OPT2024: 16th Annual Workshop on Optimization for Machine Learning

# Lion's sign noise can make training more stable

**Simon Elistratov*** SEMEN.ELISTRATOV@QSTFIN.COM
*Lomonosov Moscow State University, Moscow, Russia*

**Andrey Podivilov*** ANDREY.PODIVILOV@GMAIL.COM
*PDMI, Saint Petersburg, Russia*

**Timofei Iuzhakov†** TIUZHAKOV@CONSTRUCTOR.UNIVERSITY
*Constructor University, Bremen, Germany*
*HSE University, Moscow, Russia*

**Dmitry Vetrov†** DVETROV@CONSTRUCTOR.UNIVERSITY
*Constructor University, Bremen, Germany*

## Abstract

Lion is a novel optimization method that has outperformed traditional optimizers like Adam across a variety of tasks. Despite its empirical success, the reasons behind Lion's superiority remain unclear. In this paper, we investigate the mechanisms contributing to Lion's enhanced performance, focusing on the structured noise introduced by the use of the sign function in gradient updates. We characterize this noise by the angle of rotation between a vector and its signum. We inject this noise as a random fixed-angle rotation into normalized updates and analyze how the performance of this method compares to that of Lion. We demonstrate that this method has stronger performance than Lion in our setting. This approach reveals a relationship between the learning rate and the noise specific to the Lion method, providing insights into its improved performance metrics. Additionally, we identify an effect we term "momentum tracing" in neural networks with normalization layers and ReLU activations, which can significantly destabilize the training process. Our analysis demonstrates that the rotation noise inherent in Lion mitigates the negative impact of "momentum tracing", leading to more stable learning. These findings offer theoretical justification for Lion's effectiveness and suggest avenues for developing more robust optimization algorithms.

## 1. Introduction

Modern optimization methods leverage various properties of neural networks to enhance convergence toward solutions with superior performance. The loss landscape of neural networks possesses distinct characteristics that define criteria for effective optimization. Numerous studies have explored how these properties are connected to generalization capabilities and have contributed to the development of advanced optimization techniques.

One of the earliest and most widely adopted optimizers specifically developed for neural networks is Adam [8]. This method utilizes first and second-order momentum estimates to stabilize the training process. Several studies have demonstrated how these mechanisms help the optimizer converge to regions that offer better generalization [3, 5, 15].

. *Equal contribution.
. †Shared senior authorship.

In addition to traditional optimization methods, fixed-step approaches have also been explored for neural networks. Norm (S)GD normalizes the gradient, optimizing in its direction at each iteration with a step size equal to the learning rate [11, 13, 18]. Another method, Sign (S)GD, applies the sign function to the update direction. That results in each parameter being adjusted by the $\pm$ learning rate, depending on the sign of the corresponding gradient component [2, 10, 12, 16]. However, these methods have shown inferior performance and have not gained widespread adoption, although they have been utilized in other learning paradigms.

A novel method, Lion [4], was developed through programmatic search within a symbolic representation of optimizer space. Lion integrates various mechanisms from earlier methods, such as the sign function, decoupled weight decay, momentum, and decoupled momentum. The authors demonstrated that Lion outperforms Adam across a broad spectrum of CV and NLP tasks. However, due to Lion's novelty, there is a lack of theoretical justification and analysis explaining the reasons behind its superiority.

The use of the sign function in Lion leads to optimization that does not strictly follow the true gradient direction. This can be interpreted as an introduction of a specific noise into the optimization process. While optimization with a fixed step size in the gradient direction, as in Norm SGD, may seem more logical, it has been observed to result in poorer model performance. This paper aims to explore why Lion exhibits superior performance and how the noise introduced by perturbing the gradient direction helps stabilize the learning process.

This noise can be characterized by the angle between a gradient and its sign. Such noise can be synthetically introduced into the Norm SGD optimizer by applying random rotations with a fixed angle to the gradient update as a form of regularization. This characterization allowed us to identify the relationship between the learning rate and the noise specific to the Lion method, as well as explain why Lion might achieve better performance.

