ADASHIFT: DECORRELATION AND CONVERGENCE OF ADAPTIVE LEARNING RATE METHODS ICLR 2019 REPRODUCIBILITY CHALLENGE

Mikhail Konobeev[*‡], **Irina Saparina**[*‡], **Taras Khakhulin**[*†]
Skolkovo Institute of Science and Technology
[‡] National Research University Higher School of Economics
[†] Moscow Institute of Physics and Technology
{konobeeev.michael,irisaparina,t.khakhulin}@gmail.com

ABSTRACT

AdaShift optimization method (Zhou et al., 2019) attempts to solve the problem of Adam by temporally and spatially decorrelating momentum term and moving averages of squared gradients. The method is proposed after an analysis of Adam using net update factors that represent the influence of the current gradient on the subsequent optimization process and it was shown that decorrelating moving averages of squared gradients with momentum terms leads to balanced net update factors. In this work we validate experimental results achieved by the authors by reproducing them on top of PyTorch [1]. Overall, our results agree with those reported by the authors, however we find a small discrepancy between their implementation in TensorFlow and description of the algorithm in the paper. Additionally, we conduct more realistic experiments with generative model and show that the proposed optimizer works in a sequential online optimization problem on which Adam fails, even though the problem does not meet the condition of gradients being temporally independent.

1 INTRODUCTION

Adam (Kingma & Ba, 2014) is a popular optimization method in deep learning as it uses only information about gradients of the function being optimized and empirically achieves good results in a variety of tasks. At the same time, it was noted that in some problems it does not lead to convergence. Reddi et al. (2018) provided an analysis of the issue and described a simple optimization problem on which Adam fails.

The counterexample on which Adam fails is setup as an online optimization problem in which some gradients have large magnitude but occur rarely, while others have smaller magnitude and are more frequent. It was argued that because of the use of exponential moving average of squared past gradients as an adaptive learning rate, the influence of gradients with big magnitude disappears too quickly and the method fails to converge. Proposed modifications incorporated longer history of past squared gradients.

Zhou et al. (2019) analyzed Adam with proposed net update factors that measure the impact of a current gradient on the future optimization process. They show that in the sequential online and stochastic optimization problems these net update factors are unbalanced for Adam: gradients in the right direction have smaller net update factors than gradients in the wrong direction. It is argued that the modification proposed by Reddi et al. (2018) might not make the scaling factors very adaptive as it consist of either increasing the weight of past squared gradients, or taking element-wise maximum between the previous squared scaling factor and a new value of exponential moving average of squared gradients. It was further shown that decorrelating scaling factors with the step direction makes scaling factors balanced. The proposed AdaShift method temporally shifts the scaling fac-

[*] Equal contribution
[1] Our implementation is available online: https://github.com/MichaelKonobeev/adashift

tor from the step direction and uses an additional spacial function that is applied to each block of parameters (such blocks naturally occur in layers of deep neural networks).

In this work we describe our reproduction of the optimizer and the experiments conducted by Zhou et al. (2019) on top of PyTorch. Our results agree with those reported by the authors. However, we note a small discrepancy between authors implementation and their description of the algorithm in the paper: while in the paper spacial function is always applied before squaring the gradients, in the code the gradients are squared first. We use the same order as in the authors' code in our experiments. As additional experiments, we optimized the sequential online optimization problem proposed by Reddi et al. (2018) and our results show that AdaShift does converge to the correct point even though as stated in the paper this optimization problem does not meet the condition of gradients being temporally independent. We also train both discriminator and generator in generative model WGAN-GP (Gulrajani et al., 2017) and report discriminator loss and inception score.

