan O'Donoghue, and Arnaud Doucet. Hamiltonian descent methods. *arXiv preprint arXiv:1809.05042*, 2018.

[23] Chris J Maddison, Daniel Paulin, Yee Whye Teh, and Arnaud Doucet. Dual space preconditioning for gradient descent. *SIAM Journal on Optimization*, 31(1):991–1016, 2021.

[24] Robert McLachlan and Matthew Perlmutter. Conformal hamiltonian systems. *Journal of Geometry and Physics*, 39(4):276–300, 2001.

[25] Ryan Murray, Brian Swenson, and Soummya Kar. Revisiting normalized gradient descent: Fast evasion of saddle points. *IEEE Transactions on Automatic Control*, 64(11):4818–4824, 2019.

[26] Arkadij Semenovic Nemirovskij and David Borisovich Yudin. Problem complexity and method efficiency in optimization. 1983.

[27] Yurii Evgen'evich Nesterov. A method for solving the convex programming problem with convergence rate o (1/κˆ 2). In *Dokl. akad. nauk Sssr*, volume 269, pages 543–547, 1983.

[28] Brendan O'Donoghue and Chris J Maddison. Hamiltonian descent for composite objectives. *Advances in Neural Information Processing Systems*, 32, 2019.

[29] Daiyi Peng, Xuanyi Dong, Esteban Real, Mingxing Tan, Yifeng Lu, Gabriel Bender, Hanxiao Liu, Adam Kraft, Chen Liang, and Quoc Le. Pyglove: Symbolic programming for automated machine learning. *Advances in Neural Information Processing Systems*, 33:96–108, 2020.

[30] Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics*, 4(5):1–17, 1964.

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

[32] Esteban Real, Chen Liang, David So, and Quoc Le. Automl-zero: Evolving machine learning algorithms from scratch. In *International conference on machine learning*, pages 8007–8019. PMLR, 2020.

[33] R. T. Rockafellar. *Convex Analysis*, volume 11. Princeton University Press, 1997.

[34] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

[35] Bin Shi, Simon S Du, Michael I Jordan, and Weijie J Su. Understanding the acceleration phenomenon via high-resolution differential equations. *Mathematical Programming*, pages 1–70, 2021.

[36] Tijmen Tieleman, Geoffrey Hinton, et al. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4 (2):26–31, 2012.