An Adaptive Learning Method for Solving the Extreme Learning Rate Problem of Transformer

Jianbang Ding[1(✉)], Xuancheng Ren[1], and Ruixuan Luo[2]

[1] MOE Key Laboratory of Computational Linguistics, Peking University, Beijing, China
{jianbangding,renxc,luoruixuan97}@pku.edu.cn
[2] Center for Data Science, Peking University, Beijing, China

Abstract. Transformer, a neural sequence model entirely based on attention, has achieved great success in natural language processing and become the *de facto* default model for multiple NLP tasks. Albeit its prevalence, the attention-based structure poses unmet challenges that the widely-used adaptive optimization methods, e.g., Adam, have serious difficulty in learning and often fail to converge if applied alone. In this work, we illustrate the problem that the adaptive optimization methods produce extremely-large learning rates that break the balance of stability. We further propose AdaMod, which smooths out extremely-large learning rates with adaptive and momental upper bounds on a per-parameter basis, instead of the uniform scaling in the warmup scheme. We empirically demonstrate AdaMod can improve the learning stability and bring significant improvements to the performance of Transformers and CNNs. Moreover, empirical results verify its effectiveness and robustness across different applications.

Keywords: Transformer · Optimization · AdaMod

1 Introduction

Gradient-based optimization forms the core of first-order optimization algorithms to train deep networks. Remarkably, stochastic gradient descent (SGD) [18], one of the most dominant methods, performs well across many applications, despite its simplicity. However, one shortcoming of SGD is that it scales the gradient uniformly in all directions. This strategy requires a subtle tuning of the learning rate and limits the training speed in the early stage. To address this issue, several adaptive methods have been proposed to achieve faster convergence by computing individual learning rates for different parameters. Examples of such methods include AdaGrad [3], Adam [7], RMSProp [20] and AdaDelta [25]. In particular, Adam is regarded as the default algorithm used across many frameworks [23].

Although adaptive methods have gained great popularity, they still stumble on the stability problem of complex models. It has been observed that they may

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-44693-1_29.

F. Liu et al. (Eds.): NLPCC 2023, LNAI 14302, pp. 361–372, 2023.
https://doi.org/10.1007/978-3-031-44693-1_29

converge to bad local optima when training Transformers and have to resort to the warmup scheme [16,21], which starts with small learning rates for all the parameters until the system reaches stability and is hence used as a common heuristic in practice [5]. However, different parameters play different roles in the model so scaling all the parameters uniformly in the early training stage is not the best. The warmup scheme also introduces some additional hyperparameters and needs tedious fine-tuning. Recent work has put forward some algorithms such as AMSGrad [17] and RAdam [10] to tackle this issue but they failed to achieve considerable improvement over existing methods. In this paper, we first illustrate that the extremely-large learning rates lead to the non-convergence problem and expect to bound large learning rates on a per-parameter basis throughout the training stage.

Under this premise, we propose a new variant of Adam, that is AdaMod, to restrict the adaptive learning rates with adaptive and momental bounds. This endows learning rates with "long-term-memory" historical statics. With this framework, we can balance the gradients across layers to improve learning stability. Finally, we conduct further experiments on Transformers. Empirical results demonstrate that our method can effectively eliminate unexpected large learning rates and hence fix the non-convergence problem. Moreover, it can bring consistent and significant improvement over the vanilla Adam across different architectures such as ResNet and DenseNet.

The contributions are summarized as follows:

- We propose AdaMod, which eliminates extreme learning rates with adaptive momental bound on a per-parameter basis and brings significant improvements to the performance of Transformers without warmup.
- Experiments demonstrate that AdaMod can be used in various applications such as machine translation, document summarization, and image classification due to its robustness and effectiveness.