2 Preliminaries

Adam. General update rule for several common optimization methods can be written in the following way:

$$\theta_{t+1} \leftarrow \theta_t - \frac{\alpha_t}{\sqrt{v_t}} m_t, \tag{1}$$

where t is step number; $\theta_t \in \mathbb{R}^d$ is a vector of parameters; α_t is a learning rate; $\sqrt{v_t} := \psi(g_1, \dots, g_t) \in \mathbb{R}^d_+$ is a scaling factor; and $m_t := \varphi(g_1, \dots, g_t) \in \mathbb{R}^d$ is a function of historic gradients. In Adam m_t and v_t are obtained from exponential moving averages. Firstly, consider

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \qquad v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2, \tag{2}$$

where $\beta_1, \beta_2 \in [0, 1)$ are the exponential decay rates for m_t and v_t respectively, and $m_0 = v_0 = 0$. These equations could be rewritten in the following way:

$$m_t = (1-\beta_1) \sum_{i=1}^{t} \beta_1^{t-i} g_i, \qquad v_t = (1-\beta_2) \sum_{i=1}^{t} \beta_2^{t-i} g_i^2. \tag{3}$$

Kingma & Ba (2014) also proposed implementing bias-correction which leads to the following final equations:

$$m_t = \frac{\sum_{i=1}^t \beta_1^{t-i} g_i}{\sum_{i=1}^t \beta_1^{t-i}}, \qquad v_t = \frac{\sum_{i=1}^t \beta_2^{t-i} g_i^2}{\sum_{i=1}^t \beta_2^{t-i}}. \tag{4}$$

Online optimization problem. An online optimization problem consists of a sequence of cost functions $f_1(\theta), \dots, f_t(\theta), \dots, f_T(\theta)$. The performance of the optimizer is measured using regret function $R(T) := \sum_{t=1}^T [f_t(\theta_t) - f_t(\theta^*)]$, where $\theta^* := \arg\min_\theta \sum_{t=1}^T f_t(\theta)$.

Counterexamples. For a fixed constants $d, C_1, C_2 \in \mathbb{N}$ such that C_1 is sufficiently larger than C_2 and a scalar parameter θ consider the following sequential online optimization problem:

$$f_t(\theta) = \begin{cases} C_1\theta, & \text{if } t \text{ mod d} = 1; \\ -C_2\theta, & \text{otherwise.} \end{cases} \tag{5}$$

Reddi et al. (2018) additionally restricted the domain of f to $[-1, 1]$ which makes the optimum equal to -1. However, Adam was show to converge to the value of 1. Similarly, in stochastic optimization we consider the following f_t:

$$f_t(\theta) = \begin{cases} C\theta, & \text{with probability } p = \frac{1+\delta}{C+1}; \\ -\theta, & \text{with probability } 1-p. \end{cases} \tag{6}$$

Here optimum is attained at infinitely large negative value, but Adam increases the value of the parameter.

Algorithm 1 AdaShift (Zhou et al., 2019)

```
Require: $n, \theta_0, g_0, \{f_t(\theta)\}_{t=1}^T, \{\alpha_t\}_{t=1}^T, \{g_i\}_{i=t-n+1}^t, \beta_1, \beta_2$
set $v_0 = 0$
for $t = 1$ to $T$ do
    $g_t = \nabla f_t(\theta_t)$
    $m_t = \sum_{i=0}^{n-1} \beta_1^i g_{t-i} / \sum_{i=0}^{n} \beta_1^i$
    // Iterate over parameter blocks
    for $i = 1$ to $M$ do
        $v_t[i] = \beta_2 v_{t-1}[i] + (1 - \beta_2)\phi(g_{t-1}^2[i])$
        $v_t[i] = v_t[i]/(1 - \beta_2^t)$
        $\theta_t[i] = \theta_{t-1}[i] - \alpha_t/\sqrt{v_t[i]} \cdot g_t[i]$
    end for
end for
```

3 AdaShift: Decorrelation via Shifting

Zhou et al. (2019) showed that the problem of convergence on the synthetic counter examples stated above could be solved by decorrelating m_t and v_t. This idea is implemented by temporarily shifting gradients: the latest n gradients are used to compute the momentum term m_t and older gradients gradients are used to compute v_t. To further decorrelate the two values, a function $\phi : \mathbb{R}^k \to \mathbb{R}$ could be used. This function is applied separately to each block of parameters where a block corresponds to a kernel or a bias in a layer of a network. In all of our experiments this function returns the maximum value of a given vector unless stated otherwise. The pseudocode is presented in algorithm 1. In it we fix the order of taking squares of the gradients and applying the function ϕ: the function is applied to squared gradients as in the code provided by the authors and not the paper. Our implementation has time complexity that is independent of the number of kept gradients n and has linear space requirement in n.

