OPT2024: 16th Annual Workshop on Optimization for Machine Learning

Revisiting the Initial Steps in Adaptive Gradient Descent Optimization

Abulikemu Abuduweili ABULIKEA@ANDREW.CMU.EDU
Changliu Liu CLIU6@ANDREW.CMU.EDU
Robotics Institute, Carnegie Mellon University

Abstract

Adaptive gradient optimization methods, such as Adam, are prevalent in training deep neural networks across diverse machine learning tasks due to their ability to achieve faster convergence. However, these methods often suffer from suboptimal generalization compared to stochastic gradient descent (SGD) and exhibit instability, particularly when training Transformer models. In this work, we show the standard initialization of the second-order moment estimation ($v_0 = 0$) as a significant factor contributing to these limitations. We introduce simple yet effective solutions: initializing the second-order moment estimation with non-zero values, using either data-driven or random initialization strategies. Empirical evaluations demonstrate that our approach not only stabilizes convergence but also enhances the final performance of adaptive gradient optimizers.

1. Introduction

First-order optimization methods, such as stochastic gradient descent (SGD), have been foundational in training deep neural networks due to their robust convergence properties across various applications [3]. However, as deep learning architectures have grown more complex, there has been increasing interest in adaptive gradient optimizers, which dynamically adjust learning rates based on the gradients of individual parameters [6]. These methods often lead to faster convergence in certain tasks [12]. Among them, Adam has emerged as one of the most widely used adaptive gradient methods, successfully applied to fields such as computer vision, natural language processing, and reinforcement learning [17]. By combining the benefits of momentum and adaptive learning rates, Adam has proven particularly effective in training generative models and large language models [44]. Theoretical studies have further elucidated its convergence properties in non-convex settings, providing insights into convergence rates [45]. With careful hyperparameter tuning, Adam has achieved significant success, especially in transformer-based architectures [15, 31, 37].

Despite its fast-convergence property, Adam has been observed to suffer from instability and poor generalization in certain non-convex optimization problems, such as training transformers for language models [23, 36]. This instability often causes the optimizer to converge to suboptimal local minima, thereby limiting the model's performance. Several modifications have been proposed to address these issues. For instance, AdaBound [26] improves generalization by bounding the step size with a smooth parameter update, while RAdam [23] rectifies the variance of the second-order moment to stabilize the learning rate during early iterations. AdaBelief [47] adapts the step size based on the "belief" in the observed gradients, enhancing generalization. A broader range of studies has introduced further refinements to stabilize convergence and improve generalization performance [1, 16, 43]. The warmup heuristic, which employs a small learning rate during the initial training epochs, has been adopted to improve stability and generalizability in Adam [42].

The update rule of Adam can be understood as a combination of update direction, determined by the sign of the stochastic gradients, and update magnitude [2]. Recent works have explored the role of Sign Gradient Descent (SignGD) as a surrogate for understanding Adam's behavior [19, 20]. We identify a critical factor contributing to Adam's instability: its default initialization of the second-order moment estimation ($v_0 = 0$), which causes Adam to exhibit sign-descent behavior in its initial steps. This default setting introduces high variance in the second-moment estimation and update step size, resulting in unstable convergence, particularly during the early stages of training. This instability often prevents the optimizer from reaching well-generalized optima. To address this issue, we propose a simple yet effective modification: initializing the second-order moment estimation with non-zero values. These initial values can be derived from data-driven statistics of squared gradient, or even assigned as random positive numbers. This modification reduces the variance of the second moment and stabilizes the optimization process. Our empirical evaluations across a wide range of tasks demonstrate that the proposed initialization of the second-order moment significantly improves the stability and overall performance of adaptive gradient optimizers, particularly in non-convex settings.

2. Second-order Moment Initialization of Adam

2.1. Revisiting the Adam Optimizer

Update rule of Adam. The update rule for Adam is given by the following equations [17]:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t = \beta_1^t m_0 + (1-\beta_1) \sum_{k=0}^{t-1} \beta_1^k g_{t-k},\ \hat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{1}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 = \beta_2^t v_0 + (1-\beta_2) \sum_{k=0}^{t-1} \beta_2^k g_{t-k}^2,\ \hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{2}$$

$$\theta_t = \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{3}$$

where m_t and v_t represent the first and second moments, g_t is a gradient of objective function. β_1, β_2 are the decay rates for the first and second-moment estimates, α is the learning rate, and ϵ is a small constant preventing division by zero.

First step of Adam as sign descent. In Adam's standard implementation, the first- and second-order momentum terms are initialized to zero, $m_0 = 0, v_0 = 0$. Ignoring ϵ, as a result, the first step of the optimization process degenerates into sign descent. This behavior is illustrated as follows:

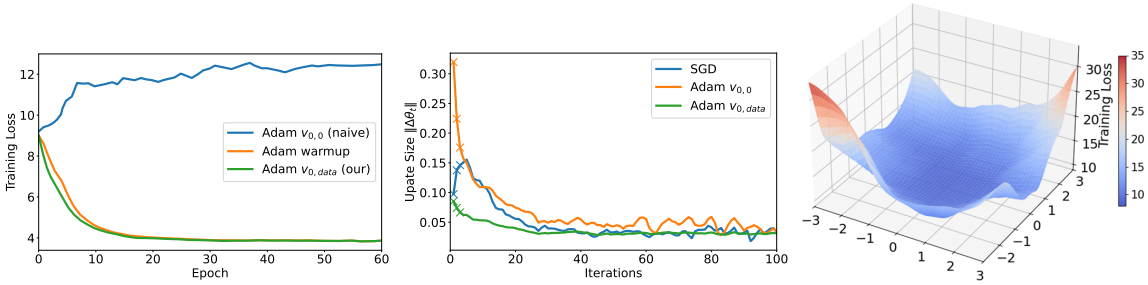
$$\Delta\theta_1 = -\alpha \frac{g_1}{\sqrt{g_1^2 + \frac{\beta_2}{1-\beta_2} v_0}} = -\alpha \cdot \text{sign}(g_1). \tag{4}$$

In this first step, Adam performs a pure sign-descent update due to the zero initialization of $m_0 = 0, v_0 = 0$. Over subsequent iterations, as more gradient information is accumulated, the influence of the initial sign descent diminishes, and the optimizer transitions into its adaptive behavior.

