TOWARDS THE GRADIENT ADJUSTMENT BY LOSS STATUS FOR NEURAL NETWORK OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Gradient descent-based algorithms are crucial in neural network optimization, and most of them only depend on local properties such as the first and second-order momentum of gradients to determine the local optimization directions. As a result, such algorithms often converge slowly in the case of a small gradient and easily fall into the local optimum. Since the goal of optimization is to minimize the loss function, the status of the loss indicates the overall progress of the optimization but has not been fully explored. In this paper, we propose a loss-aware gradient adjusting strategy (LGA) based on the loss status. LGA automatically adjusts the update magnitude of parameters to accelerate convergence and escape local optimums by introducing a loss-incentive correction term monitoring the loss and adapting gradient experience. The proposed strategy can be applied to various gradient descent-based optimization algorithms. We provide theoretical analysis on the convergence rate and empirical evaluations on different datasets to demonstrate the effectiveness of our method.

1 INTRODUCTION

Deep neural networks have achieved great success and become a ubiquitous component in various fields such as computer vision (Yoo, 2015; Khan et al., 2018; Buhrmester et al., 2021) and natural language processing (Goldberg, 2017; Yin et al., 2017; Galassi et al., 2020). The training of neural networks relies heavily on the optimization algorithm. Although great progress has been made in the optimization algorithms, choosing a proper learning rate for the parameter updates and avoiding parameters from getting trapped in local optimum are still challenging. For the former, too small a learning rate leads to painfully slow convergence, while too large a learning rate hinders convergence and causes the loss function to fluctuate around the optimum, or even diverge (Ruder, 2016). For the latter, the local optimum usually makes it notoriously hard for parameters to escape, as the gradient is close to zero in all dimensions.

To address the above challenges, many methods have been proposed. Mini-batch Stochastic Gradient Descent (SGD) (Werbos, 1974) is probabily the most widely used optimization algorithm, which can well scale to large models and large datasets. However, SGD has a relatively slow convergence rate and easily falls into the local optimum (Ruder, 2016). Several variants of SGD, such as Momentum (Qian, 1999) and Nesterov (Sutskever et al., 2013), accelerate deep network training by employing the momentum of gradients to partially preserve the previous gradients. Gradients of the current batch are only used to fine-tune the optimization direction. In this way, the optimization is more stable, converges faster, and has some ability to pass the saddle points. Taking into account that the same learning rate may not be suitable for all parameters, some optimization algorithms such as Adagrad (Duchi et al., 2011) and Adadelta (Zeiler, 2012) adaptively assign the learning rates for different parameters according to their second-order momentum. RMSprop (Tieleman & Hinton, 2012) and Adam (Kingma & Ba, 2014) incorporate the adaptive learning rate into momentum gradient descent. Besides, second-order optimization algorithms such as the Newton's method utilize the second-order derivatives to further speed up the convergence. Nevertheless, such methods require calculating the inverse of the Hessian matrix and the computational cost is high. Moreover, to apply such methods, there are strict requirements for the objective function, i.e., they must have continuous first and second-order partial derivatives, which are not always satisfied when training neural networks in complicated tasks.

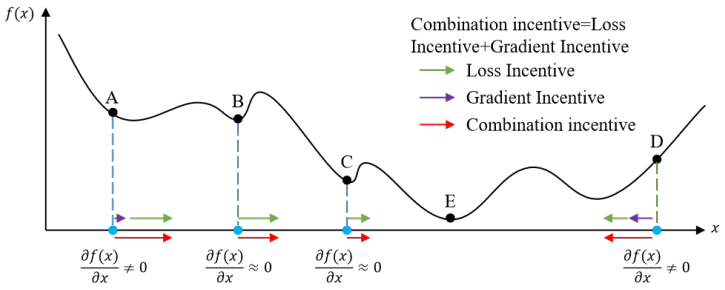


Figure 1: Effect of LGA for the parameter update. The strategy utilizes the information of loss function ($f(x)$) and gradient to adjust the update magnitude of parameters (x) to enhance the search capability for parameters. Especially for the state space (like A, B) with a small gradient and a long distance from the global optimum point (E), this strategy can provide additional excitation to evade the local optimum points.

These existing optimization algorithms generally determine the current descent direction only based on local properties of gradients (e.g., the first and second-order momentum and the Hessian matrix) in a greedy manner. If the parameters fall into local optimum during the early stage of training, this will be detrimental to the effectiveness of momentum terms and then harm the convergence efficiency of the algorithms. For instance, when the parameters fall into the local optimum (gradient is close to zero) during initialization, the first and second-order momentum cannot help the parameters to escape the local optimum quickly. Therefore, the local properties of gradients cannot always work well. A natural idea is to use the global state of optimization to improve the performance of algorithms, but it is not fully explored. Although it is difficult to detect the exact gap between the current estimate and the global optimum, the loss can be viewed as a direct indicator of global progress since it naturally measures how well the current estimate fits the training data without additional calculations. Hence, the loss may provide global information for tuning the current optimization.

In this paper, we propose a new loss-aware gradient adjusting strategy (LGA) to utilize such information. LGA can serve as a universal module to enhance various existing gradient descent-based optimization algorithms. Specifically, on the one hand, the magnitude of the gradient cannot fully reflect the distance between the current position and the optimal solution. In a flat area or the area near a saddle point, even if the gradient is small, it may still be far from the optimal solution and a large learning rate helps to go through such areas quickly. On contrary, since the goal of optimization is to minimize the loss function, if the loss is large, the current parameters are far from the optimal parameters, thus larger learning rates are required to take a larger step to avoid some local optimum. While a smaller loss means that the current parameters are closer to the optimal solutions, so the learning rate should be reduced accordingly. Inspired by this intuition, our strategy, as shown in Fig.1, adaptively adjusts the learning rate for updating parameters by introducing a loss-incentive correction term, where the weight of this term is proportional to the loss.

The contributions of this paper are summarized as follows:

1. We propose a novel strategy to adjust the magnitude of gradients by considering the loss status, which assist in assessing the global state of the optimization. The update step is larger when the loss has not been sufficiently reduced, thereby avoiding some local minimum points with large losses.

2. We propose the loss-incentive correction term to improve the performance of algorithms. It adaptively provides incentives to escape some local optimums by tracking the loss status and can serve as a general module that can be incorporated into various gradient-based optimization algorithms.

3. We conduct extensive experiments on different tasks including image classification, image segmentation, AutoEval, and video sentence grounding task. The empirical evaluations show the proposed strategy can improve different algorithms.

2 RELATED WORK

In machine learning, improving optimization algorithms to overcome the local optimal solutions is a crucial problem, and many efforts have been done (Qian, 1999; Duchi et al., 2011; Tieleman & Hinton, 2012; Kingma & Ba, 2014; Schulman et al., 2017). Momentum (Qian, 1999) was designed to simulate motion inertia by adding a term containing historical gradient experience. Adagrad (Duchi et al., 2011) can automatically adjust the learning rate by assigning different learning rates to parameters according to the accumulation of historical gradient experience. Based on Adagrad, RMSprop (Tieleman & Hinton, 2012) was designed to reduce the cumulative influence of historical gradient experience by assigning weight to the experience of historical gradient and the current descending gradient. Adam (Kingma & Ba, 2014) was proposed to dynamically set different learning rates for parameters by using the first and second-order momentum of gradients. The main goal of such approaches is adding momentum terms and dynamically adjusting the learning rate, which is not the focus of our current work.

In this paper, we mainly aim to investigate how to improve the search capability of algorithms by leveraging the loss status, and little research has been conducted in the optimization about it. Closely related are the works of (Schulman et al., 2017; Amid et al., 2022). Schulman et al. (2017) proposed clipped surrogate objective function to enhance the search capability for parameters. Amid et al. (2022) focus on improving the performance of first-order optimizers and proposed a layer-wise loss construction framework to enhance backpropagation. Unlike the above studies optimizing a surrogate objective function to improve the algorithms, this work focus on adjusting the gradient information based on the status of the objective function.

3 METHOD

Generally, the target of optimization algorithms is to minimize the loss function f_x with respect to x of a neural network. For most existing algorithms, each update of x can be formulated as follows:

$$x_{\text{t+1}} = x_t - \gamma_1 \left(f_x, x\right) \cdot \nabla f_{x_t} \tag{1}$$

where x_t is defined as the parameters of a neural network at step t and ∇f_{x_t} is defined as the first-order partial derivatives of f_{x_t} with respect to x_t. We define $\gamma_1 \left(f_x, x\right)$ as the adjustment factor to the descent gradient. Note that $\gamma_1 \left(f_x, x\right)$ can be either a fixed learning rate or a hyper-parameter consisting of the learning rate, the first and second-order momentum of the gradient. For example, $\gamma_1 \left(f_x, x\right)$ is a fixed value in SGD, while it is a variable value concerning the bias-correction terms, the first and second-order moment estimation in Adam. For the convenience of expression, it is simplified as γ_1 below, and the variation magnitude of x_t is defined as $\triangle x_t = x_{t+1} - x_t$.

Although great progress has been made by using various characteristics of gradients to design optimization algorithms, the status of the loss indicating the overall progress of the optimization has not been fully explored. In this section, we propose a loss-aware gradient adjusting strategy based on the loss status to enhance the global search capability of algorithms.

