Unbiased Quasi-hyperbolic Nesterov-gradient Momentum-based Optimizers for Accelerating Convergence

Weiwei Cheng[1], Xiaochun Yang[1*], Bin Wang[1] and Wei Wang[2,3]

[1]School of Computer Science and Engineering, Northeastern University, Shenyang, 110167, Liaoning, China.
[2]Information Hub, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, Guangdong, China.
[3]The Hong Kong University of Science and Technology, Hong Kong, China.

*Corresponding author(s). E-mail(s): yangxc@mail.neu.edu.cn; Contributing authors: 1810619@stu.neu.edu.cn; binwang@mail.neu.edu.cn; weiwcs@ust.hk;

Abstract

In the training process of deep learning models, one of the important steps is to choose an appropriate optimizer that directly determines the final performance of the model. Choosing the appropriate direction and step size (i.e. learning rate) of parameter update are decisive factors for optimizers. Previous gradient descent optimizers could be oscillated and fail to converge to a minimum point because they are only sensitive to the current gradient. Momentum-Based Optimizers (MBOs) have been widely adopted recently since they can relieve oscillation to accelerate convergence by using the exponentially decaying average of gradients to fine-tune the direction. However, we find that most of the existing MBOs are biased and inconsistent with the local fastest descent direction resulting in a high number of iterations. To accelerate convergence, we propose an Unbiased strategy to adjust the descent direction of a variety of MBOs. We further propose an Unbiased Quasi-hyperbolic Nesterov-gradient strategy (UQN) by combining our Unbiased strategy with the existing Quasi-hyperbolic and Nesterov-gradient. It makes each update step move in the local fastest descent direction, predicts the future gradient to avoid crossing the minimum point, and reduces gradient variance. We extend our strategies to multiple MBOs and prove the convergence of our strategies. The main experimental results presented in this paper are based on popular neural network models and benchmark datasets. The experimental results demonstrate the effectiveness and universality of our proposed strategies.

Keywords: Optimizer, Momentum, Accelerate Convergence, Unbiased

1 Introduction

Model training is one of the important steps in machine learning and deep learning tasks. The choice of optimizers is one of the important factors in model training, which determines the model performance [1–4]. Optimizers have been widely used in both linear models (e.g., logistic regression model) and nonlinear models (e.g., neural network model) [5–13]. The optimizer is to find the optimal parameters for a training model that determines the model's performance including the accuracy and the number of training iterations of the model [14–18].

The Gradient Descent optimizer (GD) [19] is a way to minimize the loss function $L(\theta_t)$ by updating parameters following the direction of gradient $\nabla L(\theta_t)$ of the loss function. $L(\theta_t)$ is a loss function parameterized by model parameter θ_t. Intuitively, the process of minimizing $L(\theta_t)$ by the optimizer is equivalent to a ball moving from the mountain to the valley in the direction of slope of the surface constructed by the $L(\theta_t)$, that is, its velocity direction (i.e. descent direction of the ball) is consistent with the direction of slope (i.e. the local fastest descent direction). When the ball encounters a valley (i.e. the local minimum point), its velocity may be very slow because the slope at the bottom of the valley approaches 0, and the number of iterations may be very high.

To escape from the valley, Momentum [20–22] improves GD by taking into account the momentum the ball accumulates as it goes downhill. Momentum reduces the number of iterations (i.e. accelerate convergence) by allowing the ball to accelerate [20–22]. However, by analyzing Momentum's formula for calculating the ball momentum, we find that the accumulated momentum of the ball causes the ball to deviate from the local fastest descent direction (i.e. the direction of gradient), which results in the momentum accumulated by the ball at each step does not reach the maximum.

Therefore, in this study, we propose an *Unbiased strategy* for Momentum, called U-Momentum, to adjust the ball velocity direction consistent with the direction of slope to further reduce the number of iterations.

There are many optimizers based on Momentum, and for ease of representation, we use Momentum-Based Optimizers (MBOs) to represent the family of Momentum-based optimizers [23–25]. Quasi-Hyperbolic Momentum (QHM) and Nesterov Accelerated Gradient (NAG) are two typical MBOs. QHM [26] shows that increasing the current gradient weight can effectively reduce gradient variance and accelerate convergence based on an informal and speculative motivation for variance reduction. NAG [27] illustrates that the convergence can be accelerated by adding the nesterov-gradient (the future gradient). Intuitively, NAG makes the ball smart enough to slow down before the surface rises. The future gradient can predict the next position of the ball, which makes the ball smarter and avoids crossing the minimum point [28–31].

We further apply the Unbiased strategy to both QHM and NAG to get U-QHM and U-NAG, respectively. We combine U-QHM and U-NAG to UQN, to further reduce the number of iterations. In other words, the *UQN strategy* unifies the velocity direction and the direction of slope, reduces gradient variance and prevents the ball crossing the minimum point. The main contributions of this paper are as follows:

- For MBOs, we analyze the velocity direction of the ball and find that the direction is inconsistent with the local fastest descent direction, so we propose an Unbiased strategy to make them consistent and accelerate convergence.
- For further accelerating convergence, we propose the UQN strategy by combining U-QHM and U-NAG. The UQN strategy integrates the advantages of the Unbiased strategy, QHM, and NAG, as well as can be extended to the widely used MBOs.
- We conduct a lot of comparative experiments to demonstrate the effectiveness of the Unbiased strategy and the UQN strategy. The MBOs improved by the above two strategies accelerate convergence.

The rest of this paper is organized as follows. Section 2 lists the updated rules of the widely used MBOs and describes the general form of update rule of MBOs. Section 3 mainly introduces the Unbiased strategy to adjust the velocity direction and gives three improved examples of MBOs. Section 4 introduces the UQN strategy and applies it to a number of widely used MBOs. Section 5 proves the convergence of the Unbiased strategy and the UQN strategy. Section 6 conducts the hyperparameter scanning experiments and the universality verification experiment of the Unbiased strategy and the UQN strategy. Section 7 summarizes this paper and explores possible future research directions.

2 Preliminaries

In this section, we elaborate on the definition of the symbols mentioned in this paper, as shown in Table 1. We list the widely used MBOs and summarize the general form of the update rule of MBOs.

The widely used MBOs include Momentum [20], QHM [26], NAG [27], Adam [32], QHAdam [26], Nadam [33], and AdaMax [32]. The relationship among them is shown in Fig.1. Momentum is a foundmental method of MBOs. QHM and NAG improve Momentum by increasing the weight of the current gradient and the future gradient, respectively, resulting in reducing gradient variance and avoiding crossing the minimum point. The above three MBOs adopt fixed learning rate that are unfriendly to unfamiliar models and tasks. Adam improves Momentum by using adaptive learning rate. To further accelerate convergence, Nadam combines NAG with Adam, QHAdam combines QHM with Adam, and AdaMax improves Adam by considering infinite norm.

- Momentum [20]. Momentum increases for dimensions whose gradients point in the same directions and reduces updates for dimensions whose gradients change directions. It alleviates the disadvantage that the velocity of the ball may be very

Table 1: Symbol definition.