2.2. Instability of Adam optimizer

Training Transformer models often relies on a learning rate warmup strategy [4, 5, 8]. Removing the warmup phase, however, has been observed to increase training loss [23]. To explore this

phenomenon, we conducted experiments training a Transformer model on the IWSLT'14 DE-EN dataset for a neural machine translation task. We evaluated three approaches: vanilla Adam without warmup (denoted as $v_{0,0}$), vanilla Adam with warmup, and our proposed data-driven initialization of Adam without warmup (denoted as $v_{0,data}$, described in the next section). As illustrated in Figure 1(a), vanilla Adam without warmup exhibits increased training loss during the early stages. We attribute this instability to Adam's initial sign-descent behavior, which is exacerbated by the standard zero-initialization of the second-order moment ($v_0 = 0$). While the learning rate warmup strategy effectively addresses this issue, it requires using a very small learning rate during the initial stages, limiting parameter updates and slowing down convergence. In this work, we propose a non-zero initialization strategy to directly stabilize the optimizer. Unlike warmup, our approach avoids restrictive learning rate constraints, enabling faster convergence while maintaining training stability. The detailed discussion on the impact of sign descent and shrinking gradients is presented in Appendix A.1.



(a) Training loss curve (b) Update step vs. iterations (c) Initial loss landscape

Figure 1: Training Transformers on the IWSLT'14 De-En dataset.

2.3. Non-zero Initialization of Second Order Moment

Special case: linear loss. To build intuition for initializing the second-order moment, we first study a simplified setting. Consider the linear loss function $f(\theta_t) = \langle \theta_t, g_t \rangle$ with a Noisy Gradient Oracle with Scale Parameter (NGOS), a widely used framework for analyzing training dynamics of optimizers [22, 28]. In this setting, the stochastic gradient g_t is sampled from a Gaussian distribution with mean $\bar{g}$ and and variance $\sigma^2 I$, i.e. $g_t \sim \mathcal{N}(\bar{g}, \sigma^2 I)$. This setup mimics mini-batch training in neural networks, where the stochastic gradient is provided as a noisy approximation of the full gradient. Using this framework, the expectation of first- and second-order moments is given by

$$\mathbf{E}[v_t] = \beta_2^t v_0 + (1-\beta_2)\sum_{k=0}^{t-1} \beta_2^k (\bar{g}^2 + \sigma^2 I) = \beta_2^t v_0 + (1-\beta_2^t)(\bar{g}^2 + \sigma^2 I) \tag{5}$$

These results indicate that, after a sufficient number of steps, $\mathbf{E}[v_t] \approx \bar{g}^2 + \sigma^2 I$. For v_t, which represents the second-order moment of the gradient, it must satisfy $\mathbf{E}[v_t] > 0$. This makes the standard zero initialization ($v_0 = 0$) inherently inconsistent with its purpose. To assess the stability of the optimization process and the influence of the initial state, we define the drift of the second-order moment as $\text{drift}_{v_t}(v_0) = \|\mathbf{E}[v_\infty]| - \mathbf{E}[v_0]\|$. This term quantifies the adjustment required for the second moment to transition from its initial value to its steady-state. Since v_t directly determines the adaptive learning rate, a smaller drift term indicates better stability of optimization process.

For vanilla Adam, $v_0 = 0$, the drift value is $\text{drift}_{v_t}(v_0 = 0) = \bar{g}^2 + \sigma^2$. This large drift causes significant initial adjustments of v_t, leading to potential instability in optimization. For non-zero initialization, the drift value is $\text{drift}_{v_t}(v_0 = \bar{g}^2 + \sigma^2 I) = 0$. With this initialization, v_t is immediately aligned with its steady-state value, eliminating the need for adjustments and ensuring stability from the start. When β_2 closer to 1, v_t becomes nearly deterministic and tightly concentrates around $v_t \approx \bar{g}^2 + \sigma^2 I$. Ignoring ϵ for simplicity, the Adam update rule becomes:

$$\theta_t \approx \theta_{t-1} - \alpha \frac{m_t}{\sqrt{\bar{g}^2 + \sigma^2 I}} \tag{6}$$

This ensures a stable adaptive learning rate: $\alpha \cdot (\bar{g}^2 + \sigma^2 I)^{-1/2}$. Such stability aligns with the definition of an adaptive learning rate, where v_t incorporates local geometry (e.g., Hessian information). For the linear loss case, this stability results in more consistent updates. Further illustration of the stability provided by a non-zero v_0 in RMSprop is presented in Appendix A.2.

For random initialization, $v_0 = \lambda I, \lambda > 0$, the the drift term becomes: $\text{drift}_{v_t}(v_0 = \lambda I) = |\bar{g}^2 + \sigma^2 - \lambda|$. For any $0 < \lambda < 2(\bar{g}^2 + \sigma^2)$, this drift term is smaller than that of zero initialization: $\text{drift}_{v_t}(v_0 = \lambda I) < \text{drift}_{v_t}(v_0 = 0)$. This reduced drift results in a more stable optimization process compared to $v_0 = 0$, even with random initialization.

Initialization of v_0. Inspired by the analysis of linear loss cases with stochastic gradients, we propose two different non-zero initialization strategies for the second-order moment v_0.