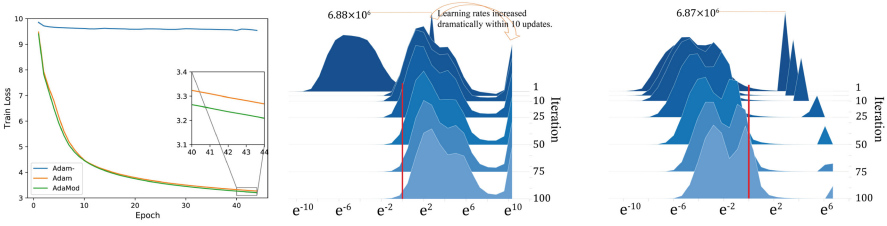
2 Background

2.1 A Brief Review of Adam

Algorithm 1. Adam

Input: initial parameter θ_0, step sizes $\{\alpha_t\}_{t=1}^{T}$, first moment decay β_1, second moment decay β_2, stochastic objective function $f(\theta)$

Initialize $m_0 = 0$, $v_0 = 0$
for $t = 1$ **to** T **do**
  $g_t = \nabla f_t(\theta_{t-1})$
  $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$
  $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ and $V_t = \mathrm{diag}(v_t)$
  $\theta_t = \theta_{t-1} - \alpha_t m_t / \sqrt{V_t}$
end for



(a) Training loss (b) Learning rates for Adam w/o warmup (c) Learning rates for Adam with warmup

Fig. 1. Training loss and learning rate distribution of Transformers on the IWSLT'14 De-En dataset. "Adam-" in (a) denotes Adam without warmup. For (b) and (c), X-axis is the original value in the log scale; Y-axis is training iterations and the height stands for frequency. Red line to show 0 centerlines. Adam fails to converge without warmup due to extremely-large learning rates, while AdaMod can fix this issue and perform better. (Color figure online)

Algorithm 1 provides a brief review of Adam for reference, and for brevity, bias-correction operations are not included. The setup is elaborated as follows. We first compute the gradient g_t of the loss function with respect to previous parameters. Second, we update the low-order moments of gradient m_t, v_t by adopting exponential averaging and computing bias-corrected versions for them. Finally, we refresh the parameter to get a new θ_t. This process needs to iterate T steps until we return the learned parameters. Noting that we refer to $\alpha/\sqrt{V}$ as the learning rate in the paper.

2.2 Extremely-Large Learning Rates Leading to Instability Issues

We first illustrate that the extremely-large learning rates will cause the non-convergence problem. For example, in the NMT experiment in Fig. 1a, the training loss converges to around 9.5 without warmup, and it decreases to below 3.5 after using warmup. In addition, the learning rate histogram is shown in Fig. 1b and Fig. 1c, where the X-axis is the original value in the log scale, Y-axis is the iteration steps and the height stands for frequency. We can observe that without warmup, there are lots of learning rates soaring over 10,000 compared to applying it. Such extremely-large learning rates may lead to the oscillation of the sequence and trap the adaptive method in exceptionally bad local optima. Meanwhile, they cannot help the optimizer escape from that, resulting in a series of non-convergence problems. Similar phenomena are observed in other tasks such as Transformer-XL [2] language modeling.

3 Related Work

Exploring how to tackle the non-convergence issue of adaptive methods is an important research interest of current machine learning research. In recent years,

many remarkable works have provided us with a better understanding of this problem with the proposal of different variants of Adam. [17] first indicated that Adam may not converge due to the lack of "long-term-memory" of past gradients and provided a theoretical guarantee of convergence. Following this track, most of the previous studies focused on how to modify the re-scaling term v_t. [26] argued that there exists an inappropriate correlation between g_t and v_t, which may result in unbalanced updates of step size. Therefore, the authors proposed decorrelating them by temporal shifting, i.e. replacing g_t with g_{t-n} for some manually chosen n to calculate v_t. In a similar vein, [6] discussed that the past gradients $\{g_1, ..., g_{t-1}\}$ are more reliable than g_t. And the authors proposed to weigh more of all past gradients when designing v_t. However, these methods do not radically avoid the non-convergence problem in practice due to the existence of unexpected large learning rates. To solve this problem, [19] considered dropping momentum and removing the larger-than-desired updates by selecting a threshold d for update clipping. However, as their main goal is to minimize the memory cost of optimization algorithms, this technique remains less explored and has limited improvement in generalization performance. To this end, [10] proposed automatic variance rectification of the adaptive learning rate based on derivations. [12] implemented a gradual transition from Adam to SGD by employing dynamic bounds on learning rates to avoid extremely-larger ones. However, its bound function is manually designed and the performance relies heavily on the selection of the final learning rate α^* of SGD.

