AdaR: An Adaptive Gradient Method With Cyclical Restarting of Moment Estimations

Yangchuan Wang[1] , **Lianhong Ding**[1] and **Peng Shi** [2] ✉
[1]Beijing Wuzi University
[2]University of Science and Technology Beijing
wyc2828@139.com, lhdingbwu@sina.com, pshi@ustb.edu.com

Abstract

Adaptive gradient methods, primarily based on Adam, are prevalent in training neural networks, adjusting step sizes via exponentially decaying averages of gradients and squared gradients. Adam assigns small weights to distant gradients, termed long-tail gradients in this paper. However, these gradients persistently influence update behavior, potentially degrading generalization performance. To address this issue, we incorporate a restart mechanism into moment estimations, proposing AdaR (ADAptive gradient methods via Restarting moment estimations). Specifically, AdaR divides a training epoch into fixed-iteration intervals, alternating between two sets of moment estimations for parameter updates and discarding prior moment estimations at the beginning of each interval. Within each interval, one set updates parameters and will be discarded in the subsequent interval, while the other is reset at the midpoint to estimate moments for updates in the subsequent interval. The restart mechanism cyclically discards distant gradients, initiates fresh moment estimations for parameter updates, and stabilizes training. By prioritizing recent gradients, the method increases estimation accuracy and enhances step size adjustment. Empirically, AdaR outperforms state-of-the-art optimization algorithms on image classification and language modeling tasks, demonstrating superior generalization and faster convergence.

1 Introduction

Gradient descent methods are effectiveness and simplicity in neural network training [He *et al.*, 2016; Huang *et al.*, 2017; Merity *et al.*, 2018; Vaswani *et al.*, 2017]. Stochastic gradient descent (SGD) selects a random subset of samples and updates parameters in the opposite gradient direction with a predefined step size [Robbins and Monro, 1951]. Despite its superior efficiency and generalization, SGD faces a computational burden from trial-and-error tuning, particularly in deep networks with error surfaces containing flat basins and sharp valleys. While small step sizes reduce oscillations in valleys but slow training in flat basins, large step sizes accelerate convergence in basins but potentially induce oscillations in sharp valleys [Zeiler, 2012].

Neural network training requires varying step sizes across coordinates, as error surfaces have dimension-specific properties. Recent studies propose various gradient methods with adaptive step sizes [Duchi *et al.*, 2011; Kingma and Ba, 2014]. AdaGrad scales per-dimension step sizes by average of squared gradients [Duchi *et al.*, 2011]. Although AdaGrad effectively captures rare features and performs well in sparse settings, its step sizes rapidly decay due to the accumulation of all past gradients. Adam (Adaptive moment estimation) adjusts per-dimension step sizes using exponential moving averages (EMA) of gradients and squared gradients to estimate first- and second-order moments, accelerating convergence [Kingma and Ba, 2014]. Adam variants exponentially decay past gradients [Zhou *et al.*, 2020; Luo *et al.*, 2019]. However, they assign non-trivial weights to distant gradients, leading to heavy-tailed gradient noise. Heavy-tailed distributions, with a higher likelihood of extreme gradients, can skew statistical distributions by amplifying outliers, leading to biased estimations and extreme step sizes in Adam-type algorithms [Zhang *et al.*, 2020; Das *et al.*, 2024]. In this paper, distant gradients persistently influence moment estimations and step size adjustment, referred to as long-tail gradients. This results in the distortion of the gradient moments, thereby impairing the prioritization of recent gradients and leading to suboptimal step size adjustments. Parameters with small gradients may receive inadequate updates, while those with large gradients may receive excessive updates. This misalignment hinders convergence performance, as biased moment estimations prevent precise step size adjustments, potentially causing oscillations or divergence. Most Adam variants fail to address this problem, resulting in suboptimal performance.

This paper proposes AdaR (ADAptive gradient methods via Restarting moment estimations), which cyclically restarts the first- and second-order moments to discard outdated gradient information and mitigate long-tail gradients. Specifically, each training epoch is divided into multiple fixed-iteration intervals, with parameters updated alternately using two sets of moment estimations. During each interval, one set updates parameters and is discarded in the subsequent interval, while

✉ Corresponding author: Peng Shi (pshi@ustb.edu.com)

the other restarts and accumulates gradients for updates in subsequent interval. In subsequent interval, the roles swap: moment estimations used in previous intervals are discarded, and the other updates parameters. The restart mechanism mitigates the influence of outdated gradients and assigns higher weight to up-to-date gradients. AdaR periodically restarts first- and second-order moment estimations to enhances step size adaptation and stabilize training. The method offers an orthogonal alternative to address long-tail gradients in existing Adam variants. Our contributions are threefold and summarized below.

- We examine the adverse effects of long-tail gradients on parameter updates. To address this, we partition the training epoch into multiple fixed-iteration intervals and incorporate a restart technique that discards distant gradients and accumulates the most recent gradients.
- We propose a novel method AdaR, which cyclically restarts moment estimations and discards distant gradients. The method effectively mitigates adverse impact of long-tail gradients, emphasizes recent gradients, and enhances step size adaptation, thereby improving generalization performance.
- We conduct experiments on image classification and language modeling tasks. Experiment results demonstrate that AdaR generalizes significantly better than popular Adam variants and SGD with fast convergence.

This study addresses long-tail gradients via a restart mechanism and proposes AdaR. The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 details the approach. Section 4 presents experiments. Section 5 provides an ablation study. Section 6 concludes this paper.