3.1 LOSS-AWARE GRADIENT ADJUSTING STRATEGY

As shown in Fig. 1, for some locally optimal solutions ($e.g. B$) with a small gradient and a long distance from the globally optimal solution (E), the energy provided by the gradient may not be enough to support parameters to jump out of the local traps. However, the loss provides additional information in these states. Inspired by it, the starting point of LGA is to realize the adaptive adjustment of parameter optimization capacity by using loss status to perceive the global information. More concretely, this strategy combines with the current loss status to automatically adjust the updated magnitude of parameters to enhance the search capability of algorithms. The specific design of LGA is shown in Eq.2.

$$x_{\text{t+1}} = x_t - \left[\gamma_1 \cdot \nabla f_{x_t} + \gamma_2 \cdot \frac{\Omega \left(f_{x_t}\right)}{\varphi \left(\nabla f_{x_t}\right)}\right] \tag{2}$$

Where γ_2 is defined as the learning rate of the loss-incentive correction term, which is related to the variation of the loss status. We define $\Omega \left(f_{x_t}\right)$ is the estimation of the gap between the current

loss status and the target status. $\varphi\left(\nabla f_{x_t}\right)$ is defined as the pre-processing process for ∇f_{x_t} to avoid $\nabla f_{x_t} = 0$. The essence of this strategy is introducing a correction term containing loss status information to increase learning incentive in the gradient descent-based algorithms.

As for some specific cases, the expected loss f_{x^*} may be far from zero (e.g., empirical loss plus a regularization term or the existence of noise), where although the loss status can bring great incentive information, the value of this status is low. $\Omega(\cdot)$ tackles this problem with truncation processing. Specifically, if $f_x \notin [\lfloor \tilde{f^*} \rfloor, \lceil \tilde{f^*} \rceil]$, $\lfloor \tilde{f^*} \rfloor$ and $\lceil \tilde{f^*} \rceil$ are the lower bound and the upper bound of the range of f_{x_t}, respectively, then $\Omega(f_x) = 0$, else $\Omega(f_x) = f_x - \lfloor \tilde{f^*} \rfloor$.

Nonetheless, for most real-world problems, it is infeasible to obtain the expectations of the upper and lower bounds of loss as prior knowledge. Generally, the final values of the objective function are different for different tasks, so setting universal proper upper and lower bounds are challenging and not suitable for learning. For the upper bound, since the objective function is usually not upper bounded and the value of it is to determine the reasonableness of the loss status, so we take the value of the first loss ($\lceil \tilde{f^*} \rceil = f_{x_0}$). As for the lower bound, which is used as a measure of the gap between the current status and the target status, we design a progressive inference mechanism to infer the reasonable lower bound. Let $\hat{\mathbb{E}}\left[\lfloor \tilde{f_t^*} \rfloor\right]$ be the estimation of $\lfloor \tilde{f_t^*} \rfloor$ at t and we wish to estimate it using a moving average of $f(x)$. The reason for using the moving average is that the objective function will gradually converge to the target status with increasing epochs. So the larger the epoch, the higher the reference value of the objective function for estimating the target status. We initialize the expectation of the lower bound as $\hat{\mathbb{E}}\left[\lfloor \tilde{f_0^*} \rfloor\right] = 0$. The update at step t ($t \geq 1$) of $\hat{\mathbb{E}}\left[\lfloor \tilde{f_t^*} \rfloor\right]$ can be written as a function of f_{x_t}:

$$\begin{aligned} \hat{\mathbb{E}}\left[\lfloor \tilde{f_t^*} \rfloor\right] &= \frac{1}{\log(t+10)} \cdot \hat{\mathbb{E}}\left[\lfloor \tilde{f_{t-1}^*} \rfloor\right] + \left(1 - \frac{1}{\log(t+10)}\right) \cdot f_{x_t} \\ &= \sum_{i=1}^{t} \left[\left(1 - \frac{1}{\log(i+10)}\right) \cdot \prod_{j=1}^{t-1} \frac{1}{\log(j+10)} \cdot f_{x_i} \right] \end{aligned} \tag{3}$$

In Eq.3, the weight term $\left(\left(1 - \frac{1}{\log(i+10)}\right) \cdot \prod_{j=1}^{t-1} \frac{1}{\log(j+10)}\right)$ decreases gradually as t increases. This property enables giving larger weights to recent loss data and fewer weights to distant loss data, which is following the variation characteristics of the loss status — the loss gradually approximates the expected value as training proceeds. Taking the estimated expectation of f_x into $\Omega(f_x)$:

$$\Omega\left(f_{x_t}\right) = \begin{cases} f_{x_t} - \hat{\mathbb{E}}\left[\lfloor \tilde{f_t^*} \rfloor\right] & f_{x_t} - \hat{\mathbb{E}}\left[\lfloor \tilde{f_t^*} \rfloor\right] \geq \varepsilon \wedge f_{x_t} \leq \lceil \tilde{f^*} \rceil \\ 0 & f_{x_t} - \hat{\mathbb{E}}\left[\lfloor \tilde{f_t^*} \rfloor\right] < \varepsilon \vee f_{x_t} > \lceil \tilde{f^*} \rceil \end{cases} \tag{4}$$

Where ε is a numerical stability constant, which can ensure there will be a gap between $f(x_t)$ and $\hat{\mathbb{E}}\left[\lfloor \tilde{f_t^*} \rfloor\right]$ when the loss-incentive correction term acts. Otherwise, if $f_{x_t} \approx \hat{\mathbb{E}}\left[\lfloor \tilde{f_t^*} \rfloor\right]$, the incentive term has essentially no effect for parameters update. Similarly, we define $\varphi(\nabla f_{x_t})$ as follows:

$$\varphi\left(\nabla f_{x_t}\right) = \begin{cases} \nabla f_{x_t} + \xi & 0 \leq \nabla f_{x_t} \\ \nabla f_{x_t} - \xi & 0 > \nabla f_{x_t} \end{cases} \tag{5}$$

In Eq.5, ξ is also a numerical stability constant, which can keep the denominator of the loss-incentive correction term in a non-zero state. Algorithm 1 outlines the key point of LGA.

3.2 Convergence Analysis

We analyze the convergence of LGA using the assumption in Kingma & Ba (2014). For any unknown convex function given, the objective of the optimization algorithms is to obtain the parameter x^* that minimizes the value of the objective function. In the convergence analysis of the loss-aware

Algorithm 1 The loss-aware gradient adjusting strategy. ∇f_{x_t} indicates the gradient information of the parameters x_t at the time step t. ξ and ε indicates a very small positive number. $\triangle x_t$ indicates the variation magnitude of x_t.

Require: γ_1: The adjustment factor of the algorithm to the descent gradient
Require: γ_2: The learning rate of the loss-incentive correction term
Require: f_{x_t}: The objective function with respect to x_t
Require: x_0: Initial parameter vector under t = 0 (Parameter initialization)
while x_t not converged **do**
  $f_{x_t} \leftarrow x_t$ (Get the loss value of the model)
  $\nabla f_{x_t} \leftarrow f_{x_t}$ and x_t (Get gradients with respect to objective function)
  if $\gamma_1 ! = constant$ **then**
    $\gamma_1 \leftarrow \nabla f_{x_t}$ or $\nabla f_{x_t} \odot \nabla f_{x_t}$ (Get the adjustment factor to the descent gradient)
  end if
  if $\left(f_{x_t} > f_{x_{t-1}}\right) \wedge (t \geq 1)$ **then**
    $\gamma_2 = \gamma_2 / 10$ (Update the hyperparameter)
  end if
  $\gamma_2 \cdot \frac{\Omega(f_{x_t})}{\varphi(\nabla f_{x_t})} \longleftarrow \nabla f_{x_t}$ and f_{x_t} (Get the loss-incentive correction term)
  $\triangle x_t = \gamma_1 \cdot \nabla f(x_t) + \gamma_2 \cdot \frac{\Omega(f_{x_t})}{\varphi(\nabla f_{x_t})}$
  $x_{t+1} = x_t - \triangle x_t$ (Update parameters)
end while
return x_{t+1}(Resulting parameters)

gradient adjusting strategy, We define the distance between the parameter x_t and x^* as $R(t)$, and use $R(t)$ to evaluate the convergence effect of the strategy, which is defined as follows:

$$R(t) = \parallel x_t - x^* \parallel^2 \tag{6}$$

In appendix A.1, we prove that LGA can guarantee the convergence of $R(t)$, and give proof of the validity and rationality of it. In the process of proof, we also use some definitions to simplify our notation, where we define $\mu \in \mathbb{R}^+$ and $L \in \mathbb{R}^+$ as the coefficients of the strong convexity and the smoothness of the objective function, respectively. We define g_t as a vector representing the first-order partial derivative of the objective function concerning the parameter x_t. Besides, we define $\lambda_t = \gamma_1 + \gamma_2 \cdot \frac{\Omega(f_{x_t})}{\varphi(\nabla f_{x_t})^2}$ that represents the integrated learning factor under the optimization algorithms with LGA. $\gamma_2 \cdot \frac{\Omega(f_{x_t})}{\varphi(\nabla f_{x_t})^2}$ represents the overall learning compensation of the objective function for parameters update at t. Through derivation as shown in the appendix, for the update of parameter x_t at any time step, the result of the optimization algorithms can be expressed as $\parallel x_t - x^* \parallel^2 = (1 - \mu \cdot \gamma_1)^t \cdot \parallel x_0 - x^* \parallel^2 + O(t)$, while the result of the optimization algorithms with LGA can be written as $\parallel x_t - x^* \parallel^2 = \left[\prod_{i=1}^{t}(1 - \mu \cdot \lambda_i)\right] \cdot \parallel x_0 - x^* \parallel^2 + O(t)$.