Furthermore, we identified an effect that we termed "momentum tracing", observed in both Lion and Norm SGD when they are applied to neural networks that include normalization layers and ReLU (or GELU) activations. This effect can lead to the destabilization of the training process. Our study investigates how the structured noise introduced by Lion mitigates the negative impact of this effect, contributing to more stable learning.

## 2. Rotation noise

Most Loss Function Landscape (LFL) theories work with rotation-invariant constructions, such as the spectral norm of the Hessian. From the rotation-invariant perspective, the choice of basis for the sign operation is random. The sign operation itself can be interpreted as a form of arbitrary rotary noise injection into gradient directions. This interpretation seems reasonable since noise in loss function gradients plays a crucial role in neural networks' optimization [14]. We begin by theoretically analyzing this noise distribution.

**Definition 1** *We define **Lion Noise Injection** (LNI) on a vector $v$ as follows: sample an orthonormal basis $B$ uniformly, express $v$ in the coordinates of $B$, and then apply the sign function to these coordinates.*

Alternatively, we can describe LNI as follows: $\mathrm{LNI}(v) = w_B\sqrt{\dim(v)}$, where $w_B$ is a normalized vector along the bisector of the hyperoctant to which $v$ belongs, with respect to a sampled basis $B$. To analyze the noise distribution, we can reformulate the procedure: instead of sampling a basis,

we can fix a basis and sample a vector $v$ uniformly from the sphere — it gives us an equivalent rotary noise distribution. For spaces with a sufficiently high dimension, this distribution is spherically symmetric and degenerates to a specific angle, see fig. 1. This is intuitive. Indeed, denote

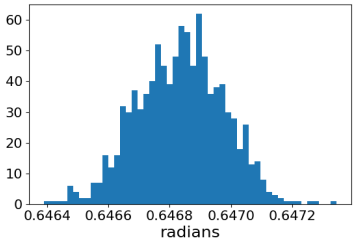


Figure 1: A histogram of the rotary noise sampled uniformly from the sphere $\mathbb{S}^{6000000}$.

by $\angle(x, y)$ an angle between vectors $x$ and $y$ and let $e = (1, 1, \ldots, 1)/\sqrt{\dim}$ be a bisector of the positive hyperoctant, where $\dim$ is the dimension of the space. Then the distribution density at an angle $\alpha$ is proportional to a total measure of a manifold

$$M(\alpha) = \{x \in \mathbb{S}^{n-1} \mid x_i \geq 0 \; \forall i \text{ and } \angle(x, e) = \alpha\}.$$

The manifold $M(\alpha)$ consists of points on the unit sphere $\mathbb{S}^{n-1}$ that both lie in the positive hyperoctant $\mathbb{H}^+$ and are equidistant to $e$. The equidistance condition from vector $e$ defines an $(n-2)$-dimensional sphere $S_\alpha$ with radius $\sin(\alpha)$. Its measure grows as $f(\alpha) = \sin^{n-2}(\alpha)$ — a rate that becomes extremely rapid at typical neural network dimensions. We can say that the full measure of $M(\alpha)$ is proportional to $f(\alpha)g(\alpha)$, where $0 \leq g \leq 1$ is a function of a different nature: it measures the fraction of $S_\alpha$ contained in $\mathbb{H}^+$ — a ratio that decreases to 0 as $\alpha$ increases. For the product $fg$ to maintain similar values near $\operatorname{argmax}(fg)$, function $g$ would need to closely mirror $f$ — which is unlikely given its dissimilar nature and $f$'s extreme growth rate.

A significant distinction between this model and practice is that in practice, gradient components' distribution tends to have heavy tails [17]. This implies that gradient directions are concentrated near the hyperoctant's border rather than being uniformly distributed over the sphere.