Symbols	Definitions
function	
$L(\theta_t)$	the loss function
$f'(t), f''(t)$	the function about t
vector	
$\theta_t \in \mathbb{R}^n$	parameter at step t
$\nabla L(\theta_t) \in \mathbb{R}^n$	gradient of $L(\theta_t)$
$g_t \in \mathbb{R}^n$	the exponentially decaying average of gradients
$M'_t \in \mathbb{R}^n$	update of g_t
$M''_t \in \mathbb{R}^n$	update of g'_t
$N'_t \in \mathbb{R}^n$	update of v_t
$N''_t \in \mathbb{R}^n$	update of v'_t
value	
$t \in \mathbb{N}$	the number of iterations
$\eta_t \in \mathbb{Q}+$	learning rate
$v_t \in \mathbb{Q}+$	the exponentially decaying average of the square of the gradients

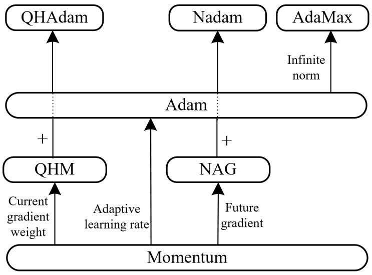


Fig. 1: The relationship among the widely used MBOs.

slow when $\nabla L(\theta_t)$ approaches 0. Fig. 2(a) describes the update of parameter of Momentum. Momentum adjusts θ_t to θ_{t+1} using the learning rate η and the exponentially decaying average of gradients g_t, where g_t is a vector and depends on the previous vector g_{t-1} and the current gradient $\nabla L(\theta_t)$. The update rule of Momentum is as follows:

$$\begin{cases} \theta_{t+1} = \theta_t - \eta \cdot g_t, \\ \quad g_t = \gamma \cdot g_{t-1} + \nabla L(\theta_t), \end{cases} \tag{1}$$

where $g_0 = \nabla L(\theta_0)$ and $\gamma \in [0, 1]$. $\gamma \to 1$ indicates that g_{t-1} has a prominent influence on g_t.

- QHM [26]. QHM is proposed based on the motivation of reducing gradient variance and has been proved to be an effective strategy for reducing the number of iterations. In essence, QHM takes the weighted sum of $\nabla L(\theta_t)$ and g'_t, which is equivalent to increasing the weight of $\nabla L(\theta_t)$. The update rule of QHM is as follows:

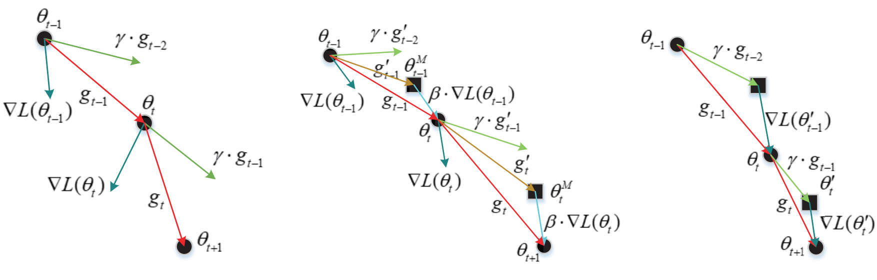
$$\begin{cases} \theta_{t+1} = \theta_t - \eta \cdot g_t, \\ \quad g_t = g'_t + \beta \cdot \nabla L(\theta_t), \\ \quad g'_t = \gamma \cdot g'_{t-1} + \nabla L(\theta_t). \end{cases} \tag{2}$$

Fig. 2(b) shows the update of parameter of QHM, where QHM first updates θ_t to a middle parameter θ_t^M using $\nabla L(\theta_t)$ and g'_{t-1}, and then updates θ_t^M to θ_{t+1} again using $\nabla L(\theta_t)$.

- NAG [27]. To solve the problem that the ball often crosses the minimum point for Momentum, NAG is proposed to enhance the prediction ability of the ball by adding the nesterov-gradient (the future gradient). NAG makes the ball smarter so that the ball can slow down when it is predicted to cross the minimum point and converge to the minimum point. The update rule of NAG is as follows:

$$\begin{cases} \theta_{t+1} = \theta_t - \eta \cdot g_t, \\ \quad g_t = \gamma \cdot g_{t-1} + \nabla L(\theta'_t). \end{cases} \tag{3}$$

The approximate future position parameter θ'_t is estimated as $\theta_t - \gamma \cdot \eta \cdot g_{t-1}$ according to the update rule of Momentum. The future gradient $\nabla L(\theta'_t)$ is the gradient of the approximate future position $L(\theta'_t) = L(\theta_t - \gamma \cdot \eta \cdot g_{t-1})$. Fig. 2(c) shows the update of parameter of NAG. NAG first predicts the approximate future position parameter θ'_t by using the previous vector g_{t-1}, the momentum term γ, and the learning rate η, and then updates θ'_t to θ_{t+1} with the future gradient $\nabla L(\theta'_t)$ and the previous vector g_{t-1}.



(a) Momentum (b) QHM (c) NAG

Fig. 2: Update of Parameter.

- Adam [32]. Adaptive Moment Estimation (Adam) is an adaptive learning rate MBO and proposed based on Momentum. Adam keeps an exponentially decaying average of gradients g_t, similar to Momentum. Momentum is unfriendly to unfamiliar tasks and models and requires a lot of time to adjust the hyperparameters since the learning rate of Momentum is a fixed constant. Adam is proposed to adjust the learning rate η_t by using the exponentially decaying average of the square of the gradients v_t. It performs smaller updates (i.e. small learning rates) for parameters associated with frequent features, and larger updates (i.e. big learning rates) for parameters associated with infrequent features. The update rule of Adam is as follows:

$$\begin{cases} \theta_{t+1} = \theta_t - \eta_t \cdot g_t, \\ \quad g_t = \gamma \cdot g_{t-1} + \nabla L(\theta_t), \\ \quad \eta_t = \dfrac{\eta}{\sqrt{v_t} + \epsilon}, \\ \quad v_t = \lambda \cdot v_{t-1} + \nabla L^2(\theta_t), \end{cases} \tag{4}$$

where $\lambda = 0.999$.

- QHAdam [26]. QHAdam combines QHM and Adam. To reduce gradient variance and accelerate convergence, QHAdam is proposed by increasing the weight of $\nabla L(\theta_t)$ to adjust g_t and increasing the weight of $\nabla L^2(\theta_t)$ to adjust η_t. The update rule of QHAdam is as follows:

$$\begin{cases} \theta_{t+1} = \theta_t - \eta_t \cdot g_t, \\ \quad g_t = g'_t + \beta \cdot \nabla L(\theta_t), \\ \quad g'_t = \gamma \cdot g'_{t-1} + \nabla L(\theta_t), \\ \quad \eta_t = \dfrac{\eta}{\sqrt{v_t} + \epsilon}, \\ \quad v_t = v'_t + \beta^2 \cdot \nabla L^2(\theta_t), \\ \quad v'_t = \lambda \cdot v'_{t-1} + \nabla L^2(\theta_t). \end{cases} \tag{5}$$

- Nadam [33]. Nadam combines NAG and Adam. Compared with NAG, Nadam adds the adaptive learning rate η_t to deal with unfamiliar models and tasks. Nadam adds the future gradient $\nabla L(\theta'_t)$ to Adam, improving the ball's predictive ability to avoid crossing the minimum point. The update rule of Nadam is as follows:

$$\begin{cases} \theta_{t+1} = \theta_t - \eta_t \cdot g_t, \\ \quad g_t = g'_t + \gamma \cdot \nabla L(\theta_t), \\ \quad g'_t = \gamma \cdot g'_{t-1} + \nabla L(\theta'_t), \\ \quad \eta_t = \dfrac{\eta}{\sqrt{v_t} + \epsilon}, \\ \quad v_t = \lambda \cdot v_{t-1} + \nabla L^2(\theta_t), \end{cases} \tag{6}$$

where $\theta'_t = \theta_t - \gamma \cdot \eta \cdot g'_{t-1}$.

- AdaMax [32]. Given Adam's unstable performance in high-dimensional data, Kingma etc. [32] propose that it is caused by the instability of the two norm of Adam in high-dimensional data. Therefore, they use the infinite norm with stable performance to adjust η_t. The update rule of AdaMax is as follows:

$$\begin{cases} \theta_{t+1} = \theta_t - \eta_t \cdot g_t, \\ \quad g_t = \gamma \cdot g_{t-1} + \nabla L(\theta_t), \\ \quad \eta_t = \dfrac{\eta}{\sqrt{v_t} + \epsilon}, \\ \quad v_t = \lambda \cdot v_{t-1} + \nabla L(\theta_t)^{\infty}, \end{cases} \tag{7}$$

where $\nabla L(\theta_t)^{\infty}$ is the infinite norm of $\nabla L(\theta_t)$.

