Hindawi
Security and Communication Networks
Volume 2023, Article ID 9785972, 1 page
https://doi.org/10.1155/2023/9785972

Retraction

Retracted: Gradient Descent Optimization in Deep Learning Model Training Based on Multistage and Method Combination Strategy

Security and Communication Networks

Received 26 December 2023; Accepted 26 December 2023; Published 29 December 2023

This article has been retracted by Hindawi, as publisher, following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of systematic manipulation of the publication and peer-review process. We cannot, therefore, vouch for the reliability or integrity of this article.

Please note that this notice is intended solely to alert readers that the peer-review process of this article has been compromised.

Wiley and Hindawi regret that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.

The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

[1] C. Zhang, M. Yao, W. Chen, S. Zhang, D. Chen, and Y. Wu, "Gradient Descent Optimization in Deep Learning Model Training Based on Multistage and Method Combination Strategy," *Security and Communication Networks*, vol. 2021, Article ID 9956773, 15 pages, 2021.

Hindawi
Security and Communication Networks
Volume 2021, Article ID 9956773, 15 pages
https://doi.org/10.1155/2021/9956773

Research Article

Gradient Descent Optimization in Deep Learning Model Training Based on Multistage and Method Combination Strategy

Chuanlei Zhang,[1] Minda Yao,[1] Wei Chen,[2,3] Shanwen Zhang,[4] Dufeng Chen,[5] and Yuliang Wu[6]

[1]*College of Artificial Intelligence, Tianjin University of Science and Technology, Tianjin 300457, China*
[2]*School of Mechanical Electronic and Information Engineering, China University of Mining and Technology (Beijing), Beijing 100083, China*
[3]*School of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221116, China*
[4]*College of Information Engineering, Xijing University, Xi'an 710123, China*
[5]*Beijing Geotechnical and Investigation Engineering Insititute, Beijing 100080, China*
[6]*Department of Emergency Management, Sichuan Staff University of Science and Technology, Chengdu 610101, China*

Correspondence should be addressed to Wei Chen; chenwdavior@163.com

Received 11 March 2021; Accepted 2 July 2021; Published 23 July 2021

Academic Editor: Chi-Hua Chen

Gradient descent is the core and foundation of neural networks, and gradient descent optimization heuristics have greatly accelerated progress in deep learning. Although these methods are simple and effective, how they work remains unknown. Gradient descent optimization in deep learning has become a hot research topic. Some research efforts have tried to combine multiple methods to assist network training, but these methods seem to be more empirical, without theoretical guides. In this paper, a framework is proposed to illustrate the principle of combining different gradient descent optimization methods by analyzing several adaptive methods and other learning rate methods. Furthermore, inspired by the principle of warmup, CLR, and SGDR, the concept of multistage is introduced into the field of gradient descent optimization, and a gradient descent optimization strategy in deep learning model training based on multistage and method combination strategy is presented. The effectiveness of the proposed strategy is verified on the massive deep learning network training experiments.

1. Introduction

Today, thanks to the contribution of deep learning and deep neural networks, artificial intelligence (AI) is a thriving field with many practical applications and active research topics. For a deep learning system, modern neural networks play the core role. These network models show superperformance in many professional fields like language translation [1], visual recognition [2], and decision-making [3]. Gradient descent is the core and foundation of a neural network. Just like the engine of a car, a deep neural network (DNN) is also composed of many parts and the core is gradient descent optimization.

In the fields of gradient descent optimization, quite a few methods have been proposed to improve the training performance of neural networks. Learning rate decay methods like cosine decay and adaptive learning rate methods like RMSprop [4] and Adam [5] are famous in the practical neural network training process. The methods based on gradient estimation, like Momentum [6] and Nesterov Accelerated Gradient (NAG) [7], are also able to facilitate the neural network model training. Although these methods work well, usually they are used alone for neural network model training. For most cases, few scholars are willing to employ multiple methods simultaneously for their model training. Loshchilov and Hutter [8] found that using a learning rate multiplier method can substantially improve Adam performance, and they advocate not to overlook the combining use of learning rate methods for Adam. These methods can achieve certain improvement effects.

Nevertheless, they are more like intuitive operations, and there is no theory to guide how to combine different learning rate methods for neural network model training.

Based on the above analysis, a gradient descent optimization strategy in deep learning model training based on multistage and method combination strategy is proposed in this paper. In summary, the main contributions in this paper are threefold:

(i) A framework to illustrate the principle of combining different gradient descent optimization methods by analyzing several adaptive methods and other learning rate methods was proposed.

(ii) Inspired by the principle of warmup, CLR, and SGDR, the concept of multistage was introduced into the field of gradient descent optimization, and a gradient descent optimization strategy in deep learning model training based on multistage and method combination was proposed.

(iii) Extensive experiments have been conducted and the results validated the effectiveness of the proposed gradient descent optimization strategy in deep learning model training based on multistage and method combination strategy.

2. Related Works

Gradient descent optimization methods have been used extensively with numerous applications for decades [9]. There are also a great number of deep learning frameworks giving practical optimization suggestions. In this section, a brief review is shown on the methods that are the most related to the research and the exploration of gradient descent optimization.

Learning rate is "the single most important hyperparameter" [10] in training neural networks. A fixed value of the learning rate does not work well. It is impossible to achieve the best training performance with a too large or too small learning rate, so different learning rate schedules [11] try to adjust the learning rate during training through a predefined schedule, which usually take a function to make learning rate decay or change parameter when epochs fall below a threshold [9]. However, these methods, based on schedules or thresholds, have to be defined in advance and they are unable to adapt to the characteristics of the training dataset [12]. Different from the learning rate decay method, the learning rate warmup method is to raise the learning rate at the beginning of training, and the most successful warmup method is gradual warmup [13]. Moreover, both CLR (Cyclical Learning Rates) [14] and SGDR (Stochastic Gradient Descent with Warm Restarts) [15] have found that occasionally increasing the learning rate during training may cause poor model performance in the short term, but the final training results are better than traditional methods.

Adaptive learning rate methods are a series of learning rate optimizing methods, which do not need to set a complex learning rate scheme like learn rate decay. These methods just set an initial parameter and they can adjust the rate well based on the specific situations. Adagrad [16] is the first widely used adaptive method. RMSprop and Adadelta [17] have both been developed independently around the same time stemming from the need to resolve Adagrad's radically diminishing learning rates problem [9]. Adaptive Moment Estimation [5], also known as Adam, takes advantage of Momentum [6] and RMSprop.

Momentum is a classic method to achieve the convergence quickly and reduce the oscillation of training performance. SGD updates each time in the direction of the current position along the negative gradient, regardless of the value of the previous directional gradient. The Momentum, by introducing a new variable, accumulates the previous gradient (obtained by exponential decay average) to accelerate the learning process. Based on this, Nesterov Accelerated Gradient (NAG) [7] is a way that adds a kind of prescience ability to Momentum term by a revision factor.

The concept of multiple stages is used extensively in daily lives and many engineering fields. Different from the one-stage methods that only employ just one single way to finish tasks, the multistage ones take specific methods at different stages, respectively, which can produce better final performance than one-stage methods. The idea of multiple stages has been introduced into deep learning in many aspects, and some of them have gained many achievements. In the field of object detection, Zeng et al. [18] took a multistage architecture to handle the complex distributed samples situation on pedestrian detection. Sermanet et al. [19] introduced an unsupervised feature learning model with multistage feature to help detect pedestrians, and Ouyang et al. [20] proposed a multistage and deformable deep convolutional neural network called DeepID-Net to assist the work of object detection. Though the object detection methods based on deep learning are mainly divided into multistage categories and one-stage categories, most objection detection methods belong to multistage methods. Moreover, a multistage deep learning framework was proposed by Yan et al. [21] to solve the problems of body part recognition. Besides employing multistage methods to improve training performance, Yuan [22] showed a multistage analysis method on real-time malware detection.

With the development of deep learning, it is increasingly difficult for us to train gradually diversified and complicated neural networks with single learning rate adjustment methods. In fact, in addition to using an optimization method alone, multiple methods can also be combined to improve the training performance of neural networks. Although some engineers have tried to use multiple methods together, these operations are intuitive. The advice that a learning rate multiplication schedule on Adam can be used to improve performance is claimed by Loshchilov and Hutter [8]. Szegedy et al. [23] reached the best performance of models using RMSprop with an exponential rate to decay learning rate. Different from the conventional suggestion that Adam does not need to adjust, Wilson et al. [24] found that the initial learning rate and decay method for Adam can be adjusted to achieve a significant improvement over its default settings in all the cases. Meanwhile, TensorFlow, Keras, and PyTorch also allow developers to set the decay factors to control built-in optimizers like SGD, Adam, and

RMSprop. These methods have certain effects; however, there is no reasonable interpretation for them, and there is no guideline on combining different optimization methods.