- ***Data-driven Initialization***, denoted as $v_{0,data}$. In the data-driven strategy, v_0 is initialized using the gradient statistics calculated from sampled training data $(x_i, y_i) \sim \mathcal{D}$, where $\mathcal{D}$ represents the training set. Specifically, for sampled data (x_i, y_i), the gradient of the loss function is computed as: $g(x_i, y_i) = \nabla_\theta f(x_i, y_i)$ for (x_i, y_i). The second-order moment is then initialized as:

$$v_0 = \sigma \cdot \left(\mathbf{E}[g(x_i, y_i)]^2 + \mathbf{VAR}[g(x_i, y_i)]\right), \text{ where } (x_i, y_i) \sim \mathcal{D}. \tag{7}$$

  Here, σ is a hyperparameter that controls the scale of v_0.

- ***Random Initialization***, denoted as $v_{0,rnd}$. This is computationally efficient and avoids the overhead associated with data-driven initialization. As shown in the previous analysis, any small positive value for v_0 enhances the stability of v_t, making random initialization a practical choice. We propose initializing v_0 using a scaled Chi-squared distribution [1]:

$$v_0 \sim \frac{\sigma}{\text{fan\_in} + \text{fan\_out}} \cdot \chi_1^2, \tag{8}$$

  where χ_1^2 denotes a chi-squared distribution with one degree of freedom. fan_in and fan_out are the input and output dimensions of the weight matrix $\theta \in \mathcal{R}^{\text{fan\_out} \times \text{fan\_in}}$, and σ is a hyperparameter that controls the scale of the distribution. Furthermore, the squared value g_t^2 of a Gaussian random gradient g_t naturally follows a scaled chi-squared distribution, providing a principled foundation for this initialization strategy.

Under the proposed initialization $v_{0,data}$ and $v_{0,rnd}$, the first update step is influenced by both the magnitude and direction of the gradient, avoiding the pure "sign descent" behavior seen with $v_0 = 0$. Such stabilization is particularly crucial for deep learning tasks with shrinking gradients, such as training Transformers. A discussion comparing the proposed initialization strategy with other optimization approaches is presented in Appendix A.3.

1. Which is also can be described as Gamma distribution $v_0 \sim \text{Gamma}\left(\frac{1}{2}, \frac{2(\text{fan\_in}+\text{fan\_out})}{\sigma}\right)$

3. Experiments

To evaluate the effectiveness of our approach, we conducted extensive experiments across a variety of tasks, including image classification with convolutional neural networks (CNNs) [10], image generation with generative adversarial networks (GANs) [7], language modeling with long short-term memory networks (LSTMs) [13], and neural machine translation with Transformers [42]. We empirically evaluate the performance of two initialization strategies — $v_{0,data}$ (Equation (7)) and $v_{0,rnd}$ (Equation (8)) — across several widely used adaptive gradient optimization methods. These methods include SGD with momentum [34, 38], Adam [17], AdamW [25], AdaBound [26], RAdam [23], and AdaBelief [47]. For each optimizer, we use the standard initialization ($v_0 = 0$) as the baseline and compare it against the proposed strategies ($v_{0,rnd}$ and $v_{0,data}$). Detailed experimental setup information is provided in Appendix B.1. To illustrate Adam's instability and the impact of initialization, we first conduct a toy experiment, detailed in Appendix B.2.

Image Classification with CNN. We evaluate the ResNet-34 architecture [10] on the CIFAR-10 image classification dataset [18]. The test accuracy at the final epoch is summarized in Table 1. The results demonstrate that the proposed initialization of v_0, represented as $v_{0,rnd}$ and $v_{0,data}$, enhances the performance of adaptive gradient optimization methods, including Adam, AdamW, AdaBound, RAdam, and AdaBelief. To further validate the effectiveness of our algorithm on a larger dataset, we conducted experiments on the ImageNet dataset, detailed in Appendix B.3.

Table 1: Test accuracy ↑ (%) of ResNet-34 on CIFAR-10 dataset.

Optimization	SGD	Adam	AdamW	AdaBound	RAdam	AdaBelief
Vanilla $v_{0,0}$	96.19±0.09	95.25±0.11	95.36±0.11	95.38±0.07	95.61±0.16	95.94±0.07
$v_{0,rnd}$	-	95.87±0.09	95.94±0.09	95.80±0.07	95.83±0.11	96.11±0.07
$v_{0,data}$	-	96.02±0.09	95.95±0.09	95.96±0.07	95.90±0.12	**96.24±0.07**

Language Modeling with LSTM. We evaluate a 2-layer LSTM network [13] on the language modeling task of Penn Treebank dataset [29]. The test perplexity (lower is better) is summarized in Table 2. The results demonstrate that both $v_{0,rnd}$ and $v_{0,data}$ significantly improve the performance of adaptive gradient methods.

Table 2: Test perplexity ↓ of 2 Layer LSTM on Penn Treebank dataset dataset.

Optimization	SGD	Adam	AdamW	AdaBound	RAdam	AdaBelief
Vanilla $v_{0,0}$	67.25±0.20	67.11±0.20	73.61±0.15	67.69±0.24	73.61±0.25	66.75±0.11
$v_{0,rnd}$	-	66.70±0.17	68.35±0.14	66.94±0.19	68.55±0.17	66.12±0.10
$v_{0,data}$	-	66.37±0.17	69.31±0.14	66.90±0.19	69.32±0.17	**65.87±0.10**

Neural Machine Translation with Transformer. We evaluated a small Transformer model [42] using the Fairseq package [30] on the IWSLT'14 German-to-English machine translation dataset. The BLEU scores [32] are summarized in Table 3. The results demonstrate that the proposed initialization strategies, $v_{0,rnd}$ and $v_{0,data}$, provide significant performance improvements for adaptive gradient optimization methods.

