AdaDiff: Adaptive Gradient Descent with the Differential of Gradient

Dian Huang1, Dawei Yin2, Lijun Zhang3, Dezhong Peng4, QiLin Li1*

1,4 College of Computer Science, Sichuan University, Chengdu, 610025, China
2 Sichuan Zhiqian Technology Co., Ltd, Chengdu, Sichuan, China
3 Chengdu Ruibei Yingte Information Technology Limited Company, Chengdu, China
*Corresponding author’s e-mail: liql367x@sc.sgcc.com.cn

Abstract. Optimization methods are crucial to train deep neural networks. Adaptive optimization methods, especially Adam, are wildly used because they aren’t sensitive to the selection of learning rate and converge fast. Recent work point out Adam has a performance gap with SGD and even not converge because of the unstable and extreme learning rates. Many variants of Adam are proposed to solve the problem, such as AMSGrad, AdaBound and AdaBelief. In this paper, we propose a new variant of Adam, called AdaDiff. AdaDiff computing gradient descent step size by the exponential moving average(EMA) of gradient and differential of gradients aiming to make the training of networks more stable. We compare our method with other optimizers on various tasks. The results show AdaDiff outperforms Adam and minimizes the performance gap with SGD.

1. Introduction
The past decade has seen the rapid development of deep learning in many areas, such as computer vision, natural language processing. As an important part of deep learning, optimization methods have attracted many researchers.

Stochastic gradient descent (SGD) is one of the dominant optimization methods used in deep neural networks. SGD has a good generalization performance. Actually, a lot of state-of-the-art results of image classification datasets are obtained by applying SGD. However, SGD is sensitive to learning rates and converges slowly because of using a fixed learning rate for all parameters. To solve the problem, many adaptive methods are proposed, such as Adagrad[1], AdaDelta[2], RMSProp, Adam[3]. Adaptive methods adjust learning rates based on past gradients for each parameter and converge faster, so become more and more popular. Especially, Adam is the default optimizer for many deep learning tasks. Nevertheless, Adam has a bad generalization ability and unstable compared to SGD. Many variants are proposed to solve the problem, such as AdaBound[4], RAdam[5], AdaBelief[6].

Nonetheless, all the methods mentioned above adjust learning rates by the past and present gradients. Inspired by proportional-integral-derivative (PID), a classic and widely used algorithm in automatic control, we propose a new variant of Adam taking the differential of gradient into account. Moreover, the suggestion of changing the position of $\epsilon$ is adopted in our method, aiming to avoid the poor performance caused by the extreme learning rates[4]. The main contributions are summarized as:

- We propose a new variant of Adam, which improves the original Adam by adjusting step size by considering the differential of gradients. We believe the introduction of the differential of gradients can stabilize the process of training deep neural networks.
Further experiments are conducted on various tasks in computer vision and natural language processing. Experiment results demonstrate that our method outperforms others in generalization ability and the speed of converge.

2. Related work
Generally, optimizers are divided into two classes, first-order and second-order methods. The second-order methods, such as quasi-Newton method[7], Hessian-Free optimization approach[8] and Kronector-factored approximate curvature (KFAC)[9], are regarded as computational complex and hence are not wildly used.

First-order methods contain non-adaptive and adaptive methods. Adaptive methods, especially Adam, are wildly used because of the ability of fast converge and non-sensitive to learning rates. However, Adam suffers from bad performance compared to the non-adaptive method (SGD). Recently, many variants of Adam have been developed aiming to minimize the performance gap of Adam and SGD. AMSGrad[10] guaranteed convergence but fail to get a considerable performance. Adabound achieves a smooth transition from Adam to SGD by dynamic bound of learning rate[4]. AdaBelief computing stepszie by the ‘belief’ in the current gradient direction[6] and achieves a better performance than AdaBound.

Algorithm 1: Adam Optimizer

Input: initial learning rate $\alpha$, initial parameter $\theta_0$, moment decay $\{\beta_1, \beta_2\}$
Initialize: $m_0 = 0, s_0 = 0$
for $t = 1$ to $T$ do
  $g_t = \nabla f_t(\theta_{t-1})$
  $m_t = \beta_1 m_{t-1} + (1 - \beta_1)g_t$
  $v_t = \beta_2 v_{t-1} + (1 - \beta_2)g_t^2$
  $\hat{m}_t = m_t / (1 - \beta_1)$
  $\hat{v}_t = v_t / (1 - \beta_2)$
  $\theta_t = \theta_{t-1} - \alpha \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$
end for

Algorithm 2: AdaDiff Optimizer

Input: initial learning rates $\alpha$, initial parameters $\theta_0$, moment decay $\{\beta_1, \beta_2\}$
Initialize: $m_0 = 0, s_0 = 0$
for $t = 1$ to $T$ do
  $g_t = \nabla f_t(\theta_{t-1})$
  $m_t = \beta_1 m_{t-1} + (1 - \beta_1)(g_t + (g_t - g_{t-1})^2)$
  $v_t = \beta_2 v_{t-1} + (1 - \beta_2)(g_t + (g_t - g_{t-1})^2)$
  $\hat{m}_t = m_t / (1 - \beta_1)$
  $\hat{v}_t = v_t / (1 - \beta_2)$
  $\theta_t = \theta_{t-1} - \alpha \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$
end for
\[ g_t = \nabla f_t(\theta_{t-1}) \]
\[ m_t = \beta_t m_{t-1} + (1 - \beta_t) g_t \]
\[ v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 + (1 - \beta_2)(g_t - g_{t-1})^2 \]
\[ \hat{m}_t = m_t / (1 - \beta_t^t) \]
\[ \hat{v}_t = (v_t + \epsilon) / (1 - \beta_2^t) \]
\[ \theta_t = \theta_{t-1} - \alpha \hat{m}_t / \sqrt{\hat{v}_t} \]

end for

4. Experiments

In this section, we conduct various experiments aiming to compare our method with others, such as SGD, Adam and AdaBelief. We choose them because SGD and Adam are the most widely used methods and AdaBelief is one of the best optimizers in recent years. In order to validate the general applicability of AdaDiff, we choose two classic tasks of deep learning: image classification in computer vision and language modeling in natural language processing. The details are displayed in Table 1. We refer to the work of AdaBelief and run all experiments for 3 random seeds.