3. Analysis of Gradient Descent Optimization Methods

3.1. Affecting Factors on Gradient Descent. Gradient descent is the core and foundation of the BP neural network. It can be seen from the update formula of the parameters and the changes before and after the update at each iteration:

$$\begin{aligned}\theta_t &= \theta_{t-1} - \alpha g_t,\\ \Delta\theta_t &= -\alpha g_t,\end{aligned} \tag{1}$$

where θ_t is the parameters of t iteration, α is the learning rate, and g_t is the gradient estimation value of t iteration. We can detect that the main factors affecting the gradient descent method are learning rate α and gradient estimation g_t. At the same time, it can be found that the learning rate and the estimated value of gradient are two unrelated factors. Therefore, optimization measures on both factors can improve the gradient descent method, respectively.

3.2. Categories of Gradient Descent Optimization Methods. From the analysis of gradient descent above, the core factors that can have an impact on performance are learning rate and gradient estimation. There are two common categories of gradient descent optimization methods, which are learning rate adjustment methods which improve the stability of models, and gradient estimation optimization methods which correct the gradient of current iteration to improve training speed.

Obviously, the adaptive learning rate methods are based on gradient instead of learning rate. And to illustrate the two categories in a more exact and specific way, several common optimization methods for corresponding categories are listed in Table 1.

4. Framework of Combining Multiple Gradient Descent Optimization Methods

4.1. Analysis of the Adaptive Methods and Learning Rate Adjustment Methods. Just like Loshchilov and Hutter [8] suggested in their paper, schedules can substantially improve Adam's performance. Although the names of these methods are called adaptive learning rate methods, in fact, these methods are kinds of optimization of gradient estimation. And the analysis of these methods is as follows.

4.1.1. Adagrad. For Adagrad, compute gradient and accumulate squared gradient of each parameter at the iteration:

$$G_t = \sum_{\tau=1}^{t} g_\tau \odot g_\tau, \tag{2}$$

where $\odot$ is an elementwise multiplication and $g_\tau \in \mathbb{R}^{|\theta|}$ is the gradient of the current parameter at the τ iteration.

The update value of parameters in Adagrad is

$$\Delta\theta_t = -\frac{\alpha}{\sqrt{G_t + \varepsilon}} \odot g_\tau, \tag{3}$$

where α is the learning rate and ε is a smoothing term that avoids division by zero. As the learning rate is predefined before training, equation (4) also can be seen as

$$\Delta\theta_t = -\alpha\left(\frac{1}{\sqrt{G_t + \varepsilon}} \odot g_\tau\right). \tag{4}$$

From equation (3), we know that G_t is the calculation of the previous gradient, so the expressions enclosed in parentheses can be seen as one kind of gradient revision. It is defined as g_t':

$$g_t' = \frac{1}{\sqrt{G_t + \varepsilon}} \odot g_\tau. \tag{5}$$

Thus, the update value of Adagrad can be described as follows:

$$\Delta\theta_t = -\alpha g_t', \tag{6}$$

which is similar to the update process of classic gradient descent. So Adagrad can be seen as an optimization method based on the gradient.

4.1.2. RMSprop. As RMSprop is the improvement version of Adagrad, the update process of RMSprop is similar to Adagrad. For RMSprop, we can calculate an exponentially decaying average of squared gradients first.

$$\begin{aligned}G_t &= \beta G_{t-1} + (1-\beta) g_t \odot g_t\\ &= (1-\beta)\sum_{\tau=1}^{t} \beta^{t-\tau} g_\tau \odot g_\tau,\end{aligned} \tag{7}$$

where β is the decay rate which is usually suggested to be set to 0.9. And the update value of parameters in RMSprop is the same as Adagrad:

$$\Delta\theta_t = -\frac{\alpha}{\sqrt{G_t + \varepsilon}} \odot g_\tau. \tag{8}$$

Similarly, the simplification idea of the Adagrad method can also be applied here. g_t' is described as follows:

$$g_t' = \frac{1}{\sqrt{G_t + \varepsilon}} \odot g_\tau, \tag{9}$$

and the update value of RMSprop is defined as

$$\Delta\theta_t = -\alpha g_t'. \tag{10}$$

Therefore, RMSprop is an optimization method based on gradients actually. On the basis of the analysis, the learning rate optimization method can be used to improve the training performance.

4.1.3. Adam. Adam is another extensively used method that adjusts the learning rate adaptively for every parameter. Adam is a combination of different gradient optimization

Table 1: The categories of common optimization methods.

	Learning rate adjustment method	Gradient estimation optimization method
Momentum		✓
Nesterov Momentum		✓
Adagrad		✓
Adadelta		✓
RMSprop		✓
Adam		✓
ReduceLROnPlateau	✓	
Piecewise constant	✓	
Exponential decay	✓	
Natural exponential decay	✓	
Polynomial decay	✓	
Cosine decay	✓	
Gradual warmup	✓	

methods. Not only is an exponentially decaying average of past squared gradients computed, like Adadelta and RMSprop, but also Adam takes an exponentially decaying average of past gradients, which is similar to Momentum.

$$M_t = \beta_1 M_{t-1} + (1-\beta_1) g_t, \tag{11}$$

$$G_t = \beta_2 G_{t-1} + (1-\beta_2) g_t \odot g_t, \tag{12}$$

where β_1 and β_2 are the decay rates which are suggested to follow the default values. M_t and G_t are defined to estimate the mean of past gradients (the first moment) and the uncentered variance of past gradients (the second moment), respectively.

As the decaying rates usually cause some biases problem, it is necessary to do the bias-correction work.

$$\begin{aligned} \widehat{M}_t &= \frac{M_t}{1-\beta_1^t}, \\ \widehat{G}_t &= \frac{G_t}{1-\beta_2^t}. \end{aligned} \tag{13}$$

Therefore, the update value of Adam is defined as

$$\Delta\theta_t = -\frac{\alpha}{\sqrt{\widehat{G}_t + \varepsilon}} \widehat{M}_t. \tag{14}$$

The gradient part of $\Delta\theta_t$ also can be defined as

$$g_t' = \frac{1}{\sqrt{\widehat{G}_t + \varepsilon}} \widehat{M}_t, \tag{15}$$

$$\begin{aligned} \Delta\theta_t &= -\alpha\left(\frac{1}{\sqrt{\widehat{G}_t + \varepsilon}} \widehat{M}_t\right) \\ &= -\alpha g_t'. \end{aligned} \tag{16}$$

From equation (16), it can be found that all the operations are based on past gradients of the current parameters, which have no relationship to learning rate. Hence, Adam could have a better performance with the assistance of learning rate methods.

4.2. Combination Framework for Different Optimization Methods. From the analysis of adaptive methods and the categorization of learning rate adjustment methods, it can be found that these adaptive methods are not pure learning rate methods and they can be seen as optimization of gradient estimation (for the whole training process, the value of learning rate is still the initial learning rate and not changed by adaptive methods. In practice, adaptive methods take gradient as an extraregulator on gradient descent, adjusting the actual updates of parameters based on the changes of current iteration's gradient, to attach the called adaptive performance). So, in this section, we will justify the combination of different learning rates by explaining the process of calculating the current learning rate in neural networks training.

According to the operation rules of polynomials, for a set of equations, multiple terms can be combined into one, and a polynomial can also be thought of as a combination of terms. Hence, the adjustment operations of the learning rate can be regarded as a combination outcome of multiple equations. That is, the value of the learning rate at a certain moment can be obtained by combining multiple learning rate adjustment methods, which is shown in Figure 1.

As shown in Figure 1, this framework for combining multiple gradient descent optimization methods can be more intuitively understood. For multiple learning rate adjustment methods which are set simultaneously, we can obtain the actual learning rate function before training, and then the change curve of the learning rate during the entire training process can be obtained.

