On the Trend-corrected Variant of Adaptive Stochastic Optimization Methods

Bingxin Zhou*, Xuebin Zheng*
The University of Sydney Business School
The University of Sydney
Sydney, Australia
{bzho3923, xzhe2914}@uni.sydney.edu.au
* *equal contribution*

Junbin Gao
The University of Sydney Business School
The University of Sydney
Sydney, Australia
junbin.gao@sydney.edu.au

***Abstract*—Adam-type optimizers, as a class of adaptive moment estimation methods with the exponential moving average scheme, have been successfully used in many applications of deep learning. Such methods are appealing due to the capability on large-scale sparse datasets with high computational efficiency. In this paper, we present a new framework for Adam-type methods with the trend information when updating the parameters with the adaptive step size and gradients. The additional terms in the algorithm promise an efficient movement on the complex cost surface, and thus the loss would converge more rapidly. We show empirically the importance of adding the trend component, where our framework outperforms the conventional Adam and AMSGrad methods constantly on the classical models with several real-world datasets.**

***Index Terms*—Stochastic Gradient Descent, ADAM, Deep Learning, Optimization**

I. Introduction

Employing first order optimization methods, such as Stochastic Gradient Descent (SGD), is the key of solving large-scale problems. The classic gradient descent algorithm is widely used to update the model parameters, denoted by x

$$x_{t+1} = x_t - \eta \nabla f(x_t), \tag{1}$$

where the gradient is denoted by $\nabla f(x_t)$ and the step size by η. While the method has shown its efficiency for many contemporary tasks, the adaptive variants of SGD outperform the vanilla SGD methods on their rapid training time. Specifically, the step size η is substituted by an adaptive step size $\eta/\sqrt{v_t}$, and v_t is generated from the squared gradient $[\nabla f(x_t)]^2$ where the operation is element-wise.

Several variants of the popular adaptive optimizers can be summarized into such common format. These optimizers share gradients calculation and parameters updating functions, but specify different moving average schemes for calculating the parameter-wise adaptive learning rate $\eta/\sqrt{v_t}$. For example, AdaGrad [1] takes the arithmetic average of historical squared gradients $[\nabla f(x_t)]^2$. Compared with the conventional momentum method, it adapts the learning rate to each parameter to suit the sparse data structure, and thus gains a rapid convergence speed [2]. RMSProp [3] was proposed to reduce the aggressiveness of the decay rate in AdaGrad . The method modifies v_t to the exponentially decayed squared gradients. Similar implementations could also be found in ADADELTA [4]. Instead of the squared gradients, the method applies squared parameter updates to define the adaptive learning rate. As a result, each update guarantees the same hypothetical units as the parameter. Later, Adam [5] modifies RMSProp with the idea from momentum methods [6]. Except for the second moment moving average, the new rule also replaces the gradient $\nabla f(x_t)$ at the end of (1) with the first-moment estimation. The method has practically shown its superiority regarding the converge speed and memory requirement. AMSGrad in [7] applies the maximum of past second-moment estimation to prevent converging to a suboptimal solutions where Adam could potentially trapped in. While the aforementioned methods are the most famous frameworks, many variants are also proposed in the last few years [8]–[10]. In general, we call such adaptive methods with shared structures as Adam-type optimizers.

So far, the adaptive methods with exponential moving average gradients have gained great attention with huge success in many deep learning tasks. However, it remains unsolved whether the simple exponential smoothing results or the level information is sufficient in capturing the landscape of the cost surface. When clear upward or downward pattern could be recognized within the moving routine, it is suggested to add a trend term on top of the level-only information.

In this paper, we propose the notion of trend corrected exponential smoothing to modify the conventional application of exponential moving average in optimizers. We name the **T**rend-corrected variant of Adam as *AdamT* and the proposed method converges consistently as Adam at $\mathcal{O}(\sqrt{T})$. To the best of our knowledge, this research work is the first to apply the trend-corrected features on gradients scaling and parameters updating in the literature. We testify our framework on the vanilla Adam method for rule implementation and performance comparison. In addition, we provide supplementary evaluations on modified AMSGrad to avoid the potential suboptimal problems suffered by Adam. By using the same naming convention, we call the trend-corrected AMSGrad as *AMSGradT*. The empirical results on some typical machine learning problems demonstrate the convergence and generalization ability of AdamT and AMSGradT in both convex and non-convex settings. It shall be emphasized that our framework is universally implementable for all adaptive update methods that involve the exponential moving average term, including but not restricted to ADADELTA, RMSProp, AdaMAX and other well-recognized methods.

For the remainder of the paper, we present in Section II the fundamental idea of Adam, AMSGrad and Holt's linear methods. In Sections III and IV, we detail the update rules of our proposed methods and experimental analysis respectively. In addition, Section V reviews recent developments of Adam-type optimizers. While many of them focus more on non-convex optimizations, there is a potential to incorporate our methods with such frameworks and this extension is expected for future works.

II. Preliminary

A. Adaptive Methods with Exponential Moving Averages

For adaptive gradient descent methods, the update rule can be written as

$$x_{t+1} = x_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$$

where m_t is the gradient updates, and conventionally it is defined to be the last gradient value $\nabla f(x_t)$. To prevent zero division, a smoothing term ϵ is included in the denominator.