Image Generation with GAN. We evaluated a deep convolutional GAN (DCGAN) [35] on the CIFAR-10 image generation task. The performance is measured using the Frechet Inception Distance (FID, lower is better) [11], which quantifies the similarity between generated images and

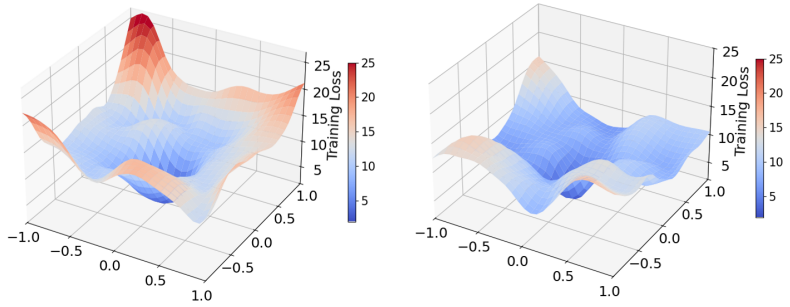
Table 3: BLEU score ↑ of Transformer on IWSTL'14 DE-EN dataset.

Optimization	SGD	Adam	AdamW	RAdam	AdaBelief
Vanilla $v_{0,0}$	28.22±0.21	30.14±0.39	35.62±0.11	34.76±0.14	35.60±0.11
$v_{0,rnd}$	-	33.71±0.19	36.06±0.11	34.97±0.14	36.12±0.11
$v_{0,data}$	-	33.64±0.20	35.98±0.11	34.84±0.14	**36.18±0.11**

Table 4: FID score ↓ of GAN on CIFAR-10 dataset dataset.

Optimization	SGD	Adam	AdamW	AdaBound	RAdam	AdaBelief
Vanilla $v_{0,0}$	237.77±147.9	54.22±4.21	52.39±3.62	118.75±40.64	48.24±1.38	47.25±0.79
$v_{0,rnd}$	-	48.60±3.19	46.94±3.21	92.36±35.76	47.70±1.32	45.91±0.78
$v_{0,data}$	-	47.02±3.20	45.25±3.07	85.45±36.31	47.84±1.24	**45.02±0.78**

the real dataset. As shown in Table 4, the proposed initialization strategies, $v_{0,rnd}$ and $v_{0,data}$, stabilize the optimization process for adaptive gradient methods, resulting in additional performance gains.



(*a*) Vanilla Adam (*b*) Adam $v_{0,rnd}$

Figure 2: Comparison of the loss landscape around the convergent points of Transformer.

Loss landscape. To analyze the converged behavior of Adam with $v_{0,rnd}$, we visualize the loss landscapes around the convergent points of Transformer models trained with Vanilla Adam and Adam $v_{0,rnd}$ on the IWSLT'14 DE-EN task. The landscapes, plotted along two normalized random directions, are shown in Figure 2. Adam $v_{0,rnd}$ produces a flatter loss landscape compared to Vanilla Adam, which is often associated with better generalization performance [9, 46]. Despite similar training losses, the flatter landscape explains Adam $v_{0,rnd}$'s s superior testing accuracy.

4. Conclusion

In this work, we revisited the initial steps of adaptive gradient optimization methods, focusing on the instability caused by the sign-descent behavior during early iterations. To address this issue, we proposed two simple yet effective approaches: data-driven initialization and random initialization of the second-moment estimate v_0. Our empirical results demonstrate that these initialization strategies significantly enhance the performance and stability of several adaptive gradient optimization methods, including Adam, particularly in challenging tasks such as training Transformer models.

References

[1] Ruslan Abdulkadirov, Pavel Lyakhov, and Nikolay Nagornov. Survey of optimization algorithms in modern neural networks. *Mathematics*, 11(11):2466, 2023.

[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] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM review*, 60(2):223–311, 2018.

[4] Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

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

[6] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[8] Ali Hassani, Steven Walton, Nikhil Shah, Abulikemu Abuduweili, Jiachen Li, and Humphrey Shi. Escaping the big data paradigm with compact transformers. *arXiv preprint arXiv:2104.05704*, 2021.

[9] Haowei He, Gao Huang, and Yang Yuan. Asymmetric valleys: Beyond sharp and flat local minima. *Advances in neural information processing systems*, 32, 2019.

[10] 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.

[11] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[12] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8):2, 2012.

[13] Sepp Hochreiter, J urgen Schmidhuber, and Corso Elvezia. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[14] Xiao Shi Huang, Felipe Perez, Jimmy Ba, and Maksims Volkovs. Improving transformer optimization through better initialization. In *International Conference on Machine Learning*, pages 4475–4483. PMLR, 2020.

[15] Kaiqi Jiang, Dhruv Malik, and Yuanzhi Li. How does adaptive optimization impact local neural network geometry? *Advances in Neural Information Processing Systems*, 36, 2024.

[16] Jean Kaddour, Oscar Key, Piotr Nawrot, Pasquale Minervini, and Matt J Kusner. No train no gain: Revisiting efficient training algorithms for transformer-based language models. *Advances in Neural Information Processing Systems*, 36, 2024.

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

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

[19] Frederik Kunstner, Jacques Chen, Jonathan Wilder Lavington, and Mark Schmidt. Noise is not the main factor behind the gap between sgd and adam on transformers, but sign descent might be. In *The Eleventh International Conference on Learning Representations*, 2023.

[20] Frederik Kunstner, Robin Yadav, Alan Milligan, Mark Schmidt, and Alberto Bietti. Heavy-tailed class imbalance and why adam outperforms gradient descent on language models. *arXiv preprint arXiv:2402.19449*, 2024.

[21] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31, 2018.

[22] Zhiyuan Li, Sadhika Malladi, and Sanjeev Arora. On the validity of modeling sgd with stochastic differential equations (sdes). *Advances in Neural Information Processing Systems*, 34:12712–12725, 2021.

[23] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. In *International Conference on Learning Representations*, 2020.

[24] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.