By contrast to the original Transformer with post-norm residual units (Post-Norm), some work [1,22,24] use pre-norm residual units (Pre-Norm) which locate the layer normalization inside the residual units to obtain well-behaved gradients. Although the Pre-Norm architecture can alleviate the non-convergence problem by normalizing the large gradients at initialization, the extreme learning rates are still found by the end of training on ResNet with Pre-Norm [12]. Although these learning rates do not cause serious non-convergence as in Transformers, they do harm the generalization performance.

4 Methods

This section describes the AdaMod method as well as its properties. Concisely, AdaMod casts dynamic upper bounds on the adaptive learning rates that prevent the calculated learning rates from escalating too fast and becoming undesirably larger than what the historical statistics suggest. This controls the variance of the adaptive learning rates and smooths out the larger-than-expected fluctuations, hence getting more stable and reliable learning rates. AdaMod names from **Ada**ptive and **Mo**mental Boun**d**. The pseudocode is provided in Algorithm 2.

4.1 Smoothing Adaptive Learning Rates

Based on Adam, which computes adaptive learning rates with estimates of first and second moments (i.e., expectation and uncentered variance) of the gradients,

Algorithm 2. AdaMod

Input: initial parameter θ_0, step sizes $\{\alpha_t\}_{t=1}^{T}$, first moment decay β_1, second moment *decay*β_2, smoothing coefficient γ, stochastic objective function $f(\theta)$

Initialize $m_0 = 0$, $v_0 = 0$, $\hat{\eta}_0 = 0$
for $t = 1$ **to** T **do**
  $g_t = \nabla f_t(\theta_{t-1})$
  $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$
  $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ and $V_t = \mathrm{diag}(v_t)$
  $\hat{\eta}_t = \gamma \hat{\eta}_{t-1} + (1 - \gamma)\alpha_t / \sqrt{V_t}$
  $\eta_t = \min(\hat{\eta}_t, \alpha_t / \sqrt{V_t})$
  $\theta_t = \theta_{t-1} - \eta_t \odot m_t$
end for

our method further estimates the first-order moments of the individual adaptive learning rates $\alpha_t / \sqrt{V_t}$. Inspired by exponential moving average (EMA) which enjoys popularity in estimating the lower-order moments of the gradients. We do average directly on the learning rates $\alpha_t / \sqrt{V_t}$ computed by Adam. Specifically, we apply the following operation:

$$\hat{\eta}_t = \gamma \hat{\eta}_{t-1} + (1 - \gamma)\alpha_t / \sqrt{V_t}, \tag{1}$$

where $\alpha_t / \sqrt{V_t}$ are the learning rates computed by Adam at step t. Thus, the current momental value $\hat{\eta}_t$ is an interpolation between the previous momental value $\hat{\eta}_{t-1}$ and the current learning rates. Since the average range of the data in the exponential moving average is $1/(1-\gamma)$, which can be proven by evaluating an infinite series, the new hyperparameter γ controls the memory of $\hat{\eta}_t$. For example, when $\gamma = 0.9$ the average range is 10 periods; when $\gamma = 0.999$ the average range is 1,000 periods, so on and so forth. It is worth noting that when $\gamma \to 0$, AdaMod degenerates to Adam.

Equation 1 can be expressed in another version, where the current momental value is an exponentially weighted moving average with discount factor γ:

$$\hat{\eta}_t = (1 - \gamma) \sum_{i=1}^{t} \gamma^{t-i} \cdot \alpha_i / \sqrt{V_i}. \tag{2}$$

This endows the current value $\hat{\eta}_t$ with "long-term-memory" of past values $\alpha_i / \sqrt{V_i}$. For general cases, we set $\hat{\eta}_0 = 0$ and $\gamma = 0.9999$.