Theorem 3.1 *Assuming that the gradient of a convex function f_x is bounded and $f_{x^*} = 0$, the distance between any x_t generated by algorithms with or without LGA is bounded, and $\mu, \lambda_t \in \mathbb{R}^+$ satisfy $0 \leq 1 - \mu \cdot \lambda_t$. For all $t \geq 1$, LGA can achieve the following guarantee:*

$$\begin{aligned} R(t) &\leq \left[\prod_{i=1}^{t}(1 - \mu \cdot \lambda_i)\right] \cdot \parallel x_0 - x^* \parallel^2 \\ &\leq (1 - \mu \cdot \gamma_1)^t \cdot \parallel x_0 - x^* \parallel^2 \end{aligned} \tag{7}$$

Theorem 3.1 indicates that the distance between the parameters and the global optimum is shortened by the loss-incentive correction term under the premise of the same time step. In particular, $\lambda_t (\lambda_t \geq \gamma_1)$ contains the information of the objective function, which has the property of adaptively adjusting the update step.

For the function f_x, there exists a constant M for such that $\nabla^2 f_x \leq M\mathbf{I}$. Based on Taylor's expansion, we can easily get $f_{x_t} - f_{x^*} \leq \frac{1}{2M} \parallel x_t - x^* \parallel^2 \leq \frac{1}{2M}(1 - \mu \cdot \gamma_1)^t \cdot \parallel x_0 - x^* \parallel^2$. A straightforward corollary is the following:

Corollary 3.1 *Assuming that the gradient of a convex function f_{x_t} is bounded, $f_{x^*} = 0$, the distance between any x_t generated by algorithms with or without LGA is bounded, and $\mu, \lambda_t \in \mathbb{R}^+$ satisfy $0 \leq 1 - \mu \cdot \lambda_t$. For all $t \geq 1$, we can obtain the following guarantee:*

$$\lim_{t \to \infty} \left[\frac{1}{2M}(1 - \mu \cdot \gamma_1)^t \cdot \parallel x_0 - x^* \parallel^2 \right] = 0 \tag{8}$$

Thus, $\lim_{t\to\infty}(f_{x_t} - f_{x^*}) = 0$ and $\lim_{t\to\infty} R(t) = 0$. Therefore, LGA can meet the convergence requirements of the optimization algorithms.

3.3 Discussion on the Mechanism of LGA

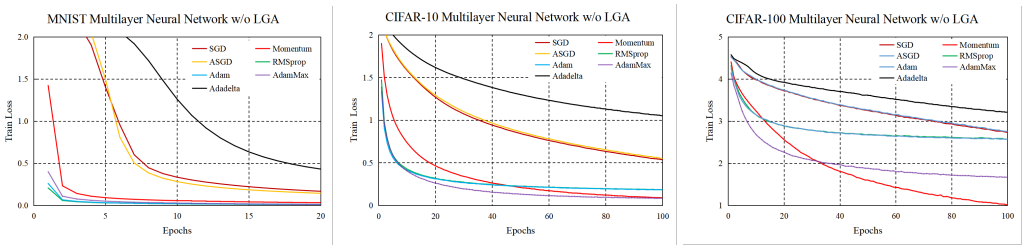
The essence of this strategy is introducing the loss-incentive term $\gamma_2 \cdot \frac{\Omega(f_{x_t})}{\varphi(\nabla f_{x_t})}$ in the parameter updates. $\Omega(f_{x_t})$ is a variable that is used to measure the magnitude of the output error caused by the parameters. With this feature, the loss-incentive term can adjust the amplitude of the extra incentive for parameters. Specifically, by tracking the loss status, this strategy can adjust the incentive degree for parameter updates. Then the parameters can get different degrees of training incentive according to the loss status, which can enhance the search capability of algorithms during training. The component $\frac{1.0}{\varphi(\nabla f_{x_t})}$ is used to estimate the probability that parameters fall into the local optimum. The reason for using the inverse of the gradient here is based on the consideration of the probability that parameters fall into the local optimum. Concretely, the parameter with a small gradient is more likely to fall into the local optimum or flat area, especially in the early training period. In this case, the energy only directly provided by the gradients may not be enough to support the parameters to find a new state quickly, and the extra energy is more likely to be required. Besides, this component can dynamically modify the vector directionality of the loss-incentive correction term to ensure that the loss-incentive term is excited in the direction of the gradient.

As discussed above, the loss-incentive correction term can provide extra energy to jump out of some local optimum solutions. More concretely, when the parameters fall into some local optimum solutions far from the global optimum, then $\gamma_2 \cdot \frac{\Omega(f_{x_t})}{\varphi(\nabla f_{x_t})} \gg \gamma_1 \cdot \nabla f_{x_t}$ and the excitation energy provided by the loss-incentive correction term can enable the parameters to jump out of those solutions. When the parameters are trapped in a local optimum close to the global optimum or the gradient is too large, the excitation energy provided by $\gamma_2 \cdot \frac{\Omega(f_{x_t})}{\varphi(\nabla f_{x_t})}$ is not helpful for the parameters avoiding the local optimum. Furthermore, if $\gamma_2 \cdot \frac{\Omega(f_{x_t})}{\varphi(\nabla f_{x_t})} \ll \gamma_1 \cdot \nabla f_{x_t}$, $\gamma_1 \cdot \nabla f_{x_t}$ dominates the parameter update and $\gamma_2 \cdot \frac{\Omega(f_{x_t})}{\varphi(\nabla f_{x_t})}$ has little influence for parameter updates. If the parameters are close to the global optimum, then $\gamma_2 \cdot \frac{\Omega(f_{x_t})}{\varphi(\nabla f_{x_t})} \approx 0$, which shows that the term has no negative effect on the convergence of the original optimization algorithm in the case. Taken together, by adding incentive based on the loss status, LGA gives the model more chances to escape and find a new local minimum, rather than be trapped in a local minimum or flat area.

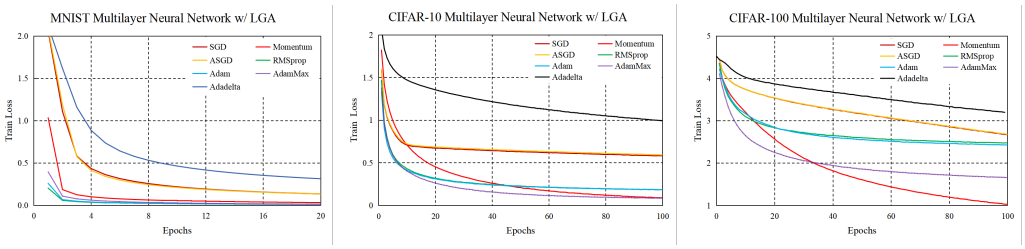
4 Experiment

4.1 Datasets and Baselines

Experimental setting. The image classification task is a typical nonlinear problem. To empirically validate the proposed strategies, we experiment on MNIST dataset (LeCun, 1998), CIFAR-10 dataset (Krizhevsky et al., 2009), CIFAR-100 dataset (Krizhevsky et al., 2009) and TinyImageNet (Deng et al., 2009). Besides, we relegate the additional results of the image segmentation task, the AutoEval task, the video sentence grounding task, and the autoencoder task to the appendix.



(a) Training loss of different algorithms without LGA.



(b) Training loss of different algorithms with LGA.

Figure 2: Results of LGA at different training stages on MNIST, CIFAR-10 and CIFAR-100.

Table 1: Comparison of different optimization algorithms without and with LGA on MNIST, CIFAR-10 and CIFAR-100. We report the classifier accuracy $(\%)$ of corresponding train sets.

Algorithm	MNIST		CIFAR-10		CIFAR-100	
	w/o LGA	w/ LGA	w/o LGA	w/ LGA	w/o LGA	w/ LGA
SGD	94.95	**95.95**	81.72	**83.40**	31.76	**31.77**
Momentum	98.98	**99.04**	**97.04**	97.03	72.59	**72.68**
ASGD	95.70	**96.01**	81.16	**82.80**	31.71	**32.19**
RMSprop	99.59	**99.60**	93.50	**93.52**	34.86	**35.66**
Adam	99.55	**99.57**	93.57	**93.59**	34.98	**37.65**
Adamax	99.63	**99.64**	**97.21**	97.18	55.74	**55.84**
Adadelta	88.59	**91.18**	62.12	**76.08**	22.78	**23.07**

MNIST has a training set of 60000 28×28 grayscale examples and a test set of 10000 examples in 10 classes and we use LeNet-5 (LeCun et al., 2015) as the model. CIFAR-10 consists of 60000 32×32 colour images in 10 classes and we use Densenet-40-12 (Huang et al., 2017) as the model. CIFAR-100 has 100 classes. Each class has 600 color images of size 32×32, of which 500 images are used as the training set and 100 images as the test set. We use the ResNet50 (He et al., 2016) for the classification task of it. TinyImageNet contains 200 object classes and each consisting of 500 training images and 50 validation images, and 50 test images. We used the training images and validation images to build the training and test sets for the classification task based on ResNet50.