For different gradient estimation-based optimization methods, they are based on gradient but have various method logics. As we know, in the training process, each

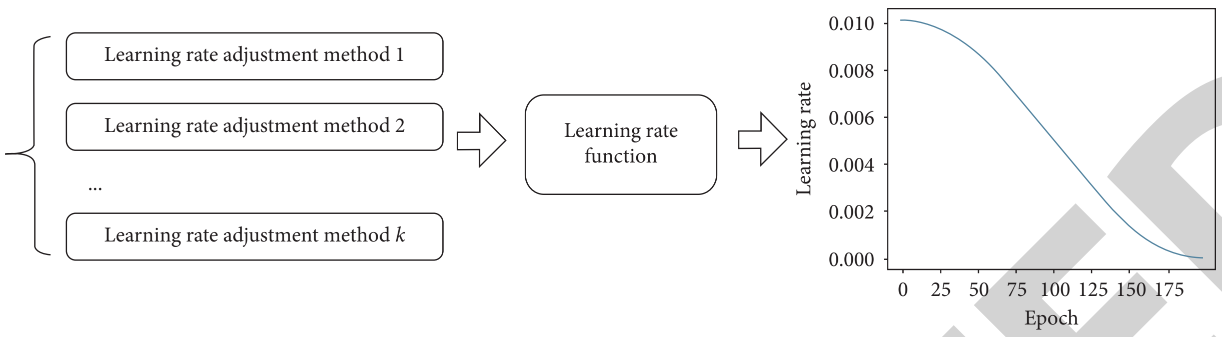

Learning rate adjustment method 1
Learning rate adjustment method 2
...
Learning rate adjustment method k
Learning rate function
Learning rate
0.010
0.008
0.006
0.004
0.002
0.000
0 25 50 75 100 125 150 175
Epoch

Figure 1: Framework for combining multiple gradient descent optimization methods.

parameter has its own gradient value at every moment, and a parameter only has one gradient value at the same time. Therefore, two and more different optimization methods (based on the same gradient) can be combined without conflicts, and the widely used Adam which has been analyzed above is the typical example which can be seen as the combination of RMSprop and Momentum.

5. Multistage Gradient Descent Optimization Strategy

In this section, a gradient descent optimization strategy in deep learning model training based on multistage and method combination strategy is proposed.

5.1. Multistage Gradient Descent Optimization Strategy. Just like the other applications of multiple stages in deep learning, we propose a new strategy in the field of gradient descent optimization which can improve the performance of models with the idea of multiple stages. And this strategy is inspired by three methods: warmup, CLR, and SGDR. Warmup takes a pair of opposite methods for the beginning and the following timepieces of training. Both CLR and SGDR divide the whole training process into several timepieces and have an adjustment on the learning rate value of each timepiece.

This kind of strategy is to divide the whole training process into several timepieces and set separate methods for different timepieces. Different from the fixed cyclical timepieces or the predefined changes on cyclical timepieces, this strategy is more flexible in training, and one optimization method can be used for a long timepiece. One long timepiece can be regarded as a continuous timepiece that is composed of several short timepieces. Based on this, one optimization method can be used for this long timepiece to achieve the performance of training in a long timepiece. Meanwhile, as each timepiece has its specific method, not only the optimization methods can be arranged before training, but the method of the next timepiece also can be changed following the current timepiece based on the training situation in the current timepiece. To illustrate this strategy clearly, the features and differences of the proposed multiple stages strategy and CLR/SGDR are presented in Table 2.

The training procedure can be separated into three steps. In the first step, the training process is divided into several timepieces equally. And in the second step, different or the same optimization methods are set in separate timepieces. In the third step, if the arrangement of optimization methods is inappropriate for the next timepieces, the arrangement can be changed. The third step is optional. Figure 2 shows the process vividly, and the training time can be divided into several timepieces. Then, learning rate methods can be arranged to separate timepieces based on the training situation.

5.2. Combinations of Different Gradient Descent Optimization Methods. We have analyzed the combination cases of learning rate adjustment methods and gradient estimation-based optimization methods. In this part, we will try to figure out how to combine different gradient descent optimization methods.

For each of these two combination classes, various optimization methods of the current class can be utilized together to improve the training performance, and this can be called inner combination strategy (the combined methods belong to the same class). Meanwhile, we have to face some more complicated situations where the inner combination strategies do not work well and need a new strategy that is equipped with two or more classes of methods to handle these complex situations powerfully, which is called mixed combination strategy (the combined combination of methods belong different classes). The effectiveness of the adaptive learning rate adjustment methods and other learning rate methods shows that the two classes optimization methods can work simultaneously to accelerate deep learning networks training. The combination idea is demonstrated in Figure 3. Figure 3(a) presents the inner combination cases of learning rate adjustment methods or gradient estimation-based methods and Figure 3(b) is the representation of mixed cases, where learning rate adjustment methods and gradient estimation-based methods can work together.

5.3. Multistage Strategy Combining Multiple Gradient Descent Optimization Methods. Though the function of these methods can be combined as a whole, in practice, the neural network executes all methods one by one. And such kind of

Table 2: The differences between multistage strategy and CLR/SGDR.

Features	CLR and SGDR	Proposed multistage strategy
The length of timepieces	Fixed or predefined	Changeable
Methods for each timepiece	Only predefined	Predefined but changeable

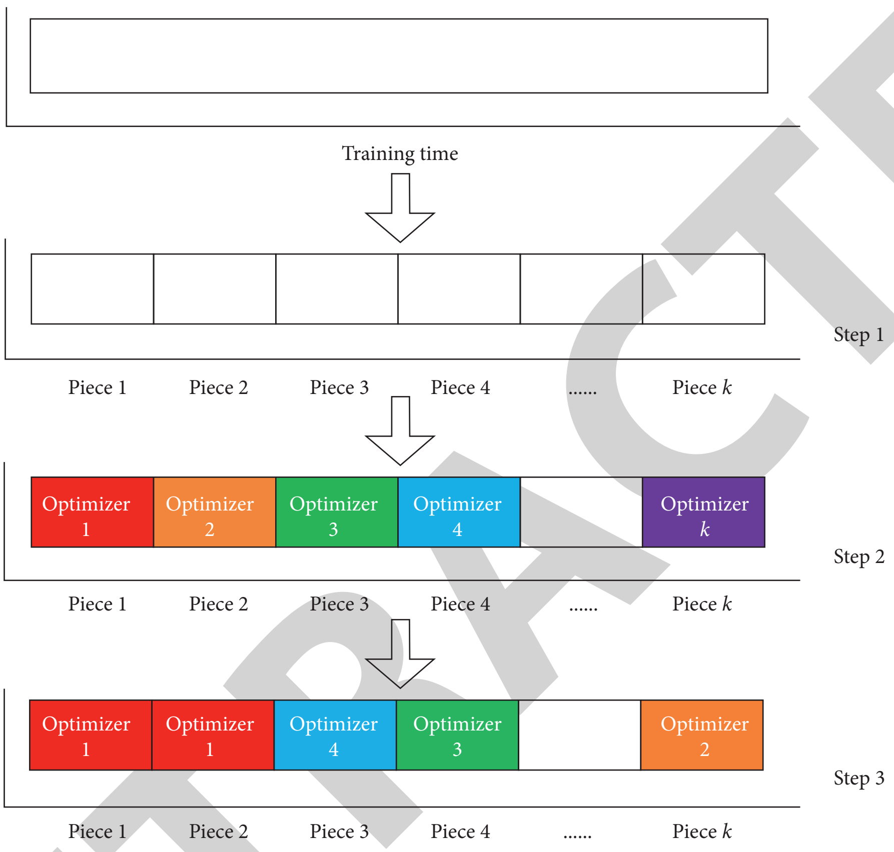


Figure 2: The training procedure of multistage strategy.

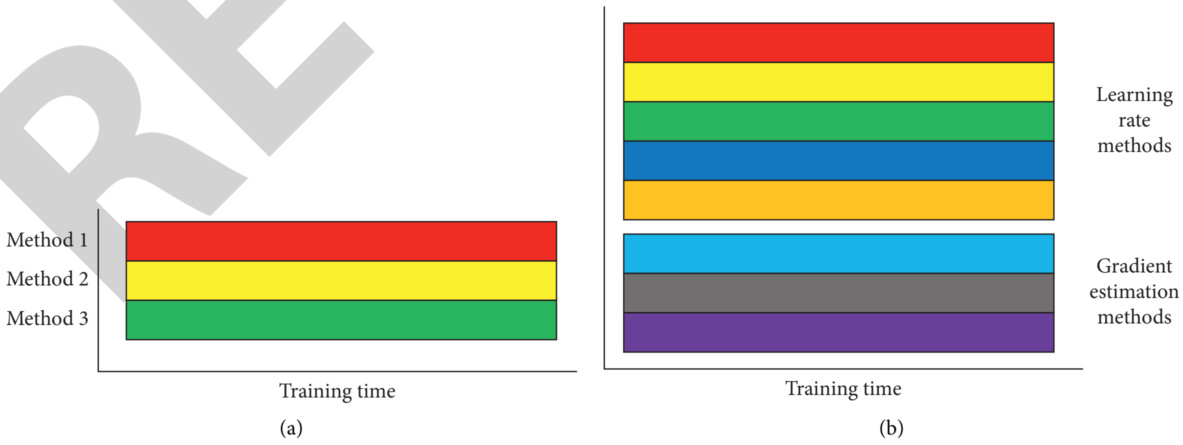


Figure 3: Combination strategy. Each method has its own color to mark itself. (a) Inner combination case; (b) mixed combination case.

implementation inevitably will lead to some conflicting operations. For instance, the conflicting operations happen when learning rate decay and warmup are triggered in the same iteration, and this, which takes a rise operation and a decay operation on learning rate simultaneously, would waste lots of time and computing resources. Actually, the result can be obtained at a relatively low cost by calculating the final value of the learning rate before operations on it at the current iteration, and this can work well because we just execute the significant part which removed the conflicting part from the whole. To avoid conflicting situations, in the proposed multistage learning rate adjustment strategy, these methods which have conflicting operations can be arranged in separated timepieces.