We initially focus our analysis and modifications on Adam [5], one of the most popular optimizers in the last few years. The adaptive step size involves the squared gradients

$$v_t = \alpha v_{t-1} + (1-\alpha) \nabla f(x_t) \circ \nabla f(x_t) \tag{2}$$

where the operation "$\circ$" denotes an element-wise multiplication. In terms of the gradient m_t, Adam takes the exponentially weighted average of all previous gradients instead of solely relying on the last gradient value $\nabla f(x_t)$

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) \nabla f(x_t). \tag{3}$$

While the two moment estimates from (2) & (3) could potentially counteract towards zero, the series $\hat{m}_t$ and $\hat{v}_t$ are considered for bias-correction. Formally, the rules are defined as:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}; \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}.$$

To allow rare mini-batches provide informative gradients for a promising optimal solution, AMSGrad [7] includes long-term memory in squared gradients. Instead of using the latest v_t, AMSGrad employs the following update rule for the second moment estimate

$$v_t^{\max} = \max(v_{t-1}^{\max}, v_t),$$

and then use $v_t^{\max}$ to make the learning rate adaptive. As a result, a non-increasing step size is utilized during the updating process, which avoids the problems suffered from the conventional adaptive learning methods, including Adam.

B. Trend Corrected Exponential Smoothing

The idea of extracting a smoothed new point from all the previous information is called the exponential weighted moving average. The method was extended in [11] by including the trend behaviours within the series, namely trend-corrected exponential smoothing or Holt's linear method. Consider a time series $\{y_t\}$ for $t = 1, 2, \ldots, T$, our target is to find the smoothing results $\widehat{y_{t+1|t}}$. Holt's linear method formulates the conditional forecasting by summing up two exponential smoothing equations

$$\begin{aligned} \widehat{y_{t+1|t}} &= \ell_t + b_t \\ \ell_t &= \beta(\ell_{t-1} + b_{t-1}) + (1-\beta) y_t \\ b_t &= \gamma b_{t-1} + (1-\gamma)(\ell_t - \ell_{t-1}). \end{aligned}$$

For a new estimation, we first update the level term ℓ_t with the weighted average of the last observation y_t and its estimation $\widehat{y_{t|t-1}}$. The trend term b_t is updated afterwards as the weighted average of the estimated trend $\ell_t - \ell_{t-1}$ and its previous estimation b_{t-1}. The smoothing parameters for the level and the trend are denoted as β and γ respectively. Both values could be selected between 0 and 1.

Including a damping factor ϕ is also suggested in [12], so that the Holt's linear method with damping factor has the following form

$$\begin{aligned} \widehat{y_{t+1|t}} &= \ell_t + \phi b_t \\ \ell_t &= \beta(\ell_{t-1} + \phi b_{t-1}) + (1-\beta) y_t \\ b_t &= \gamma \phi b_{t-1} + (1-\gamma)(\ell_t - \ell_{t-1}). \end{aligned}$$

The damped method will be identical to Holt's linear method if $\phi = 1$, and will be the same as simple exponential moving average method if we set $\phi = 0$. When ϕ is restricted to be positive, the damping factor could be used to control the significance of the trend component.

The damped trend methods are considerably popular for forecasting tasks [13]. Such methods inherent both level and trend information from historical series, while staying flexible enough to adjust the influence of the trend term via ϕ. On top of that, involving the damped factor could to some extend reduce the volatility of the smoothed line.

III. Methodology

We introduce the trend-corrected variants of two adam-type stochastic optimization methods, namely *AdamT* and *AMSGradT*. The proposed methods are based on [5] and [7] with added Holt's linear trend information for both of the first moment estimate and the second raw moment estimate. Specifically, we use trend-corrected exponential weighted moving averages in the final parameter update step instead of the level-only estimates used in Adam and AMSGrad.

A. Algorithm for AdamT

Consider the gradient of a stochastic objective function $f(x)$ evaluated at T iterations as a time series $\{\nabla f(x_t)\}$ for $t = 1, 2, \ldots, T$. According to the Holt's linear trend method illustrated in Section II-B, we write two series $\{\ell_t^m\}$ and $\{b_t^m\}$

as the exponential weighted moving averages which estimate the level and trend information of the first moment $\nabla f(x_t)$:

$$\ell_t^m = \beta_1 m_{t-1} + (1-\beta_1)\nabla f(x_t) \tag{4}$$
$$b_t^m = \gamma_1\phi_1 b_{t-1}^m + (1-\gamma_1)(\ell_t^m - \ell_{t-1}^m) \tag{5}$$
$$m_t = \ell_t^m + \phi_1 b_t^m \tag{6}$$

where β_1, γ_1 and ϕ_1 have the same functionality as explained in Section II-B and these are regarded as hyperparameters in our algorithm. Equation (6) combines the level and the trend information of first moment, which will be used for calculating the final update rule and the trend-corrected level estimates. The procedures for the second raw moment $\nabla f(x_t) \circ \nabla f(x_t)$ is analogous:

$$\ell_t^v = \beta_2 v_{t-1} + (1-\beta_2)\nabla f(x_t) \circ \nabla f(x_t) \tag{7}$$
$$b_t^v = \gamma_2\phi_2 b_{t-1}^v + (1-\gamma_2)(\ell_t^v - \ell_{t-1}^v) \tag{8}$$
$$v_t = \ell_t^v + \phi_2 b_t^v \tag{9}$$

The hyperparameters β_2, γ_2 and ϕ_2 here share the same corresponding meanings as before. The moving averages $\{\ell_t^v\}$ and $\{b_t^v\}$ estimate the level and trend of the second raw moment respectively. The term v_t combines these two information, which will be used in the calculations of final update rule and trend-corrected level estimates of the second raw moment.