2 Related Work

Adam variants estimate the first- and second-order moments by exponentially decaying past gradients. However, the residual influence of remote gradients continues to affect step size adjustment. Zhou *et al.* [2020] indicated that the gradient noise in Adam follows a heavy-tailed symmetric α-stable ($S\alpha S$) distribution, resulting in step sizes that may poorly reflect the local landscape. To address this, AdaBound dynamically applies clipping operation to impose upper and lower bounds on step sizes, mitigating extreme values [Luo *et al.*, 2019]. It ensures a smooth transition from Adam to SGD as adaptive step sizes converge to a fixed value. ACClip mitigates long-tail gradients through coordinate-wise clipping [Zhang *et al.*, 2020]; however, its heuristic threshold introduces substantial computational cost. Since the influence of g_{t-n} persists across time steps, AdaShift decorrelates moments by temporally shifting gradients to mitigate the exponential moving effect of m_t [Zhou *et al.*, 2019]. However, it suffers from poor empirical performance. Adam_Win minimizes a dynamic loss using a proximal point method (PPM) regularizer, increasing optimization convexity [Zhou *et al.*, 2023]. However, it adds substantial tuning complexity and performs poorly in language modeling tasks, as shown in Subection 4.5. AdaPlus [Guan, 2024] leverages the geometry of the loss landscape to adjust step sizes, combining AdaBelief [Zhuang *et al.*, 2020], AdamW [Loshchilov and Hutter, 2019], and Nesterov [Nesterov, 1983]. However, it struggles with large-scale experiments, where increased computational complexity poses significant challenges.

AAdam incorporates a fraction of past updates into recent updates to track parameter changes [Tato and Nkambou, 2018], but may introduce gradient noise from the sign operation. AdamP projects the weight norm to control its growth, thus slowing the decay of step sizes [Heo *et al.*, 2021]. However, the projection operation incurs non-trivial computational overhead. HN_Adam adaptively switches between Adam and AMSGrad based on the dynamic threshold, increasing computational complexity and memory usage [Reyad *et al.*, 2023]. Moreover, AMSGrad performing second-moment maximization rapidly decays step sizes, potentially slowing convergence. ACGB-Adam employs the adaptive deviation-gradient coefficient and random block coordinate to control update direction and magnitude, respectively [Liu *et al.*, 2023], but is evaluated solely on simple MNIST and CIFAR-10 datasets.

The restart technique has been employed to schedule step sizes [Loshchilov and Hutter, 2017]. To our best knowledge, this is the first application of the restart in moment estimations of Adam to address long-tail gradients. By periodically restarting the first- and second-order moment estimations, the method discards distant gradients, emphasizes recent gradients, and enhances step size adjustment.

3 Methodology

To address the long-tail gradient issue, this section provides the detail of AdaR, with its pseudocode presented in Algorithm 1. The method divides each training epoch into fixed-iteration intervals, where it restarts moment estimations and accumulates the most recent gradients. The method updates parameters by alternately using two sets of moment estimations. Within each interval, one set is active for updates throughout the interval, while the other restarts at the midpoint and remains inactive. The inactive set begins accumulating gradients at the midpoint of the interval and will be used for parameter updates in the subsequent interval. In the subsequent interval, the previously inactive set with recent gradients updates parameters throughout, while the previously active set restarts at the midpoint, then accumulating gradients. The method ensures precise adaptation to the most recent gradients, wisely adjusting step sizes.

As depicted in Figure 1, the method alternates between two sets of moment estimations for parameter updates, represented by distinct colors. Bold arrows represent the active set used for parameter updates, while narrow arrows represent the inactive set, preparing for updates in the subsequent interval. Within each interval, a single set of moment estimations is active for parameter updates. At the start of each interval, the roles of two sets are swapped; the active set with the most recent gradients is used to update parameter. At the midpoint, the active set continues updating parameters, while the inactive set restarts and accumulates recent gradients. In the subsequent interval, the accumulated moments with recent gradients update parameters, while the previously active

set restarts at the midpoint of the interval and accumulates recent gradients to estimate moments. This enhances step size adaptation to recent gradients and stabilizes training.

Consider two sets of moment estimations, Set A and Set B. In the first interval, both sets are initialized to zero, with Set A updating parameters throughout the interval. At the midpoint, Set B restarts and accumulates moment estimations for updates in the subsequent interval, with Set A updating parameters. In the second interval, Set B with recent gradients updates parameters throughout, while Set A restarts at the interval midpoint and accumulates gradients. In the third interval, Set A updates parameters throughout the interval, with Set B is reset at the midpoint to prepare for updates in the subsequent interval. The restart mechanism ensures continuous adaptation to the most recent gradients throughout the training.

In Algorithm 1, the length of interval is denoted as the number of iterations λ, with the midpoint at $\lambda/2$. As $\lambda \to \infty$, AdaR reduces to vanilla Adam. The method alternates between two sets of moment estimations at intervals. The *flag* variable tells the active and inactive sets. The active set updates parameters throughout the interval, while the inactive set restarts at the midpoint and accumulates recent gradients to estimate moments. In each interval, specifically, AdaR updates parameters using the active set throughout the interval. At the midpoint ($\lambda/2$), the inactive set restarts and accumulates recent gradients to estimate moments. In the subsequent interval, these accumulated moments update parameter, while previously active moments are discarded to ensure alignment with recent gradients. Step size clipping, implemented as $\text{clip}(\eta_t, \alpha_{low}, \alpha_{high})$, works well in practice [Zhang *et al.*, 2019], and the step size of each update is limited within the bound $[\alpha_{low}, \alpha_{high}]$, say, [0.0001, 0.5].