In order to explain this process more clearly, the process is presented in Figure 4. The left part shows the two strategies introduced above, and the right part demonstrates the advantages of the combination of the two strategies. Obviously, this strategy is a combination of the proposed combination framework and combination strategy, using their greatest strengths to further optimize the gradient descent method.

Many methods belong to learning rate adjustment methods and the typical conflicting situations are as follows. For the combination of learning rate decay and warmup, the actual operations on the value of learning rate decrease and raise which is the typical opposite operations, and the effect of this combination might be worse than separate use of them. However, when they are allocated into different timepieces, it is possible to achieve the amazing performance of training which is more accurate and stable.

The conflicting situation of gradient estimation is likely to happen with learning rate methods. Thus, there are scenarios in which some methods with similar principles, like RMSprop and Adam, which are the original method and improved one, are employed together. It is better to allocate these methods to different timepieces and they are used separately. This is to say, for some timepieces, only one method might be a better choice. If the two methods are used at the same time, unpredictable results could be produced. Therefore, arranging these conflicting methods separately is a wise way to avoid some unexpected results.

The details of combination methods also can be presented in Table 3, where "*Y*" represents the workable timepieces, "*N*" represents the nonworkable timepieces, LR method means learning rate, and GE method means gradient estimation-based method. Thus, a simple schedule table can show the methods in the whole training process.

6. Experiment

In this section, we evaluated the performance of the proposed strategies by training ResNet and LSTM (Long-Short Term Memory) [25] on the MNIST [26], CIFAR-10 [27], and IMDB [28]. CNN (convolutional neural network) and RNN (recurrent neural network) are the famous models in deep learning, and the ResNet and LSTM are the typical and extensively used CNN and RNN models. The MNIST dataset is a handwritten digit dataset with 10 types of digital labels which contain 60,000 examples for training and 10,000 examples for testing. These handwritten digits have been standardized in size and are located in the center of the image, which is a fixed size (28×28 pixels). CIFAR-10 is a classic CV (Computer Vision) dataset that consisted of 60,000 pieces of 328×32 color images, and the dataset can be split into 10 classes and 100 classes, respectively, with 50,000 training images and 10,000 test images. Moreover, the IMDB dataset is another typical dataset for analyzing the performance of RNN models like LSTM. This dataset contains 50,000 comments from the Internet Movie Database, all of which have obvious emotional bias. Among them, 25,000 comments are used as the training set and 25,000 comments are used as the test set, and each part accounts for 50% positive and negative comments. The experiments are conducted with hardware configuration ThinkPad T480s with Intel Core i5-8250 CPU, 24 GB ddr4 2133 MHz memory, and a GPU of NVIDIA MX150.

6.1. Conflict of Learning Rate Adjustment Methods. The famous warmup also consists of the conflicting operations that raise and decrease the learning rate at the beginning and the rest of the timepieces, but it can achieve a good training performance. Thus, the obvious consequence is that better performance of training can be obtained by arranging conflicting methods in separate timepieces.

To explore the influence of conflicting situations when combining different learning rate adjustment methods, for example, in every iteration, learning rate increases 1% and then reduces 1%. In case of the disturbances on learning rate from gradient methods, we take SGD on the models of LSTM and ResNet-20 to evaluate the performance.

The result is shown in Figure 5, which shows the accuracy and loss during training. It can be found straightforward that the curves of rising and decay are similar to the only decay one. And the performance of conflict cases is a little poor.

After the analysis of training performance, we involve execution time as another performance indicator. Because the optimization methods are executed one by one, there is no doubt that too many operations on learning rate value will slow down the executing speed. The documents of the executing time about conflicting situations and non-ones on ResNet20 with common optimizers are shown in Tables 4 and 5 by units of ms/batch and s/epoch, respectively. The tendency can be vividly found from the two tables that the complexity and amount of operations on the learning rate will prolong the executing time of optimizers. This analysis indicates that users should avoid the low-level conflicting operations and time costing complex operations.

6.2. Results of Multistage Strategy. There are many indicators to justify this strategy and evaluate the performance. We choose the most intuitive way to demonstrate it, taking different methods to compare the performance of different adjustment strategies and executing 10 epochs of training for each method. The performances of these are shown in Tables 6 and 7. For each of the two tables, the outcome of the first line which uses SGD for the whole training is worse than the performance of the next two lines which take another optimization method on the second timepiece of training. Therefore, it can be vividly concluded that the performances of a multistage strategy are better than the ones that only take a single optimization method during the whole training.

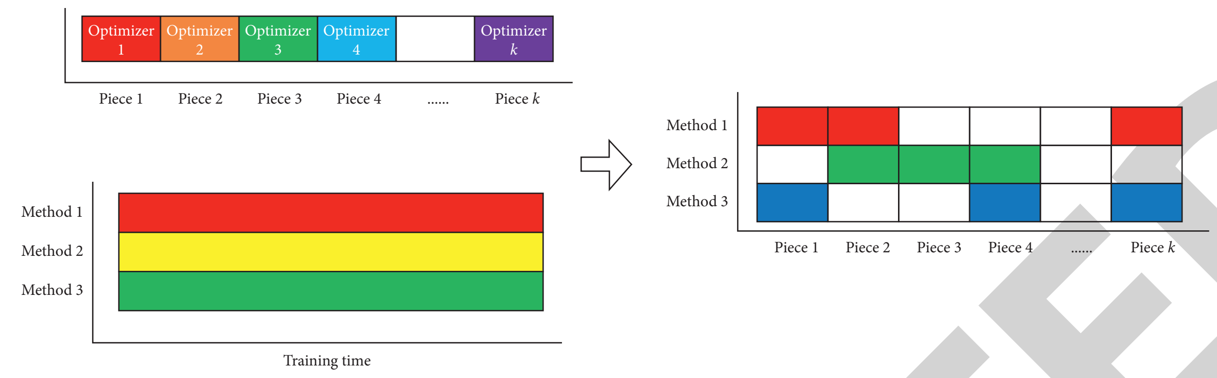


FIGURE 4: Multistage strategy combining multiple gradient descent optimization methods. Each method is present as special color, and the timepieces are filled with the corresponding method color.

TABLE 3: Schedule table for optimization methods used in the training process.

Optimization		Pieces	
Methods	Piece 1	Piece 2	Piece 3
LR method 1	Y	N	N
LR method 2	Y	Y	N
LR method 3	N	Y	Y
LR method 4	Y	N	Y
LR method 5	Y	Y	Y
GE method 1	Y	N	N
GE method 2	N	Y	N
GE method 3	N	N	Y