In our algorithm, we set the initial values of the series $\{\ell_t^m\}, \{\ell_t^v\}, \{b_t^m\}$ and $\{b_t^v\}$ to be zero vectors, so that $\ell_0^m = \ell_0^v = b_0^m = b_0^v = 0$. The series $\{m_t\}$ and $\{v_t\}$, as a result, are also initialized as zero vectors. As observed in [5], the exponential weighted moving averages could bias towards zero, especially during the early training stage. We perform the bias correction for the two level estimates $\{\ell_t^m\}$ and $\{\ell_t^v\}$ by following [5]. For the two trend estimates $\{b_t^m\}$ and $\{b_t^v\}$, we correct the bias in a different way by taking into account the effect of damping parameters (ϕ_1, ϕ_2). Thus, the bias-corrected version of the series $\{m_t\}$ and $\{v_t\}$ can be written as:

$$\hat{m}_t = \frac{\ell_t^m}{1-\beta_1^t} + \frac{(1-\gamma_1\phi_1)b_t^m}{(1-\gamma_1)(1-(\gamma_1\phi_1)^t)} \tag{10}$$
$$\hat{v}_t = \frac{\ell_t^v}{1-\beta_2^t} + \frac{(1-\gamma_2\phi_2)b_t^v}{(1-\gamma_2)(1-(\gamma_2\phi_2)^t)}. \tag{11}$$

The justification for the two bias-corrected trend estimates $\{b_t^m\}$ and $\{b_t^v\}$ is provided below, where we have to take into account the effect of the corresponding damping factors. Here, we give the justification for the trend estimate $\{b_t^m\}$, and the procedures for $\{b_t^v\}$ is analogous. Note that we can write the trend estimate b_t^m into the following compact summation form:

$$\begin{aligned} b_t^m &= \gamma_1\phi_1 b_{t-1}^m + (1-\gamma_1)(\ell_t^m - \ell_{t-1}^m) \\ &= (1-\gamma_1)\sum_{i=1}^{t}(\gamma_1\phi_1)^{t-i}(\ell_i^m - \ell_{i-1}^m). \end{aligned}$$

To find how the expectation of the trend estimates b_t^m relates to the expectation of the difference between the level estimates at successive timesteps $(\ell_t^m - \ell_{t-1}^m)$, we take the expectation for both sides of the above equation:

$$\begin{aligned} \mathbb{E}[b_t^m] &= \mathbb{E}[(1-\gamma_1)\sum_{i=1}^{t}(\gamma_1\phi_1)^{t-i}(\ell_i^m - \ell_{i-1}^m)] \\ &= (1-\gamma_1)\sum_{i=1}^{t}(\gamma_1\phi_1)^{t-i}\mathbb{E}[(\ell_i^m - \ell_{i-1}^m)] \\ &= \mathbb{E}[(\ell_t^m - \ell_{t-1}^m)](1-\gamma_1)\sum_{i=1}^{t}(\gamma_1\phi_1)^{t-i} + \zeta, \end{aligned}$$

where ζ can be considered as a small constant, since the factor $(\gamma_1\phi_1)^{t-i}$ will be tiny if the associated expectation $\mathbb{E}[(\ell_i^m - \ell_{i-1}^m)]$ is too far away in the past in the case that $\mathbb{E}[(\ell_i^m - \ell_{i-1}^m)]$ is non-stationary. If $\mathbb{E}[(\ell_i^m - \ell_{i-1}^m)]$ is stationary, the constant ζ will be zero. To further simplify the above equation, we apply the formula for the sum of geometric sequence:

$$\mathbb{E}[b_t^m] = \mathbb{E}[(\ell_t^m - \ell_{t-1}^m)](1-\gamma_1)\left(\frac{1-(\gamma_1\phi_1)^t}{1-\gamma_1\phi_1}\right) + \zeta.$$

This suggests that we can use the term $(1-\gamma_1)[1-(\gamma_1\phi_1)^t]/[1-\gamma_1\phi_1]$ to correct the bias and close the discrepancy between the above two expectations at the presence of the damping factor ϕ_1.

The final adaptive update rule is similar to Adam with the bias-corrected first moment estimate and the second raw moment estimate:

$$x_{t+1} = x_t - \frac{\eta}{\sqrt{|\hat{v}_t|}+\epsilon}\hat{m}_t \tag{12}$$

where ϵ is a positive tiny number added in the denominator to avoid zero-division case. Please note that the series $\{\hat{m}_t\}$ and $\{\hat{v}_t\}$ in AdamT are different from that of Adam. The two series are trend-corrected (also bias-corrected) estimates of both moments. Also, we use the absolute value of the series $\{\hat{v}_t\}$ under the square root in the denominator due to the possible negative values from the series $\{\hat{v}_t\}$.

The direction of the effective step $\Delta_t = \eta \cdot \hat{m}_t/\sqrt{|\hat{v}_t|}$ (with $\epsilon = 0$) in the parameter space depends on the joint effect of the first moment level and trend estimates. In the update rule (12), we only care about the magnitude of $\hat{v}_t$ by taking the absolute value and thus the ratio $\hat{m}_t/\sqrt{|\hat{v}_t|}$ can be seen as a signal-to-noise ratio. Note that the effective step Δ_t in our algorithm is also invariant to the scale of the gradients. Specifically, re-scaling the gradients $\nabla f(x_t)$ with a factor c will scale ℓ_t^m and b_t^m by a factor c, and will scale ℓ_t^v and b_t^v by a factor c^2. This results in scaling $\hat{m}_t$ and $\hat{v}_t$ by a factor c and c^2 respectively, and finally cancel out in the parameter update rule $(c \cdot \hat{m}_t)/(\sqrt{|c^2 \cdot \hat{v}_t|}) = \hat{m}_t/\sqrt{|\hat{v}_t|}$.