The core innovation of AdaR is the moment restart, which periodically restarts moment estimations to discards distant gradients and prioritize recent gradients. Note that AdaR is compatible with existing Adam variants. The interval length was empirically selected via a coarse- then fine-grained search, with 12 recommended as default. AdaR is robust to varying interval lengths, as demonstrated in the Subsection 5.1. Except for the interval length, AdaR introduces no new parameters. Theoretically, AdaR only adjusts the sliding window size of gradient accumulation while preserving the update rule of Adam. Thus, we conjecture that it retains the same convergence guarantee as Adam. A theoretical analysis is a promising avenue for future research.

Luo *et al.* [2019] noted that extreme step sizes impair the generalization of adaptive gradient methods, proposing AdaBound as a solution. AdaBound uses dynamic bounds to truncate extreme step sizes. However, the heuristic bounds may lack performance consistency across tasks. Figure 2 illustrates that step sizes of AdaBound, Adam, and AdaR are initially large due to limited samples, followed by a reduction. As training progresses, the accumulation of past gradients exacerbates the long-tail gradient issue, resulting in biased moment estimations and step size adjustments. The step sizes exhibit significant oscillations in Adam. To mitigate oscillations, AdaBound adopts heuristic functions to truncate step sizes, which gradually converge to a small value. However,

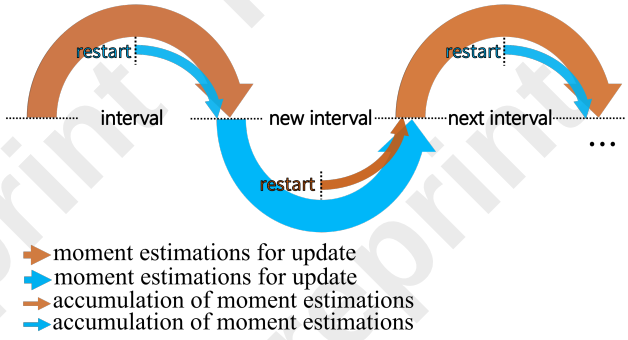

Figure 1: The moment restart of AdaR. Without loss of generality, orange arrows denote Set A and blue arrows denote Set B. AdaR alternates between Set A and Set B at intervals. At the start of each interval, the roles of Set A and Set B are swapped. At the midpoint of interval, the Set not used for parameter updates is reset to zero.

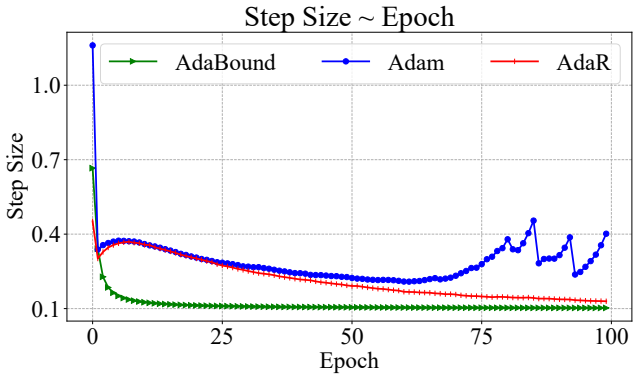

Figure 2: Step sizes of AdaBound, Adam, and AdaR with ResNet-34 model on the CIFAR-10 dataset.

heuristic functions limit the adaptability of step sizes [Wang *et al.*, 2025]. Notely, AdaR initially exhibits large step sizes, enabling rapid convergence, but gradually reduces step sizes via restart mechanism to mitigate oscillations.

4 Experiment

We evaluate AdaR on image classification and language modeling tasks. Image classification tasks involve convex and non-convex problems. For convex optimization, we train logistic regression model [LaValley, 2008] on the MNIST dataset [LeCun *et al.*, 1998]. For non-convex tasks, we train VggNet-11 [Simonyan and Zisserman, 2015], ResNet-34 [He *et al.*, 2016], and DenseNet-121 [Huang *et al.*, 2017] on the CIFAR-10 and CIFAR-100 datasets [Krizhevsky, 2009], as well as ResNet-18 [He *et al.*, 2016] on the Tiny-ImageNet dataset. In language modeling, we train 1-layer LSTM [Merity *et al.*, 2018] on the Penn Treebank (PTB) dataset [Marcus *et al.*, 1993] and Transformer [Vaswani *et al.*, 2017] on the Wikitext-2 dataset [Bojanowski *et al.*, 2017]. AdaR[1] is compared against Adam [Kingma and Ba, 2014], AdaBound [Luo *et al.*, 2019], AdaShift [Zhou *et al.*, 2019], NosAdam [Huang *et al.*, 2019], Adam_Win [Zhou *et al.*, 2023], AdaPlus [Guan, 2024], and SGD [Robbins

Algorithm 1 ADAptive gradient methods via Restarting moment estimations (AdaR)