As shown in fig. 2, Lion does exhibit the degenerate angle property, although the observed angle consistently exceeds the theoretical estimate. Moreover, the angle depends heavily on the learning rate. The exact mechanism behind this relationship remains unclear. One hypothesis stems from an observation that the cosine of the angle is proportional to a ratio of $L^2/L^1$ gradient norms. In our future work, we aim to demonstrate that Lion regularizes $L^1$-norm of the gradient with a strength proportional to the learning rate.

Consider the following optimization scheme that mirrors Lion (see definition 3).

**Definition 2** *Define **Enim** (Evolved Noise Injection Momentum) as an algorithm that modifies Lion by replacing the sign operation with LNI. The update rule of Enim:*

$$\begin{cases} m_{t+1} = \beta_2 m_t + (1 - \beta_2)\nabla L(w_t) \\ w_{t+1} = (1 - \eta\lambda)w_t + \eta \operatorname{LNI}(\beta_1 m_t - (1 - \beta_1)\nabla L(w_t)) \end{cases} \tag{1}$$

*where $\eta$ is the learning rate, $\lambda$ is the weight decay coefficient, $m$ is momentum, $w$ is the network weights.*

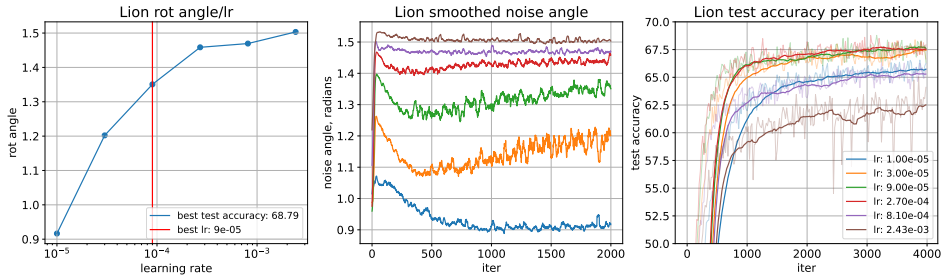


Figure 2: Multiple runs of Lion with different learning rates on ResNet9 trained for 200 epochs on CIFAR100. (Left:) Lion runs with larger learning rates correspond to larger angles. (Center:) The angle by which Lion's sign operation rotates the update vector. The observed angles exceed the theoretical estimation (which is based on the assumption that the gradients are uniformly sampled from the sphere).

To test whether this view of the sign operation is reasonable, we compare Lion directly to Enim. We calibrate Enim's noise injections to match Lion's empirically observed sign noise. More precisely, for each learning rate we first select an angle matching Lion's sign operation angle (observed with this learning rate). Then, at each iteration, we rotate the update vector by this angle in a random direction and normalize it to match Lion's step size. The right graph of fig. 3 shows that Enim with matched noise injections performs slightly better than Lion (see the right graph of fig. 2).

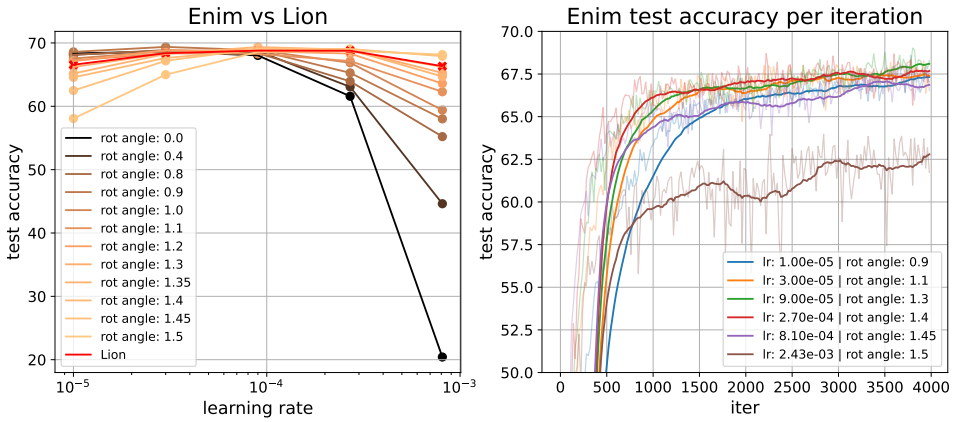


Figure 3: Enim's performance on ResNet9 trained for 200 epochs on CIFAR100, with noise angles matching those empirically observed for Lion's sign operation. The right graph (together with the right graph of fig. 2) shows that Enim performs slightly better than Lion, even without additional noise angle tuning.