Metric. We use the final loss status and accuracy performance as the evaluation criteria. Lower loss and higher accuracy correspond to better performance and vice versa.

Baselines. We use SGD, Momentum, ASGD, Adadelta, RMSprop, Adam, and Adamax for baselines. In practice, we use the same parameter initialization and the same random seed for each optimization algorithm when comparing the performance of the algorithms with or without LGA.

4.2 Experimental Results

To empirically evaluate the proposed strategy, we employ a learning rate of 0.001, $\varepsilon = 0.1$, $\xi = 0.01$ and carry out 5 experiments for any combination between the original algorithms and the proposed strategy on all datasets except TinyImageNet (a learning rate of 0.01 and 1 experiment), and report the average score as the experiment results. For all experiments, the batch size is set to 16 for

Table 2: Results on MNIST under different learning rates. The epochs of all algorithms are set to 20. We report the classifier accuracy $(\%)$ of the train set.

ALGORITHM	W/O LGA				W/ LGA			
	1E-1	1E-2	1E-3	1E-4	1E-1	1E-2	1E-3	1E-4
SGD	99.57	98.30	95.95	28.06	**99.81**	**99.07**	**95.95**	**86.44**
MOMENTUM	28.29	99.51	98.98	95.32	**28.32**	**99.87**	**99.04**	**96.03**
ASGD	99.84	98.33	95.70	28.59	**99.85**	**99.07**	**96.01**	**87.63**
RMSPROP	10.49	**97.02**	99.59	**99.09**	**10.62**	99.60	**98.84**	99.08
ADAM	10.44	97.71	95.55	**99.05**	**10.45**	**97.91**	**95.57**	99.01
ADAMAX	95.25	99.34	99.30	97.87	**95.97**	**99.52**	**99.64**	**97.88**
ADADELTA	99.56	96.08	88.59	17.77	**99.57**	**97.44**	**91.18**	**83.17**

Table 3: Results of different optimization algorithms on MNIST, CIFAR-10 and CIFAR-100. The epochs of original algorithms are set to 20, 100 and 100 respectively during training. while the epochs of algorithms with LGA depends on when the accuracy performance of the original algorithm is achieved or the epoch reaches the threshold. We report the classifier accuracy $(\%)$ of test sets and the ratio of epoch ($\frac{w/}{w/o} \in (0, 1]$).

Dataset	Algorithm	SGD	Momentum	ASGD	RMSprop	Adam	Adamax	Adadelta
MNIST	w/o LGA	95.47	98.78	96.02	98.91	99.01	99.03	89.62
	w/ LGA	**95.53**	**98.79**	**96.10**	**98.98**	**99.02**	**99.10**	**89.96**
	Ratio	0.76	0.89	0.9	0.29	0.48	0.69	0.63
CIFAR10	w/o LGA	75.71	85.97	75.21	85.57	87.90	88.72	62.12
	w/ LGA	**76.03**	**86.19**	**75.31**	**87.62**	**88.19**	**88.96**	**62.18**
	Ratio	0.85	0.69	0.82	0.58	0.45	0.64	0.82
CIFAR100	w/o LGA	40.29	70.49	**40.26**	45.07	46.00	66.52	29.03
	w/ LGA	**40.30**	**70.84**	40.06	**45.12**	**46.25**	**66.70**	**29.32**
	Ratio	0.98	0.86	0.98	0.74	0.60	0.95	0.86

TinyImageNet, 64 for MNIST, 128 for CIFAR-10, and CIFAR-100. The epoch is set to 20 for MNIST and the models are trained for 100 epochs for CIFAR-10, CIFAR-100, and TinyImageNet. The hyper-parameter γ_2 is set $1e^{-8}$ for 10 classification task (MNIST and CIFAR-10), $1e^{-9}$ for 100 classification task (CIFAR-100) and $1e^{-10}$ for 200 classification tasks (TinyImageNet). We focus on the performance of our strategy under the same initialization condition settings.

Results of the training sets. As shown in Fig.2, most algorithms with LGA can obtain better performances, especially in short-term conditions. The most likely reason for the situation is that some parameters fall into the local optimum solutions during training and cannot quickly escape it due to not enough update energy provided by optimization algorithms. Whereas, LGA can provide extra energy connecting with the loss status to help parameters escape some local optimum solutions. Compared with other algorithms, some algorithms (e.g., Adam) with LGA do not produce significant improvement, and the reason for this phenomenon is due to $\gamma_2 \cdot \frac{\Omega(f_{x_t})}{\varphi(\nabla f_{x_t})} \ll \gamma_1 \cdot \nabla f_{x_t}$ in Adam, then $\gamma_2 \cdot \frac{\Omega(f_{x_t})}{\varphi(\nabla f_{x_t})}$ has little positive influence on the parameter update, which is also reflected in Table 1. Besides, We observe that LGA does not yield noticeable system improvement in CIFAR-100. There are two reasons for this state of affairs: 1) γ_2 is set smaller; 2) for the dataset with more categories, the training data may vary greatly from batch to batch, which will make the model sensitive to parameter changes in the early stages of training and then limits the setting of γ_2. However, it does not affect the role of the loss-incentive correction term in improving the search capability of algorithms (see Table 1 and Table 2).

Table 1 shows the results of the comparison of the performance of different optimization algorithms without and with LGA on MNIST, CIFAR-10, and CIFAR-100. This table shows impressive results that most of the algorithms with LGA achieve better performance on the training set, which is consistent with the notion that LGA can effectively enhance the search capability of algorithms.

Table 2 shows that most algorithms with LGA can obtain better performance compared to the original algorithms under different learning rates. Especially, LGA makes algorithms only including the

Table 4: Results of different optimization algorithms on TinyImageNet. The epochs of algorithms with LGA depends on when the accuracy performance of the original algorithm is achieved or the epoch reaches the threshold (100). The classifier accuracy $(\%)$ of the validation set and the ratio of epoch ($\frac{w/}{w/o} \in (0, 1]$) are reported.

ALGORITHM	SGD	MOMENTUM	ASGD	ADAM	ADAMMAX
W/O LGA	48.59	48.57	48.18	41.44	25.56
W/ LGA	**49.67**	**50.45**	**48.23**	**41.49**	**26.30**
RATIO	0.55	0.14	0.50	0.66	0.16

Table 5: Comparison of different optimization algorithms under different learning rate schedulers on MNIST. We report the classifier accuracy $(\%)$ of train sets, the corresponding standard deviation (σ) and mean (μ). Higher accuracy, higher μ and smaller σ correspond to better performance.

Algorithm	CosineAnnealingLR		ExponentialLR		ReduceLROnPlateau		StepLR	
	w/o LGA	w/ LGA	w/o LGA	w/ LGA	w/o LGA	w/ LGA	w/o LGA	w/ LGA
SGD	91.16	**94.20**	91.16	**93.21**	94.95	**95.95**	76.68	**90.93**
Momentum	98.67	**98.72**	**98.67**	98.48	98.98	**99.07**	97.87	**97.98**
ASGD	92.73	**94.22**	92.73	**93.51**	95.70	**96.01**	81.86	**91.60**
RMSprop	99.98	99.98	**99.98**	99.95	**99.66**	99.60	99.86	**99.87**
Adam	99.98	99.98	99.98	99.98	99.55	**99.65**	99.81	99.81
Adamax	99.61	**99.62**	99.61	**99.93**	99.63	**99.64**	99.08	**99.10**
Adadelta	76.47	**88.42**	76.47	**99.44**	88.59	**91.18**	52.18	**85.46**
μ	94.09	**96.45**	94.09	**96.01**	96.72	**97.30**	86.76	**94.96**
σ	7.94	**4.05**	7.94	**4.40**	3.79	**2.94**	16.64	**5.23**

first-order momentum of gradients far surpass the original performance in a small learning rate, such as SGD and ASGD under the condition that the learning rate is equal to $1e^{-4}$. Besides, the result further shows that LGA can provide some robustness against choosing a small learning rate and enhance the optimization capacity of parameters with a large learning rate. Taken together, the above results suggest LGA can take advantage of the loss status to escape some local optimum solutions to achieve a better state during training.

Results of the test sets. In Table 3, we report the performance of LGA on several typical optimization algorithms. We clearly observe that most algorithms with LGA can accomplish the training task better and faster (the computational cost of LGA is about 6% of the original algorithm to the computational time) under the same initialization condition settings. Additionally, we report the best performance and the final performance of different algorithms in Table 6 and Table 7, respectively. These data indicate our method effectively improves the performance of algorithms.