We summarize the general form of update rule of MBOs by analyzing Eq. (1-7). The update rule of MBOs is $\theta_{t+1} = \theta_t - \eta_t \cdot g_t$, where g_t and η_t are respectively the exponentially decaying average of gradients and the learning rate at step t. The direction of g_t is the descent direction of the ball. We mainly introduce two parts: g_t adjustment rule and η_t adjustment rule. Firstly, we introduce g_t adjustment rule of MBOs.

$$\begin{cases} g_t = g'_t + M'_t, \\ g'_t = \gamma \cdot g'_{t-1} + M''_t, \end{cases} \tag{8}$$

where $g_0 = \nabla L(\theta_0)$ and θ_0 is the random initial parameter. Fig. 3 shows the update of parameter of MBOs [22], in which the name of each vector is marked. Vectors g'_{t-1}, g'_t, M'_t, and M''_t are used to adjust g_t.

Expanding Eq. (8), we get $g'_t = \gamma^t \cdot \nabla L(\theta_0) + \sum_{i=0}^{t-1} \gamma^i \cdot M''_{t-i}$. We know that M''_{t-i} is accumulated into g'_t. Let $M'_t = \omega' \cdot \nabla L(f'(t))$ and $M''_t = \omega'' \cdot \nabla L(f''(t))$, where $\omega', \omega'' \in \mathbb{Q}+$ and $\nabla L(f'(t))$, $\nabla L(f''(t))$ are gradients of $f'(t)$ and $f''(t)$, respectively. $f'(t)$ and $f''(t)$ are functions of t. Different optimizers have different values of ω', ω'', $f'(t)$, and $f''(t)$. For example, ω', ω'', $f'(t)$, and $f''(t)$ of QHM are β, 1, θ_t, and θ_t, respectively. ω', ω'', $f'(t)$, and $f''(t)$ of NAG are 0, 1, 0, and $\theta_t - \gamma \cdot \eta \cdot g'_{t-1}$, respectively.

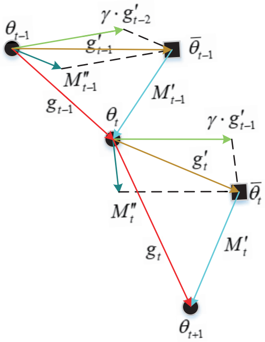


Fig. 3: The update of parameter of MBOs.

Then, we introduce η_t adjustment rule of MBOs, which has two forms: fixed constant and adaptive adjustment variable [34–36]. η_t adjustment rule is summarized in the following form:

$$\begin{cases} \eta_t = \dfrac{\eta}{\sqrt{v_t} + \epsilon}, \\ v_t = v'_t + (N'_t)^2, \\ v'_t = \lambda \cdot v'_{t-1} + (N''_t)^2, \end{cases} \tag{9}$$

where η_t is the learning rate at step t and η is a given constant. ϵ is a really small number to keep the denominator from being 0. v_t is the exponentially decaying average of the square of the gradients. $\lambda \in [0, 1]$ and $\lambda \to 1$ indicate that v'_{t-1} has a great influence on η_t. According to Eq. (9), the square of N''_t and v'_{t-1} are accumulated into v'_t and the square of N'_t and v'_t are accumulated into v_t. Similar to the form of M'_t and M''_t, different optimizers have different values of N'_t and N''_t. To show the differences among optimizers more clearly, we transform Eq. (1-7) into the general forms Eq. (8) and Eq. (9), the variable values of which are shown in Table 2.

Table 2: The variable values of MBOs.

Optimizers	$M'_t = \omega' \cdot \nabla L(f'(t))$		$M''_t = \omega'' \cdot \nabla L(f''(t))$		N'_t	N''_t	η_t
	ω'	$\nabla L(f'(t))$	ω''	$\nabla L(f''(t))$			
Momentum	0	/	1	$\nabla L(\theta_t)$	/	/	η
QHM	β	$\nabla L(\theta_t)$	1	$\nabla L(\theta_t)$	/	/	η
NAG	0	/	1	$\nabla L(\theta'_t)$	/	/	η
Adam	0	/	1	$\nabla L(\theta_t)$	0	$\nabla L(\theta_t)$	/
QHAdam	β	$\nabla L(\theta_t)$	1	$\nabla L(\theta_t)$	$\beta \cdot \nabla L(\theta_t)$	$\nabla L(\theta_t)$	/
Nadam	γ	$\nabla L(\theta_t)$	1	$\nabla L(\theta'_t)$	0	$\nabla L(\theta_t)$	/
AdaMax	0	/	1	$\nabla L(\theta_t)$	0	$\sqrt{\nabla L(\theta_t)^\infty}$	/

3 Unbiased strategy based on MBOs

In this section, we first show that the widely used MBOs are biased, i.e. the descent direction of the ball (i.e. the direction of g_t) is inconsistent with the local fastest descent direction (i.e. the direction of $\nabla L(\theta_t)$), which require much more the number of iterations to find optimal parameters. Ideally the directions of g_t and $\nabla L(\theta_t)$ are desired to be consistent, so that the solver of optimizers can descend to the minimum point fast and then converge [19, 23]. We then propose an *Unbiased strategy* to make the expectation of g_t and $\nabla L(\theta_t)$ equal. Finally, we apply the Unbiased strategy to Momentum, QHM, and NAG.

3.1 Consistency between g_t and $\nabla L(\theta_t)$

We first propose a necessary and sufficient condition to check if the direction of g_t is consistent with the direction of $\nabla L(\theta_t)$. And then we find that the widely used

MBOs shown in Section 2 are biased, that is, they do not satisfy this condition, which results in a high number of iterations.

Theorem 1 *The direction of g_t of MBOs is consistent with the direction of gradient $\nabla L(\theta_t)$ iff $\omega'' \cdot \sum_{i=0}^{t} \gamma^i + \omega' = 1$.*

Proof The general form of g_t is Eq. (8). We expand g'_t and g_t to get the following:

$$g'_0 = M''_0, g_0 = M''_0 + M'_0;$$
$$g'_1 = \gamma \cdot M''_0 + M''_1, g_1 = \gamma \cdot M''_0 + M''_1 + M'_1;$$
$$\cdots$$
$$g'_t = \sum_{i=0}^{t} \gamma^i \cdot M''_{t-i}, g_t = \sum_{i=0}^{t} \gamma^i \cdot M''_{t-i} + M'_t = \omega'' \cdot \sum_{i=0}^{t} \gamma^i \cdot \nabla L(f''(t)) + \omega' \cdot \nabla L(f'(t)).$$

When $\omega'' \cdot \sum_{i=0}^{t} \gamma^i + \omega' = 1$, the expectation $E(g_t) = \nabla L(\theta_t)$ and g_t is an unbiased estimate of $\nabla L(\theta_t)$, which means the direction of g_t of MBOs is consistent with the direction of $\nabla L(\theta_t)$. □

Lemma 1 The directions of g_t of the widely used MBOs shown in Section 2 are inconsistent with the direction of $\nabla L(\theta_t)$.

Proof According to Table 2, the parameters $\omega' = 0$ and $\omega'' = 1$ in update rule of Momentum. From Theorem 1, $\omega'' \cdot \sum_{i=0}^{t} \gamma^i + \omega' = \lim_{t\to\infty} \sum_{i=0}^{t} \gamma^i \neq 1$, so the expectation $E(g_t) \neq \nabla L(\theta_t)$ and the direction of g_t of Momentum is inconsistent with the direction of $\nabla L(\theta_t)$.