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

[26] Liangchen Luo, Yuanhao Xiong, Yan Liu, and Xu Sun. Adaptive gradient methods with dynamic bound of learning rate. In *International Conference on Learning Representations*, 2018.

[27] Jerry Ma and Denis Yarats. On the adequacy of untuned warmup for adaptive optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8828–8836, 2021.

[28] Sadhika Malladi, Kaifeng Lyu, Abhishek Panigrahi, and Sanjeev Arora. On the sdes and scaling rules for adaptive gradient algorithms. *Advances in Neural Information Processing Systems*, 35:7697–7711, 2022.

[29] Mitch Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

[30] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

[31] Yan Pan and Yuanzhi Li. Toward understanding why adam converges faster than sgd for transformers. *arXiv preprint arXiv:2306.00204*, 2023.

[32] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[33] R Pascanu. On the difficulty of training recurrent neural networks. *arXiv preprint arXiv:1211.5063*, 2013.

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

[35] Alec Radford. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[36] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018.

[37] Mohamed Reyad, Amany M Sarhan, and Mohammad Arafa. A modified adam algorithm for deep neural network optimization. *Neural Computing and Applications*, 35(23):17095–17112, 2023.

[38] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[39] 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.

[40] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[41] Yuandong Tian, Yiping Wang, Beidi Chen, and Simon S Du. Scan and snap: Understanding training dynamics and token composition in 1-layer transformer. *Advances in Neural Information Processing Systems*, 36:71911–71947, 2023.

[42] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[43] Yizhou Wang, Yue Kang, Can Qin, Huan Wang, Yi Xu, Yulun Zhang, and Yun Fu. Momentum is all you need for data-driven adaptive optimization. In *2023 IEEE International Conference on Data Mining (ICDM)*, pages 1385–1390. IEEE, 2023.

[44] Robin Yadav, Frederik Kunstner, Mark Schmidt, and Alberto Bietti. Why adam outperforms gradient descent on language models: A heavy-tailed class imbalance problem. In *OPT 2023: Optimization for Machine Learning*.

[45] Dongruo Zhou, Jinghui Chen, Yuan Cao, Yiqi Tang, Ziyan Yang, and Quanquan Gu. On the convergence of adaptive gradient methods for nonconvex optimization. *arXiv preprint arXiv:1808.05671*, 2018.

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

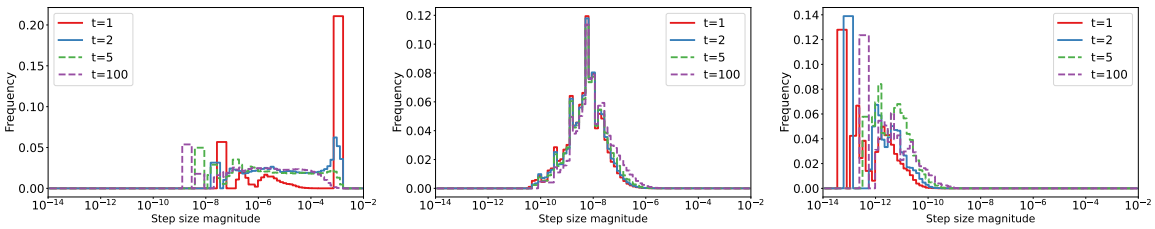
[47] Juntang Zhuang, Tommy Tang, Yifan Ding, Sekhar C Tatikonda, Nicha Dvornek, Xenophon Papademetris, and James Duncan. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. *Advances in neural information processing systems*, 33:18795–18806, 2020.

Appendix A. Additional Details about Second-order Moment Initialization

A.1. Impact of sign descent and shrinking gradients

In this section, we analyze the non-convergence behavior of vanilla Adam, focusing on the large initial step sizes observed during neural network training. Neural networks often exhibit a flat loss landscape at the beginning of training, with gradients that are small in magnitude. This phenomenon is particularly pronounced when training Transformers, as noted in prior works [14, 33, 41]. The initial loss landscape of the Transformer model is visualized in Figure 1(*c*), where the loss is plotted along two random directions as described in [21]. The visualization highlights that the loss landscape is extremely flat, and gradients are correspondingly small. When training such networks with Adam, the "sign descent" behavior during the initial step can amplify these small gradients disproportionately, resulting in overly large parameter updates. To further investigate this phenomenon, Figure 1(*b*) illustrates the norm of the update step $\|\Delta\theta_t\|$ during training for three optimizers: SGD, vanilla Adam, and Adam with the proposed initialization $v_{0,data}$. The results show that the first update step size for vanilla Adam $v_{0,0}$ is significantly larger compared to Adam $v_{0,data}$ or SGD. These large initial updates can push the optimizer away from initial regions in the parameter space, making recovery and convergence more challenging. In contrast, SGD exhibits much smaller update steps during the initial stages, even when using a larger learning rate (lr=0.1) than Adam (lr=0.001) in our experiments.

Figure 1(*b*) shows that the first update step size for vanilla Adam $v_{0,0}$ is significantly larger compared to Adam $v_{0,data}$ or SGD. To further illustrate the update step sizes, Figure 3 presents histograms of the absolute values of parameter updates for different optimizers. For vanilla Adam (Figure 3(*a*)), many parameters are updated with a step size equal to the learning rate in the first step $(t = 1)$ due to its "sign descent" behavior. Subsequently, the update step sizes decrease. In contrast, Adam with non-zero initialization (Figure 3(*b*)) achieves relatively stable update step sizes throughout training, avoiding the large initial jumps seen in vanilla Adam. This behavior aligns closely with SGD (Figure 3(*c*)), which consistently maintains stability in its updates from the start.



(*a*) Vanilla Adam $v_{0,0}$ (*b*) Adam with $v_{0,data}$ (*c*) SGD

Figure 3: Histogram of update step distribution across coordinates.