4 Experiments

4.1 Synthetic Examples

We reproduce experiments on both of the synthetic examples and present the results in figure 1. In the stochastic online optimization problem the results are comparable with those reported by the authors: AdaShift moves the parameter in the right direction while also being faster than AMSGrad. Additionally, we note that even on the sequential online optimization problem which does not meet the requirement of gradients being temporarily independent, AdaShift still converges to the optimal value of -1, even though it has larger regret than AMSGrad.

For sequential online optimization problem we set the following values for the optimization problem parameters $C_1 = 1010, C_2 = 10, d = 101$. The hyperparameters of the optimizers are $\beta_1 = 0.9, \beta_2 = 0.99, \varepsilon = 0$, and the learning rate is $\alpha_t = \alpha_0/\sqrt{t}$ where $\alpha_0 = 1$ and t is the iteration number. The number of kept gradients n is equal to 10. For stochastic experiment use the same hyperparameters as the authors, namely, we set $C = 101, \delta = 0.002, \beta_1 = 0, \beta_2 = 0.999, \alpha = 10^{-3}$ without decay and the number of kept gradients $n = 1$.

4.2 MNIST

During our verification of Logistic Regression and Multilayer Perceptrion experiments on MNIST we use the same experiment setup as the authors. For Multilayer Perceptron we use two hidden layers with 256 hidden units with no activation. We compare Adam, AMSGrad and AdaShift in two modes: with reduce-max spatial operation (max-AdaShift) and without spatial operation (non-AdaShift). For both models and all optimizers we set $\beta_1 = 0.0, \beta_2 = 0.999$; the learning rate for Adam, AMSGrad and non-AdaShift was set to 0.001, while for max-AdaShift it was set to 0.01. For AdaShift we use $n = 10$ latest gradients. We train the models for 200 epochs with batch size equal to 64 as in the paper, but all methods converge in less than 10 epochs.