The natural question then becomes how different noise angles interact with Enim's performance and to what extent we can enhance its performance by varying the noise. See the left graph of fig. 3 (an analogous graph for noise is presented in the Appendix). As it turns out, the angles observed

empirically for Lion yield near-optimal performance for Enim, at least in our small-scale CIFAR100 setting. One important observation is that the optimal noise angle for Enim increases when we increase the learning rate. This is counter-intuitive from LFL theories perspective, as both angle growth and learning rate growth make positive contributions to the noisiness of the optimization process. One explanation for this phenomenon lies in two effects described below. The first, less dramatic one, is that Enim with small learning rates and large angles does not fully converge in 200 epochs. The second one, which we term "momentum tracing", is discussed in the next section. For Enim we observe that large noise angles soften this effect, preventing the optimization process from destabilizing.

## 3. Momentum Tracing

Most modern neural networks incorporate normalization layers, such as Batch Normalization [7] and Layer Normalization [1], followed by ReLU or similar activations. When the preactivation feature map components are negative, ReLU outputs zero, resulting in no backpropagation signal through those components. Consequently, the corresponding components of the bias term in the normalization layer receive zero gradient.

After the gradient becomes zero, these components continue to update in the direction of momentum for an extended period until the gradient becomes non-zero again or the momentum value diminishes to numerical precision limits. If the momentum components associated with the bias term of the normalization layers are negative, a positive feedback loop can occur. In this scenario, the bias components shift further into negative values, reducing the preactivation values even more, which in turn maintains zero gradients.

By the moment when the gradient signal suddenly reappears (e.g., due to skip-connections), the weight values contributing to the corresponding activation may have shifted into suboptimal regions, causing abrupt changes that destabilize the training process. We term this effect "momentum tracing". With both convolutional and linear layers multiple parameters lose the gradient signal when the corresponding BatchNorm activation outputs zero.

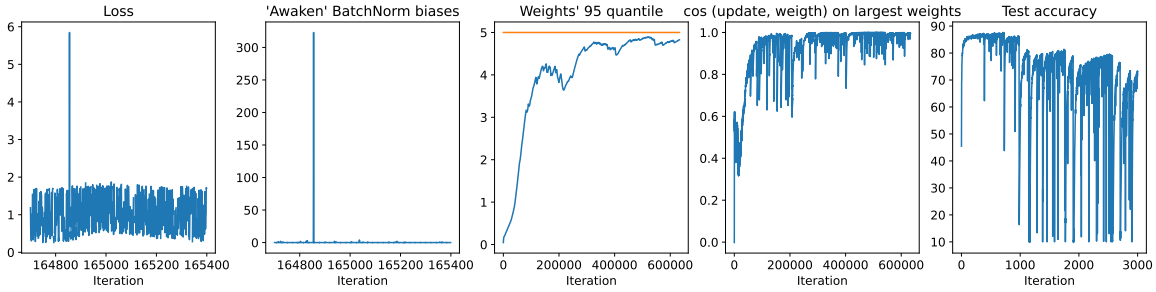


Figure 4: A display of Lion divergence with ResNet18 on CIFAR-10 . (Center left:) A number of BN biases with a near-zero momentum that receive a non-zero gradient on the current iteration. (Center:) Blue — a 95 quantile of networks' parameters values. Under Lion-weight decay dynamics weights converge to $\pm 5 = \pm 1/\lambda$ when their gradients are identical to zero. (Center right:) Cosine similarity between 5% largest weights and their updates converges to 1 as the weights converge to $1/\lambda$.