A.2. Linear Loss

To simplify the analysis, we consider the RMSprop update rule (ignoring ϵ) for a linear loss. The update for the parameter θ_t can be expressed as:

$$\mathbf{E}[\Delta\theta_t] = -\alpha\mathbf{E}\left[\frac{g_t}{\sqrt{v_t}}\right] \tag{9}$$

Using a Taylor expansion of $1/\sqrt{v_t}$ around $\mathbf{E}[v_t]$, we approximate:

$$\frac{1}{\sqrt{v_t}} \approx= \frac{1}{\mathbf{E}[v_t]} - \frac{1}{2\mathbf{E}[v_t]^{\frac{3}{2}}}(v_t - \mathbf{E}[v_t]) \tag{10}$$

Substituting this into the expectation, we have:

$$\mathbf{E}[\Delta\theta_t] \approx -\alpha \left(\frac{\mathbf{E}[g_t]}{\sqrt{\mathbf{E}[v_t]}} - \frac{\mathbf{E}[g_t(v_t - \mathbf{E}[v_t])]}{2\mathbf{E}[v_t]^{\frac{3}{2}}} \right) \tag{11}$$

Considering $\mathbf{E}[g_t] = \bar{g}$, and that g_t and $v_t - \mathbf{E}[v_t]$ are uncorrelated, we have: $\mathbf{E}[g_t(v_t - \mathbf{E}[v_t])] = \mathbf{E}[g_t] \cdot \mathbf{E}[v_t - \mathbf{E}[v_t]] = 0$. This simplifies the expression to:

$$\mathbf{E}[\Delta\theta_t] \approx -\alpha \frac{\bar{g}}{\sqrt{\mathbf{E}[v_t]}} \tag{12}$$

$$\approx -\alpha \frac{\bar{g}}{\sqrt{\beta_2^t v_0 + (1 - \beta_2^t)(\bar{g}^2 + \sigma^2 I)}} \tag{13}$$

Case 1: vanilla Adam ($v_0 = 0$). When $v_0 = 0$, the update becomes:

$$\mathbf{E}[\Delta\theta_t] \approx -\alpha \frac{\bar{g}}{\sqrt{(1 - \beta_2^t)(\bar{g}^2 + \sigma^2 I)}} \tag{14}$$

In this setting, the denominator is initially small due to $(1 - \beta_2^t)$ approaching 0 as $t \to 0$. The small denominator leads to excessively large initial updates, particularly when $\bar{g}$ is small or σ^2 is large. This instability can cause erratic optimization behavior, especially in the early stages of training.

Case 2: non-zero initialization ($v_0 = \bar{g}^2 + \sigma^2 I$). When $v_0 = \bar{g}^2 + \sigma^2 I$, the update becomes:

$$\mathbf{E}[\Delta\theta_t] \approx -\alpha \frac{\bar{g}}{\sqrt{\bar{g}^2 + \sigma^2}}. \tag{15}$$

In this setting, the denominator is well-scaled from the start, incorporating the correct statistical variance. This prevents excessively large updates during early iterations, ensuring better stability. The step sizes remain consistent across iterations, aligning with the principles of adaptive gradient methods. Additionally, the incorporation of gradient statistics $\bar{g}^2 + \sigma^2 I$ ensures that v_t adapts appropriately to the local geometry of the loss function, such as the Hessian information. For a linear loss, this stabilization leads to smoother convergence, providing a more robust optimization process. It is worth noting that the above analysis can be readily extended to other adaptive gradient methods, such as Adam.

A.3. Revisiting Previous Works on Stabilizing the Initial Steps of Adam

Warmup. The warmup technique [27, 42] implicitly adjusts the initialization of the second-moment estimate v by employing a smaller learning rate during the initial steps. While the optimizer's state updates normally, the parameter changes are minimal due to the extremely small learning rate. This approach effectively mitigates the sign-descent behavior observed in Adam's early steps. However, warmup introduces additional hyperparameters (e.g., the scheduler) that require careful tuning and necessitates several steps of training where the network parameters are not effectively updated. This

can be inefficient, particularly in resource-constrained settings. In contrast, our method directly addresses the aggressive sign-descent issue by initializing v_0 with non-zero values, eliminating the need for a warmup phase. Our experimental results demonstrate that random initialization of v_0 stabilizes the training process effectively, without requiring extra tuning or wasted iterations.

RAdam. RAdam [23] avoids the sign-descent issue by behaving like SGD [27] during the initial steps. This is achieved by introducing a rectification term, dynamically adjusting the optimizer's behavior to stabilize updates in the early iterations. While RAdam successfully addresses initial-step instability, it adds complexity to the optimization process through the computation of the rectification term. In contrast, our approach provides a simpler and more intuitive solution by directly adjusting the initialization of the moment estimates, without modifying the core algorithm or introducing additional dynamic terms.

AdaBound. AdaBound [26] tightly bounds the update size during the initial steps, preventing excessively large updates caused by sign-descent behavior. However, this approach introduces dynamic bounds that require careful tuning of the bounding functions, adding additional complexity to the optimization process. Our initialization strategy simplifies this issue by stabilizing updates without the need for dynamic bounds, making it a more efficient and practical alternative.

AdaBelief. AdaBelief [47] reduces the impact of initial sign-descent behavior by refining the variance estimation, leading to more reliable adaptive learning rates. However, this comes at the cost of increased computational complexity due to the need for precise variance estimation. By contrast, our method provides stability during the initial steps without additional computational overhead, offering a straightforward alternative to improve early optimization dynamics.

Our initialization strategy can be seamlessly integrated into existing methods, such as RMSprop, AdamW, RAdam, AdaBound, AdaBelief, and even Warmup. By addressing the aggressive sign-descent behavior directly through non-zero initialization of v_0, we enhance the stability of these optimizers in their early steps. Importantly, this random initialization incurs no extra computational costs and avoids the need for additional hyperparameter tuning.