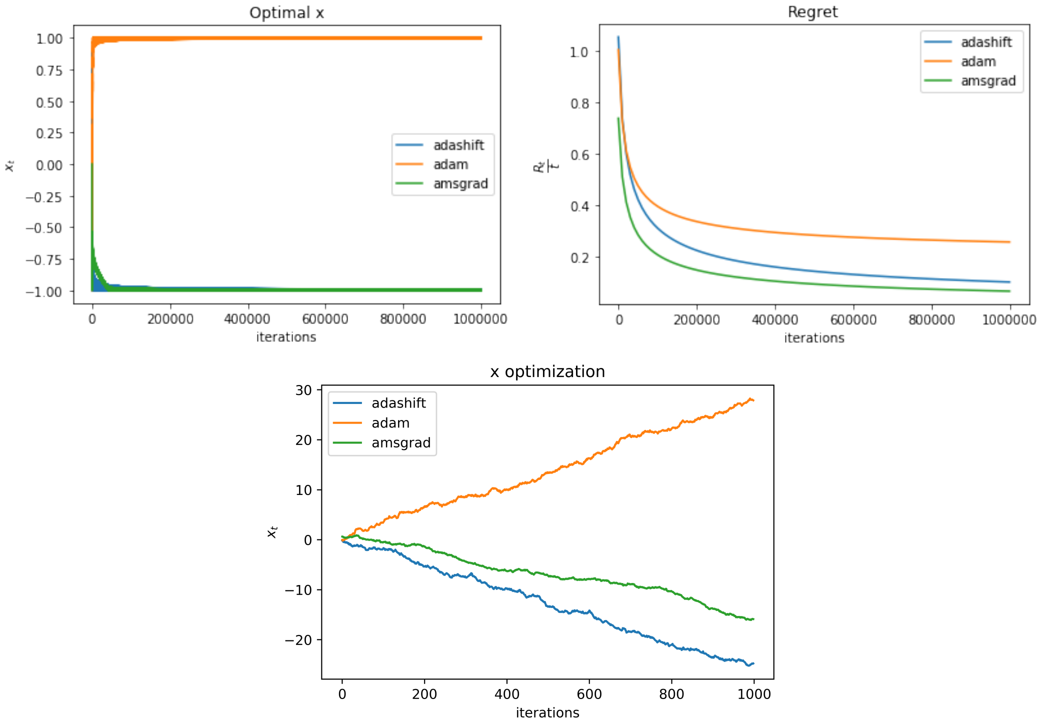

Figure 1: Results on the synthetic examples described in section 2. The plots on the top correspond to the sequential online optimization problem (5) (this experiment was not reported by the authors). The plot on the bottom corresponds to the stochastic optimization problem (6).

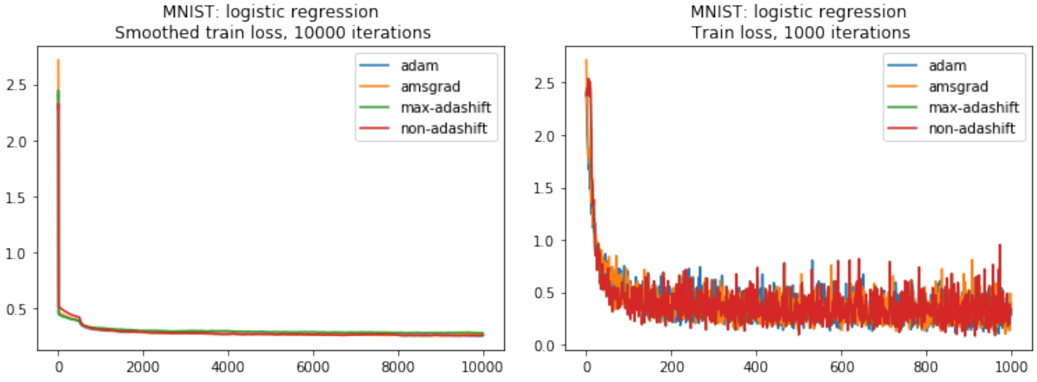

Figure 2: Experiments with logistic regression on MNIST. Curves are smoothed by averaging over a window of fixed size.

In figure 2 we see that the results for Logistic Regression task for all methods are very similar to each other and correspond to those reported in the paper. In the experiment with MLP we verify that non-AdaShift has higher training loss than other methods, but higher test accuracy. The results are noisy, so for the final plots we use smoothing in the same way as the authors.

4.2.1 WGAN-GP

In this experiment we train discriminator with different optimizers and use Adam for generator in WGAN-GP (Gulrajani et al., 2017). In comparison, authors only train discriminator against a fixed generator.

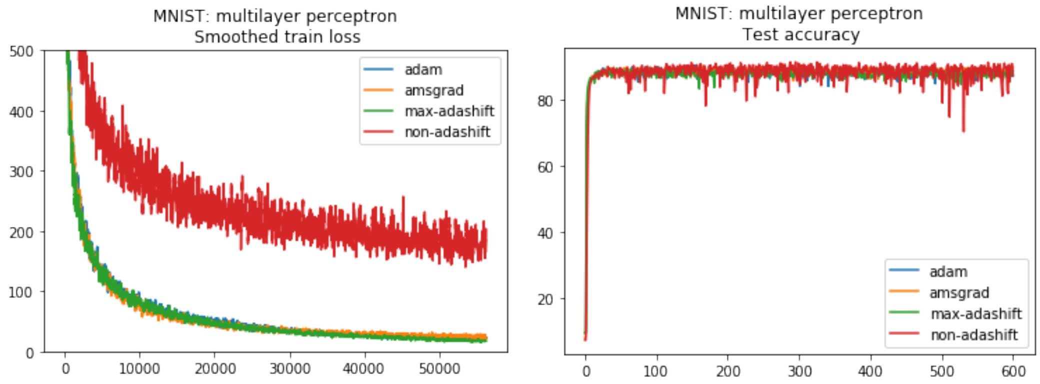

Figure 3: Experiments with MLP.