This effect can occur with any optimization method. However, in methods like Adam, the magnitude of the update for a component decreases rapidly at a geometric progression rate of approximately $\beta_1/\sqrt{\beta_2}$, where commonly used values are $\beta_1 = 0.9$ and $\beta_2 = 0.999$. This rapid decay prevents the optimizer from pushing the weights too far into suboptimal regions. In contrast, methods like Norm Gradient Descent (Norm GD) with momentum and decoupled momentum mechanisms similar to Lion exhibit a slower decay of the momentum component, typically with $\beta_2 = 0.99$. In Lion, the momentum decay rate is similar to that of Norm GD, but due to the use of the signum function, the method can take larger update steps for a component, which would be expected to lead to even greater destabilization. However, the noise introduced by the method reduces the frequency of zero gradients.

In experiments with Enim, this effect is most noticeable when the noise is set to zero and with large learning rates. The larger the learning rate, the more significant the steps the optimizer takes in the direction of momentum when the gradient has diminished. Introducing noise by rotating the update direction by a random fixed angle can result in gradients becoming non-zero earlier, preventing the optimizer from taking excessive steps in the momentum direction.

## 4. Conclusion

In this work, we analyzed the characteristics of the specific sign noise introduced by the Lion optimizer. We demonstrated that Lion adapts the rotation noise in accordance with the learning rate to stabilize the training process, which can significantly impact model performance. Additionally, we introduced a novel optimization method that employs random rotations of the gradient update at fixed angles, serving as a tool for analyzing Lion's behavior.

**Future work.** We plan to delve deeper into the mechanisms of noise adaptation in Lion and provide theoretical explanations for why the rotation angle positively correlates with the learning rate value. Furthermore, we intend to analyze the features of the loss landscape associated with Lion's attractors, which will enable us to clearly formulate the inductive bias properties of the Lion optimizer.

## Acknowledgments

This research was supported by the grant for research centers in the field of AI provided by the Analytical Center for the Government of the Russian Federation (ACRF) in accordance with the agreement on the provision of subsidies (identifier of the agreement 000000D730321P5Q0002) and the agreement with HSE University №70-2021-00139. The empirical results were supported in part through the computational resources of HPC facilities at HSE University [9]. Part of the experiments were conducted using the Constructor Research Platform [6].

## References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. URL https://arxiv.org/abs/1607.06450.

[2] Lukas Balles, Fabian Pedregosa, and Nicolas Le Roux. The geometry of sign gradient descent. *arXiv preprint arXiv:2002.08056*, 2020.

[3] Matias D Cattaneo, Jason M Klusowski, and Boris Shigida. On the implicit bias of adam. *arXiv preprint arXiv:2309.00079*, 2023.

[4] Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, et al. Symbolic discovery of optimization algorithms. *Advances in neural information processing systems*, 36, 2024.

[5] Jeremy M Cohen, Behrooz Ghorbani, Shankar Krishnan, Naman Agarwal, Sourabh Medapati, Michal Badura, Daniel Suo, David Cardoze, Zachary Nado, George E Dahl, et al. Adaptive gradient methods at the edge of stability. *arXiv preprint arXiv:2207.14484*, 2022.

[6] Constructor Research Platform. URL https://constructor.tech/products/research-platform.

[7] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. URL http://arxiv.org/abs/1502.03167.

[8] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[9] P. S. Kostenetskiy, R. A. Chulkevich, and V. I. Kozyrev. HPC resources of the higher school of economics. *Journal of Physics: Conference Series*, 1740:012050, 2021.

[10] Xiuxian Li, Kuo-Yi Lin, Li Li, Yiguang Hong, and Jie Chen. On faster convergence of scaled sign gradient descent. *IEEE Transactions on Industrial Informatics*, 20(2):1732–1741, 2023.

[11] Danilo P Mandic. A generalized normalized gradient descent algorithm. *IEEE signal processing letters*, 11(2):115–118, 2004.

[12] Emmanuel Moulay, Vincent Léchappé, and Franck Plestan. Properties of the sign gradient descent algorithms. *Information Sciences*, 492:29–39, 2019.