Note that our proposed method AdamT has two extra computational steps, that is (5) & (6). However, the computational complexity of these two steps is almost linear in time. Therefore, we can conclude that AdamT yields a superior performance compared with Adam (the results will be shown in the experiment section) with a minimal additional computational cost.

In our algorithm, we set the hyperparameters $\beta_1, \gamma_1, \beta_2, \gamma_2$ according to the suggestions in [5]. The smoothing parameters for the first moment estimates are set to 0.9, that is $\beta_1 = \gamma_1 = 0.9$, while the smoothing parameters for the second raw moment estimates are set to 0.999, that is $\beta_2 = \gamma_2 = 0.999$. We empirically find that the good default values of the two damping parameters can be set to $\phi_1 = \phi_2 = 0.5$. The pseudo-code of our AdamT is provided in Algorithm 1.

Algorithm 1 The **Adam** optimizer modified with Holt's Linear **T**rend method. Empirically suggested default values for the hyperparameters are $\beta_1 = \gamma_1 = 0.9, \beta_2 = \gamma_2 = 0.999, \phi_1 = \phi_2 = 0.5, \eta = 0.0001$. All of the operations on vectors are element-wise.

```
Input: Learning rate $\eta$; Smoothing factors $\beta_1$, $\gamma_1$, $\beta_2$, $\gamma_2$;
    Damping factors $\phi_1, \phi_2$; Noisy objective function $f(x)$
    with parameters $x$
Output: Optimal model parameters $x^*$
 1: Initialization
        $x_1$: Initial parameter values
        $\ell_0^m \leftarrow 0$: Initial first moment level estimate
        $\ell_0^v \leftarrow 0$: Initial second raw moment level estimate
        $b_0^m \leftarrow 0$: Initial first moment trend estimate
        $b_0^v \leftarrow 0$: Initial second raw moment trend estimate
 2: for $t = 1$ to $T$ do
 3:     $\ell_t^m \leftarrow \beta_1 m_{t-1} + (1-\beta_1)\nabla f(x_t)$
 4:     $b_t^m \leftarrow \gamma_1\phi_1 b_{t-1}^m + (1-\gamma_1)(\ell_t^m - \ell_{t-1}^m)$
 5:     $\ell_t^v \leftarrow \beta_2 v_{t-1} + (1-\beta_2)\nabla f(x_t) \circ \nabla f(x_t)$
 6:     $b_t^v \leftarrow \gamma_2\phi_2 b_{t-1}^v + (1-\gamma_2)(\ell_t^v - \ell_{t-1}^v)$
 7:     $\hat{m}_t \leftarrow \ell_t^m/(1-\beta_1^t) + [(1-\gamma_1\phi_1)b_t^m]/[(1-\gamma_1)(1-(\gamma_1\phi_1)^t)]$
 8:     $\hat{v}_t \leftarrow \ell_t^v/(1-\beta_2^t) + [(1-\gamma_2\phi_2)b_t^v]/[(1-\gamma_2)(1-(\gamma_2\phi_2)^t)]$
 9:     $x_{t+1} \leftarrow x_t - \eta\hat{m}_t/(\sqrt{|\hat{v}_t|} + \epsilon)$
10: end for
11: return $x^*$
```

B. Algorithm for AMSGradT

We introduce here the variant AMSGradT to overcome the non-convergence and suboptimal problems of the proposed AdamT by using the same technique illustrated in [7]. These issues of AdamT are inherited from the base approach Adam [5]. Essentially, we follow the same procedures in (4)-(6) and (10) for the first moment estimation with bias-correction. For the second raw moment estimate (7)-(9), we add one more step after (9):

$$v_t^{\max} = \max(v_{t-1}^{\max}, v_t),$$

where v_t is the second raw moment estimation from (9) before bias correction. We set the initial value of the series $\{v_t^{\max}\}$ to be $v_0^{\max} = 0$. As a result, the new bias-correction step which is used to replace (11) can be written as

$$\hat{v}_t^{\max} = \frac{(\ell_t^v)^{\max}}{1-\beta_2^t} + \frac{(1-\gamma_2\phi_2)(b_t^v)^{\max}}{(1-\gamma_2)(1-(\gamma_2\phi_2)^t)},$$

where $(\ell_t^v)^{\max}$ and $(b_t^v)^{\max}$ are the corresponding level and trend information that used to calculate $v_t^{\max}$. Finally, we replace $\hat{v}_t$ in the final update rule (12) with the bias-corrected estimates $\hat{v}_t^{\max}$.

IV. Experiments

We evaluate the two proposed algorithms *AdamT* and *AMSGradT* on both convex and non-convex real-world optimization problems with several popular types of machine learning models. The models we considered in the experiments include logistic regression which has a well-known convex loss surface, and different neural network models, including feedforward neural networks, convolutional neural networks and variational autoencoder. Neural Networks with non-linear activation function typically have an inherent non-convex loss surface which is more challenging for an optimization method.

We compare our methods with the baseline approaches Adam [5] and AMSGrad [7], and then demonstrate the effectiveness of the trend information of the gradients infused in our proposed algorithms. The experiment results show that our methods AdamT and AMSGradT converge more quickly and reach a better minimum point than Adam and AMSGrad respectively. The observation evidences that the added trend information effectively helps AdamT and AMSGradT to better capture the landscape of loss surface.

In each of the following experiments, we use the same set of initial values for the models, so that the initial model losses (the loss value at epoch $= 0$) are identical for all the optimization methods. In terms of the hyperparameters, all the smoothing parameters (β_1, β_2 in Adam & AMSGrad and $\beta_1, \beta_2, \gamma_1, \gamma_2$ in AdamT & AMSGradT) are set at their corresponding default values which are provided in Algorithm 1. The damping factors (ϕ_1, ϕ_2) are searched within the range $[0.1, 1.0)$ and the learning rate η is also tuned through a grid search $\{1e-4, 5e-4, 1e-3, 5e-3\}$ to produce the best results for all of the optimizers. All the experiments and optimizers are written in PyTorch and the implementations of our proposed optimizers can be found at https://github.com/xuebin-zh/AdamT.