Input: $x_1 \in \mathcal{F}$, step size α, β_1, β_2, interval length λ

Initialize time step: $t \leftarrow 1$
Initialize first and second moments: $m_0^{(1)} \leftarrow 0, v_0^{(1)} \leftarrow 0, m_0^{(2)} \leftarrow 0, v_0^{(2)} \leftarrow 0$
Initialize *flag*: *flag* = True
for $t = 1$ to N **do**
  $g_t \leftarrow \nabla f_t(x_t; \theta_t)$
  $m_t^{(1)} \leftarrow \beta_1 m_{t-1}^{(1)} + (1-\beta_1) g_t$
  $m_t^{(2)} \leftarrow \beta_1 m_{t-1}^{(2)} + (1-\beta_1) g_t$
  $v_t^{(1)} \leftarrow \beta_2 v_{t-1}^{(1)} + (1-\beta_2) g_t^2$
  $v_t^{(2)} \leftarrow \beta_2 v_{t-1}^{(2)} + (1-\beta_2) g_t^2$
  if *flag* = True **then**
    $\hat{m}_t \leftarrow \frac{m_t^{(1)}}{1-\beta_1^t}$ and $\hat{v}_t \leftarrow \frac{v_t^{(1)}}{1-\beta_2^t}$
  else
    $\hat{m}_t \leftarrow \frac{m_t^{(2)}}{1-\beta_1^t}$ and $\hat{v}_t \leftarrow \frac{v_t^{(2)}}{1-\beta_2^t}$
  end if
  $\eta_t \leftarrow \frac{\alpha}{\sqrt{\hat{v}_t}+\varepsilon}$
  $\tilde{\eta}_t \leftarrow \text{clip}(\eta_t, \alpha_{low}, \alpha_{high})$
  $\theta_{t+1} \leftarrow \theta_t - \tilde{\eta}_t \hat{m}_t$
  if $t \bmod \lambda/2 = 0$ **then**
    if $t \bmod \lambda = 0$ **then**
      $flag = \neg flag$
    end if
    if $t \bmod \lambda \neq 0 \wedge flag =$ True **then**
      $m_t^{(2)} \leftarrow 0$ and $v_t^{(2)} \leftarrow 0$
    else if $t \bmod \lambda \neq 0 \wedge flag =$ False **then**
      $m_t^{(1)} \leftarrow 0$ and $v_t^{(1)} \leftarrow 0$
    end if
  end if
end for

and Monro, 1951]. Each method is run 3 times, with the best result reported. All experiments are conducted on an NVIDIA RTX A4000 GPU (16 GB) and an AMD EPYC 7551P CPU, using Python 3.8 and PyTorch library [Paszke, 2019]. Dataset details are provided below.

MNIST consists of 60,000 training and 10,000 test samples of 28×28 grayscale images of handwritten digits.

CIFAR-10 comprises 50,000 training and 10,000 test 32×32 color images across 10 classes.

CIFAR-100 contains 50,000 training and 10,000 test 32×32 pixel images across 100 classes.

TinyImageNet contains 200 classes, each with 500 training, 25 validation, and 25 test images, resizing from 64×64 to 224×224 pixels.

Penn Treebank (PTB) comprises 0.93 million training, 0.073 million validation, and 0.082 million testing tokens.

Wikitext-2 includes 1.9 million training, 0.17 million validation, and 0.19 million testing tokens.

We conduct grid searches to select optimal parameters for all algorithms across tasks. The base step sizes for all algorithms and the final step size for AdaBound are selected from the range $\{1 \times 10^{-5}, 5 \times 10^{-5}, 1 \times 10^{-4}, \cdots, 1\}$. The parameter β_1 is selected from $\{0.9, 0.99\}$, and β_2 is chosen from $\{0.99, 0.999\}$. SGD tunes momentum from the set $\{0.1, 0.2, \cdots, 0.9\}$. For all image classification tasks, adaptive gradient methods use a step size of 0.001, $\beta_1 = 0.9$, and $\beta_2 = 0.999$, while SGD employs a step size of 0.1 and momentum of 0.9. For language modeling tasks, SGD utilizes step sizes of 30 for LSTM and 0.1 for Transformer. For LSTM, adaptive methods apply a step size of 0.01 with $\varepsilon = 10^{-16}$, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. For Transformer, a step size of 0.001, with $\beta_1 = 0.9$ and $\beta_2 = 0.999$, is adopted for all adaptive methods.

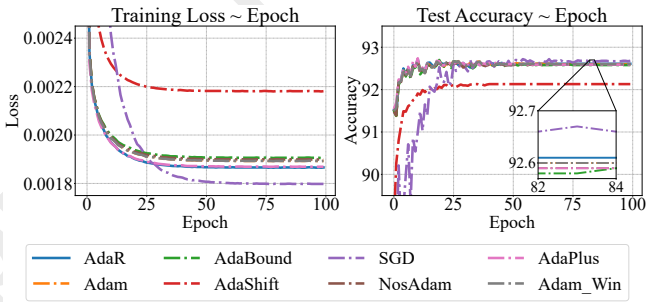

Figure 3: Training loss and test accuracy of logistic regression model on the MNIST dataset.

AdaR	92.61	AdaShift	92.13	AdaPlus	92.59
Adam	92.60	AdaBound	92.58	Adam_Win	92.60
SGD	**92.68**	NosAdam	92.59		

Table 1: Test accuracy of logistic regression model on the MNIST dataset.

4.1 Logistic Regression on the MNIST Dataset

We train logistic regression model [LaValley, 2008] on the MNIST dataset using a cross-entropy loss, providing a convex condition. A weight decay of 5×10^{-4} is applied to all algorithms. Each optimizer is executed for 100 epochs with a batch size of 128. Performance is evaluated by training loss and test accuracy, the latter primarily measuring generalization, as shown in Figure 3 and Table 1.