[13] 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.

[14] Arvind Neelakantan, Luke Vilnis, Quoc V. Le, Ilya Sutskever, Lukasz Kaiser, Karol Kurach, and James Martens. Adding gradient noise improves learning for very deep networks, 2015. URL https://arxiv.org/abs/1511.06807.

[15] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*, 2019.

[16] Mher Safaryan and Peter Richtárik. Stochastic sign descent methods: New algorithms and better theory. In *International Conference on Machine Learning*, pages 9224–9234. PMLR, 2021.

[17] Umut Simsekli, Levent Sagun, and Mert Gurbuzbalaban. A tail-index analysis of stochastic gradient noise in deep neural networks, 2019. URL https://arxiv.org/abs/1901.06053.

[18] Shen-Yi Zhao, Yin-Peng Xie, and Wu-Jun Li. On the convergence and improvement of stochastic normalized gradient descent. *Science China Information Sciences*, 64:1–13, 2021.

## Appendix A. Lion scheme

**Definition 3** *The update rule of Lion:*

$$\begin{cases} m_{t+1} = \beta_2 m_t + (1-\beta_2)\nabla L(w_t) \\ w_{t+1} = (1-\eta\lambda)w_t + \eta\,\mathrm{sign}(\beta_1 m_t - (1-\beta_1)\nabla L(w_t)) \end{cases} \tag{2}$$

*where $\eta$ is the learning rate, $\lambda$ is the weight decay coefficient, $m$ — momentum, $w$ — network weights.*

## Appendix B. Rotation angle dependence

Specified effects was better visible for methods with momentum and affine transform in normalization layers.

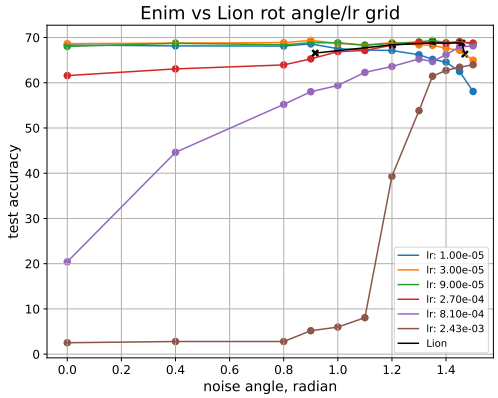


Figure 5: For bigger learning rate values decreasing noise level dramatically deteoriate model performance. Also, it is noticeable that Enim has slightly better performance than Lion.

## Appendix C. Momentum tracing

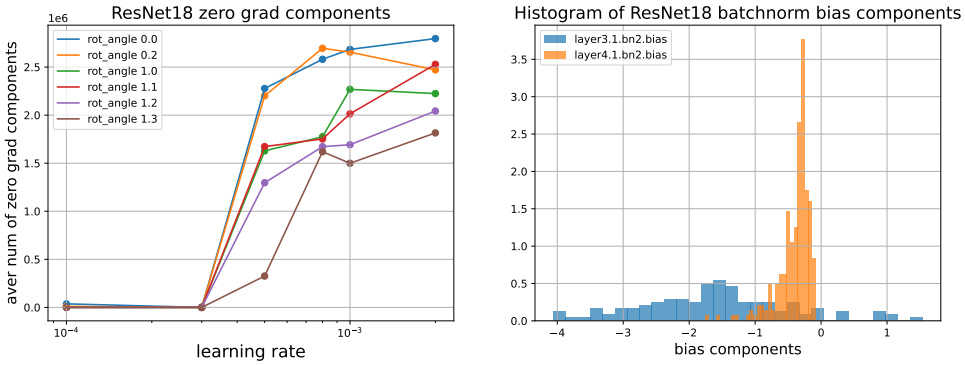


Figure 6: (Left:) Relationship between learning rate, noise and average number of zero components in gradient. The more zero components gradient have, the more the method can be destabilized due to momentum tracing. (Right:) Distribution of bias term in batch normalization layer. Most components are negative, which could lead to positive feedback loop.