Appendix B. Additional Details of Experiments

B.1. Experimental Setting

We empirically evaluate the performance of the proposed data-driven initialization (Equation (7)) and random initialization (Equation (8)) strategies across several widely-used adaptive gradient optimization methods. These include SGD with momentum (SGDM) [34, 38], Adam [17], AdamW [25], AdaBound [26], RAdam [23], and AdaBelief [47]. Each optimizer is tested using its standard initialization ($v_0 = 0$) as the baseline, which is then compared against the proposed strategies $v_{0,data}$ and $v_{0,rnd}$. Following experimental protocols established in prior works [23, 43, 47], we perform thorough hyperparameter tuning for learning rate, β_1, β_2, and ϵ. To ensure statistical robustness, each experiment is repeated with five random seeds, and we report the mean results along with standard deviations. For data-driven initialization, gradient statistics are computed using 5,000 random samples prior to training, with the scaling factor set to $\sigma = 1$. For random initialization, the scaling factor is set to $\sigma = 100$, demonstrating the tuning-friendly nature of the proposed approach.

Image Classification with CNN. We evaluate the ResNet-34 [10] architecture on the CIFAR-10 image classification dataset [18]. Each model is trained for 200 epochs with a batch size of 128, and the learning rate is decayed by a factor of 0.2 at epochs 60, 120, and 160. Label smoothing [40] with a smoothing factor of 0.1 is applied. In addition to CIFAR-10, we perform experiments on the

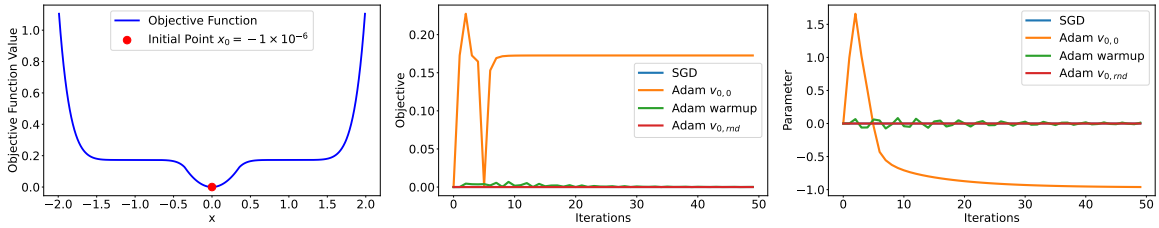
ImageNet ILSVRC 2012 dataset [39] using ResNet-18 as the backbone network. Each optimizer is executed for 100 epochs with a cosine annealing learning rate schedule, which has demonstrated superior performance compared to step-based decay strategies [24]. For SGD, we use the momentum factor of 0.9, a common default setting [10], with a tuned learning rate of 0.1. For adaptive gradient methods (Adam, AdamW, RAdam, AdaBound, AdaBelief), we use the learning rate of 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$.

Language Modeling with LSTM. We evaluate a 2-layer LSTM [13] on the Penn Treebank dataset [29]. Models are trained for 200 epochs with a batch size of 20, and the learning rate is reduced by a factor of 0.1 at epochs 100 and 145. For SGD, we use a learning rate of 30 and a momentum factor of 0.9. Adam, AdamW, AdaBound, and AdaBelief use a learning rate of 0.01, while RAdam uses a learning rate of 0.001. All adaptive methods are configured with $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

Neural Machine Translation with Transformer. We experiment with a small Transformer model [42] implemented using the Fairseq package [30] on the IWSLT'14 German-to-English machine translation dataset. The model is trained with a length penalty of 1.0, a beam size of 5, and an initial warmup step size of 10^{-7}. Training is conducted for 55 epochs, and results are reported as the average of the last 5 checkpoints. Adaptive learning methods use a learning rate of 0.0015. Adam, AdamW, AdaBound, and AdaBelief are configured with $\beta_1 = 0.9$, $\beta_2 = 0.98$, while RAdam uses $\beta_1 = 0.9$, $\beta_2 = 0.999$.

Image Generation with GAN. We evaluate a deep convolutional GAN (DCGAN) [35] on the CIFAR-10 image generation task. Both the generator and discriminator networks use CNN architectures. Models are trained for 200,000 iterations with a batch size of 64. Learning rate is fixed at 0.0002 for both the generator and discriminator across all optimizers. All other hyperparameters are set to their default values for fair comparison.

B.2. Toy Experiments of Adam's Instability and Initialization



(*a*) Saddle objective function (*b*) Objective value vs. iterations (*c*) Parameter vs. iterations

Figure 4: Optimization of the saddle objective function with different methods.

We conduct a toy experiment to illustrate the instability of Adam with its standard zero initialization and the effectiveness of our proposed non-zero initialization. For this demonstration, we use the random initialization strategy $v_{0,rnd}$. The objective function is a non-convex saddle function:

$$f(x) = \begin{cases} (x-b)^n, & \text{if } x \geq x_s \\ -(x+b)^n, & \text{if } x \leq -x_s \\ x^2 + d, & \text{if } -x_s < x < x_s \end{cases} \tag{16}$$

Here x_s is a switch point, b is a bias and d is a shift ensuring smooth transition at the switch points.

$$x_s = \left(\frac{s}{n}\right)^{\frac{1}{n-1}} + b,\ d = (x_s - b)^n - x_s^2 \tag{17}$$

The parameter n represents the degree of the polynomial. In our experiment, we set $n = 7$, $b = 1$, and $s = 0.5$. The purpose of the experiment is to observe the optimization behavior under different initializations. We use the Adam optimizer with the following hyperparameters: $\alpha = 1, \beta_1 = 0.9, \beta_2 = 0.999$. For scenarios requiring smaller learning rates, the objective function can be scaled down to achieve similar conclusions.