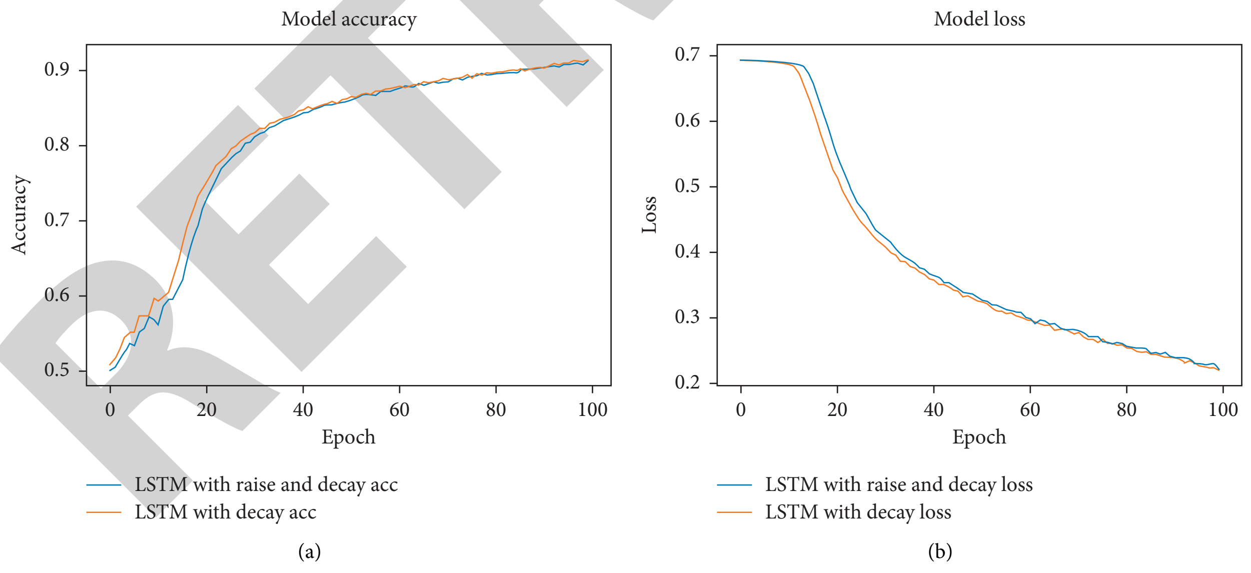
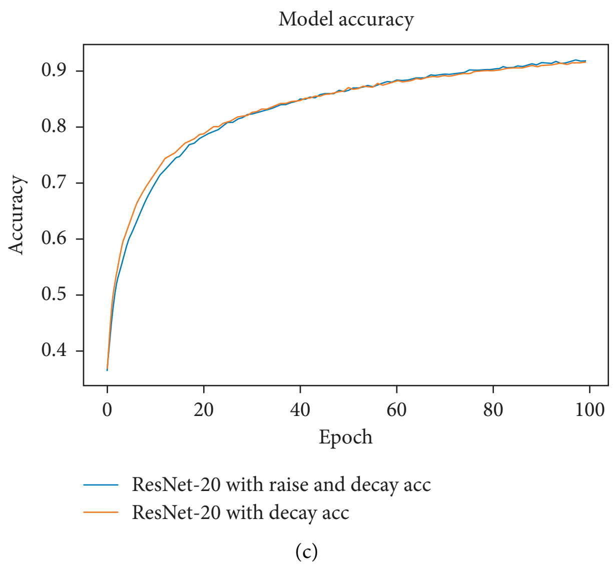


FIGURE 5: Continued.



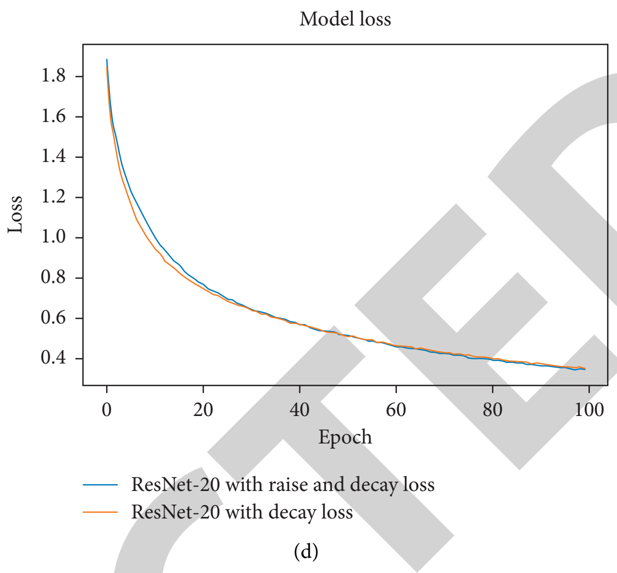


Figure 5: Comparison between a conflict operation and a single operation of learning rate on CIFAR10. (a, b) The comparison result of LSTM, and (c, d) the comparison result of ResNet-20.

Table 4: Executing time of conflicting situations on learning rate.

Methods	Decay	1R1D	2R2D	3R3D	5R5D
RMSprop	64	69	70	71	72
Adam	70	78	79	81	83
NAdam [29]	79	91	92	93	95
AdaMax [5]	74	89	90	92	94
Adadelta	83	104	106	108	112
Adagrad	82	107	108	109	111

Unit: ms/batch; kRkD: raise *k* times and decrease *k* times on learning rate for each iteration.

Table 5: Executing time of conflicting situations on the learning rate.

Methods	Decay	1R1D	2R2D	3R3D	5R5D
RMSprop	100	108	109	111	113
Adam	109	122	124	126	129
NAdam	124	142	145	146	149
AdaMax	115	139	141	143	147
Adadelta	129	162	165	168	176
Adagrad	128	167	170	171	173

Unit: s/epoch; kRkD: raise *k* times and decrease *k* times on learning rate for each iteration.

Table 6: The performance of the two timepieces training.

	ResNet-20 on Cafri-10		LSTM on IMDB	
	Val-loss	Val-acc	Val-loss	Val-acc
SGD + SGD	0.9409	0.7419	0.6890	0.5831
SGD + RMSprop	**0.8670**	**0.7706**	**0.4336**	**0.8367**
SGD + Adam	0.8751	0.7641	0.9210	0.8100

Table 7: The performance of the three timepieces training.

	ResNet-20 on Cafri-10		LSTM on IMDB	
	Val-loss	Val-acc	Val-loss	Val-acc
SGD + SGD + SGD	0.8673	0.7599	0.6341	0.6461
SGD + RMSprop + SGD	**0.5108**	**0.8822**	**0.4824**	**0.8330**
SGD + Adam + SGD	0.5994	0.8524	0.9417	0.8112

6.3. Results of Combination Strategies. Two typical learning rate adjustment methods called decay and cosine decay are used to make a study. On the ResNet20 network, the comparison of the training performance of single and combination strategies, respectively, under 100 epochs is shown in Figure 6.

Now, we compare the performance of single decay and a combination of cosine decay and single decay. Based on the results in Figure 6, accuracy and loss curves during the training process of 100 epochs show that the big benefit of the combination strategy can bring in training.

To measure the performance on combination strategies, taking the cosine decay with decay of $1e-6$ as the learning rate part, meanwhile, SGD with Momentum, Adam, and RMSprop are three typical gradient estimation optimization methods. Figures 7 and 8 are the comparisons of a single category method and combined categories methods, and it can be vividly detected that the performance of a combined one is greater than noncombined.

6.4. Multistage Strategy Combining Multiple Gradient Descent Optimization Methods. The researchers evaluate this strategy with extensive experiments, and the outcomes are shown in Tables 8–15. The whole training time of 20 epochs is divided into two timepieces equally and follows the

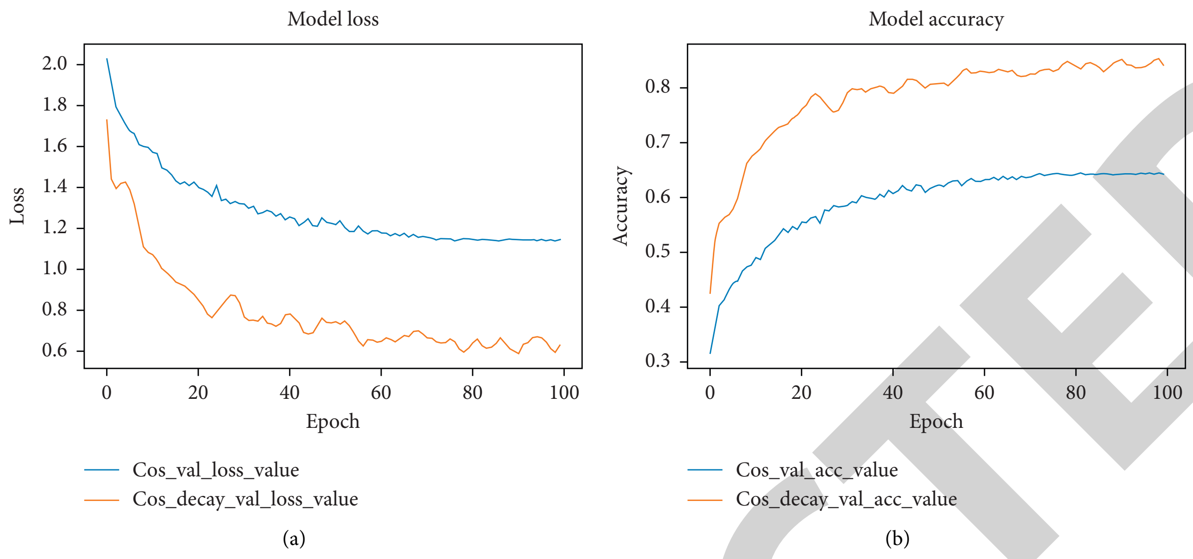


Figure 6: Performance of decay and combination of cosine decay and decay during 100 epochs of training on MNIST. (a, b) The accuracy and loss of the result (init-lr = 0.01).