To further verify the effectiveness of LGA, we experimented on the TinyImageNet with ResNet50 and experimented on the MNIST under different learning rate schedulers. As shown in Table 4, the algorithms with LGA accomplish the training faster and better under the condition of achieving the same test accuracy, on TinyImageNet which indicates that LGA provides an effective incentive for parameters during training by tracking the loss status. In Table 5, we compare the performance of several typical learning rate schedulers on MNIST. The algorithms with LGA mostly outperform the original algorithms. The results of μ and σ indicate that LGA performs better and achieves more consistent performance for different optimizers. Further, LGA can extract and make good use of the gradient experience and the loss status for the parameter updates.

5 CONCLUSION

In this paper, we propose a novel loss-aware gradient adjusting strategy based on the loss status, which tracks the loss status to modify the magnitude of the parameter updates and then improves the search capability of algorithms. The strategy has been evaluated in a wide range of experiments, and has shown good effectiveness and stability for different optimization algorithms. Our strategy has the potential to be extended for application to other tasks, which will be studied in the future.

REFERENCES

Ehsan Amid, Rohan Anil, and Manfred Warmuth. Locoprop: Enhancing backprop via local loss optimization. In *International Conference on Artificial Intelligence and Statistics*, pp. 9626–9642. PMLR, 2022.

Vanessa Buhrmester, David Münch, and Michael Arens. Analysis of explainers of black box deep neural networks for computer vision: A survey. *Machine Learning and Knowledge Extraction*, 3 (4):966–989, 2021.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Weijian Deng, Stephen Gould, and Liang Zheng. What does rotation prediction tell us about classifier accuracy under varying testing environments? In *International Conference on Machine Learning*, pp. 2579–2589. PMLR, 2021.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

Andrea Galassi, Marco Lippi, and Paolo Torroni. Attention in natural language processing. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

Jiyang Gao, Chen Sun, Zhenheng Yang, and Ram Nevatia. Tall: Temporal activity localization via language query. In *Proceedings of the IEEE international conference on computer vision*, pp. 5267–5275, 2017.

Yoav Goldberg. Neural network methods for natural language processing. *Synthesis lectures on human language technologies*, 10(1):1–309, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Salman Khan, Hossein Rahmani, Syed Afaq Ali Shah, and Mohammed Bennamoun. A guide to convolutional neural networks for computer vision. *Synthesis Lectures on Computer Vision*, 8(1): 1–207, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

Yann LeCun et al. Lenet-5, convolutional neural networks. *URL: http://yann. lecun. com/exdb/lenet*, 20(5):14, 2015.

Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12 (1):145–151, 1999.

Marcus Rohrbach, Michaela Regneri, Mykhaylo Andriluka, Sikandar Amin, Manfred Pinkal, and Bernt Schiele. Script data for attribute-based recognition of composite activities. In *European conference on computer vision*, pp. 144–157. Springer, 2012.

Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pp. 1139–1147. PMLR, 2013.

Tijmen Tieleman and G Hinton. Divide the gradient by a running average of its recent magnitude. coursera neural netw. *Mach. Learn*, 6:26–31, 2012.

Paul Werbos. New tools for prediction and analysis in the behavioral sciences. *Ph. D. dissertation, Harvard University*, 1974.

Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*, 2017.

Hyeon-Joong Yoo. Deep convolution neural networks in computer vision: a review. *IEIE Transactions on Smart Processing and Computing*, 4(1):35–43, 2015.

Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Hao Zhang, Aixin Sun, Wei Jing, and Joey Tianyi Zhou. Span-based localizing network for natural language video localization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 6543–6554, 2020.

A Appendix

A.1 Convergence Proof for Adaptive Gradient Strategy.

We define the objective function f which is convex for $x \in \mathbb{R}^d$, $\mu \in \mathbb{R}^+$ and $L \in \mathbb{R}^+$ is defined as the coefficients of the strong convexity and the smoothness of the objective function, respectively.

Lemma 1 *For* $\forall x, y \in \mathbb{R}^d, f_y \leq f_x + (\nabla f_x)^T \cdot (y - x) + \frac{L}{2} \parallel x - y \parallel^2$.

Lemma 2 *For* $\forall x, y \in \mathbb{R}^d, f_y \geq f_x + (\nabla f_x)^T \cdot (y - x) + \frac{\mu}{2} \parallel x - y \parallel^2$.

The above lemma can be used to prove the upper bound of the proposed strategy. And the proposed strategy can be written as follows:

$$x_{\text{t+1}} = x_t - \left[\gamma_1 \cdot \nabla f_{x_t} + \gamma_2 \cdot \frac{\Omega\left(f_{x_t}\right)}{\varphi\left(\nabla f_{x_t}\right)}\right]$$

In the equation above, ∇f_{x_t} is defined as the first-order partial derivative of f_{x_t} with respect to the parameter x at t. We also define γ_1 as the adjustment factor of the algorithm to the descent gradient. γ_1 is not just the learning rate, it could be a hyper-parameter composed of the learning rate, the first-order momentum of gradients, and the second-order momentum of gradients. γ_2 is defined as the learning rate of the loss-incentive correction term, which is related to the change of f_{x_t} (Section 3.1). We define the distance between the parameter x_t and x^* (the global optimum) at time t as $R(t)$, and use $R(t)$ to evaluate the convergence effect of the strategy, which is defined as follows:

$$R(t) = \parallel x_t - x^* \parallel^2$$

Theorem A.1 *Assuming that the gradient of a convex function f_x is bounded and $f_{x^*} = 0$, the distance between any x_t generated by algorithms with or without the adaptive gradient strategy is bounded, and $\mu, \lambda_t \in \mathbb{R}^+$ satisfy $0 \leq 1 - \mu \cdot \lambda_t$. For all $t \geq 1$, the adaptive gradient strategy can achieve the following guarantee:*

$$\begin{aligned} R(t) &\leq \left[\prod_{i=1}^{t}(1 - \mu \cdot \lambda_i)\right] \cdot \parallel x_t - x^* \parallel^2 \\ &\leq (1 - \mu \cdot \gamma_1)^t \cdot \parallel x_t - x^* \parallel^2 \end{aligned}$$

Proof A.1 *We will prove the convergence for adaptive gradient strategy. The policy equation can be converted to the following.*

$$x_{t+1} = x_t - \left[\gamma_1 \cdot \nabla f_{x_t} + \gamma_2 \cdot \frac{\Omega\left(f_{x_t}\right)}{\varphi\left(\nabla f_{x_t}\right)}\right]$$

Let $g_t = \nabla f_{x_t}$. *Then we have*

$$g_t = (x_t - x_{t+1}) \cdot \frac{1}{\gamma_1 + \gamma_2 \cdot \frac{\Omega(f_{x_t})}{\varphi\left(\nabla f_{x_t}\right)^2}}$$

To construct the relationship between x_t *and* x^*, *let's multiply both sides of this equation by the same term* $(x_t - x^*)$,

$$g_t^T \cdot (x_t - x^*) = \left[(x_t - x_{t+1}) \cdot \frac{1}{\gamma_1 + \gamma_2 \cdot \frac{\Omega(f_{x_t})}{\varphi\left(\nabla f_{x_t}\right)^2}}\right]^T \cdot (x_t - x^*)$$

To simplify the notation, we define $\lambda_t = \gamma_1 + \gamma_2 \cdot \frac{\Omega(f_{x_t}))}{\varphi\left(\nabla f_{x_t}\right)^2}$, $\lambda_t \geq \gamma_1 > 0$.

$$\begin{aligned} g_t^T \cdot (x_t - x^*) &= \frac{1}{\lambda_t}(x_t - x_{t+1})^T \cdot (x_t - x^*) \\ &= \frac{1}{2\lambda_t} \cdot \left[\parallel x_t - x_{t+1} \parallel^2 + \parallel x_t - x^* \parallel^2 - \parallel x_{t+1} - x^* \parallel^2\right] \\ &= \frac{1}{2\lambda_t} \cdot \left[\lambda_t^2 \cdot \parallel g_t \parallel^2 + \parallel x_t - x^* \parallel^2 - \parallel x_{t+1} - x^* \parallel^2\right] \\ &= \frac{\lambda_t}{2} \cdot \parallel g_t \parallel^2 + \frac{1}{2\lambda_t} \cdot \parallel x_t - x^* \parallel^2 - \frac{1}{2\lambda_t} \cdot \parallel x_{t+1} - x^* \parallel^2 \end{aligned}$$

Based on Taylor's expansion and the strong convexity of the objective function (Lemma 2), $f_{x^*} \geq f_{x_t} + (\nabla f_{x_t})^T \cdot \parallel x^* - x_t \parallel + \frac{\mu}{2} \cdot \parallel x^* - x_t \parallel^2$, *then we can lower bound of* $g_t^T \cdot (x_t - x^*)$.

$$g_t^T \cdot (x_t - x^*) = -(\nabla f_{x_t})^T \cdot (x^* - x_t) \geq f(x_t) - f(x^*) + \frac{\mu}{2} \cdot \parallel x^* - x_t \parallel^2$$

The following inequality holds by using the lower bound.

$$f_{x_t} - f_{x^*} + \frac{\mu}{2} \cdot \parallel x^* - x_t \parallel^2 \leq \frac{\lambda_t}{2} \cdot \parallel g_t \parallel^2 + \frac{1}{2\lambda_t} \cdot \left[\parallel x_t - x^* \parallel^2 - \parallel x_{t+1} - x^* \parallel^2\right]$$