Figure 3 shows that AdaR achieves the highest test accuracy among adaptive methods, indicating superior generalization, although it slightly underperforms SGD. In contrast, AdaShift performs poorly, as evidenced by low test accuracy and slow convergence. In Table 1, AdaR outperforms Adam, AdaBound, AdaShift, NosAdam, AdaPlus, and Adam_Win in test accuracy by 0.01%, 0.03%, 0.48%, 0.02%, 0.02%, and 0.01%, respectively, while falling short of SGD by 0.07%. By emphasizing recent gradients to enhance step size adaptation, AdaR achieves superior generalization. The gap with SGD may reflect its effectiveness in convex tasks.

4.2 CNNs on the CIFAR-10/100 Datasets

We train VggNet-11 [Simonyan and Zisserman, 2015], ResNet-34 [He *et al.*, 2016], and DenseNet-121 [Huang

[1] Code at https://github.com/tHappo/AdaR

Optimizer	VggNet-11	ResNet-34	DenseNet-121
AdaR	**89.43**	**94.52**	**94.71**
Adam	87.98	93.62	93.57
AdaBound	87.85	92.49	93.24
AdaShift	84.33	92.06	93.84
SGD	89.20	93.24	94.11
NosAdam	86.02	93.83	93.93
AdaPlus	88.80	92.99	94.10
Adam_Win	87.36	93.73	93.76

Table 2: Test accuracy of VggNet-11, ResNet-34, and DenseNet-121, on the CIFAR-10 dataset.

Optimizer	VggNet-11	ResNet-34	DenseNet-121
AdaR	**64.15**	**75.85**	**77.38**
Adam	57.20	74.37	74.36
AdaBound	59.19	71.45	73.67
AdaShift	52.43	70.92	74.22
SGD	61.40	74.70	75.60
NosAdam	47.77	73.08	74.87
AdaPlus	61.88	72.57	75.79
Adam_Win	57.97	74.77	74.45

Table 3: Test accuracy of VggNet-11, ResNet-34, and DenseNet-121, on the CIFAR-100 dataset.

et al., 2017] on the CIFAR-10 and CIFAR-100 datasets. VggNet-11 uses 3×3 convolutions with max pooling. ResNet-34 adopts batch normalization and residual connections per layer. DenseNet-121 utilizes dense connections for feature extraction. Training samples are normalized by the mean (0.485, 0.456, 0.406) and standard deviation (0.229, 0.224, 0.225). AdaBound employs a final step size of 0.1. All models use a batch size of 128 and weight decay of 10^{-4}, trained for 100 epochs with step size decay by a factor of 10 at epochs 45 and 75. Performance curves for CIFAR-10 and CIFAR-100 are shown in Figures 4 and 5, with numerical results in Tables 2 and 3, respectively.

Figures 4 and 5 show AdaR achieving the highest test accuracy across all neural networks on both CIFAR-10 and CIFAR-100. Table 3 shows AdaR outperforming other methods on CIFAR-100 by 2.27%, 1.08%, and 1.59% for VggNet-11, ResNet-34, and DenseNet-121, respectively. Table 2 shows AdaR outperforming Adam/SGD on CIFAR-10 by 1.45/0.23, 0.9/1.28, and 1.14/0.6 (%) for the corresponding networks, respectively. AdaR performs on par with or even surpass SGD. The improvement stems from the mitigation of long-tail gradients and enhanced step size adaptation, better aligning with recent landscape. In contrast, NosAdam shows instability with VggNet-11 on CIFAR-10/100, probably due to excessive weighting of past gradients.

4.3 ResNet-18 on the TinyImageNet Dataset

We train ResNet-18 [He *et al.*, 2016] on the challenging TinyImageNet data. Training samples are augmented via random horizontal flips, random resized crops, and normalization using mean and standard deviation. All optimizers use a batch size of 256 and weight decay of 0.0005. We train all models for 90 epochs with step size decay at epochs 70 and 80. AdaBound employs a final step size of 0.1.

In Figure 6, AdaR achieves the highest test accuracy of 65.06%, outperforming Adam (57.96%), AdaBound (57.24%), NosAdam (57.94%), AdaPlus (57.32%), and Adam_Win (58.36%) by 7.10%, 7.82%, 7.12%, 7.74%, and 6.70%, respectively. Additionally, AdaR outperforms SGD (61.18%), the canonical algorithm for this dataset, by 3.88%, further demonstrating its superior generalization of this challenging benchmark.

Optimizer	Training perplexity	Test perplexity
AdaR	**58.74**	**83.32**
Adam	84.41	94.73
AdaBound	63.02	85.82
SGD	64.13	85.53
NosAdam	78.69	85.64
AdaPlus	79.10	89.97
Adam_Win	80.69	86.36

Table 4: Training and test PPL of LSTM on the PTB dataset (lower is better).

4.4 Long Short-Term Memory Model on the Penn TreeBank Dataset

We train LSTM model on the PTB dataset with default settings from Merity *et al.* [2018]. The model has a single layer with 1150 hidden units and 400-dimensional embeddings. Dropout is set as 0.3 for hidden layers, 0.65 for input embeddings, and 0.1 for the embedding layer. All models use a backpropagation through time (BPTT) length of 70, weight decay of 1.2×10^{-6}, batch size of 80, and gradient clipping at 0.25. We train all models for 200 epochs, with step size decay at epochs 100 and 145.

Figure 7 shows AdaR converging quickly and achieving the lowest training and test perplexities. Table 4 confirms that AdaR outperforms Adam, AdaBound, SGD, NosAdam, AdaPlus, and Adam_Win in training/test perplexity by 25.67/11.41, 4.28/2.50, 5.39/2.21, 19.95/2.32, 20.36/6.65, and 21.95/2.04, respectively. AdaR mitigates the long-tail gradient issue and improves generalization. In contrast, AdaShift stagnates after 100 epochs, probably due to temporal shift failing to find solutions for loss minimization.