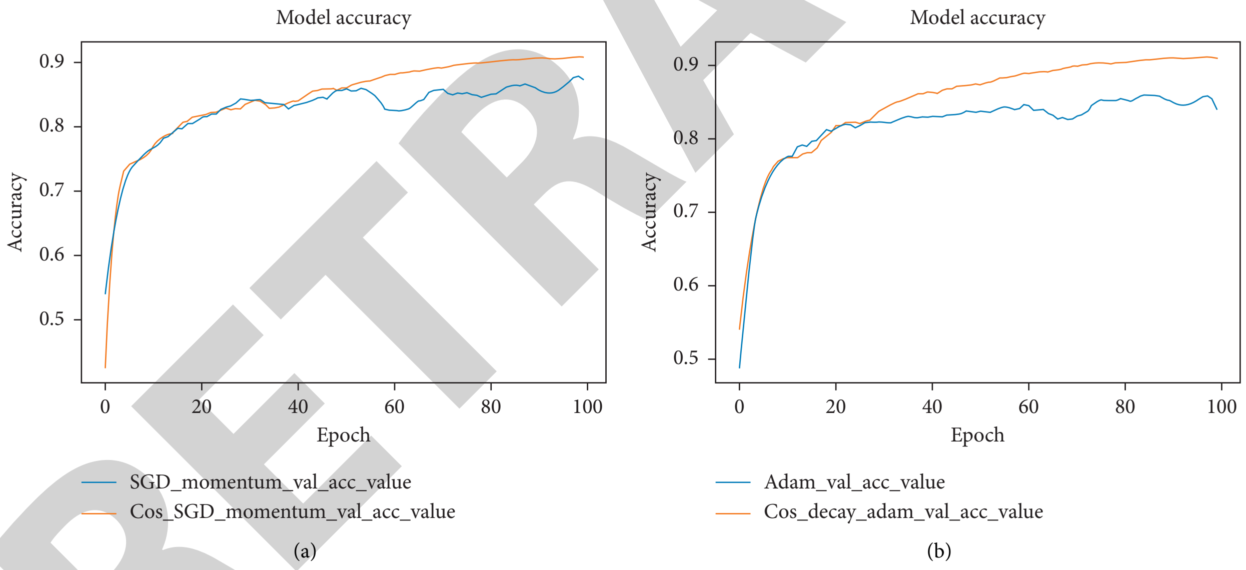
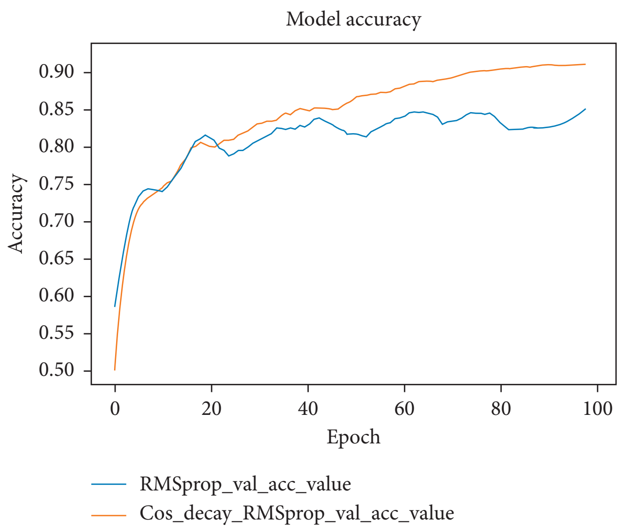
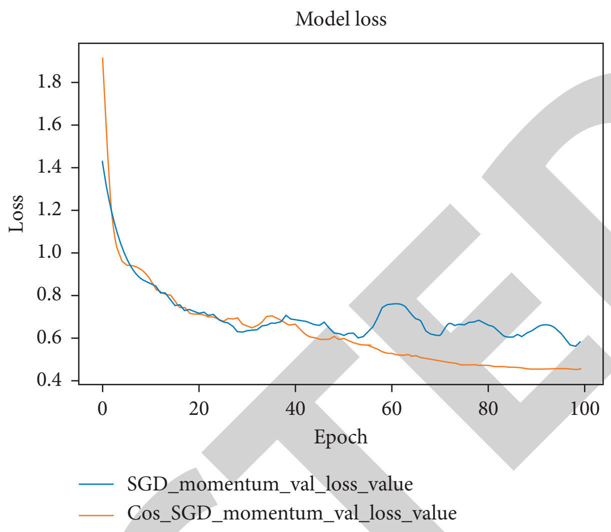


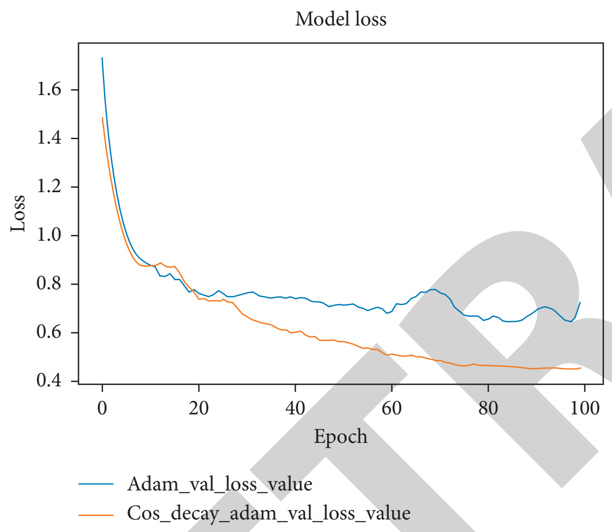
Figure 7: Continued.



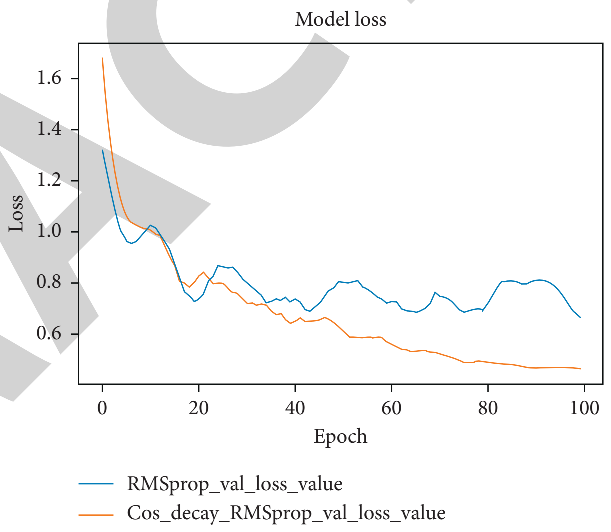
(c)



(d)



(e)



(f)

Figure 7: Combined strategy versus noncombined strategy (1). A combined strategy of SGD with Momentum, Adam, and RMSprop with cosine decay versus SGD with Momentum, Adam, and RMSprop. (a, d) The comparison results of SGD with Momentum and cosine decay, (b, e) the comparison results of Adam and cosine decay, and (c, f) the comparison results of RMSprop and cosine decay.

multistage strategy to take various gradient descent optimization methods on these two timepieces. We take the same method on both timepieces to simulate non-one that takes one method for the whole training.

The conclusions can be drawn from the training performance that, (1) compared with multistage strategies, no single method alone is able to reach the best performance. From Tables 8 to 15, we can see that the best result follows the multistage strategy in all experiments. (2) Not all the combined strategies of multiple stages methods will work in all circumstances. For instance, in Table 15, the performance of the "(Adam + d) + Adam" method is worse than the "(Adam + d) + (Adam + d)" with ResNet and LSTM. Therefore, it is crucial for users to realize this point and resort to other effective multistage methods (3) For the results of ResNet-20 on Cafri-10 and LSTM on IMDB, there is no method which can achieve the highest accuracy on these two cases at the same time. Hence, taking advantage of the flexible feature of the proposed strategy is a wise choice.