4.2 Bounding Adaptive Learning Rates

For the current momental value $\hat{\eta}_t$, we further take it as the adaptive upper bound for $\alpha_t / \sqrt{V_t}$ to eliminate extremely-large learning rates:

$$\eta_t = \min(\hat{\eta}_t, \alpha_t / \sqrt{V_t}). \tag{3}$$

where η_t is the final learning rate obtained by the bounding operation. Intuitively, this can be seen as clipping the learning rates element-wisely so that the

output is constrained by the different current momental values. Our proposed momental bounding strategy is significantly different from previous work such as Adafactor and AdaBound. These methods rely on manually chosen thresholds or bounding functions to truncate learning rates or updates. Compared to AMSGrad, AdaMod replace the absolute maximum of previous v_t values with a running average.

$$\theta_t = \theta_{t-1} - \eta_t \odot m_t, \tag{4}$$

Then, we use η_t and m_t to make a parameter update. This process needs to iterate T steps until an approximate solution is returned.

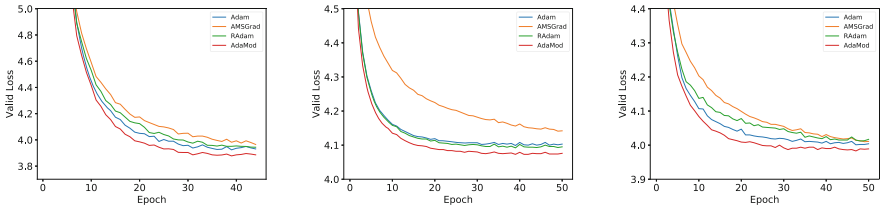
5 Experiments

This section performs a thorough evaluation of AdaMod optimizer on different deep learning tasks against fine-tuned baselines. We refer to several benchmarks for Transformers: **IWSLT'14 De-En/WMT'14 En-De** for neural machine translation, and **CNN-DailyMail** for document summarization. To verify the versatility of AdaMod, we also add image classification on **CIFAR-10/CIFAR-100** [8] with CNN and language modeling on **Penn Treebank** [13] with LSTM. We compare our method with popular optimization algorithms including SGDM, AdaBound, Adam, AMSGrad and RAdam [10]. We classify AdaMod, RAdam, AMSGrad, and Adam into Adam-like algorithms, and the others into SGD-like methods. It is worth noting that except SGDM, the rest belong to adaptive methods. To achieve better performance, we apply decoupled weight decay to all adaptive methods in our experiments on the basis of [11]'s work, and adopt the warmup scheme for all the methods except AdaMod and RAdam when training Transformers. For statistical significance, we conduct each experiment for 5 random trials and report $p - value$ for the significance test. Noting that full hyperparameter tuning details and more experiments results (running Pre-Norm Transformers on IWSLT'14, CNN on CIFAR-10/CIFAR-100, and LSTM on Penn Treebank) are reported in the supplementary material.

5.1 Neural Machine Translation

Machine translation is one of the most important applications in NLP [21]. To evaluate the effectiveness of AdaMod, we train Transformer-based models on two widely used datasets: IWSLT'14 De-En and WMT'14 En-De.

Our experiments are based on the vanilla Transformers [21] implementation from the *fairseq* open library [14]. Due to the limited size of the IWSLT'14 dataset, we use a relatively small model in training. The size of embeddings and hidden states is set to 512 and the number of heads in multi-head attention is set to 4. For WMT'14, we train the transformer base version and the big version respectively. Both the two models consist of a 6-layer encoder and a 6-layer decoder. The size of the embedding is set to 512 for the base model and 1024 for the big. We set $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 1e - 9$. We use a linear warmup for Adam in the first 4000 updates. For IWSLT'14, we run training on



(a) Transformer-Small (b) Transformer-Base (c) Transformer-Big

Fig. 2. Validation loss for Transformer-based model. For (a) is trained on IWSLT'14 De-En, (b) and (c) on WMT'14 En-De. AdaMod without warmup shows both faster convergence and strong final performance compared with Adam with warmup.