For the sake of saving space, we only visualize the convergence of Adam and AdamT in the paper for comparisons. The relative convergence performance of AMSGrad and AMSGradT is almost the same in the corresponding experiments.

A. Logistic Regression for Fashion-MNIST

We first evaluate our methods on the logistic regression (LR) for multi-class classification problem with Fashion-MNIST dataset [14] which is a MNIST-like dataset of fashion products. The dataset has $60,000$ training samples and $10,000$ testing samples. Each of them has 28×28 pixels. Each of the samples is classified into one of the 10 fashion products. The cross-entropy loss function has a well-behaved convex surface. The learning rate η is set to be constant during the training procedure. We use minibatch training with size set to 128.

The training results are reported in Fig. 1. Since the superiority of our method over Adam is relatively small in

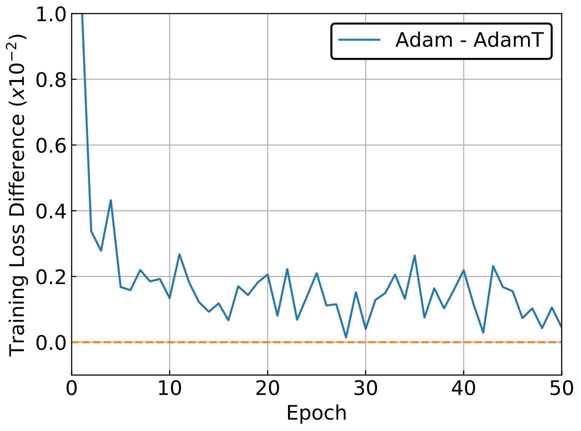

Fig. 1: Training loss difference of the logistic regression on Fashion-MNIST dataset for classification task.

this experiment, the plot of loss value against epoch cannot visualize the difference. Instead, we plot the loss difference of the two optimizers, which is ($Loss_{\text{Adam}} - Loss_{\text{AdamT}}$) against training epoch. The difference above zero reflect the advantage of our AdamT. Fig. 1 indicates that AdamT converges faster at the early training stage and constantly outperforms Adam during the rest of the training phase, though the advantage is relatively small in this experiment. The loss surface of logistic regression is convex and well-behaved so that the trend information of AdamT cannot further provide much useful information for optimization, which results in a small advantage in this experiment.

B. *Feedforward Neural Networks for SVHN*

To investigate the performance on non-convex objective functions, we conduct the experiment with feedforward neural networks on *The Street View House Numbers* (SVHN) dataset [15] for a digit classification problem. We pre-process this RGB image dataset into grayscale for dimension reduction by taking the average across the channels for each pixel in the image. The samples are 32×32 grayscale images. The neural network used in this experiment has two fully-connected hidden layers, each of which has $1,400$ hidden units and ReLU activation function is used for the two hidden layers. The softmax cross-entropy loss function is used for training.

To evaluate the performance of the optimizers in noisy settings, we apply a stochastic regularization method in the model for a separate experiment. Specifically, we include two dropout layers [16], where one is applied between the two hidden layers and the other one is used before the output layer. The dropout probability is set to 0.5 for both of the two dropout layers. In the experiments, we use a constant learning rate η and minibatch training with size set to 128.

We examine the convergence with and without dropout layers for the two optimizers. According to Fig. 2, we find that AdamT outperforms Adam obviously. In terms of the training process, AdamT yields a faster convergence and reaches a better position than Adam for the models, both with and without dropout layers. The superior performance of AdamT is also shown in the test phase, which demonstrates that AdamT also has a better generalization ability than Adam. For the model without dropout, the test results show that the model is prone to over-fitting and our method performs on a par with Adam. Comparing to logistic regression, the loss surface in this experiment becomes complex and non-convex. The trend estimates of the gradients from AdamT can provide more meaningful information of the landscape of the loss surface, and it encourages a better performance on the AdamT.

C. *Convolutional Neural Networks for CIFAR-10*

Convolutional neural network (CNN) is the main workhorse for Computer Vision tasks. We train a CNN model on standard CIFAR-10 dataset for a multi-class classification task. The dataset contains $50,000$ training samples and $10,000$ test samples, and each sample is an RGB 32×32 image. We pre-process the dataset by normalizing the pixel values to the range $[-1, 1]$ for a more robust training. The CNN model employed in this experiment is similar to the model used in [7], which has the following architecture. There are 2 stages of alternating convolution and max pooling layers. Each convolution layer has 64 channels and kernel size 5×5 with stride 1. Each max pooling layer is applied with a kernel size of 2×2. After that, there is a fully-connected layer with 600 hidden units and a dropout probability 0.5 followed by the output layer with 10 units. We use ReLU for the activation function and softmax cross-entropy for the loss function. The model is trained with a tuned constant learning rate and minibatch size 128 same as the previous experiments.

We evaluate the test loss after each epoch during the training procedure and cease the training once the test loss of the model starts to increase. The training curves are reported in Fig. 3. We can observe that the proposed AdamT clearly excels Adam on the training loss, and this superiority translates into an advantage of AdamT during the early stage of the test loss.