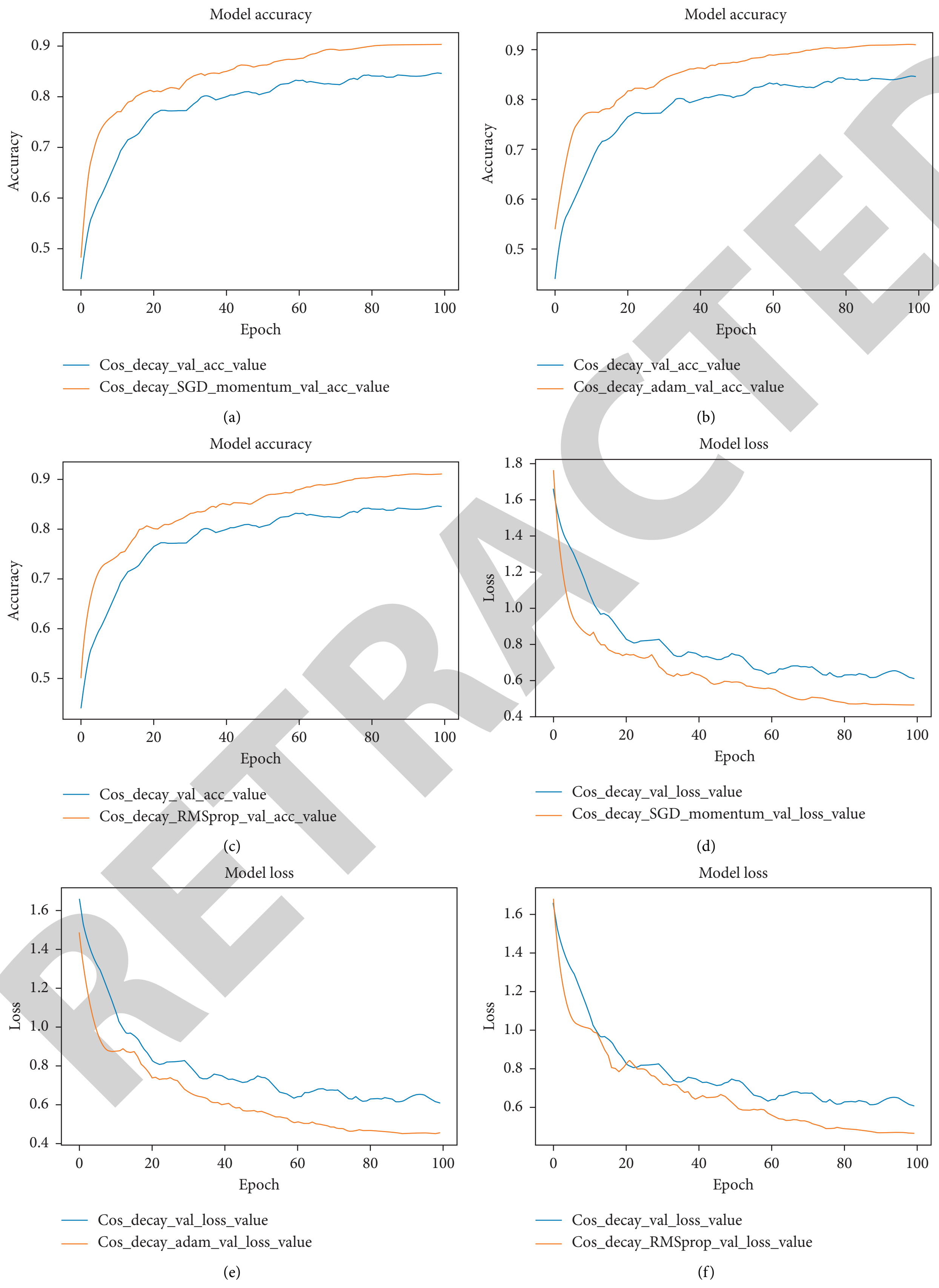


Figure 8: Combined strategy versus noncombined strategy (2). A combined strategy of SGD with Momentum, Adam, and RMSprop with cosine decay versus cosine decay. (a, d) The comparison results of SGD with Momentum and cosine decay, (b, e) the comparison results of Adam and cosine decay, and (c, f) the comparison results of RMSprop and cosine decay.

Table 8: Performance of the proposed method, SGD.

	ResNet-20 on Cafri-10		LSTM on IMDB	
	Val-loss	Val-acc	Val-loss	Val-acc
SGD + SGD	1.0178	0.6948	0.6919	0.5570
SGD + (SGD + M)	1.0763	0.7134	0.4408	0.7971
SGD + (SGD + d)	0.9607	0.7168	0.6890	0.5777
SGD + (SGD + M + d)	0.9040	0.7557	0.4353	0.7982
SGD + RMSprop	0.9408	0.7419	**0.4287**	**0.8367**
SGD + (RMSprop + d)	1.0131	0.7298	0.4342	0.8237
SGD + Adam	**0.8751**	**0.7641**	0.9210	0.8100
SGD + (Adam + d)	1.0692	0.7274	0.8172	0.8130

m: Momentum, D: decay by $1e-6$ every iteration, and "()": take methods at the same timepiece. The bold values represent the best results.

Table 9: Performance of the proposed method, RMSprop.

	ResNet-20 on Cafri-10		LSTM on IMDB	
	Val-loss	Val-acc	Val-loss	Val-acc
RMSprop + SGD	0.5968	0.8513	0.4859	**0.8358**
RMSprop + (SGD + d)	**0.5889**	**0.8529**	**0.4676**	0.8344
RMSprop + (SGD + M)	0.7347	0.8057	0.5672	0.8311
RMSprop + (SGD + M + d)	0.8832	0.7754	0.5583	0.8309
RMSprop + RMSprop	0.8139	0.7978	0.8128	0.8143
RMSprop + (RMSprop + d)	0.7960	0.7927	0.7715	0.8126
RMSprop + Adam	0.9790	0.7513	0.9848	0.8129
RMSprop + (Adam + d)	0.6724	0.8258	0.9556	0.8126

The bold values represent the best results.

Table 10: Performance of the proposed method, Adam.

	ResNet-20 on Cafri-10		LSTM on IMDB	
	Val-loss	Val-acc	Val-loss	Val-acc
Adam + SGD	**0.6088**	**0.8494**	0.9167	0.8135
Adam + (SGD + M)	0.6582	0.8335	1.0421	**0.8156**
Adam + (SGD + d)	0.6108	0.8451	**0.9032**	0.8140
Adam + (SGD + M + d)	0.7453	0.8093	1.1045	0.8150
Adam + RMSprop	0.6929	0.8304	1.1457	0.8089
Adam + (RMSprop + d)	0.8948	0.7816	1.1166	0.8038
Adam + Adam	0.8138	0.7999	1.2044	0.8060
Adam + (Adam + d)	1.1411	0.7164	1.3086	0.8089

The bold values represent the best results.

Table 11: Performance of the proposed method, SGD with Momentum.

	ResNet-20 on Cafri-10		LSTM on IMDB	
	Val-loss	Val-acc	Val-loss	Val-acc
(SGD + M) + SGD	**0.6313**	**0.8359**	**0.3745**	0.8170
(SGD + M) + (SGD + M)	0.6970	0.8191	0.4320	0.8130
(SGD + M) + (SGD + d)	0.6468	0.8321	0.3924	0.8225
(SGD + M) + (SGD + M + d)	0.7518	0.7969	0.3808	0.8318
(SGD + M) + RMSprop	0.8214	0.7911	0.4959	0.8300
(SGD + M) + (RMSprop + d)	1.0059	0.7338	0.4626	**0.8325**
(SGD + M) + Adam	0.9730	0.7499	0.8488	0.8143
(SGD + M) + (Adam + d)	0.8190	0.7899	0.9352	0.8083

The bold values represent the best results.

Table 12: Performance of the proposed method, SGD with decay.

	ResNet-20 on Cafri-10		LSTM on IMDB	
	Val-loss	Val-acc	Val-loss	Val-acc
(SGD + d) + SGD	1.0363	0.6929	0.6888	0.5778
(SGD + d) + (SGD + M)	**0.7713**	**0.7875**	0.4271	0.8044
(SGD + d) + (SGD + d)	1.1088	0.6678	0.6888	0.5871
(SGD + d) + (SGD + M + d)	0.8374	0.7716	**0.3906**	0.8250
(SGD + d) + RMSprop	1.2161	0.7117	0.4230	**0.8371**
(SGD + d) + (RMSprop + d)	1.2169	0.6751	0.4650	0.8346
(SGD + d) + Adam	0.8920	0.7580	0.8291	0.7968
(SGD + d) + (Adam + d)	1.1164	0.6890	0.7607	0.8131

The bold values represent the best results.

Table 13: Performance of the proposed method, SGD with Momentum and decay.