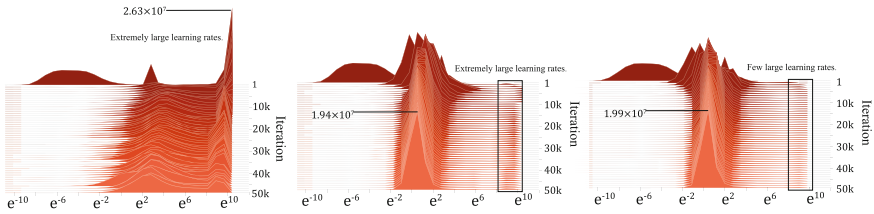
Table 1. BLEU score on Neural Machine Translation. We train the small Transformer on IWSLT'14 De-En, the base and the big model on WMT'14 En-De. Report for $Median(Mean \pm Std)$.

Method	Trans-Small	Trans-Base	Trans-Big
SGDM	29.52 (29.53 ± 0.26)	23.33 (23.35 ± 0.22)	24.47 (24.57 ± 0.24)
AdaBound	34.28 (34.32 ± 0.18)	27.02 (27.05 ± 0.13)	28.24 (28.25 ± 0.09)
Adam	34.62 (34.58 ± 0.15)	27.11 (27.10 ± 0.13)	28.31 (28.31 ± 0.12)
AMSGrad	33.81 (33.84 ± 0.08)	25.98 (26.01 ± 0.10)	27.46 (27.45 ± 0.05)
RAdam	34.72 (34.71 ± 0.06)	27.15 (27.17 ± 0.06)	28.17 (28.20 ± 0.05)
AdaMod	**34.88** (34.85 ± 0.05)	**27.39** (27.39 ± 0.04)	**28.58** (28.57 ± 0.05)

1 NVIDIA RTX 2080Ti GPU, the maximum tokens per batch is set as 4000, weight decay as 1e-4, and dropout rate as 0.3. As for WMT'14, we conduct training on 4 T V100 GPUs and set the maximum tokens as 8192. Note that γ is set as 0.999 for the base model, and 0.9999 for the other two. To make the experiments more convincing, We also compare AdaMod with Adam on the Pre-Norm Transformers in the supplementary material.

Performance Comparison. We use BLEU [15] as the metric to evaluate the performance and give the results in Table 1. We also report p-value between Adam and AdaMod for the significance test (5.42e−3, 1.92e−3, 1.85e−3, all less than 0.01). As discussed above, Adam relies on the warmup scheme when training Transformers to avoid the non-convergence problem (The same goes for other methods except AdaMod and RAdam). As for AdaMod, it can train Transformers without warmup and achieve higher BLEU scores (around 0.3 gains) on both two datasets. Moreover, valid loss values are shown in Fig. 2. It can be seen that AdaMod converges faster against other methods throughout the training stage.

Learning Rates Comparison. We further compare the learning rates histogram of Transformers on the IWSLT'14 De-En between Adam and AdaMod. As shown



(a) Adam without warmup (b) Adam with warmup (c) AdaMod

Fig. 3. The learning rate comparison of Transformers on the IWSLT'14 De-En. AdaMod subtly restrains extremely-large learning rates throughout the training.

Table 2. F1-ROUGE score for CopyTransformer on CNN-DailyMail. Report for $Median(Mean \pm Std)$.

CNN-DM	ROUGE-1	ROUGE-2	ROUGE-L
SGDM	38.34 (38.30 ± 0.15)	16.53 (16.58 ± 0.16)	35.35 (35.29 ± 0.22)
AdaBound	37.75 (37.72 ± 0.09)	16.07 (16.04 ± 0.11)	34.83 (34.86 ± 0.14)
Adam	39.22 (39.24 ± 0.08)	17.19 (17.18 ± 0.13)	36.38 (36.34 ± 0.11)
AMSGrad	39.04 (39.05 ± 0.02)	16.85 (16.86 ± 0.03)	36.07 (36.06 ± 0.05)
RAdam	39.42 (39.44 ± 0.04)	17.23 (17.25 ± 0.05)	36.44 (36.45 ± 0.03)
AdaMod	**39.51** (39.51 ± 0.03)	**17.37** (17.37 ± 0.03)	**36.80** (36.82 ± 0.06)