Rearrange the inequality and substitute $\parallel x_{t+1} - x^* \parallel^2$ *with* $R(t+1)$,

$$R(t+1) = \parallel x_{t+1} - x^* \parallel^2 \leq 2\lambda_t \cdot (f_{x^*} - f_{x_t}) + \lambda_t^2 \cdot \parallel g_t \parallel^2 + (1 - \mu \cdot \lambda_t) \cdot \parallel x^* - x_t \parallel^2$$

Regard $\left[2\lambda_t \cdot (f_{x^*} - f_{x_t}) + \lambda_t^2 \cdot \parallel g_t \parallel^2\right]$ *as noise* $O(t)$, *we can upper bound of the terms.*

$$\begin{aligned} 2\lambda_t \cdot (f_{x^*} - f_{x_t}) + \lambda_t^2 \cdot \parallel g_t \parallel^2 &\leq 2\lambda_t \cdot (f_{x_{t+1}} - f_{x_t}) + \lambda_t^2 \cdot \parallel g_t \parallel^2 \\ &\leq \left(-\lambda_t^2 \cdot \parallel g_t \parallel^2\right) + \lambda_t^2 \cdot \parallel g_t \parallel^2 \\ &= 0 \end{aligned}$$

Use the upper bound on $R(t)$, *we have*

$$\begin{aligned} R(t+1) &= \parallel x_{t+1} - x^* \parallel^2 \\ &\leq (1 - \mu \cdot \lambda_t) \cdot \parallel x^* - x_t \parallel^2 \\ &\leq \left\{1 - \mu \cdot \left[\gamma_1 + \gamma_2 \cdot \frac{\Omega(f_{x_t})}{\varphi\left(\nabla f_{x_t}\right)^2}\right]\right\} \cdot \parallel x^* - x_t \parallel^2 \end{aligned}$$

Therefore, the above equation indicates that the ability of parameters to approximate the global optimum is improved by the loss incentive correction term under the premise of the same time step. In other words, the strategy can accelerate the convergence rate of parameters.

From $\lambda_t \geq \gamma_1 > 0$, we can get it $1 - \mu \cdot \lambda_t \leq 1 - \mu \cdot \gamma_1$. Thus, we have the following inequality.

$$\begin{aligned} R(t) =& \parallel x_t - x^* \parallel^2 \\ &\leq (1 - \mu \cdot \lambda_t) \cdot \parallel x_{t-1} - x^* \parallel^2 \\ &\leq (1 - \mu \cdot \lambda_t) \cdot (1 - \mu \cdot \lambda_{t-1}) \cdot \parallel x_{t-2} - x^* \parallel \\ &\leq \left[\prod_{i=1}^{t} (1 - \mu \cdot \lambda_i) \right] \cdot \parallel x_0 - x^* \parallel^2 \\ &\leq (1 - \mu \cdot \gamma_1)^t \cdot \parallel x_0 - x^* \parallel^2 \end{aligned}$$

For the function f_x, there exists a constant M for such that $\nabla^2 f_x \leq M\mathbf{I}$. Based on Taylor's expansion, for t at any time, we can easily get $f_{x_t} - f_{x^*} \leq \frac{1}{2M} \cdot \parallel x_t - x^* \parallel^2 \leq \frac{1}{2M} \cdot (1 - \mu \cdot \gamma_1)^t \cdot \parallel x_0 - x^* \parallel^2$. Combined with the above analysis, the adaptive gradient strategy can achieve the following guarantee:

$$\begin{aligned} \lim_{t \to \infty} (f_{x_t} - f_{x^*}) &= \lim_{t \to \infty} \left[\frac{1}{2M} \cdot \parallel x_t - x^* \parallel^2 \right] \\ &= \lim_{t \to \infty} \left[\frac{1}{2M} \cdot (1 - \mu\gamma_1)^t \cdot \parallel x_0 - x^* \parallel^2 \right] \\ &= 0 \end{aligned}$$

Thus, $\lim_{t \to \infty} (f_{x_t} - f_{x^*}) = 0$ and $\lim_{t \to \infty} R(t) = 0$. Thereby, under the conditions above, the adaptive gradient strategy can meet the convergence requirements of the optimization algorithms.

A.2 Analysis under Adam

In the work of Kingma & Ba (2014), the authors made the following assumptions: 1) the function f is convex and has bounded gradient ($\|\nabla f\|_2 \leq G, \|\nabla f\|_\infty \leq G_\infty$), 2) the distance between any x_t is bounded ($\|x_i - x_j\|_2 \leq D$, i and $j \in 1, \cdots, T$, all $x \in \mathbb{R}^d$). α is the stepsize, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ are the exponential decay rates for the moment estimates. g_t^2 indicates the elementwise square $g_t \odot g_t$ in $Adam$. Besides, the corresponding functions are set as follows.

The estimation to the first-order momentum:

$$\begin{aligned} m_t &= \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \\ &= (1 - \beta_1) \sum_{s=1}^{t} (\beta_1^{t-s}) g_t \end{aligned}$$

The estimation to the second-order momentum:

$$\begin{aligned} v_t &= \beta_1 \cdot v_{t-1} + (1 - \beta_1) \cdot g_t^2 \\ &= (1 - \beta_2) \sum_{s=1}^{t} (\beta_2^{t-s}) g_t^2 \end{aligned}$$

After the bias correction, the first-order momentum and the second-order momentum

$$\hat{m_t} = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v_t} = \frac{v_t}{1 - \beta_2^t}$$

We also focus on the i-th dimension of the parameter vector $x_t \in \mathbb{R}_d$. From the update rules presented in $Adam$, subtract the scalar $x_{,i}^*$, then the distance ($S1_{t,i}$) between $x_{t+1,i}$ and $x_{,i}^*$ can be

written as:

$$
\begin{aligned}
S1_{t,i} &= (x_{t+1,i} - x^*_{,i})^2 \\
&= \left[(x_{t,i} - x^*_{,i}) - \left(\alpha_t \cdot \frac{1}{1-\beta_1^t} \cdot \frac{\beta_1 \cdot m_{t-1,i} + (1-\beta_1) \cdot g_{t,i}}{\sqrt{\hat{v_{t,i}}}} \right) \right]^2
\end{aligned}
$$

Similarly, we can have the distance ($S2_{t,i}$) between $x_{t+1,i}$ and $x^*_{,i}$ in $Adam$ with LGA:

$$
\begin{aligned}
S2_{t,i} &= (x_{t+1,i} - x^*_{,i})^2 \\
&= \left[(x_{t,i} - x^*_{,i}) - \left(\alpha_t \cdot \frac{1}{1-\beta_1^t} \cdot \frac{\beta_1 \cdot m_{t-1,i} + (1-\beta_1) \cdot g_{t,i}}{\sqrt{\hat{v_{t,i}}}} + \gamma_2 \cdot \frac{\Omega\left(f_{x_t}\right)}{\varphi\left(\nabla f_{x_t}\right)} \right) \right]^2
\end{aligned}
$$

Then the difference (ΔS) between the $Adam$ without LGA and the $Adam$ with LGA is following:

$$
\begin{aligned}
\Delta S_{t,i} &= S1_{t,i} - S2_{t,i} \\
&= \gamma_2 \cdot \frac{\Omega\left(f_{x_t}\right)}{\varphi\left(\nabla f_{x_t,i}\right)} \cdot \left[2 \cdot (x_{t,i} - x^*_{,i}) - 2 \cdot \alpha_t \cdot \frac{1}{1-\beta_1^t} \cdot \frac{\beta_1 \cdot m_{t-1,i} + (1-\beta_1) \cdot g_{t,i}}{\sqrt{\hat{v_{t,i}}}} - \gamma_2 \cdot \frac{\Omega\left(f_{x_t}\right)}{\varphi\left(\nabla f_{x_t,i}\right)} \right]
\end{aligned}
$$