4.5 Transformer on the WikiText-2 Dataset

We train a small Transformer with 2 self-attention heads, 200-dimensional embeddings, and 2 layers from Vaswani *et al.* [2017] on the WikiText-2 dataset. Dropout of 0.2 is applied, with a BPTT of 35, weight decay of 10^{-5}, batch size of 20, and gradient clipping at 0.25. Training spans 20 epochs, with step size decay by a factor of 10 at epoch 10.

Figure 8 shows AdaR achieving the lowest test perplexity with superior generalization, while Adam, NosAdam, and Adam_Win exhibit overfitting with high test perplexity. Table 5 shows that AdaR outperforms Adam, AdaBound, SGD, NosAdam, AdaPlus, and Adam_Win in test perplexity by 91.53, 39.33, 28.94, 143.41, 16.9, and 129.42, respectively.

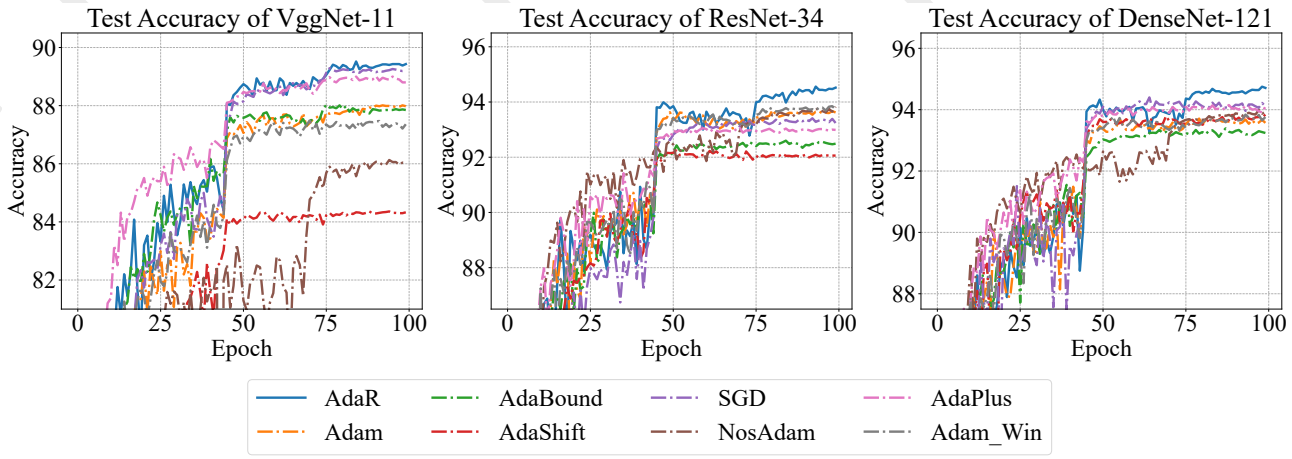

Figure 4: Test accuracy of VggNet-11, ResNet-34, and DenseNet-121 on the CIFAR-10 dataset.

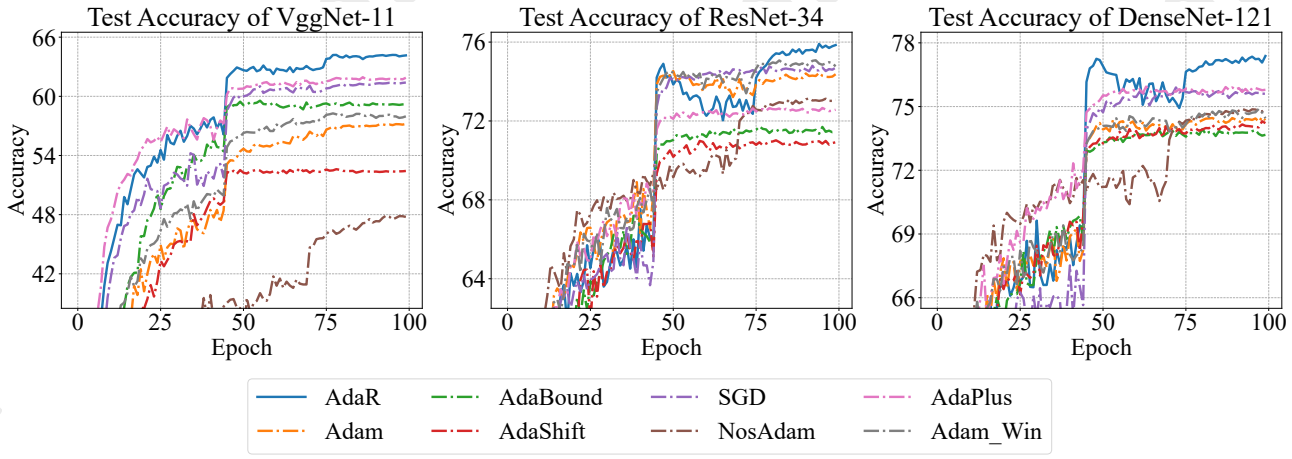

Figure 5: Test accuracy of VggNet-11, ResNet-34, and DenseNet-121 on the CIFAR-100 dataset.

Optimizer	Training perplexity	Test perplexity
AdaR	88.52	**155.94**
Adam	37.94	247.47
AdaBound	62.21	195.27
SGD	149.41	184.88
NosAdam	**33.78**	299.35
AdaPlus	108.42	172.84
Adam_Win	34.46	285.36

Table 5: Training and test PPL of Transformer on the WikiText-2 dataset (lower is better).