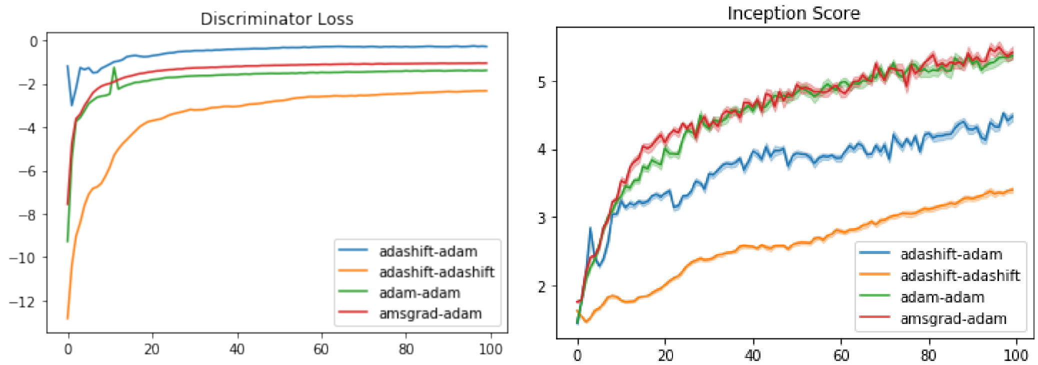

Figure 4: Discriminator loss and Inception Score on the generative model WGAN-GP depending on epoch number. The first optimizer in label corresponds to discriminator and the second one to generator.

WGAN-GP uses Wasserstein distance as a measure of similarity between distributions: $W(p_r, p_g) = \inf_{\gamma \sim \Pi(p_r, p_g)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|]$. WGAN-GP penalizes the model if the gradient norm moves away from its target norm value 1 and the loss function for the discriminator is the following:

$$L(p_r, p_g) = \max_{w \in W} \mathbb{E}_{x\sim p_r}[f_w(x)] - \mathbb{E}_{x\sim p_g}[f_w(x)] + \lambda \mathbb{E}_{x\sim p_x}[(\|\nabla f_w(x)\|_2 - 1)^2], \tag{7}$$

where λ is a penalty coefficient.

We report the discriminator loss and inception score in figure 4. From the figure it is noticeable that for AdaShift the discriminator loss is higher during training while the inception score is lower. We also trained a model in which both generator and discriminator were optimized with different AdaShift optimizers but it had worse inception score than the pair in which AdaShift is used for discriminator and Adam is used for generator.

During training we make 5 discriminator iterations per generator update and set learning rates to be constant and equal to 2e-4 for all optimizers. The values of exponential moving average rates are $\beta_1 = 0, \beta_2 = 0.999$, batch size is 128 and penalty coefficient λ is equal to 10.

We also attempted to reproduce the results reported by the authors. In figure 5 we report the discriminator loss that is a modification of the loss (7). For this result we set batch size to 64, learning rates for Adam and AMSGrad equal to 1e-5 and learning rate for AdaShift to 2e-4. The penalty coefficient λ is set to 0.1. We note that while in the paper Adam outperforms AMSGrad, in our experiment the converse is true. At the same time we confirm that AdaShift achieves the best result.

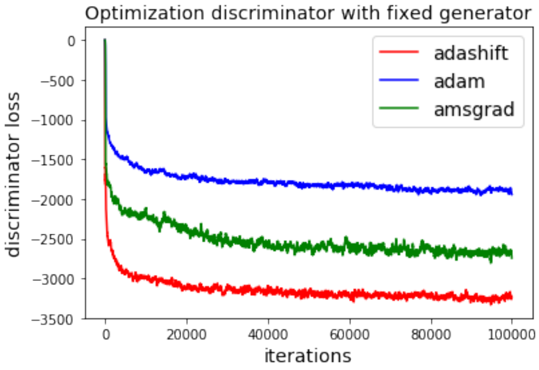

Figure 5: WGAN-GP training discriminator against fixed generator. The curves are smoothed by averaging over a window of fixed size.