Similarly, according to Table 2, the parameters $\omega' = \beta$ and $\omega'' = 1$ in update rule of QHM, and the parameters $\omega' = 0$ and $\omega'' = 1$ in update rule of NAG. We can know that $\lim_{t\to\infty} \sum_{i=0}^{t} \gamma^i + \beta \neq 1$ and $\lim_{t\to\infty} \sum_{i=0}^{t} \gamma^i \neq 1$, so the directions of g_t of QHM and NAG are also inconsistent with the directions of $\nabla L(\theta_t)$. Adam is proposed based on Momentum and just changes the learning rate, the direction of g_t for Adam is also inconsistent with the direction of $\nabla L(\theta_t)$. Similarly, QHAdam, Nadam, and AdaMax are based on Adam, so they also inherit the same disadvantages. □

3.2 Unbiased strategy

In order to make the directions of g_t and $\nabla L(\theta_t)$ consistent, we need to adjust ω'' in the update rule so that $\omega'' \cdot \sum_{i=0}^{t} \gamma^i + \omega' = 1$, i.e. $E(g_t) = \nabla L(\theta_t)$. We adjust ω'' to Ω'', and then we make $\Omega'' \cdot \sum_{i=0}^{t} \gamma^i + \omega' = 1$, i.e. $\Omega'' \cdot \sum_{i=0}^{t} \gamma^i = 1 - \omega'$. We multiply both sides of this equation by $1-\gamma$, i.e., $\Omega'' \cdot (1-\gamma) \cdot \sum_{i=0}^{t} \gamma^i = (1-\gamma) \cdot (1-\omega')$. Since $(1-\gamma) \cdot \sum_{i=0}^{t} \gamma^i = 1$ when t is large enough, so $\Omega'' = (1-\gamma) \cdot (1-\omega')$. Therefore, based on Eq. (8), we adjust $M''_t = \omega'' \cdot \nabla L(f''(t))$ to $\Omega'' \cdot \nabla L(f''(t)) =$

$\frac{1-\omega'}{\omega''} \cdot (1-\gamma) \cdot \omega'' \cdot \nabla L(f''(t)) = \frac{1-\omega'}{\omega''} \cdot (1-\gamma) \cdot M_t''$, so g_t adjustment rule with Unbiased strategy is shown in Eq. (10).

$$\begin{cases} g_t = g_t' + M_t', \\ g_t' = \gamma \cdot g_{t-1}' + \dfrac{1-\omega'}{\omega''} \cdot (1-\gamma) \cdot M_t'', \end{cases} \tag{10}$$

where $0 < \gamma < 1$.

Next, we apply the Unbiased strategy to those MBOs with non-adaptive learning rate, i.e., Momentum, QHM and NAG, to amend the direction of g_t. We do not apply Unbiased strategy to Adam, QHAdam, Nadam and AdaMax that adopt adaptive learning rate for the following reasons. The adaptive learning rate MBOs adjust η_t without changing g_t. At the same time, we avoid the impact of the adaptive learning rate on Unbiased strategy and better study the improvement effect of Unbiased strategy.

3.2.1 Unbiased Momentum (U-Momentum)

We first show how to add the Unbiased strategy on Momentum to get Unbiased Momentum (U-Momentum). We import $M_t' = 0$, $M_t'' = \nabla L(\theta_t)$, $\omega' = 0$, and $\omega'' = 1$ shown in Table 2 into Eq. (10) to adjust g_t, the g_t adjustment rule of U-Momentum is shown in Eq. (11).

$$g_t = g_t' = \gamma \cdot g_{t-1}' + (1-\gamma) \cdot \nabla L(\theta_t). \tag{11}$$

Based on Eq. (8), Eq. (11), $M_t' = \omega' \cdot \nabla L(f'(t))$, and $M_t'' = \omega'' \cdot \nabla L(f''(t))$, we know $\omega' = 0$ and $\omega'' = (1-\gamma)$ of U-Momentum, so $\omega'' \cdot \sum_{i=0}^{t} \gamma^i + \omega' = (1-\gamma) \cdot \sum_{i=0}^{t} \gamma^i + 0 = 1$, i.e., the direction of g_t is consistent with the direction of $\nabla L(\theta_t)$.

3.2.2 Unbiased QHM (U-QHM)

U-QHM is obtained by applying the Unbiased strategy to QHM. Similar to U-Momentum, we import $M_t' = \beta \cdot \nabla L(\theta_t)$, $M_t'' = \nabla L(\theta_t)$, $\omega' = \beta$, and $\omega'' = 1$ of QHM shown in Table 2 into Eq. (10), and get the g_t adjustment rule of U-QHM as shown in Eq. (12).

$$\begin{cases} g_t = g_t' + \beta \cdot \nabla L(\theta_t), \\ g_t' = \gamma \cdot g_{t-1}' + (1-\beta) \cdot (1-\gamma) \cdot \nabla L(\theta_t). \end{cases} \tag{12}$$

Based on Eq. (8), Eq. (12), $M_t' = \omega' \cdot \nabla L(f'(t))$, and $M_t'' = \omega'' \cdot \nabla L(f''(t))$, we know $\omega' = \beta$ and $\omega'' = (1-\beta) \cdot (1-\gamma)$ of U-QHM, so $\omega'' \cdot \sum_{i=0}^{t} \gamma^i + \omega' = (1-\beta) \cdot (1-\gamma) \cdot \sum_{i=0}^{t} \gamma^i + \beta = 1$, i.e., the direction of g_t is consistent with the direction of $\nabla L(\theta_t)$.

3.2.3 Unbiased NAG (U-NAG)

We show how to add the Unbiased strategy on NAG to get Unbiased NAG (U-NAG). Similar to the above two, we import $M_t' = 0$, $M_t'' = \nabla L(\theta_t')$, $\omega' = 0$, and $\omega'' = 1$ of NAG shown in Table 2 into Eq. (10), and get the g_t adjustment rule of U-NAG as shown in Eq. (13).

$$g_t = g_t' = \gamma \cdot g_{t-1}' + (1-\gamma) \cdot \nabla L(\theta_t'), \tag{13}$$

where θ_t' is the approximate future position parameter. Based on Eq. (8), Eq. (13), $M_t' = \omega' \cdot \nabla L(f'(t))$, and $M_t'' = \omega'' \cdot \nabla L(f''(t))$, we know $\omega' = 0$ and $\omega'' = 1-\gamma$, so $\omega'' \cdot \sum_{i=0}^{t} \gamma^i + \omega' = (1-\gamma) \cdot \sum_{i=0}^{t} \gamma^i + 0 = 1$, i.e., the direction of g_t is consistent with the direction of $\nabla L(\theta_t)$.

4 UQN: combining U-QHM and U-NAG

To further accelerate convergence, we propose UQN by combining U-QHM and U-NAG. UQN integrates the advantages of Unbiased strategy, QHM, and NAG, therefore, it could accelerate convergence by unifying the directions of g_t and $\nabla L(\theta_t)$, reducing gradient variance, and avoiding crossing the minimum point. Next, we first apply UQN to non-adaptive learning rate MBOs (i.e. Momentum, QHM, and NAG) and prove that UQN has the effect of reducing gradient variance through Theorem 2. Then, we apply UQN to the widely used adaptive learning rate MBOs shown in Section 2.

4.1 Analysis of gradient variance of MBOs with non-adaptive learning rate using UQN

In this subsection, we apply UQN to Momentum (the basic MBO). Based on Eq. (12) and Eq. (13), the update rule of UQN-Momentum is shown in Eq. (14).

$$\begin{cases} \theta_{t+1} = \theta_t - \eta_t \cdot g_t, \\ \quad g_t = g_t' + \beta \cdot \nabla L(\theta_t - \gamma \cdot \eta \cdot g_{t-1}'), \\ \quad g_t' = \gamma \cdot g_{t-1}' + (1-\beta) \cdot (1-\gamma) \cdot \nabla L(\theta_t), \\ \quad \eta_t = \eta. \end{cases} \tag{14}$$

As can be seen from Fig. 4, UQN-Momentum, UQN-QHM, and UQN-NAG are all obtained by adding Unbiased strategy, current gradient weight, and future gradient on Momentum. Therefore, the update rules of UQN-QHM and UQN-NAG are the same as that of UQN-Momentum and shown in Eq. (14). Then, we obtain the gradient variance of the UQN-Momentum in Theorem 2 and find that the UQN strategy can reduce gradient variance by analyzing Fig. 5.