D. *Deep Generative Models For MNIST*

Variational Autoencoder (VAE) [17], [18] is one of the most popular deep generative models for density estimation and image generation. In this experiment, we train a VAE model on the standard MNIST dataset which contains $60,000$ training samples and $10,000$ test samples. Each sample is one 28×28 black-and-white image of the handwritten digit. The VAE model used in this experiment exactly matches the architecture presented in [17]: Gaussian encoder and Bernoulli decoder, both of which are implemented by feedforward neural networks with single hidden layer and there are 500 hidden units in each hidden layer. We employ the hyperbolic tangent activation function for the model and set the dimensionality of the latent space as 20. We use the constant learning rate and set the minibatch size to 128.

We examine the Evidence Lower Bound (ELBO) of the training and test phases for the two optimizers. See Fig. 4 for

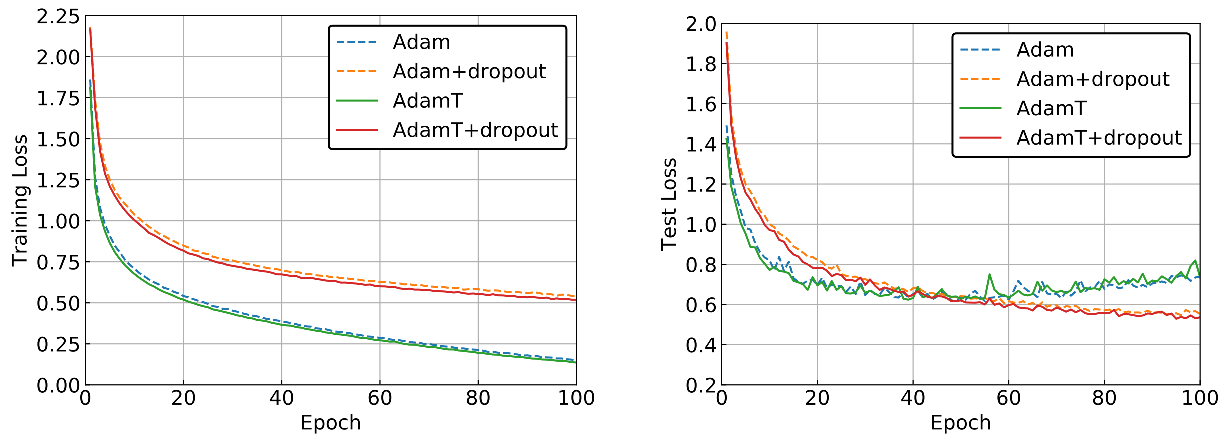

Fig. 2: Training loss (left) and test loss (right) of the feedforward neural network on SVHN dataset for a classification problem. The model architecture is fc1400-fc1400.

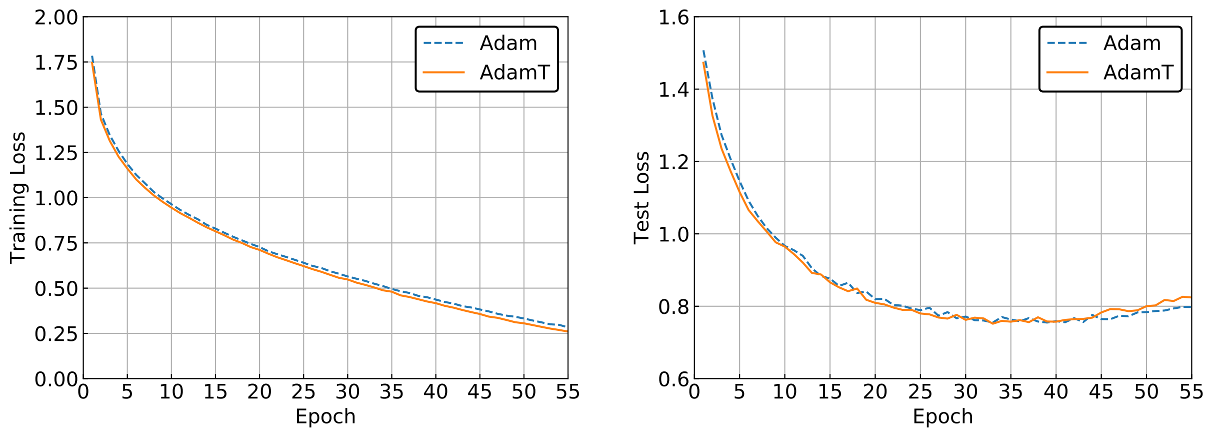

Fig. 3: Training loss (left) and test loss (right) of the convolutional neural network on CIFAR-10 dataset for a classification task. The model architecture is c64-c64-fc600.

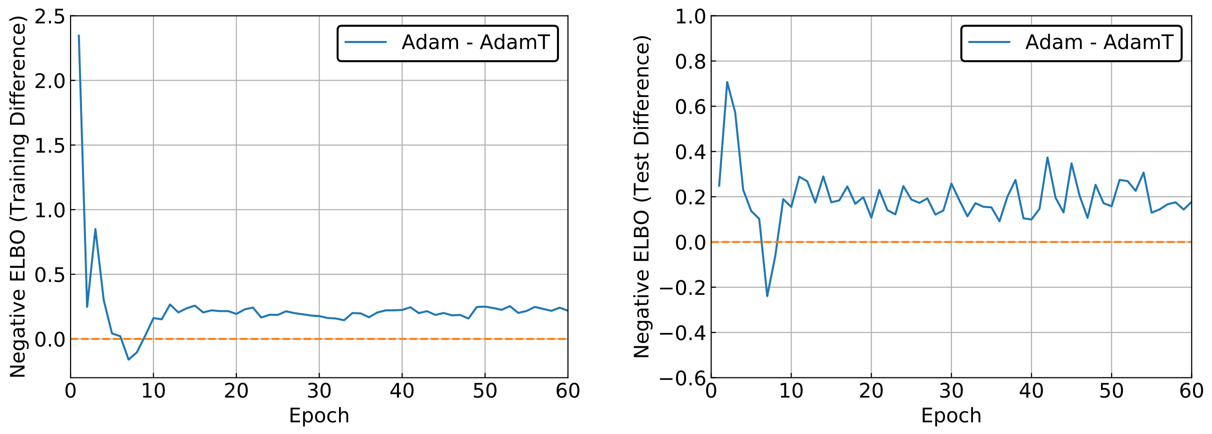

Fig. 4: Training difference of negative ELBO (left) and test difference of negative ELBO (right) of the variational autoencoder on MNIST for density estimation.