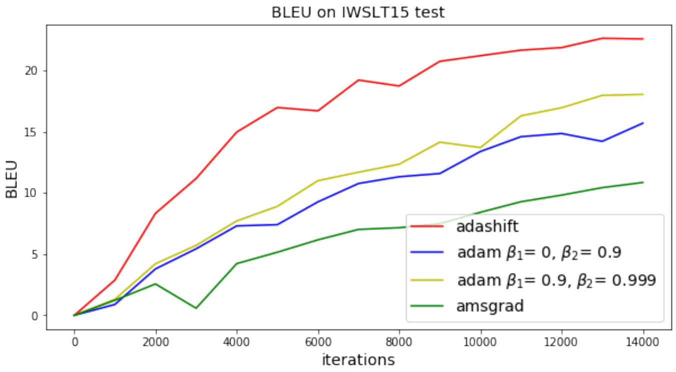

Figure 6: Results on the neural machine translation with seq2seq model.

In this experiment, we note the atypical way in which the authors conducted their experiments with a generative model, specifically, by fixing generator and training only discriminator. Additionally, we note that the penalty coefficient significantly differed from the one used in the paper that proposed this method: authors set λ to 0.1. We also note that authors use maxGP version of the loss function in their code. In our opinion, motivation and discussion of these choices would be beneficial.

5 Neural Machine Translation

We compare Adam, AMSGrad and AdaShift as optimizers for training Neural Machine Translation models. The authors use original implementation of GNMT (Luong et al., 2017), but it is available only in TensorFlow, so we add AMSGrad and AdaShift in OpenNMT-py system (Klein et al., 2017) and use it with similar parameters: we choose LSTM type of cells, bidirectional encoder, two layer in the encoder and the decoder; the size of hidden units and embeddings is 512 and we use Luong attention. For the inference we use beam-search with the size 10. We test the system on IWSLT-15 dataset with Vietnames as the source language and English as the target. For all methods we use $\beta_1 = 0.0, \beta_2 = 0.9$, the learning rate for Adam, AMSGrad was 0.0001, for max-AdaShift the learning rate was 0.01. For AdaShift we use $n = 40$ kept gradients and we notice that with the default value $n = 10$ as used in the previous experiments, training procedure doesn't converge. Additionally, we decided to test Adam optimizer with $\beta_1 = 0.9, \beta_2 = 0.999$, because this setting is more realistic for NMT. In figure 6 it could be seen that while with these hyperparameters Adam does work better, its performance is still significantly worse than that of AdaShift.

As we can see on the figure 6 AdaShift achieves the best result in terms of BLEU score and converges faster than other methods. The results in the paper are similar to ours, but the curves are not exactly the same as in our experiments. The reason can be slight difference in NMT frameworks. It is not clear how many iterations authors used, because on the plot they also have 15000 iterations, but in the code they train the system with 30000 iterations. Our results are similar to theirs with 15000 iterations.

6 Conclusion

In this work we reproduced most of the experiments from the paper and confirmed that AdaShift i) fixes the problem with Adam convergence ii) can outperform other optimizers such as Adam and AMSGrad. We found that in most tasks the proposed method is an efficient and easy way to stabilize the training process. We conducted more realistic experiment with generative models and although in our experiment with WGAN-GP task Wasserstein distance was close to zero with AdaShift, the quality of the generated images was worse than with other optimizers according to the inception score. Experiment with NMT model shows that AdaShift can be the best option in a translation task. Overall, we conclude that AdaShift is a viable alternative to other optimization methods and that it can outperform them significantly on some tasks while solving the convergence problem of Adam.

References

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pp. 5767–5777, 2017.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2014.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. OpenNMT: Open-source toolkit for neural machine translation. In *Proc. ACL*, 2017. doi: 10.18653/v1/P17-4012. URL https://doi.org/10.18653/v1/P17-4012.

Minh-Thang Luong, Eugene Brevdo, and Rui Zhao. Neural machine translation (seq2seq) tutorial. *https://github.com/tensorflow/nmt*, 2017.

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=ryQu7f-RZ.

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. URL https://openreview.net/forum?id=HkgTkhRcKQ.