All results, across image classification and language modeling tasks, confirm the effectiveness of AdaR in mitigating long-tail gradients and improving generalization via restart mechanism.

5 Ablation Study

AdaR enhances efficacy and generalization through its restart mechanism, with ablation studies demonstrating its robustness to variations in interval length and step size parameters. ResNet-18 is trained on CIFAR-10 dataset for 100 epochs, with step size decay by a factor of 10 at epochs 45 and 75, using a batch size of 128 and weight decay of 10^{-4}. All algorithms use default parameters: $\varepsilon = 10^{-8}$, $\beta_1 = 0.9$, and $\beta_2 = 0.999$.

5.1 The Effect of the Interval Length

We evaluate AdaR with interval lengths of 12, 50 and 100. All methods employ a step size of 0.001. Figure 9 shows that AdaR achieves favorable performance across varying intervals, confirming the robustness to the interval length parameter, with minimal need for manual tuning. An interval length of 12 iterations is recommended.

5.2 The Effect of the Step Size

We evaluate AdaR with step sizes selected from the set $\{5\times10^{-4}, 10^{-3}, 5\times10^{-3}, 10^{-2}\}$. All algorithms are trained for 100 epochs. In Figure 10, AdaR shows favorable performance across a large range of step sizes, highlighting its robustness to the step size parameter.

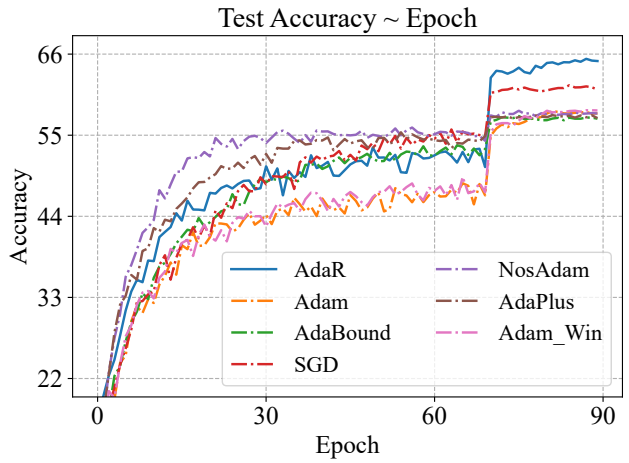

Figure 6: Test accuracy of ResNet-18 on the TinyImageNet dataset.

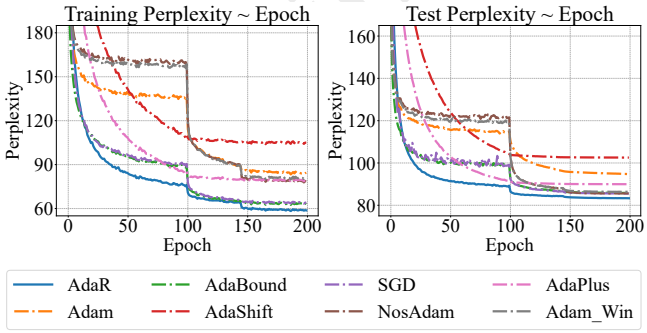

Figure 7: Training and test PPL of LSTM on the PTB dataset.

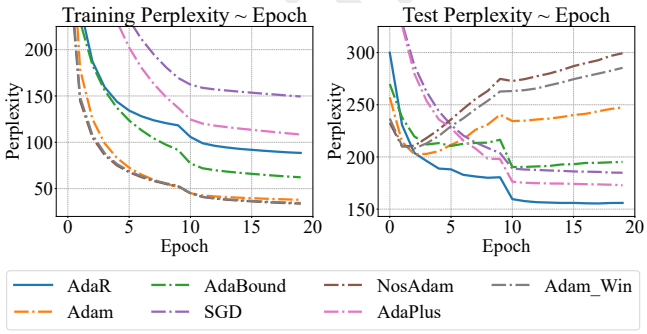

Figure 8: Training and test PPL of Transformer on the WikiText-2 dataset.

6 Conclusion

This paper presents AdaR, which mitigates long-tail gradients via a restart mechanism. By cyclically restarting moment estimations, AdaR effectively mitigates the influence of distant gradients, enhancing step size adjustment and improving convergence. Extensive experiments show that AdaR outperforms existing methods in generalization with strong robustness. We hope these findings could provide useful insights into adaptive gradient methods and develop more robust algorithms for the community.

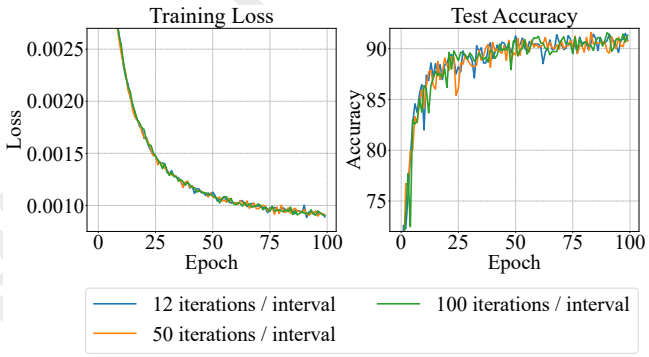

Figure 9: AdaR is applied to ResNet-18 on CIFAR-10 to ablate the effect of interval length, evaluated by training loss and test accuracy.