Theorem 2 $\lim_{t\to\infty} Variance(UQN-Momentum) = \alpha \cdot \Sigma$, *where* Σ *is the gradient variance of Momentum and* $\alpha = \frac{2}{1+\gamma} \cdot \beta^2 - 2\frac{1-\gamma}{1+\gamma} \cdot \beta + \frac{1-\gamma}{1+\gamma}$.

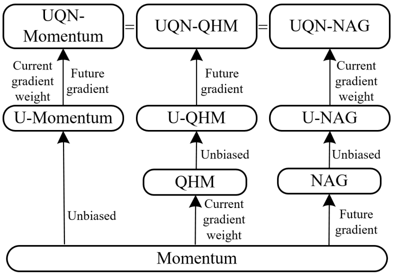


Fig. 4: UQN-Momentum, UQN-QHM, and UQN-NAG.

Proof We expand g_t in Eq. (14) and get the following:

$$\begin{aligned} g_t =&(1-\beta)\cdot(1-\gamma)\cdot\gamma^t\cdot\nabla L(\theta_0)+\cdots \\ &+(1-\beta)\cdot(1-\gamma)\cdot\gamma^0\cdot\nabla L(\theta_t)+\beta\cdot\nabla L(\theta_{t+1}). \end{aligned}$$

We assume that $\nabla L(\theta_{t+1-i})$ is an independent identically distributed random vector [26]. The coefficient δ_i of $\nabla L(\theta_{t+1-i})$ is:

$$\delta_i = \begin{cases} \beta & i=0 \\ (1-\beta)\cdot(1-\gamma)\cdot\gamma^{i-1} & i=1,\ldots,t+1 \end{cases}$$

So the gradient variance of UQN-Momentum is:

$$\lim_{t\to\infty} Variance(UQN{-}Momentum) = \lim_{t\to\infty}\sum_{i=0}^{t+1}\delta_i^2\cdot\Sigma,$$

where

$$\begin{aligned} \lim_{t\to\infty}\sum_{i=0}^{t+1}\delta_i^2 &= \beta^2+(1-\beta)^2\cdot\frac{1-\gamma}{1+\gamma} \\ &= \frac{2}{1+\gamma}\cdot\beta^2-2\frac{1-\gamma}{1+\gamma}\cdot\beta+\frac{1-\gamma}{1+\gamma}. \end{aligned}$$

Therefore, $\alpha=\lim_{t\to\infty}\sum_{i=0}^{t+1}\delta_i^2$ and the proof of Theorem 2 is completed.

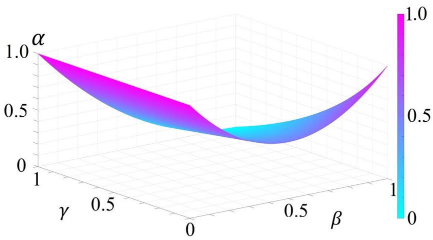


Fig. 5: Variance of UQN-Momentum.

□

From Theorem 2, the gradient variance of UQN-Momentum increases with α. The relationship of α to γ and β can be clearly illustrated in Fig. 5, from which it can be seen that $\alpha < 1$ when $\beta \in (0, 1)$, so the gradient variance of UQN-Momentum is less than that of Momentum. Therefore, the UQN strategy applied to Momentum can reduce gradient variance. Based on the update rules Eq. (1-7), similar to Theorem 2, it can be proved that UQN also has the advantage of reducing gradient variance on other MBOs.

4.2 Improving MBOs with Adaptive Learning Rate using UQN

According to Theorem 2, we know that the UQN integrates both U-QHM and U-NAG and effectively reduce gradient variance of MBOs. Now we combine UQN with adaptive learning rate MBOs (i.e. Adam, QHAdam, Nadam, and AdaMax). The general form of the update rule is listed as follows:

$$\begin{cases} \theta_{t+1} = \theta_t - \eta_t \cdot g_t, \\ \quad g_t = g'_t + \beta \cdot \nabla L(\theta'_t), \\ \quad g'_t = \gamma \cdot g'_{t-1} + \dfrac{1-\beta}{\omega''} \cdot (1-\gamma) \cdot M''_t, \\ \quad \eta_t = \dfrac{\eta}{\sqrt{v_t} + \epsilon}, \\ \quad v_t = v'_t + \beta^2 \cdot \nabla L^2(\theta_t), \\ \quad v'_t = \lambda \cdot v'_{t-1} + (N''_t)^2, \end{cases} \tag{15}$$

where $\theta'_t = \theta_t - \gamma \cdot \eta \cdot g'_{t-1}$ and $\lambda = 0.999$. Below we illustrate how to improve multiple MBOs and list corresponding update rules.

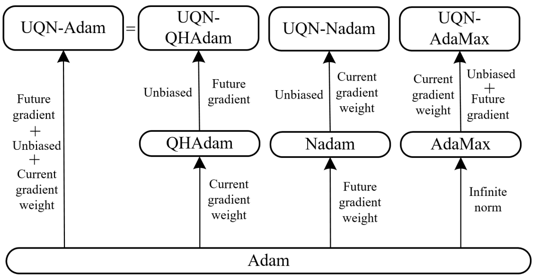


Fig. 6: Adaptive learning rate MBOs with UQN.

As can be seen from Fig. 6, UQN-Adam and UQN-QHAdam are obtained by adding Unbiased strategy, current gradient weight, and future gradient on Adam. Therefore, the update rules of UQN-Adam and UQN-QHAdam are the same.

4.2.1 UQN-Adam & UQN-QHAdam

We first show how to add the UQN strategy on Adam and QHAdam to get UQN-Adam and UQN-QHAdam, respectively. Based on Table 2, we know Adam and QHAdam have the same M''_t, ω'', and N''_t. We import $M''_t = \nabla L(\theta_t)$, $\omega'' = 1$, and $N''_t = \nabla L(\theta_t)$ into Eq. (15) to adjust g_t and η_t of UQN-Adam and UQN-QHAdam. The update rules of UQN-Adam and UQN-QHAdam are the same, as shown in Eq. (16).

$$\begin{cases} \theta_{t+1} = \theta_t - \eta_t \cdot g_t, \\ \quad g_t = g'_t + \beta \cdot \nabla L(\theta'_t), \\ \quad g'_t = \gamma \cdot g'_{t-1} + (1-\beta) \cdot (1-\gamma) \cdot \nabla L(\theta_t), \\ \quad \eta_t = \dfrac{\eta}{\sqrt{v_t} + \epsilon}, \\ \quad v_t = v'_t + \beta^2 \cdot \nabla L^2(\theta_t), \\ \quad v'_t = \lambda \cdot v'_{t-1} + \nabla L^2(\theta_t). \end{cases} \tag{16}$$

4.2.2 UQN-Nadam

Similar to UQN-Adam, we import $M''_t = (\sqrt{\gamma}+1) \cdot \nabla L(\theta'_t)$, $\omega'' = (\sqrt{\gamma}+1)$, and $N''_t = \nabla L(\theta_t)$ of Nadam shown in Table 2 into Eq. (15) to get g_t and η_t. The update rule of UQN-Nadam is as follows:

$$\begin{cases} \theta_{t+1} = \theta_t - \eta_t \cdot g_t, \\ \quad g_t = g'_t + \beta \cdot \nabla L(\theta'_t), \\ \quad g'_t = \gamma \cdot g'_{t-1} + (1-\beta) \cdot (1-\gamma) \cdot \nabla L(\theta'_t), \\ \quad \eta_t = \dfrac{\eta}{\sqrt{v_t} + \epsilon}, \\ \quad v_t = v'_t + \beta^2 \cdot \nabla L^2(\theta_t), \\ \quad v'_t = \lambda \cdot v'_{t-1} + \nabla L^2(\theta_t). \end{cases} \tag{17}$$