For $t > 1$, $\left(\gamma_2 \cdot \frac{\Omega(f_{x_t})}{\varphi(\nabla f_{x_{t,i}})} \right)^2 \geq 0$, $(1-\beta_1) > 0$, $(1-\beta_1^t) > 0$, $\Omega(f_{x_t}) \geq 0$, $a_t = \frac{\alpha}{\sqrt{t}} > 0$:

$$
\begin{aligned}
\Delta S_{t,i} &= S1_{t,i} - S2_{t,i} \\
&= \gamma_2 \cdot \frac{\Omega\left(f_{x_t}\right)}{\varphi\left(\nabla f_{x_{t,i}}\right)} \cdot \left[2 \cdot (x_{t,i} - x^*_{,i}) - 2 \cdot \alpha_t \cdot \frac{1}{1-\beta_1^t} \cdot \frac{\beta_1 \cdot m_{t-1,i} + (1-\beta_1) \cdot g_{t,i}}{\sqrt{\hat{v_{t,i}}}} - \gamma_2 \cdot \frac{\Omega\left(f_{x_t}\right)}{\varphi\left(\nabla f_{x_{t,i}}\right)} \right] \\
&\leq 2\gamma_2 \frac{\Omega\left(f_{x_t}\right)}{\varphi\left(\nabla f_{x_{t,i}}\right)} (x_{t,i} - x^*_{,i}) - 2\alpha_t \frac{\Omega\left(f_{x_t}\right)}{\varphi\left(\nabla f_{x_{t,i}}\right)} \cdot \frac{1}{1-\beta_1^t} \cdot \frac{\beta_1 \cdot m_{t-1,i} + (1-\beta_1) \cdot g_{t,i}}{\sqrt{\hat{v_{t,i}}}} \\
&= 2\gamma_2 \frac{\Omega\left(f_{x_t}\right)}{\varphi\left(\nabla f_{x_{t,i}}\right)} (x_{t,i} - x^*_{,i}) - 2\alpha_t \frac{\Omega\left(f_{x_t}\right)}{\varphi\left(\nabla f_{x_{t,i}}\right)} \cdot \left(\frac{1}{1-\beta_1^t} \cdot \frac{\beta_1 \cdot m_{t-1,i}}{\sqrt{\hat{v_{t,i}}}} + \frac{1}{1-\beta_1^t} \cdot \frac{(1-\beta_1) \cdot g_{t,i}}{\sqrt{\hat{v_{t,i}}}} \right) \\
&\leq 2\gamma_2 \frac{\Omega\left(f_{x_t}\right)}{\varphi\left(\nabla f_{x_{t,i}}\right)} (x_{t,i} - x^*_{,i}) - 2\alpha_t \frac{\Omega\left(f_{x_t}\right)}{\varphi\left(\nabla f_{x_{t,i}}\right)} \cdot \left(\frac{1}{1-\beta_1^t} \cdot \frac{\beta_1 \cdot m_{t-1,i}}{\sqrt{\hat{v_{t,i}}}} \right)
\end{aligned}
$$

From the assumption, $\|x_i - x_j\|_2 \leq D$, we have:

$$
\left| 2\gamma_2 \frac{\Omega\left(f_{x_t}\right)}{\varphi\left(\nabla f_{x_{t,i}}\right)} (x_{t,i} - x^*_{,i}) \right| \leq 2\gamma_2 \left| \frac{\Omega\left(f_{x_t}\right)}{\varphi\left(\nabla f_{x_{t,i}}\right)} \right| \sqrt{D}
$$

We focus on $\frac{\beta_1 \cdot m_{t-1,i}}{\sqrt{\hat{v_{t,i}}}}$ and make the following transformations for $m_{t,i}$:

$$\begin{aligned}
|m_{t,i}| &= \sum_{s=1}^{t}(1-\beta_1)(\beta_1^{t-s}) \cdot |g_{s,i}| \\
&= \sum_{s=1}^{t} \frac{(1-\beta_1)(\beta_1^{t-s})}{\sqrt{(1-\beta_2)(\beta_2^{t-s})}} \cdot \sqrt{(1-\beta_2)(\beta_2^{t-s})}\,|g_{s,i}| \\
&= \frac{1-\beta_1}{\sqrt{1-\beta_2}} \sum_{s=1}^{t} \frac{\beta_1^{t-s}}{\sqrt{\beta_2^{t-s}}} \cdot \sqrt{(1-\beta_2)(\beta_2^{t-s})}\,|g_{s,i}| \\
&\leq \frac{1-\beta_1}{\sqrt{1-\beta_2}} \sum_{s=1}^{t} \sqrt{(1-\beta_2)(\beta_2^{t-s})}\,|g_{s,i}| \\
&= \frac{1-\beta_1}{\sqrt{1-\beta_2}} \sqrt{v_{t,i}} \\
&= \frac{1-\beta_1}{\sqrt{1-\beta_2}} \sqrt{(1-\beta_2^t)\hat{v_{t,i}}}
\end{aligned}$$

Disregarding the case where the gradient is mutated, $m_{t,i} \approx m_{t-1,i}$, then we have:

$$\begin{aligned}
\left| -2\alpha_t \frac{\Omega\left(f_{x_t}\right)}{\varphi\left(\nabla f_{x_{t,i}}\right)} \cdot \left(\frac{1}{1-\beta_1^t} \cdot \frac{\beta_1 \cdot m_{t-1,i}}{\sqrt{\hat{v_{t,i}}}} \right) \right| &\leq 2\alpha_t \left| \frac{\Omega\left(f_{x_t}\right)}{\varphi\left(\nabla f_{x_{t,i}}\right)} \cdot \left(\frac{1}{1-\beta_1^t} \cdot \frac{\frac{1-\beta_1}{\sqrt{1-\beta_2}} \sqrt{(1-\beta_2^t)\hat{v_{t,i}}}}{\sqrt{\hat{v_{t,i}}}} \right) \right| \\
&= 2\alpha_t \left| \frac{\Omega\left(f_{x_t}\right)}{\varphi\left(\nabla f_{x_{t,i}}\right)} \right| \cdot \left(\frac{1}{1-\beta_1^t} \cdot \frac{1-\beta_1}{\sqrt{1-\beta_2}} \sqrt{1-\beta_2^t} \right)
\end{aligned}$$

We can use absolute value inequality to get the upper bound for the $|\Delta S_{t,i}|$:

$$|\Delta S_{t,i}| \leq 2\gamma_2 \left| \frac{\Omega\left(f_{x_t}\right)}{\varphi\left(\nabla f_{x_{t,i}}\right)} \right| \sqrt{D} + 2\alpha_t \left| \frac{\Omega\left(f_{x_t}\right)}{\varphi\left(\nabla f_{x_{t,i}}\right)} \right| \cdot \left(\frac{1}{1-\beta_1^t} \cdot \frac{1-\beta_1}{\sqrt{1-\beta_2}} \sqrt{1-\beta_2^t} \right)$$

$\frac{\gamma_2}{|\varphi(g_t)|} \ll \frac{\alpha_t}{|\varphi(g_t)|}$ under the setting of LGA. Therefore, we can have the following regret bound:

$$|\Delta S_{t,i}| \leq 2\alpha_t \left| \frac{\Omega\left(f_{x_t}\right)}{\varphi\left(\nabla f_{x_{t,i}}\right)} \right| \cdot \left(\frac{1}{1-\beta_1^t} \cdot \frac{1-\beta_1}{\sqrt{1-\beta_2}} \sqrt{1-\beta_2^t} \right)$$

From the setting of LGA and Adam, $\left|\varphi\left(\nabla f_{x_{t,i}}\right)\right| \geq \xi$ and $\beta_1, \beta_2 \in [0,1)$, so $\left| \frac{1}{\varphi\left(\nabla f_{x_{t,i}}\right)} \right| \leq \frac{1}{\xi}$ and $\frac{1}{1-\beta_1^t} \cdot \frac{1-\beta_1}{\sqrt{1-\beta_2}} \sqrt{1-\beta_2^t} < \frac{1}{1-\beta_1} \frac{1}{\sqrt{1-\beta_2}}$. Then

$$\begin{aligned}
|\Delta S_{t,i}| &\leq 2\alpha_t \left| \frac{\Omega\left(f_{x_t}\right)}{\varphi\left(\nabla f_{x_{t,i}}\right)} \right| \cdot \frac{1}{1-\beta_1} \frac{1}{\sqrt{1-\beta_2}} \\
&\leq 2\frac{\alpha_t}{\xi} \frac{1}{1-\beta_1} \frac{1}{\sqrt{1-\beta_2}} \cdot \Omega(f_{x_t})
\end{aligned}$$

We derive the regret bound by summing across all the dimensions for $i \in 1, \cdots, d$ in the upper bound of ΔS_t, and ΔS have the following regret bound for $t \in 1, \cdots, T$:

$$\begin{aligned}
\Delta S &= \sum_{t=1}^{T} \sum_{i=1}^{d} \Delta S_{t,i} \\
&\leq \sum_{t=1}^{T} d \cdot 2\frac{\alpha_t}{\xi} \frac{1}{1-\beta_1} \frac{1}{\sqrt{1-\beta_2}} \cdot \Omega(f_{x_t}) \\
&= \frac{2d\alpha}{\xi(1-\beta_1)\sqrt{1-\beta_2}} \sum_{t=1}^{T} \frac{\Omega(f_{x_t})}{\sqrt{t}}
\end{aligned}$$

$\Omega(f_{x_t})$ will gradually converge to 0 during training, then $\lim_{T\to\infty} \frac{\Delta S}{T} = 0$. Essentially, ΔS_t can be regarded as a noise generated by LGA during training and is closely related to the loss status and gradient. When the loss has not been sufficiently reduced and the gradient is close to zero, then the LGA will make a big noise to gives the model more chances to escape and find new local optimum. If the loss has been sufficiently reduced or the gradient is too large, LGA will not generate noise to affect the optimization of the original algorithms.

A.3 Comparison Results on The Performance of Algorithms in Test Sets

The results of comparison of the best performance and the final performance of different optimization algorithms without and with LGA on MNIST, CIFAR-10 and CIFAR-100 are shown in Table 6 and Table 7, respectively, which look similar to the results shown in Table 1.

Table 6: Comparison of the best performance of different optimization algorithms without and with LGA on MNIST, CIFAR-10, and CIFAR-100. The classifier accuracy $(\%)$ of test sets is reported.