in Fig. 3, where the X-axis is the original value in the log scale, and Y-axis is iteration steps and the height stands for frequency. As mentioned above, the histogram of Adam is distorted seriously due to the extremely-large learning rates. This phenomenon has been alleviated after adopting warmup, while there is still a spike of learning rates between e^8 to e^{10}. Although they are not enough to fail to converge, they still hurt the generalization performance. In contrast, AdaMod filters the unexpected large learning rates without warmup. Specifically, in the early stage, AdaMod successfully suppressed the abnormal upward trend in learning rates, and keep them within a reasonable range to balance the gradients across layers

5.2 Document Summarization

We also consider abstractive document summarization task on the CNN-DailyMail corpus, which is a standard news corpus and widely used for text summarization. Following [4]'s work, our experiment is based on a Transformer containing 4 layers in each block, and one of the attention-heads is selected as the copy-distribution. Apart from this, the size of the word embedding is set to 512 and the model shares it between the encoder and decoder. We set $\beta_1 = 0.9$, $\beta_2 = 0.998$, and a linear warmup in the first 8000 updates for all the methods except AdaMod and RAdam. The dropout rate is set as 0.2 and the batch size is

4096. The gradients are accumulated 4 times for each update. For AdaMod, the memory coefficient γ is set as 0.9999. All the models are trained on 2 NVIDIA RTX 2080Ti GPUs.

We evaluate by F1-ROUGE [9], i.e., ROUGE-1, ROUGE-2, ROUGE-L, and show the results in Table 2. Note that all the p-values between Adam and AdaMod are less than 0.01 (1.1e−4, 9.6e−3, 2.3e−5). As we can see from the table, AdaMod brings a significant improvement over Adam, and outperforms the rest methods across all three scores, despite without warmup. That is, not only in terms of Rouge-1 related to content but the fluency of the language is also ameliorated by applying AdaMod. It indicates that our method improves the performance of the model on the basis of solving the non-convergence problem and demonstrates the versatility of AdaMod on different natural language tasks based on Transformers.

6 Analysis

Robustness to Different α. To investigate the robustness of AdaMod, we conduct experiments with the Transformer-small on the IWSLT'14 De-En as in the Sect. 5.1. We test Adam with warmup and AdaMod with different α (i.e. initial step size), which is chosen from multiples of 5e−4. The scores are reported in Table 3. It can be found that Adam is very sensitive to α when training Transformers, and a slight disturbance to α will cause poor convergence (below 10). At this time Adam has to resort to other strategies such as gradient clipping or accumulation to get more stable learning rates. However, AdaMod can still converge to highly similar results in the interval of twice the original step size (34.90 ± 0.1). For example, when α is increased by 2.5 times, it can achieve relatively good performance (above 30).

Table 3. BLEU score of Adam and AdaMod with different α using Transformer-small on IWSLT'14 De-En. "×" denotes divergence.

Method	$\alpha = 5e-4$	$\alpha = 7.5e-4$	$\alpha = 1e-3$	$\alpha = 1.25e-3$	$\alpha = 1.5e-3$
Adam	34.62	6.70	1.84	×	×
AdaMod	**34.88**	**34.99**	**34.86**	32.09	×

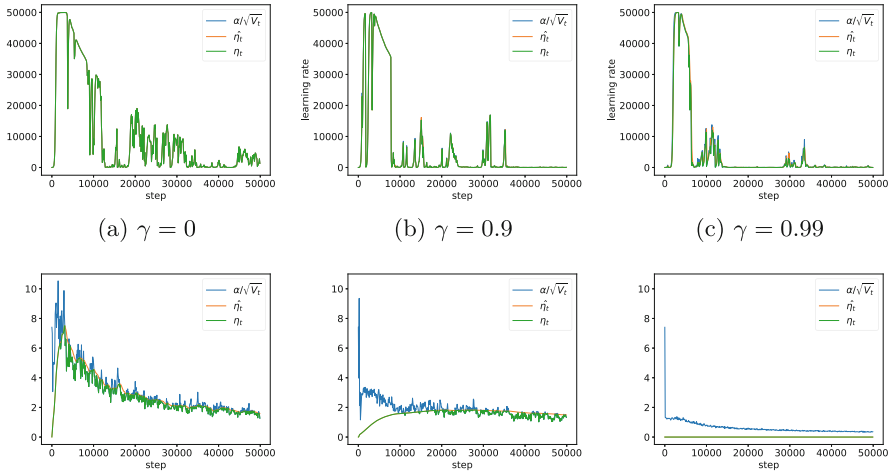
Training with Small Batches. We also show that AdaMod enables warmup-free, validation-based training even for small batches. Table 4 demonstrates that Adam depends on a large batch size to obtain stable gradients. As the batch size decreases, the performance of the model drops rapidly until Adam fails to converge. In the case of a small batch, the unexpected gradient fluctuations