4.2.3 UQN-AdaMax

Similar to the above two, we import $M''_t = \nabla L(\theta_t)$, $\omega'' = 1$, and $N''_t = \sqrt{\nabla L(\theta_t)^\infty}$ of AdaMax shown in Table 2 to get g_t and η_t. The update rule of UQN-AdaMax is as follows:

$$\begin{cases} \theta_{t+1} = \theta_t - \eta_t \cdot g_t, \\ g_t = g'_t + \beta \cdot \nabla L(\theta'_t), \\ g'_t = \gamma \cdot g'_{t-1} + (1-\beta)\cdot(1-\gamma)\cdot \nabla L(\theta_t), \\ \eta_t = \dfrac{\eta}{\sqrt{v_t} + \epsilon}, \\ v_t = v'_t + \beta^2 \cdot \nabla L^2(\theta_t), \\ v'_t = \lambda \cdot v'_{t-1} + \nabla L(\theta_t)^\infty. \end{cases} \tag{18}$$

5 Convergence analyses

In this section, we prove the convergence of the optimizers with the Unbiased strategy and the UQN strategy, respectively.

Theorem 3 *The MBOs improved by the Unbiased strategy accelerate convergence iff*

$$k_i < \tau \cdot \frac{\mu^2}{2m},$$

where $\tau = 1 - \gamma \cdot (1 - \omega')$, m *is the mass of the particles and* μ *is the friction coefficient of Newton force field.*

Proof Qian etc. [21] think that the gradient descent with a momentum term is equivalent to a Newtonian particle moving through a viscous medium under the influence of a conservative force field. They equate the parameter update rule of Momentum to a Newton equation, which can be written in the following form:

$$\begin{cases} \theta_{t+1} = \theta_t - \eta_t \cdot g_t, \\ g_t = \dfrac{m}{(\Delta t)^2} \cdot g_{t-1} + \nabla L(\theta_t), \\ \eta_t = \dfrac{(\Delta t)^2}{m + \mu \cdot \Delta t}. \end{cases} \tag{19}$$

According to Eq. (1) and Eq. (19), Qian etc. [21] obtain the following formula:

$$\begin{cases} m = \gamma \cdot (\Delta t)^2, \\ \mu = \dfrac{(1 - \gamma \cdot \eta) \cdot \Delta t}{\eta}. \end{cases}$$

And they proof that the Momentum accelerates convergence iff

$$k_i < \frac{\mu^2}{2m} = \frac{(1 - \gamma \cdot \eta)^2}{2\gamma\eta^2}.$$

Similarly, according to Eq. (10), the update rule for MBOs with the Unbiased strategy is rewritten as:

$$\begin{cases} \theta_{t+1} = \theta_t - \eta_t \cdot g_t, \\ g_t = \dfrac{m'}{(\Delta t)^2 \cdot \tau} \cdot g_{t-1} + \nabla L(\theta_t), \\ \eta_t = \dfrac{(\Delta t)^2 \cdot \tau}{m' + \mu' \cdot \Delta t}, \end{cases} \tag{20}$$

where $\tau = 1 - \gamma \cdot (1 - \omega')$. According to Eq. (19) and Eq. (20), we obtain the formula as follows:

$$\begin{cases} m' = \gamma \cdot (\Delta t)^2 \cdot \tau, \\ \mu' = \dfrac{(1 - \gamma \cdot \eta) \cdot \tau \cdot \Delta t}{\eta}. \end{cases}$$

So, the MBOs with the Unbiased strategy accelerate convergence iff

$$k_i < \frac{\mu'^2}{2m'} = \tau \cdot \frac{(1 - \gamma \cdot \eta)^2}{2\gamma\eta^2} = \tau \cdot \frac{\mu^2}{2m}.$$

□

Lemma 2 The MBOs improved by the UQN strategy accelerate convergence iff

$$k_i < \tau \cdot \frac{\mu^2}{2m},$$

where $\tau = 1 - \gamma \cdot (1 - \beta)$.

Proof According to Eq. (15), we know $\omega' = \beta$. Similar to Theorem 3, we reconstruct Eq. (15) in the form of Newton equation and deduce m' and μ'. So we obtain that the MBOs improved by the UQN strategy accelerate convergence iff $k_i < \tau \cdot \frac{\mu^2}{2m}$, where $\tau = 1 - \gamma \cdot (1 - \beta)$. □

According to Theorem 3 and Lemma 2, it is theoretically guaranteed that our proposed Unbiased strategy and UQN strategy are convergent and can be applied to the widely used MBOs mentioned in this paper.

6 Experiments

We compare the widely used MBOs with their Unbiased and UQN variants by conducting the image classification experiments under the same datasets and models. We choose the classification problem and use softmax regression, multi-layer perceptron, and convolutional neural networks as training models. All MBOs are implemented within Pytorch[1] and the experiments are conducted on a server with Intel Xeon Silver 4210R and Nvidia GPU Quadro RTX 8000.

6.1 Experimental setting

We choose the following three popular datasets for image classification.

- **MNIST dataset (MD) [37]** is a handwritten digital image dataset that contains samples with 10 classes. It is divided into training set and test set, of which the training set contains $60,000$ samples and the test set contains $10,000$ samples. Each sample contains 28×28 pixels.
- **CIFAR-**10 **dataset (CD) [38]** contains color images with 10 classes. It is also divided into training set and test set, of which the training set contains $50,000$ samples and the test set contains $10,000$ samples. The image size is 32×32 pixels.

[1] https://pytorch.org/

- **ILSVRC2012-10% dataset (ID) [39]** consists of $1,000$ categories with approximately $1,000$ images per category, totaling approximately $1,200,000$ training images, $50,000$ validation images, and $150,000$ test images. We randomly select 10% of the class. The image size is 256×256 pixels.

The training models that are fed into optimizers are as follows:

- **Logistic Regression model (LM) [40]** is favored by the deep learning field for its simplicity, parallelization, and strong interpretation.
- **VGG-16 model (VM) [41]** contains 16 layers, 13 convolution layers, and three fully connected layers. It is widely used in image defogging, super-resolution style migration, etc.
- **ResNet-101 model (RM) [9]** contains an input convolution layer, 99 building blocks, and a full connection layer. It solves the problem of deep layer effect descent and is widely adopted in image and text fields.

Next, we determine the hyperparameters through parameter sweeping. The sweep grids for MBOs are as follows:

$$\eta \in \{0.001, 0.01, 0.10, 0.25, 0.50, 0.75, 0.90\},$$
$$\gamma \in \{0, 0.25, 0.50, 0.75, 0.90, 0.95, 0.99, 0.999\},$$
$$\beta \in \{0, 0.25, 0.50, 0.75, 0.90, 0.95, 0.99, 0.999\}.$$

Fig. 7 shows the effect of hyperparameters (η, γ, and β) on model accuracy under the same experimental setting. We select the optimal hyperparameters: $\eta = 0.001$, $\gamma = 0.9$, and $\beta = 0.5$.

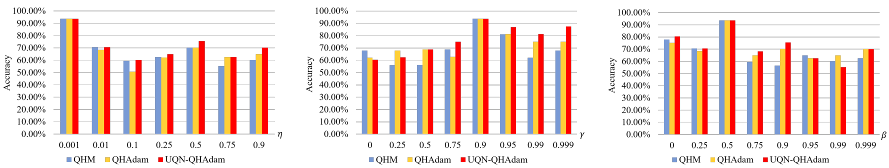


Fig. 7: Effect of hyperparameters (η, γ, and β) on model accuracy. (base on VM-CD)

6.2 Experimental results

In this subsection, we show the effect of both Unbiased strategy and UQN strategy. Tables 3–6 show the accuracies and corresponding epochs of models after convergence. Tables 3–6 indicate that by applying the Unbiased strategy and the UQN strategy, the accuracies of most of training models either increase or keep the same, and most of their numbers of epochs decrease, which means that these strategies accelerate convergence without sacrificing accuracy.