Algorithm	MNIST		CIFAR-10		CIFAR-100	
	w/o LGA	w/ LGA	w/o LGA	w/ LGA	w/o LGA	w/ LGA
SGD	95.47	**96.28**	75.75	**77.60**	40.32	**40.50**
Momentum	98.78	**98.80**	86.22	**86.70**	70.49	**71.12**
ASGD	96.02	**96.25**	75.25	**77.30**	**40.26**	40.23
RMSprop	98.95	**99.04**	88.00	**88.12**	45.07	**45.84**
Adam	99.01	**99.03**	88.28	**88.55**	47.91	**47.98**
Adamax	99.07	**99.12**	89.03	**89.19**	66.60	**66.64**
Adadelta	89.62	**91.93**	62.12	**76.19**	29.27	**29.64**

Table 7: Comparison of different optimization algorithms without and with LGA on MNIST, CIFAR-10, and CIFAR-100. The epochs are set to 20, 100, and 100 respectively during training. We report the classifier accuracy $(\%)$ of corresponding test sets.

Algorithm	MNIST		CIFAR-10		CIFAR-100	
	w/o LGA	w/ LGA	w/o LGA	w/ LGA	w/o LGA	w/ LGA
SGD	95.47	**96.28**	75.71	**77.55**	40.29	**40.50**
Momentum	98.78	**98.80**	85.97	**86.70**	70.49	**71.12**
ASGD	96.02	**96.25**	75.21	**77.15**	**40.26**	40.23
RMSprop	**98.91**	98.84	87.57	**87.59**	45.07	45.07
Adam	**99.01**	98.99	87.90	87.90	46.00	**47.98**
Adamax	99.03	99.03	88.72	**89.07**	66.52	**66.64**
Adadelta	89.62	**91.93**	62.12	**76.08**	29.03	**29.30**

A.4 Experimental Results on CIFAR-10 Under Different Learning Rates

We provide additional results using different learning rates on CIFAR-10 in Table 8. These data support the conclusion LGA can effectively provide some robustness against choosing a small learning rate and enhance the optimization capacity of parameters with a large learning rate.

A.5 Experimental Results on Image Segmentation Task

Image segmentation is the task of partitioning a natural image into multiple contiguous regions. We practice our method in the work of `https://github.com/kenandaoerdect/Image-segmentation-using-pytorch`, where the source code of the work and the dataset are available. The results are shown in Table 9.

Table 8: Comparison of different optimization algorithms without and with LGA on CIFAR-10 under different learning rates. We report the classifier accuracy $(\%)$ of corresponding test sets.

ALGORITHM	W/O LGA				W/ LGA			
	1E-1	1E-2	1E-3	1E-4	1E-1	1E-2	1E-3	1E-4
SGD	87.59	86.44	75.71	43.32	**88.21**	**86.69**	**77.55**	**59.40**
MOMENTUM	79.82	**89.28**	85.97	75.83	**84.20**	88.82	**86.70**	**77.59**
ASGD	86.92	85.11	75.21	46.15	**87.50**	**85.78**	**77.15**	**59.96**
RMSPROP	**18.79**	43.66	87.57	84.61	18.13	**69.61**	**87.59**	**85.79**
ADAM	18.33	**68.46**	87.90	85.9	**19.61**	68.17	87.90	**87.19**
ADAMAX	42.60	84.46	88.72	82.41	**52.24**	**84.80**	**89.07**	**82.61**
ADADELTA	87.46	81.29	68.12	36.29	**87.99**	**81.34**	**76.08**	**51.29**

Table 9: Results of different optimization algorithms on image segmentation task. The epochs of algorithms are set to 50 during training. We report the train loss as the metric of the performance of algorithms and the ratio of the epoch when the algorithms with LGA achieve the final performance of the original algorithms to the epoch of the original algorithms is reported ($\frac{w/}{w/o} \in (0, 1]$). The lower loss and the lower ratio, the more effective LGA is.

Algorithm	SGD	Momentum	ASGD	RMSprop	Adam	Adamax	Adadelta
w/o LGA	0.218	0.181	0.058	0.220	0.373	0.317	0.261
w/ LGA	**0.102**	**0.089**	**0.055**	**0.172**	0.373	0.317	**0.103**
Ratio	0.23	0.17	0.23	0.23	0.16	0.23	0.43

A.6 EXPERIMENTAL RESULTS ON VIDEO SENTENCE GROUNDING TASK

A newly proposed cross-modality task, namely video sentence grounding, is chosen as a additional benchmark to further evaluate the efficiency of our method,. Video sentence grounding is proposed to localize the target moment of a given video which is specified by a natural language query. We evaluate our method on a recent work (Zhang et al., 2020) and a widely used dataset TACoS (Rohrbach et al., 2012) with a split (Gao et al., 2017) of 10146, 4589 and 4083 moment-query pairs for training, validation and testing respectively.We report the experimental results of the loss in Table 10.

Table 10: Comparison of different optimization algorithms on video sentence grounding task. The epochs of algorithms are set to 50 during training. We report the train loss as the metric of the performance of algorithms and the ratio of the epoch when the algorithms with LGA achieve the final performance of the original algorithms to the epoch of the original algorithms is reported ($\frac{w/}{w/o} \in (0, 1]$).

Algorithm	SGD	Momentum	ASGD	RMSprop	Adam	Adadelta
w/o LGA	15.78	12.56	15.60	5.73	5.75	15.60
w/ LGA	**14.15**	**12.09**	**13.98**	**5.65**	5.75	**14.30**
Ratio	0.23	0.47	0.37	0.93	0.96	0.23

A.7 EXPERIMENTAL RESULTS ON AUTOEVAL TASK

AutoEval is a task that predicts model accuracy based on only unlabeled test sets. In the work of Deng et al. (2021), they train semantic classification and rotation prediction in a multi-task way and then study the statistical correlation between semantic classification and rotation prediction under varying environments. We experiment with our method on the above work on MNIST and report the loss function of semantic classification, the loss function of rotation prediction, the accuracy of the semantic classification, and the accuracy of rotation on the training set in Table 13.

Table 11: Comparison of different optimization algorithms on AutoEval task. The epochs of algorithms are set to 50 during training based on LeNet-5. The loss function of semantic classification ($loss_{cls}$) and the loss function of rotation prediction ($loss_{rot}$) are reported. The lower $loss_{cls}$ and $loss_{rot}$, the more effective LGA is.

Metric	Algorithm	SGD	Momentum	ASGD	RMSprop	Adam	Adamax	Adadelta
$loss_{cls}$	w/o LGA	0.027	0.002	0.030	0.148	0.020	0.017	52.314
	w/ LGA	**0.019**	0.002	**0.019**	**0.136**	**0.010**	**0.013**	**37.760**
$loss_{rot}$	w/o LGA	0.019	0.006	0.022	0.099	**0.011**	0.014	1.624
	w/ LGA	**0.017**	**0.005**	**0.017**	**0.046**	0.013	**0.012**	**1.350**

A.8 Experimental Results on AutoEncoder

Training autoencoders on benchmark datasets is considered as a standard problem for evaluating optimization methods on neural networks. It is especially for evaluating the escape from local optimal solutions. We practice our method in MNIST and CIFAR-10 following the work of `https://blog.csdn.net/nanke_4869/article/details/113497459` and the work of `https://github.com/chenjie/PyTorch-CIFAR-10-autoencoder`. The results are shown in Table 12.

Table 12: Comparison of different optimization algorithms. The final loss function is reported.

Dataset	Algorithm	SGD	Momentum	ASGD	RMSprop	Adam	Adamax	Adadelta
Mnist	w/o LGA	0.182	**0.079**	0.177	0.077	0.072	0.071	0.180
	w/ LGA	**0.166**	0.080	**0.165**	**0.076**	0.072	0.071	**0.167**
CIFAR-10	w/o LGA	0.691	0.582	0.691	0.566	0.562	0.561	0.691
	w/ LGA	**0.587**	**0.564**	**0.590**	0.566	0.562	**0.560**	**0.591**

A.9 Effects of the estimation of the lower bound

To study the proposed lower bound estimation, we compare our method to two common estimation operations. First, directly averaging all the loss status; second, summing up the loss status with momentum strategy, and the momentum is set to 0.9. Table 13 presents the experimental results of LGA with different estimation method on MNIST and CIFAR-10. We see that, the proposed estimation method outperforms the two variants.

Table 13: LGA with different estimation methods. The final loss function is reported.

Dataset	Algorithm	SGD	Momentum	ASGD	RMSprop	Adam	Adadelta
Mnist	LGA(average)	0.0318	0.0050	0.0313	**0.1053**	0.0695	0.0839
	LGA(momentum)	0.0318	0.0050	0.0312	0.1300	**0.0694**	0.0836
	LGA	**0.0316**	**0.0049**	**0.0310**	0.1324	0.0700	**0.0835**
CIFAR-10	LGA(average)	1.4087	0.5079	1.4015	0.1823	0.0856	1.6784
	LGA(momentum)	**0.4878**	0.0885	0.4737	0.1844	**0.1847**	0.9670
	LGA	**0.4878**	**0.0878**	**0.4705**	**0.1817**	0.1889	**0.9633**