The optimization process begins at an initial point $x_0 = -10^{-6}$, which is close to the true optimum $x^\star = 0$, as shown in Figure 4(*a*). Figure 4(*b*) and Figure 4(*c*) present the loss values and parameter convergence over iterations for the various methods. As observed, standard Adam with $v_0 = 0$ converges to a suboptimal local minimum around $x_\infty \approx -1$, significantly deviating from the true optimum. In contrast, Adam with the proposed non-zero initialization $v_{0,rnd}$ successfully converges to the true optimum. For comparison, both the SGD optimizer and Adam with a warmup strategy also converge to the true optimum, demonstrating their stability.

The final converged parameter values for each method are summarized in Table 5. These results highlight that the proposed method achieves the lowest loss among all optimization techniques, underscoring its effectiveness in handling this optimization task.

Table 5: Final converged parameter values for different optimization methods.

Adam ($v_{0,0}$)	**Adam (warmup)**	**Adam** ($v_{0,\mathbf{rnd}}$)	**SGD**
-0.96	0.01	1×10^{-7}	9×10^{-7}

B.3. Image Classification on Imagenet dataset

To further validate the effectiveness of our algorithm on a more comprehensive dataset, we conducted experiments on the ImageNet dataset [39], utilizing ResNet-18 as the backbone network. As shown in Table 6, both $v_{0,rnd}$ and $v_{0,data}$ provide significant performance gains across several adaptive gradient optimization methods.

Table 6: Test accuracy ↑ (%) of ResNet-18 on ImageNet dataset.

Optimization	SGD	Adam	AdamW	AdaBound	RAdam	AdaBelief
Vanilla $v_{0,0}$	70.23±0.07	63.79±0.12	67.93±0.12	68.13±0.11	67.62±0.11	70.08±0.10
$v_{0,rnd}$	-	65.99±0.11	68.95±0.11	68.80±0.11	68.83±0.11	70.69±0.10
$v_{0,data}$	-	66.13±0.11	68.49±0.11	68.96±0.11	68.99±0.11	**70.77±0.10**

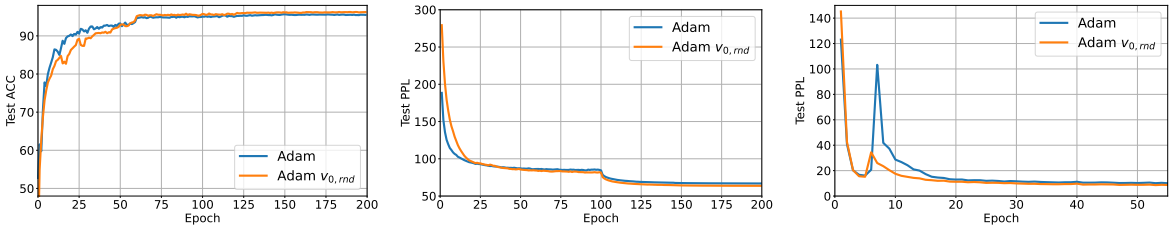
B.4. Language Modeling with 3-Layer LSTM

We evaluate a 3-layer LSTM network on the Penn Treebank dataset [29]. The test perplexity results are summarized in Table 7. Similar to the findings with the 2-layer LSTM, the proposed initialization strategies provide additional performance gains for adaptive gradient optimization methods.

Table 7: Test perplexity ↓ of 3 Layer LSTM on Penn Treebank dataset dataset.

Optimization	SGD	Adam	AdamW	AdaBound	RAdam	AdaBelief
Vanilla $v_{0,0}$	63.52±0.16	64.10±0.25	69.91±0.20	63.52±0.11	70.10±0.16	61.33±0.19
$v_{0,rnd}$	-	62.68±0.19	66.43±0.18	62.75±0.11	68.05±0.16	61.29±0.15
$v_{0,data}$	-	62.46±0.20	66.38±0.18	62.07±0.11	68.14±0.16	**60.70±0.14**

B.5. Training curve



(*a*) CIFAR-10 Test Accuracy (*b*) PTB Test Perplexity (*c*) IWSTL'14 Test Perplexity

Figure 5: Comparison of Vanilla Adam and Adam $v_{0,rnd}$ on (a) CIFAR-10 image classification task. (b) Penn Treebank language modeling task. (c) IWSTL'14 machine translation task.

We compare the training curves of Vanilla Adam and Adam with random initialization $v_{0,rnd}$, as it is more computationally efficient [2]. In the CIFAR-10 image classification task in Figure 5(*a*), while Adam $v_{0,rnd}$ exhibits slightly lower accuracy in the initial steps, it achieves more stable convergence and higher final accuracy. For the Penn Treebank language modeling task in Figure 5(*b*), Adam $v_{0,rnd}$ results in lower perplexity at convergence compared to Vanilla Adam. For Transformer models on the IWSLT'14 DE-EN machine translation dataset (with warmup) in Figure 5(*c*), Adam $v_{0,rnd}$ demonstrates faster convergence, more stable optimization, and lower perplexity at the end of training.

B.6. Ablation Study

The scaling factor σ is a key hyperparameter in the proposed initialization method Equations (7) and (8). To evaluate the impact of σ,we conducted an ablation study on the CIFAR-10 image classification task, as summarized in Table 8. The results show that for a wide range of σ values, such as $\sigma \in [1, 1000]$, the performance consistently outperforms zero initialization. This highlights the robustness and tuning-friendly nature of the proposed approach, as it achieves stable improvements across different σ settings.

Table 8: Impact of σ on CIFAR-10 Test Accuracy.

σ	0	0.1	1	10	100	1000
$v_{0,rnd}$	95.25	95.45	95.74	**95.89**	95.87	95.84
$v_{0,data}$	95.25	95.70	**96.02**	95.92	95.85	95.72

2. We note that, The training behavior of $v_{0,data}$ is similar to $v_{0,rnd}$.