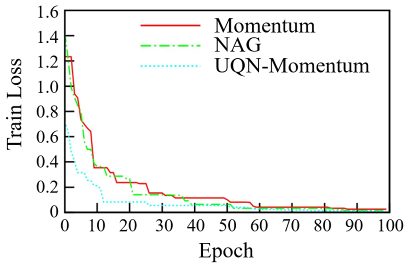
Tables 3 and 4 show the effect of Unbiased strategy on Momentum, QHM, and NAG. From Table 3, we can see that U-Momentum improves the accuracies of LM

Table 3: Accuracy of Unbiased strategy.

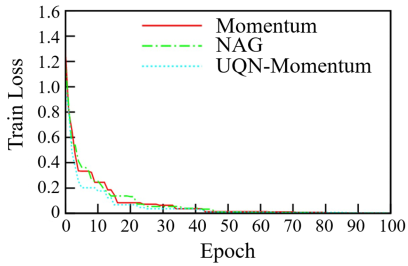
Optimizers	Accuracy (Model - Dataset)				
	LM - MD	VM - CD	VM - ID	RM - CD	RM - ID
Momentum	86.90%	87.50%	81.40%	93.75%	82.10%
U-Momentum	**87.49%**	**93.75%**	**82.10%**	**93.75%**	**82.10%**
QHM	84.23%	93.75%	82.10%	93.75%	82.10%
U-QHM	**85.35%**	**93.75%**	**82.10%**	**93.75%**	**82.10%**
NAG	87.31%	93.75%	82.10%	87.50%	78.50%
U-NAG	**87.54%**	**93.75%**	**82.10%**	**87.50%**	**81.40%**

Table 4: Convergence of Unbiased strategy.

Optimizers	Iteration or Epoch (Model - Dataset)				
	LM - MD	VM - CD	VM - ID	RM - CD	RM - ID
Momentum	1300	83	92	22	35
U-Momentum	**1100**	**67**	**75**	**16**	**29**
QHM	1100	71	69	36	23
U-QHM	**800**	**59**	**52**	**25**	**16**
NAG	1300	87	82	34	28
U-NAG	**900**	**68**	**59**	**28**	**22**



(a) VGG-16 loss



(b) ResNet-101 loss

Fig. 8: Train loss of using VGG-16 and ResNet-101 on CIFAR-10.

Table 5: Accuracy of UQN strategy.

Optimizers	Accuracy (Model - Dataset)				
	LM - MD	VM - CD	VM - ID	RM - CD	RM - ID
Momentum	86.90%	87.50%	81.40%	93.75%	82.10%
QHM	84.23%	93.75%	82.10%	93.75%	82.10%
NAG	**87.31%**	**93.75%**	82.10%	87.50%	78.50%
UQN-Momentum	86.76%	93.00%	**82.10%**	**93.75%**	**82.10%**
Adam	89.86%	93.75%	81.40%	81.25%	82.10%
QHAdam	89.08%	93.75%	82.10%	81.25%	82.10%
UQN-Adam	**90.70%**	**93.75%**	**82.10%**	**93.75%**	**82.10%**
Nadam	88.69%	93.75%	82.10%	93.75%	82.10%
UQN-Nadam	**89.24%**	**93.75%**	**82.10%**	**93.75%**	**82.10%**
AdaMax	87.64%	93.00%	81.40%	93.75%	81.40%
UQN-AdaMax	**88.12%**	**93.00%**	**81.40%**	**93.75%**	**81.40%**

Table 6: Convergence of UQN strategy.

Optimizers	Iteration or Epoch (Model - Dataset)				
	LM - MD	VM - CD	VM - ID	RM - CD	RM - ID
Momentum	1300	83	92	22	35
QHM	1100	71	69	36	23
NAG	1300	87	82	34	28
UQN-Momentum	**600**	**50**	**49**	**22**	**14**
Adam	1300	94	86	22	30
QHAdam	400	90	69	13	20
UQN-Adam	**300**	**38**	**41**	**9**	**12**
Nadam	800	64	76	18	25
UQN-Nadam	**600**	**45**	**69**	**16**	**20**
AdaMax	900	68	82	23	28
UQN-AdaMax	**800**	**56**	**64**	**15**	**25**

on MD, and VM on CD and ID. U-QHM and U-NAG also improve accuracies of LM on MD. Meanwhile, as Table 4 shows, the number of epochs decrease significantly since the Unbiased strategy adjust the direction of g_t to be consistent with the direction of gradient. For the linear model LM on dataset MD (i.e. LM - MD), the improved optimizers by using the Unbiased strategy save on average approximately 24% epochs than the originals. For VM on datasets CD and ID (i.e. VM - CD and VM - ID), the improved optimizers save on average about 22% epochs. For RM on datasets CD and ID (i.e. RM - CD and RM - ID), the improved optimizers save on average about 24% epochs.

Fig. 8 shows the comparison of the train loss of VM and RM by using different optimizers Momentum, NAG, and UQN-Momentum. Clearly, UQN-Momentum contribute smaller train loss than Momentum and NAG. According to Fig. 8, the UQN-Momentum curve flattens out earlier, which indicates that UQN-Momentum converges faster than Momentum and NAG.

Finally, we show the effect of applying the UQN strategy on MBOs. As shown in Tables 5 and 6, the UQN strategy significantly accelerates convergence without affecting accuracy. It is worth noting that the accuracies of UQN-Momentum using LM on MD and VM on CD are slightly lower than those of NAG. Because UQN-Momentum falls too fast and misses the global minimum point. However, the epochs of UQN-Momentum are saved on average about 49% than those of NAG. The accuracies of the remaining improved optimizers are all higher or equal to those of the previous optimizers. In terms of the number of epoch, we find that the improved optimizers are optimal in intra-group comparison.

Notice that, from Tables 3 and 5, we can see that the training model RM is special since Momentum, U-Momentum, and UQN-Momentum achieve the same accuracy for it, which indicates that Momentum is good enough for RM. Compare with the other two training models LM and VM, RM constructs a parameter space with fewer minimum points, so all the three optimizers can reach convergence. It is interesting to see for the training model RM on CD, among the three optimizers, U-Momentum spends the least number of epochs to reach convergence, which Momentum and UQN-Momentum spend the same (see Tables 4 and 6). Table 6 shows the reason. For

RM on CD, Momentum spends 22 epochs while QHM spends 36 epochs and NAG spends 34 epochs to reach convergence. Both QHM and NAG spend more number of epochs than Momentum, therefore, by combining with the Unbiased strategy, UQN-Momentum improves the three optimizers Momentum, QHM, and NAG.

7 Conclusion

In this paper, we first put forward a general form of update rules for the widely used MBOs. Then, we propose an Unbiased strategy that is applied to MBOs and enables them to accelerate convergence. On this basis, combined with Quasi-hyperbolic and Nesterov-gradient, the UQN strategy is proposed to further improve MBOs. In addition, we prove that the Unbiased strategy and UQN strategy converge theoretically. Finally, we demonstrate the effectiveness of the Unbiased strategy and UQN strategy through several comparative experiments. Potential area of future work is to research a relationship between gradient and batch size base on UQN strategy.

Declarations

- Ethical Approval and Consent to participate
  Not applicable.
- Human and Animal Ethics
  Not applicable.
- Consent for publication
  Not applicable.
- Availability of supporting data
  The data sets supporting the results of this article are included within the article.
- Competing interests
  The authors have no relevant financial or non-financial interests to disclose.