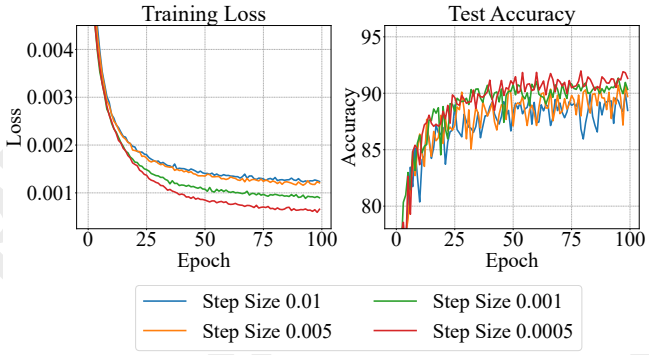

Figure 10: AdaR is applied to ResNet-18 on CIFAR-10 to ablate the effects of step size, evaluated by training loss and test accuracy.

Acknowledgments

This work was supported by National Key R&D Program of China (No.2023YFC3805700). The authors thank the anonymous reviewers for their helpful comments. We sincerely thank Associate Professor Hong Han for her valuable assistance in polishing the English language of the paper.

Contribution Statement

Lianhong Ding and Yangchuan Wang share equal contribution.

References

[Bojanowski *et al.*, 2017] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the association for computational linguistics*, 5:135–146, 2017.

[Das *et al.*, 2024] Aniket Das, Dheeraj Mysore Nagaraj, Soumyabrata Pal, Arun Suggala, and Prateek Varshney. Near-optimal streaming heavy-tailed statistical estimation with clipped SGD. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[Duchi *et al.*, 2011] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning

and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[Guan, 2024] Lei Guan. Adaplus: Integrating nesterov momentum and precise stepsize adjustment on adamw basis. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5210–5214. IEEE, 2024.

[He *et al.*, 2016] 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.

[Heo *et al.*, 2021] Byeongho Heo, Sanghyuk Chun, Seong Joon Oh, Dongyoon Han, Sangdoo Yun, Gyuwan Kim, Youngjung Uh, and Jung-Woo Ha. Adamp: Slowing down the slowdown for momentum optimizers on scale-invariant weights. In *International Conference on Learning Representations*, 2021.

[Huang *et al.*, 2017] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[Huang *et al.*, 2019] Haiwen Huang, Chang Wang, and Bin Dong. Nostalgic adam: weighting more of the past gradients when designing the adaptive learning rate. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 2556–2562, 2019.

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

[Krizhevsky, 2009] A Krizhevsky. Learning multiple layers of features from tiny images. *Master's thesis, University of Tront*, 2009.

[LaValley, 2008] Michael P LaValley. Logistic regression. *Circulation*, 117(18):2395–2399, 2008.

[LeCun *et al.*, 1998] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[Liu *et al.*, 2023] Miaomiao Liu, Dan Yao, Zhigang Liu, Jingfeng Guo, and Jing Chen. An improved adam optimization algorithm combining adaptive coefficients and composite gradients based on randomized block coordinate descent. *Computational intelligence and neuroscience*, 2023(1):4765891, 2023.

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

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

[Luo *et al.*, 2019] Liangchen Luo, Yuanhao Xiong, and Yan Liu. Adaptive gradient methods with dynamic bound of learning rate. In *International Conference on Learning Representations*, 2019.

[Marcus *et al.*, 1993] 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.

[Merity *et al.*, 2018] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. Regularizing and optimizing lstm language models. In *International Conference on Learning Representations*, 2018.

[Nesterov, 1983] Yurii Nesterov. A method for solving the convex programming problem with convergence rate o (1/k2). In *Dokl akad nauk Sssr*, volume 269, page 543, 1983.

[Paszke, 2019] A Paszke. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

[Reyad *et al.*, 2023] 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.

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

[Simonyan and Zisserman, 2015] K Simonyan and A Zisserman. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations (ICLR 2015)*. Computational and Biological Learning Society, 2015.

[Tato and Nkambou, 2018] Ange Tato and Roger Nkambou. Improving adam optimizer, 2018.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[Wang *et al.*, 2025] Yangchuan Wang, Lianhong Ding, Peng Shi, Juntao Li, and Ruiping Yuan. Improving generalization performance of adaptive gradient method via bounded step sizes. *Neurocomputing*, 617:128966, 2025.

[Zeiler, 2012] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[Zhang *et al.*, 2019] Michael Zhang, James Lucas, Jimmy Ba, and Geoffrey E Hinton. Lookahead optimizer: k steps forward, 1 step back. *Advances in neural information processing systems*, 32, 2019.

[Zhang *et al.*, 2020] Jingzhao Zhang, Sai Praneeth Karimireddy, Andreas Veit, Seungyeon Kim, Sashank Reddi, Sanjiv Kumar, and Suvrit Sra. Why are adaptive methods good for attention models? *Advances in Neural Information Processing Systems*, 33:15383–15393, 2020.

[Zhou *et al.*, 2019] Zhiming Zhou, Qingru Zhang, Guansong Lu, Hongwei Wang, Weinan Zhang, and Yong

Yu. Adashift: Decorrelation and convergence of adaptive learning rate methods. In *International Conference on Learning Representations*, 2019.

[Zhou *et al.*, 2020] 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.

[Zhou *et al.*, 2023] Pan Zhou, Xingyu Xie, and Shuicheng YAN. Win: Weight-decay-integrated nesterov acceleration for adaptive gradient algorithms. In *The Eleventh International Conference on Learning Representations*, 2023.

[Zhuang *et al.*, 2020] 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.