TABLE I: The final training loss and test loss of each experiment. The reported numbers are the averages over 10 repeated experiments with corresponding standard deviations.

	Training Loss		Test Loss	
	Adam	Adam+Trend	Adam	Adam+Trend
LR	0.3645 ± 0.0005	$\mathbf{0.3634 \pm 0.0003}$	0.4403 ± 0.0028	$\mathbf{0.4390 \pm 0.0025}$
FNN*	0.1468 ± 0.0036	$\mathbf{0.1386 \pm 0.0018}$	$\mathbf{0.7492 \pm 0.0288}$	0.7639 ± 0.0140
FNN	0.5390 ± 0.0046	$\mathbf{0.5173 \pm 0.0045}$	0.5478 ± 0.0066	$\mathbf{0.5376 \pm 0.0073}$
CNN	0.2661 ± 0.0039	$\mathbf{0.2488 \pm 0.0034}$	$\mathbf{0.8201 \pm 0.0111}$	0.8228 ± 0.0182
VAE	244.1535 ± 0.1202	$\mathbf{244.0036 \pm 0.1398}$	245.9947 ± 0.1996	$\mathbf{245.8023 \pm 0.1600}$
	AMSGrad	AMSGrad+Trend	AMSGrad	AMSGrad+Trend
LR	0.3686 ± 0.0005	$\mathbf{0.3676 \pm 0.0003}$	0.4390 ± 0.0028	$\mathbf{0.4374 \pm 0.0020}$
FNN*	0.1622 ± 0.0029	$\mathbf{0.1532 \pm 0.0022}$	$\mathbf{0.6982 \pm 0.0186}$	0.7060 ± 0.0142
FNN	0.5419 ± 0.0047	$\mathbf{0.5261 \pm 0.0052}$	0.5475 ± 0.0040	$\mathbf{0.5422 \pm 0.0037}$
CNN	0.3055 ± 0.0108	$\mathbf{0.2968 \pm 0.0090}$	0.7974 ± 0.0149	$\mathbf{0.7804 \pm 0.0118}$
VAE	246.4323 ± 0.1369	$\mathbf{246.3635 \pm 0.1982}$	248.1823 ± 0.1715	$\mathbf{248.1336 \pm 0.2331}$

TABLE II: The final training and test classification accuracy of each experiment. The reported numbers are the averages over 10 repeated experiments with corresponding standard deviations.

	Training Accuracy		Test Accuracy	
	Adam	Adam+Trend	Adam	Adam+Trend
LR	0.8732 ± 0.0016	$\mathbf{0.8740 \pm 0.0010}$	0.8063 ± 0.0438	$\mathbf{0.8188 \pm 0.0337}$
FNN*	0.9599 ± 0.0058	$\mathbf{0.9625 \pm 0.0037}$	$\mathbf{0.8792 \pm 0.0224}$	0.8771 ± 0.0237
FNN	0.8881 ± 0.0027	$\mathbf{0.8933 \pm 0.0033}$	0.8542 ± 0.0349	$\mathbf{0.8646 \pm 0.0104}$
CNN	0.9674 ± 0.0001	$\mathbf{0.9703 \pm 0.0011}$	0.6563 ± 0.0313	$\mathbf{0.6875 \pm 0.0625}$
VAE	-	-	-	-
	AMSGrad	AMSGrad+Trend	AMSGrad	AMSGrad+Trend
LR	0.8722 ± 0.0015	$\mathbf{0.8732 \pm 0.0010}$	0.8188 ± 0.0337	$\mathbf{0.8250 \pm 0.0250}$
FNN*	0.9551 ± 0.0049	$\mathbf{0.9600 \pm 0.0032}$	0.8958 ± 0.0264	$\mathbf{0.9000 \pm 0.0292}$
FNN	0.8834 ± 0.0018	$\mathbf{0.8864 \pm 0.0021}$	0.8458 ± 0.0250	$\mathbf{0.8563 \pm 0.0197}$
CNN	0.9529 ± 0.0045	$\mathbf{0.9563 \pm 0.0036}$	0.6688 ± 0.0563	$\mathbf{0.6875 \pm 0.0484}$
VAE	-	-	-	-

the convergence results. Due to the scale issue, we plot the difference between the ELBOs produced by the two optimizers. Similar to the first experiment, we plot the difference value $(ELBO_{\text{Adam}} - ELBO_{\text{AdamT}})$ against the epoch for training and testing. We observe that our AdamT has a much faster convergence at the early stage of training than Adam and constantly excels Adam during the rest of the training phase. The superior performance of AdamT in this experiment also translates into a clear advantage in the test phase.