- Funding
  The work is partially supported by the National Key Research and Development Program of China (2020YFB1707901), National Natural Science Foundation of China (62072088, 61991404), Ten Thousand Talent Program (ZX20200035), and Liaoning Distinguished Professor (XLYC1902057).

- Authors' contributions
  Weiwei Cheng and Xiaochun Yang wrote the main manuscript text and Bin Wang and Wei Wang contributed ideas, prepared Figures 1-9, and proofread the paper. All authors reviewed the manuscript.

- Acknowledgements
  The work is partially supported by the National Key Research and Development Program of China (2020YFB1707901), National Natural Science Foundation of China (62072088, 61991404), Ten Thousand Talent Program (ZX20200035), and Liaoning Distinguished Professor (XLYC1902057).

- Authors' information
  Weiwei Cheng, Xiaochun Yang, and Bin Wang: School of Computer Science and Engineering, Northeastern University, Shenyang, Liaoning, China.

Wei Wang: Information Hub, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, Guangdong, China; The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong.

References

[1] Nebel, B.: On the compilability and expressive power of propositional planning formalisms. J. Artif. Intell. Res. **12**, 271–315 (2000)

[2] Lessard, L., Recht, B., Packard, A.K.: Analysis and design of optimization algorithms via integral quadratic constraints. SIAM J. Optim. **26**(1), 57–95 (2016)

[3] Zhong, H., Chen, Z., Qin, C., Huang, Z., Zheng, V.W., Xu, T., Chen, E.: Adam revisited: a weighted past gradients perspective. Frontiers Comput. Sci. **14**(5), 145309 (2020)

[4] Jin, D., He, J., Chai, B., He, D.: Semi-supervised community detection on attributed networks using non-negative matrix tri-factorization with node popularity. Frontiers of Computer Science **15**(4), 1–11 (2021)

[5] Ye, Y., Gong, S., Liu, C., Zeng, J., Jia, N., Zhang, Y.: Online belief propagation algorithm for probabilistic latent semantic analysis. Frontiers Comput. Sci. **7**(4), 526–535 (2013)

[6] Tan, Z., Chen, S.: On the learning dynamics of two-layer quadratic neural networks for understanding deep learning. Frontiers Comput. Sci. **16**(3), 163313 (2022)

[7] Bühlmann, P., Yu, B.: Boosting with the l 2 loss: regression and classification. Journal of the American Statistical Association **98**(462), 324–339 (2003)

[8] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Commun. ACM **60**(6), 84–90 (2017)

[9] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)

[10] Gehring, J., Auli, M., Grangier, D., Yarats, D., Dauphin, Y.N.: Convolutional sequence to sequence learning. In: Proceedings of the 34th International Conference on Machine Learning, ICML 2017, vol. 70, pp. 1243–1252 (2017)

[11] Dong, Q., Niu, S., Yuan, T., Li, Y.: Disentangled graph recurrent network for document ranking. Data Sci. Eng. **7**(1), 30–43 (2022)

[12] He, J., Liu, H., Zheng, Y., Tang, S., He, W., Du, X.: Bi-labeled LDA: inferring

interest tags for non-famous users in social network. Data Sci. Eng. **5**(1), 27–47 (2020)

[13] Abburi, H., Parikh, P., Chhaya, N., Varma, V.: Fine-grained multi-label sexism classification using a semi-supervised multi-level neural approach. Data Sci. Eng. **6**(4), 359–379 (2021)

[14] Xue, H., Xu, H., Chen, X., Wang, Y.: A primal perspective for indefinite kernel SVM problem. Frontiers Comput. Sci. **14**(2), 349–363 (2020)

[15] Jain, P., Kakade, S.M., Kidambi, R., Netrapalli, P., Sidford, A.: Accelerating stochastic gradient descent. arXiv preprint arXiv:1704.08227 (2017)

[16] Cyrus, S., Hu, B., Scoy, B.V., Lessard, L.: A robust accelerated optimization algorithm for strongly convex functions. In: 2018 Annual American Control Conference, ACC 2018, pp. 1376–1381 (2018)

[17] Scoy, B.V., Freeman, R.A., Lynch, K.M.: The fastest known globally convergent first-order method for minimizing strongly convex functions. IEEE Control. Syst. Lett. **2**(1), 49–54 (2018)

[18] Kidambi, R., Netrapalli, P., Jain, P., Kakade, S.M.: On the insufficiency of existing momentum schemes for stochastic optimization. In: 6th International Conference on Learning Representations, ICLR 2018 (2018)

[19] Robbins, H., Monro, S.: A stochastic approximation method. The annals of mathematical statistics, 400–407 (1951)

[20] Polyak, B.T.: Some methods of speeding up the convergence of iteration methods. Ussr computational mathematics and mathematical physics **4**(5), 1–17 (1964)

[21] Qian, N.: On the momentum term in gradient descent learning algorithms. Neural Networks **12**(1), 145–151 (1999)

[22] Ruder, S.: An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747 (2016)

[23] Zhou, B., Liu, J., Sun, W., Chen, R., Tomlin, C.J., Yuan, Y.: pbsgd: Powered stochastic gradient descent methods for accelerated non-convex optimization. In: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020, pp. 3258–3266 (2020)

[24] Luo, L., Huang, W., Zeng, Q., Nie, Z., Sun, X.: Learning personalized end-to-end goal-oriented dialog. In: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, pp. 6794–6801 (2019)

[25] Wu, Y., He, K.: Group normalization. In: Computer Vision - ECCV 2018.

Lecture Notes in Computer Science, vol. 11217, pp. 3–19 (2018)

[26] Ma, J., Yarats, D.: Quasi-hyperbolic momentum and adam for deep learning. In: 7th International Conference on Learning Representations, ICLR 2019 (2019)

[27] Nesterov, Y.E.: A method for solving the convex programming problem with convergence rate o (1/kˆ 2). In: Dokl. Akad. Nauk Sssr, vol. 269, pp. 543–547 (1983)

[28] Reddi, S.J., Kale, S., Kumar, S.: On the convergence of adam and beyond. In: 6th International Conference on Learning Representations, ICLR 2018 (2018)

[29] Bottou, L., Curtis, F.E., Nocedal, J.: Optimization methods for large-scale machine learning. SIAM Rev. **60**(2), 223–311 (2018)

[30] Pouyanfar, S., Sadiq, S., Yan, Y., Tian, H., Tao, Y., Reyes, M.E.P., Shyu, M., Chen, S., Iyengar, S.S.: A survey on deep learning: Algorithms, techniques, and applications. ACM Comput. Surv. **51**(5), 1–36 (2019)

[31] Ben-Tal, A., Nemirovskii, A.: Lectures on Modern Convex Optimization - Analysis, Algorithms, and Engineering Applications. MPS-SIAM series on optimization, (2001)

[32] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: 3rd International Conference on Learning Representations, ICLR 2015 (2015)

[33] Dozat, T.: Incorporating nesterov momentum into adam. In: International Conference on Learning Representations Workshop (2016)

[34] Zeiler, M.D.: ADADELTA: an adaptive learning rate method. CoRR (2012)

[35] Baydin, A.G., Cornish, R., Martínez-Rubio, D., Schmidt, M., Wood, F.: Online learning rate adaptation with hypergradient descent. In: 6th International Conference on Learning Representations, ICLR 2018 (2018)

[36] Vapnik, V.N.: Adaptive and learning systems for signal processing communications, and control. Statistical learning theory (1998)

[37] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)

[38] Krizhevsky, A., Hinton, G., *et al.*: Learning multiple layers of features from tiny images. In: Tech Report (2009)

[39] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M.S., Berg, A.C., Fei-Fei, L.: Imagenet large scale visual recognition challenge. IJCV **115**(3), 211–252 (2015)

[40] Wright, R.E.: Logistic regression. (1995)

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