become more drastic, further exacerbating the negative effect of the large learning rates. This gap will become even greater when training complex models. However, AdaMod can converge to similar results (34.85 ± 0.05) by endowing learning rates with long-term memory evne if the batch size has dropped to nearly one-third of the original.

Table 4. BLEU score of Adam and AdaMod with different batches using Transformer-small on IWSLT'14 De-En. "×" denotes divergence.

Method	$bz = 4000$	$bz = 2000$	$bz = 1500$	$bz = 1000$	$bz = 500$
Adam	34.62	33.72	4.43	3.78	×
AdaMod	**34.88**	**34.89**	**34.83**	34.31	3.64

How Momental Bounds Regulate Learning Rates? As discussed above, $\alpha/\sqrt{V_t}$ have a large variance in the early training stage, leading to extremely-large learning rates, which could hamper performance and even cause stability issues.



(a) $\gamma = 0$ (b) $\gamma = 0.9$ (c) $\gamma = 0.99$

(d) $\gamma = 0.999$ (e) $\gamma = 0.9999$ (f) $\gamma = 1$

Fig. 4. How momental bounds regulate learning rates? The learning rates of a parameter were randomly sampled from the encoder of the Transformer. As γ increases, the growth trend of $\alpha/\sqrt{V_t}$ is suppressed by momental bound $\hat{\eta}_t$, and a new stable learning rates η_t is returned. Noting that when $\gamma = 0$, AdaMod degenerates to Adam.

By computing the momentum $\hat{\eta}_t$, historical statistics of learning rates could be fully considered. Employing it as the momental bound could adaptively suppress the growth trend of learning rates. To visualize this process, we supplement

an analysis experiment on IWSLT'14. Specifically, we randomly sample a parameter from the self-attention layer to observe its learning rate, and the phenomena are shown in Fig. 4. It can be seen that when historical statistics are not considered (i.e. $\gamma = 0$) or statistics insufficient (i.e. $\gamma = 0.9, 0.99$), the effect of momental bounds is not obvious. The parameter's learning rate surges to 50000 in the early term and collapses dramatically in the later period. However as γ increases, the growth trend of $\alpha/\sqrt{V_t}$ is suppressed by momental bound (i.e. $\gamma = 0.999, 0.9999$), and a new stable learning rates η_t is returned, eliminating large rates from the root, which verifies our motivation. When $\gamma \to 1$, η_t becomes more stable and reliable, which can be approximated as a tailor-made learning rate applied to each parameter of the model, rather than scales all the parameters uniformly like SGD. It is likely that AdaMod combines the advantages of both types of adaptive and non-adaptive methods. *We recommend a* $\boldsymbol{\gamma}$ *in* $\{$*0.999,0.9999*$\}$ *as preferred for its usually behaving a good performance across most models in practice.*

7 Conclusion

In this paper, we illustrate that popular adaptive algorithms fail to converge when training Transformers due to the large learning rates. For that, we design a concise strategy to constrain the spikes to avoid the non-convergence issue. Our proposed algorithm, AdaMod, exerts momental bounds on a per-parameter basis to prevent them from becoming undesirably larger than what the historical statistics suggest, hence getting more stable and reliable learning rates. Empirical results demonstrate our method gains steady performance improvements to Transformers across different applications.