| Dataset    | Network Type | Architecture |
|------------|--------------|--------------|
| CIFAR-10   | Convolutional| ResNet-34    |
| CIFAR-10   | Convolutional| DenseNet-121 |
| CIFAR-100  | Convolutional| ResNet-34    |
| CIFAR-100  | Convolutional| DenseNet-121 |
| Penn Treebank | Recurrent       | 2-layer LSTM |
| Penn Treebank | Recurrent       | 2-layer LSTM |

4.1. Image Classification

Firstly, we conduct experiments on Cifar datasets. Weight decay is 5e-4 for all optimizers. For SGD, we search learning rate among \{10, 1, 0.1, 0.01, 0.001\} and set \( \beta = 0.9 \) which is the default setup. For adaptive methods, such as Adam, AdaBelief and AdaDiff, we use the default parameters of Adam. Specifically, \( \text{lr} = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8} \). We run 200 epochs and reduce learning rates by 10 at 150-th epoch as recommended.

We choose test accuracy as evaluation indices and the results are showed in figure1. We can see that our method gets similar performance to AdaBelief and SGD and outperforms Adam whatever dataset and network architecture. The results illustrate that AdaDiff converges as fast as adaptive methods and achieves the same performance as SGD.

4.2. Language Modeling

We then experiment on Penn TreeBank dataset. For SGD, we search learning rate among \{100, 10, 1, 0.1, 0.01\} and set \( \beta = 0.9 \) which is the default setup. For adaptive methods, we search learning rate among \{0.1, 0.01, 0.001, 0.0001\} and search \( \epsilon \) among \{10e-3, 10e-5, 10e-8, 10e-12, 10e-16\}. Other parameters keep the default setup of Adam. We run 200 epochs and reduce learning rates by 10 at 100-th and 145-th epoch respectively.
Test perplexity is used to evaluating performance and the results are reported in figure2 and table2. The results show our proposed AdaDiff has the lowest test perplexity and converges faster than other methods for all the models on Penn Treebank. Besides, the improvement is more significant if the model is more complex. In particular, AdaDiff surpasses 1.8% than Adam in 2-layer LSTM, while in 3-layer LSTM, AdaDiff performs over 4.6% better than Adam.
Table 2. Test perplexity on Penn Treebank (lower is better)

| Layer   | AdaDiff  | Adabelief | Adam     | SGD      |
|---------|----------|-----------|----------|----------|
| 2-layer | 65.26 ± 0.07 | 66.73 ± 0.02 | 67.06 ± 0.35 | 67.01 ± 0.18 |
| 3-layer | 60.26 ± 0.10 | 61.22 ± 0.17 | 64.89 ± 0.02 | 63.83 ± 0.06 |

5. Future work
Though achieving good goals, there still remains some work that is worth doing in future. For example, SGD is regarded as the best optimization method in image classification tasks, despite AdaDiff perform well as SGD on cifar, we don’t validate it on larger datasets, such as Imagenet. It’s a pity caused by our constrained computing resources. Similarly, more complex models, such as pre-trained language models, are also worth experimenting in order to validate AdaDiff’s stability. We will conduct relative experiments if we have enough resources.

6. Conclusion
In this study, we investigate some classical and typical optimizers and analyze their features. In order to solve problems of adaptive methods, such as bad performance and unstable. We propose AdaDiff, a new variant of Adam. AdaDiff adjusts learning rates by EMA of gradient and differential of gradients. Besides, we change the position of $\varepsilon$. Then, we compare our method with other methods on different tasks. The results show AdaDiff performs well on all datasets compared to SGD, Adam and Adabelief. As an adaptive optimizer, AdaDiff has a more stable learning rate and better generalization ability than other adaptive optimizers.

References
[1] Duchi J, Hazan E, Singer Y. (2011)Adaptive subgradient methods for online learning and stochastic optimization. Journal of machine learning research, 12: 2121-2159.
[2] Zeiler M D. (2012)Adadelta: an adaptive learning rate method. Computer Science.
[3] Kingma D P, Ba J. (2015)Adam: A method for stochastic optimization. In: International Conference on Learning Representations.
[4] Luo L, Xiong Y, Liu Y, et al. (2018)Adaptive gradient methods with dynamic bound of learning rate. In: International Conference on Learning Representations.
[5] Liu L, Jiang H, He P, et al. (2019)On the variance of the adaptive learning rate and beyond. In: International Conference on Learning Representations.
[6] Zhuang J, Tang T, Ding Y, et al. (2020)Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. Advances in Neural Information Processing Systems, 33.
[7] Le Q V, Ngiam J, Coates A, et al. (2011)On optimization methods for deep learning. In: International conference on machine learning.
[8] Martens J. (2010)Deep learning via hessian-free optimization. In: International conference on machine learning. pp. 735-742.
[9] Martens J, (2015)Grosse R. Optimizing neural networks with kronecker-factored approximate curvature. JMLR.org.
[10] Reddi S J, Kale S, Kumar S. (2019)On the Convergence of Adam and Beyond.