	ResNet-20 on Cafri-10		LSTM on IMDB	
	Val-loss	Val-acc	Val-loss	Val-acc
(SGD + M + d) + SGD	**0.6324**	**0.8340**	**0.3736**	0.8366
(SGD + M + d) + (SGD + M)	0.8533	0.7737	0.3770	**0.8369**
(SGD + M + d) + (SGD + d)	0.6485	0.8278	0.3970	0.8234
(SGD + M + d) + (SGD + M + d)	0.8289	0.7868	0.4274	0.8036
(SGD + M + d) + RMSprop	0.9795	0.7593	0.4949	0.8245
(SGD + M + d) + (RMSprop + d)	0.8371	0.7884	0.4587	0.8279
(SGD + M + d) + Adam	1.2722	0.7058	0.8569	0.8091
(SGD + M + d) + (Adam + d)	0.7481	0.8112	0.9056	0.8070

The bold values represent the best results.

Table 14: Performance of the proposed method, RMSprop with decay.

	ResNet-20 on Cafri-10		LSTM on IMDB	
	Val-loss	Val-acc	Val-loss	Val-acc
(RMSprop + d) + SGD	0.6149	0.8464	0.4872	**0.8338**
(RMSprop + d) + (SGD + d)	0.9866	0.7530	0.5555	0.8289
(RMSprop + d) + (SGD + M)	**0.6012**	**0.8493**	**0.4848**	0.8341
(RMSprop + d) + (SGD + M + d)	0.7868	0.7915	0.5462	0.8257
(RMSprop + d) + RMSprop	0.7120	0.8212	0.7487	0.8161
(RMSprop + d) + (RMSprop + d)	0.9602	0.7771	0.8846	0.7980
(RMSprop + d) + Adam	0.8002	0.7920	0.9164	0.8114
(RMSprop + d) + (Adam + d)	0.8254	0.8000	1.0931	0.8146

The bold values represent the best results.

Table 15: Performance of the proposed method, Adam with decay.

	ResNet-20 on Cafri-10		LSTM on IMDB	
	Val-loss	Val-acc	Val-loss	Val-acc
(Adam + d) + SGD	0.5980	0.8513	0.9323	0.8143
(Adam + d) + (SGD + d)	**0.5959**	**0.8528**	**0.9291**	0.8150
(Adam + d) + (SGD + M)	0.6745	0.8217	1.0951	**0.8172**
(Adam + d) + (SGD + M + d)	0.6960	0.8232	1.1128	0.8137
(Adam + d) + RMSprop	0.7657	0.8078	1.3052	0.7917
(Adam + d) + (RMSprop + d)	0.7760	0.8100	1.0837	0.8137
(Adam + d) + Adam	1.0136	0.7399	1.1720	0.8048
(Adam + d) + (Adam + d)	0.9641	0.7509	1.2429	0.8060

The bold values represent the best results.

7. Conclusion

In this paper, we investigated an optimization method combination framework and a multistage combination strategy for gradient descent optimization methods to improve the training of DNNs. Our combination framework has improvements compared to noncombined ones for the 100 epochs training on MNIST. Meanwhile, we conducted extensive experiments of ResNet-20 on Cafri-10 and LSTM on IMDB for 20 epochs to evaluate the proposed multistage strategy which is based on the proposed method combination framework. The experiment results demonstrated that multistage applications of optimization methods have better performance than one-stage ones. In future, we will extend the proposed strategy to more classic CNNs training.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research was supported by the Tianjin Municipal Science and Technology Bureau under Grant no. 18JCZDJC32100. The author Chuanlei Zhang received this grant, and the URL of the sponsor's website is http://kxjs.tj.gov.cn/. This research was also funded by the National Natural Science Foundation of China under Grant nos. 51874300 and U1510115. The author Wei Chen received these grants, and the URL of the sponsor's website is http://www.nsfc.gov.cn/. This research was also funded by the Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, under Grant no. 20190902. The author Wei Chen received the grant, and the URL of the sponsor's website is http://www.sim.ac.cn/.

References

[1] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proceedings of the Advances in Neural Information Processing Systems*, vol. 27, pp. 3104–3112, Montreal, Canada, December 2014.

[2] K. He, X. Zhang, S. Ren et al., "Deep residual learning for image recognition,"pp. 770–778https://ieeexplore.ieee.org/xpl/conhome/7776647/proceeding, Las Vegas, NV, USA, June 2016.

[3] D. Silver, T. Hubert, J. Schrittwieser et al., "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[4] G. E. Hinton, "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent," Available at: https://www.cs.toronto.edu/%7Etijmen/csc321/slides/lecture_slides_lec6.pdf, 2012.

[5] D. P. Kingma and J. L. Ba, "Adam: a method for stochastic optimization," in *Proceedings of ICLR 2015: International Conference on Learning Representations 2015*, San Diego, CA, USA, May 2015.

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

[7] Y. Nesterov, "Gradient methods for minimizing composite functions," *Mathematical Programming*, vol. 140, no. 1, pp. 125–161, 2013.

[8] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proceedings of International Conference on Learning Representations 2018*, Vancouver, Canada, April 2018.

[9] S. Ruder, "An overview of gradient descent optimization algorithms," 2016, https://arxiv.org/abs/1609.04747.

[10] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science*, G. Montavon, G. B. Orr, and K. R. Müller, Eds., vol. 7700, Springer, Berlin, Germany.

[11] H. Robbins and S. Monro, "A stochastic approximation method," *The Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400–407, 1951.

[12] C. Darken, J. Chang, and J. Moody, "Learning rate schedules for faster stochastic gradient search," *Neural networks for signal processing*, vol. 2, pp. 3–12, 1992.

[13] P. Goyal, P. Dollar, R. Girshick et al., "Accurate, large Minibatch SGD: training imagenet in 1 hour," 2017, https://arxiv.org/abs/1706.02677.

[14] L. N. Smith, "Cyclical learning rates for training neural networks," in *Proceedings of 2017 IEEE winter conference on applications of computer vision*, pp. 464–472, WACV), Santa Rosa, CA, USA, March 2017.

[15] I. Loshchilov and F. Hutter, "SGDR: stochastic gradient descent with warm restarts," in *Proceedings of ICLR 2016: International Conference on Learning Representations 2016*, San Juan, Puerto Rico, May 2016.

[16] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. 61, pp. 2121–2159, 2011.

[17] M. D. Zeiler, "Adadelta: an adaptive learning rate method," 2012, https://arxiv.org/abs/1212.5701.

[18] X. Zeng, W. Ouyang, and X. Wang, "Multi-stage contextual deep learning for pedestrian detection," in *Proceeding of the IEEE International Conference on Computer Vision*, pp. 121–128, Sydney, Australia, December 2013.

[19] P. Sermanet, K. Kavukcuoglu, S. Chintala et al., "Pedestrian detection with unsupervised multi-stage feature learning," in *Proceeding of the IEEE Conference. on Computer Vision and Pattern Recognition*, pp. 3626–3633, Portland, OR, USA, June 2013.

[20] W. Ouyang, P. Luo, X. Zeng et al., "DeepID-Net: multi-stage and deformable deep convolutional neural networks for object detection," in *Proceedings of 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2403–2412, Boston, MA, USA, June 2015.

[21] Z. Yan, Y. Zhan, Z. Peng et al., "Bodypart recognition using multi-stage deep learning," *Lecture Notes in Computer Science*, in *International Conference on information processing in medical imaging*, pp. 449–461, Isle of Skye, UK, June 2015.

[22] X. Yuan, "Phd forum: deep learning-based real-time malware detection with multi-stage analysis," in *Proceeding of the IEEE*

Conference on Smart Computing (SMARTCOMP), pp. 1-2, Hong Kong, China, May 2017.

[23] C. Szegedy, V. Vanhoucke, S. Ioffe et al., "Rethinking the inception architecture for computer vision," in *Proceeding of the IEEE Conference. on Computer Vision and Pattern Recognition*, pp. 2818–2826, Las Vegas, NV, USA, June 2016.

[24] A. C. Wilson, R. Roelofs, M. Stern et al., "The marginal value of adaptive gradient methods in machine learning," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 4151–4161, long beach CA, USA, December 2017.

[25] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[26] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceeding of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[27] A. Krizhevsky, "Learning multiple layers of features from tiny images," Available at:https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf, 2009.

[28] A. Maas, R. Daly, P. Pham et al., "Learning word vectors for sentiment analysis," in *Proceeding. of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, Portland, OR, USA, June 2011.

[29] T. Dozat, "Incorporating nesterov momentum into Adam," in *Workshop on International Conference on Learning Representations*, San Juan, Puerto Rico, May 2016.