E. Quantitative Evaluations

We compare our proposed algorithms AdamT and AMSGradT with the corresponding baseline approaches Adam and AMSGrad respectively based on the final training loss, test loss, classification performance (except for the VAE model) in each experiment. The results recorded in Table I and Table II are the average values along with the standard deviations calculated over 10 repeated experiments with random initializations. FNN* denotes the feedforward neural networks without dropout layers while FNN represents the same model equipped with dropout layers. The results show that our proposed two trend-corrected variants have a superior performance over their corresponding baseline methods in most of the conducted experiments under different evaluation metrics, except for the models (FNN* and CNN) which are prone to over-fitting. From the documented experiment results, we conclude that the additional trend estimates can provide meaningful information of the landscape of the loss surface and thus yield a better performance for the trend-corrected schemes.

V. Related Works

We consider the class of adaptive moment estimation methods with exponential moving average scheme as Adam-type learning algorithms. The fundamental idea was proposed in [5] and quickly extended to many variants. Some examples include AdaMax [5], Nadam [8] and AdamW [9].

Despite the efficiency in practice, the problematic short-term memory of the gradients prevent the conventional Adam-type methods from a promising global convergences [7]. For the convex settings, they proposed AMSGrad that promises a global optimization with a comparable performance. Except for some other recent studies for convex optimization [19]–[21], several works developed optimization methods for non-convex problems. Padam [22], [23] introduces a partial adaptive parameter to interpolate between SGD with momentum and Adam, so that adjacent learning rates could decrease smoothly. AdaUSM [24] appends the idea of unified momentum for non-decreasing sequence of weights. AdaFom [25] obtains first-order stationary by taking simple average on the second moment estimation. More conditions for pursuing global convergence were summarized in [26], basing on the currently successful variants.

VI. Discussion and Conclusion

In this work, we propose a new scheme to calculate the adaptive step size with trend-corrected exponential smoothing. The effectiveness of incorporating the trend information is investigated on the plain Adam method, which is the fundamental form of all Adam-type methods proposed in the literature as well as one of the most popular and widely used optimizers in practice with many irreplaceable advantages. On top of that, we also testify the contribution of the modified component on AMSGrad. Empirical results in Section IV demonstrate a performance gain on both optimizers in terms of convergence speed and robustness.

We leave some potentials for future developments. First, although we focused primarily on Adam, we believe that similar ideas could also be extended to other popular adaptive gradient methods for theoretical and experimental analysis. Also, despite the computational feasibility, the initial implementation on Adam inherits its non-converge flaw. While this work focus on investigating the effectiveness of adding additional trend scheme instead of fixing the inherent issues, we try the modification with AMSGrad, as one of the solutions, to demonstrate the potential to extend our framework to optimizers that promise convergence on convex settings. For non-convex scenarios, some potential works in the recent literature are discussed in Section V.

References

[1] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[2] S. Ruder, "An overview of gradient descent optimization algorithms," *preprint arXiv:1609.04747*, 2016.

[3] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural Networks for Machine Learning*, vol. 4, no. 2, pp. 26–31, 2012.

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

[5] D. P. Kingma and J. Ba, "ADAM: A method for stochastic optimization," in *Proceedings of International Conference on Learning Representation (ICLR)*, 2015.

[6] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural Networks*, vol. 12, no. 1, pp. 145–151, 1999.

[7] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of ADAM and beyond," in *Proceedings of International Conference on Learning Representation (ICLR)*, 2018.

[8] T. Dozat, "Incorporating Nesterov momentum into ADAM," in *Proceedings of 4th International Conference on Learning Representations, Workshop Track*, 2016.

[9] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations*, 2019.

[10] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, "On the variance of the adaptive learning rate and beyond," in *International Conference on Learning Representations*, 2020.

[11] C. C. Holt, "Forecasting seasonals and trends by exponentially weighted moving averages," *International Journal of Forecasting*, vol. 20, no. 1, pp. 5–10, 2004.

[12] E. S. Gardner Jr and E. McKenzie, "Forecasting trends in time series," *Management Science*, vol. 31, no. 10, pp. 1237–1246, 1985.

[13] R. J. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice*. OTexts, 2018.

[14] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms," *preprint arXiv:1708.07747*, 2017.

[15] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[16] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[17] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proceedings of International Conference on Learning Representations*, 2014.

[18] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *Proceedings of the 31st International Conference on Machine Learning*, 2014.

[19] Y. Xu, Q. Lin, and T. Yang, "Stochastic convex optimization: Faster local growth implies faster global convergence," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3821–3830, JMLR. org, 2017.

[20] Z. Chen, Y. Xu, E. Chen, and T. Yang, "Sadagrad: Strongly adaptive stochastic gradient methods," in *International Conference on Machine Learning*, pp. 912–920, 2018.

[21] Y. K. Levy, A. Yurtsever, and V. Cevher, "Online adaptive methods, universality and acceleration," in *Advances in Neural Information Processing Systems*, pp. 6500–6509, 2018.

[22] J. Chen and Q. Gu, "Closing the generalization gap of adaptive gradient methods in training deep neural networks," *preprint arXiv:1806.06763*, 2018.

[23] D. Zhou, Y. Tang, Z. Yang, Y. Cao, and Q. Gu, "On the convergence of adaptive gradient methods for nonconvex optimization," *preprint arXiv:1808.05671*, 2018.

[24] F. Zou and L. Shen, "On the convergence of weighted AdaGrad with momentum for training deep neural networks," *preprint arXiv:1808.03408*, 2018.

[25] X. Chen, S. Liu, R. Sun, and M. Hong, "On the convergence of a class of ADAM-type algorithms for non-convex optimization," in *Proceedings of International Conference on Learning Representation (ICLR)*, 2019.

[26] F. Zou, L. Shen, Z. Jie, W. Zhang, and W. Liu, "A sufficient condition for convergences of ADAM and RMSProp," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 11127–11135, 2019.