References

1. Chen, M.X., et al.: The best of both worlds: Combining recent advances in neural machine translation. In: ACL (1), pp. 76–86. Association for Computational Linguistics (2018)
2. Dai, Z., Yang, Z., Yang, Y., Carbonell, J.G., Le, Q.V., Salakhutdinov, R.: Transformer-xl: attentive language models beyond a fixed-length context. In: ACL (1), pp. 2978–2988. Association for Computational Linguistics (2019)
3. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. J. Mach. Learn. Res. **12**, 2121–2159 (2011)
4. Gehrmann, S., Deng, Y., Rush, A.M.: Bottom-up abstractive summarization. In: EMNLP, pp. 4098–4109. Association for Computational Linguistics (2018)
5. Gotmare, A., Keskar, N.S., Xiong, C., Socher, R.: A closer look at deep learning heuristics: Learning rate restarts, warmup and distillation. In: ICLR (Poster). OpenReview.net (2019)
6. Huang, H., Wang, C., Dong, B.: Nostalgic adam: weighting more of the past gradients when designing the adaptive learning rate. In: IJCAI, pp. 2556–2562. ijcai.org (2019)

7. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: ICLR (Poster) (2015)
8. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images. Tech. rep, Citeseer (2009)
9. Lin, C.Y.: Rouge: a package for automatic evaluation of summaries. In: Text Summarization Branches out, pp. 74–81 (2004)
10. Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., Han, J.: On the variance of the adaptive learning rate and beyond. arXiv preprint arXiv:1908.03265 (2019)
11. Loshchilov, I., Hutter, F.: Fixing weight decay regularization in adam. arXiv preprint arXiv:1711.05101 (2017)
12. Luo, L., Xiong, Y., Liu, Y., Sun, X.: Adaptive gradient methods with dynamic bound of learning rate. In: ICLR (Poster). OpenReview.net (2019)
13. Marcus, M.P., Santorini, B., Marcinkiewicz, M.A.: Building a large annotated corpus of English: the penn treebank. Comput. Linguist. **19**(2), 313–330 (1993)
14. Ott, M., et al.: fairseq: a fast, extensible toolkit for sequence modeling. In: NAACL-HLT (Demonstrations), pp. 48–53. Association for Computational Linguistics (2019)
15. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, pp. 311–318. Association for Computational Linguistics (2002)
16. Popel, M., Bojar, O.: Training tips for the transformer model. Prague Bull. Math. Linguistics **110**, 43–70 (2018)
17. Reddi, S.J., Kale, S., Kumar, S.: On the convergence of adam and beyond. In: ICLR. OpenReview.net (2018)
18. Robbins, H., Monro, S.: A stochastic approximation method. The annals of mathematical statistics, pp. 400–407 (1951)
19. Shazeer, N., Stern, M.: Adafactor: adaptive learning rates with sublinear memory cost. In: ICML. Proceedings of Machine Learning Research, vol. 80, pp. 4603–4611. PMLR (2018)
20. Tieleman, T., Hinton, G.: Lecture 6.5-rmsprop: divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks Mach. Learn. **4**(2), 26–31 (2012)
21. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)
22. Wang, Q., Li, B., Xiao, T., Zhu, J., Li, C., Wong, D.F., Chao, L.S.: Learning deep transformer models for machine translation. In: ACL (1), pp. 1810–1822. Association for Computational Linguistics (2019)
23. Wilson, A.C., Roelofs, R., Stern, M., Srebro, N., Recht, B.: The marginal value of adaptive gradient methods in machine learning. In: Advances in Neural Information Processing Systems, pp. 4148–4158 (2017)
24. Xiong, R., et al.: On layer normalization in the transformer architecture. CoRR abs/2002.04745 (2020)
25. Zeiler, M.D.: Adadelta: an adaptive learning rate method. arXiv preprint arXiv:1212.5701 (2012)
26. Zhou, Z., Zhang, Q., Lu, G., Wang, H., Zhang, W., Yu, Y.: Adashift: decorrelation and convergence of adaptive learning rate methods. In: ICLR (Poster). OpenReview.net (2019)